ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Кваліфікаційна наукова
праця на правах рукопису

БАБАКОВ РОМАН МАРКОВИЧ

УДК 004.2

ДИСЕРТАЦІЯ

СТРУКТУРИ І МЕТОДИ СИНТЕЗУ МІКРОПРОГРАМНИХ АВТОМАТІВ З ОПЕРАЦІЙНИМ ПЕРЕТВОРЕННЯМ КОДІВ СТАНІВ

Специальність **05.13.05 – комп'ютерні системи та компоненти**
технічні науки

Подається на здобуття наукового ступеня доктора технічних наук

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.
_____(підпис)_____ Р. М. Бабаков

Харків – 2020

АНОТАЦІЯ

Бабаков Р.М. Структури і методи синтезу мікропрограмних автоматів з операційним перетворенням кодів станів. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора технічних наук за спеціальністю «**05.13.05 – комп'ютерні системи та компоненти**». – Донецький національний університет імені Василя Стуса, м. Вінниця. – Харківський національний університет радіоелектроніки Міністерства освіти і науки України, Харків, 2020.

У дисертаційній роботі вирішено актуальну наукову проблему розробки, обґрунтування і дослідження теоретичних основ, структур, моделей і методів, спрямованих на зменшення апаратурних витрат у схемі мікропрограмного автомата за рахунок адаптації схеми автомата до характеристик імплементованого алгоритму керування.

Об'єктом дослідження є процес оптимізації схем мікропрограмних автоматів. Предметом дослідження є методологічні основи, структури, моделі й методи синтезу мікропрограмних автоматів, спрямовані на зменшення апаратурних витрат у схемі автомата.

Дисертаційне дослідження базується на системному аналізі результатів сучасних теоретичних і прикладних розробок вітчизняних і зарубіжних учених в галузі цифрових автоматів. Для вирішення поставлених задач використано: теорія кінцевих автоматів, теорія універсальних алгебр, теорія графів, методи теорії множин, лінійної алгебри та булевої алгебри; комп'ютерне моделювання для аналізу і перевірки вірогідності отриманих теоретичних положень.

В дисертаційній роботі виконано аналіз сучасних теоретико-методичних та практичних концепцій синтезу і оптимізації цифрових пристроїв керування, визначені основні класи пристроїв керування та операційних автоматів, сформульовані їх переваги та недоліки. Визначені основні сучасні напрями оптимізації цифрових пристроїв керування та запропоновано новий підхід до

зменшення апаратурних витрат в логічній схемі мікропрограмного автомата завдяки принципу перетворення кодів станів за допомогою кінцевої множини арифметико-логічних операцій. Пропонується комплекс взаємопов'язаних завдань, вирішення яких полягає у розробці структур і методів синтези мікропрограмних автоматів з операційним перетворення кодів станів.

Запропоновано:

– принцип перетворення кодів станів МПА за допомогою кінцевої множини арифметико-логічних операцій (принцип операційного перетворення кодів станів), застосування якого приводить до представлення схеми формування переходів автомата у вигляді операційного автомата, де кожна операція використовується для реалізації окремої підмножини мікропрограмних переходів;

– формалізація принципу операційного перетворення кодів станів на основі математичного апарата універсальних алгебр, відповідно до якої функція переходів мікропрограмного автомата представляється кінцевою множиною часткових функцій переходів, із кожною з яких ототожнюється власний закон перетворення певної підмножини кодів станів автомата;

– структура операційного автомата переходів, призначеного для перетворення кодів станів мікропрограмного автомата, у якій множина реалізованих арифметико-логічних операцій відповідає множині часткових функцій переходів;

– структура мікропрограмного автомата з операційним автоматом переходів, що допускає відповідність арифметико-логічних операцій окремим мікропрограмним переходам;

– структура мікропрограмного автомата з операційним автоматом переходів, що припускає відповідність арифметико-логічних операцій станам автомата;

– математична модель мікропрограмного автомата з операційним перетворенням кодів станів, що представляє собою систему ізоморфізмів алгебр, у якій кожний ізоморфізм виражає єдиний спосіб інтерпретації та перетворення кодів станів автомата в рамках певної підмножини мікропрограмних переходів.

Сформульовано наукові задачі структурного і алгебраїчного синтезу мікропрограмного автомата з операційним перетворенням кодів станів, для вирішення яких розроблена відповідна методологічна база:

– метод структурного представлення процесу синтезу мікропрограмного автомата з операційним автоматом переходів, що дозволяє класифікувати методи синтезу мікропрограмних автоматів з операційним перетворенням кодів станів;

– метод використання транзитних станів, що дозволяє уникнути застосування додаткових операцій переходів за рахунок додавання проміжних станів, зменшивши, таким чином, апаратурні витрати в схемі операційного автомата переходів та схемі формування кодів операцій переходів;

– метод урахування імовірностей істинності логічних умов, використання якого дозволяє оцінити вплив додавання транзитних станів на середню кількість мікрокоманд, що виконуються за один цикл мікропрограми;

– метод збільшення розрядності кодів станів автомата, що передбачає використання більшої за мінімально достатню кількості двійкових розрядів і дозволяє збільшити кількість способів кодування станів та припустиму кількість транзитних станів при виконанні алгебраїчного синтезу мікропрограмного автомата з операційним автоматом переходів.

Отримали подальший розвиток методи заміни вхідних змінних та зменшення кількості істотних вхідних змінних внаслідок їх застосування та адаптації до мікропрограмного автомата з операційним автоматом переходів. В результаті отримані нові модифіковані структури мікропрограмних автоматів з операційним автоматом переходів, які характеризуються додатковим зменшенням апаратурних витрат у порівнянні зі структурами, що не використовують дані методи.

В роботі запропоновано метод алгебраїчного синтезу мікропрограмного автомата з операційним автоматом переходів, який передбачає пошук формальних розв’язків задачі алгебраїчного синтезу шляхом часткового перебору варіантів. Метод припускає збільшення кількості станів автомата за рахунок транзитних станів та враховує імовірності істинності логічних умов. Алгоритмічна

спрямованість методу дозволяє його використання в системах автоматизованого проектування пристроїв керування цифрових систем.

Проведені експериментальні дослідження ефективності запропонованих структур мікропрограмних автоматів з операційним автоматом переходів, які дозволили визначити їх переваги за критерієм апаратурних витрат у порівнянні з раніше відомими структурами. За результатами проведеного комп'ютерного моделювання із використанням мови опису апаратури VHDL отримані аналітичні залежності апаратурних витрат в логічних схемах досліджуваних структур мікропрограмних автоматів від параметрів автомата та умов проектування.

Удосконалено метод дослідження апаратурних витрат в логічній схемі мікропрограмного автомата, який полягає в тому, що структурні блоки автомату та їх параметри ототожнюються зі стандартними функціональними блоками цифрових систем, а результати експериментальних досліджень, отримані для стандартних функціональних блоків, розповсюджуються на елементи досліджених структур мікропрограмних автоматів.

Практичне значення отриманих результатів полягає у тому, що на основі отриманих та викладених в дисертації теоретичних наукових положень, висновків, пропозицій та рекомендацій розроблені працездатні практичні високоефективні пристрої керування, які дозволяють знизити загальну вартість цифрових систем та зберегти високу швидкодію, властиву пристроям керування на базі мікропрограмних автоматів, за рахунок адаптації схеми пристрою до параметрів імплементованих алгоритмів керування. Практичні результати, що отримано, підтверджені актами впровадження та доводять коректність теоретичних положень дисертаційної роботи, високу ефективність розроблених структур, моделей, методів та методології.

Дисертаційна робота виконана відповідно до планів науково-дослідних робіт Донецького національного університету імені Василя Стуса в рамках таких держбюджетних тем: НДР «Дослідження ефективності мікропрограмного автомата з операційним автоматом переходів» (номер ДР 0117U004097); НДР

«Методологічні аспекти синтезу мікропрограмного автомата з операційним автоматом переходів» (номер ДР 0117U004098).

Результати дисертаційної роботи впроваджені: на підприємстві ТОВ «С-інжинірінг»; в науково-технічному спеціальному конструкторському бюро «Полісвіт» державного науково-виробничого підприємства «Об'єднання "Коммунар"»; в освітньому європейському проекті ERASMUS+ «ALLIOT» – Internet of Things: Emerging Curriculum for Industry and Human Applications; у навчальному процесі кафедри комп'ютерних інтелектуальних систем та мереж Одеського національного політехнічного університету при викладанні дисциплін «Технології проектування комп'ютерних систем» та «Проектування і діагностика систем критичного застосування».

Матеріали дисертації повною мірою викладені у 30 публікаціях, з них 1 – в одноосібній монографії (12 авт. арк.), 22 у фахових періодичних виданнях України з технічних наук, з них 11 статей опубліковані одноосібно, 1 стаття англійською мовою, 1 стаття в електронному виданні; 22 статті включено у міжнародні наукометричні бази, 7 з яких включено в бази Scopus та WoS; 7 тез доповідей у матеріалах міжнародних наукових конференцій (4 англійською мовою, включені до міжнародної наукометричної бази Scopus).

Ключові слова: мікропрограмний автомат, операційний автомат, мінімізація апаратурних витрат, структурний синтез, операційне перетворення кодів станів.

СПИСОК ОПУБЛІКОВАНИХ РОБІТ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

в яких опубліковані основні наукові результати дисертації:

1. Бабаков Р. М. Операционное преобразование кодов состояний в микропрограммном автомате: монография. Винница: «ТВОРИ», 2019. 208 с.

2. Бабаков Р. М. Алгебраический синтез микропрограммного автомата с операционным автоматом переходов. *Информационные технологии и компьютерная инженерия*. 2017. № 39. Т. 2. С. 35–41. (Журнал індексується міжнародною наукометричною базою Google Scholar).

3. Бабаков Р. М. Исследование аппаратурных затрат в микропрограммном автомате с операционным автоматом переходов. *Радиоэлектроника, информатика, управление*. 2017. № 4. С. 106–115. (Входить до міжнародних наукометричних баз **Web of Science**, DOAJ, BASE, Index Copernicus, Google Scholar).

4. Бабаков Р. М. Математическая модель микропрограммного автомата с операционным автоматом переходов. *Наукові праці Донецького національного технічного університету. Серія: Інформатика, кібернетика та обчислювальна техніка*. Красноармійськ: ДВНЗ «ДонНТУ». 2016. № 1 (22). С. 54–57. (Журнал індексується міжнародною наукометричною базою Google Scholar).

5. Бабаков Р. М. Обобщение математической модели микропрограммного автомата на счетчике. *Радиоэлектроника, информатика, управление*. 2018. № 1. С. 100–109. (Входить до міжнародних наукометричних баз **Web of Science**, BASE, Index Copernicus, Google Scholar).

6. Бабаков Р. М. Промежуточная алгебра переходов в микропрограммном автомате. *Радиоэлектроника, информатика, управление*. 2016. № 1. С. 64–73. (Входить до міжнародних наукометричних баз **Web of Science**, DOAJ, BASE, Index Copernicus, Google Scholar).

7. Бабаков Р. М. Синтез логической схемы микропрограммного автомата с операционным автоматом переходов. *Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: технічні науки*. 2018. № 4. С. 96–99. (Журнал індексується міжнародною наукометричною базою Google Scholar).

8. Бабаков Р. М. Синтез микропрограммного автомата с операционным автоматом переходов методом полного перебора. *Вчені записки Таврійського національного університету імені В.І. Вернадського*. Серія: технічні науки. 2018. № 1. С. 70–74. (Журнал індексується міжнародною наукометричною базою Google Scholar).

9. Бабаков Р. М. Урахування імовірності станів у мікропрограмному автоматі з операційним автоматом переходів. *Наукові праці Вінницького національного технічного університету*. 2017. № 2. URL:

https://praci.vntu.edu.ua/index.php/praci/article/view/505/500. (Входить до міжнародної наукометричної бази РІНЦ).

10. Бабаков Р. М. Формальное решение задачи алгебраического синтеза микропрограммного автомата с операционным автоматом переходов. *Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: технічні науки*. 2018. № 2. С. 103–107. (Журнал індексується міжнародною базою Google Scholar).

11. Babakov R. M. Using of method of replacement of input variables in microprogram finite-state machine with datapath of transitions. *Технологический аудит и резервы производства*. 2017. № 4/2 (36). С. 18–23. (Входить до міжнародних наукометричних баз Index Copernicus, BASE, DOAJ, EBSCO, РІНЦ, Google Scholar).

12. Бабаков Р.М. Методологические аспекты синтеза микропрограммных автоматов с операционным автоматом переходов. *Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: технічні науки*. 2019. № 2. С. 82–86. (Журнал індексується міжнародною базою Google Scholar).

13. Бабаков Р. М., Ярош И. В. Использование транзитных состояний в микропрограммном автомате с операционным автоматом переходов. *Наукові праці Донецького національного технічного університету. Серія: Обчислювальна техніка та автоматизація*. Красноармійськ: ДВНЗ «ДонНТУ». 2016. № 1 (29). С. 56–64. (Журнал індексується міжнародною наукометричною базою Google Scholar).

14. Бабаков Р. М., Ярош И. В. Операционный автомат переходов. *Наукові праці Донецького національного технічного університету. Серія: Обчислювальна техніка та автоматизація*. Красноармійськ: ДВНЗ «ДонНТУ». 2015. № 1 (28). С. 33–40. (Журнал індексується міжнародною наукометричною базою Google Scholar).

15. Бабаков Р. М., Ярош И. В. Формирование кодов операций переходов в микропрограммном автомате с операционным автоматом переходов. *Наукові праці Донецького національного технічного університету. Серія: Інформатика,*

кібернетика та обчислювальна техніка. Красноармійськ: ДВНЗ «ДонНТУ». 2015. Випуск 1 (20). С. 11–16. (Журнал індексується міжнародною наукометричною базою Google Scholar).

16. Баркалов А. А., Бабаков Р. М. Алгебраическая интерпретация микропрограммного автомата с операционным автоматом переходов. *Кибернетика и системный анализ*. 2016. № 2. С. 22–29. (Входить до міжнародних наукометричних баз **Scopus**, INSPEC, EBSCO, Google Scholar).

17. Баркалов А. А., Бабаков Р. М. Модификация микропрограммного автомата с операционным автоматом переходов и заменой входных переменных. *Управляющие системы и машины*. 2017. № 6. С. 35–40. (Входить до міжнародних наукометричних баз РІНЦ, Google Scholar).

18. Баркалов А. А., Бабаков Р. М. Операционная реализация функции выходов микропрограммного автомата. *Управляющие системы и машины*. 2017. № 3. С. 57–62. (Входить до міжнародних наукометричних баз РІНЦ, Google Scholar).

19. Баркалов А. А., Бабаков Р. М. Операционное формирование кодов состояний в микропрограммных автоматах. *Кибернетика и системный анализ*. 2011. № 2. С. 21–26. (Входить до міжнародних наукометричних баз **Scopus**, INSPEC, EBSCO, Google Scholar).

20. Баркалов А. А., Бабаков Р. М. Операционный автомат переходов с дополненным множеством операций переходов. *Наукові праці Донецького національного технічного університету. Серія «Інформатика, кібернетика і обчислювальна техніка»*. Донецьк: ДВНЗ «ДонНТУ». 2011. № 14 (188). С. 80–84. (Журнал індексується міжнародною наукометричною базою Google Scholar).

21. Баркалов А. А., Бабаков Р. М. Определение области эффективного применения микропрограммного автомата с операционным автоматом переходов. *Кибернетика и системный анализ*. 2018. № 3. С. 27–37. (Входить до міжнародних наукометричних баз **Scopus**, INSPEC, EBSCO, Google Scholar).

22. Баркалов А. А., Бабаков Р. М. Структурная классификация методов синтеза микропрограммного автомата с операционным автоматом переходов.

Кибернетика и системный анализ. 2019. № 2. С. 3–9. (Входить до міжнародних наукометричних баз **Scopus**, INSPEC, EBSCO, Google Scholar).

23. Баркалов А. А., Бабаков Р. М. Уменьшение максимального количества существенных входных переменных в микропрограммном автомате с операционным автоматом переходов. *Управляющие системы и машины*. 2018. № 2. С. 42–50. (Входить до міжнародних наукометричних баз РІНЦ, Google Scholar).

які засвідчують апробацію матеріалів дисертації:

24. Бабаков Р. М. Микропрограммный автомат с операционным автоматом переходов. *Тези доповідей Міжнародної наукової конференції «Сучасна інформатика: проблеми, досягнення та перспективи розвитку», присвяченої 60-річчю заснування Інституту кібернетики імені В.М. Глушкова НАН України* (м. Київ, 13–15 грудня 2017 р.) Київ, 2017. С. 4–6.

25. Бабаков Р. М. Синтез микропрограммного автомата с операционным автоматом переходов методом полного перебора. *Матеріали Всеукраїнської науково-практичної конференції з міжнародною участю «Інформатика та системні науки (ІСН-2017)* (м. Полтава, 16–18 березня 2017 р.) Полтава: ПУЕТ, 2017. С. 23–25. (Входить до міжнародної наукометричної бази Google Scholar).

26. Баркалов А. А., Бабаков Р. М. Операционное формирование переходов в управляющих автоматах. *Матеріали доповідей ІІІ Міжнародної науково-практичної конференції молодих учених, аспірантів, студентів «Сучасна інформаційна Україна: інформатика, економіка, філософія»* (м. Донецьк, 14–15 травня 2009 р.) Донецьк, 2009. Т. 1. С. 18–20. (Входить до міжнародної наукометричної бази Google Scholar).

27. Babakov R., Barkalov A., Titarenko L. Research of Efficiency of Microprogram Final-State Machine with Datapath of Transitions. *Proceedings of 14th International Conference «The Experience of Designing and Application of CAD Systems in Microelectronics» (CADSM)* (Україна, Поляна, 21–25 лютого 2017 р). Львів, 2017. С. 203–206. (Входить до міжнародної наукометричної бази **Scopus**).

28. Barkalov A. A., Titarenko L. A., Babakov R. M. Compositional Microprogram Control Unit with Operational Automaton of Transitions. *Proceedings of IEEE East-West Design & Test Symposium (EWDTS'2012)* (Kharkov, Ukraine, September 14–17). Kharkov, 2012. P. 151–154. (Входить до міжнародної наукометричної бази **Scopus**).

29. Barkalov A. A., Titarenko L. A., Babakov R. M. Compositional Microprogram Control Unit with Operational Automaton of Transitions. *12th IFAC Conference on Programmable Devices and Embedded Systems* (Velke Karlovice, Czech Republic, September 25–27), Velke Karlovice, 2013. P. 239–244. (Входить до міжнародних наукометричних баз **Scopus**, Google Scholar).

30. Barkalov A., Babakov R. Structural representation of synthesis methods of finite state machine with datapath of transitions. *Conference Proceedings of 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT'2018)* (м. Київ, 24–27 травня 2018 р). Київ, 2018. С. 242–246. (Входить до міжнародної наукометричної бази **Scopus**).

ABSTRACT

Babakov R.M. Structures and methods of synthesis of finite state machines with operational transformation of state codes. – Qualified scientific work as a manuscript.

Thesis for obtaining of doctoral degree in technical sciences in the specialty 05.13.05 "Computer systems and components". – Vasyl' Stus Donetsk National University, Vinnytsia. – Kharkiv National University of Radio Electronics, Ministry of Education and Science of Ukraine, Kharkiv, 2020.

In the thesis, the actual scientific problem of development, foundation and research of theoretical bases, structures, models and methods aimed at reducing of hardware expenses in the logical circuit of the finite state machine, is solved by adaptation of the FSM circuit to the characteristics of the implemented control algorithm.

The object of the research encompasses process of optimization of finite state machines cicuits. The subject of the research is the methodological foundations, structures, models and methods of synthesis of finite state machines, aimed at reducing the hardware expenses in the circuit of the machine.

The research is based on a systematic analysis of the results of modern theoretical and applied developments made by domestic and foreign scientists in the digital control units sphere. Solution of the set tasks involved application of: finite state machine theory, universal algebra theory, graph theory, methods of set theory, linear algebra and Boolean algebra; computer simulation for the analysis and validation of theoretical assumptions.

In the thesis, the analysis of modern theoretical, methodological and practical concepts of synthesis and optimization of digital control units was carried out, the basic classes of control units and datapaths are defined and their advantages and disadvantages are formulated. The main modern ways of digital control units optimization are identified and new approaches to reducing hardware expenses in the logical circuit of finite state machine based on principle of transformation of state codes using the finite set of arithmetical and logical operations, are proposed. The complex of relative problems which solution consists in development of structures and methods of synthesis of finite state machines with operational transformations of the state codes is offered.

The thesis proposes:

– principle of transformation of state codes of finite state machine by means of a finite set of arithmetic-logical operations (principle of operational transformation of state codes), which application leads to the implementation of the circuit of formation of finite state machine transitions in the form of datapath, where each operation is used to realize a separate subset of microprogram transitions;

– formalization of the principle of operational transformation of state codes with using of universal algebras, according to which the function of transitions of a finite state machine is represented by a finite set of partial functions of transitions, each of

which is identical to its own law of transformation of a certain subset of state codes of the finite state machine;

– structure of datapath of transitions intended to transform the state codes of the finite state machine, in which the set of implemented arithmetic-logical operations corresponds to the set of partial functions of transitions;

– structure of the finite state machine with the datapath of transitions that allows conformity of arithmetic-logical operations to separate microprogram transitions;

– structure of the finite state machine with datapath of transitions, which allows the conformity of arithmetic-logical operations to separate states;

– mathematical model of a finite state machine with operational transformation of state codes, which is a system of isomorphisms of algebras, where each isomorphism expresses the only way to interpret and transform the state codes of a finite state machine within a certain subset of microprogram transitions.

The scientific problems of structural and algebraic synthesis of the finite state machine with operational transformation of states codes were formulated, for which the appropriate methodological base was developed:

– method of structural representation of the synthesis process of finite state machine with datapath of transitions, which allows to classify methods of synthesis of finite state machines with operational transformation of state codes;

– method of transit states using, which avoids the use of additional transition operations by adding intermediate states, thus reducing the hardware expenses in the circuit of datapath of transitions and the circuit of formation of codes of transition operations;

– method of accounting of probabilities of the truth of logical conditions, which using allows to estimate the influence of the addition of transit states on the average number of microinstructions executed in one microprogram cycle;

– method of increasing capacity of state codes of the finite state machine, which involves the use of more than minimum sufficient number of bits and allows to increase the number of ways of state encoding and the allowable number of transit states when performing algebraic synthesis of finite state machine with datapath of transitions.

Methods for replacing input variables and reducing the number of significant input variables were further developed due to their use and adaptation to a finite state machine with datapath of transitions. As a result, new modified structures of finite state machine with datapath of transitions were obtained, which are characterized by an additional reduction of hardware expenses compared to structures that do not use these methods.

In the thesis, method of algebraic synthesis of a finite state machine with datapath of transitions was proposed, which involves the search for formal solutions to the problem of algebraic synthesis by performing a partial search algorithm. The method assumes an increase in the number of states of the finite state machine due to transit states and takes into account the probabilities of the truth of the logical conditions. Algorithmic orientation of the method allows its using in computer-aided design of control units of digital systems.

Experimental studies of the effectiveness of the proposed structures of finite state machine with datapath of transitions were conducted, which allowed to determine their advantages by the criterion of hardware costs in comparison with previously known structures. According to the results of computer simulation using the VHDL language, analytical dependences of the hardware expenses in the logical circuits of the studied finite state machine structures on the machine parameters and design conditions were obtained.

The method of investigation of hardware expenses in the logical circuit of finite state machine was improved, which consists in the fact that the structural blocks of the finite state machine and their parameters are identified with the standard functional blocks of digital devices, and the results of experimental research obtained for the standard functional blocks are extended to the elements of the finite state machines.

The practical value of the research findings is that, based on the theoretical findings, conclusions, suggestions and recommendations obtained and presented in the thesis, workable practical high-performance control units have been developed that allow to reduce the overall cost of digital systems and provide high performance inherent in finite state machines, by adapting the device structure to the parameters of

the implemented control algorithms. The obtained practical results are confirmed by the acts of implementation and prove the correctness of the theoretical provisions of the thesis, high efficiency of the developed structures, models, methods and methodology.

The thesis conforms with the research plan on Vasyl' Stus Donetsk National University within the framework of taxplayer-funded research in the field of: academic research work "Investigation of the Efficiency of a Finite State Machine with Datapath of Transitions" (No. research work 0117U004097); academic research work "Methodological Aspects of Synthesis of a Finite State Machine with Datapath of transitions" (No. research work 0117U004098).

The findings of the thesis have been implemented: at the limited liability company "S-engineering"; at the scientific and technical special design bureau "Polisvit" of the state research and production enterprise "Kommunar Association"; in the European educational project ERASMUS+ "ALLIOT" –- Internet of Things: Emerging Curriculum for Industry and Human Applications; in the educational process of the Department of Computer Intelligent Systems and Networks of the Odessa National Polytechnic University in studying the disciplines "Computer Systems Design" and " Design and Diagnostics of Systems of Critical Application".

The thesis materials are fully presented in 30 publications, namely 1 of them is in sole monograph (12 author's sheets), 22 of them in Ukrainian professional periodicals on technical sciences, of which 11 articles are in sole authorship, 1 article is in English, 1 article is in Ukrainian electronic professional periodicals on technical sciences, 22 articles are included in international science and technology bases, 7 of which are included in Scopus and WoS; 7 conference abstracts in international scientific conference materials (4 of them are in English and included in Scopus international scientific and technology base).

Key words: finite state machine, datapath, hardware expenses minimization, structural synthesis, operational transformation of state codes.

LIST OF PUBLICATIONS

The list of publications, which reflect the main scientific results of the thesis:

1. Бабаков Р. М. Операционное преобразование кодов состояний в микропрограммном автомате: монография. Винница: «ТВОРИ», 2019. 208 с.

2. Бабаков Р. М. Алгебраический синтез микропрограммного автомата с операционным автоматом переходов. *Информационные технологии и компьютерная инженерия*. 2017. № 39. Т. 2. С. 35–41. (Журнал індексується міжнародною наукометричною базою Google Scholar).

3. Бабаков Р. М. Исследование аппаратурных затрат в микропрограммном автомате с операционным автоматом переходов. *Радиоэлектроника, информатика, управление*. 2017. № 4. С. 106–115. (Входить до міжнародних наукометричних баз **Web of Science**, DOAJ, BASE, Index Copernicus, Google Scholar).

4. Бабаков Р. М. Математическая модель микропрограммного автомата с операционным автоматом переходов. *Наукові праці Донецького національного технічного університету. Серія: Інформатика, кібернетика та обчислювальна техніка*. Красноармійськ: ДВНЗ «ДонНТУ». 2016. № 1 (22). С. 54–57. (Журнал індексується міжнародною наукометричною базою Google Scholar).

5. Бабаков Р. М. Обобщение математической модели микропрограммного автомата на счетчике. *Радиоэлектроника, информатика, управление*. 2018. № 1. С. 100–109. (Входить до міжнародних наукометричних баз **Web of Science**, BASE, Index Copernicus, Google Scholar).

6. Бабаков Р. М. Промежуточная алгебра переходов в микропрограммном автомате. *Радиоэлектроника, информатика, управление*. 2016. № 1. С. 64–73. (Входить до міжнародних наукометричних баз **Web of Science**, DOAJ, BASE, Index Copernicus, Google Scholar).

7. Бабаков Р. М. Синтез логической схемы микропрограммного автомата с операционным автоматом переходов. *Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: технічні науки*. 2018. № 4. С. 96–99. (Журнал індексується міжнародною наукометричною базою Google Scholar).

8. Бабаков Р. М. Синтез микропрограммного автомата с операционным автоматом переходов методом полного перебора. *Вчені записки Таврійського національного університету імені В.І. Вернадського*. Серія: технічні науки. 2018. № 1. С. 70–74. (Журнал індексується міжнародною наукометричною базою Google Scholar).

9. Бабаков Р. М. Урахування імовірності станів у мікропрограмному автоматі з операційним автоматом переходів. *Наукові праці Вінницького національного технічного університету*. 2017. № 2. URL: https://praci.vntu.edu.ua/index.php/praci/article/view/505/500. (Входить до міжнародної наукометричної бази РІНЦ).

10. Бабаков Р. М. Формальное решение задачи алгебраического синтеза микропрограммного автомата с операционным автоматом переходов. *Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: технічні науки*. 2018. № 2. С. 103–107. (Журнал індексується міжнародною базою Google Scholar).

11. Babakov R. M. Using of method of replacement of input variables in microprogram finite-state machine with datapath of transitions. *Технологический аудит и резервы производства*. 2017. № 4/2 (36). С. 18–23. (Входить до міжнародних наукометричних баз Index Copernicus, BASE, DOAJ, EBSCO, РІНЦ, Google Scholar).

12. Бабаков Р.М. Методологические аспекты синтеза микропрограммных автоматов с операционным автоматом переходов. *Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: технічні науки*. 2019. № 2. С. 82–86. (Журнал індексується міжнародною базою Google Scholar).

13. Бабаков Р. М., Ярош И. В. Использование транзитных состояний в микропрограммном автомате с операционным автоматом переходов. *Наукові праці Донецького національного технічного університету. Серія: Обчислювальна техніка та автоматизація*. Красноармійськ: ДВНЗ «ДонНТУ». 2016. № 1 (29). С. 56–64. (Журнал індексується міжнародною наукометричною базою Google Scholar).

14. Бабаков Р. М., Ярош И. В. Операционный автомат переходов. *Наукові праці Донецького національного технічного університету. Серія: Обчислювальна техніка та автоматизація*. Красноармійськ: ДВНЗ «ДонНТУ». 2015. № 1 (28). С. 33–40. (Журнал індексується міжнародною наукометричною базою Google Scholar).

15. Бабаков Р. М., Ярош И. В. Формирование кодов операций переходов в микропрограммном автомате с операционным автоматом переходов. *Наукові праці Донецького національного технічного університету. Серія: Інформатика, кібернетика та обчислювальна техніка*. Красноармійськ: ДВНЗ «ДонНТУ». 2015. Випуск 1 (20). С. 11–16. (Журнал індексується міжнародною наукометричною базою Google Scholar).

16. Баркалов А. А., Бабаков Р. М. Алгебраическая интерпретация микропрограммного автомата с операционным автоматом переходов. *Кибернетика и системный анализ*. 2016. № 2. С. 22–29. (Входить до міжнародних наукометричних баз **Scopus**, INSPEC, EBSCO, Google Scholar).

17. Баркалов А. А., Бабаков Р. М. Модификация микропрограммного автомата с операционным автоматом переходов и заменой входных переменных. *Управляющие системы и машины*. 2017. № 6. С. 35–40. (Входить до міжнародних наукометричних баз РІНЦ, Google Scholar).

18. Баркалов А. А., Бабаков Р. М. Операционная реализация функции выходов микропрограммного автомата. *Управляющие системы и машины*. 2017. № 3. С. 57–62. (Входить до міжнародних наукометричних баз РІНЦ, Google Scholar).

19. Баркалов А. А., Бабаков Р. М. Операционное формирование кодов состояний в микропрограммных автоматах. *Кибернетика и системный анализ*. 2011. № 2. С. 21–26. (Входить до міжнародних наукометричних баз **Scopus**, INSPEC, EBSCO, Google Scholar).

20. Баркалов А. А., Бабаков Р. М. Операционный автомат переходов с дополненным множеством операций переходов. *Наукові праці Донецького національного технічного університету. Серія «Інформатика, кібернетика і*

обчислювальна техніка». Донецьк: ДВНЗ «ДонНТУ». 2011. № 14 (188). С. 80–84. (Журнал індексується міжнародною наукометричною базою Google Scholar).

21. Баркалов А. А., Бабаков Р. М. Определение области эффективного применения микропрограммного автомата с операционным автоматом переходов. *Кибернетика и системный анализ*. 2018. № 3. С. 27–37. (Входить до міжнародних наукометричних баз **Scopus**, INSPEC, EBSCO, Google Scholar).

22. Баркалов А. А., Бабаков Р. М. Структурная классификация методов синтеза микропрограммного автомата с операционным автоматом переходов. *Кибернетика и системный анализ*. 2019. № 2. С. 3–9. (Входить до міжнародних наукометричних баз **Scopus**, INSPEC, EBSCO, Google Scholar).

23. Баркалов А. А., Бабаков Р. М. Уменьшение максимального количества существенных входных переменных в микропрограммном автомате с операционным автоматом переходов. *Управляющие системы и машины*. 2018. № 2. С. 42–50. (Входить до міжнародних наукометричних баз РІНЦ, Google Scholar).

Results that confirm the approbation of the thesis:

24. Бабаков Р. М. Микропрограммный автомат с операционным автоматом переходов. *Тези доповідей Міжнародної наукової конференції «Сучасна інформатика: проблеми, досягнення та перспективи розвитку», присвяченої 60-річчю заснування Інституту кібернетики імені В.М. Глушкова НАН України* (м. Київ, 13–15 грудня 2017 р.) Київ, 2017. С. 4–6.

25. Бабаков Р. М. Синтез микропрограммного автомата с операционным автоматом переходов методом полного перебора. *Матеріали Всеукраїнської науково-практичної конференції з міжнародною участю «Інформатика та системні науки (ІСН-2017)* (м. Полтава, 16–18 березня 2017 р.) Полтава: ПУЕТ, 2017. С. 23–25. (Входить до міжнародної наукометричної бази Google Scholar).

26. Баркалов А. А., Бабаков Р. М. Операционное формирование переходов в управляющих автоматах. *Матеріали доповідей ІІІ Міжнародної науково-практичної конференції молодих учених, аспірантів, студентів «Сучасна інформаційна Україна: інформатика, економіка, філософія»* (м. Донецьк, 14–15

травня 2009 р.) Донецьк, 2009. Т. 1. С. 18–20. (Входить до міжнародної наукометричної бази Google Scholar).

27. Babakov R., Barkalov A., Titarenko L. Research of Efficiency of Microprogram Final-State Machine with Datapath of Transitions. *Proceedings of 14th International Conference «The Experience of Designing and Application of CAD Systems in Microelectronics» (CADSM)* (Україна, Поляна, 21–25 лютого 2017 р). Львів, 2017. С. 203–206. (Входить до міжнародної наукометричної бази **Scopus**).

28. Barkalov A. A., Titarenko L. A., Babakov R. M. Compositional Microprogram Control Unit with Operational Automaton of Transitions. *Proceedings of IEEE East-West Design & Test Symposium (EWDTS'2012)* (Kharkov, Ukraine, September 14–17). Kharkov, 2012. P. 151–154. (Входить до міжнародної наукометричної бази **Scopus**).

29. Barkalov A. A., Titarenko L. A., Babakov R. M. Compositional Microprogram Control Unit with Operational Automaton of Transitions. *12th IFAC Conference on Programmable Devices and Embedded Systems* (Velke Karlovice, Czech Republic, September 25–27), Velke Karlovice, 2013. P. 239–244. (Входить до міжнародних наукометричних баз **Scopus**, Google Scholar).

30. Barkalov A., Babakov R. Structural representation of synthesis methods of finite state machine with datapath of transitions. *Conference Proceedings of 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT'2018)* (м. Київ, 24–27 травня 2018 р). Київ, 2018. С. 242–246. (Входить до міжнародної наукометричної бази **Scopus**).

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

АЖЛ – Автомат із «жорсткою» логікою

АЛП – Арифметико-логічний пристрій

АПЛ – Автомат з програмувальною логікою

АЧ – Адресна частина

БФ – Булева функція

ГСА – Граф-схема алгоритму

ДНФ – Диз’юнктивна нормальна форма

ЗП – Запам’ятовуючий пристрій

КЛБ – Комбінаційний логічний блок

КЛС – Комбінаційна логічна схема

КС – Комбінаційна схема

ЛУ – Логічна умова

МК – Мікрокоманда

МО – Мікрооперація

МП – Мікропрограма

МПА – Мікропрограмний автомат

МПК – Мікропрограмний пристрій керування

ОА – Операційний автомат

ОАП – Операційний автомат переходів

ОП – Операція переходів

ОТП – Операційна таблиця переходів

ОП – Операційний пристрій

ОЧ – Операційна частина

ПЛІС – Програмувальна логічна інтегральна схема

ПЛМ – Програмувальна логічна матриця

ПСТ – Пряма структурна таблиця

РАМК – Регістр адреси мікрокоманди

РП – Регістр пам’яті

САПР – Система автоматизованого проектування

СТ – Лічильник

СФМО – Схема формування мікрооперацій

СФП – Схема формування переходів

КА – Керуючий автомат

КП – Керуюча пам’ять

ПК – Пристрій керування

ВСТУП

Обґрунтування вибору теми дослідження. Стрімкий розвиток комп'ютерної індустрії визначає необхідність забезпечення обчислювального процесу ефективними апаратними засобами, що поєднують високу продуктивність і прийнятну вартість. Одним з основних напрямків досліджень у даній області є здешевлення цифрових систем, що сприяє розширенню сфери їх застосування. Останнє десятиліття завдання зменшення апаратурних витрат у цифрових системах як і раніше зберігає свою актуальність, у тому числі й у зв'язку з розвитком елементного базису цифрових пристроїв [56, 62, 65, 66, 110, 144, 153, 178, 193, 201, 209, 218, 229, 245, 247, 251, 269, 271, 275, 276].

При проектуванні цифрових систем широко застосовується принцип мікропрограмного керування, сформульований в 1951 році М. Уілксом і деталізований у роботах В. М. Глушкова, Г. Б. Жинтеліса, С. О. Майорова, Б. Н. Малиновського, Г. І. Новікова, О. В. Палагіна та ін. Згідно з даним принципом, одним із центральних вузлів цифрової системи є пристрій керування (ПК). Функцією ПК є координація роботи усіх блоків системи на основі заданого алгоритму керування, а характеристики ПК в значній мірі визначають характеристики цифрової системи в цілому. Сучасні підходи до проектування ПК цифрових систем є розвитком теорії автоматів, основи якої були закладені в роботах відомих учених С. Кліні, Г. Мілі, Е. Мура, Дж. фон Неймана, В. М. Глушкова, М. О. Гаврилова, Д. Хафмена та ін. Розвиток досліджень у цій області привів до появи спеціальних методів структурного синтезу для різних класів ПК, що суттєво розширило можливості теорії автоматів при проектуванні сучасних складних цифрових систем. Тут у першу чергу необхідно відзначити роботи представників наукових шкіл В. М. Глушкова, М. О. Гаврилова, А. Д. Закревського, В. М. Лазарєва, С. І. Баранова, В. А. Склярова, Е. О. Якубайтіса, а також роботи Є. М. Вавілова, В. І. Варшавського, В. А. Горбатова, А. В. Каляєва, А. М. Меліхова, П. П. Пархоменко, Д. О. Поспєлова, І. В. Прангішвілі, З.Л. Рабіновича та інших.

Одним з різновидів ПК є мікропрограмний автомат (МПА, автомат з «жорсткою» логікою). МПА має максимальну швидкодію серед відомих класів ПК за рахунок реалізації багатоспрямованих мікропрограмних переходів за один такт роботи пристрою. Однак імплементація алгоритму роботи МПА у вигляді мережі логічних елементів збільшує вартість схеми автомата. В теперішній час постійно зростаюча складність алгоритмів керування, чітко виражена тенденція мікропрограмної реалізації функцій, що раніше виконувались програмно, прогрес в області елементної бази викликають необхідність модифікації існуючих та розробки нових, більш досконалих методів структурного синтезу МПА, у тому числі орієнтованих на зменшення апаратурних витрат у схемі автомата.

Сучасним елементним базисом для реалізації логічної схеми МПА є програмувальні логічні інтегральні схеми (ПЛІС), до яких належать мікросхеми типу FPGA, CPLD, SoPC та інші. Висока щільність логічних вентилів на кристалі ПЛІС дозволяє реалізувати на базі одного чипа пристрій керування, операційний пристрій, цифро-аналогові та аналого-цифрові перетворювачі, фільтри та інші частини складної цифрової системи. Як і для попередніх базисів реалізації цифрових систем, у випадку ПЛІС виникає проблема оптимізації характеристик окремих блоків. Вирішення цієї проблеми дозволяє збільшити функціональні можливості системи в рамках однієї мікросхеми ПЛІС. Таким чином, проблема розробки ефективних методів синтезу цифрових систем в цілому та МПА зокрема є актуальною.

Науково-методичною основою досліджень, представлених в дисертаційній роботі, є праці вітчизняних та закордонних вчених і фахівців. Відчутний внесок у розв'язок проблеми оптимізації апаратурних витрат у мікропрограмних автоматах внесли такі відомі у нас в країні та за кордоном вчені, як Баранов С. І., Баркалов О. О., Палагін О. В., Харченко В. С., Кривуля Г. Ф., Опанасенко В. М., Бібіло П. М., Соловйов В. В. та ін. Серед вчених далекого зарубіжжя найбільш відомими в досліджуваному напрямку роботами є праці D. Kania, G. De Micheli, M. Adamsky, T. Luba, M. Petrovski, T. Kam, B. Echerman, A. Klimovich, M. J. Avedillo, R. Brayton та ін.

Проведені дослідження та їх аналіз свідчать, що оптимізація апаратурних витрат у логічній схемі ПК в значній мірі пов'язана з адаптацією структури ПК до характеристик алгоритму керування, який імплементується пристроєм керування. Все це актуалізує і породжує спрямованість теоретичних та прикладних досліджень дисертаційної роботи, що присвячена розробці нових структур мікропрограмних автоматів і методів їх синтезу, які сприяють зниженню апаратурних витрат у логічній схемі автомата.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційна робота виконана відповідно до планів науково-дослідних робіт Донецького національного університету імені Василя Стуса в рамках наступних держбюджетних тем

– НДР «Дослідження ефективності мікропрограмного автомата з операційним автоматом переходів» (номер держреєстрації 0117U004097), автор брав участь у виконанні роботи як науковий керівник;

– НДР «Методологічні аспекти синтезу мікропрограмного автомата з операційним автоматом переходів» (номер держреєстрації 0117U004098), автор брав участь у виконанні роботи як науковий керівник.

Мета та завдання дослідження. Метою дисертаційної роботи є розробка, обґрунтування і дослідження теоретичних основ, структур, моделей і методів, спрямованих на зменшення апаратурних витрат в схемі мікропрограмного автомата за рахунок адаптації схеми автомата до характеристик імплементованого алгоритму керування.

Для досягнення мети в дисертаційній роботі вирішувались наступні завдання:

– аналіз існуючих методів синтезу і оптимізації цифрових автоматів;

– розробка загальної концепції представлення функції переходів автомата у вигляді множини часткових функцій;

– формалізоване узагальнення структурних особливостей відомої структури мікропрограмного автомата на лічильнику та формулювання нового принципу

перетворення кодів станів автомата за допомогою кінцевої множини арифметико-логічних операцій;

– розробка структури і математичної моделі мікропрограмного автомата, заснованих на запропонованому принципі перетворення кодів станів за допомогою кінцевої множини операцій і орієнтованих на зменшення апаратурних витрат у логічній схемі автомата;

– модифікація розробленої структури мікропрограмного автомата із застосуванням відомих методів оптимізації апаратурних витрат;

– розробка методології синтезу розроблених структур мікропрограмних автоматів;

– визначення області ефективного застосування розроблених структур мікропрограмних автоматів з оптимізованими витратами апаратури.

Об'єкт дослідження – процес оптимізації схем мікропрограмних автоматів.

Предметом дослідження є методологічні основи, структури, моделі й методи синтезу мікропрограмних автоматів, спрямовані на зменшення апаратурних витрат у схемі автомата.

Методи досліджень. Для вирішення поставлених у дисертації завдань використані: теорія кінцевих автоматів, теорія універсальних алгебр, теорія графів, методи теорії множин, лінійної алгебри та булевої алгебри; комп'ютерне моделювання для аналізу і перевірки вірогідності отриманих теоретичних положень.

Наукова новизна отриманих результатів. У дисертаційній роботі вирішено актуальну наукову проблему розробки, обґрунтування і дослідження теоретичних основ, структур, моделей і методів, спрямованих на зменшення апаратурних витрат в схемі мікропрограмного автомата за рахунок адаптації схеми автомата до характеристик імплементованого алгоритму керування. Основні результати, що визначають наукову новизну дисертаційної роботи, полягають у наступному:

Вперше запропонований принцип реалізації функції переходів мікропрограмного автомата на основі перетворення кодів станів за допомогою

кінцевої множини арифметико-логічних операцій, що дозволяє представити схему формування переходів мікропрограмного автомата у формі операційного автомата і сприяє зниженню апаратурних витрат за рахунок багаторазового використання функціональних блоків операційного автомата при реалізації мікропрограмних переходів.

Вперше виконано формалізацію принципу операційного перетворення кодів станів на основі математичного апарата універсальних алгебр, відповідно до якої функція переходів мікропрограмного автомата представляється кінцевою множиною часткових функцій переходів, із кожною з яких ототожнюється власний закон перетворення певної підмножини кодів станів автомата.

Вперше розроблені: структура операційного автомата переходів, призначеного для перетворення кодів станів мікропрограмного автомата, у якій множина реалізованих арифметико-логічних операцій відповідає множині часткових функцій переходів; структура мікропрограмного автомата з операційним автоматом переходів, що допускає відповідність арифметико-логічних операцій окремим мікропрограмним переходам; структура мікропрограмного автомата з операційним автоматом переходів, що припускає відповідність арифметико-логічних операцій станам автомата.

Вперше розроблено математичну модель мікропрограмного автомата з операційним перетворенням кодів станів, що представляє собою систему ізоморфізмів алгебр, у якій кожний ізоморфізм виражає єдиний спосіб інтерпретації та перетворення кодів станів автомата в рамках певної підмножини мікропрограмних переходів.

Вперше сформульовані наукові задачі структурного і алгебраїчного синтезу мікропрограмного автомата з операційним автоматом переходів, для вирішення яких розроблена методологічна база, що містить наступні складові: метод структурного представлення процесу синтезу мікропрограмного автомата з операційним автоматом переходів; метод використання транзитних станів; метод урахування ймовірностей істинності логічних умов; метод збільшення розрядності кодів станів.

Вперше розроблений метод алгебраїчного синтезу мікропрограмного автомата з операційним автоматом переходів, умовою застосування якого є фіксована множина операцій в операційному автоматі переходів.

Одержав подальший розвиток метод заміни вхідних змінних, застосування якого дозволило одержати модифіковану структуру мікропрограмного автомата з операційним автоматом переходів, яка характеризується меншою кількістю вхідних сигналів структурних блоків автомата в порівнянні зі структурою-прототипом, що може сприяти додатковому зниженню апаратурних витрат у логічній схемі автомата.

Одержав подальший розвиток метод зменшення максимальної кількості істотних вхідних змінних, застосування якого дозволило одержати модифіковану структуру мікропрограмного автомата з операційним автоматом переходів, яка характеризується, у порівнянні зі структурою-прототипом, зменшеною кількістю операцій, реалізованих операційним автоматом переходів і, як наслідок, зменшеними витратами апаратури в логічній схемі автомата.

Удосконалено методику оцінки апаратурних витрат на реалізацію логічної схеми мікропрограмного автомата, яка відрізняється тим, що структурні блоки мікропрограмного автомата з операційним автоматом переходів ототожнюються з типовими функціональними вузлами цифрових систем, за рахунок чого спрощується урахування параметрів автомата при моделюванні процесу синтезу схеми автомата.

Достовірність наукових положень, висновків і рекомендацій дисертаційної роботи підтверджується: коректністю постановки завдань досліджень і теоретичних положень, на яких ґрунтується їхнє рішення з урахуванням загальноприйнятих або обґрунтованих припущень, результатами комп’ютерного моделювання та впровадження запропонованих структур, засобів і методів у виробничий, науково-дослідний та навчальний процеси.

Особистий внесок здобувача.

Всі наукові результати дисертаційної роботи, що виносяться на захист, отримані автором особисто. Розроблені автором підходи, структури, моделі й

методи синтезу мікропрограмних автоматів з операційним перетворенням кодів станів розглянуті як у працях без співавторства [1-12, 24, 25], так і в працях, виконаних у співавторстві.

В роботах, написаних у співавторстві, здобувачеві належать такі результати: [13] – метод використання транзитних станів; [14] – структурна організація і класифікація операційного автомата переходів; [15] – способи формування кодів операцій переходів; [16] – представлення мікропрограмного автомата з операційним автоматом переходів за допомогою математичного апарата теорії універсальних алгебр; [17] – використання методу заміни вхідних змінних у модифікованій структурі мікропрограмного автомата з операційним автоматом переходів; [18] – аналіз особливостей реалізації функції виходів мікропрограмного автомата з використанням кінцевої множини операцій; [19, 26] – принцип операційного перетворення кодів станів у мікропрограмному автоматі; [20, 28, 29] – спільна реалізація мікропрограмних переходів операційним і канонічним способами; [21, 27] – визначення області ефективного застосування мікропрограмного автомата з операційним автоматом переходів; [22, 30] – структурний підхід до класифікації методів синтезу мікропрограмних автоматів з операційним перетворенням кодів станів; [23] – метод зменшення кількості істотних вхідних змінних у мікропрограмному автоматі з операційним автоматом переходів без заміни вхідних змінних.

Апробація результатів дисертації. Основні положення дисертаційної роботи апробовані на таких міжнародних конференціях:

на 3-й Міжнародній науково-практичній конференції молодих учених, аспірантів, студентів «Сучасна інформаційна Україна: інформатика, економіка, філософія» (Донецьк, 2009 р.);

на 10-му Міжнародному симпозіумі «IEEE East-West Design & Test Symposium (EWDTS-2012)» (Харків, 2012 р.);

на 12-й Міжнародній конференції «IFAC Conference on Programmable Devices and Embedded Systems» (Велке Карловіце, Чехія, 2013 р.);

на 14-й Міжнародній конференції «The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM)» (Поляна, 2017);

на Всеукраїнській науково-практичній конференції з міжнародною участю «Інформатика та системні науки (ІСН-2017)» (Полтава, 2017);

на Міжнародній науковій конференції «Сучасна інформатика: проблеми, досягнення та перспективи розвитку», присвяченій 60-річчю заснування Інституту кібернетики імені В.М. Глушкова НАН України (Київ, 2017);

на 9-й Міжнародній конференції «2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT'2018)» (Київ, 2018).

Структура дисертації. Дисертаційна робота складається із вступу, п'яти розділів, висновків, списку використаних джерел, додатку.

У ***першому розділі*** виконано аналіз існуючих способів побудови пристроїв керування цифрових систем. Розглянуті сучасні підходи до оптимізації схем пристроїв керування. Досліджені архітектурні властивості відомих типів операційних автоматів.

Аналіз особливостей розглянутих класів пристроїв керування дозволив з'ясувати, що в умовах безперервного зростання складності алгоритмів керування цифровими системами та відповідного підвищення вимог до швидкодії пристроїв керування найбільш перспективним виявляється мікропрограмний автомат, який характеризується максимальною швидкодією серед інших класів ПК за рахунок реалізації багатоспрямованих мікропрограмних переходів за один такт своєї роботи.

Аналіз проблеми оптимізації апаратурних витрат в логічній схемі мікропрограмного автомата показав, що у відомих структурах МПА використовується канонічний принци перетворення кодів станів за системою булевих рівнянь. Як наслідок, відомі методи, спрямовані на зменшення апаратурних витрат в схемах МПА, переважно полягають у збільшенні кількості рівнів логічного перетворення сигналів та призводять до зменшення швидкодії пристрою у порівнянні із канонічною реалізацією автомата. У якості альтернативи пропонується єдиний підхід, що полягає у перетворенні кодів станів автомата за

допомогою кінцевої множини арифметико-логічних операцій. Цей підхід може сприяти зменшенню апаратурних витрат за рахунок багаторазового використання окремих функціональних блоків в процесі реалізації мікропрограмних переходів автомата. Це свідчить про актуальність виконання дослідження і визначає тему дисертаційної роботи.

Наприкінці розділу, виходячи з проведеного аналізу, сформульовано проблему, поставлені мета та завдання наукового дослідження.

У ***другому розділі*** запропоновано теоретичну основу побудови мікропрограмного автомата з операційним перетворенням кодів станів. На прикладі відомої структури мікропрограмного автомата на лічильнику показано, що функція переходів автомата може бути представлена у вигляді множини часткових функцій переходів, у якій кожна часткова функція відповідає певній підмножині мікропрограмних переходів автомата. Виконано формалізацію даного підходу для абстрактного і структурного автоматів із використанням математичного апарата універсальних алгебр. Для автомата на лічильнику зроблений ряд узагальнень, на основі яких сформульовано принцип операційного перетворення кодів станів в мікропрограмному автоматі. Введено поняття проміжної алгебри переходів, що передбачає скалярно-векторну інтерпретацію кодів станів автомата.

Отримані результати дозволили сформулювати основні положення принципу операційного перетворення кодів станів МПА, згідно з якими функція переходів автомата реалізується за допомогою кінцевої множини арифметико-логічних операцій, що виконуються над кодом поточного стану та вхідними сигналами автомата. Даний принцип передбачає реалізацію функції переходів автомата за допомогою множини операцій замість її канонічної реалізації за системою булевих рівнянь та спрямований на зменшення апаратурних витрат в логічній схемі МПА.

Третій розділ присвячений структурній організації мікропрограмного автомата з операційним перетворенням кодів станів.

Запропоновано математичну модель мікропрограмного автомата з операційним перетворенням кодів станів, що являє собою систему ізоморфізмів алгебр, відповідних до абстрактного, структурного та проміжного рівнів представлення часткових функцій переходів автомата. На основі математичної моделі розроблено структуру операційного автомата переходів (ОАП), функцією якого є перетворення коду поточного стану автомата у код наступного стану за допомогою множини операцій. Операційна частина ОАП представляється у вигляді сукупності комбінаційних схем, що імплементують операції, використовувані для перетворення кодів станів автомата (операції переходів). Запропоновано структуру мікропрограмного автомата з операційним автоматом переходів (МПА з ОАП), в якому функція переходів реалізується сумісним функціонуванням ОАП та схеми формування кодів операцій переходів.

Розроблено модифіковану структуру МПА з ОАП, в якій операції переходів зіставляються не окремим мікропрограмним переходам, а станам автомата. Даний підхід дозволяє спростити схему формування кодів операцій переходів за рахунок зменшення кількості вхідних сигналів. Також запропоновано ряд структурних модифікацій МПА з ОАП, що базуються на відомих методах оптимізації апаратурних витрат в схемах автоматів, таких як заміна вхідних змінних та зменшення максимальної кількості істотних вхідних змінних. Проаналізовано можливість та доцільність реалізації функції виходів мікропрограмного автомата за допомогою кінцевої множини арифметико-логічних операцій у порівнянні із канонічним підходом.

У ***четвертому розділі*** розглянуті питання структурного синтезу мікропрограмних автоматів з операційним перетворенням кодів станів.

Запропоновані основні етапи структурного синтезу МПА з ОАП, до яких належать алгебраїчний синтез та синтез логічної схеми автомата. Сформульовано задачу алгебраїчного синтезу МПА з ОАП, яка полягає у формуванні системи ізоморфізмів відповідно до математичної моделі автомата. Уведені поняття формального, ефективного та оптимального розв’язків задачі алгебраїчного синтезу. Визначене поняття методології алгебраїчного синтезу МПА з ОАП як

наукової області, що охоплює будь-які теоретичні і практичні питання, пов’язані із синтезом даного класу МПА.

В рамках методології алгебраїчного синтезу МПА з ОАП запропоновано ряд методів, які можуть використовуватись при розробці методів синтезу МПА з ОАП: структурне представлення процесу синтезу МПА з ОАП, алгебраїчний синтез методом повного перебору, додавання транзитних станів, урахування ймовірностей істинності логічних умов, збільшення розрядності кодів станів автомата.

Запропоновано метод алгебраїчного синтезу мікропрограмного автомата з операційним автоматом переходів, що визначається наступними обмеженнями: пошук формальних розв’язків задачі алгебраїчного синтезу виконується шляхом часткового перебору варіантів; припускається збільшення кількості станів автомата; структура операційного автомата переходів є фіксованою. Досліджені особливості другого етапу структурного синтезу МПА з ОАП, що полягає у синтезі логічної схеми автомата за результатами алгебраїчного синтезу.

У ***п’ятому розділі*** проведені дослідження розроблених структур мікропрограмного автомата з операційним перетворенням кодів станів.

Запропоновано загальну методологію досліджень, згідно з якою ефективність розроблених структур МПА з ОАП визначається відношенням чисельно виражених апаратурних витрат у досліджуваній структурі та канонічній структурі МПА. Витрати апаратури в кожній розглянутій структурі є сумою витрат в окремих структурних блоках, які ототожнюються з типовими функціональними блоками цифрових систем. Для типових функціональних блоків розроблені VHDL-моделі, за допомогою яких проведені експериментальні дослідження та отримані аналітичні залежності апаратурних витрат від параметрів блоків при їх реалізації в базисі ПЛІС FPGA.

За результатами експериментальних досліджень для досліджуваних структур МПА отримані аналітичні залежності апаратурних витрат від параметрів автомата. Проведене дослідження ефективності МПА Мілі і Мура з операційним автоматом переходів у порівнянні із канонічним МПА для автоматів різної

складності. Досліджено вплив окремих параметрів автомата на ефективність структур, що дозволило визначити область ефективного застосування запропонованих структур МПА з ОАП у вигляді сукупності діапазонів значень відповідних параметрів. За результатами досліджень мікропрограмний автомат Мілі з операційним автоматом переходів має виграш в апаратурних витратах у порівнянні з канонічним МПА в середньому до 10%, у той час як для автомата Мура виграш складає в середньому 20-37 %, за деяких умов сягаючи 50% і більше.

Експериментальні дослідження запропонованих структур і методів синтезу наведені в п’ятому розділі дисертаційної роботи. Запропоновані структури і методи синтезу мікропрограмного автомата з операційним перетворення кодів станів порівняно з канонічною структурою мікропрограмного автомата потребують менших витрат апаратури на реалізацію логічної схеми автомата.

Практичне значення отриманих результатів полягає у тому, що на основі отриманих та викладених в дисертації теоретичних наукових положень, висновків, пропозицій та рекомендацій розроблені працездатні практичні високоефективні пристрої керування, що дозволяють знизити загальну вартість цифрових систем та зберегти високу швидкодію, властиву пристроям керування на базі мікропрограмних автоматів, за рахунок адаптації схеми пристрою до параметрів імплементованого алгоритму керування.

Результати дисертаційної роботи впроваджені в технологічні процеси промислових підприємств, в науково-дослідних організаціях, використовуються для практичної реалізації цілей освітнього європейського проекту, спрямованого на розроблення магістерських і аспірантських курсів, а також курсів підвищення кваліфікації фахівців з Інтернету речей, а також для підготовки спеціалістів у вищих навчальних закладах України. Впровадження, підтверджені відповідними актами, використовуються на наступних підприємствах і в установах:

– на підприємстві ТОВ «С-інжинірінг» при проектуванні спеціалізованих пристроїв керування;

– в науково-технічному спеціальному конструкторському бюро «Полісвіт» (державне науково-виробниче підприємство «Об’єднання "Коммунар"») при розробленні бортових авіаційних обчислювальних систем та пристроїв;

– в освітньому європейському проекті ERASMUS+ «ALLIOT» – Internet of Things: Emerging Curriculum for Industry and Human Applications, 573818-EPP-1-1-2016-1-UK-EPPKA2-CBHE-JP при розробці модуль ТММ 6.4 «Development and hardware optimization of control units for IoT devices» в рамках курсу ТМ6 «Internet of Things for industrial systems»;

– у навчальному процесі кафедри комп’ютерних інтелектуальних систем та мереж Одеського національного політехнічного університету при викладанні дисциплін «Технології проектування комп’ютерних систем» та «Проектування і діагностика систем критичного застосування».

Публікації. Матеріали дисертації повною мірою викладені у 30 публікаціях, з них 1 – в одноосібній монографії (12 авт. арк.), 22 у фахових періодичних виданнях України з технічних наук, з них 11 статей опубліковані одноосібно, 1 стаття англійською мовою, 1 стаття в електронному виданні; 7 статей включені в наукометричні бази Scopus та WoS; 7 тез доповідей у матеріалах міжнародних наукових конференцій (4 англійською мовою, включені до міжнародної наукометричної бази Scopus).

Структура та обсяг роботи. Дисертація складається із вступу, 5 розділів, висновків, списку використаних джерел, додатків. Загальний обсяг роботи складає 399 сторінок тексту, що містять 2 анотації на 19 сторінках, 79 рисунків, 57 таблиць, список використаних джерел з 289 найменувань на 27 сторінках, 4 додатки на 40 сторінках.

1 СТРУКТУРНИЙ СИНТЕЗ ЦИФРОВИХ АВТОМАТІВ

1.1 Канонічний синтез цифрових автоматів

В теорії обчислювальної техніки одним з основних понять є цифровий автомат, під яким розуміється перетворювач дискретної інформації [87]. Цифрові автомати прийнято розділяти на автомати без пам'яті (комбінаційні схеми) і автомати з пам'яттю (послідовнісні схеми). Найпростішим цифровим автоматом без пам'яті є логічний вентиль, що реалізує одну з функцій булевої алгебри. З'єднання вентилів у вигляді мережі без зворотних зв'язків породжує складні комбінаційні схеми (суматор, дешифратор та інші). Найпростішим автоматом з пам'яттю є тригер, що має два стійкі стани. Сукупність тригерів являє собою регістр, що зберігає стан послідовнісної схеми.

Математичною моделлю абстрактного цифрового автомата є п’ятикомпонентний вектор [3, 76, 78, 80, 84, 85, 114, 143, 163, 236]:

$$S_A =< A, Z, W, \delta, \lambda >, \qquad (1.1)$$

де $A = \{a_1, ..., a_M\}$ – множина станів; $Z = \{z_1, ..., z_F\}$ – множина вхідних сигналів (вхідний алфавіт) автомата; $W = \{w_1, ..., w_G\}$ – множина вихідних сигналів (вихідний алфавіт) автомата; δ – функція переходів, що реалізує відображення множини $D_\delta \subseteq A \times Z$ на A ($a_s = \delta(a_m, z_f)$, де $a_m, a_s \in A, z_f \in Z$); λ – функція виходів, що реалізує відображення множини $D_\lambda \subseteq A \times Z$ на W ($w_g = \lambda(a_m, z_f)$, де $a_m \in A, z_f \in Z, w_g \in W$). Множини A, Z, W іноді називаються алфавітом станів, вхідним та вихідним алфавітами відповідно. Автомат називається *кінцевим*, якщо кінцеві його множини.

Зазвичай практичне використання автомата можливо тільки в тому випадку, якщо його робота починається з певного стану, який називають початковим станом автомата [143, 163]. При визначенні поведінки автомата на інтервалі дискретного часу початковий стан відповідає стану автомата в початковий момент часу [163]. Абстрактний цифровий автомат з виділеним початковим

станом називається *ініціальним* та представляється шестикомпонентним вектором:

$$S_A =< A, Z, W, \delta, \lambda, a_П >, \tag{1.2}$$

де $a_П \in A$ – початковий стан автомата. Саме у вигляді моделі (1.2) абстрактний автомат визначається в роботах [5, 29, 87, 107, 119, 147].

Абстрактний автомат прийнято розглядати як пристрій, що має один вхідний і один вихідний канали, і в кожний дискретний момент часу виконує перетворення вхідного слова у вихідне (рис. 1.1) [29].

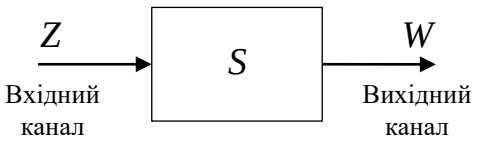


Рисунок 1.1 – Абстрактний автомат

Найбільше поширення на практиці одержали три типи автоматів:

1. Автомат Мілі, що задається рівняннями

$$\begin{cases} a(t+1) = \delta(a(t), z(t)), \\ \omega(t) = \lambda(a(t), z(t)). \end{cases} \tag{1.3}$$

2. Автомат Мура, що задається рівняннями

$$\begin{cases} a(t+1) = \delta(a(t), z(t)), \\ \omega(t) = \lambda(a(t)). \end{cases} \tag{1.4}$$

3. Комбінаційна схема, що задається рівняннями

$$\begin{cases} A = \{a_1\}; \\ \omega(t) = \lambda(a(t)). \end{cases} \tag{1.5}$$

Побудова абстрактних автоматів зводиться до вирішення задачі опису необхідної, можливої і забороненої поведінки шляхом синтезу систем рівнянь без урахування конкретної структури (функціональної схеми) автомата [84, 85, 100, 114]. За етапом абстрактного синтезу йде етап структурного синтезу, що має метою побудову логічної схеми автомата з елементів заданого типу. Найбільш загальною моделлю на цьому етапі є структурний автомат – пристрій, що реалізує

закон поведінки абстрактного автомата і являє собою мережу з логічних елементів.

У відмінності від абстрактного автомата, що має один вхідний і один вихідний канали, структурний автомат має:

1. L вхідних каналів, на які надходять елементи множини вхідних сигналів $X = \{x_1, ..., x_L\}$, $x_l \in \{0,1\}$, де $L \geq \lceil \log_2 F \rceil$, $F = |Z|$, Z – вхідний алфавіт абстрактного автомата.

2. N вихідних каналів, з яких знімаються елементи множини вихідних сигналів $Y = \{y_1, ..., y_N\}$, $x_n \in \{0,1\}$, де $N \geq \lceil \log_2 G \rceil$, $G = |W|$, W – вихідний алфавіт абстрактного автомата.

Структурний автомат задається п'ятіркою

$$S_S = < X, Y, A, d, l >, \qquad (1.6)$$

де d – функція переходів, l – функція виходів. Кожному стану $a_m \in A$ ставиться у відповідність вектор $< e_1, ..., e_R >$, компонентами якого є стани елементів пам'яті $e_r \in \{0,1\}$, $r = \overline{1,R}$, $R \geq \lceil \log_2 M \rceil$.

Для синтезу схеми структурного автомата необхідна наявність структурно повної системи елементарних автоматів, з яких і синтезується схема. Існує загальний конструктивний прийом – *канонічний метод структурного синтезу*, який дозволяє синтезувати схему структурного автомата за таблицями або графом абстрактного автомата. Даний метод був запропонований В. М. Глушковим [87] і припускає наявність структурно повної системи, що включає елементарні автомати двох типів: автомати з пам'яттю, які мають більш ніж один стан (елементи пам'яті), і автомати без пам'яті, що мають рівно один стан (логічні елементи).

Елементи пам'яті в цьому випадку розглядаються як автомати Мура з нетривіальною пам'яттю, що мають повноту системи переходів і системи виходів. Автомат має повноту системи переходів, якщо з будь-якого стану $a_m \in A$ можливий перехід у будь-який інший стан. Автомат має повноту системи виходів, якщо кожному стану $a_m \in A$ відповідає унікальний вихідний сигнал $w_g \in W$. Це

дозволяє ототожнити стани і виходи. Автомати з повнотою системи переходів і виходів називаються повними автоматами. Використання повних автоматів як елементів пам'яті структурних автоматів дозволяє зменшити число елементів пам'яті до $R = \lceil \log_k M \rceil$, де k – кількість станів елемента пам'яті.

Канонічний метод структурного синтезу заснований на доведеній В. М. Глушковим теоремі про структурну повноту. Усяка система елементарних автоматів, яка містить повний автомат Мура з нетривіальною пам'яттю і яку-небудь функціонально повну систему логічних елементів, є структурно повною. Існує загальний конструктивний прийом (канонічний метод структурного синтезу), який дозволяє звести синтез структурного автомата до синтезу його комбінаційної схеми.

Результатом застосування канонічного методу є система булевих рівнянь, що задає залежність вихідних сигналів і сигналів, що подаються на входи запам'ятовувальних елементів, від сигналів, що надходять на входи автомата ззовні, і сигналів, що знімаються з виходів запам'ятовувальних елементів. Ці рівняння називаються *канонічними*.

Модель структурного автомата (рис. 1.2) складається з комбінаційної схеми КС і елементів пам'яті $П_1, ..., П_R$, які утворюють пам'ять автомата.

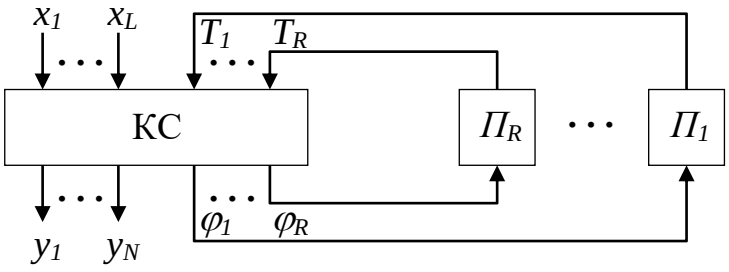


Рисунок 1.2 – Модель структурного автомата

На входи КС надходять вхідні сигнали $x_1, ..., x_L$ та сигнали $T_1, ..., T_R$, що формуються на виходах елементів пам'яті та утворюють множину внутрішніх змінних T структурного автомата. На виходах КС формуються вихідні сигнали

$y_1, ..., y_N$ та функції збудження елементів пам'яті $\varphi_1, ..., \varphi_R$, що утворюють множину функцій збудження пам'яті автомата Φ. Зазвичай пам'ять автомата є синхронною, а схема КС – асинхронною, що дозволяє виділити в роботі автомата інтервал часу, який називають тактом. Існують і асинхронні автомати [8, 115, 120, 176, 244, 288], однак у даній роботі вони не розглядаються.

Комбінаційна схема КС задається системою канонічних рівнянь (1.7), яка у скороченій формі що представляється виразом (1.8).

$$\begin{cases} \varphi_1 = \varphi_1(x_1, ..., x_L, T_1, ..., T_R), \\ ... \\ \varphi_N = \varphi_N(x_1, ..., x_L, T_1, ..., T_R), \\ y_1 = y_1(x_1, ..., x_L, T_1, ..., T_R), \\ ... \\ y_N = y_N(x_1, ..., x_L, T_1, ..., T_R); \end{cases} \quad (1.7)$$

$$\begin{cases} \Phi = \Phi(X, T); \\ Y = Y(X, T). \end{cases} \quad (1.8)$$

Оскільки що в системі (1.8) вихідні сигнали залежать від вхідних сигналів і станів, то система рівнянь (1.8) задає автомат Мілі. Автомат Мура задається системою рівнянь

$$\begin{cases} \Phi = \Phi(X, T); \\ Y = Y(T). \end{cases} \quad (1.9)$$

У загальному випадку канонічний метод структурного синтезу цифрових автоматів включає наступні етапи [29, 87]:

- завдання абстрактного автомата позначеною таблицею переходів-виходів;
- кодування елементів множин станів, входів і виходів;
- створення структурних таблиць переходів і виходів;
- вибір елементів пам'яті;
- створення розширеної таблиці переходів-виходів із внесенням до неї функцій збудження елементів пам'яті;
- вибір елементного базису для проектування цифрового автомата;

– формування функцій збудження елементів пам'яті та функцій виходів автомата;
– побудова логічної схеми автомата в обраному елементному базисі.

1.2 Методи схемної імплементації алгоритмів

Як відомо, алгоритм керування може бути реалізований або логічною схемою з «жорсткими» зв'язками (апаратний спосіб), або програмою, що виконується на ЕОМ (програмний спосіб) [122, 145]. У роботі [91] ці методи названі відповідно способами структурної та оперативної фіксації алгоритму. До теперішнього часу сформувалися два напрямки реалізації систем керування – програмувальні мікропроцесорні системи і схеми з «жорсткою» логікою. Кожному із цих методів властиві свої переваги та недоліки: системи із «жорсткою» логікою мають високу швидкодію, а мікропроцесорні системи відрізняються гнучкістю й можливістю перепрограмування. При цьому є очевидна тенденція зсуву центру ваги убік апаратної реалізації алгоритмів керування [130]. Дана дисертаційна робота присвячена апаратним методам реалізації алгоритмів.

В основі керування процесом обробки інформації операційним пристроєм лежить *принцип мікропрограмного керування*, запропонований М. Уілксом у 1951 р. [282, 283] і деталізований в роботах В. М. Глушкова [89, 90]. Згідно із цим принципом будь-яка складна операція, що виконується цифровою системою, представляється у вигляді послідовності елементарних операцій обробки інформації. Ці елементарні операції називаються мікрооперaціями, причому за один такт роботи цифрової системи може виконуватися набір мікрооперацій, називаний мікрокомандою. Для керування послідовністю мікрокоманд використовуються спеціальні логічні умови, що визначають стан процесу виконання операції. Алгоритм виконання операції, записаний у термінах мікрокоманд і логічних умов, називається мікропрограмою [125].

Таким чином, керуючий автомат в складі операційного пристрою може розглядатися як пристрій, що є схемною імплементацією мікропрограми і формує під впливом логічних умов розподілену в часі послідовність мікрокоманд, відповідну до мікропрограми. При формалізованому проектуванні КА використовується велика кількість математичних моделей, основною з яких є модель кінцевого детермінованого автомата, запропонована С. Кліні [1].

Для завдання автоматів використовуються стандартні й початкові мови [121]. До стандартних мов належать таблиці переходів, матриці переходів, діаграми переходів, таблиці включень тощо. [2, 82, 87, 93, 99, 129]. До початкових мов, найбільш зручних для початкового завдання автомата, належать мова логічних схем алгоритмів, мова регулярних виразів, предикативна мова [121].

У практиці інженерного проектування для завдання алгоритмів функціонування автомата широко використовується структурно-орієнтована мова граф-схем алгоритмів (ГСА), запропонована Л. А. Калужніним [105], яка представляє собою графічний аналог логічних схем алгоритмів, уведених О. А. Ляпуновим у 1953 р. для опису блок-схем програм для ЕОМ. Приймемо дану мову до використання в рамках дисертаційного дослідження.

Граф-схема алгоритму – це кінцевий орієнтований зв'язний граф, що містить вершини чотирьох типів: початкову, кінцеву, операторну та умовну [29]. У кожній операторній вершині записується мікрокоманда $Y_t \subseteq Y$, у кожній умовній вершині записується один з елементів множини логічних умов X. Іноді до складу ГСА вводяться вершини додаткових типів: фіктивні операторні вершини, вершини очікування, вершини маркування (для аналізу ГСА з використанням мереж Петрі), вершини розпаралелювання і синхронізації (при розгляді паралельних алгоритмів). Також у складі ГСА можуть бути виділені більші елементи, представлені підмножинами її вершин і дуг: операторні лінійні ланцюги, фрагменти (початковий, паралельний, альтернативний, циклічний з перед- або постумовою) та інші. Будучи засобом опису алгоритму, ГСА задовольняє ряду вимог [29], які забезпечують масовість, детермінованість і

результативність алгоритму, що задається. Завдання ГСА відповідає початковому етапу абстрактного синтезу КА [87], що завершується оцінкою станів і побудовою таблиці переходів автомата, яка є завданням графа автомата у вигляді списку.

Розглянемо три найбільш відомі методи реалізації керуючих автоматів [36, 58].

Мікропрограмний автомат.

Мікропрограмний автомат (МПА) (автомат з «жорсткою» логікою) імплементує алгоритм керування, представляючи його у вигляді композиції комбінаційної схеми КС і регістру пам'яті РП (рис. 1.3). Наявність регістру РП пояснюється в такий спосіб. У результаті роботи МПА формується розподілена в часі послідовність наборів мікрооперацій $Y(0), Y(1), ..., Y(t)$, де t – автоматний час, що задається сигналами синхронізації *Clock*.

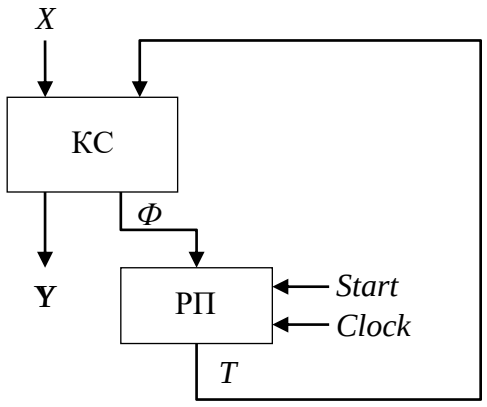


Рисунок 1.3 – Структура мікропрограмного автомата

Початковий момент часу $t = 0$ задається одиночним імпульсом *Start*. Для формування такої послідовності необхідна інформація про передісторію роботи схеми, яка визначається вхідними наборами $X(0), ..., X(t-1)$, що поступають в попередні моменти часу. Таким чином, вихідний сигнал $Y(t)$ у момент часу t визначається як

$$Y(t) = f(X(0), ..., X(t-1), X(T)). \qquad (1.10)$$

Функції виду (1.11) є громіздкими і на практиці не можуть бути реалізовані апаратно за наявністю циклів з невідомим числом повторень. Для

відображення передісторії роботи схеми використовуються стани МПА, що утворюють множину $A=\{a_1,...,a_M\}$. Стани $a_m \in A$ кодуються двійковими кодами $K(a_m)$ розрядності

$$R=\lceil \log_2 M \rceil, \tag{1.11}$$

для чого використовуються внутрішні змінні з множини $T=\{T_1,...,T_R\}$. Код поточного стану зберігається в регістрі РП, що складається з R тригерів, синхронізованих сигналом *Clock*. Код початкового стану $a_1 \in A$ заноситься в РП за сигналом *Start*, зміна вмісту РП відбувається за сигналом *Clock* на підставі функцій збудження тригерів РП, що утворюють множину $\Phi=\{\varphi_1,...,\phi_R\}$. Як правило, РП реалізується на тригерах D-типу [151, 221].

Комбінаційна схема формує функції збудження

$$\Phi=\Phi(X,T) \tag{1.12}$$

і вихідні сигнали, спосіб формування яких залежить від моделі МПА [87]. В автоматі Мілі система мікрооперацій Y реалізується у вигляді

$$Y=Y(X,T), \tag{1.13}$$

а в автоматі Мура вихідні сигнали залежать тільки від кодів станів:

$$Y=Y(T). \tag{1.14}$$

Метод синтезу МПА по ГСА включає наступні етапи [29]:

– формування позначеної ГСА;

– кодування станів МПА;

– формування прямої структурної таблиці МПА;

– формування систем функцій переходів і виходів;

– реалізація схеми МПА в заданому елементному базисі.

Порівняльний аналіз схем автоматів Мілі й Мура, побудованих для однієї і тієї ж ГСА, приводить до наступних висновків:

– автомат Мура має більше станів і більше переходів, ніж еквівалентний автомат Мілі;

– система вихідних сигналів автомата Мура має регулярний характер, тому що залежить тільки від станів МПА.

Відзначимо, що в практиці інженерного проектування модель Мура використовується частіше [151], тому що їй властива стійкість функціонування. Крім того, регулярність системи (1.15) дозволяє ефективно використовувати для її реалізації базис запам'ятовувальних пристроїв (ЗП) – ПЗП, ППЗП та інші [36].

У схемі на рис. 1.9 обробка інформації виконується тільки схемою КС, що утворює єдиний рівень перетворення логічних сигналів. В літературі такі схеми прийнято називати однорівневими [36, 58]. Однорівневі схеми мають максимально можливу швидкодію, однак їм властивий один серйозний недолік – максимальні витрати апаратури в логічній схемі автомата, що приводять до подорожчання проектованого пристрою. Якщо критерієм оптимальності схеми автомата є мінімум апаратурних витрат, то використовуються багаторівневі схеми автоматів.

У цей час застосовуються три групи методів оптимізації логічної схеми автомата [37]:

1. Методи структурної редукції, що приводять до збільшення числа рівнів перетворення логічних сигналів у схемі автомата.

2. Методи гетерогенної реалізації, що полягають у використанні різного елементного базису для реалізації схем різних рівнів.

3. Алгоритмічні методи, що полягають в оптимальному кодуванні станів і додаткових змінних автомата.

Класифікація методів оптимізації апаратурних витрат у логічній схемі автомата Мілі наведена в роботі [36]. Часто методи оптимізації використовуються спільно. Це означає, що при побудові багаторівневих схем (структурна редукція) оптимізація схеми полягає в оптимальному кодуванні змінних (алгоритмічні методи) і застосуванні різного елементного базису для реалізації різних блоків структурної схеми автомата (гетерогенна реалізація).

Як правило, моделі МПА використовуються у випадку проектування швидкодіючих операційних пристроїв [125]. Якщо ж швидкодія системи має вторинний характер, а головна мета проекту – зменшення вартості системи, то керуючий автомат може бути реалізований у вигляді автомата з «програмувальною» логікою.

Мікропрограмний пристрій керування.

Мікропрограмний пристрій керування (МПК), який також називають автоматом з «програмувальною» логікою (АПЛ), заснований на операційно-адресному представленні керуючих слів (мікрокоманд), що зберігаються у керуючій пам'яті (КП) [125]. Метод синтезу МПК включає наступні етапи [58]:

– перетворення заданої граф-схеми алгоритму;

– формування мікрокоманд заданого формату;

– адресація мікрокоманд;

– кодування операційної і адресної частин мікрокоманд;

– формування вмісту керуючої пам'яті;

– синтез схеми МПК в заданому елементному базисі.

Як правило, формати мікрокоманд включають наступні поля: FY, FX, FA_0, FA_1. Поле FY (операційна частина мікрокоманди) містить інформацію про виконувані у даному такті t мікрооперації $y_n \in Y$ $(t = 0, 1, ...)$. Поле FX містить інформацію про логічну умову $x_l \in X$, яка перевіряється в момент часу t $(t = 0, 1, ...)$. Поле FA_0 містить адресу A^{t+1} мікрокоманди, яка буде виконуватись в наступний момент часу (адресу переходу), якщо перехід виконується безумовно, або якщо логічна умова $x_l = 0$. Поле FA_1 містить адресу переходу при $x_l = 1$. Поля FX, FA_0, FA_1 утворюють адресну частину мікрокоманди. Структура МПК наведена на рис. 1.4. Такий МПК функціонує наступним чином.

Чергова МК зчитується з КП за адресою, що перебуває в регістрі адреси мікрокоманди (РАМК). Операційна частина (ОЧ) мікрокоманди надходить до схеми формування мікрооперацій (СФМО) і перетворюється в мікрооперації *Y*, що керують операційним автоматом. Адресна частина (АЧ) використовується схемою формування адреси СФА поряд з логічними умовами *X* для формування в РАМК адреси наступної мікрокоманди. Функціонування завершується після формування ознаки закінчення мікропрограми y_k.

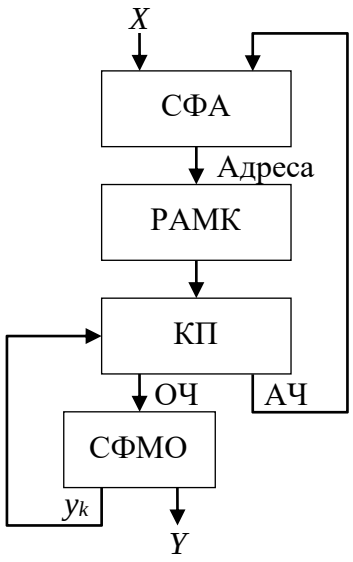


Рисунок 1.4 – Структура мікропрограмного пристрою керування

МПК відрізняються універсальністю й регулярністю структури, а також досить простими алгоритмами синтезу. Метод синтезу схеми залежить від способу адресації мікрокоманд [97, 113], при цьому існують три основні способи:

1. Примусова адресація, при якій формат МК включає поля FY, FX, FA_0 та FA_1. Цей метод приводить до найбільш швидких мікропрограм, що мають мінімальну кількість мікрокоманд. Однак довжина МК є найбільшою, і керуюча пам’ять використовується неефективно (рис. 1.5, *а*).

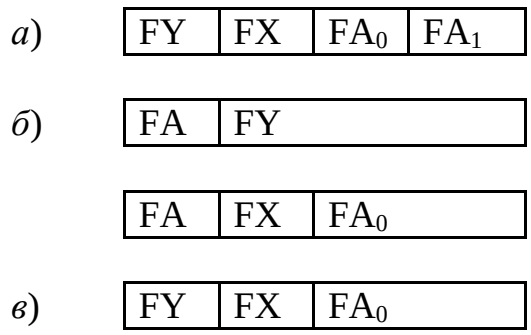


Рисунок 1.5 – Формати мікрокоманд з примусовою (*а*), природньою (*б*) та комбінованою (*в*) адресацією

2. Природня адресація мікрокоманд, при якій існують два типи мікрокоманд. Операційна МК містить поле ознаки $\mathrm{FA}=0$ і поле FY, при цьому адреса наступної мікрокоманди стає на одиницю більшою за адресу поточної

мікрокоманди. Керуюча мікрокоманда містить поле ознаки $FA=1$, поле FX та поле адреси переходу FA_0. Якщо ЛУ, що перевіряється, дорівнює одиниці, адреса наступної мікрокоманди стає на одиницю більшою за адресу поточної МК. Такий підхід породжує найбільш довгі мікропрограми та збільшує час виконання алгоритму керування, однак мікрокоманди мають мінімальну розрядність (рис. 1.5, *б)*.

3. Комбінована адресація мікрокоманд, при якій МК включає поля FY, FX і FA_0 (рис. 1.5, *в)*. Якщо ЛУ, що перевіряється, дорівнює одиниці, адреса переходу стає на одиницю більшою за адресу поточної МК. Такий підхід приводить до проміжних результатів у порівнянні із примусовою і природньою адресацією і найбільш часто використовується на практиці.

Усім МПК властивий один істотний недолік – якщо перехід із поточного стану є *L*-спрямованим, тобто залежить від *L* логічних умов, він виконується за *L* тактів. Це приводить до збільшення числа мікрокоманд у мікропрограмі за рахунок уведення $L-1$ додаткових МК безумовного переходу і, як наслідок, до збільшення часу виконання мікропрограми і необхідної ємності КП. Якщо багатоспрямовані переходи виконуються менш ніж за *L* тактів, то це різко ускладнює процес синтезу схеми МПК [36]. При цьому автомати з «жорсткою» логікою реалізують будь-які переходи за один такт, але відрізняються нерегулярністю структури, що ускладнює їхню апаратну реалізацію.

Як правило, при проектуванні цифрових пристроїв з використанням керуючих автоматів схема КА повинна задовольняти певним критеріям. До них належать апаратурні витрати, швидкодія, енергоспоживання, надійність, сумісність із певним елементним базисом та інші. На сьогоднішній день відома безліч методів і підходів до оптимізації даних характеристик. Аналіз сучасних публікацій дозволяє зазначити ряд наукових проблем у цій області:

1. Оптимізація апаратурних витрат у логічній схемі КА за рахунок багаторівневого перетворення логічних сигналів. В контексті даної проблеми можуть бути виділені наступні напрямки досліджень:

– оптимізація апаратурних витрат у схемах мікропрограмних автоматів Мілі й Мура [68, 183, 185, 189, 205, 222, 229, 238, 241, 245, 212, 264, 256];

– оптимізація апаратурних витрат у схемах мікропрограмних пристроїв керування [50, 61, 63, 64, 65, 190, 192, 194, 197, 202, 203, 205, 208, 246, 279, 285, 286];

– синтез і оптимізація ієрархічних, розподілених і реконфігурованих структур КА [144, 183, 227, 248, 257, 260, 261, 262, 272, 273, 276, 284].

2. Оптимізація апаратурних витрат у логічній схемі КА з урахуванням особливостей заданої граф-схеми алгоритму і кодування станів [152, 182, 186, 196, 213, 215, 216, 224, 233, 237, 241, 250, 254, 284].

3. Особливості реалізації КА в базисі сучасних ПЛІС [56, 67, 68, 153, 177, 198, 199, 200, 205, 210, 218, 226, 232, 243, 249, 275].

4. Оптимізація апаратурних витрат у логічній схемі КА за рахунок особливостей використовуваного елементного базису [55, 66, 110, 184, 193, 200, 201, 206, 207, 209, 219, 251, 266, 269]. Окремо можуть бути виділені роботи, присвячені реалізації автоматів з використанням модулів пам'яті, у тому числі пам'яті, розташованої на кристалі ПЛІС сучасних серій [62, 191, 211, 217, 247, 274].

5. Оптимізація енергоспоживання, надійності, швидкодії та інших характеристик керуючих автоматів [52, 177, 184, 220, 225, 228, 235, 253, 265, 270, 277, 280, 281, 287].

1.3 Засоби автоматизації проектування цифрових автоматів

Сьогодні провідним елементним базисом проектування цифрових обчислювальних систем є базис ПЛІС (FPGA, CPLD, SoC) [81, 94, 126, 128, 162]. Синтез цифрових пристроїв у даному базисі є сьогодні актуальним напрямком досліджень у сучасній науці [153, 234, 239, 240, 242, 243, 257, 259, 260, 268, 271]. Синтезу і оптимізації цифрових автоматів на ПЛІС присвячені, наприклад, наступні роботи: [67, 152, 200, 207, 218, 247, 275]. Підходи, розглянуті в роботах

[211, 217, 231, 247, 266, 269], пов'язані з використанням модулів пам'яті, що розташовані на кристалі ПЛІС сучасних серій. У роботах [225, 277, 280, 281, 287] розглянуті підходи до оптимізації енергоспоживання цифрових пристроїв у базисі ПЛІС.

На базис ПЛІС орієнтований ряд САПР, у яких проект пристрою може представлятися як у графічній формі, так і у формі текстового опису на мовах проектування апаратури (Hardware Description Languages, HDL) [75, 111, 157, 166, 178, 252, 258, 278]. Широке поширення одержала мова опису апаратури VHDL, що дозволяє за підтримки відповідних САПР здійснювати моделювання роботи проектованого пристрою при його реалізації в базисі ПЛІС провідних серій [9, 73, 74, 139, 150, 158, 162, 212, 223, 289].

Традиційною одиницею виміру складності ПЛІС, а також цифрових пристроїв на їхній базі є еквівалентний логічний вентиль (ЕЛВ), під яким умовно розуміється логічний елемент «2І-НЕ» [94, 162]. Для ПЛІС перших серій дана одиниця відповідала їхній внутрішній структурі, однак для наступних поколінь ПЛІС, більш різноманітних за внутрішньою архітектурою, підрахунок ЕЛВ на кристалі став важким через відсутність елементів типу 2І-НЕ в явному вигляді на кристалі ПЛІС. У цей час різні виробники ПЛІС (Xilinx, Altera, Actel, Cypress та інші) використовують різні одиниці виміру складності ПЛІС [94], що виражають кількість стандартних блоків даного типу ПЛІС. Так, для ПЛІС типу FPGA одиницею виміру складності зазвичай є табличний перетворювач (Look-Up Table, LUT) [94, 251].

Якщо раніше базовим етапом проектування цифрових пристроїв було створення принципової електричної схеми, то сьогодні основою проекту є опис пристрою однією з мов RTL-рівня (Register Transfer Level, рівень регістрових передач), наприклад, мовою VHDL. Процес проектування все більше нагадує процес програмування, де роль компілятора виконують системи синтезу. В ідеалі необхідно, щоб стандартний RTL-код міг бути реалізований на будь-якій елементній базі. Вплив ефективності систем синтезу при переході від RTL-коду

до елементного базису виробника інтегральних схем не менш суттєвий, ніж вплив якості компіляторів в традиційному програмуванні.

Одним із провідних виробників ПЛІС сьогодні виступає компанія Xilinx, що пропонує розроблювачеві не тільки широкий спектр серій мікросхем, але й професійні засоби проектування цифрових систем на їхній базі [101, 162]. Найбільш актуальну інформацію щодо апаратних та програмних продуктів, що пропонуються Xilinx, а також про особливості їх застосування при проектуванні цифрових систем можна дізнатись на офіційному сайті компанії (www.xilinx.com).

В програмних продуктах компанії Xilinx синтез схем за RTL-описом здійснюється за допомогою технології XST (Xilinx Synthesis Technology, «технологія синтезу Xilinx»), вбудованої в САПР Xilinx ISE та в його сучасну версію Xilinx Vivado. Використання особливостей цієї технології при проектуванні цифрових пристроїв вимагає дотримання певних вимог при написанні синтезованого VHDL-коду. Розглянемо основні підходи до VHDL-опису мікропрограмних автоматів, рекомендовані компанією Xilinx та у відповідній літературі [75, 101, 107].

VHDL-опис МПА на основі систем булевих рівнянь.

Нехай МПА заданий ГСА Γ_{1-1}, що відзначена станами автомата Мілі $a_0 - a_2$ та станами автомата Мура $b_0 - b_3$ (рис. 1.6).

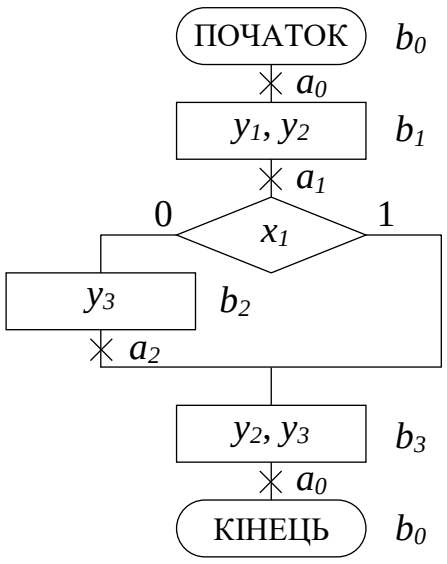


Рисунок 1.6 – Граф-схема алгоритму Γ_{1-1}

Закодуємо стани МПА Мілі і Мура за допомогою змінних T_1, T_2 дворозрядними двійковими кодами, десяткове значення яких відповідає індексу стану: $K(a_0)=00$, $K(a_1)=01$, $K(a_2)=10$; $K(b_0)=00$, $K(b_1)=01$, $K(b_2)=10$, $K(b_3)=11$. Тоді для автомата Мілі система рівнянь (1.8) матиме наступний вигляд [29, 31, 36]:

$$\begin{cases} D_1 = a_1\bar{x}_1 = \bar{T}_1T_2\bar{x}_1; \\ D_2 = a_0 = \bar{T}_1\bar{T}_2. \end{cases}$$

$$\begin{cases} y_1 = a_0 = \bar{T}_1\bar{T}_2; \\ y_2 = a_0 \vee a_2 \vee a_1x_1 = \bar{T}_1\bar{T}_2 \vee T_1\bar{T}_2 \vee \bar{T}_1T_2x_1 = \bar{T}_2 \vee \bar{T}_1T_2x_1; \\ y_3 = a_1\bar{x}_1 \vee a_2 \vee a_1x_1 = a_1 \vee a_2 = \bar{T}_1T_2 \vee T_1\bar{T}_2. \end{cases} \quad (1.15)$$

У випадку автомата Мура дані система рівнянь (1.9) буде такою:

$$\begin{cases} D_1 = b_1\bar{x}_1 \vee b_2 \vee b_1x_1 = b_1 \vee b_2 = \bar{T}_1T_2 \vee T_1\bar{T}_2; \\ D_2 = b_0 \vee b_2 \vee b_1x_1 = \bar{T}_1\bar{T}_2 \vee T_1\bar{T}_2 \vee \bar{T}_1T_2x_1. \end{cases}$$

$$\begin{cases} y_1 = b_1 = \bar{T}_1T_2; \\ y_2 = b_1 \vee b_3 = \bar{T}_1T_2 \vee T_1T_2 = T_2; \\ y_3 = b_2 \vee b_3 = T_1\bar{T}_2 \vee T_1T_2 = T_1. \end{cases} \quad (1.16)$$

В даних системах функція переходів відповідає виразу (1.12) та передбачає побудову регістру пам’яті на базі тригерів D-типу [29, 36].

Синтезована частина VHDL-опису для автомата Мура представлена лістингом А.1. Блок архітектури містить єдиний процес, який відповідає регістру пам’яті, а також реалізовану систему рівнянь (1.16) у вигляді сигналів.

Для автомата Мілі відмінності будуть лише в системі функцій переходів та виходів, яка для нашого прикладу відповідатиме системі (1.15).

Для проведення поведінкового моделювання (Behavioral Modelling) може бути використаний окремий модуль, функцією якого є генерація сигналу синхронізації *C*, сигналу скидання *Reset* та вхідного сигналу x_1 (лістинг А.2). В архітектурі даного модуля розглянутий вище модуль FSM використовується в якості компонента.

Система автоматизованого проектування Xilinx Vivado містить інструмент «RTL Analysis», який дозволяє представити VHDL-опис, наведений на лістингу А.1, у вигляді функціональної схеми пристрою (рис. 1.7).

Як можна бачити, схема автомата містить два D-тригери T_reg(1) та T_reg(2), які утворюють регістр пам’яті RTL_REG_SYNC, а також множину логічних елементів, що реалізують функції переходів та виходів відповідно до системи рівнянь (1.16).

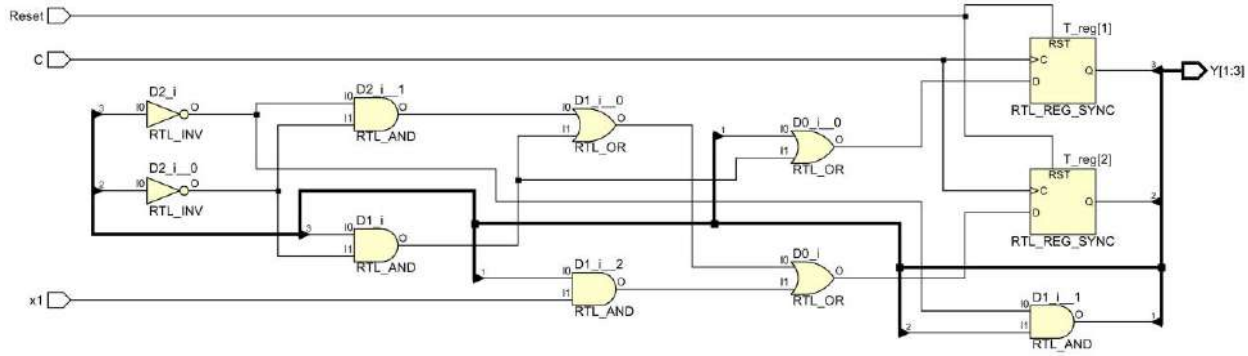


Рисунок 1.7 – Результат RTL-аналізу VHDL-опису МПА Мура (Γ_{1-1})

VHDL-модель, представлена лістингом А.1, дозволяє виконати синтез схеми в базисі внутрішніх елементів обраної ПЛІС, для чого в Xilinx Vivado використовується інструмент Synthesis, оснований на технології XST. Результат синтезу для нашого прикладу наведений на рис. 1.8.

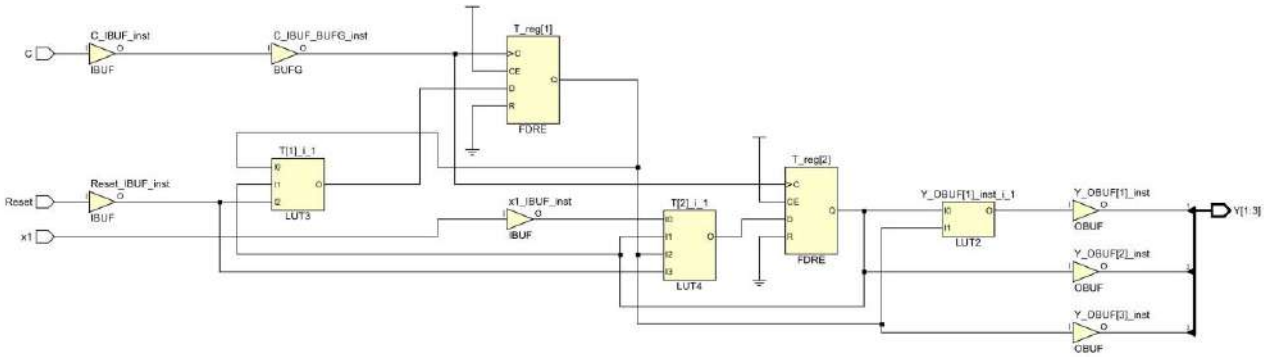


Рисунок 1.8 – Результат синтезу VHDL-опису МПА Мура,
на основі системи булевих рівнянь (Γ_{1-1})

Розташування елементів та міжелементних зв’язків на кристалі ПЛІС здійснюється на етапі імплементації відповідно до результатів синтезу [101, 162].

Перевагою VHDL-моделі МПА, представленої лістингом А.1, є можливість проведення попередньої мінімізації системи булевих рівнянь та застосування різних методів оптимізації апаратурних витрат [29, 31, 36, 58], що дозволяє отримати максимально оптимізовану схему пристрою. Недоліком є те, що будь-які зміни в алгоритмі керування, що імплементується автоматом, вимагають проведення повторного синтезу та оптимізації схеми автомата із відповідними змінами в VHDL-моделі.

Інструмент синтезу XST здатен за певних умов знаходити в VHDL-описі фрагменти коду, що відповідають опису кінцевого автомата (під кінцевим автоматом будемо розуміти автомат із невизначеними кодами станів). Даний процес називають екстракцією кінцевого автомата (FSM extraction). Для знайденого кінцевого автомата інструмент XST виконує наступні дії:

– кодування станів за обраною методикою;

– синтез регістрової схеми відповідно до обраного способу кодування станів;

– синтез і оптимізація схеми для функцій переходів і виходів.

Для забезпечення можливості автоматичної екстракції кінцевого автомата в його VHDL-описі слід дотримуватись наступних положень:

1. Стани автомата задаються у вигляді множини літералів, об’єднаних в елементі перелічуваного типу (Enumeration Type).

2. Регістр пам’яті має бути синхронним та мати можливість переводу у початковий стан за сигналом скидання.

3. Реалізація системи функцій переходів та виходів реалізується за допомогою оператора *case*.

Дані вимоги дозволяють задати автомат мовою VHDL із використанням одного, двох чи трьох процесів [218]. Розглянемо ці способи.

VHDL-опис кінцевого автомата із використанням одного процесу.

Загальна структура синтезованої і моделюючої частин VHDL-моделі аналогічна моделям, представленим лістингами А.1 і А.2 відповідно. Різниця полягає в описі архітектури об'єкта «FSM». Розглянемо VHDL-модель автомата Мілі, заданого ГСА Γ_{1-1}, у випадку його реалізації у вигляді одного процесу (лістинг А.3).

Для автомата Мура VHDL-модель на базі одного процесу матиме вигляд відповідно до лістингу А.4. В основі обох архітектур лежить єдиний процес, у якого в списку чутливості знаходиться сигнал синхронізації *C*. Оператор *process* містить в собі послідовність операторів, що виконуються і синтезуються послідовно. Результат RTL-аналізу VHDL-моделі автомата Мура, свідчить про те, що синхронне формування мікрооперацій приводить до використання тригерів для їх зберігання між тактами синхронізації, а невелика кількість станів дозволяє реалізувати логіку функцій переходів та виходів за допомогою каскаду мультиплексорів (рис. 1.9).

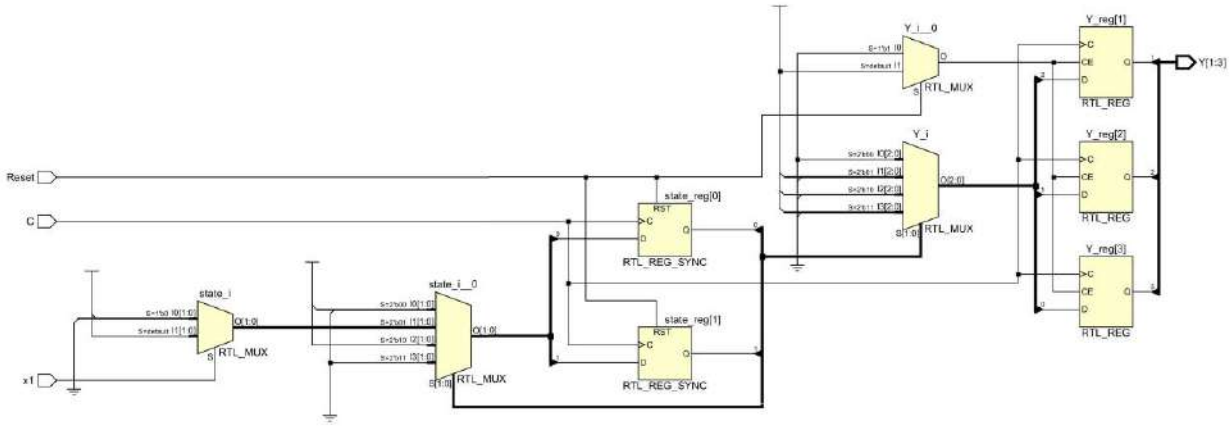


Рисунок 1.9 – Результат RTL-аналізу VHDL-опису МПА Мура (Γ_{1-1}) із використанням одного процесу

VHDL-опис кінцевого автомата у вигляді двох процесів.

Даний підхід полягає у відокремленій реалізації функцій переходів і виходів шляхом їх реалізації у різних процесах. Приклад такої реалізації для МПА Мілі

(ГСА Γ_{1-1}) представлений лістингом А.5. В тілі першого процесу, що активується зміною сигналу *C*, реалізовані регістр пам'яті та функція переходів. В другому процесі, що активується зміною стану автомата або вхідного сигналу x_1, реалізована функція виходів.

Результат RTL-аналізу автомата з архітектурою, наведеною у лістингу А.5, зображений на рис. 1.10. Відсутність сигналу синхронізації в списку чутливості другого процесу дозволяє САПР формувати вихідні сигнали асинхронно і не використовувати тригери для їх зберігання.

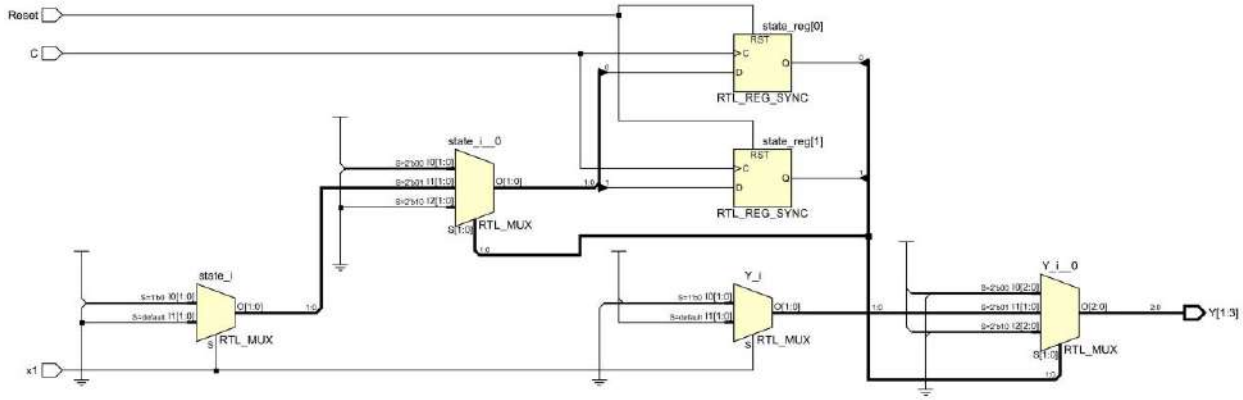


Рисунок 1.10 – Результат RTL-аналізу VHDL-опису МПА Мілі (Γ_{1-1}) із використанням двох процесів

У випадку автомата Мура перший процес залишиться незмінним, а в другому процесі вихідні сигнали формуватимуться в залежності лише від поточного стану автомата.

VHDL-опис кінцевого автомата у вигляді трьох процесів.

В цьому випадку окремі процеси використовуються для опису регістра пам'яті, функції переходів та функції виходів (лістинг А.6). В блоці архітектури задаються два сигнали *state* та *next_state*, що відповідають поточному та наступному станам автомата.

У першому процесі, що відповідає регістру пам'яті, реалізовані лише функції скидання регістру у початковий стан (в нашому прикладі – у стан a_0) та завантаження в регістр нового значення коду стану. Даний процес є єдиними

синхронним процесом в блоці архітектури автомата. У другому процесі реалізована функція переходів, але код поточного стану подається не відразу в змінну *state*, а в змінну *next_state*, тобто, фактично, на вхід регістра пам'яті. Третій процес відповідає функції виходів автомата і співпадає з другим процесом в лістингу А.5.

Результат RTL-аналізу, відповідний до лістингу А.6, зображений на рис. 1.11 і майже збігається із рис. 1.10. Можна бачити, що не зважаючи на використання двох сигналів *state* та *next_state* лише сигнал *state* синтезується як регістр, в той час як сигналу *next_state* відповідає каскад мультиплексорів.

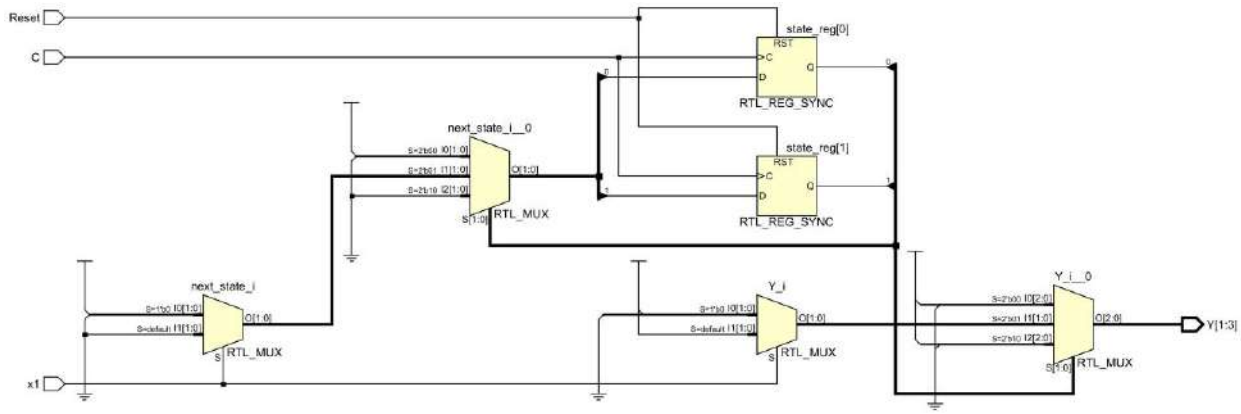


Рисунок 1.11 – Результат RTL-аналізу VHDL-опису МПА Мілі (Γ_{1-1}) із використанням трьох процесів

Представлення VHDL-опису автомата у вигляді трьох процесів не має жодних суттєвих переваг або недоліків у порівнянні із попереднім способом, окрім, можливо, більш структурованого підходу до побудови автомата. Також при використанні старих версій САПР виробників ПЛІС існує імовірність того, що процес автоматичної екстракції і синтезу кінцевого автомата, заданого у вигляді одного процесу, може не підтримуватись. Таким чином, VHDL-моделі автомата у вигляді двох або трьох процесів будемо вважати тотожними і переважнішими за модель із єдиним процесом.

Незалежно від того, скількома процесами описаний кінцевий автомат, система XST у складі САПР Xilinx Vivado здатна виконувати екстракцію

кінцевого автомата із VHDL-кода та виконувати кодування станів згідно обраного способу кодування. Для цього в розділі налаштування процесу синтезу передбачений параметр «**-fsm_extraction**», який може набувати наступні значення:

1. «One-hot». Для кодування кожного стану використовується окремий тригер. Кількість тригерів дорівнює кількості станів автомата. В кожен момент часу лише один тригер може мати одиничне значення. Для формування значення кожного тригера використовується логічне рівняння, в якому кількість термів дорівнює кількості переходів у відповідний стан.

2. «Sequential». Інструмент XST знаходить в автоматі довгі послідовності станів, що складаються із безумовних переходів, та здійснює кодування станів в їх межах послідовними двійковими кодами мінімально достатньої розрядності. В результаті на адресні входи LUT-елементів не подаються вхідні сигнали автомата, а подається лише код поточного стану, який зазвичай має невелику розрядність у порівнянні з кількістю вхідних сигналів. Послідовне кодування станів забезпечує більш оптимальне заповнення комірок статичної пам’яті LUT-елементів та зменшує кількість невикористаних комірок.

3. «Johnson». Кодування станів виконується із використанням коду Джонсона. Кожне значення цього коду містить лише одну безперервну послідовність одиничних бітів, а будь-які два сусідніх значення у впорядкованій послідовності значень розрізняються лише одним розрядом. Код Джонсона є циклічним кодом з надлишком і дозволяє знизити кількість електричних завад, що викликаються одночасним перемиканням кількох розрядів регістрової схеми. Таким чином, при використання коду Джонсона для кодування станів автомата кількість задіяних тригерів буде більшою, ніж у випадку послідовного кодування.

4. «Gray». Кодування станів виконується із використанням коду Грея, в якому два сусідніх значення у впорядкованій послідовності значень розрізняються значенням одного двійкового розряду, а кількість розрядів співпадає із кількістю у випадку послідовного кодування. Як і код Джонсона, код Грея доцільно використовувати для кодування ланцюгів станів, оскільки кожен автоматний

перехід в такому ланцюгу буде супроводжуватись зміною лише одного розряду в регістрі пам’яті автомата.

5. «Auto». Інструмент XST обирає один із вище розглянутих методів кодування на свій розсуд за результатами аналізу VHDL-опису автомата. Вибір методу кодування залежить також від інших налаштувань XST (наприклад, від провідної стратегії оптимізації – апаратурні витрати чи швидкодія), але узагальнений підхід такий: якщо автомат містить невелику кількість станів, використовується кодування «One-hot»; при середній кількості станів використовується код Джонсона; при великій кількості станів використовується код Грея. Компанія Xilinx рекомендує для використання саме режим «Auto».

6. «None». Спосіб кодування станів не регламентується. Якщо у п’яти розглянутих вище режимах кодування станів відзначається у протоколі процесу синтезу (приклад фрагменту такого протоколу наведений на рис. 1.12), то у даному випадку інформація про кодування станів не надається. Зазвичай вважається, що коди станів відповідають порядковому номеру ідентифікаторів станів при їх переліченні в VHDL-описі, починаючи з нуля ($K(a_0)=0$, $K(a_1)=1$, ...), однак інструмент XST офіційно не визначає це і залишає за собою право кодування станів на власний розсуд. Ця особливість не заважає синтезу коректної схеми автомата, але не дозволяє застосовувати додаткові методі оптимізації, основані на відомих значення кодів станів [29, 31, 36].

```
INFO: [Synth 8-802] inferred FSM for state register 'state_reg' in module 'FSM'
---------------------------------------------------------------------------------------------
          State |             New Encoding |        Previous Encoding
---------------------------------------------------------------------------------------------
             a0 |                      001 |                       00
             a1 |                      010 |                       01
             a2 |                      100 |                       10
---------------------------------------------------------------------------------------------
INFO: [Synth 8-3354] encoded FSM with state register 'state_reg' using encoding 'one-hot' in module 'FSM'
```

Рисунок 1.12 – Приклад результату автоматичної екстракції кінцевого автомата у фрагменті протоколу виконання синтезу автомата у випадку кодування станів «One-hot»

В роботі [51] проведене дослідження впливу методу кодування станів на кількість апаратурних витрат і швидкодію пристрою. Дослідження проведене на наборі тестових автоматів IWLS LGSynth93 за допомогою САПР Xilinx ISE 11.3 для мікросхеми XC5VLX30 із використанням VHDL-опису автоматів у вигляді двох процесів. Результати дослідження демонструють суттєвий вплив способу кодування станів на кількість апаратурних витрат в схемі автомата. Втім, слід відзначити, що САПР Xilinx ISE 11.3 має більше способів кодування станів автомата у порівнянні із САПР Xilinx Vivado, що робить доцільним осучаснення розглянутих в [51] результатів досліджень із використанням новішої версії САПР компанії Xilinx.

Проведений аналіз можливостей САПР Xilinx Vivado в напрямку автоматичного синтезу мікропрограмних автоматів дозволяє зробити висновок, що застосування усіх відомих методів оптимізації логічної схеми автомата, за винятком розглянутих способів кодування станів засобами XST, має проводитись безпосередньо проектувальником на етапі структурного синтезу МПА із відповідним втіленням результатів у VHDL-моделі автомата.

1.4 Концепція операційного перетворення кодів станів у мікропрограмному автоматі

Короткий аналіз теоретичних основ побудови цифрових автоматів, проведений у попередніх параграфах, свідчить про наступне.

1. У мікропрограмному автоматі основні апаратурні витрати визначаються складністю комбінаційної схеми КС (рис. 1.3), що реалізує умовні й безумовні мікропрограмні переходи і формує мікрооперації $y_n \in Y$. Розглядаючи системи функцій (1.12) та (1.13) / (1.14) роздільно, схему КС структурно можна представити у вигляді двох частин: схеми формування переходів СФП і схеми формування мікрооперацій СФМО (рис. 1.13). Наявність зв'язку, показаного пунктиром, дозволяє вважати дану структуру автоматом Мілі, відсутність зв'язку – автоматом Мура.

При синтезі СФП канонічним способом за системою булевих рівнянь виду (1.12) збільшення складності ГСА приводить до обов'язкового збільшення складності схеми внаслідок «унікальності» мікропрограмних переходів, що є наслідком унікальності кодів станів. Так, додавання в ГСА одного безумовного переходу приводить до додавання одного терма в систему канонічних рівнянь функції переходів. Додавання умовного переходу додає тим більше термів, чим більше гілок (напрямків) реалізує даний перехід. Додавання нового стану впливає на складність СФП тим, що додає в ГСА вхідний і вихідний переходи. Таким чином, складність схеми СФП збільшується в деякій залежності від кількості мікропрограмних переходів, на характер якої може впливати фактор мінімізації системи рівнянь функції переходів.

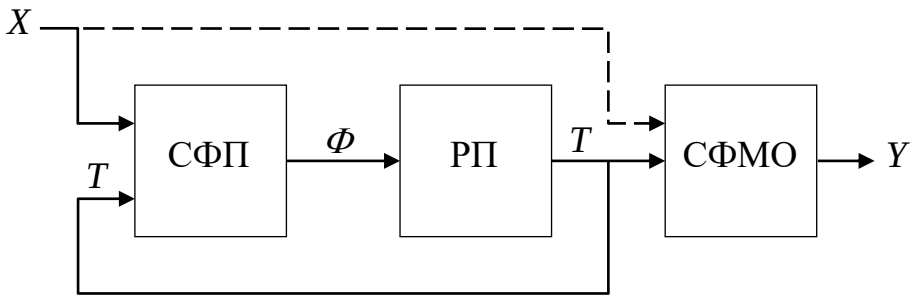


Рисунок 1.13 – Роздільне представлення комбінаційної схеми в МПА

Складність схеми формування мікрооперацій визначається числом різних мікрокоманд, що формуються автоматом. Отже, приріст апаратурних витрат, викликаний додаванням у ГСА нової операторної вершини, буде залежати від її вмісту. У загальному випадку при синтезі СФМО канонічним способом за системою рівнянь виду (1.13) або (1.14) кожна нова МО додає в систему один терм. Зважаючи на те, що СФМО може бути реалізована в базисі, відмінному від базису реалізації СФП (наприклад, у базисі запам'ятовувальних пристроїв при реалізації СФП у базисі ПЛІС), складність схеми СФМО може визначатися іншим способом, наприклад, інформаційною ємністю ЗП. Використання різних методів оптимізації СФМО (наприклад, кодування наборів МО) дозволяє звести до мінімуму зростання складності схеми СФМО при збільшенні складності ГСА [36].

2. У мікропрограмних пристроях керування (рис. 1.4) апаратурні витрати визначаються переважно інформаційною ємністю керуючої пам'яті. Зростання ємності керуючої пам'яті зі збільшенням складності ГСА залежить від використовуваного типу адресації.

При використанні примусової адресації додавання в ГСА операторної або умовної вершини додає в мікропрограму одну мікрокоманду стандартного формату. При додаванні операторної вершини інформаційне поле FX (рис. 1.5, *а*) заповнюється кодом безумовного переходу, а поле FA_0 – довільним значенням; при додаванні умовної вершини поле FY заповнюється нулями (внаслідок додавання «порожньої» операторної вершини). При цьому очевидна неефективність використання полів FX, FA_0 та FY (і, як наслідок, надмірність в інформаційній ємності керуючої пам'яті), оскільки їх вміст визначається не стільки структурою ГСА, скільки особливостями організації логічної схеми МПК. Зростання використовуваної ємності керуючої пам'яті з ростом числа вершин ГСА, є близьким до лінійного і пропорційним розрядності мікрокоманди з даним типом адресації.

У випадку природньої адресації при додаванні в ГСА операторної вершини додається операційна МК (рис. 1.5, *б)*, а при додаванні умовної вершини – керуюча МК. Якщо всі мікрокоманди розміщені в єдиному адресному просторі, то приріст інформаційної ємності схеми КП буде визначатися розрядністю, максимальної серед разрядностей двох дані типів мікрокоманд, що також приводить до неефективного використання й надмірності частини інформаційної ємності КП. Оскільки для конкретної ГСА при незмінній розрядності адреси мікрокоманди розрядність останньої є постійною величиною, можна стверджувати, що у випадку природньої адресації ріст складності схеми КП також може вважатися лінійним.

При використанні комбінованої адресації додавання в ГСА нової вершини також приводить до лінійного росту інформаційної ємності КП, оскільки, незалежно від типу вершини, в КП додається мікрокоманда формату рис. 1.5, *в,* розрядність якої для заданої ГСА є фіксованою. Заповнення полів мікрокоманди

для різних типів вершин, що додаються багато в чому схоже із примусовою адресацією і також приводить до збільшення ємності КП.

Таким чином, при збільшенні складності ГСА зростання апаратурних витрат у схемі мікропрограмного пристрою керування є більш прогнозованим у порівнянні з мікропрограмним автоматом.

Спостережуване сьогодні зростання складності обчислювальних систем веде до збільшення складності алгоритмів, що імплементуються керуючим автоматом. Це висуває підвищені вимоги до КА стосовно швидкодії, яким найкраще задовольняє мікропрограмний автомат. У той же час відносно висока складність і вартість схеми МПА відбивається на відповідних характеристиках обчислювальної системи і може обмежувати область застосування даного класу керуючих автоматів. Це дозволяє вважати питання оптимізації апаратурних витрат у схемі МПА актуальною науковою проблемою.

Для оптимізації апаратурних витрат в логічній схемі мікропрограмного автомата в дисертаційній роботі пропонується новий концептуальний підхід, основна ідея якого полягає в наступному [32, 40, 41].

Нехай МПА Мура заданий ГСА Γ_{1-2}, зображеної на рис. 1.14. Дана ГСА містить 15 станів $a_0 - a_{14}$, для кодування яких достатньо чотирьох двійкових розрядів. Для розглянутого підходу вміст операторних вершин може бути довільним і на рисунку не показаний. Символами $x_1 - x_4$ позначені логічні умови, що відповідають вхідним сигналам мікропрограмного автомата.

Закодуємо стани $a_0 - a_{14}$ унікальними кодами так, як показано на рис. 1.15. Тут у кожній вершині ГСА, яка відповідає окремому стану, зазначений двійковий код стану та відповідний йому десятковий еквівалент, що є інтерпретацією двійкового коду як цілого числа без знака.

Кожна гілка ГСА, відповідна до переходу з одного стану в інший, відзначена деякою операцією. Виконання даної операції над кодом поточного стану дозволяє одержати код стану переходу. Наприклад, перехід зі стану a_2 до стану a_3 виконується за допомогою операції інкремента, перехід зі стану a_8 до

стану a_7 виконується за допомогою операції ділення на 2, а перехід зі стану a_{11} до стану a_{14} виконується за допомогою операції суми за модулем 2 із двійковою константою 1011.

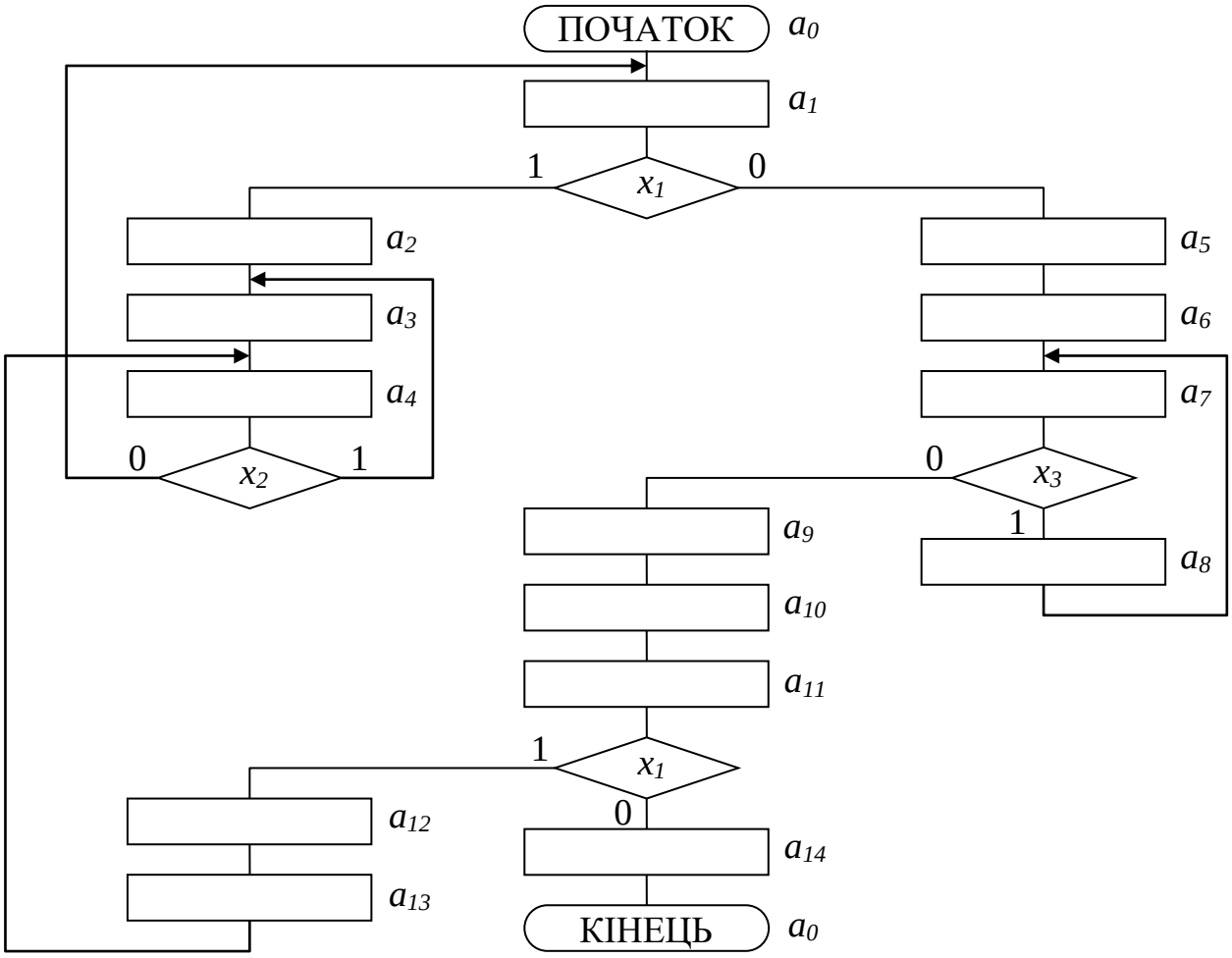


Рисунок 1.14 – Граф-схема алгоритму Γ_{1-2}

В розглянутому прикладі для реалізації всіх переходів ГСА Γ_{1-2} достатньо трьох операцій $O_1 - O_3$, які визначаються виразами (1.17) – (1.19) і перетворюють код поточного стану $K(a^t)$ в код стану переходу $K(a^{t+1})$.

$$O_1 : K(a^{t+1}) = K(a^t) + 1_{10}; \tag{1.17}$$

$$O_2 : K(a^{t+1}) = K(a^t) \oplus 1011_2; \tag{1.18}$$

$$O_3 : K(a^{t+1}) = K(a^t) \div 2_{10}. \tag{1.19}$$

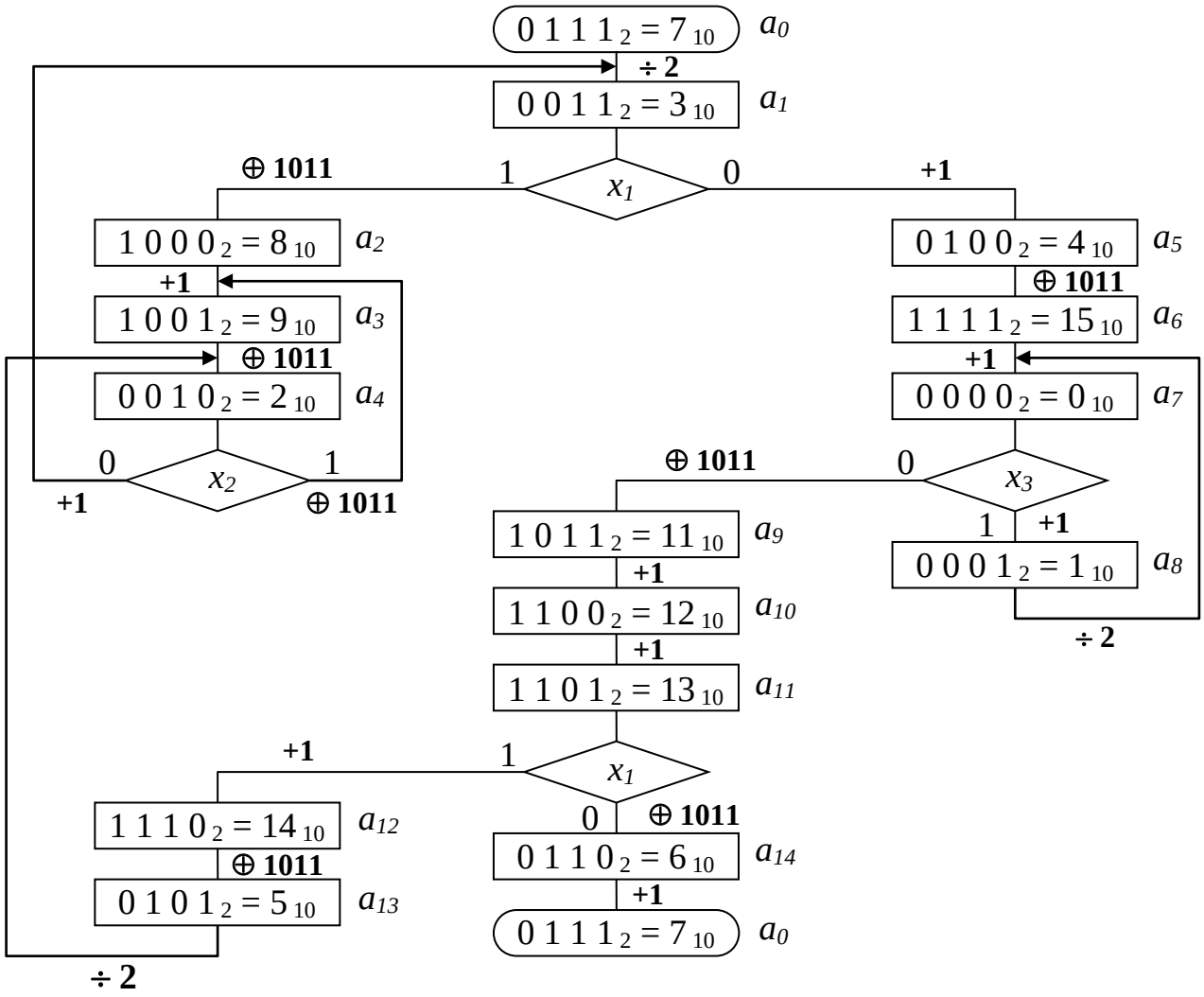


Рисунок 1.15 – Закодована ГСА Γ_{1-2}

Операція O_1 є арифметичною і схемотехнічно може бути реалізована у вигляді інкрементора. Операція O_2 є логічною і реалізується побітово з використанням чотирьох двовходових елементів XOR. Операція O_3 є арифметичною, однак у випадку інтерпретації двійкових кодів станів як беззнакових цілих може бути схемотехнічно реалізована у вигляді блоку логічного зсуву праворуч на один розряд. Таким чином, кожній операції може бути зіставлена комбінаційна схема, що характеризується фіксованим значенням апаратурних витрат при заданій розрядності двійкового коду стану.

Важливою особливістю цих операцій є те, що схемна реалізація кожної з них залежить лише від розрядності коду стану і не залежить від кількості автоматних переходів, що реалізуються за її допомогою в рамках заданої ГСА.

Той факт, що операція O_1 використовується в ГСА Γ_{1-2} для реалізації дев'яти переходів з дев'ятнадцяти, ніяк не позначається на внутрішній архітектурі схеми чотирирозрядного інкрементора. Якщо в деякій ГСА, що містить 1000 станів, операція O_1 використовується для реалізації 500 (або будь-якої іншої кількості) автоматних переходів, її комбінаційна схема завжди буде являти собою стандартний десятирозрядний інкрементор. При цьому очікувано, що апаратурні витрати на реалізацію 500 переходів канонічним способом за системою булевих рівнянь будуть вищі за витрати у десятирозрядному інкрементері, і зі збільшенням кількості реалізованих переходів їх кількість зростатиме.

Розглянутий приклад демонструє, що множина переходів мікропрограмного автомата може бути реалізована не канонічним шляхом за системою булевих рівнянь, а за допомогою кінцевого числа арифметико-логічних операцій. Це дозволяє побудувати схему формування переходів МПА у вигляді набору комбінаційних схем (функціональних блоків), апаратурні витрати на реалізацію яких не залежать від кількості реалізованих ними автоматних переходів при заданій розрядності коду стану. Такий спосіб гіпотетично може сприяти зменшенню апаратурних витрат у схемі, що реалізує функцію переходів МПА, у порівнянні із її канонічною реалізацією за системою булевих рівнянь.

Відомі сьогодні методи оптимізації апаратурних витрат переважно обмежені тим фактом, що код стану автомата розглядається як бітовий вектор, а його перетворення при виконанні мікропрограмних переходів здійснюється за допомогою однієї або кількох систем булевих рівнянь. Недостатність теоретичних досліджень в напрямку неканонічної реалізації функції переходів МПА унеможливлює використання запропонованого вище підходу для оптимізації витрат апаратури при проектуванні цифрових пристроїв. Дане протиріччя дозволяє визначити наукову проблему дисертаційного дослідження, яка полягає у розробці теоретично обґрунтованої методології синтезу мікропрограмних автоматів, спрямованої не зменшення апаратурних витрат в логічній схемі МПА за рахунок використання спеціальних структурних рішень, що дозволяють адаптувати структуру МПА до характеристик алгоритму керування,

імплементованого автоматом. Під методологією будемо розуміти взаємозалежну сукупність ефективних принципів, методик, методів, алгоритмів і рекомендацій, що визначають послідовність обґрунтованих кроків, на кожному з яких задіється певна методика, метод або алгоритм.

Головними складовими методології мають бути наступні її компоненти, що підлягають розробці й дослідженню:

1. Нові структури і математичні моделі МПА, що базуються на оригінальному принципі реалізації мікропрограмних переходів.

2. Загальна методика синтезу розроблених структур мікропрограмних автоматів, у якій в якості критерію оптимізації виступають апаратурні витрати в схемі формування автоматних переходів. Методика повинна являти собою взаємозалежну сукупність методів і алгоритмів, заснованих на строгих математичних умовах або практичних рекомендаціях прийняття рішень.

3. Методи подальшої оптимізації розроблених структур за різними критеріями за рахунок використання існуючих підходів і методів, що застосовуються до мікропрограмних автоматів. Результатом розробки таких методів можуть бути нові структури автоматів, що враховують умови проектування та експлуатації обчислювальних систем.

4. Визначена область ефективного застосування розроблених структур мікропрограмних автоматів.

1.5 Висновки до першого розділу і постановка завдання дисертаційної роботи

1. В першому розділі вирішене наукове завдання аналізу існуючих методів синтезу і оптимізації цифрових автоматів, а також запропоновано концепцію реалізації автоматних переходів шляхом перетворення кодів станів автомата за допомогою кінцевої множини арифметико-логічних операцій.

2. Найбільш загальною математичною моделлю цифрових пристроїв є абстрактний автомат. Його побудова (абстрактний синтез) зводиться до опису

необхідної, можливої і забороненої поведінки шляхом синтезу систем рівнянь без урахування конкретної структури (функціональної схеми) автомата. Даною моделлю описуються автомати без пам'яті (комбінаційні схеми) та автомати Мілі й Мура.

3. Наступним після етапу абстрактного синтезу автомата є етап структурного синтезу, метою якого є побудова логічної схеми автомата з елементів заданого типу. На етапі структурного синтезу найбільш загальною математичною моделлю є структурний автомат, що реалізує закон поведінки абстрактного автомата та являє собою мережу з логічних елементів.

4. Канонічний метод структурного синтезу дозволяє виконати синтез схеми структурного автомата за графом абстрактного автомата. Результатом його застосування є система канонічних булевих рівнянь, що задає залежність вихідних сигналів і сигналів, що подаються на входи запам'ятовувальних елементів, від сигналів, що надходять на входи автомата ззовні, та сигналів з виходів запам'ятовувальних елементів.

5. Керуючий автомат являє собою пристрій, що є схемною імплементацією мікропрограми. Відомі методи схемної імплементації алгоритмів визначають дві основні структури керуючих автоматів: мікропрограмний автомат і мікропрограмний пристрій керування. Можливість реалізації автоматних переходів за один такт робить мікропрограмний автомат більш кращим з погляду швидкодії при використанні в сучасних цифрових обчислювальних системах. Разом з тим, високі апаратурні витрати в логічній схемі мікропрограмного автомата актуалізують наукову проблему їх оптимізації, особливо важливу у світлі зростання складності обчислювальних систем і алгоритмів керування, що імплементуються мікропрограмним автоматом.

6. Сучасні САПР ПЛІС надають можливості автоматичного синтезу мікропрограмних автоматів за їх VHDL-описом. При цьому припустимі різні способи кодування станів, що суттєво впливають на результуючі характеристики схеми автомата. Однак автоматизація процесу кодування станів ускладнює

застосування багатьох відомих методів оптимізації схеми автомата, які передбачають власний підхід до цього процесу.

7. Розбивка системи логічних функцій структурного автомата на підсистеми функцій переходів і функцій виходів дає можливість проводити роздільну оптимізацію даних підсистем з метою задоволення заданим критеріям проектування. Перетворення коду поточного стану автомата в код наступного стану можливе як канонічним способом із використанням системи булевих рівнянь функції переходів, так і неканонічним способом за допомогою кінцевої множини арифметико-логічних операцій, для яких складність схемної реалізації не залежить від кількості реалізованих автоматних переходів.

Метою дисертаційної роботи є розробка, обґрунтування і дослідження теоретичних основ, структур, моделей і методів, спрямованих на зменшення апаратурних витрат в схемі мікропрограмного автомата за рахунок адаптації схеми автомата до характеристик імплементованого алгоритму керування.

Для досягнення мети необхідно вирішити наступні завдання:

– вибір математичного апарата для формалізації запропонованих підходів і методів;

– аналіз і узагальнення конструктивних особливостей відомих структур мікропрограмних автоматів;

– розробка нових структур мікропрограмних автоматів, орієнтованих на зменшення апаратурних витрат у схемі формування переходів;

– модифікація розроблених структур мікропрограмних автоматів із використанням відомих методів оптимізації;

– розробка методології синтезу запропонованих структур мікропрограмних автоматів;

– дослідження залежності апаратурних витрат від параметрів автомата в розроблених структурах мікропрограмного автомата;

– дослідження розроблених структур мікропрограмних автоматів з метою визначення області їх ефективного застосування.

2 ОПЕРАЦІЙНЕ ПЕРЕТВОРЕННЯ КОДІВ СТАНІВ У МІКРОПРОГРАМНОМУ АВТОМАТІ

2.1 Алгебраїчна інтерпретація автоматів

Сучасна теорія автоматів має тісний зв'язок з теорією універсальних алгебр. Алгебраїчний підхід дозволяє використовувати алгебраїчні результати в теорії автоматів, сприяючи встановленню зв'язку теорії автоматів з іншими областями математики. Математичний апарат універсальних алгебр лежить в основі т.зв. *алгебраїчної теорії автоматів*, що вивчає дискретні автомати з абстрактно-алгебраїчної точки зору [3, 78, 85, 107, 116, 129, 136, 137].

Базовим об'єктом вивчення в теорії універсальних алгебр є *алгебраїчна система*, обумовлена як абстрактний об'єкт

$$G = < A, F, P >, \qquad (2.1)$$

що складається з трьох множин: непустої множини A, множини операцій $F = \{F_0, ..., F_\xi, ...\}$, визначених на множині A, і множини предикатів (відношень) $P = \{P_0, ..., P_\eta, ...\}$, заданих на множині A [6, 10, 116, 127, 136]. Під *операцією* тут розуміється функціональне відношення будь-який арності, задане на множині A. При цьому особливо підкреслюється функціональний характер даного відношення, при якому кожному елементу з області визначення зіставляється не більш ніж один елемент в області значень [127, 137, 171]. Під *предикатом* розуміється функція на множині A, значення якої називаються *модальностями* або *ступенями істинності* та лежать в особливій множині, що складається із двох елементів – «істина» та «хибність» [92, 127].

Множину A називають *носієм* (*основою*, *основною множиною*) системи G. Алгебраїчна система G називається *кінцевою*, якщо множина A кінцева [127]. Набір множин $< F, P >$ утворює *сигнатуру* або *структуру* алгебраїчної системи [10]. У випадку кінцевого носія дані множини також кінцеві. Сигнатуру з

порожньою множиною відносин називають *функціональною*, з порожньою множиною операцій – *предикатною* [159].

Алгебраїчна система (2.1) називається *універсальною алгеброю* або просто *алгеброю*, якщо її сигнатура функціональна, і *моделлю* (*реляційною системою*), якщо її сигнатура предикатна [127]. Таким чином, алгеброю є наступний двоелементний об'єкт [92, 127]:

$$G = < A, F > . \qquad (2.2)$$

Іноді носій алгебри представляється декількома сукупностями різнорідних об'єктів, об'єднання яких у єдину множину є неприродним [77]. У цьому випадку виявляється зручним представляти носій у вигляді сімейства непересічних множин. Така структуризація носія дозволяє розглядати алгебри, задані на декількох різних носіях.

Нехай $A = \{A_1, ..., A_k\}$ – множина основ, $F = \{f_1, ..., f_m\}$ – сигнатура, причому $f_i : A_i^1 \times A_i^2 \times ... \times A_i^{k_i} \to A_j$. Тоді система (2.2) називається *багатоосновною* (*k-основною*, *багатосортною*, *гетерогенною*) *алгеброю* [78, 92, 106, 131, 136]. Багатоосновная алгебра має декілька носіїв, а кожна операція сигнатури є відповідністю, що діє із прямого добутку деяких носіїв у деякий носій [131, 136].

Розглянемо алгебраїчну інтерпретацію абстрактного і структурного автоматів, що задаються виразами (1.1) та (1.6) відповідно.

Абстрактний автомат.

Відповідно до виразу (1.1), кінцевий абстрактний автомат представляється сукупністю трьох кінцевих множин – станів, вхідних і вихідних слів, і двох детермінованих функцій – переходів і виходів. У такому вигляді автомат може розглядатися як алгебраїчна система, а через відсутність на множинах яких-небудь відношень більш ніж нульової арності – як *трьохосновна алгебра із двома операціями* [78, 136, 175]. У теорії автоматів представлення автомата у вигляді алгебри дає можливість використовувати алгебраїчні прийоми при вирішенні

таких задач, як декомпозиція автоматів, їх класифікація, опис поведінки та інші [137].

У рамках дисертації будемо називати алгебру, що ототожнюється з абстрактним автоматом (1.1), *абстрактною алгеброю* G_A:

$$G_A =< A_A, F_A >, \tag{2.3}$$

де A_A – носій абстрактної алгебри, що містить усі алфавіти абстрактного автомата:

$$A_A =< A, Z, W >, \tag{2.4}$$

F_A – сигнатура абстрактної алгебри, що включає абстрактні функції переходів і виходів:

$$F_A = \{\delta, \lambda\}. \tag{2.5}$$

Смисл виразів (2.3) – (2.5) полягає в перерахуванні об'єктів, які повинні бути визначені для завдання абстрактного автомата. При цьому спосіб завдання об'єктів не має значення і може бути різним. Наприклад, функція переходів δ може задаватися у вигляді таблиці переходів, графа або матриці [3].

Функції δ та λ реалізують різні відповідності і можуть розглядатися окремо одна від іншої. Це виявляється корисним, наприклад, у тих випадках, коли вихідні реакції автомата невідомі або не представляють інтересу, а об'єктом дослідження є доступна спостереженню еволюція його внутрішніх станів під впливом вхідних сигналів. У роботах [78, 85, 137, 146, 163, 175] згадуються т.зв. *автомати без виходів* (*напівавтомати, X-автомати, редукти*) – об'єкти, що задаються тільки за допомогою множини станів, вхідного алфавіту та функції переходів. Об'єкти, у яких функція переходів відсутня, але присутня функція виходів, належать до класу *автоматів без пам'яті* [87].

Абстрактний автомат без функції виходів може розглядатися як двохосновна алгебра

$$G_\delta = \ < A_\delta, F_\delta >=< \{A, Z\}, \{\delta\} >, \tag{2.6}$$

без функції переходів – як трьохосновна алгебра

$$G_\lambda = \ < A_\lambda, F_\lambda >=< \{A, Z, W\}, \{\lambda\} >. \tag{2.7}$$

Вираз (2.7) справедливий для автомата Мілі. Для автомата Мура необхідно з (2.7) виключити компонент *Z*:

$$G_\lambda = \ < A_\lambda, F_\lambda > = < \{A, W\}, \{\lambda\} > . \qquad (2.8)$$

Будемо називати алгебру (2.6) *абстрактною алгеброю переходів*, алгебри (2.7), (2.8) – *абстрактною алгеброю виходів*. Тоді автомат (1.1) може розглядатися як двійка:

$$U = < G_\delta, G_\lambda >, \qquad (2.9)$$

а ініціальний автомат (1.2) утворюється додаванням у (2.9) виділеного початкового стану $a_П \in A$:

$$S = < U, a_П > = < G_\delta, G_\lambda, a_П > . \qquad (2.10)$$

В абстрактному автоматі функція переходів δ є відповідністю

$$\delta : (D_\delta \subseteq A \times Z) \to A. \qquad (2.11)$$

Будучи представленою у вигляді (2.11), функція δ вважається, у загальному випадку, частково визначеною на множині $A \times Z$. Виходячи з математичного визначення відповідності [171], функція (2.11) є множина кортежів

$$< a_i, z_j, a_k > \in \delta, \qquad (2.12)$$

кожний з яких відповідає одному автоматному переходу зі стану $a_i \in A$ під впливом вхідного сигналу $z_j \in Z$ до стану $a_k \in A$. Застосовуючи до множини кортежів операцію розбивки, можна одержати сімейство непустих підмножин, що попарно не перетинаються, максимально можливе число яких дорівнює числу переходів автомата [37, 45]. Кожну таку підмножину переходів назвемо *частковою функцією переходів* δ_i, розуміючи тут під терміном «*часткова*» її неповне визначення на області визначення D_δ функції переходів δ:

$$\delta_i : (D_{\delta_i} \subseteq D_\delta) \to A. \qquad (2.13)$$

При цьому функція переходів δ є множина підмножин кортежів (2.12), тобто множина часткових функцій переходів [29]:

$$\delta = \{\delta_1, ..., \delta_{N_\delta}\}, \qquad (2.14)$$

де $1 \le N_\delta \le B$, B – число переходів автомата, що дорівнює числу непустих клітин таблиці переходів. Зрозуміло також, що $\bigcup_{i=1}^{N_\delta} \delta_i = \delta$.

Подібна розбивка може бути виконана і для функції виходів, яка у випадку автомата Мілі реалізує відповідність

$$\lambda : (D_\lambda \subseteq A \times Z) \to W, \tag{2.15}$$

що є множиною кортежів виду

$$< a_i, z_j, w_k > \in \lambda, \tag{2.16}$$

у випадку автомата Мура – відповідність

$$\lambda : (D_\lambda \subseteq A) \to W, \tag{2.17}$$

що є множиною кортежів виду

$$< a_i, w_k > \in \lambda \ . \tag{2.18}$$

За аналогією з (2.13) і (2.14), функція (2.15) може розглядатися як сімейство

$$\lambda = \{\lambda_1, ..., \lambda_{N_\lambda}\}, \tag{2.19}$$

часткових функцій виходів виду

$$\lambda_i : (D_{\lambda_i} \subseteq D_\lambda) \to W, \tag{2.20}$$

кожна з яких є підмножиною кортежів виду (2.16) або (2.18).

Якщо вважати, що в автоматі Мілі вихідний сигнал формується при кожному переході зі стану в стан, а в автоматі Мура – у кожному стані, то максимально можливе число N_λ відповідностей виду (2.20) дорівнює для автомата Мілі числу переходів *B*, для автомата Мура – числу станів *M*.

Пов'яжемо з кожною частковою функцією переходів δ_i власну алгебру наступного виду:

$$G_{\delta_i} = < A_{\delta_i}, F_{\delta_i} > = < \{A_{\delta_i}, Z_{\delta_i}\}, \{\delta_i\} > . \tag{2.21}$$

У даному виразі $A_{\delta_i} \subseteq A$ є множина усіх станів, що присутні у кортежах (2.12) часткової абстрактної функції δ_i. Іншими словами, до A_{δ_i} входять ті й тільки ті стани, переходи з яких або переходи в які реалізуються функцією δ_i.

Множина $Z_{\delta_i} \subseteq Z$ є множина вхідних сигналів, що присутні у кортежах (2.12) функції δ_i. Оскільки $A_{\delta_i} \subseteq A$, $Z_{\delta_i} \subseteq Z$ і для будь-яких $a \in A_{\delta_i}$ та $z \in Z_{\delta_i}$ виконується рівність $\delta_i(a, z) = \delta(a, z)$, вираз (2.21) є *підалгеброю* алгебри (2.6) [127]. Будемо називати алгебру (2.21) *абстрактною підалгеброю переходів*. В алгебраїчній теорії автоматів поняттю підалгебри відповідає поняття *підавтомата* [174].

Множині часткових функцій переходів відповідає множина абстрактних підалгебр переходів, яку можна ототожнювати з абстрактною алгеброю переходів (2.6):

$$G_\delta = \{G_{\delta_1}, ..., G_{\delta_{N_\delta}}\}. \qquad (2.22)$$

При цьому компоненти A_δ та F_δ алгебри G_δ формуються шляхом об'єднання відповідних компонентів усіх підалгебр G_{δ_i}: $A_\delta = \bigcup_{i=1}^{N_\delta} A_{\delta_i} = \{\bigcup_{i=1}^{N_\delta} A_{\delta_i}, \bigcup_{i=1}^{N_\delta} Z_{\delta_i}\}$, $F_\delta = \bigcup_{i=1}^{N_\delta} F_{\delta_i} = \{\bigcup_{i=1}^{N_\delta} \delta_i\}$.

Відзначимо, що в той час як часткові функції δ_i утворюють розбивку, тобто $\forall (i, j \in [1, N_\delta], i \neq j): \delta_i \cap \delta_j = \varnothing$, сімейства множин A_{δ_i} та Z_{δ_i} утворюють на множинах A і Z відповідні покриття, що в загальному випадку припускає $\forall (i, j \in [1, N_\delta])$: $A_{\delta_i} \cap A_{\delta_j} \neq \varnothing$ та $Z_{\delta_i} \cap Z_{\delta_j} \neq \varnothing$. Присутність одного і того ж стану $a \in A$ в носіях декількох підалгебр означає, що переходи в даний стан та / або з нього виконуються за допомогою різних часткових функцій. Присутність одного і того ж вхідного сигналу $z \in Z$ в носіях декількох підалгебр говорить про те, що даний сигнал використовується різними частковими функціями δ_i, та у випадку імплементації кожної часткової функції у вигляді окремої схеми повинен подаватися в кожну схему, де він використовується.

За аналогією з (2.22), у вигляді множини відповідних підалгебр представляються абстрактна алгебра виходів (2.7) автомата Мілі та абстрактна алгебра виходів (2.8) автомата Мура:

$$G_\lambda = \{G_{\lambda_1}, ..., G_{\lambda_{N_\lambda}}\}, \quad (2.23)$$

де для автомата Мілі

$$G_{\lambda_i} = <A_{\lambda_i}, F_{\lambda_i}> = <\{A_{\lambda_i}, Z_{\lambda_i}, W_{\lambda_i}\}, \{\lambda_i\}>, \quad (2.24)$$

для автомата Мура –

$$G_{\lambda_i} = <A_{\lambda_i}, F_{\lambda_i}> = <\{A_{\lambda_i}, W_{\lambda_i}\}, \{\lambda_i\}>. \quad (2.25)$$

Тут $A_{\lambda_i} \subseteq A$; $Z_{\lambda_i} \subseteq Z$; $W_{\lambda_i} \subseteq W$; $\forall(i, j \in [1, N_\lambda], i \neq j): \lambda_i \cap \lambda_j = \varnothing$; у загальному випадку $\forall(i, j \in [1, N_\lambda]): A_{\lambda_i} \cap A_{\lambda_j} \neq \varnothing$, $Z_{\lambda_i} \cap Z_{\lambda_j} \neq \varnothing$, $W_{\lambda_i} \cap W_{\lambda_j} \neq \varnothing$. Як і для алгебри (2.22), компоненти A_λ та F_λ формуються шляхом об'єднання відповідних компонентів усіх підалгебр. Наприклад, для автомата Мілі маємо: $A_\lambda = \bigcup_{i=1}^{N_\lambda} A_{\lambda_i} = \{\bigcup_{i=1}^{N_\lambda} A_{\lambda_i}, \bigcup_{i=1}^{N_\lambda} Z_{\lambda_i}, \bigcup_{i=1}^{N_\lambda} W_{\lambda_i}\}$, $F_\lambda = \bigcup_{i=1}^{N_\lambda} F_{\lambda_i} = \{\bigcup_{i=1}^{N_\lambda} \lambda_i\}$.

Таким чином, абстрактний автомат може розглядатися як:

– абстрактна алгебра (2.3) з носієм (2.4) і сигнатурою (2.5);

– двійка, утворена абстрактною алгеброю переходів (2.6) і абстрактною алгеброю виходів (2.7) / (2.8);

– двійка, утворена двома множинами: множиною підалгебр переходів (2.22) і множиною підалгебр виходів (2.23).

Структурний автомат.

Модель (1.6) структурного автомата також є трьохосновною алгеброю, носій A_S якої утворений множинами структурного автомата, а сигнатура F_S – структурною функцією переходів *d* та структурною функцією виходів *l*:

$$A_S = \{X, A, Y\}, \quad (2.26)$$

$$F_S = \{d, l\}. \quad (2.27)$$

Відзначимо, що у виразі (2.26) множина станів *A* відрізняється від однойменної множини у виразі (2.4) тим, що містить не абстрактні, а структурні стани, представлені своїми структурними (двіковими) кодами [87].

Назвемо алгебру, що ототожнюється зі структурним автоматом, *структурною алгеброю* G_S:

$$G_S = < A_S, F_S > = < \{A, X, Y\}, \{d, l\} > . \quad (2.28)$$

Поділ функцій переходів і виходів дає дві окремі алгебри: *структурну алгебру переходів*

$$G_d = < A_d, F_d > = < \{A, X\}, \{d\} > \quad (2.29)$$

і *структурну алгебру виходів* (вираз (2.30) – для автомата Мілі, (2.31) – для автомата Мура):

$$G_l = < A_l, F_l > = < \{A, X, Y\}, \{l\} >, \quad (2.30)$$

$$G_l = < A_l, F_l > = < \{A, Y\}, \{l\} > . \quad (2.31)$$

При цьому структурний автомат (1.6) може розглядатися як двійка:

$$S = < G_d, G_l > . \quad (2.32)$$

У виразі (1.6) структурна функція переходів d є відповідністю

$$d : (D_d \subseteq X \times A) \to A. \quad (2.33)$$

Модель (1.6) використовує представлення усіх елементів множин абстрактного автомата у вигляді слів у структурному (двійковому) алфавіті. Кодування станів, вхідних і вихідних сигналів словами в структурному алфавіті приводить до їхнього представлення у вигляді наборів значень змінних, визначених на структурному алфавіті. Будемо також вважати, що пам'ять структурного автомата будується в базисі тригерів D-типу. Це дозволяє ототожнювати результат структурної функції переходів (структурний код стану переходу) із множиною функцій збудження елементів пам'яті структурного автомата.

У випадку $|X| = L$ та $\lceil log_2 |A| \rceil = R$ вираз (2.33) набуває вигляду

$$d : (D_d \subseteq x_1 \times x_2 \times ... \times x_L \times e_1 \times e_2 \times ... \times e_R) \to e_1 \times e_2 \times ... \times e_R, \quad (2.34)$$

де кожний з елементів x та e є множина букв структурного (двійкового) алфавіту, тобто множина {0, 1} [114, 154]. Фактично, вираз (2.34) є відповідність

$$d : (D_d \subseteq \{0,1\}^{L+R}) \to \{0,1\}^R, \quad (2.35)$$

елементами якої є кортежі, що складаються із $(L + 2R)$ букв $q \in \{0, 1\}$:

$$< q_1, ..., q_{L+2R} > \in d. \quad (2.36)$$

Для автомата Мілі структурна функція виходів l є відповідність

$$l:(D_l \subseteq X \times A) \to Y. \tag{2.37}$$

При $|Y| = N$ вираз (2.37) набуває вигляд

$$l:(D_l \subseteq x_1 \times x_2 \times ... \times x_L \times e_1 \times e_2 \times ... \times e_R) \to y_1 \times y_2 \times ... \times y_N \tag{2.38}$$

або

$$l:(D_l \subseteq \{0,1\}^{L+R}) \to \{0,1\}^N. \tag{2.39}$$

Елементами відповідності (2.38) є кортежі, що складаються із $(L+R+N)$ букв $q \in \{0,1\}$:

$$< q_1, ..., q_{L+R+N} > \in l. \tag{2.40}$$

Для автомата Мура вирази (2.37) – (2.40) матимуть, відповідно, вигляд (2.41) – (2.44):

$$l:(D_l \subseteq A) \to Y, \tag{2.41}$$

$$l:(D_l \subseteq e_1 \times e_2 \times ... \times e_R) \to y_1 \times y_2 \times ... \times y_N, \tag{2.42}$$

$$l:(D_l \subseteq \{0,1\}^R) \to \{0,1\}^N, \tag{2.43}$$

$$< q_1, ..., q_{R+N} > \in l. \tag{2.44}$$

Структурна функція переходів (2.33) і структурні функції виходів (2.37) та (2.41), будучи множинами кортежів (2.36), (2.40) та (2.44) відповідно, можуть бути представлені у вигляді множин *часткових структурних функцій* переходів і виходів:

$$d = \{d_1, ..., d_{N_d}\}, \tag{2.45}$$

$$l = \{l_1, ..., l_{N_l}\}, \tag{2.46}$$

у яких

$$d_i:(D_{d_i} \subseteq D_d) \to A, \tag{2.47}$$

$$l_i:(D_{l_i} \subseteq D_l) \to Y. \tag{2.48}$$

Відзначимо, що у виразі (2.45) максимально можлива кількість часткових функцій N_d визначається числом кортежів (2.36), у виразі (2.46) максимальне значення N_l визначається числом кортежів (2.40) для автомата Мілі і (2.44) для

автомата Мура. Якщо у випадку абстрактного автомата кількість часткових абстрактних функцій переходів $1 \le N_\delta \le B$, а кількість часткових абстрактних функцій виходів $1 \le N_\lambda \le B$ для автомата Мілі і $1 \le N_\lambda \le M$ для автомата Мура, то для значень N_d та N_l інша справа.

Як відомо, кожному символу z_f вхідного алфавіту абстрактного автомата в процесі кодування ставиться у відповідність вектор виду $< x_1, ..., x_L >$, утворений значеннями змінних $x_1 - x_L$, що відповідають однойменним вхідним структурним сигналам. Оскільки спосіб кодування символів множини Z може бути довільним, можливі ситуації, коли у векторі $< x_1, ..., x_L >$, що є відповідним до вхідного символу $z_f \in Z$, деяка змінна x_l може набувати довільне значення [31]. У цьому випадку кожному автоматному переходу, що виконується під впливом вхідного сигналу z_f, у структурній функції переходів d відповідають два вектори виду (2.36), у першому з яких компонент, відповідний до структурного сигналу x_l, дорівнює нулю, у другому – одиниці. Подібна ситуація виникає у структурній функції виходів l автомата Мілі, приводячи до двох векторів виду (2.40), які відрізняється значенням змінної x_l.

У загальному випадку кількість фіктивних змінних у коді символу z_f може бути більше однієї. Такі ситуації часто виникають у процесі структурного синтезу мікропрограмних автоматів, заданих у вигляді ГСА, коли перехід зі стану в стан залежить лише від частини логічних умов, що ототожнюються зі структурними вхідними сигналами. Наприклад, якщо на вхід структурного автомата подаються $L = 10$ вхідних сигналів $x_1 - x_{10}$, а перехід зі стану a_i до стану a_j здійснюється за умовою x_5=1, то сигнали $x_1 - x_4$, $x_6 - x_{10}$ є для даного переходу фіктивними і можуть набувати довільні значення [114]. Таким чином, при дев'ятьох фіктивних сигналах даному автоматному переходу будуть відповідати $2^9 = 512$ різних векторів виду (2.36).

В роботі [31] пропонується кожний структурний вхідний сигнал умовно представляти заданим на тризначній множині {0, 1, –}, де елемент

«–» відповідає довільному значенню з {0, 1}. При цьому у множину Z вхідних сигналів абстрактного автомата вводять спеціальний сигнал z_0, у структурному коді якого все компоненти вектора $< x_1, ..., x_L >$ мають значення «–». Вважається, що даний сигнал завжди присутній на вході автомата незалежно від значень структурних вхідних сигналів і може формально використовуватися в кортежах (2.36) та (2.40), відповідних до безумовних переходів автомата. Іноді структурний код сигналу z_0 позначають логічною одиницею, що підкреслює незалежність сигналу z_0 від структурних вхідних сигналів множини X.

Будемо вважати, що кожна із часткових функцій d_i та l_i є, відповідно, множиною кортежів виду (2.36) та (2.40) із тризначним представленням структурних вхідних сигналів. Тоді, як і для абстрактних часткових функцій, маємо $1 \le N_d \le B$, а також $1 \le N_l \le B$ для автомата Мілі і $1 \le N_l \le M$ для автомата Мура. При $N_d = B$ кожному переходу автомата буде зіставлена окрема часткова функція переходу виду (2.47), а при $N_d = 1$ всі переходи автомата реалізуються однією загальною структурною функцією переходів.

Установимо тепер формальний зв'язок між абстрактною та структурною алгебрами. Однотипність алгебраїчних систем характеризується можливістю встановлення між ними різних відображень, таких як *гомоморфізм*, *ізоморфізм* та інші [112, 127]. Скористаємося поняттям *гомоморфізму автоматів*, яке збігається з поняттям гомоморфізму універсальних алгебр [175]. В. М. Глушков визначає гомоморфізм автоматів в такий спосіб [85]. Гомоморфізм ψ автомата $A(Z_A, A_A, W_A, \delta_A, \lambda_A)$ в автомат $B(Z_B, A_B, W_B, \delta_B, \lambda_B)$ є сукупність трьох однозначних відображень: $\psi_1 : Z_A \to Z_B$, $\psi_2 : A_A \to A_B$, $\psi_3 : W_A \to W_B$, що задовольняють для будь-яких елементів $a \in A_A$ та $z \in Z_A$ співвідношенням:

$$\psi_1(\delta_A(a, z)) = \delta_B(\psi_1(a), \psi_2(z)), \tag{2.49}$$

$$\psi_3(\lambda_A(a, z)) = \lambda_B(\psi_1(a), \psi_2(z)). \tag{2.50}$$

Хоча мікропрограмний автомат може бути заданий як абстрактною алгеброю (2.3), так і структурною алгеброю (2.28), встановлення гомоморфізму

між даними алгебрами, з формальної точки зору, неможливо. Причиною є невідповідність носіїв (2.4) і (2.26), а точніше – різні потужності множин, що утворюють носії. Той факт, що в загальному випадку $(|Z| = F) \neq (|X| = L)$ та $(|W| = G) \neq (|Y| = N)$, не дозволяє встановити між елементами відповідних множин взаємно однозначну відповідність, обов'язкову для гомоморфізму.

Перетворимо структурну алгебру (2.28) у такий спосіб.

1. Нехай носій A_S містить наступні три множини:

– множина $K_S(Z) = \{K_S(z_1), K_S(z_2), ..., K_S(z_F)\}$ структурних кодів вхідних сигналів абстрактного автомата, де кожний елемент $K_S(z_f)$ є вектор $< x_1, ..., x_L >$ значень структурних вхідних сигналів, заданих на трьохелементному алфавіті {0, 1, –};

– множина $K_S(A) = \{K_S(a_1), K_S(a_2), ..., K_S(a_M)\}$ структурних кодів станів абстрактного автомата, заданих векторами $< e_1, ..., e_R >$, $e_r \in \{0, 1\}$;

– множина $K_S(W) = \{K_S(w_1), K_S(w_2), ..., K_S(w_G)\}$ структурних кодів вихідних сигналів абстрактного автомата, заданих векторами $< y_1, ..., y_N >$, $y_n \in \{0, 1\}$.

2. Структурну функцію переходів *d*, що утворює сигнатуру F_S, визначимо виразом (2.51). Структурну функцію виходів *l* автомата Мілі визначимо виразом (2.52), автомата Мура – виразом (2.53).

$$d:(D_d \subseteq K_S(Z) \times K_S(A)) \to K_S(A), \quad (2.51)$$

$$l:(D_l \subseteq K_S(Z) \times K_S(A)) \to K_S(W), \quad (2.52)$$

$$l:(D_l \subseteq K_S(A)) \to K_S(W)). \quad (2.53)$$

Тоді структурна алгебра (2.28) набуває наступного вигляду:

$$G_S = < A_S, F_S > = < \{K_S(Z), K_S(A), K_S(W)\}, \{d, l\} >. \quad (2.54)$$

Відзначимо, що структурна алгебра (2.54) сама собою не припускає якихось конкретних способів вибору елементів носіїв і завдання структурних функцій переходів і виходів, а лише перераховує ті елементи, які повинні бути визначені

для опису структурного автомата. Способи визначення її компонентів залежать від використовуваного методу структурного синтезу.

Структурні алгебри переходів і виходів визначаються аналогічно (2.29) – (2.31) у такий спосіб:

$$G_d = <A_d, F_d> = <\{K_S(A), K_S(Z)\}, \{d\}>, \quad (2.55)$$

$$G_l = <A_l, F_l> = <\{K_S(A), K_S(Z), K_S(W)\}, \{l\}>, \quad (2.56)$$

$$G_l = <A_l, F_l> = <\{K_S(A), K_S(W)\}, \{l\}>. \quad (2.57)$$

Структурна алгебра (2.54) відрізняється від (2.28) тим, що формальними аргументами і значеннями її операцій є не змінні структурні сигнали, а елементи носіїв абстрактної алгебри, представлені у вигляді векторів елементарних структурних сигналів. Це дає можливість *встановити гомоморфізм* абстрактної алгебри (2.3) у структурну алгебру (2.54). Оскільки між носіями даних алгебр можуть бути встановлені однозначні відповідності $K_S(A) \to A$, $K_S(Z) \to Z$, $K_S(W) \to W$, гомоморфізм алгебр (2.3) та (2.54) є *ізоморфізмом* (взаємно однозначним гомоморфізмом, [85]). Очевидно, що існують ізоморфізми між алгебрами переходів (2.6) та (2.55), а також між алгебрами виходів: (2.7) та (2.56) для автомата Мілі, (2.8) та (2.57) для автомата Мура.

Функції (2.51) – (2.53) є множини кортежів (2.58) – (2.60) відповідно.

$$< K_S(a_i), K_S(z_j), K_S(a_k) > \in d, \quad (2.58)$$

$$< K_S(a_i), K_S(z_j), K_S(w_k) > \in l, \quad (2.59)$$

$$< K_S(a_i), K_S(w_k) > \in l. \quad (2.60)$$

Структурна функція переходів (2.51) може бути представлена у вигляді сімейства часткових структурних функцій переходів (2.45), структурні функції виходів (2.52) та (2.53) – у вигляді сімейств часткових структурних функцій виходів (2.46), причому

$$d_i : (D_{d_i} \subseteq D_d) \to K_S(A), \quad (2.61)$$

$$l_i : (D_{l_i} \subseteq D_l) \to K_S(W). \quad (2.62)$$

Ізоморфізм абстрактної алгебри (2.3) і структурної алгебри (2.54) встановлює в тому числі відображення між елементами відповідних носіїв даних алгебр. Зіставляючи кожному компоненту будь-якого кортежу (2.12) відповідний йому елемент із носіїв структурної алгебри, можна одержати тотожний кортеж (2.58). Так само будь-якому кортежу (2.16) може бути поставлений у відповідність кортеж (2.59), будь-якому кортежу (2.18) – кортеж (2.60). Отже, до кожної часткової функції переходів δ_i абстрактного автомата, що задається виразом (2.13) і є деякою підмножиною множини кортежів (2.12), може бути поставлена у відповідність часткова функція переходів d_i, що задається виразом (2.61) і є підмножиною множини кортежів (2.58). Аналогічно, до кожної часткової функції виходів λ_i абстрактного автомата, що задається виразом (2.20), може бути поставлена у відповідність часткова функція виходів l_i, що задається виразом (2.62).

Таким чином, можна стверджувати, що ізоморфізм абстрактної та структурної алгебр можливий при будь-якій розбивці функцій абстрактного автомата на множини часткових функцій. Це справедливо як для алгебр переходів (2.6) та (2.55), так і для алгебр виходів (2.7) / (2.8) та (2.56) / (2.57) відповідно.

Ґрунтуючись на структурній алгебрі (2.54), визначимо, за аналогією з виразами (2.21), (2.24) та (2.25), структурні підалгебри переходів і виходів.

Структурною підалгеброю переходів будемо називати алгебру виду

$$G_{d_i} = <A_{d_i}, F_{d_i}> = <\{K_S(A_{d_i}), K_S(Z_{d_i})\}, \{d_i\}>. \qquad (2.63)$$

Тут $K_S(A_{d_i}) \subseteq K_S(A)$ – множина структурних кодів станів, що присутні у кортежах (2.58) часткової структурної функції d_i; $K_S(Z_{d_i}) \subseteq K_S(Z)$ – множина структурних кодів вхідних сигналів, що використовуються для кодування вхідних сигналів у кортежах (2.58) функції d_i. Структурну алгебру переходів (2.54), за аналогією з (2.22), представимо у вигляді множини структурних підалгебр переходів:

$$G_d = \{G_{d_1}, ..., G_{d_{N_d}}\}. \qquad (2.64)$$

Структурною підалгеброю виходів автомата Мілі будемо називати алгебру

$$G_{l_i} = \langle A_{l_i}, F_{l_i} \rangle = \langle \{K_S(A_{l_i}), K_S(Z_{l_i}), \mathrm{K_S}(W_{l_i})\}, \{l_i\} \rangle, \qquad (2.65)$$

де множини $K_S(A_{l_i}) \subseteq K_S(A)$, $K_S(Z_{l_i}) \subseteq K_S(Z)$, $K_S(W_{l_i}) \subseteq K_S(W)$ містять ті елементи відповідних множин, які присутні в кортежах (2.59) часткової функції l_i.

Структурною підалгеброю виходів автомата Мура, за аналогією з (2.25), будемо називати наступну алгебру:

$$G_{l_i} = \langle A_{l_i}, F_{l_i} \rangle = \langle \{K_S(A_{l_i}), K_S(W_{l_i})\}, \{l_i\} \rangle. \qquad (2.66)$$

При цьому структурну алгебру виходів, за аналогією з абстрактною алгеброю виходів (2.23), будемо представляти у вигляді множини своїх структурних підалгебр:

$$G_l = \{G_{l_1}, ..., G_{l_{N_l}}\}. \qquad (2.67)$$

Як і у випадку абстрактного автомата, компоненти структурної алгебри переходів (2.64) і структурної алгебри виходів (2.67) формуються шляхом об'єднання елементів відповідних компонентів усіх своїх підалгебр.

Таким чином, для структурного автомата будемо вважати припустимими наступні форми представлення:

– у вигляді структурної алгебри (2.54);

– у вигляді двох алгебр: структурної алгебри переходів (2.55) і структурної алгебри виходів (2.56) / (2.57);

– двох множин: множини структурних підалгебр переходів (2.64) і множини структурних підалгебр виходів (2.67).

2.2 Алгебраїчна інтерпретація автомата на лічильнику

Абстрактна алгебра (2.3) і структурна алгебра (2.54) описують поведінку автомата як перетворювача дискретної інформації. У структурній алгебрі переходів (2.55) функція переходів *d* формально є множиною кортежів (2.58); у структурних алгебрах (2.56) та (2.57) функції виходів *l* є множинами кортежів

(2.59) і (2.60) відповідно. Представлення функції (переходів або виходів) структурного автомата у вигляді множини кортежів аналогічно табличному представленню функції і не визначає однозначно інші способи її представлення. Одним зі способів представлення функцій автомата є їх схемна імплементація, обумовлена структурою автомата разом з відповідним методом синтезу.

Розглянемо окремо функцію переходів автомата. Відповідно до запропонованого В. М. Глушковим канонічного методу структурного синтезу функція переходів представляється у векторному вигляді і задається системою канонічних скалярних рівнянь [87]. Система скалярних рівнянь будується відомим способом [31, 87], при якому для кожної пари $< K_S(a_i), K_S(z_j) >$ кортежу (2.58), що відповідає одному переходу автомата, формується терм, присутність якого в тому або іншому рівнянні системи визначається елементом $K_S(a_k)$ даного кортежу.

Максимально можлива кількість різних термів у системі рівнянь функції переходів дорівнює числу B переходів автомата і зростає зі збільшенням B. Як наслідок, зростають і витрати апаратури в комбінаційній схемі, що реалізує систему канонічних рівнянь функції переходів. Деякому зниженню апаратурних витрат можуть сприяти відомі методи мінімізації систем булевих рівнянь. Однак застосування даних методів не гарантує задоволення обмежень по витратах апаратури: черговий терм, що додається в те або інше рівняння, може виявитися для цього рівняння простою незайвою імплікантою [124], яка не піддається операціям поглинання і склеювання. Крім того, для автоматів, що зустрічаються на практиці, система функцій переходів складається з десятків і сотень функцій від десятків і сотень змінних, що робить непридатними класичні методи мінімізації булевих функцій [31].

У структурній теорії автоматів відомий клас пристроїв, які називають автоматами на лічильнику або С-автоматами (від англ. «counter» – лічильник) [25, 36, 69]. У їхній основі лежить принцип спеціального кодування станів, при якому на множині станів заданої ГСА виділяються т.зв. лінійні послідовності станів, всередині яких коди станів задаються в природньому (послідовному) порядку.

Дамо алгебраїчну інтерпретацію автомата на лічильнику. Нехай задані довільний абстрактний автомат, що визначається алгеброю (2.3), та відповідний йому автомат на лічильнику, що визначається алгеброю (2.54). Функції переходів даних автоматів утворюють сигнатури абстрактної алгебри переходів (2.6) і структурної алгебри переходів (2.55) відповідно. У силу ізоморфізму алгебр (2.3) та (2.54), алгебри переходів (2.6) та (2.55) також ізоморфні:

$$G_\delta \leftrightarrow G_S. \tag{2.68}$$

Той факт, що в С-автоматі частина переходів реалізується із використанням інкрементного лічильника, частина – у вигляді системи канонічних рівнянь, дозволяє представити структурну функцію переходів d у вигляді двох часткових функцій d_1 та d_2. Кожна часткова функція може розглядатися як сигнатура окремої структурної підалгебри переходів:

$$G_{d_1} = \ <A_{d_1}, F_{d_1}> = <\{K_S(A_{d_1}), K_S(Z_{d_1})\}, \{d_1\}>; \tag{2.69}$$

$$G_{d_2} = \ <A_{d_2}, F_{d_2}> = <\{K_S(A_{d_2}), K_S(Z_{d_2})\}, \{d_2\}>. \tag{2.70}$$

Представлення функції d у вигляді двох часткових функцій припускає аналогічне представлення абстрактної функції переходів δ. Часткові абстрактні функції переходів δ_1 та δ_2 утворюють сигнатури двох абстрактних підалгебр переходів:

$$G_{\delta_1} = <A_{\delta_1}, F_{\delta_1}> = <\{A_{\delta_1}, Z_{\delta_1}\}, \{\delta_1\}>; \tag{2.71}$$

$$G_{\delta_2} = <A_{\delta_2}, F_{\delta_2}> = <\{A_{\delta_2}, Z_{\delta_2}\}, \{\delta_2\}>. \tag{2.72}$$

З ізоморфізму (2.68) випливають ізоморфізми відповідних підалгебр:

$$G_{\delta_1} \leftrightarrow G_{d_1}; \tag{2.73}$$

$$G_{\delta_2} \leftrightarrow G_{d_2}. \tag{2.74}$$

Кожна часткова функція переходів (абстрактна або структурна), розглянута як елемент сигнатури деякої алгебри, формально визначена лише на тих наборах аргументів, які зустрічаються в її кортежах. На інших наборах аргументів часткова функція вважається невизначеною через відсутність відповідних автоматних переходів. Однак з погляду структурної теорії автоматів функція

переходів вважається визначеною на всіх наборах своїх аргументів, причому на тих наборах, які відсутні в її кортежах, значення функції вважаються довільними [87]. Це дозволяє, за потреби, довизначати функцію довільним чином, одержуючи для неї безліч схемних реалізацій.

Структурна схема С-автомата, у якій підкреслене представлення структурної функції переходів у вигляді двох часткових функцій, наведена на рис. 2.1.

Тут схема $КС_{d_1}$ реалізує часткову структурну функцію переходів d_1 і становить собою *R*-розрядний інкрементор. Структурний код *T* поточного стану, представлений набором структурних змінних $<T_1,...,T_R>$, у якому $T_r \in \{0,1\}$, інтерпретується на вході $КС_{d_1}$ як *R*-розрядне беззнакове ціле, представлене у структурному (двійковому) алфавіті. На виході інкрементора формується структурний код стану переходу, який позначений символом d_1 та ототожнюється зі значенням однойменної часткової функції переходів.

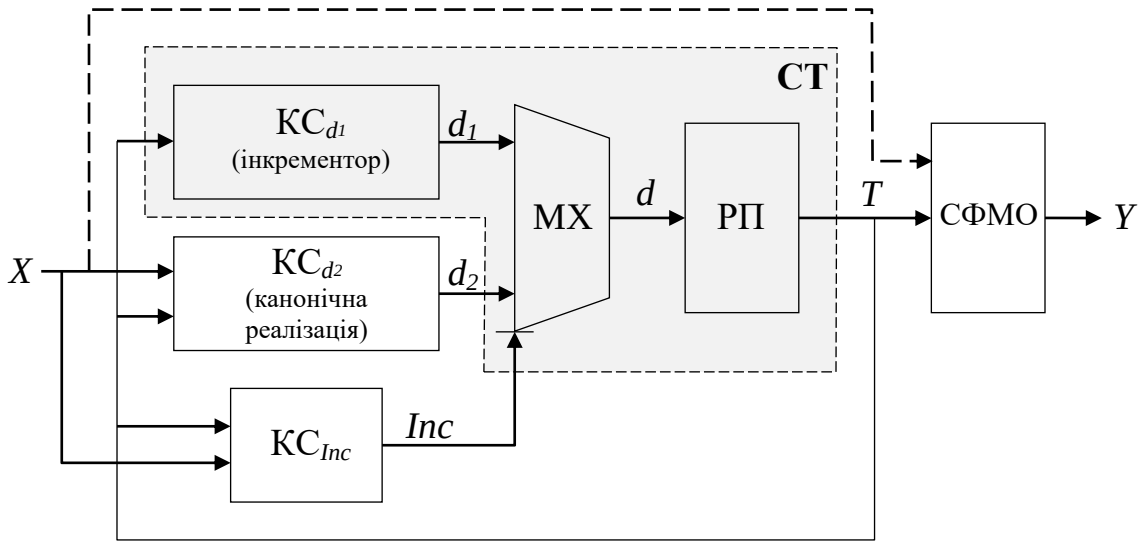


Рисунок 2.1 – Структурна схема С-автомата

Часткова функція d_2 представлена комбінаційною схемою $КС_{d_2}$. Дана схема синтезується за відомою методикою і реалізує систему канонічних рівнянь, відповідну до функції d_2. На вхід $КС_{d_2}$ надходять структурний код поточного

стану $T =< T_1,...,T_R >$ та структурний код вхідного слова, представлений набором $X =< x_1,...,x_L >$, у якому $x_l \in \{0, 1, -\}$. Структурний код стану переходу, що формується на виході $КС_{d_2}$, позначений символом d_2 і ототожнюється зі значенням однойменної часткової функції переходів.

Оскільки кожна часткова структурна функція переходів, представлена у вигляді логічної схеми, визначена на всіх наборах аргументів, але при цьому окремий автоматний перехід реалізується завжди якоюсь однією частковою функцією, на етапі структурного синтезу виникає задача організації вибору потрібної часткової функції при кожному переході автомата. В С-автоматі ця задача вирішується мультиплексуванням одночасно формованих значень двох часткових функцій. На рис. 2.1 схема $КС_{Inc}$ виробляє сигнал *Inc*, що керує мультиплексором МХ. Структурний код d стану переходу, що формується на виході мультиплексора, ототожнюється зі значенням структурної функції переходів автомата та надходить у регістр пам'яті РП. Відзначимо, що в структурі С-автомата, наведеній у роботі [36], блокам $КС_{d_1}$, МХ та РП відповідає інкрементний лічильник СТ.

Розглянутий спосіб вибору часткової функції переходів в С-автоматі дозволяє формально представити структурну функцію переходів d у наступному вигляді:

$$d = d\{d_1, d_2, Inc\}, \tag{2.75}$$

де $d_1 = d_1\{T\}$, $d_2 = d_2\{T, X\}$, $Inc = Inc\{T, X\}$. Оскільки вибір часткової функції здійснюється у відповідності зі значенням логічного сигналу *Inc*, функція (2.75) може бути представлена в інтервальному вигляді:

$$d = \begin{cases} d_1, Inc = 0; \\ d_2, Inc = 1. \end{cases} \tag{2.76}$$

При збільшенні числа переходів автомата, реалізованих частковою функцією d_2, будуть зростати витрати апаратури в схемах $КС_{d_2}$ та $КС_{Inc}$. При збільшенні числа переходів, реалізованих частковою функцією d_1, будуть

зростати витрати апаратури лише в схемі $КС_{Inc}$, у той час як у схемі $КС_{d_1}$ вони будуть залишатися незмінними (при збереженні розрядності R коду стану). Це приводить у загальному випадку до зниження апаратурних витрат у схемі формування переходів С-автомата в порівнянні з канонічною реалізацією усієї множини переходів. Чим більша частина переходів реалізується частковою функцією d_1, тем значніше економія апаратурних витрат. У зв'язку із цим С-автомати рекомендується використовувати для лінійних ГСА, у яких не менш ніж 75 % вершин є операторними [36].

Відзначимо наступне. Незважаючи на те, що кожна часткова функція реалізує власну підмножину переходів, один і той самий стан автомата може зустрічатися в кортежах декількох часткових функцій. Наприклад, перехід до стану a_i може виконуватися за допомогою часткової функції δ_1, перехід із цього стану – за допомогою часткової функції δ_2. Якщо ізоморфізми (2.73) та (2.74) задавати незалежно один від одного, можливі ситуації, коли той самий структурний код привласнений двом різним станам, або одному стану привласнені два різні структурні коди. Сказане стосується також і структурних кодів вхідних сигналів. Подібні неоднозначності унеможливлюють подальший синтез логічної схеми автомата.

Вирішення даної проблеми полягає у *взаємозалежному завданні ізоморфізмів* (2.73) і (2.74), при якому:

– кожний стан a_i отримує єдиний і унікальний структурний код $K_S(a_i)$ у відповідних носіях усіх структурних підалгебр, у яких він зустрічається;

– кожний вхідний сигнал z_j отримує єдиний і унікальний структурний код $K_S(z_j)$ у відповідних носіях усіх структурних підалгебр, у яких він зустрічається.

В алгебраїчній формі дані вимоги можуть бути виражені вимогою існування ізоморфізму (2.68). Даний ізоморфізм встановлює в тому числі взаємно однозначні відповідності між кожним станом $a_i \in A$ та його структурним кодом $K_S(a_i) \in K_S(A)$, а також між кожним вхідним сигналом $z_j \in Z$ та його

структурним кодом $K_S(z_j) \in K_S(Z)$. У результаті за кожним станом і вхідним сигналом закріплюється єдиний структурний код незалежно від кількості заданих підалгебр.

Відзначимо, що існування ізоморфізму (2.68) саме собою не визначає однозначно ізоморфізми (2.73) та (2.74). Оскільки при існуванні ізоморфізму (2.68) функція переходів може бути розбита на часткові функції в загальному випадку багатьма способами, може існувати безліч способів завдання ізоморфізмів (2.73), (2.74) при заданому ізоморфізмі (2.68). Однак, у кожному разі, для забезпечення можливості схемної реалізації функції переходів ізоморфізми (2.73) та (2.74) повинні бути задані таким чином, щоб при цьому існував ізоморфізм (2.68).

Застосовувана в С-автоматі операція інкременту є арифметичною операцією і визначена для скалярних величин. У процесі синтезу С-автомата на етапі формування лінійних послідовностей станів кожний стан $a_i \in A_{\delta_1}$ отримує унікальний код $K_I(a_i)$ із множини натуральних чисел. При цьому не має значення, у якій системі числення вказувати такий код – важливо лише його числове, скалярне значення. Будемо поки називати подібні числові коди, незалежно від використовуваного діапазону їх значень, «скалярними» кодами станів. «Скалярні» коди призначаються всім станам, переходи в які або переходи з яких виконуються за допомогою операції інкременту. Таким чином, результатом кодування всіх станів множини A_{δ_1} є множина «скалярних» кодів $K_I(A_{\delta_1})$. У той же час стани, відсутні в кортежах функції δ_1, «скалярних» кодів не отримують.

Позначимо операцію інкременту символом O_{Inc}. Помітимо, що множину $K_I(A_{\delta_1})$ разом з операцією O_{Inc} можна розглядати як алгебру

$$G_I = < A_I, F_I > = < K_I(A_{\delta_1}), O_{Inc} >. \qquad (2.77)$$

Тут операція O_{Inc} формально представляється множиною кортежів виду $< K_I(a_i), K_I(a_j) > \in O_{Inc}$. Компонент $K_I(a_i) \in K_I(A_{\delta_1})$ є «скалярним» кодом

вихідного стану, компонент $K_I(a_j) \in K_I(A_{\delta_1})$ – «скалярним» кодом стану переходу. Очевидно, що подібний кортеж не містить інформації про вхідний сигнал, під впливом якого виконується перехід, і в такому виді не може бути поставлений у відповідність кортежам часткових функцій δ_1 та d_1.

Додамо в алгебру (2.77) додатковий носій Z_{δ_1}, ідентичний однойменному носієві з абстрактної підалгебри G_{δ_1}. В результаті отримаємо двохосновну алгебру

$$G_I = < A_I, F_I > = < \{K_I(A_{\delta_1}), Z_{\delta_1}\}, \{O_{Inc}\} >, \tag{2.78}$$

у якій операція O_{Inc} задається множиною кортежів виду

$$< K_I(a_i), z_j, K_I(a_k) > \in O_{Inc}, \tag{2.79}$$

де $K_I(a_i), K_I(a_j) \in K_I(A_{\delta_1})$, $z_j \in Z_{\delta_1}$.

Тепер, враховуючи можливість встановлення попарних взаємно однозначних відповідностей між носіями A_{δ_1}, $K_I(A_{\delta_1})$ та $K_S(A_{d_1})$, а також між носіями Z_{δ_1} та $K_S(Z_{d_1})$, кожному кортежу виду (2.12) у функції δ_1 і відповідному йому кортежу виду (2.58) у функції d_1 може бути поставлений у відповідність єдиний кортеж виду (2.79). Це дозволяє задати операцію O_{Inc} такою множиною кортежів (2.79), щоб у рамках алгебри (2.78) вона виконувала перетворення «скалярних» кодів станів за тим же законом, що й функції δ_1 та d_1 у своїх підалгебрах, реалізуючи, по суті, ті ж автоматні переходи, що й функції δ_1 та d_1.

Усе це дозволяє говорити про взаємний ізоморфізм трьох алгебр: алгебри (2.78), абстрактної підалгебри G_{δ_1} та структурної підалгебри G_{d_1}:

$$G_{\delta_1} \leftrightarrow G_I \leftrightarrow G_{d_1}. \tag{2.80}$$

Даний вираз містить три ізоморфізми: $G_{\delta_1} \leftrightarrow G_{d_1}$, $G_{\delta_1} \leftrightarrow G_I$ та $G_I \leftrightarrow G_{d_1}$. Однак, враховуючи вимогу їх одночасного існування, будемо для стислості розглядати дані ізоморфізми як єдиний ізоморфізм виду (2.80).

Відзначимо також наступне. Якщо структурні коди станів, що зустрічаються в кортежах часткової функції d_2, можуть задаватися довільно (із припустимої множини структурних кодів), то у випадку функції d_1 структурні коди повинні задаватися таким чином, щоб уможливити їхнє перетворення за допомогою операції інкременту.

Завдання «скалярних» кодів станів у природньому (інкрементному) порядку, з математичної точки зору, є завдання транзитивного замикання для відношення інкременту на деякій множині скалярних чисел. На довільній множині структурних кодів, для яких невідома їхня «скалярна» інтерпретація, відношення інкременту не може бути визначене. Із цієї причини вибір «скалярних» кодів станів множини A_{δ_1} повинен передувати вибору їх структурних кодів. У свою чергу, завдання структурних кодів станів множини A_{δ_2} виконується лише після завдання структурних кодів станів множини A_{δ_1}, а синтез схем КС_{d_2} та КС_{Inc} можливий лише після завдання структурних кодів усіх станів.

Таким чином, у процесі синтезу С-автомата має місце наступний порядок завдання алгебр переходів:

1. Завдання абстрактних підалгебр переходів (2.71) та (2.72) у відповідності зі сформованими лінійними послідовностями станів.

2. Завдання алгебри (2.78) відповідно до підалгебри (2.71).

3. Завдання структурної підалгебри (2.69) відповідно до алгебри (2.78).

4. Завдання структурної підалгебри (2.70) з урахуванням сформованої підалгебри (2.69).

Зважаючи на те, що алгебра (2.78) відіграє в С-автоматі допоміжну роль при побудові структурної алгебри переходів за заданою абстрактною алгеброю переходів, назвемо алгебру G_I *проміжною алгеброю переходів*, розуміючи під нижнім індексом «*I*» термін «*intermediate*» – «проміжний» [16, 19]. «Скалярні» коди станів, що належать носієві проміжної алгебри, будемо називати

проміжними кодами станів. Операцію, що утворює сигнатуру проміжної алгебри переходів, будемо називати *операцією переходів* (ОП) [40, 48].

В алгебраїчній формі еквівалентність абстрактного автомата та відповідного йому структурного С-автомата виражається існуванням наступних ізоморфізмів:

1. Ізоморфізм (2.80) проміжної алгебри G_I з відповідними їй абстрактною і структурною підалгебрами G_{δ_1} та G_{d_1}. Даний ізоморфізм забезпечує реалізацію частини переходів автомата за допомогою операції інкременту.

2. Ізоморфізм (2.68) абстрактної і структурної алгебр переходів. Даний ізоморфізм забезпечується спільним завданням ізоморфізмів (2.73) та (2.74), при якому зберігаються унікальність і однозначність структурних кодів станів і вхідних сигналів незалежно від способу розбивки функції переходів на часткові функції.

Якщо хоча б один з ізоморфізмів (2.80) та (2.68) не може бути заданий, то не може бути побудований С-автомат, еквівалентний заданому абстрактному автомату. Таким чином, кожний з ізоморфізмів є необхідною умовою існування С-автомата, а достатньою умовою є одночасне існування обох ізоморфізмів. Зведемо ізоморфізми (2.80) та (2.68) у систему (2.81), яку назвемо *необхідною і достатньою умовою існування автомата на лічильнику*:

$$\begin{cases} G_{\delta_1} \leftrightarrow G_I \leftrightarrow G_{d_1}; \\ G_{\delta} \leftrightarrow G_d. \end{cases} \tag{2.81}$$

Процес побудови системи (2.81) припускає, серед іншого, виконання перших трьох етапів структурного синтезу С-автомата, наведених у [36]:

– формування лінійних послідовностей станів;

– кодування станів;

– формування таблиці переходів автомата.

Після завдання системи (2.81) йде завершальний етап структурного синтезу – реалізація в заданому елементному базисі логічної схеми автомата (комбінаційних схем $КС_{d_2}$, $КС_{Inc}$ та схеми формування мікрооперацій) у

відповідності зі сформованою множиною структурних кодів станів і розбивкою функції переходів на часткові функції.

Таким чином, метод структурного синтезу С-автомата може бути скорочено представлений двома етапами:

1. Завдання системи ізоморфізмів (2.81).

2. Синтез логічної схеми автомата в заданому елементному базисі.

Назвемо етап завдання системи ізоморфізмів (2.81) *алгебраїчним синтезом функції переходів С-автомата*. Результатом виконання даного етапу слід вважати структурну алгебру переходів та її підалгебри. При цьому алгебраїчний синтез функції переходів не торкається функції виходів автомата, яка, у випадку автомата на лічильнику, може бути реалізована будь-яким зручним способом з урахуванням обраних структурних кодів станів і вхідних сигналів.

Представлення функції переходів С-автомата у вигляді двох часткових функцій обумовлене використанням проміжної алгебри переходів і відбиває поєднання в одній структурі двох різних способів реалізації переходів автомата – канонічного та інкрементного. Для канонічної структури мікропрограмного автомата розбивка функції переходів на часткові функції не має сенсу, оскільки в цьому випадку кожна часткова функція буде реалізовувати лише деякий фрагмент єдиної системи канонічних рівнянь, а необхідна при цьому реалізація вибору часткової функції приведе до збільшення апаратурних витрат у схемі пристрою.

2.3 Проміжні алгебри переходів

Уведемо для автомата на лічильнику ряд узагальнень [16, 17].

Узагальнення 1. *У сигнатурі проміжної алгебри переходів може використовуватися операція, відмінна від операції інкременту.*

Нехай мікропрограмний автомат заданий ГСА Γ_{2-1}, яка позначена станами автомата Мура (рис. 2.2). Оскільки вміст операторних вершин не має стосунку до системи переходів автомата, у даному і наступних прикладах він може бути довільним і деталізуватися не буде.

Задамо по ГСА Γ_{2-1} автомат, подібний С-автомату, у якому в якості операції проміжної алгебри переходів замість операції інкременту використовується операція декременту. Як і в С-автоматі, у даному автоматі (назвемо його D-автоматом) частина переходів реалізується за допомогою операції проміжної алгебри, частина – канонічним способом. Отже, маємо аналогічне С-автомату представлення абстрактної та структурної функцій переходів у вигляді пар часткових функцій.

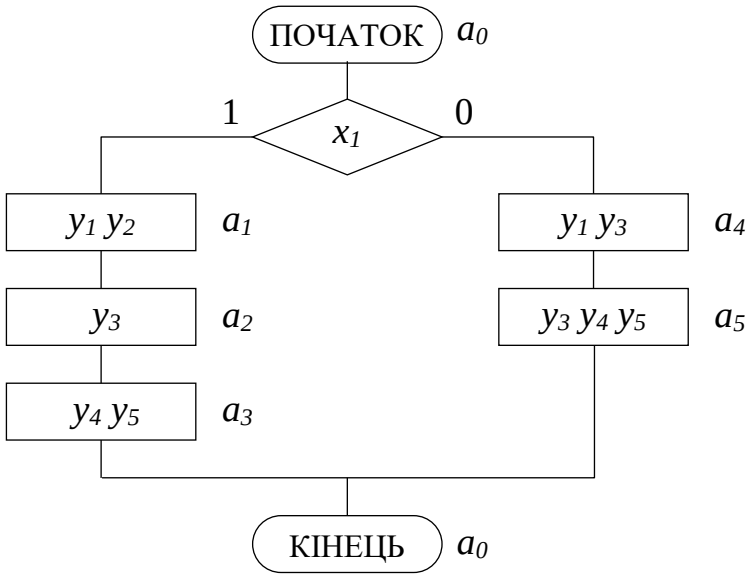


Рисунок 2.2 – Граф-схема алгоритму Γ_{2-1}

Під завданням D-автомата в цьому випадку будемо розуміти його алгебраїчний синтез, тобто завдання аналогічної (2.81) системи ізоморфізмів, у якій сигнатура проміжної алгебри утворена операцією декременту над беззнаковими цілими числами. Для цього скористаємося відомою методикою синтезу С-автомата, але коди станів всередині лінійних послідовностей станів будемо задавати в «декрементному» порядку.

У ГСА Γ_{2-1} присутня єдина логічна умова x_1, якій в структурному автоматі відповідає однойменний структурний вхідний сигнал, що утворює множину X. В силу зроблених у п. 2.1 припущень сигнал x_1 може набувати значення з множини $\{1,\ 0,\ -\}$. Таким чином, можуть бути сформовані три різні вектори виду $< x_1, ..., x_L >$: $<1>$, $<0>$ та $<->$. В абстрактному автоматі даним векторам

відповідають три вхідні сигнали $z_0 - z_2$, для яких вектори є структурними кодами. Нехай z_0 відповідає безумовному переходу, z_1 – переходу за умовою $x_1 = 1$, z_2 – переходу за умовою $x_1 = 0$. Тоді

$$K_S(z_0) = <->,\ K_S(z_1) = <1>,\ K_S(z_2) = <0>,$$

$$K_S(Z) = \{K_S(z_0), K_S(z_1), K_S(z_2)\} = \{<->, <1>, <0>\}\ .$$

Задамо абстрактний автомат, що відповідає ГСА Γ_{2-1}, абстрактною алгеброю переходів виду (2.6):

$$\begin{cases} G_\delta = <\{A, Z\}, \{\delta\}>; \\ A = \{a_0, a_1, a_2, a_3, a_4, a_5\}; \\ Z = \{z_0, z_1, z_2\}; \\ \delta = \{<a_0, z_1, a_1>, <a_1, z_0, a_2>, <a_2, z_0, a_3>, <a_3, z_0, a_0>, \\ \quad <a_0, z_2, a_4>, <a_4, z_0, a_5>, <a_5, z_0, a_0>\}. \end{cases} \tag{2.82}$$

Виділимо в ГСА Γ_{2-1} дві послідовності станів: $<a_0, a_1, a_2, a_3>$ та $<a_4, a_5, a_0>$. Реалізуємо переходи всередині даних послідовностей за допомогою операції декременту, інші переходи – канонічним способом. Для цього розіб'ємо абстрактну функцію переходів δ на дві часткові функції в такий спосіб:

$$\delta_1 = \{<a_0, z_1, a_1>, <a_1, z_0, a_2>, <a_2, z_0, a_3>, <a_4, z_0, a_5>, <a_5, z_0, a_0>\};$$

$$\delta_2 = \{<a_0, z_2, a_4>, <a_3, z_0, a_0>\}.$$

Сформуємо абстрактні підалгебри переходів G_{δ_1} та G_{δ_2}, що задаються, відповідно, виразами (2.71) та (2.72). Вважаючи, що сигнатура підалгебри G_{δ_1} утворена частковою функцією δ_1, а сигнатура підалгебри G_{δ_2} утворена частковою функцією δ_2, визначимо носії даних підалгебр:

$$A_{\delta_1} = \{a_0, a_1, a_2, a_3, a_4, a_5\},\ Z_{\delta_1} = \{z_0, z_1\};$$

$$A_{\delta_2} = \{a_0, a_3, a_4\},\ Z_{\delta_2} = \{z_0, z_2\}.$$

Надамо станам множини A_{δ_1} унікальні проміжні коди $K_I(a_i)$ із діапазону цілих чисел [0; 7] так, щоб усередині кожної лінійної послідовності станів код будь-якого стану, крім початкового, був на одиницю меншим за код попереднього

стану. Обрані в такий спосіб проміжні коди станів множини A_{δ_1} наведені в табл. 2.1.

Таблиця 2.1

Проміжні коди станів множини A_{δ_1} (ГСА Γ_{2-1})

a_i	a_0	a_1	a_2	a_3	a_4	a_5
$K_I(a_i)$	3	2	1	0	5	4

Реалізуємо переходи всередині кожної лінійної послідовності станів за допомогою операції декременту. Для цього задамо проміжну алгебру

$$G_I = < A_I, F_I > = < \{K_I(A_{\delta_1}), Z_{\delta_1}\}, \{O_{Dec}\} >, \qquad (2.83)$$

де $K_I(A_{\delta_1}) = \{K_I(a_0), ..., K_I(a_5)\}$; $Z_{\delta_1} = \{z_0, z_1\}$; O_{Dec} – операція декременту, що задається для сформованих лінійних послідовностей станів наступною множиною кортежів виду (2.79):

$$\begin{aligned} O_{Dec} = \{ & < K_I(a_0), z_1, K_I(a_1) >, < K_I(a_1), z_0, K_I(a_2) >, \\ & < K_I(a_2), z_0, K_I(a_3) >, < K_I(a_4), z_0, K_I(a_5) >, \\ & < K_I(a_5), z_0, K_I(a_0) > \} = \\ = \{ & < 3, z_1, 2 >, < 2, z_0, 1 >, < 1, z_0, 0 >, < 5, z_0, 4 >, < 4, z_0, 3 > \}. \end{aligned}$$

Оскільки існує бієктивна відповідність $A_{\delta_1} \leftrightarrow K_I(A_{\delta_1})$, що задається табл. 2.1, а також бієктивна відповідність між кортежами функцій δ_1 та O_{Dec}, сформована проміжна алгебра (2.83) є ізоморфною до абстрактної підалгебри G_{δ_1}. Задамо тепер структурну підалгебру переходів G_{d_1}, ізоморфну алгебрі (2.83).

Структурні коди станів множини A_{δ_1} задамо згідно зі значеннями їх проміжних кодів у трьохрозрядному беззнаковому цілочисельному форматі (табл. 2.2). Тоді структурна підалгебра G_{d_1} визначається виразом (2.84).

Таблиця 2.2

Структурні коди станів множини A_{δ_1} (ГСА Γ_1)

a_i	a_0	a_1	a_2	a_3	a_4	a_5
$K_S(a_i)$	0 1 1	0 1 0	0 0 1	0 0 0	1 0 1	1 0 0

$$
\begin{cases}
G_{d_1} = <\{K_S(A_{d_1}),\ K_S(Z_{d_1})\}, d_1>; \\
K_S(A_{d_1}) = \{K_S(a_0), \ldots, K_S(a_5)\} = \{011, 010, 001, 000, 101, 100\}; \\
K_S(Z_{d_1}) = \{-,\ 1\}; \\
d_1 = \{< K_S(a_0), x_1, K_S(a_1) >, < K_S(a_1), 1, K_S(a_2) >, < K_S(a_2), 1, K_S(a_3) >, \\
\quad < K_S(a_4), 1, K_S(a_5) >,\ < K_S(a_5), 1, K_S(a_0) >\} = \\
\quad = \{< 011, 1, 010 >, < 010, -, 001 >, < 001, -, 000 >, \\
\quad < 101, -, 100 >,\ < 100, -, 011 >\}.
\end{cases}
\tag{2.84}
$$

Наявність бієкцій $A_{\delta_1} \leftrightarrow K_S(A_{d_1})$, $Z_{\delta_1} \leftrightarrow K_S(Z_{d_1})$ та $\delta_1 \leftrightarrow d_1$ свідчить про ізоморфізм підалгебри (2.84) і проміжної алгебри (2.83). Отже, підалгебра (2.84) ізоморфна також і сформованій раніше абстрактній підалгебрі G_{δ_1}. Таким чином, ізоморфізм (2.80) системи (2.81) можна вважати заданим.

Задамо тепер другий ізоморфізм системи (2.81) для D-автомата. Для цього побудуємо структурну алгебру переходів G_d, обумовлену виразом (2.55) і ізоморфну абстрактній алгебрі переходів (2.82). Виходячи з (2.64), алгебра G_d може бути задана через завдання всіх її підалгебр. Оскільки у випадку D-автомата $G_d = \{G_{d_1}, G_{d_2}\}$ і підалгебра G_{d_1} вже задана виразом (2.84), для завдання G_d залишається задати структурну підалгебру G_{d_2}, ізоморфну абстрактній підалгебрі G_{δ_2}, таким чином, щоб забезпечити існування ізоморфізму (2.68).

Визначаючи підалгебру G_{d_2} виразом (2.70), задамо кожний з її компонентів. Носій $K_S(A_{d_2})$ сформуємо зі структурних кодів станів, що утворюють носій A_{δ_2} підалгебри G_{δ_2}. При цьому будемо вважати, що для станів, що входять у

множину $A_{\delta_1} \cap A_{\delta_2}$, структурні коди вже задані в процесі завдання підалгебри G_{d_1} (табл. 2.2). Залишається задати структурні коди тих станів із множини A_{δ_2}, які поки його не мають, тобто всіх станів із множини $A_{\delta_2} \setminus (A_{\delta_1} \cap A_{\delta_2})$.

Відповідно сформованим по ГСА Γ_1 лінійним послідовностям станів, у множину A_{δ_1} увійшли всі стани автомата. В результаті маємо $A_{\delta_1} \cap A_{\delta_2} = A_{\delta_2}$ і $A_{\delta_2} \setminus (A_{\delta_1} \cap A_{\delta_2}) = \varnothing$. Оскільки всі стани автомата вже одержали структурні коди на етапі формування структурної підалгебри G_{d_1}, окрема процедура завдання структурних кодів станів із множини $A_{\delta_2} \setminus (A_{\delta_1} \cap A_{\delta_2})$ в даному конкретному випадку не потрібна. Таким чином, носій $K_S(A_{d_2})$ структурної підалгебри G_{d_2} сформований. Носій $K_S(Z_{d_2})$ формується однозначно із раніше обраних структурних кодів вхідних сигналів. Після формування своїх носіїв підалгебра G_{d_2} приймає наступний вигляд:

$$\begin{cases} G_{d_2} = <\{K_S(A_{d_2}),\ K_S(Z_{d_2})\}, d_2>; \\ K_S(A_{d_2}) = \{K_S(a_0), K_S(a_3), K_S(a_4)\} = \{011, 000, 101\}; \\ K_S(Z_{d_2}) = \{K_S(z_0), K_S(z_2)\} = \{-,\ 0\}; \\ d_2 = \{< K_S(a_0), \bar{x}_1, K_S(a_4)>, < K_S(a_3), 1, K_S(a_0)\} = \\ \quad = \{<011, 0, 101>, <000, -, 011>\}. \end{cases} \tag{2.85}$$

Тепер, коли задані всі структурні підалгебри переходів, сформуємо компоненти структурної алгебри переходів G_S^d відповідно до виразу (2.55):

$$K_S(A) = K_S(A_{d_1}) \cup K_S(A_{d_2}) = \{K_S(a_0), ..., K_S(a_5)\} = \\ = \{011, 010, 001, 000, 101, 100\};$$

$$K_S(Z) = K_S(Z_{d_1}) \cup K_S(Z_{d_2}) = \{-, 1, 0\};$$

$$d = \{< K_S(a_0),\ K_S(z_1),\ K_S(a_1) >, < K_S(a_1),\ K_S(z_0),\ K_S(a_2) >,$$
$$< K_S(a_2),\ K_S(z_0),\ K_S(a_3) >, < K_S(a_3),\ K_S(z_0),\ K_S(a_0) >,$$
$$< K_S(a_0),\ K_S(z_2),\ K_S(a_4) >, < K_S(a_4),\ K_S(z_0),\ K_S(a_5) >,$$
$$< K_S(a_5),\ K_S(z_0),\ K_S(a_0) >\} =$$
$$= \{< 011, 1, 010 >, < 010, -, 001 >, < 001, -, 000 >, < 000, -, 011 >,$$
$$< 011, 0, 010 >, < 101, -, 100 >, < 100, -, 011 >\}.$$

Існування бієкцій $A \leftrightarrow K_S(A)$, $Z \leftrightarrow K_S(Z)$ та $\delta \leftrightarrow d$ дозволяє вважати алгебри G_δ та G_d ізоморфними. Даний ізоморфізм, разом зі встановленим раніше ізоморфізмом (2.80), говорять про можливість побудови D-автомата по ГСА Γ_{2-1}, тобто підтверджують зроблене вище узагальнення 1.

Узагальнення 2. *В автоматі допускається використання декількох проміжних алгебр.*

Нехай мікропрограмний автомат Мура заданий ГСА Γ_{2-1}. Абстрактний автомат, що відповідає даній ГСА, задається абстрактною алгеброю (2.82). Визначимо вхідні сигнали та їх структурні коди так само, як і в прикладі для узагальнення 1: $Z = \{z_0, z_1, z_2\}$, $K_S(z_0) = <->$, $K_S(z_1) = <1>$, $K_S(z_2) = <0>$, $K_S(Z) = \{K_S(z_0), K_S(z_1), K_S(z_2)\} = \{<->, <1>, <0>\}$.

Виділимо в ГСА Γ_{2-1} дві послідовності станів: $< a_0, a_1, a_2, a_3 >$ та $< a_0, a_4, a_5 >$. Побудуємо автомат, у якому переходи всередині першої послідовності реалізуються за допомогою схеми інкрементора, всередині другої послідовності – за допомогою схеми декрементора, інші переходи – канонічним способом. У рамках даної роботи будемо називати такий автомат CD-автоматом. При його побудові обмежимося етапом алгебраїчного синтезу.

Три способи реалізації автоматних переходів виражаються в розбивці абстрактної функції переходів δ і структурної функції переходів d на три часткові функції $\delta_1 - \delta_3$ та $d_1 - d_3$ відповідно. Нехай функції $\delta_1 - \delta_3$ утворюють сигнатури відповідних абстрактних підалгебр $G_{\delta_1} - G_{\delta_3}$, функції $d_1 - d_3$ – сигнатури відповідних структурних підалгебр $G_{\delta_1} - G_{\delta_3}$.

За аналогією з розглянутим раніше С-автоматом, задамо проміжну алгебру

$$G_{I_1} =< A_{I_1}, F_{I_1} >=< \{K_{I_1}(A_{\delta_1}),\ Z_{\delta_1}\}, O_1 >, \qquad (2.86)$$

у якій O_1 – операція інкременту, що виконується над скалярними числами цілого типу і представляє собою множину кортежів, подібних кортежу (2.79); $K_{I_1}(A_{\delta_1})$ – множина проміжних кодів тих станів, переходи з яких або переходи в які виконуються за допомогою часткової функції переходів δ_1.

Щоб часткова структурна функція переходів d_1 реалізовувала функцію δ_1 з використанням схеми інкрементора, задамо ізоморфізм

$$G_{\delta_1} \leftrightarrow G_{I_1} \leftrightarrow G_{d_1}, \qquad (2.87)$$

подібний ізоморфізму (2.80) для С-автомата.

За аналогією з розглянутим вище D-автоматом, задамо проміжну алгебру

$$G_{I_2} =< A_{I_2}, F_{I_2} >=< \{K_{I_2}(A_{\delta_2}), Z_{\delta_2}\}, O_2 >, \qquad (2.88)$$

у якій O_2 – операція декременту, яка виконується над скалярними числами цілого типу; $K_{I_2}(A_{\delta_2})$ – множина проміжних кодів тих станів, переходи з яких або переходи в які виконуються за допомогою часткової функції переходів δ_2. Ця алгебра повинна задаватися з урахуванням можливості існування ізоморфізму

$$G_{\delta_2} \leftrightarrow G_{I_2} \leftrightarrow G_{d_2}, \qquad (2.89)$$

при якому часткова структурна функція переходів d_2 реалізує функцію δ_2 з використанням схеми декрементора.

Оскільки в CD-автоматі функція δ_3 реалізується канонічним способом, при переході від абстрактної підалгебри G_{δ_3} до структурної підалгебри G_{δ_3} проміжна алгебра не використовується.

Як і для С-автомата, схемна реалізація функції переходів CD-автомата можлива при виконанні наступних вимог:

1. Кожному вхідному сигналу в носіях $K_S(Z_d)$ усіх структурних підалгебр, у яких він зустрічається, повинен відповідати єдиний і унікальний структурний код із множини $K_S(Z)$.

2. Кожному стану автомата в носіях $K_S(A_d)$ усіх структурних підалгебр, у яких воно зустрічається, повинен відповідати єдиний і унікальний структурний код із множини $K_S(A)$.

Перша вимога забезпечується тим фактом, що в розглянутому прикладі множина $K_S(Z)$ відома до виконання алгебраїчного синтезу і визначається заданою ГСА. Друга вимога забезпечується спільним завданням ізоморфізмів $G_{\delta_1} \leftrightarrow G_{d_1}$, $G_{\delta_2} \leftrightarrow G_{d_2}$ та $G_{\delta_3} \leftrightarrow G_{d_3}$. Як і у випадку C- і D-автоматів, для CD-автомата обидві вимоги виражаються в алгебраїчній формі ізоморфізмом (2.68). Таким чином, система ізоморфізмів (2.81) у випадку CD-автомата буде поєднувати ізоморфізми (2.87), (2.89) і (2.68), тобто матиме наступний вигляд:

$$\begin{cases} G_{\delta_1} \leftrightarrow G_{I_1} \leftrightarrow G_{d_1}; \\ G_{\delta_2} \leftrightarrow G_{I_2} \leftrightarrow G_{d_2}; \\ G_{\delta} \leftrightarrow G_{d}. \end{cases} \qquad (2.90)$$

Задамо дану систему. Розіб'ємо абстрактну функцію переходів δ алгебри (2.82) на часткові функції $\delta_1 - \delta_3$ в такий спосіб:

$$\delta_1 = \{< a_0, z_1, a_1 >, < a_1, z_0, a_2 >, < a_2, z_0, a_3 >\};$$

$$\delta_2 = \{< a_0, z_2, a_4 >, < a_4, z_0, a_5 >\};$$

$$\delta_3 = \{< a_3, z_0, a_0 >, < a_5, z_0, a_0 >\}.$$

Відповідно до даної розбивки сформуємо три абстрактні підалгебри переходів $G_{\delta_1} - G_{\delta_3}$:

$$G_{\delta_1} = < A_{\delta_1}, F_{\delta_1} > = < \{A_{\delta_1}, Z_{\delta_1}\}, \{\delta_1\} > = \\ = < \{\{a_0, a_1, a_2, a_3\}, \{z_0, z_1\}\}, \{\delta_1\} >; \qquad (2.91)$$

$$G_{\delta_2} = < A_{\delta_2}, F_{\delta_2} > = < \{A_{\delta_2}, Z_{\delta_2}\}, \{\delta_2\} > = \\ = < \{\{a_0, a_4, a_5\}, \{z_0, z_2\}\}, \{\delta_2\} >; \qquad (2.92)$$

$$G_{\delta_3} = < A_{\delta_3}, F_{\delta_3} > = < \{A_{\delta_3}, Z_{\delta_3}\}, \{\delta_3\} > = \\ = < \{\{a_0, a_3, a_5\}, \{z_0\}\}, \{\delta_3\} >. \qquad (2.93)$$

Задамо ізоморфізм (2.87) системи (2.90). Проміжні коди станів алгебри G_{I_1} задамо таким чином, щоб у кожному кортежі функції O_1 проміжний код вихідного

стану був на одиницю меншим за код стану переходу. Нехай, наприклад, $K_{I_1}(a_0)=2$, $K_{I_1}(a_1)=3$, $K_{I_1}(a_2)=4$, $K_{I_1}(a_3)=5$. Тоді проміжна алгебра G_{I_1} матиме наступний вигляд:

$$\begin{cases} G_{I_1} = \ < A_{I_1}, F_{I_1} > = < \{K_{I_1}(A_{\delta_1}), Z_{\delta_1}\}, \{O_1\} >; \\ K_{I_1}(A_{\delta_1}) = \{K_{I_1}(a_0), K_{I_1}(a_1), K_{I_1}(a_2), K_{I_1}(a_3)\} = \{2,3,4,5\}; \\ Z_{\delta_1} = \{z_0, z_1\}; \\ O_1 = \{< 2, z_1, 3 >, < 3, z_0, 4 >, < 4, z_0, 5 >\}. \end{cases} \tag{2.94}$$

Зіставляючи елементам множини $K_{I_1}(A_{\delta_1})$ трьохрозрядні структурні (двійкові) коди, що є двійковою формою відповідних проміжних кодів станів, задамо структурну підалгебру G_{d_1} в такий спосіб:

$$\begin{cases} G_{d_1} = \ < A_{d_1}, F_{d_1} > = < \{K_S(A_{d_1}), K_S(Z_{d_1})\}, \{d_1\} >; \\ K_S(A_{d_1}) = \{K_S(a_0), K_S(a_1), K_S(a_2), K_S(a_3)\} = \{010, 011, 100, 101\}; \\ K_S(Z_{d_1}) = \{K_S(z_0), K_S(z_1)\} = \{-, 1\}; \\ d_1 = \{< 010, 1, 011 >, < 011, -, 100 >, < 100, -, 101 >\}. \end{cases} \tag{2.95}$$

Оскільки дана підалгебра ізоморфна як абстрактній підалгебрі G_{δ_1}, що задається виразом (2.91), так і проміжній алгебрі G_{I_1}, що задається виразом (2.94), ізоморфізм (2.87) може вважатися заданим.

Задамо ізоморфізм (2.89) системи (2.90). Множину $K_{I_2}(A_{\delta_2})$ сформуємо з урахуванням наступних вимог.

1. У кожному кортежі функції O_2 проміжний код вихідного стану повинен бути на одиницю більшим за код стану переходу.

2. Оскільки стани, що утворюють підмножину $A_{\delta_1} \cap A_{\delta_2} \subseteq A_{\delta_2}$, вже одержали структурні коди в процесі формування підалгебри (2.95), для забезпечення існування ізоморфізму (2.68) структурні коди цих станів у підалгебрі G_{d_2} повинні збігатися з їхніми структурними кодами в підалгебрі (2.95). У розглянутому прикладі таким станом є a_0, для якого вже задані коди $K_{I_1}(a_0)=2$ та $K_S(a_0)=010$.

Нехай $K_{I_2}(a_0)=K_{I_1}(a_0)=2$, $K_{I_2}(a_4)=1$, $K_{I_2}(a_5)=0$. З урахуванням даних кодів проміжна алгебра G_{I_2} приймає наступний вигляд:

$$\begin{cases} G_{I_2} = < A_{I_2}, F_{I_2} > = < \{K_{I_2}(A_{\delta_2}), Z_{\delta_2}\}, \{O_2\} >; \\ K_{I_2}(A_{\delta_2}) = \{K_{I_2}(a_0), K_{I_2}(a_4), K_{I_2}(a_5)\} = \{2, 1, 0\}; \\ Z_{\delta_2} = \{z_0, z_2\}; \\ O_2 = \{< 2, z_2, 1 >, < 1, z_0, 0 >\}. \end{cases} \tag{2.96}$$

Зіставляючи елементам множини $K_{I_2}(A_{\delta_2})$ трьохрозрядні структурні (двійкові) коди, що дорівнюють їхній двійковій формі, одержуємо структурну підалгебру

$$\begin{cases} G_{d_2} = < A_{d_2}, F_{d_2} > = < \{K_S(A_{d_2}), K_S(Z_{d_2})\}, \{d_2\} >; \\ K_S(A_{d_2}) = \{K_S(a_0), K_S(a_4), K_S(a_5)\} = \{010, 001, 000\}; \\ K_S(Z_{d_2}) = \{K_S(z_0), K_S(z_2)\} = \{-, 0\}; \\ d_2 = \{< 010, 0, 001 >, < 001, -, 000 >\}. \end{cases} \tag{2.97}$$

Отримана підалгебра ізоморфна як абстрактній підалгебрі G_{δ_2}, яка задається виразом (2.92), так і проміжній алгебрі G_{I_2}, яка задається виразом (2.96). Отже, може вважати заданим ізоморфізм (2.89) системи (2.90).

Часткову функцію переходів δ_3 реалізуємо в канонічний спосіб, для чого задамо структурну підалгебру

$$G_{d_3} = < \{K_S(A_{d_3}), K_S(Z_{d_3})\}, \{d_3\} >, \tag{2.98}$$

ізоморфну підалгебрі G_{δ_3}. Для забезпечення існування ізоморфізму (2.68) ізоморфізм $G_{d_3} \leftrightarrow G_{\delta_3}$ повинен задаватися разом із уже заданими ізоморфізмами $G_{d_1} \leftrightarrow G_{\delta_1}$ та $G_{d_2} \leftrightarrow G_{\delta_2}$. Оскільки структурні коди вхідних сигналів у нашому випадку визначаються заданою ГСА і однакові для всіх структурних підалгебр, для існування ізоморфізму (2.68) достатньо забезпечити єдність і унікальність структурних кодів станів.

Задамо структурні коди станів множини A_{δ_3} в такий спосіб. Якщо стан $a_i \in A_{d_3}$ вже має структурний код в одній із раніше заданих структурних

підалгебр, то цей же код береться як структурний код у підалгебрі G_{d_3}. Якщо стан $a_i \in A_{\delta_3}$ відсутній у підалгебрах A_{δ_1} та A_{δ_2}, то в носії A_{δ_3} підалгебри G_{d_3} йому надається будь-який невикористаний структурний код із множини припустимих структурних кодів.

У розглянутому прикладі $A_{\delta_3} \setminus ((A_{\delta_1} \cup A_{\delta_2}) \cap A_{\delta_3}) = \varnothing$, тобто всі стани автомата вже одержали структурні коди в процесі формування підалгебр G_{d_1} і G_{d_2}. Таким чином, у даному конкретному випадку окрема процедура завдання структурних кодів станів множини $A_{\delta_3} \setminus ((A_{\delta_1} \cup A_{\delta_2}) \cap A_{\delta_3})$ не потрібна. Тоді структурна підалгебра G_{d_3}, обумовлена виразом (2.98), приймає наступний вигляд:

$$\begin{cases} G_{d_3} = \ < A_{d_3}, F_{d_3} > = < \{K_S(A_{d_3}), K_S(Z_{d_3})\}, \{d_3\} >; \\ K_S(A_{d_3}) = \{K_S(a_0), K_S(a_3), K_S(a_5)\} = \{010, 101, 000\}; \\ K_S(Z_{d_3}) = \{K_S(z_0)\} = \{-\}; \\ d_3 = \{< 101, -, 010 >, < 000, -, 010 >\}. \end{cases} \tag{2.99}$$

Тепер може бути задана структурна алгебра переходів G_d. Для цього слід виконати операцію об'єднання відповідних носіїв, а також об'єднати в одну множину кортежі всіх часткових структурних функцій. Одержувана в результаті алгебра описується виразом (2.100).

$$\begin{cases} G_d = \ < A_d, F_d > = < \{K_S(A_d), K_S(Z_d)\}, \{d\} >; \\ K_S(A_d) = K_S(A_{d_1}) \cup K_S(A_{d_2}) \cup K_S(A_{d_3}) = \\ \qquad = \{K_S(a_0), K_S(a_3), K_S(a_5), K_S(a_5), K_S(a_5), K_S(a_5)\} = \\ \qquad = \{010, 011, 100, 101, 001, 000\}; \\ K_S(Z_d) = K_S(Z_{d_1}) \cup K_S(Z_{d_2}) \cup K_S(Z_{d_3}) = \\ \qquad = \{K_S(z_0), K_S(z_1), K_S(z_1)\} = \{-, 1, 0\}; \\ d = d_1 \cup d_2 \cup d_3 = \{< 010, 1, 011 >, < 011, -, 100 >, < 100, -, 101 >, \\ \quad < 010, 0, 001 >, < 001, -, 000 >, < 101, -, 010 >, < 000, -, 010 >\}. \end{cases} \tag{2.100}$$

Аналіз виразів (2.82) та (2.100) дозволяє твердити про існування ізоморфізму (2.68). Таким чином, усі ізоморфізми системи (2.90) задані, що підтверджує можливість існування CD-автомата і зроблене вище узагальнення 2.

На рис. 2.3 наведена структурна схема CD-автомата. Дана структура аналогічна структурі C-автомата (рис. 2.1) і організована в такий спосіб.

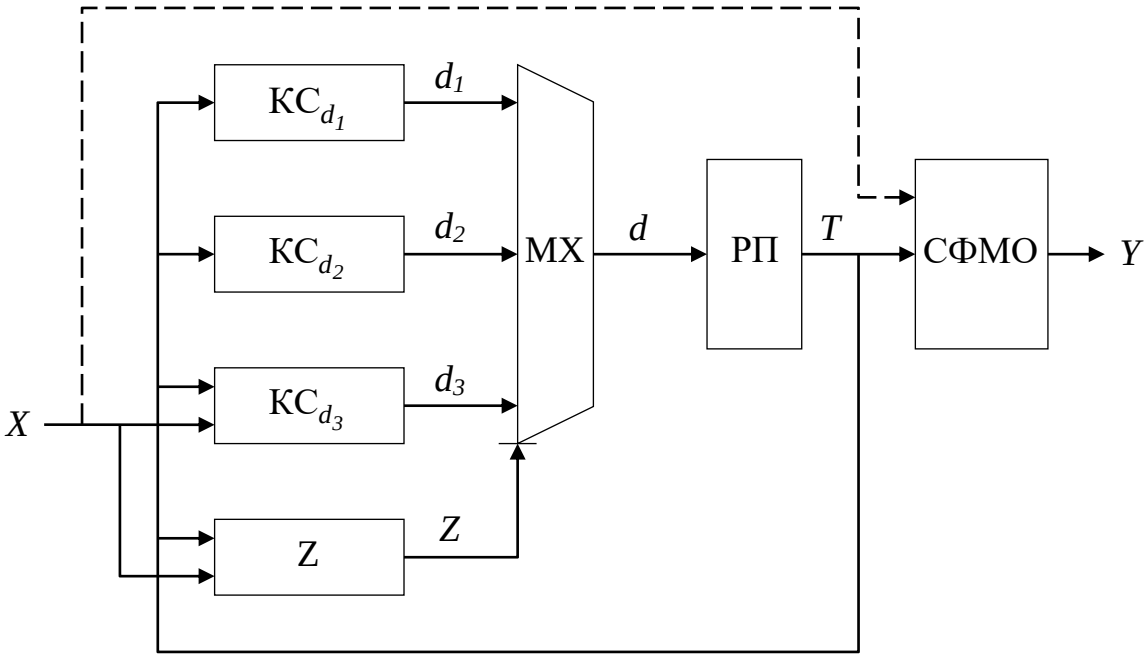


Рисунок 2.3 – Структурна схема CD-автомата

Блок $КС_{d_1}$ становить собою інкрементор, на вхід якого подається R-розрядний код стану T. На виході формується значення часткової структурної функції d_1, що довизначена якимось чином на всій множині R-розрядних структурних кодів станів.

Блок $КС_{d_2}$ становить собою R-розрядний декрементор, на виході якого формується значення часткової структурної функції d_2, що довизначена якимось чином на всій множині R-розрядних структурних кодів станів.

Блок $КС_{d_3}$ реалізує часткову структурну функцію переходів d_3, представлену у вигляді системи канонічних рівнянь. Відзначимо, що на рис. 2.3 на вхід $КС_{d_3}$ подаються вхідні сигнали X, хоча в кортежах часткової функції d_3 (вираз 2.99) сигнали x_1 або $\bar{x}_1$ відсутні. Справа в тому, що в розглянутому

прикладі функція d_3 реалізує тільки безумовні переходи, однак у загальному випадку вона може залежати від різних сигналів логічних умов, що й відображене на рис. 2.3.

Мультиплексор МХ становить собою *R*-розрядний мультиплексор з трьох напрямків, що керується дворозрядним кодом *Z*. Даний код формується Z-підсхемою (аналогом схеми $КС_{Inc}$ на рис. 2.1) і визначає часткову структурну функцію переходів, яка використовується в поточному такті роботи автомата.

Узагальнення 3. *У проміжних алгебрах припустимі різні інтерпретації структурних кодів станів*.

У структурній алгебрі переходів кодом стану є бітовий вектор, унікальний у рамках обраної множини структурних кодів станів. В обчислювальній техніці бітовий вектор розглядається не тільки як унікальний набір бітів, але й допускає різну числову (скалярну) інтерпретацію [141]. Деякі стандартні способи інтерпретації бітових векторів наведені на рис. 2.4.

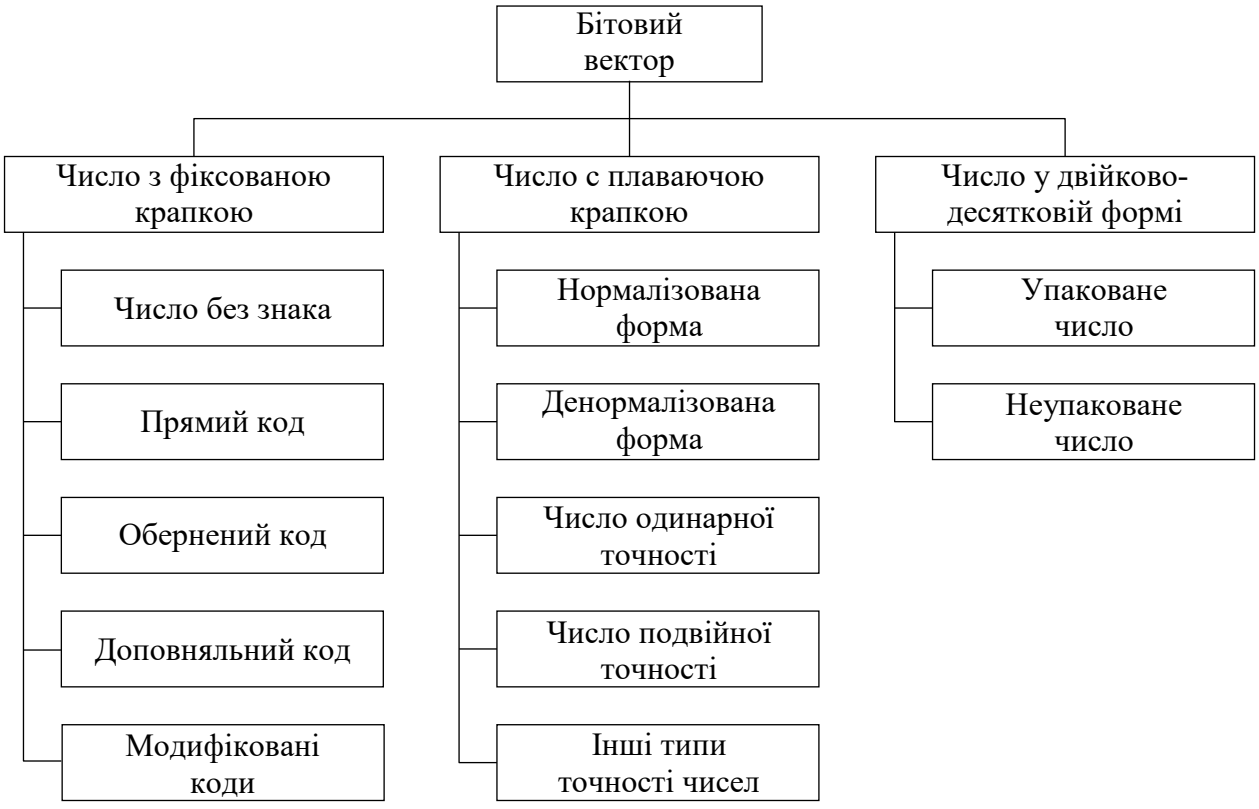


Рисунок 2.4 – Деякі способи числової інтерпретації бітового вектора

При фіксованій кількості двійкових розрядів кожний з даних форматів відповідає деякій множині чисел, над якою може бути визначена множина операцій різної арності. При цьому будь-яка операція може бути реалізована схемним шляхом у більшості відомих елементних базисів, зазвичай допускаючи різні архітектурні варіанти реалізації, що характеризуються витратами апаратури, швидкодією, регулярністю, універсальністю тощо.

Окремий автоматний перехід є перетворенням одного структурного (двійкового) вектора в інший і може бути реалізований при різних інтерпретаціях структурних кодів станів. Нехай для стану a_i заданий структурний код $K_S(a_i) = 11101011_2$. Цей код може інтерпретуватися як ціле без знака, що дорівнює 235_{10}, або як ціле в прямому коді, що дорівнює $-107_{\text{ПК}}$, або як ціле в доповняльному коді, що дорівнює $-21_{\text{ДК}}$.

Нехай в автоматі виконується перехід, при якому потрібно перетворити код стану

$$K_S(a_i) = 11101011_2 = 235_{10} = -107_{\text{ПК}} = -21_{\text{ДК}}$$

у код

$$K_S(a_j) = 00010100_2 = 20_{10} = +20_{\text{ПК}} = +20_{\text{ДК}}.$$

Інтерпретуючи бітові вектори кодів станів як числа в різних форматах, перетворення $K_S(a_i)$ в $K_S(a_j)$ можна виконати, наприклад, відповідно до рис. 2.5, де центральні блоки містять операції, необхідні для перетворення $K_S(a_i)$ в $K_S(a_j)$

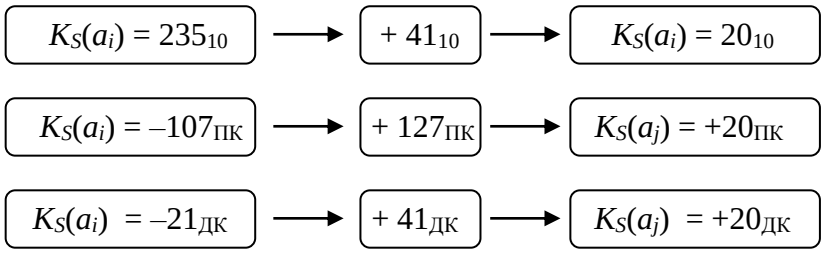


Рисунок 2.5 – Еквівалентні перетворення бітового вектора для його різних інтерпретацій

У даному прикладі всі операції виконуються над восьмирозрядними двійковими операндами з відкиданням переносу зі старшого розряду. При цьому кожна із цих операцій дає в результаті однаковий бітовий вектор $K_S(a_j)$.

На рис. 2.5 усі три перетворення виконуються за допомогою операції підсумовування з константою, але при цьому числова інтерпретація операндів різна. Це дозволяє зіставити кожному перетворенню окрему проміжну алгебру з унікальними серед даних алгебр носієм і сигнатурою. При цьому схемна реалізація операції в кожному випадку буде залежати від способу інтерпретації структурних кодів станів.

З іншого боку, при деякій обраній інтерпретації структурних кодів одне й те саме перетворення може виконуватися за допомогою різних операцій. Так, на рис. 2.6 показані кілька варіантів перетворення $K_S(a_i)$ в $K_S(a_j)$ для розглянутого вище прикладу при інтерпретації структурних кодів станів як чисел у доповняльному коді.

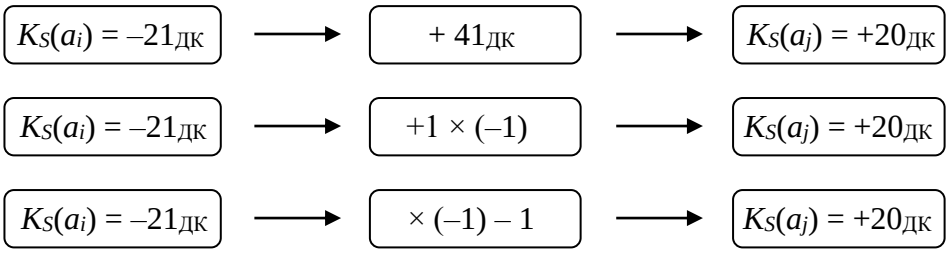


Рисунок 2.6 – Еквівалентні перетворення бітового вектора в форматі доповняльного коду за допомогою різних операцій

Усі операції, представлені на рис. 2.5 і 2.6, дозволяють виконати перетворення двійкового вектора «11101011» у вектор «00010100». Помітимо, що дане перетворення також може бути виконане за допомогою логічної операції порозрядної інверсії, яка належить до операцій булевої алгебри і визначена на множині слів двійкового алфавіту. При використанні операції інверсії як сигнатури проміжної алгебри яка-небудь числова (скалярна) інтерпретація для її

операндів не потрібні, внаслідок чого проміжні коди станів будуть збігатися з їхніми структурними кодами.

У прикладі для узагальнення 2 структурна підалгебра G_{d_2}, що задається виразом (2.97), виконує два переходи, у яких перетворення структурних кодів $010 \to 001$ та $001 \to 000$ виконуються за допомогою схеми декрементора. Помітимо, що еквівалентні перетворення структурних кодів можуть бути виконані за допомогою операції логічного зсуву операнда на один розряд праворуч із заповненням вивільнюваного старшого розряду нулем. Якщо в проміжній алгебрі G_{I_2}, заданої виразом (2.88), у якості операції O_2 замість операції декремента використовувати операцію зсуву, вираз (2.96), що задає структурну алгебру G_{I_2}, прийме наступний вигляд:

$$\begin{cases} G_{I_2} = < A_{I_2}, F_{I_2} > = < \{K_{I_2}(A_{\delta_2}), Z_{\delta_2}\}, \{O_2\} > ; \\ K_{I_2}(A_{\delta_2}) = \{K_{I_2}(a_0), K_{I_2}(a_4), K_{I_2}(a_5)\} = \{010, 001, 000\}; \\ Z_{\delta_2} = \{z_0, z_2\}; \\ O_2 = \{< 010, z_2, 001 >, \ < 001, z_0, 000 >\}. \end{cases} \qquad (2.101)$$

При цьому в структурі на рис. 2.3 блок КС_{d_2} буде являти собою схему логічного зсуву, яка реалізується перекомутацією проводів. Оскільки витрати апаратури в схемі зсуву менші, ніж у схемі декрементора, завдання проміжної алгебри переходів G_{I_2} у вигляді (2.101) виявляється в цьому випадку більш кращим у порівнянні з (2.96).

У розглянутому прикладі вибір способу інтерпретації структурних кодів станів виконується після розбивки функції переходів на часткові функції. У загальному ж випадку сама розбивка функції переходів може виконуватися з урахуванням обраного способу інтерпретації структурних кодів. Саме так відбувається в С-автоматі, де з самого початку передбачається цілочисельна беззнакова інтерпретація структурних кодів, відповідно до якої формуються лінійні послідовності станів.

Узагальнення 4. *Вхідні сигнали автомата можуть виступати в якості аргументів операції переходів*.

Відповідно до узагальнення 3, у проміжній алгебрі структурний код стану інтерпретується як деяка скалярна або векторна величина. Разом з тим свою інтерпретацію можуть мати і структурні коди вхідних сигналів, причому інтерпретації кодів станів і вхідних сигналів можуть як збігатися, так і бути різними. Пояснимо сказане декількома прикладами.

Приклад 1. Нехай деяка ГСА Γ_{2-2}, позначена станами автомата Мілі, містить фрагмент, зображений на рис. 2.7.

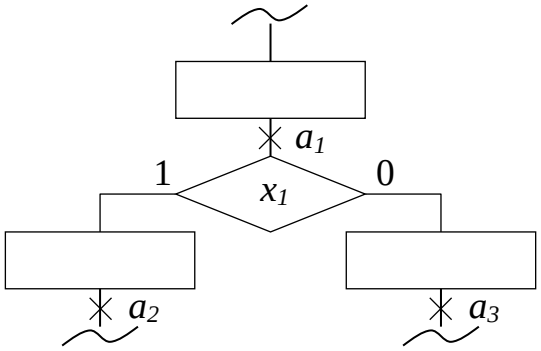


Рисунок 2.7 – Фрагмент ГСА Γ_{2-2}

Даний фрагмент містить два переходи зі стану a_1: перехід за умовою $x_1 = 1$ та перехід за умовою $x_1 = 0$. Двом типам переходів в абстрактному автоматі відповідають два вхідні сигнали z_1 та z_2, структурні коди яких, при $x_1 \in \{1, 0, -\}$, визначимо як $K_S(z_1) = <1>$, $K_S(z_2) = <0>$. Оскільки в даному фрагменті безумовні переходи відсутні, абстрактний вхідний сигнал z_0 з кодом $K_S(z_0) = <->$ розглядатися не буде.

Нехай задана часткова абстрактна функція переходів

$$\delta_1 = \{<a_1, z_1, a_2>, <a_1, z_2, a_3>\},$$

що реалізує переходи всередині фрагмента на рис. 2.7 і утворюює сигнатуру абстрактної підалгебри

$$\begin{aligned} G_{\delta_1} &= <A_{\delta_1}, F_{\delta_1}> = <\{A_{\delta_1}, Z_{\delta_1}\}, \{\delta_1\}> = \\ &= <\{\{a_1, a_2, a_3\}, \{z_1, z_2\}\}, \{\delta_1\}>. \end{aligned} \qquad (2.102)$$

Нехай також задана ізоморфна підалгебрі G_{δ_1} проміжна алгебра

$$G_{I_1} =< A_{I_1}, F_{I_1} >=<\{K_{I_1}(A_{\delta_1}), K_{I_1}(Z_{\delta_1})\}, O_1 >, \qquad (2.103)$$

у якій операція переходів O_1 є множина кортежів виду

$$< K_{I_1}(a_i), K_{I_1}(z_j), K_{I_1}(a_k) >\in O_1. \qquad (2.104)$$

Компонентом $K_{I_1}(a_i)$ кортежу (2.104) є проміжний код поточного стану автомата, $K_{I_1}(a_k)$ – проміжний код стану переходу, $K_{I_1}(z_j)$ – проміжний код вхідного сигналу, під впливом якого здійснюється перехід зі стану a_i в стан a_k. Таким чином, особливістю алгебри (2.103) є використання проміжних кодів не тільки для станів, але й для вхідних сигналів.

Умовимося в даному прикладі множини $K_{I_1}(A_{\delta_1})$ та $K_{I_1}(Z_{\delta_1})$ формувати із множини натуральних чисел, причому в такий спосіб, щоб у загальному випадку $K_{I_1}(A_{\delta_1}) \cap K_{I_1}(Z_{\delta_1}) \neq \varnothing$. Дамо наступну числову інтерпретацію вхідних сигналів: $K_{I_1}(z_1) = 1$, $K_{I_1}(z_2) = 0$. З урахуванням даної інтерпретації задамо операцію переходів O_1, як операцію над скалярними величинами, у такий спосіб:

$$O_1 \colon K_{I_1}(a_k) = K_{I_1}(a_i) \cdot 2 + K_{I_1}(z_j). \qquad (2.105)$$

Алгебраїчно даний вираз є складеною функцією, у якій внутрішньою функцією є множення аргументу $K_{I_1}(a_i)$ на константу 2, а зовнішньою функцією – підсумовування результату внутрішньої функції з аргументом $K_{I_1}(z_j)$.

Оскільки елемент $K_{I_1}(z_j)$ може набувати одне з двох можливих значень (0 або 1), кожному значенню $K_{I_1}(a_i)$ в операції O_1 будуть відповідати два різні значення $K_{I_1}(a_k)$. Це дозволяє за допомогою операції (2.105) реалізувати двоспрямований умовний перехід, якому в операції переходів відповідають два кортежі (2.104).

Операція O_1 проміжної алгебри (2.103) формально включає наступні два кортежі:

$$O_1 = \{< K_{I_1}(a_1), K_{I_1}(z_1), K_{I_1}(a_2) >, < K_{I_1}(a_1), K_{I_1}(z_2), K_{I_1}(a_3) >\}.$$

Надамо стану a_1 проміжний код $K_{I_1}(a_1)$, унікальний в рамках використовуваної множини проміжних кодів станів. Підставляючи значення

$K_{I_1}(a_1)$ у вираз (2.105) на місце аргументу $K_{I_1}(a_i)$, можна отримати проміжні коди станів переходів для різних значень проміжних кодів вхідних сигналів.

Нехай $K_{I_1}(a_1)$=5. Тоді можуть бути визначені проміжні коди станів переходів $K_{I_1}(a_2)$ та $K_{I_1}(a_3)$: при $K_{I_1}(z_1)$=1 (при переході з a_1 за умовою $x_1=1$) $K_{I_1}(a_2)=K_{I_1}(a_1)\cdot 2+K_{I_1}(z_1)$=5·2+1=11, при $K_{I_1}(z_2)$=0 (при переході з a_1 по $x_1=0$) $K_{I_1}(a_3)=K_{I_1}(a_1)\cdot 2+K_{I_1}(z_2)$=5·2+0=10. Отримані коди дозволяють задати проміжну алгебру (2.103) у такий спосіб:

$$\begin{cases} G_{I_1}=<\{K_{I_1}(A_{\delta_1}),K_{I_1}(Z_{\delta_1})\},\{O_1\}>; \\ K_{I_1}(A_{\delta_1})=\{K_{I_1}(a_1),K_{I_1}(a_2),K_{I_1}(a_3)\}=\{5,11,10\}; \\ K_{I_1}(Z_{\delta_1})=\{K_{I_1}(z_1),K_{I_1}(z_2)\}=\{1,0\}; \\ O_1=\{<5,1,11>,<5,0,10>\}. \end{cases} \quad (2.106)$$

Зіставляючи проміжним кодам станів структурні коди, що дорівнюють їхній чотирирозрядній двійковій формі, задамо структурну підалгебру G_{d_1}, ізоморфну одночасно підалгебрі G_{δ_1} та алгебрі G_{I_1}:

$$\begin{cases} G_{d_1}=<\{K_S(A_{d_1}),K_S(Z_{d_1})\},\{d_1\}>; \\ K_S(A_{d_1})=\{K_S(a_1),K_S(a_2),K_S(a_3)\}=\{0101,1011,1010\}; \\ K_S(Z_{d_1})=\{K_S(z_1),K_S(z_2)\}=\{1,0\}; \\ d_1=\{<0101,1,1011>,<0101,0,1010>\}. \end{cases} \quad (2.107)$$

У структурному автоматі операція O_1, задана виразом (2.105), може бути реалізована у вигляді функціональної схеми, зображеної на рис. 2.8.

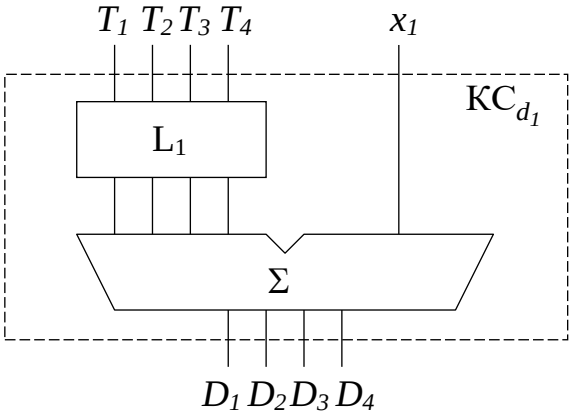


Рисунок 2.8 – Функціональна схема для операції (2.105)

Тут структурний код поточного стану, представлений змінними $T_1 - T_4$, надходить на вхід схеми L_1, що виконує операцію логічного зсуву ліворуч, еквівалентну множенню на 2. Подвоєний код поточного стану подається на одне плече суматора Σ, у той час як на друге плече подається структурний код вхідного сигналу, представлений однорозрядним значенням сигналу x_1. Схема суматора організована так, що значення сигналу x_1 додається до молодшого розряду першого операнда, забезпечуючи таким чином формування структурного коду стану переходу, представленого структурними змінними $D_1 - D_4$, відповідно до виразу (2.105). Блоки L_1 та Σ утворюють блок $КС_{d_1}$, у якому структурний вхідний сигнал x_1 має числову (скалярну) інтерпретацію, тобто бере участь у процесі перетворення коду стану як деяке число.

Відзначимо також наступне. Припустимо, що в структурному автоматі, що імплементує ГСА Γ_{2-2}, структурна функція переходів d представлена трьома частковими функціями $d_1 - d_3$. У цьому випадку структурна схема автомата буде відповідати схемі на рис. 2.3, у якій блок $КС_{d_1}$ реалізує два автоматні переходи часткової функції відповідно до виразу (2.107).

Як було відзначено раніше, у структурі на рис. 2.3 роль Z-підсхеми полягає у формуванні коду Z часткової структурної функції переходів d_i, яка використовується в поточному такті роботи автомата для формування коду стану переходу. При обраному структурному коді $K_S(a_1) = 0101_2$ Z-підсхема повинна формувати код часткової функції d_1 у двох випадках: при структурному коді поточного стану $T_1T_2T_3T_4 = 0101$ та структурному вхідному сигналі $x_1 = 1$ або при $T_1T_2T_3T_4 = 0101$ та $x_1 = 0$. Таким чином, незважаючи на те, що перехід зі стану a_1 формально залежить від значення логічної умови x_1, результат операції O_1 використовується у випадку $T_1T_2T_3T_4 = 0101$ при будь-якому значенні x_1, тобто незалежно від значення даної логічної умови. Це дозволяє не використовувати змінну x_1 в системі рівнянь Z-підсхеми при формуванні коду часткової функції

d_1. Інакше кажучи, структурна змінна x_1 «переміщається» з Z-підсхеми в схему операції O_1, спрощуючи Z-підсхему та ускладнюючи схему $КС_{d_1}$.

Приклад 2. Нехай заданий фрагмент деякої ГСА Γ_{2-3}, позначеної станами автомата Мура (рис. 2.9). Даний фрагмент містить чотири переходи зі стану a_1: за умовами $x_1x_2 = 00$, $x_1x_2 = 01$, $x_1x_2 = 10$ та $x_1x_2 = 11$. В абстрактному автоматі ці переходи виконуються під впливом абстрактних вхідних сигналів $z_1 - z_4$, структурні коди яких визначаються значеннями компонентів вектора $< x_1x_2 >$. Наприклад, $K_S(z_1) = <00>$, $K_S(z_2) = <01>$, $K_S(z_3) = <10>$, $K_S(z_4) = <11>$.

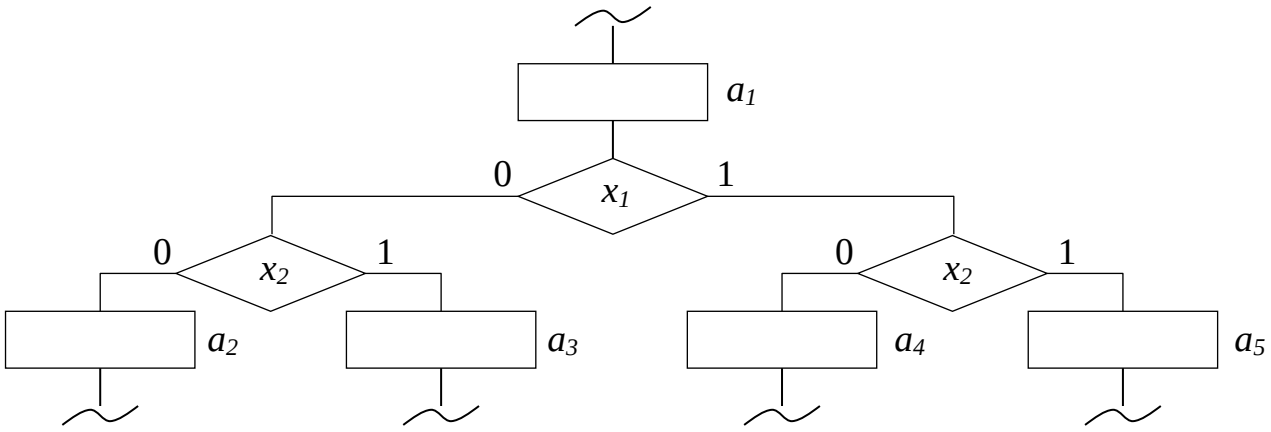


Рисунок 2.9 – Фрагмент ГСА Γ_{2-3}

Нехай переходи зі стану a_1 реалізуються частковою абстрактною функцією переходів δ_1:

$$\delta_1 = \{< a_1, z_1, a_2 >, < a_1, z_2, a_3 >, < a_1, z_3, a_4 >, < a_1, z_4, a_5 >\}.$$

Задамо абстрактну підалгебру переходів, сигнатура якої утворена частковою функцією δ_1:

$$\begin{aligned} G_{\delta_1} &= < A_{\delta_1}, F_{\delta_1} > = < \{A_{\delta_1}, Z_{\delta_1}\}, \{\delta_1\} > = \\ &= < \{\{a_1, a_2, a_3, a_4, a_5\}, \{z_1, z_2, z_3, z_4\}\}, \{\delta_1\} >. \end{aligned} \tag{2.108}$$

Як і в попередньому прикладі, визначимо проміжну алгебру виду (2.103), ізоморфну підалгебрі (2.108), у якій операція переходів O_1 утворена множиною кортежів виду (2.104).

Умовимось в даному прикладі розглядати елементи множин $K_{I_1}(A_{\delta_1})$ та $K_{I_1}(Z_{\delta_1})$ як чотирирозрядні бітові вектори. Оскільки вхідні сигнали автомата не залежать від його поточного стану, той самий бітовий вектор може одночасно належати множинам $K_{I_1}(A_{\delta_1})$ та $K_{I_1}(Z_{\delta_1})$, тобто в загальному випадку $K_{I_1}(A_{\delta_1}) \cap K_{I_1}(Z_{\delta_1}) \neq \varnothing$.

Для вхідних сигналів $z_1 - z_4$ що відповідають векторам логічних умов $< x_1 x_2 >$, задамо проміжні коди чотирирозрядними двійковими векторами виду $<1, K_S(z_i), 1> = <1, x_1, x_2, 1>$. У даних векторах старший і молодший розряди дорівнюють одиницям, проміжні розряди утворені розрядами структурного коду вхідного сигналу, яким відповідають структурні вхідні сигнали x_1 та x_2. Таким чином, $K_{I_1}(z_1) = 1001$, $K_{I_1}(z_2) = 1011$, $K_{I_1}(z_3) = 1101$, $K_{I_1}(z_4) = 1111$.

З урахуванням векторної інтерпретації проміжних кодів станів і вхідних сигналів задамо операцію переходів O_1 в аналітичній формі як порозрядну логічну операцію «сума за модулем 2»:

$$O_1: \quad K_{I_1}(a_k) = K_{I_1}(a_i) \oplus K_{I_1}(z_j). \qquad (2.109)$$

Вираз (2.109) дозволяє при відомих проміжних кодах $K_{I_1}(z_1) - K_{I_1}(z_4)$ одержати для деякого значення $K_{I_1}(a_1)$ проміжні коди станів переходів $a_2 - a_5$. Нехай $K_{I_1}(a_1) = <0101>$. Тоді

$$K_{I_1}(a_2) = K_{I_1}(a_1) \oplus K_{I_1}(z_1) = <0101> \oplus <1001> = <1100>;$$

$$K_{I_1}(a_3) = K_{I_1}(a_1) \oplus K_{I_1}(z_2) = <0101> \oplus <1011> = <1110>;$$

$$K_{I_1}(a_4) = K_{I_1}(a_1) \oplus K_{I_1}(z_3) = <0101> \oplus <1101> = <1000>;$$

$$K_{I_1}(a_5) = K_{I_1}(a_1) \oplus K_{I_1}(z_4) = <0101> \oplus <1111> = <1010>.$$

Тепер проміжна алгебра (2.103), сигнатура якої утворена операцією (2.109), може бути задана в такий спосіб:

$$\begin{cases} G_{I_1} = <\{K_{I_1}(A_{\delta_1}), K_{I_1}(Z_{\delta_1})\}, \{O_1\}>; \\ K_{I_1}(A_{\delta_1}) = \{K_{I_1}(a_1), K_{I_1}(a_2), K_{I_1}(a_3), K_{I_1}(a_4), K_{I_1}(a_5)\} = \\ \qquad = \{0101, 1100, 1110, 1000, 1010\}; \\ K_{I_1}(Z_{\delta_1}) = \{K_{I_1}(z_1), K_{I_1}(z_2), K_{I_1}(z_3), K_{I_1}(z_4)\} = \\ \qquad = \{1001, 1011, 1101, 1111\}; \\ O_1 = \{<0101, 1001, 1100>, <0101, 1011, 1110>, \\ \qquad <0101, 1101, 1000>, <0101, 1111, 1010>\}. \end{cases} \quad (2.110)$$

Відзначимо, що дана алгебра ізоморфна підалгебрі G_{δ_1}, що задається виразом (2.108). Задамо тепер структурну підалгебру G_{d_1}, ізоморфну проміжній алгебрі (2.110).

Оскільки проміжні коди станів представлені у векторній формі, будемо вважати їхні структурні коди рівними проміжним кодам. Тоді підалгебра G_{d_1} задається виразом (2.111).

Операція переходів (2.109) досить просто реалізується схемним шляхом (рис. 2.10). Слід підкреслити, що стани із множини A_{δ_1} та вхідні сигнали із множини Z_{δ_1}, що зустрічаються в кортежах інших часткових функцій, повинні мати у всіх структурних підалгебрах ті ж структурні коди, що й у підалгебрі (2.111). Дана вимога обумовлена вимогою спільного завдання ізоморфизмів відповідних абстрактних і структурних підалгебр переходів.

$$\begin{cases} G_{d_1} = <\{K_S(A_{d_1}), K_S(Z_{d_1})\}, \{d_1\}>; \\ K_S(A_{d_1}) = \{K_S(a_1), K_S(a_2), K_S(a_3), K_S(a_4), K_S(a_5)\} = \\ \qquad = \{0101, 1100, 1110, 1000, 1010\}; \\ K_S(Z_{d_1}) = \{K_S(z_1), K_S(z_2), K_S(z_3), K_S(z_4)\} = \\ \qquad = \{00, 01, 10, 11\}; \\ d_1 = \{<0101, 00, 1100>, <0101, 01, 1110>, \\ \qquad <0101, 10, 1000>, <0101, 11, 1010>\}. \end{cases} \quad (2.111)$$

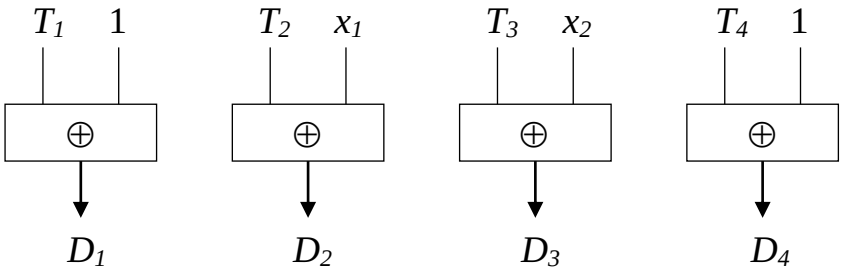


Рисунок 2.10 – Функціональна схема для операції (2.109)

Приклад 3. Розглянемо фрагмент ГСА Γ_{2-4}, що позначена станами автомата Мілі (рис. 2.11).

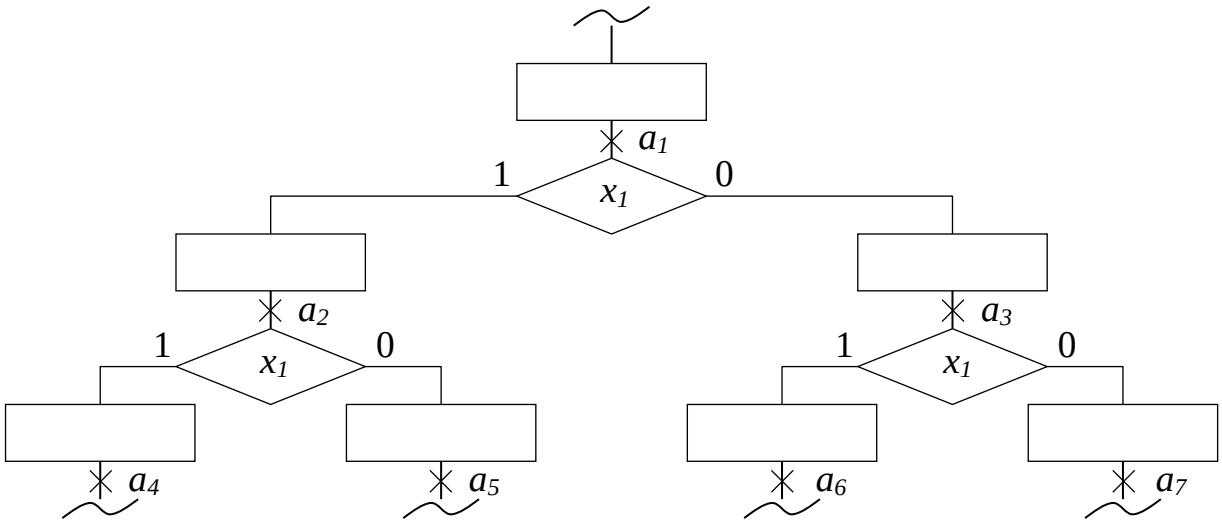


Рисунок 2.11 – Фрагмент ГСА Γ_{2-4}

Єдиний структурний сигнал x_1, що визначає переходи в рамках даного фрагмента, визначений на множині $\{1, 0, -\}$. У рамках даного фрагмента сигнал x_1 не буде набувати значення «–» через відсутність безумовних переходів. Два значення, що залишилися, визначають структурні коди $K_S(z_1) = <1>$ та $K_S(z_2) = <0>$ абстрактних вхідних сигналів z_1 та z_2.

Представимо множину переходів в рамках фрагмента на рис. (2.11) у вигляді часткової функції переходів δ_1:

$$\delta_1 = \{< a_1, z_1, a_2 >, < a_1, z_2, a_3 >, < a_2, z_1, a_4 >,$$
$$< a_2, z_2, a_5 >, < a_3, z_1, a_6 >, < a_3, z_2, a_7 >\}.$$

Задамо абстрактну підалгебру переходів, сигнатура якої утворена частковою функцією δ_1:

$$G_{\delta_1} =< A_{\delta_1}, F_{\delta_1} >=< \{A_{\delta_1}, Z_{\delta_1}\}, \{\delta_1\} >= \\ =< \{\{a_1, a_2, a_3, a_4, a_5, a_6, a_7\}, \{z_1, z_2\}\}, \{\delta_1\} > . \quad (2.112)$$

Визначимо ізоморфну даній підалгебрі проміжну алгебру виду (2.103), у якій проміжні коди станів інтерпретуються як натуральні числа, проміжні коди вхідних сигналів – як логічні величини, що набувають значення із множини {ІСТИНА, ХИБНІСТЬ}: $K_{I_1}(z_1) = \text{ІСТИНА}$, $K_{I_1}(z_2) = \text{ХИБНІСТЬ}$.

Логічна величина може використовуватися в умовній (інтервальній) функції для вибору одного з можливих способів обчислення результату. Задамо операцію переходу O_1 проміжної алгебри (2.103) виразом

$$O_1: \; K_{I_1}(a_k) = \begin{cases} K_{I_1}(a_i) + 5, \text{ якщо } K_{I_1}(z_j) = K_{I_1}(z_1); \\ K_{I_1}(a_i)/2 + 5, \text{ якщо } K_{I_1}(z_j) = K_{I_1}(z_2), \end{cases} \quad (2.113)$$

у якому $K_{I_1}(a_i)$, $K_{I_1}(z_j)$ та $K_{I_1}(a_k)$ – компоненти вектора виду (2.104).

Якщо прийняти $K_{I_1}(a_1) = 1$, то відповідно до рис. 2.11 і виразу (2.113) одержуємо $K_{I_1}(a_2) = 6$, $K_{I_1}(a_3) = 5$, $K_{I_1}(a_4) = 11$, $K_{I_1}(a_5) = 8$, $K_{I_1}(a_6) = 10$, $K_{I_1}(a_7) = 7$. Тепер проміжна алгебра (2.104) задається виразом (2.114) і в такому виді є ізоморфною абстрактній підалгебрі (2.112).

$$\begin{cases} G_{I_1} =< A_{I_1}, F_{I_1} >=< \{K_{I_1}(A_{\delta_1}), K_{I_1}(Z_{\delta_1})\}, \{O_1\} >; \\ K_{I_1}(A_{\delta_1}) = \{K_{I_1}(a_1), ..., K_{I_1}(a_7)\} = \{1, 6, 5, 11, 8, 10, 7\}; \\ K_{I_1}(Z_{\delta_1}) = \{K_{I_1}(z_1), K_{I_1}(z_2)\} = \{\text{ІСТИНА, ХИБНІСТЬ}\}; \\ O_1 = \{< 1, \text{ІСТИНА}, 6 >, < 1, \text{ ХИБНІСТЬ}, 5 >, < 6, \text{ ІСТИНА}, 1 >, \\ \quad < 6, \text{ ХИБНІСТЬ}, 8 >, < 5, \text{ ІСТИНА}, 10 >, < 5, \text{ ХИБНІСТЬ}, 7 >\}. \end{cases} \quad (2.114)$$

Задамо структурну підалгебру переходів G_{d_1}, ізоморфну абстрактній підалгебрі (2.112) і проміжній алгебрі (2.114). Для цього сформуємо структурні коди станів як чотирирозрядні двійкові вектори, що є цілочисельною беззнаковою формою відповідних проміжних кодів станів (табл. 2.3).

Таблиця 2.3

Кодування станів множини A_{δ_1} (фрагмент ГСА Γ_{2-4})

a_i	a_1	a_2	a_3	a_4	a_5	a_6	a_7
$K_{I_1}(a_i)$	1	6	5	11	8	10	7
$K_S(a_i)$	0001	0110	0101	1011	1000	1010	0111

Тепер структурна підалгебра G_{d_1}, ізоморфна G_{δ_1} та G_{I_1}, задається в такий спосіб:

$$\begin{cases} G_{d_1} = <\{K_S(A_{d_1}), K_S(Z_{d_1})\}, \{d_1\}>; \\ K_S(A_{d_1}) = \{K_S(a_1), ..., K_S(a_7)\} = \\ \qquad = \{0001, 0110, 0101, 1011, 1000, 1010, 0111\}; \\ K_S(Z_{d_1}) = \{K_S(z_1), K_S(z_2)\} = \{1, 0\}; \\ d_1 = \{<0001, 1, 0110>, <0001, 0, 0101>, <0110, 1, 1011>, \\ \qquad <0110, 0, 1000>, <0101, 1, 1010>, <0101, 0, 0111>\}. \end{cases} \tag{2.115}$$

Схемна реалізація операції переходу, заданої виразом (2.113), показана на рис. 2.12.

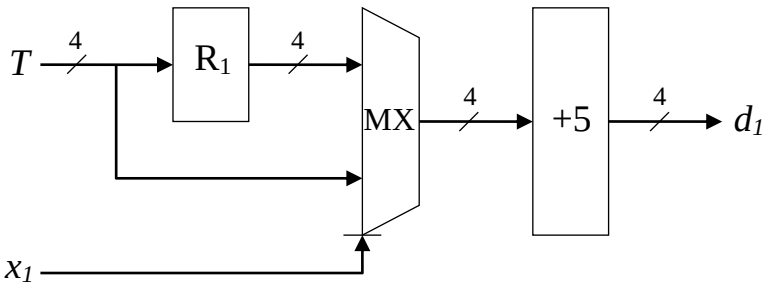


Рисунок 2.12 – Функціональна схема для операції (2.113)

Блок R_1 виконує логічний зсув чотирирозрядного коду поточного стану на один розряд праворуч, що еквівалентне операції ділення на два. Мультиплексор МХ під керуванням структурного сигналу x_1 формує на своєму виході або код T поточного стану, або його половинне значення. Блок «+5» виконує підсумовування вхідного значення з константою $5_{10} = 0101_2$ за правилами

підсумовування беззнакових величин з відкиданням переносу зі старшого розряду. Чотирирозрядний вихід даного блоку ототожнюється зі значенням часткової структурної функції d_1.

Узагальнення 5. *Кількість проміжних алгебр може збігатися з кількістю часткових функцій переходів.*

У розглянутому вище С-автоматі часткова функція δ_2 реалізує ті переходи, які реалізуються без використання проміжної алгебри. Відповідна їй структурна функція переходів d_2, що враховує структурне представлення кодів станів і вхідних сигналів, задається у вигляді системи канонічних рівнянь і реалізується відомим способом у вигляді комбінаційної схеми. Те ж саме можна сказати про функцію δ_2 D-автомата і функцію δ_3 CD-автомата. Таким чином, у розглянутих вище прикладах частина переходів була реалізована канонічним способом без використання проміжної алгебри переходів [42].

Однак у загальному випадку припустима ситуація, коли на базі проміжних алгебр реалізуються всі переходи автомата. Як приклад розглянемо автомат Мура, що задається позначеною ГСА Γ_{2-5} (рис. 2.13). У даній ГСА присутні п'ять типів переходів: безумовний перехід, перехід за умовою x_1, перехід за $\bar{x}_1$, перехід за x_2 та перехід за $\bar{x}_2$. В еквівалентному абстрактному автоматі даним типам переходів відповідають п'ять абстрактних вхідних сигналів $z_0 - z_4$, структурні коди яких є значення вектора $< x_1, x_2 >$: $K_S(z_0) =< -, - >$, $K_S(z_1) =< 0, - >$, $K_S(z_2) =< 1, - >$, $K_S(z_3) =< -, 0 >$, $K_S(z_4) =< -, 1 >$.

Абстрактна алгебра переходів для ГСА Γ_{2-5} задається системою (2.116). Представимо абстрактну функцію переходів δ у вигляді двох часткових функцій δ_1 та δ_2, на основі яких сформуємо дві абстрактні підалгебри переходів (вирази 2.117 та 2.118).

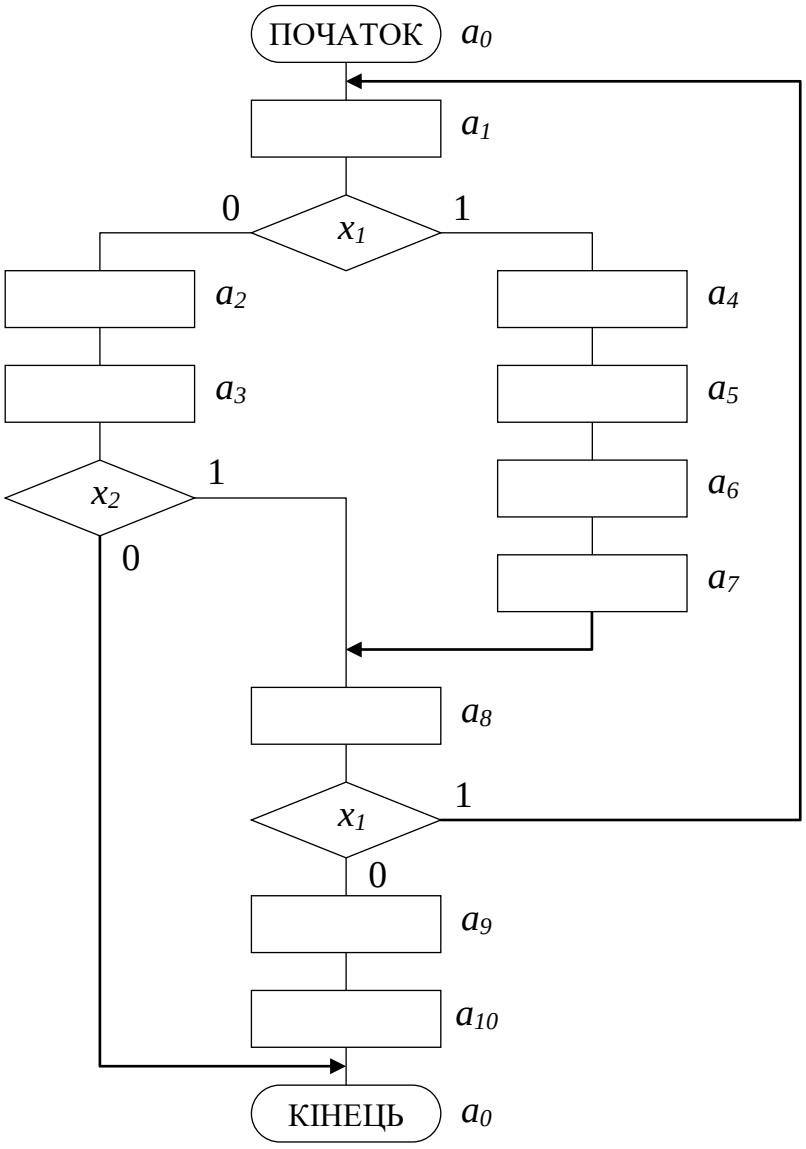


Рисунок 2.13 – Граф-схема алгоритму Γ_{2-5}

$$
\begin{cases}
G_\delta = \ < A_\delta, F_\delta > = < \{A, Z\}, \{\delta\} >; \\
A = \{a_0, ..., a_{10}\}; \\
Z = \{z_0, ..., z_4\}; \\
\delta = \{< a_0, z_0, a_1 >, < a_1, z_1, a_2 >, < a_1, z_2, a_4 >, < a_2, z_0, a_3 >, \\
\qquad < a_3, z_3, a_0 >, < a_3, z_4, a_8 >, < a_4, z_0, a_5 >, < a_5, z_0, a_6 >, \\
\qquad < a_6, z_0, a_7 >, < a_7, z_0, a_8 >, < a_8, z_1, a_9 >, < a_8, z_1, a_9 >, \\
\qquad < a_9, z_0, a_{10} >, < a_{10}, z_0, a_0 >\}.
\end{cases}
\tag{2.116}
$$

$$\begin{cases} G_{\delta_1} = <\{A_{\delta_1}, Z_{\delta_1}\}, \{\delta_1\}>; \\ A_{\delta_1} = \{a_0, ..., a_{10}\}; \\ Z_{\delta_1} = \{z_0, z_1, z_4\}; \\ \delta_1 = \{< a_1, z_1, a_2 >, < a_3, z_4, a_8 >, < a_4, z_0, a_5 >, < a_5, z_0, a_6 >, \\ \quad < a_6, z_0, a_7 >, < a_8, z_1, a_9 >, < a_{10}, z_0, a_0 >\}. \end{cases} \quad (2.117)$$

$$\begin{cases} G_{\delta_2} = <\{A_{\delta_2}, Z_{\delta_2}\}, \{\delta_2\}>; \\ A_{\delta_2} = \{a_0, a_1, a_2, a_3, a_4, a_7, a_8, a_9, a_{10}\}; \\ Z_{\delta_2} = \{z_0, z_1, z_2, z_3\}; \\ \delta_2 = \{< a_0, z_0, a_1 >, < a_1, z_2, a_4 >, < a_2, z_0, a_3 >, < a_3, z_3, a_0 >, \\ \quad < a_7, z_0, a_8 >, < a_8, z_2, a_1 >, < a_9, z_0, a_{10} >\}. \end{cases} \quad (2.118)$$

Задамо проміжну алгебру

$$G_{I_1} = <A_{I_1}, F_{I_1}> = <\{K_{I_1}(A_{\delta_1}), Z_{\delta_1}\}, O_1>, \quad (2.119)$$

ізоморфну підалгебрі (2.117), і проміжну алгебру

$$G_{I_2} = <A_{I_2}, F_{I_2}> = <\{K_{I_2}(A_{\delta_2}), Z_{\delta_2}\}, O_2>, \quad (2.120)$$

ізоморфну підалгебрі (2.118).

При формуванні проміжних алгебр переходів будемо виходити з того, що для наявних одинадцяти станів структурні коди представляються чотирирозрядними двійковими векторами. З урахуванням цього виберемо інтерпретацію структурних кодів станів, що дозволяє задавати проміжні коди станів.

В обох проміжних алгебрах проміжні коди станів будемо задавати із целочисельного діапазону [0; 15]. Даний діапазон відповідає чотирирозрядній двійковій формі структурних кодів станів. З урахуванням діапазону значень проміжних кодів задамо операції переходів O_1 і O_2 в такий спосіб.

Визначимо операцію O_1 виразом (2.121).

$$O_1: \quad K_{I_1}(a_k) = (K_{I_1}(a_i) + 12) \bmod 16. \quad (2.121)$$

Тут «12» – десяткова константа, що додається до коду поточного стану, «mod 16» – операція обчислення залишку від цілочисельного ділення на 16. Наприклад,

якщо $K_{I_1}(a_i)=7$, то $K_{I_1}(a_k)=(7+12)\bmod 16=3$. Помітимо, що для чотирирозрядних структурних кодів станів дія «+12» схемотехнічно реалізується на чотирирозрядному суматорі, а дія «mod 16» забезпечується відкиданням вихідного переносу старшого розряду суматора.

Визначимо операцію O_2 як операцію цілочисельного ділення на 2 з відкиданням залишку:

$$O_2:\quad K_{I_2}(a_k)=K_{I_2}(a_i)/2. \tag{2.122}$$

Схема для даної операції становить собою чотирирозрядну схему зсуву на один розряд праворуч із заповненням вивільнюваного старшого розряду нулем і реалізується шляхом перекомутації проводів без використання логічних елементів.

Застосування обраних операцій для реалізації переходів заданої ГСА можливо тільки при спеціально обраних значеннях проміжних кодів станів. Задамо проміжні коди (а разом і структурні коди) станів так, як показано в табл. 2.4. Оскільки в обох алгебрах переходів структурні коди станів мають однакову інтерпретацію та однакові значення, у таблиці кодам $K_{I_1}(a_i)$ та $K_{I_2}(a_i)$ відповідає єдине значення $K_I(a_i)$.

Таблиця 2.4

Кодування станів ГСА Γ_{2-5}

a_i	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
$K_I(a_i)$	3	1	13	6	0	12	8	4	2	14	7
$K_S(a_i)$	0011	0001	1101	0110	0000	1100	1000	0100	0010	1110	0111

У графічному вигляді основні результати алгебраїчного синтезу – взаємозв'язок обраних операцій переходів і кодів станів – показані на рис. 2.14. Тут всередині кожної вершини, що відповідає окремому стану автомата, зазначені проміжний і структурний коди даного стану, а гілки, відповідні до вихідних

переходів, позначені відповідною операцією переходу. Для стислості операція переходу O_1 позначена як «+12», O_2 – як «/2».

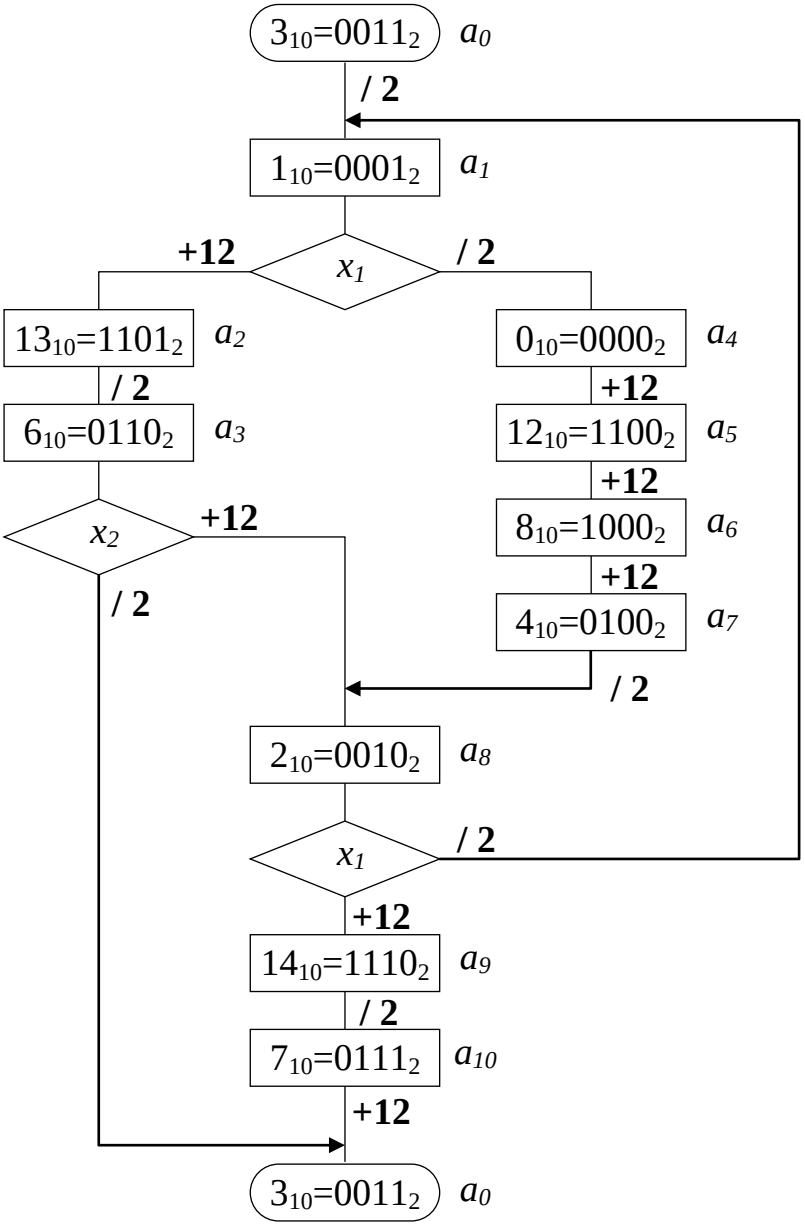


Рисунок 2.14 – Графічне представлення результатів алгебраїчного синтезу мікропрограмного автомата по ГСА Γ_{2-5}

Аналіз рис. 2.14 підтверджує можливість реалізації всіх без винятку переходів ГСА Γ_{2-5} з використанням операцій (2.121) та (2.122). При цьому необхідність у реалізації частини переходів без використання проміжної алгебри (у вигляді системи канонічних рівнянь) відсутня, що підтверджує зроблене вище узагальнення 5.

Помітимо, що в операціях переходів, які були використані в даному прикладі, аргументом є тільки проміжний код поточного стану. Вхідні сигнали автомата не є аргументами операцій переходів, і яка-небудь інтерпретація структурних кодів вхідних сигналів у рамках проміжних алгебр не потрібна. Це дозволяє вважати присутність вхідних сигналів у проміжних алгебрах формальністю, необхідною для встановлення ізоморфізмів між проміжними алгебрами і відповідними їм абстрактною та структурною підалгебрами переходів.

Подальші дії із синтезу автомата розглянемо коротко. Після того, як обрані операції переходів і задані проміжні коди станів, що забезпечують реалізацію переходів автомата відповідно до заданої ГСА, необхідно задати проміжну алгебру (2.119), ізоморфну абстрактній підалгебрі (2.117), і проміжну алгебру (2.120), ізоморфну абстрактній підалгебрі (2.118). Потім мають бути задані структурні підалгебри переходів, ізоморфні відповідним абстрактним підалгебрам і проміжним алгебрам, а зі структурних підалгебр має бути складена структурна алгебра переходів виду (2.55). Дані дії виконуються за аналогією з раніше розглянутими прикладами.

Оскільки для кожного стану автомата використовується єдиний і унікальний структурний код, стає можливим завдання ізоморфізму (2.68). Одержувана в результаті система ізоморфізмів, схожа із системою (2.90), говорить про можливість структурної реалізації автомата. При цьому структура автомата для розглянутого прикладу буде аналогічна структурі, що зображена на рис. 2.3, але без блоку КС_{d_3} .

2.4 Принцип операційного перетворення кодів станів

У дисертаційній роботі центральне місце займає викладений у пп. 2.1-2.3 підхід до побудови функції переходів мікропрограмного автомата, обумовлений наступними положеннями [15, 16, 40, 44]:

1. Структурні коди станів і вхідних сигналів отримують деяку інтерпретацію – скалярну або векторну, виходячи з якої для станів і вхідних сигналів формуються спеціальні проміжні коди.

2. При використанні проміжних кодів станів і вхідних сигналів автоматний перехід (перетворення коду поточного стану в код наступного стану) розглядається як виконання деякої операції (операції переходів) над кодом поточного стану та кодом вхідного сигналу.

3. На множині переходів автомата виділяється підмножина, у якій всі переходи реалізуються за допомогою однієї операції переходів, визначеної на множинах проміжних кодів станів та вхідних сигналів.

4. Структурна імплементація операції переходів становить собою логічну схему, що дозволяє виконувати перетворення структурних кодів станів та вхідних сигналів, еквівалентні перетворенням відповідних проміжних кодів станів та вхідних сигналів за допомогою цієї операції переходів.

5. У загальному випадку на множині переходів автомата виділяється кілька непересічних підмножин, у кожній з яких:

– структурні коди станів і вхідних сигналів інтерпретуються незалежно від інших підмножин переходів;

– проміжні коди станів і вхідних сигналів формуються в спосіб, незалежний від інших підмножин переходів;

– усі переходи реалізуються за допомогою однієї операції переходів, визначеної на множинах проміжних кодів станів та вхідних сигналів даної підмножини переходів;

– апаратурні витрати в логічній схемі, що імплементує операцію переходів, не залежать (або залежать незначно) від кількості переходів у даній підмножині.

6. Переходи, що не належать до жодної зі сформованих підмножин, реалізуються канонічним способом згідно до системи булевих рівнянь.

7. Структурний синтез автомата можливий лише у випадку унікальності структурних кодів станів і вхідних сигналів.

Дані положення формально викладені з використанням математичного апарата теорії універсальних алгебр, згідно з яким механізм реалізації автоматних переходів, заснований на спеціальній інтерпретації й обробці структурних кодів станів і вхідних сигналів, представлений проміжною алгеброю переходів. Оскільки в проміжній алгебрі роль перетворювача кодів станів виконує певна операція, назвемо даний підхід *принципом операційного перетворення кодів станів* [16, 40]. Даний принцип відрізняється від канонічного підходу тим, що використовує правила перетворення кодів станів та вхідних сигналів, що не залежать від їх конкретних значень.

Реалізацію функції переходів мікропрограмного автомата відповідно до принципу операційного перетворення кодів станів будемо називати *операційною реалізацією функції переходів*. Процес формального завдання мікропрограмного автомата із операційною реалізацією функції переходів за допомогою системи ізоморфізмів, що включає ізоморфізми кожної проміжної алгебри та відповідних абстрактної і структурної підалгебр, а також ізоморфізм абстрактної і структурної алгебр переходів, будемо називати *алгебраїчним синтезом функції переходів мікропрограмного автомата*.

2.5 Висновки до другого розділу

1. В другому розділі вирішено наукові завдання розробки загальної концепції представлення функції переходів у вигляді множини часткових функцій, формалізованого узагальнення структурних особливостей відомої структури мікропрограмного автомата на лічильнику та формулювання нового принципу операційного перетворення кодів станів.

2. Математичний апарат теорії універсальних алгебр є одним із засобів формального опису цифрових автоматів. Із його використанням абстрактний автомат та еквівалентний йому структурний автомати можуть бути представлені двохосновними алгебрами – абстрактною алгеброю та структурною алгеброю

відповідно. Еквівалентність абстрактного і структурного автоматів в алгебраїчній формі виражається ізоморфізмом даних алгебр.

3. Виділення в автоматі функцій переходів і виходів дозволяє представити автомат у вигляді системи двох алгебр – алгебри переходів і алгебри виходів. У випадку абстрактного автомата носії даних алгебр утворені станами, абстрактними вхідними і абстрактними вихідними сигналами; у випадку структурного автомата – відповідними структурними кодами. З ізоморфізму абстрактної і структурної алгебр випливають ізоморфізми абстрактної і структурної алгебр переходів та абстрактної і структурної алгебр виходів.

4. Розбивка множини переходів автомата на підмножини рівносильна представленню функції переходів у вигляді декількох часткових функцій. Кожна часткова функція переходів може розглядатися як сигнатура деякої алгебри, що є підалгеброю алгебри переходів. Ізоморфізм абстрактної і структурної алгебр переходів забезпечує ізоморфізми відповідних абстрактної і структурної підалгебр.

5. Прикладом представлення функції переходів у вигляді двох часткових функцій є автомат на лічильнику, в якому частина переходів реалізується за допомогою інкрементора, частина – в канонічний спосіб. Процес перетворення кодів станів за допомогою інкрементора припускає інтерпретацію структурних (двійкових) кодів станів як скалярних величин у цілочисельному діапазоні, що дозволяє формувати лінійні послідовності станів. Множина скалярних кодів станів разом з операцією інкремента утворюють алгебру, ізоморфну до відповідних абстрактної та структурної підалгебр. Дана алгебра відіграє допоміжну роль при побудові структурної підалгебри переходів відповідно до абстрактної підалгебри і в дисертаційній роботі названа проміжною алгеброю переходів. Коди станів і вхідних сигналів, що утворюють носій проміжної алгебри переходів, названі відповідно проміжними кодами станів і проміжними кодами вхідних сигналів, а операція, що утворює сигнатуру проміжної алгебри, названа операцією переходів.

6. У загальному випадку функція переходів автомата може бути представлена більш ніж двома частковими функціями, припускаючи використання двох або більше проміжних алгебр. При цьому структурний синтез автомата можливий за умови, що кожному стану і вхідному сигналу надані єдині та унікальні структурні коди в рамках відповідних множин структурних кодів. При представленні функції переходів у вигляді декількох часткових функцій виконання даної умови забезпечується спільним завданням ізоморфізмів кожної проміжної алгебри переходів і відповідних до неї абстрактної та структурної підалгебр. Процес завдання системи ізоморфізмів, що включає ізоморфізми кожної проміжної алгебри та відповідних абстрактної і структурної підалгебр, а також ізоморфізм абстрактної і структурної алгебр переходів, у дисертаційній роботі названий алгебраїчним синтезом функції переходів мікропрограмного автомата.

7. Алгебраїчна інтерпретація автомата на лічильнику припускає ряд узагальнень, відповідно до яких при алгебраїчному синтезі довільного автомата відсутні обов'язкові вимоги до способу формування часткових функцій переходів і кількості проміжних алгебр, до способів інтерпретації в проміжних алгебрах структурних кодів станів і вхідних сигналів, до вибору операцій переходів та їх аргументів. Відсутність даних вимог ускладнює формалізацію процесу алгебраїчного синтезу автоматів.

8. Підхід, при якому перетворення кодів станів у мікропрограмному автоматі здійснюється за допомогою операцій проміжних алгебр, що використовують певну інтерпретацію структурних кодів станів і вхідних сигналів, названий в дисертаційній роботі принципом операційного перетворення кодів станів. Використання таких операцій переходів, апаратурні витрати в логічній схемі яких не залежать або залежать незначно від кількості переходів, що реалізуються операцією, може сприяти зниженню апаратурних витрат в логічній схемі мікропрограмного автомата у порівнянні з реалізацією функції переходів канонічним методом.

3 ОРГАНІЗАЦІЯ МІКРОПРОГРАМНОГО АВТОМАТА З ОПЕРАЦІЙНИМ АВТОМАТОМ ПЕРЕХОДІВ

3.1 Структурна і математична моделі мікропрограмного автомата з операційним перетворенням кодів станів

Нехай абстрактний автомат (1.1) заданий двома алгебрами: абстрактною алгеброю переходів (2.6) та абстрактною алгеброю виходів (2.7) у випадку автомата Мілі або (2.8) у випадку автомата Мура. При кількості переходів $B>1$ функція переходів δ, що утворює сигнатуру абстрактної алгебри переходів, може бути представлена множиною $\{\delta_1, ..., \delta_{N_\delta}\}$ часткових функцій виду (2.13), де $1< N_\delta \le B$. Це дозволяє ототожнити з алгеброю (2.6) множину $\{G_{\delta_1}, ..., G_{\delta_{N_\delta}}\}$ абстрактних підалгебр переходів G_{δ_i} виду (2.21).

Визначимо структурний автомат як двійку, утворену структурною алгеброю переходів (2.55) і структурною алгеброю виходів (2.56) у випадку автомата Мілі або (2.57) у випадку автомата Мура. Структурна функція переходів d може бути представлена у вигляді множини $\{d_1, ..., d_{N_d}\}$ часткових функцій виду (2.61). Будемо вважати, що для еквівалентних абстрактного та структурного автоматів кількість N_d часткових структурних функцій переходів завжди дорівнює кількості N_δ часткових абстрактних функцій. Це дозволяє задати множину $\{d_1, ..., d_{N_d}\}$ таким чином, щоб при будь-якому $i \in [1;\ N_\delta = N_d]$ часткові функції δ_i та d_i були тотожними, тобто реалізовували одні й ті самі автоматні переходи, хоча й для різних способів представлення станів і вхідних сигналів. Тотожність відповідних часткових абстрактної і структурної функцій переходів забезпечує існування ізоморфізмів $G_{\delta_i} \leftrightarrow G_{d_i}$ для кожного $i \in [1;\ N_\delta]$.

Нехай задана множина $\{G_{I_1}, ..., G_{I_{N_I}}\}$ проміжних алгебр переходів виду

$$G_{I_i} = < A_{I_i}, F_{I_i} > = < \{K_{I_i}(A_{I_i}), K_{I_i}(Z_{I_i})\}, \{O_i\} > . \qquad (3.1)$$

Тут A_{I_i} – носій проміжної алгебри, F_{I_i} – її сигнатура. Носій утворений двома множинами: множиною $K_{I_i}(A_{I_i})$ проміжних кодів станів $a \in A_{I_i}$ та множиною $K_{I_i}(Z_{I_i})$ проміжних кодів вхідних сигналів $z \in Z_{I_i}$. Сигнатура утворена єдиною операцією переходів

$$O_i : \ K_{I_i}(A_{I_i}) \times K_{I_i}(Z_{I_i}) \to K_{I_i}(A_{I_i}), \tag{3.2}$$

яка формально є множиною кортежів виду

$$< K_{I_i}(a_k), K_{I_i}(z_l), K_{I_i}(a_m) > \in O_i . \tag{3.3}$$

Будемо вважати, що для будь-якого автомата кількість N_I проміжних алгебр $N_\delta - 1 \le N_I \le N_\delta$. У випадку $N_I = N_\delta$ всі переходи автомата реалізуються із використанням проміжних алгебр. При $N_I = N_\delta - 1$ існує деяка підмножина переходів, яка визначається частковими функціями δ_{N_δ} та d_{N_d} і реалізується канонічним способом.

Припустимо, що для будь-якого $i \in [1;\ N_I]$ підалгебри G_{δ_i}, G_{d_i} та проміжна алгебра G_{I_i} задані так, що існують їхні попарні ізоморфізми:

$$G_{\delta_i} \leftrightarrow G_{I_i} \leftrightarrow G_{d_i} . \tag{3.4}$$

Вираз (3.4) говорить про те, що часткова абстрактна функція переходів δ_i, операція переходів O_i та часткова структурна функція переходів d_i реалізують одні й ті самі автоматні переходи. Це дозволяє вважати вірним наступне твердження:

У структурному автоматі, еквівалентному заданому абстрактному автомату, структурні коди станів і вхідних сигналів, що утворюють носій структурної підалгебри G_{d_i}, можуть перетворюватися логічною схемою, що імплементує операцію переходів O_i проміжної алгебри G_{I_i}, відповідно до закону, визначеного абстрактною функцією переходів δ_i із сигнатури абстрактної підалгебри G_{δ_i}.

Для N_I проміжних алгебр переходів множина ізоморфізмів виду (3.4) може бути об'єднана в систему:

$$\begin{cases} G_{\delta_1} \leftrightarrow G_{I_1} \leftrightarrow G_{d_1}; \\ G_{\delta_2} \leftrightarrow G_{I_2} \leftrightarrow G_{d_2}; \\ \ldots \\ G_{\delta_{N_I}} \leftrightarrow G_{I_{N_I}} \leftrightarrow G_{d_{N_I}}. \end{cases} \tag{3.5}$$

Як було відзначено в п. 2.2, для забезпечення можливості функціонування структурного автомата має виконуватись вимога одиничності й унікальності структурних кодів станів і вхідних сигналів, що виражається ізоморфізмом (2.68). Разом з тим у випадку $N_I = N_\delta - 1$ даний ізоморфізм буде неявно містити в собі ізоморфізм $G_{\delta_{N_\delta}} \leftrightarrow G_{d_{N_d}}$, відповідальний за реалізацію частини автоматних переходів канонічним способом без використання проміжної алгебри переходів. Таким чином, додавання ізоморфізму (2.68) у систему (3.5) уможливлює структурний синтез функції переходів:

$$\begin{cases} G_{\delta_1} \leftrightarrow G_{I_1} \leftrightarrow G_{d_1}; \\ G_{\delta_2} \leftrightarrow G_{I_2} \leftrightarrow G_{d_2}; \\ \ldots \\ G_{\delta_{N_I}} \leftrightarrow G_{I_{N_I}} \leftrightarrow G_{d_{N_I}}; \\ G_\delta \leftrightarrow G_d. \end{cases} \tag{3.6}$$

Зіставимо кожній частковій структурній функції переходів d_i окрему комбінаційну схему КС_{d_i}. У випадку $N_I = N_d$ кожна КС_{d_i} імплементує операцію переходів O_i із сигнатури проміжної алгебри G_{I_i}. При $N_I = N_d - 1$ комбінаційна схема $\text{КС}_{d_{N_d}}$, відповідна до часткової функції d_{N_d}, синтезується канонічним способом. Хоча ізоморфізм $G_{\delta_{N_\delta}} \leftrightarrow G_{d_{N_d}}$ встановлюється без використання проміжної алгебри, ніщо не заважає розглядати часткову структурну функцію d_{N_d} як окрему операцію переходів O_{N_d}, що утворює сигнатуру підалбери $G_{d_{N_d}}$ і визначена на множині структурних (не проміжних!) кодів станів з її носія. Це

дозволяє умовно вважати кількість операцій переходів завжди рівною кількості N_d часткових структурних функцій переходів.

У загальному випадку на вхід кожної $КС_{d_i}$ надходять *R*-розрядний структурний код $T =< T_1, ..., T_R >$ поточного стану автомата і множина структурних вхідних сигналів $X = \{x_1, ..., x_L\}$. На виході кожної схеми $КС_{d_i}$ формується значення часткової структурної функції переходів d_i, що являє собою *R*-розрядний структурний код стану переходу.

Організуємо вибір часткової функції переходів за аналогією з CD-автоматом (рис. 2.3). До кожної часткової структурної функції d_i поставимо у відповідність код $K(d_i)$, що представляється структурними сигналами $z_1, ..., z_{R_Z}$, де $R_Z = \lceil \log_2(N_d) \rceil$. Вектор $Z =< z_1, ..., z_{R_Z} >$ будемо використовувати для мультиплексування множини *R*-розрядних значень часткових структурних функцій переходів. Одержувана в результаті структурна схема зображена на рис. 3.1. Оскільки дана схема являє собою, по суті, реалізацію множини операцій переходів, назвемо дану схему *операційною частиною* (ОЧ) [44, 45].

Відповідно до даної схеми структурна функція переходів *d* визначається, за аналогією з виразом (2.75), у такий спосіб:

$$d = d(d_1, ..., d_{N_d}, Z). \qquad (3.7)$$

Представимо операційну частину у вигляді окремого блоку. Структурний код стану переходу з виходу ОЧ подамо на вхід регістру пам'яті РП. Вихід РП подамо у якості зворотного зв'язку на вхід ОЧ (рис. 3.2).

Порівняння структури на рис. 3.2 зі структурою, зображеною на рис. 1.3, дозволяє розглядати композицію блоків ОЧ та РП як *операційний автомат*. У порівнянні із традиційною структурою даний ОА має наступні відмінні риси:

1. Множина функцій операційного автомата представлена множиною структурних функцій переходів $d_1 - d_{N_d}$. Оскільки кожній структурній функції d_i відповідає операція переходів O_i, множину функцій ОА на рис. 3.2 можна ототожнювати з множиною операцій переходів.

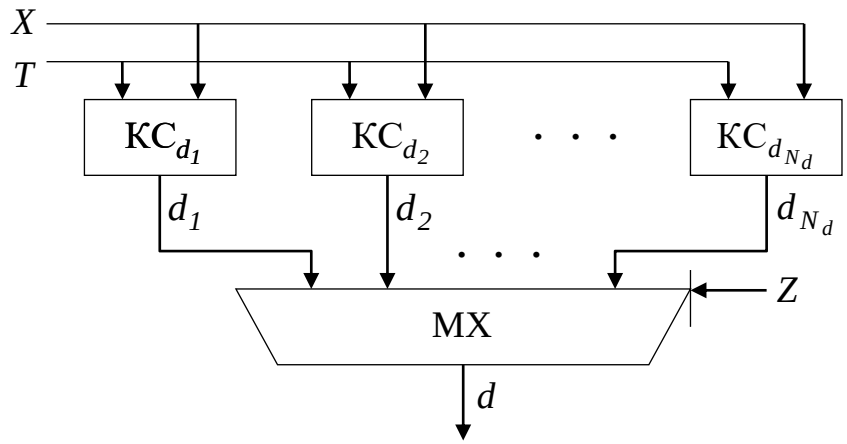


Рисунок 3.1 – Структурна схема операційної частини

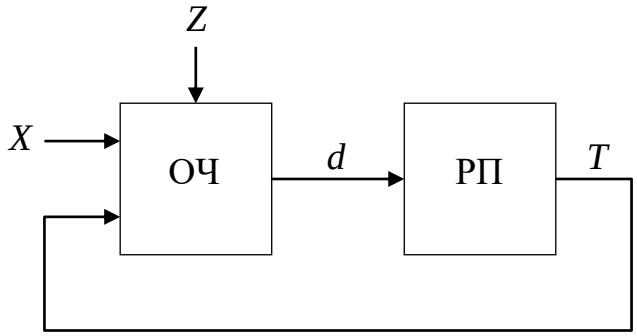


Рисунок 3.2 – Композиція операційної частини і регістра пам’яті

2. Пам'ять автомата представлена єдиним регістром РП, що виступає у кожній виконуваній операції в якості регістру вхідних даних та регістру результату.

3. Роль керуючих сигналів виконують сигнали *Z*, які застосовуються до мультиплексора результату МХ і становлять, таким чином, код операції переходів.

4. У традиційному ОА процес обробки даних припускає виникнення різних помилкових ситуацій (переповнення, ділення на нуль тощо). Подібні ситуації знаходять відображення у множині сигналів логічних умов, що формуються у схемі на рис. 1.3 спеціальною Ψ-підсхемою та аналізуються надалі логічною схемою керуючого автомата. На відміну від універсального ОА, блок ОЧ завжди проектується для конкретної ГСА таким чином, щоб виникнення в процесі обробки кодів станів яких-небудь ситуацій, що вимагають аналізу, було

неможливим. Як наслідок, блок ОЧ не формує ніяких ознак результату виконання операцій, і в його структурі на рис. 3.1 схема формування логічних умов відсутня.

5. У якості зовнішніх даних виступають сигнали логічних умов *X*. Відповідно до узагальнення 4 (параграф 2.3), обмеження на їхню інтерпретацію і способи використання в загальному випадку відсутні.

6. Результат *R*, що формується на рис. 1.3 блоком Ф, у схемі на рис. 3.2 окремо не виділяється і збігається з виходом *d* блоку ОЧ. Якщо результат роботи традиційного ОА може бути у разі потреби переданий у зовнішню схему відразу з виходу функціонального блоку без попереднього завантаження в один зі своїх регістрів, то результат роботи блоку ОЧ завжди завантажується в регістр РП, бо ніякими зовнішніми схемами не використовується.

Оскільки роль блоку ОЧ полягає в перетворенні структурних кодів станів за допомогою певної множини операцій переходів, назвемо ОА, зображений на рис. 3.2, *операційним автоматом переходів* (ОАП) [27, 44, 186, 187]. Проведені структурні аналогії між традиційним операційним автоматом і операційним автоматом переходів підкреслює рис. 3.3.

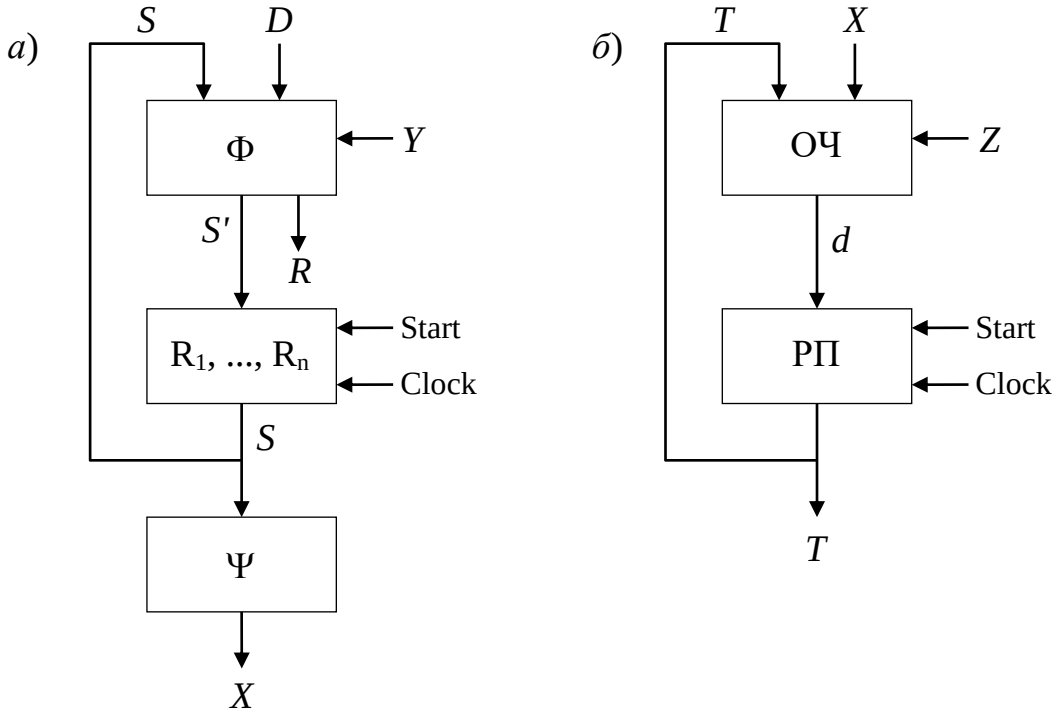


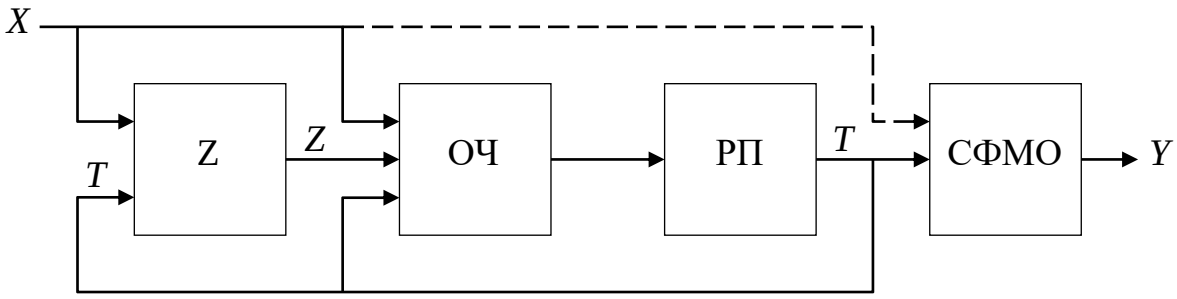
Рисунок 3.3 – Структурна аналогія між традиційним операційним автоматом (*а*) та операційним автоматом переходів (*б*)

Для формування R_Z-розрядного коду Z будемо використовувати спеціальну Z-підсхему за аналогією з CD-автоматом (рис. 2.3). Z-підсхема реалізує функцію

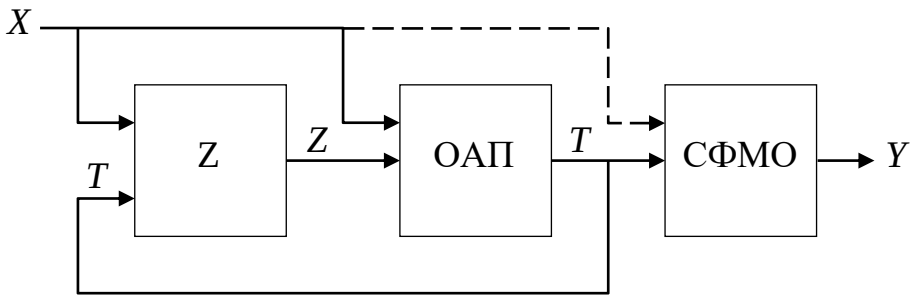
$$Z = Z(T,\ X) \tag{3.8}$$

і задається системою канонічних рівнянь.

Додавання в структурну схему, зображену на рис. 3.2, Z-підсхеми та схеми формування мікрооперацій (СФМО) приводить до структури, що є узагальненням структури CD-автомата для N_d часткових структурних функцій переходів (рис. 3.4). На рис. 3.4, *а* операційний автомат переходів показаний у вигляді композиції блоків ОЧ та РП, на рис. 3.4, *б* – у вигляді єдиного блоку. Назвемо дану структуру *мікропрограмним автоматом з операційним автоматом переходів* (МПА з ОАП) [15, 16, 27, 44] і умовимося в дисертаційній роботі позначати її символом U_1.



а)



б)

Рисунок 3.4 – Структурна схема мікропрограмного автомата
с операційним автоматом переходів (структура U_1)
(*а* – ОАП у вигляді композиції «ОЧ+РП», *б* – у вигляді єдиного блоку)

Наявність у структурі U_1 зв'язку, показаного пунктиром, дозволяє класифікувати її як автомат Мілі, відсутність зв'язку – як автомат Мура. Наявність Z-підсхеми, що будується за системою канонічних рівнянь, дозволяє класифікувати МПА з ОАП як автомат з «жорсткою» логікою.

Використання в структурі U_1 схеми формування мікрооперацій вимагає додавання в систему (3.6) ізоморфізму абстрактної та структурної алгебр виходів, що приводить до системи (3.9).

$$\begin{cases} G_{\delta_1} \leftrightarrow G_{I_1} \leftrightarrow G_{d_1}; \\ G_{\delta_2} \leftrightarrow G_{I_2} \leftrightarrow G_{d_2}; \\ \ldots \\ G_{\delta_{N_I}} \leftrightarrow G_{I_{N_I}} \leftrightarrow G_{d_{N_I}}; \\ G_{\delta} \leftrightarrow G_{d}; \\ G_{\lambda} \leftrightarrow G_{l}. \end{cases} \qquad (3.9)$$

Дану систему пропонується розглядати в якості *математичної моделі мікропрограмного автомата з операційним автоматом переходів* [13, 16, 19, 45]. Якщо система (3.9) задана, то:

– задані абстрактна і структурна алгебри переходів, причому таким чином, що переходи автомата реалізуються за допомогою множини операцій переходів;

– задані абстрактна і структурна алгебри виходів, носії яких збігаються з однойменними носіями абстрактної та структурної алгебр переходів відповідно.

Інакше кажучи, сформована система ізоморфізмів (3.9) говорить про те, що для деякого абстрактного автомата існує еквівалентний йому структурний автомат, побудований на основі принципу операційного перетворення кодів станів.

3.2 Структурна організація операційного автомата переходів

Проаналізуємо структуру операційного автомата переходів з позицій теорії, викладеної в главах 6 і 7 роботи [125]. Основним у даному аналізі є той факт, що в

структурі ОАП є присутнім тільки один регістр результату, представлений регістром пам'яті.

Відповідно до [125] будь-яка мікрооперація містить (у синтаксичному змісті) оператор присвоювання, ліворуч від якого перебуває один з регістрів результату. Оскільки в ОАП регістром результату завжди є РП, усі мікрооперації містять РП у лівій частині оператора присвоювання і відрізняються один від іншого тільки правою частиною. Згідно зі структурою блоку ОЧ (рис. 3.1), праворуч від оператора присвоювання може бути одна із часткових структурних функцій переходів. Таким чином, в ОАП кожна мікрооперація представляється однією з функцій d_i, а кількість мікрооперацій завжди дорівнює N_d.

Оскільки пам'ять ОАП одночасно є пам'яттю мікропрограмного автомата, один такт роботи ОАП збігається з одним тактом роботи МПА. Одиничність регістру результату дозволяє ОАП виконувати в кожному такті тільки одну мікрооперацію, реалізуючи в такий спосіб один перехід МПА. Неможливість виконання декількох МО в одному такті роботи ОАП узгоджується з неможливістю виконання мікропрограмним автоматом декількох переходів в одному такті своєї роботи. Як наслідок, в ОАП ніякі МО не можуть бути сумісними [125]. Функціональна сумісність мікроопераций неможлива внаслідок обмеження на виконання тільки однієї МО в кожному такті роботи ОАП, структурна сумісність неможлива внаслідок наявності в ОАП єдиного регістру результату.

Для довільного ОА припустима різна структурна організація [27, 42, 44]. Так, в І-автоматі використовується принцип закріплення комбінаційних схем, що використовуються для виконання мікроопераций, за кожним з регістрів результату. Структурна організація М-автоматів базується на узагальненні комбінаційних схем стосовно всіх регістрів результату. Наявність в ОАП єдиного регістру результату наділяє структуру ОАП ознаками, характерними для І-автоматів та М-автоматів одночасно. Дійсно, в ОАП за регістром РП закріплена власна множина комбінаційних схем, які не використовуються для виконання мікроопераций з іншими регістрами ОА. У той же час множина комбінаційних

схем ОАП є загальною для всіх наявних регістрів результату. Таким чином, ОАП можна класифікувати одночасно як І-автомат та як М-автомат. В теорії дані класи ОА мають діаметрально протилежні властивості, однак у структурі з єдиним регістром результату ці відмінності нівелюються.

У той же час ОАП не може бути віднесений до класу ІМ-автоматів, що володіють проміжними властивостями між автоматами І- та М-типів. Як відомо, основний акцент у структурі ІМ-автоматів робиться на функціональну сумісність мікрооперацій, що забезпечує можливість виконання декількох МО в одному такті своєї роботи [125]. Оскільки в ОАП мікрооперації не мають ані функціональну, ані структурну сумісність, характерні риси, властиві ІМ-автоматам, у структурі ОАП відсутні.

Також ОАП може бути класифікований як операційний автомат з канонічною структурою (К-автомат). Однак така властивість К-автоматів, як максимальна в порівнянні з іншими структурними організаціями продуктивність, тут не проявляється, оскільки за один такт ОАП здатний виконати лише одну мікрооперацію.

Структурна оптимізація операційних автоматів, що полягає в побудові і розширенні бібліотеки структурних моделей автоматів, охоплює також питання синтезу і оптимізації комбінаційних схем, що є основою цифрових пристроїв. Проектуванню схем ОА присвячена безліч праць, основні результати яких так чи інакше відбиті в роботах [4, 7, 35, 96, 103, 124, 125, 132, 140, 142, 147, 148, 149, 154, 155, 160, 164, 166, 170, 221]. Так, для зменшення витрат апаратури в схемі ОАП можуть бути застосовані поняття еквівалентних мікрооперацій і узагальнених операторів, викладені в [125]. Виділення в операціях переходів однакових функцій дозволяє вважати відповідні їм мікрооперації еквівалентними і представляти їх у формі узагальнених операторів.

Розглянемо приклад. Нехай у МПА з ОАП використано три проміжні алгебри переходів, відповідні сигнатури яких утворені наступними операціями переходів $O_1 - O_3$:

$$O_1 : K_{I_1}(a^{t+1}) := K_{I_1}(a^t) + 2; \tag{3.10}$$

$$O_2 : K_{I_2}(a^{t+1}) := K_{I_2}(a^t) + 2 \cdot K_{I_2}(a^t); \tag{3.11}$$

$$O_3 : K_{I_3}(a^{t+1}) := \overline{K_{I_3}(a^t)} + 1. \tag{3.12}$$

Тут a^t – поточний стан автомата, a^{t+1} – стан переходу.

В ОАП операціям $O_1 - O_3$ відповідають часткові структурні функції переходів $d_1 - d_3$, при реалізації яких поточний стан і стан переходу ототожнюються з регістром пам'яті РП. Це дозволяє представити вираження (3.10) – (3.12) у наступному вигляді:

$$d_1 : РП := РП + 2; \tag{3.13}$$

$$d_2 : РП := РП + 2 \cdot РП; \tag{3.14}$$

$$d_3 : РП := \overline{РП} + 1. \tag{3.15}$$

Даним функціям може бути зіставлений узагальнений оператор виду

$$d_{1-3} : РП := A_1 + A_2, \tag{3.16}$$

де $A_1 = \begin{cases} РП \text{ при } Z = K(d_1); \\ РП \text{ при } Z = K(d_2); \\ \overline{РП} \text{ при } Z = K(d_3), \end{cases}$ $A_2 = \begin{cases} 2 \text{ при } Z = K(d_1); \\ 2 \cdot РП \text{ при } Z = K(d_2); \\ 1 \text{ при } Z = K(d_3). \end{cases}$

Узагальненому оператору (3.16) в блоці ОЧ відповідає наступний фрагмент функціональної схеми:

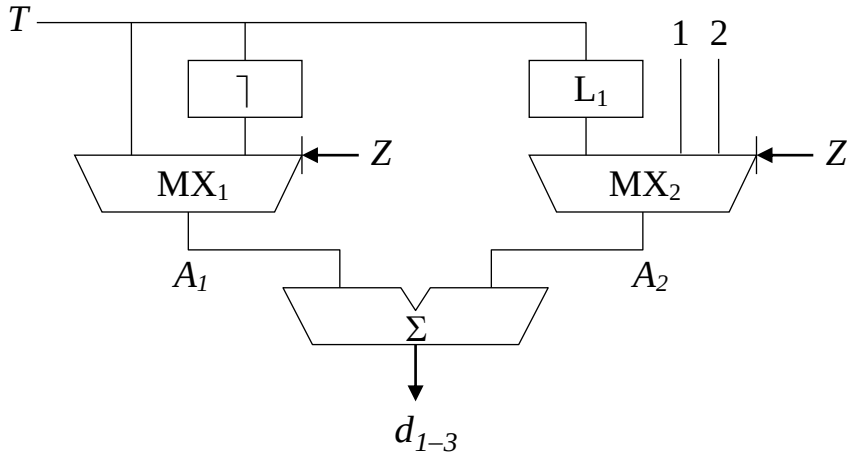


Рисунок 3.5 – Фрагмент функціональної схеми ОЧ,
що відповідає узагальненому оператору (3.16)

У даній схемі мультиплексори $МХ_1$ та $МХ_2$ управляються кодом операції переходів Z і формують, відповідно, аргументи A_1 та A_2 оператора (3.16). Блок «¬» виконує порозрядну інверсію R-розрядного коду поточного стану T; блок «L_1» виконує логічний зсув ліворуч, реалізуючи в такий спосіб арифметичне множення на 2. На вхід $МХ_2$ подаються також константи «1» і «2», представлені в R-розрядному двійковому форматі.

3.3 Формування кодів операцій переходів

3.3.1 Базова організація схеми формування кодів операцій переходів

Сигнали $Z = \{z_1, ..., z_{R_Z}\}$, що надходять із Z-підсхеми в блок ОЧ, виконують в ОАП функцію керуючих сигналів. При цьому Z-підсхема стосовно ОАП виступає в якості керуючого автомата, хоча й не є таким за своєю структурою через відсутність власної пам'яті. Z-підсхему слід розглядати разом із блоком ОЧ як єдину комбінаційну схему, що реалізує функцію переходів відповідно до виразу (3.7). Як і всяка комбінаційна схема, Z-підсхема може задаватися системою канонічних рівнянь наступного виду:

$$\begin{cases} z_1 = z_1(T_1, ..., T_R, x_1, ..., x_L); \\ z_2 = z_2(T_1, ..., T_R, x_1, ..., x_L); \\ ... \\ z_{R_Z} = z_{R_Z}(T_1, ..., T_R, x_1, ..., x_L), \end{cases} \quad (3.17)$$

де $T_1, ..., T_R$ – структурний код поточного стану; $x_1, ..., x_L$ – множина сигналів логічних умов; $z_1, ..., z_{R_Z}$ – множина сигналів коду операції переходів.

Дана система схожа із системою (1.12) для схеми формування переходів (СФП) канонічної структури МПА. Для обох систем максимальна кількість термів у кожному рівнянні дорівнює числу переходів автомата, однак кількість рівнянь у системах у загальному випадку різна.

Рівняння системи (3.17) можуть бути отримані зі спеціальної таблиці, яка відбиває залежність коду ОП від структурного коду поточного стану та логічних умов і містить наступні стовпці:

a_m – поточний стан автомата;

$K_S(a_m)$ – структурний код поточного стану;

a_s – стан переходу;

$K_S(a_s)$ – структурний код стану переходу;

X_h – вхідні змінні з множини $X = \{x_1, ..., x_L\}$, що набувають одиничне значення при переході зі стану a_m в стан a_s;

O_h – операція переходів, що реалізує перехід зі стану a_m в стан a_s за умовою X_h;

z_h – змінні з множини $z = \{z_1, ..., z_{R_Z}\}$, що набувають одиничне значення в коді операції переходу, вказаної в стовпці O_h;

h – порядковий номер рядка таблиці.

Будемо називати таблицю з розглянутою структурою *операційною таблицею переходів* (ОТП) мікропрограмного автомата з операційним автоматом переходів. Кожний рядок ОТП відповідає одному переходу автомата і визначає операцію переходів, яка повинна бути виконана над кодом $K_S(a_m)$ для його перетворення в код $K_S(a_s)$ при переході зі стану a_m в стан a_s за умовою X_h. У випадку представлення рівнянь системи (3.17) у формі ДНФ, по кожному рядку ОТП формується один кон'юнктивний терм виду

$$F_h = A_m^h X_h, \tag{3.18}$$

де A_m^h – кон'юнкція змінних $T_1, ..., T_R$, відповідна до коду $K_S(a_m)$ з рядка з номером h, X_h – кон'юнкція змінних із множини X, відповідна до вмісту стовпця X_h даного рядка. При цьому рівняння для функції z_i системи (3.17) визначається виразом

$$z_i = \bigvee_{h=1}^{H} C_{ih} F_h = \bigvee_{h=1}^{H} C_{ih} A_m^h X_h, \tag{3.19}$$

де C_{ih} – булева змінна, яка дорівнює одиниці у випадку, якщо й тільки якщо в стовпці Z_h h-го рядка таблиці записана функція z_i.

Складемо операційну таблицю переходів для прикладу, розглянутого в п. 2.3 для узагальнення 5. Закодуємо операції переходів O_1 та O_2 однорозрядними двійковими кодами виду $< z_1 >$ в такий спосіб: $K_Z(O_1)=0$, $K_Z(O_1)=1$. Одержувана в результаті ОТП представлена таблицею 3.1 і містить 14 рядків згідно з числом переходів ГСА Γ_{2-5} (рис. 2.14).

Відповідно до побудованої ОТП система (3.17) представляється єдиним виразом для змінної z_1:

$$z_1 = T_1T_2\overline{T}_3\overline{T}_4 \vee \overline{T}_1\overline{T}_2\overline{T}_3\overline{T}_4x_1 \vee T_1T_2\overline{T}_3T_4 \vee \overline{T}_1T_2T_3\overline{T}_4\overline{x}_2 \vee$$
$$\vee \overline{T}_1T_2\overline{T}_3\overline{T}_4 \vee \overline{T}_1\overline{T}_2T_3\overline{T}_4x_1 \vee T_1T_2T_3\overline{T}_4.$$

Таблиця 3.1

Операційна таблиця переходів МПА зі структурою U_1 (ГСА Γ_{2-5})

a_m	$K_S(a_m)$	a_s	$K_S(a_s)$	X_h	O_h	z_h	h
a_0	0011	a_1	0001	1	O_2	z_1	1
a_1	0001	a_2	1101	$\overline{x}_1$	O_1	–	2
		a_4	0000	x_1	O_2	z_1	3
a_2	1101	a_3	0110	1	O_2	z_1	4
a_3	0110	a_8	0010	x_2	O_1	–	5
		a_0	0011	$\overline{x}_2$	O_2	z_1	6
a_4	0000	a_5	1100	1	O_1	–	7
a_5	1100	a_6	1000	1	O_1	–	8
a_6	1000	a_7	0100	1	O_1	–	9
a_7	0100	a_8	0010	1	O_2	z_1	10
a_8	0010	a_1	0001	x_1	O_2	z_1	11
		a_9	1110	$\overline{x}_1$	O_1	–	12
a_9	1110	a_{10}	0111	1	O_2	z_1	13
a_{10}	0111	a_0	0011	1	O_1	–	14

Як видно з табл. 3.1, операційна таблиця переходів за своєю структурою схожа із прямою структурною таблицею (ПСТ) мікропрограмного автомата [29], однак має одна принципову відмінність.

Пряма структурна таблиця є розширенням таблиці переходів МПА і використовується для побудови системи рівнянь (1.12). Методика синтезу МПА з канонічною структурою припускає довільне завдання кодів станів за умови їх унікальності. При обраних структурних кодах станів процес побудови ПСТ є тривіальним і виконується для заданої ГСА єдиним можливим способом. Це дозволяє розглядати пряму структурну таблицю як один зі способів завдання мікропрограмного автомата, обумовлений вхідною ГСА і не залежний суттєво від вибору структурних кодів станів.

У МПА з ОАП функція переходів вважається заданою тоді, коли формування структурних кодів станів, вибір операцій переходів та їх зіставлення переходам автомата виконані в такий спосіб, щоб забезпечити коректну реалізацію всіх переходів. Виконання цих умов неможливе без попереднього виконання алгебраїчного синтезу функції переходів автомата. У зв'язку із цим операційну таблицю переходів слід розглядати не як вхідні дані для синтезу МПА з ОАП, а як результат одного з етапів синтезу. Інформація, що міститься в ОТП, є вхідною для синтезу Z-підсхеми, однак не є вхідною для синтезу МПА з ОАП у цілому, оскільки не визначається однозначно заданою ГСА.

Відзначимо наступне. При синтезі ОАП теоретично можливий випадок, коли кожний автоматний перехід реалізується окремою операцією переходів, тобто $N_d = B$. У цьому випадку блок ОЧ буде складатися з B комбінаційних схем, для мультиплексування яких потрібен двійковий код розрядності $R_Z = \log_2 B$. Оскільки при наявності в ГСА хоча б однієї умовної вершини кількість переходів B завжди більше числа станів M, у випадку $R_Z = \log_2 B$ величина R_Z може виявитися більше розрядності R коду стану. Більше число рівнянь у системі (3.17) у порівнянні із системою (1.12) при однаковому (без урахування мінімізації) числі термів приводить до того, що витрати апаратури в

одній лише Z-підсхемі виявляються вищими за витрати в схемі формування переходів канонічного автомата. Якщо до витрат в Z-підсхемі додати витрати в блоці ОЧ, який при $N_d = B$ містить максимально можливу кількість комбінаційних схем і мультиплексор відповідної складності, то можна стверджувати, що при $R_Z > R$ використання структури U_1 є недоцільним, оскільки апаратурні витрати в його схемі будуть вищими за МПА з канонічною структурою.

Також недоцільність використання МПА з ОАП прогнозована у випадку $R_Z = R$. При рівній кількості рівнянь у системах (1.12) та (3.13) апаратурні витрати в Z-підсхемі будуть близькі до витрат в схемі СФП канонічного автомата, однак додавання до них витрат у схемі блоку ОЧ залишає МПА з ОАП у програші.

Таким чином, при синтезі Z-підсхеми за системою рівнянь (3.17) виграш в апаратурних витратах у МПА зі структурою U_1 у порівнянні з канонічним автоматом можливий лише при виконанні умови

$$R_Z < R. \tag{3.20}$$

Дана умова не є достатньою, оскільки не враховує витрати апаратури в схемі блоку ОЧ. Позначимо через $H_Z^{U_1}$ чисельно виражені витрати апаратури в Z-підсхемі, через $H_{ОЧ}^{U_1}$, $H_{РП}^{U_1}$, $H_{СФП}^{U_1}$, $H_{СФМО}^{U_1}$ – витрати відповідно у блоках ОЧ, РП, СФП та СФМО схеми МПА зі структурою U_1; $H_{СФП}$, $H_{РП}$, $H_{СФМО}$ – витрати у відповідних блоках канонічної структури МПА. Складемо нерівність, у якій в лівій частині знаходяться сумарні витрати апаратури в схемі МПА з ОАП, у правій частині – витрати в схемі канонічного МПА:

$$H_Z^{U_1} + H_{ОЧ}^{U_1} + H_{РП}^{U_1} + H_{СФМО}^{U_1} < H_{СФП} + H_{РП} + H_{СФМО}. \tag{3.21}$$

Вважаючи, що $H_{СФМО} = H_{СФМО}^{U_1}$ та $H_{РП} = H_{РП}^{U_1}$, маємо наступну умову, що забезпечує перевагу МПА з ОАП перед МПА з канонічною структурою за критерієм апаратурних витрат:

$$H_Z^{U_1} + H_{ОЧ}^{U_1} < H_{СФП}. \tag{3.22}$$

Для зменшення значення $H_{ОЧ}^{U_1}$ можуть бути використані різні методи оптимізації схем операційних автоматів [125], розгляд яких стосовно до ОЧ лежить за рамками даної дисертаційної роботи. Разом з тим для зниження апаратурних витрат у лівій частині нерівності (3.22) у даній роботі пропонується ряд підходів, що враховують специфіку МПА з ОАП:

– зіставлення операцій переходів станам автомата;

– заміна вхідних змінних;

– зменшення максимальної кількості вхідних змінних, що суттєво впливають на автоматні переходи.

Розглянемо ці підходи.

3.3.2 Зіставлення операцій переходів станам автомата

Зниження апаратурних витрат в Z-підсхемі можливе за рахунок зменшення числа її входів [16, 28]. Виключимо з рівнянь системи (3.17) аргументи x_i, в результаті чого одержимо наступну систему рівнянь:

$$\begin{cases} z_1 = z_1(T_1, ..., T_R); \\ z_2 = z_2(T_1, ..., T_R); \\ ... \\ z_{R_Z} = z_{R_Z}(T_1, ..., T_R). \end{cases} \tag{3.23}$$

У даній системі код $Z =< z_1, ..., z_{R_Z} >$ операції переходу формується на підставі лише коду поточного стану автомата. Якщо у випадку системи (3.17) операція переходів ставиться у відповідність окремому переходу автомата, то у випадку (3.23) операція переходів зіставляється окремому стану автомата незалежно від того, скільки переходів існує з даного стану. Таким чином, функція Z-підсхеми визначається наступним виразом:

$$Z = Z(T). \tag{3.24}$$

Структура МПА з ОАП, зображена на рис. 3.4, *б*, у випадку побудови Z-підсхеми за системою (3.23), набуває вигляд рис. 3.6. У рамках дисертації дану структуру будемо позначати символом U_2. Відзначимо, що відсутність сигналів X

на вході Z-підсхеми уможливлює її реалізацію в базисі запам'ятовувальних пристроїв (ПЗП, ППЗП, блочна пам’ять FPGA).

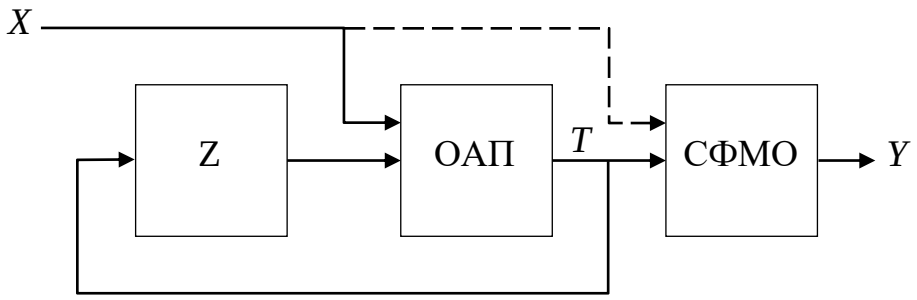


Рисунок 3.6 – Структурна схема МПА з ОАП із Z-підсхемою, що будується за системою рівнянь (3.23) (структура U_2)

Операційна таблиця переходів для структури U_2 відрізняється від ОТП для структури U_1 відсутністю стовпця X_h. При цьому вміст стовпця O_h визначається тільки вмістом стовпця a_m. Для випадку представлення рівнянь системи (3.23) у формі ДНФ по кожному рядкові ОТП формується кон’юнктивний терм виду

$$F_h = A_m^h, \tag{3.25}$$

а рівняння для функцій z_i системи (3.23), за аналогією з (3.19), визначаються виразом

$$z_i = \bigvee_{h=1}^{H} C_{ih} F_h = \bigvee_{h=1}^{H} C_{ih} A_m^h. \tag{3.26}$$

Розглянемо приклад, що підтверджує можливість викладеного підходу. Нехай автомат заданий ГСА Γ_{3-1}, що зображена на рис. 3.7. Проведемо алгебраїчний синтез ОАП коротко, вказавши лише його результати та опускаючи формальне завдання ізоморфізмів абстрактних і структурних подалгебр та проміжних алгебр переходів.

Закодуємо стани $a_0 - a_5$ проміжними кодами із множини цілих чисел у діапазоні $[0-7]$. Нехай $K_I(a_0)=2$, $K_I(a_1)=4$, $K_I(a_2)=6$, $K_I(a_3)=1$, $K_I(a_4)=7$, $K_I(a_5)=5$. Структурні коди станів $<T_1,T_2,T_3>$ визначимо рівними до відповідних проміжних кодів, представлених у форматі трьохрозрядних

двійкових беззнакових чисел: $K_S(a_0)=010$, $K_S(a_1)=100$, $K_S(a_2)=110$, $K_S(a_3)=001$, $K_S(a_4)=111$, $K_S(a_5)=101$.

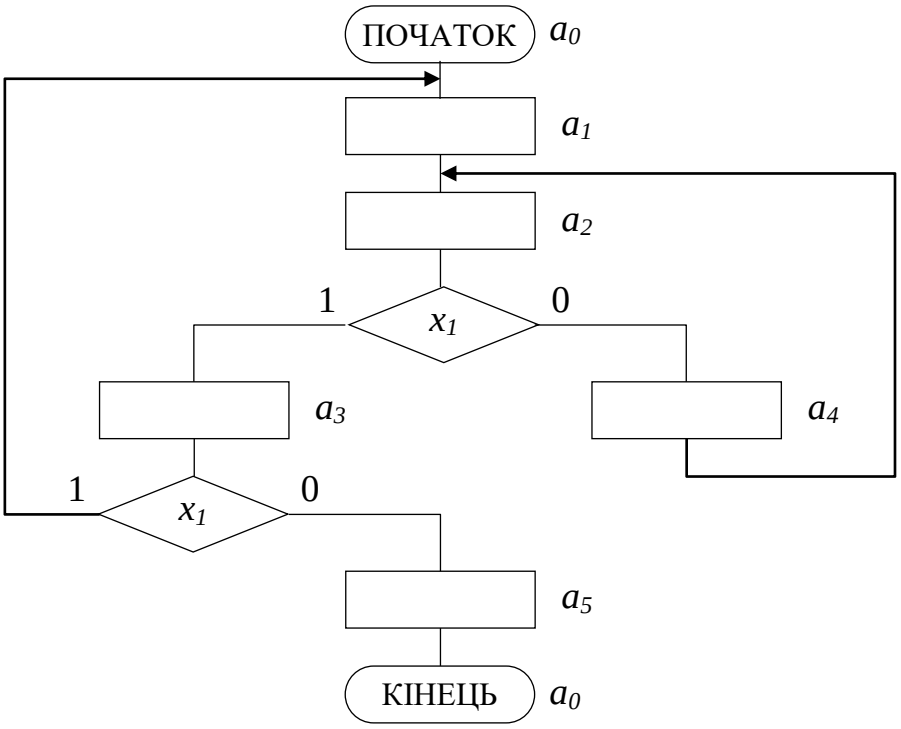


Рисунок 3.7 – Граф-схема алгоритму Γ_{3-1}

Задамо наступні операції переходів, визначені на множині проміжних кодів станів:

$$O_1: \quad K_I(a^{t+1}) = K_I(a^t) + 2. \tag{3.27}$$

$$O_2: \quad K_I(a^{t+1}) = K_I(a^t) \times 2. \tag{3.28}$$

$$O_3: \quad K_I(a^{t+1}) = \begin{cases} K_I(a^t) \times 2 + 3, \text{ якщо } x_1 = 0; \\ K_I(a^t) + 3, \text{ якщо } x_1 = 1. \end{cases} \tag{3.29}$$

У виразах (3.27) – (3.29) a^t – поточний стан автомата, a^{t+1} – стан переходу. Також будемо вважати, що результат кожної ОП завжди приводиться до діапазону [0, 7], що математично виражається операцією «mod 8» (взяття залишку від цілочисельного ділення на 8). При схемній реалізації операцій переходів операція «mod 8» реалізується автоматично за умови, що всі функціональні блоки

(множення, додавання та ін.) будуються із розрахунку обробки трьохрозрядних двійкових кодів.

Зіставимо кожному стану автомата одну з операцій переходів так, як показано в табл. 3.2.

Таблиця 3.2

Зіставлення операцій переходів станам автомата (ГСА Γ_{3-1})

a_i	a_0	a_1	a_2	a_3	a_4	a_5
O_j	O_1	O_1	O_3	O_3	O_2	O_2

Подібне зіставлення дозволяє реалізувати всі автоматні переходи, які містяться в ГСА Γ_{3-1}, за допомогою ОП $O_1 - O_3$, що ілюструє рис. 3.8.

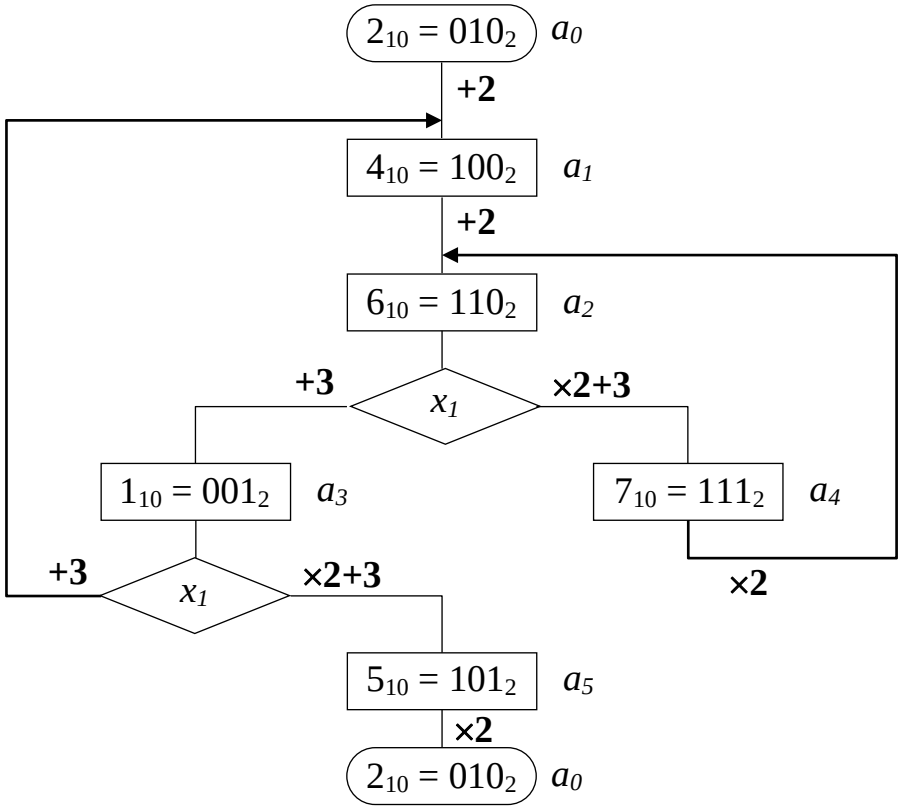


Рисунок 3.8 – Графічне представлення результатів алгебраїчного синтезу МПА з ОАП (Γ_{3-1})

Наприклад, перехід з a_4 в a_2 виконується шляхом множення на 2 структурного коду $K_S(a_4) = 111$ з наступним відкиданням старшого розряду

результату множення. Очевидно, що при цьому буде отриманий необхідний структурний код $K_S(a_2)=110$. Ототожнюючи операції переходів O_1-O_3 зі структурними функціями переходів d_1-d_3, закодуємо їх двійковими кодами $<z_1, z_2>$ в такий спосіб: $K(O_1)=00$, $K(O_2)=01$, $K(O_3)=10$. Складемо операційну таблицю переходів (табл. 3.3), відповідно до якої система рівнянь (3.19) буде мати наступний вигляд:

$$\begin{cases} z_1 = T_1T_2\overline{T}_3 \vee \overline{T}_1\overline{T}_2T_3; \\ z_2 = T_1T_2T_3 \vee T_1\overline{T}_2T_3. \end{cases} \qquad (3.30)$$

Таблиця 3.3

Операційна таблиця переходів МПА зі структурою U_2 (ГСА Γ_{3-1})

a_m	$K_S(a_m)$	a_s	$K_s(a_m)$	O_h	Z_h	h
a_0	010	a_1	100	O_1	–	1
a_1	100	a_2	110	O_1	–	2
a_2	110	a_3	001	O_3	z_1	3
		a_4	111			4
a_3	001	a_1	100	O_3	z_1	5
		a_5	101			6
a_4	111	a_2	110	O_2	z_2	7
a_5	101	a_0	010	O_2	z_2	8

Як було відзначено вище, відсутність вхідних змінних X у списку аргументів функції (3.24) спрощує синтез Z-підсхеми в базисі запам'ятовувальних пристроїв. Для функції переходів МПА з ОАП, що задається табл. 3.3, вміст ЗП Z-підсхеми відповідає табл. 3.4, де символи «*» відповідають довільним значенням сигналів z_1, z_2.

Таблиця 3.4

Вміст запам'ятовувального пристрою Z-підсхеми (ГСА Γ_{3-1})

Адреса	z_1 z_2
0 0 0	* *
0 0 1	1 0
0 1 0	0 0
0 1 1	* *
1 0 0	0 0
1 0 1	0 1
1 1 0	1 0
1 1 1	0 1

При проектуванні логічної схеми МПА зі структурою U_2 слід враховувати, що виключення сигналів логічних умов із системи (3.17) може приводити до таких негативних наслідків, як збільшення кількості та ускладнення операцій переходів, які нівелюють виграш в апаратурних витратах, що досягається в Z-підсхемі. Так, у розглянутому прикладі операція O_3, що використовується для виконання умовних переходів, не може бути розбита на декілька більш простих операцій з метою їх використання для реалізації інших переходів автомата, оскільки, згідно (3.24), стану автомата може бути зіставлена лише одна ОП. До того ж, якби переходи зі стану a_3 виконувалися за іншою ЛУ або за декількома ЛУ, то станам a_2 та a_3 були б зіставлені різні ОП, що мають окремі комбінаційні схеми у блоці ОЧ та різні коди в Z-підсхемі.

3.3.3 Заміна вхідних змінних

Характерною рисою МПА є те, що майже завжди для будь-якого стану $a_i \in A$ виконується умова

$$|X(a_i)| << L, \tag{3.31}$$

де $X(a_i)$ – підмножина множини X, елементи якої суттєво впливають на переходи зі стану a_i [31]. Виконання для заданої ГСА нерівності (3.31) дозволяє

застосувати до МПА з ОАП відомий метод заміни вхідних змінних, відповідних до сигналів логічних умов [31, 36, 180].

Нехай МПА заданий ГСА, у якій будь-який перехід залежить не більш ніж від G логічних умов. В цьому випадку на вхід Z-підсхеми та ОАП буде подаватися множина сигналів $P=\{p_1,...,p_G\}$, а в структуру автомата додається спеціальна М-підсхема, що реалізує функцію

$$P = P(X,T). \tag{3.32}$$

Додавання М-підсхеми в структуру U_1 приводить до структури МПА з ОАП і заміною вхідних змінних, що зображена на рис. 3.9. Позначимо дану структуру символом U_3.

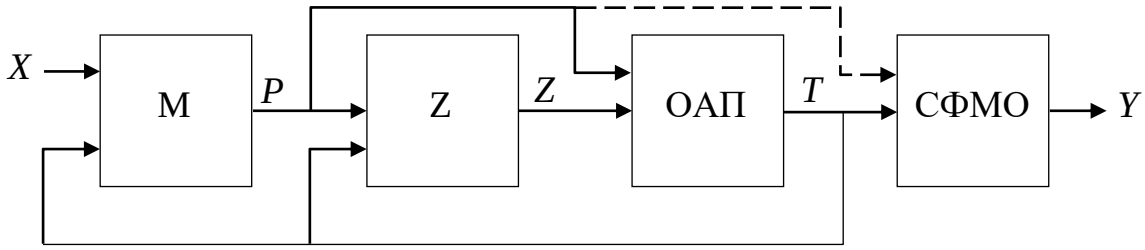


Рисунок 3.9 – Структурна схема МПА з ОАП
і заміною вхідних змінних (структура U_3)

Тут Z-підсхема реалізує функцію

$$Z = Z(T,P) \tag{3.33}$$

і будується за системою рівнянь

$$\begin{cases} z_1 = z_1(T_1,...,T_R,\, p_1,...,\, p_G); \\ z_2 = z_2(T_1,...,T_R,\, p_1,...,\, p_G); \\ ... \\ z_{R_Z} = z_{R_Z}(T_1,...,T_R,\, p_1,...,\, p_G). \end{cases} \tag{3.34}$$

М-підсхема реалізує функцію

$$P = P(X,T) \tag{3.35}$$

і будується за системою рівнянь

$$\begin{cases} p_1 = p_1(T_1, ..., T_R, x_1, ..., x_L); \\ p_2 = p_2(T_1, ..., T_R, x_1, ..., x_L); \\ ... \\ p_G = p_G(T_1, ..., T_R, x_1, ..., x_L). \end{cases} \quad (3.36)$$

Схема СФМО реалізує функцію

$$Y = Y(T, P) \quad (3.37)$$

у випадку автомата Мілі та функцію (1.14) у випадку автомата Мура. Таким чином, наявність у структурі U_3 зв'язку, показаного на рис. 3.9 пунктиром, дозволяє вважати дану структуру автоматом Мілі, відсутність зв'язку – автоматом Мура.

Складемо нерівність, при виконанні якої апаратурні витрати в логічній схемі МПА зі структурою U_3 будуть меншими за витрати в схемі зі структурою U_1:

$$H_M^{U_3} + H_Z^{U_3} + H_{ОЧ}^{U_3} + H_{РП}^{U_3} + H_{СФМО}^{U_3} < H_Z^{U_1} + H_{ОЧ}^{U_1} + H_{РП}^{U_1} + H_{СФМО}^{U_1}, \quad (3.38)$$

де $H_M^{U_3}$ – витрати апаратури в М-підсхемі структури U_3. Відзначимо, що при побудові схеми МПА в гетерогенному елементному базисі М-підсхема може бути реалізована на стандартних мультиплексорах або ПЛМ із великою кількістю входів та малою кількістю виходів [31, 36].

Метод кодування вхідних змінних припускає, що зростання апаратурних витрат за рахунок додавання М-підсхеми супроводжується одночасним їхнім зниженням в Z-підсхемі та СФМО. Так, кодування вхідних змінних дозволяє знизити число входів Z-підсхеми і схеми СФМО (у випадку автомата Мілі) до значення $(G + R)$, де R – розрядність коду стану автомата. При цьому крім зменшення витрат в Z-підсхемі та СФМО спрощується реалізація даних блоків у базисі запам'ятовувальних пристроїв. Таким чином, при $G << L$ в загальному випадку можна чекати виконання нерівностей $H_Z^{U_3} < H_Z^{U_1}$ та (у випадку автомата Мілі) $H_{СФМО}^{U_3} < H_{СФМО}^{U_1}$.

Що стосується операційного автомата переходів, то кодування вхідних змінних не виявляє принципового впливу на його внутрішню структуру.

Розглянемо приклад синтезу МПА зі структурою U_3. Нехай МПА заданий ГСА Γ_{3-2}, що позначена станами автомата Мура (рис. 3.10).

ГСА містить $M = 10$ станів $a_0 - a_9$, для кодування яких потрібно $R = 4$ двійкових розряди. Застосуємо до ГСА Γ_{3-2} метод заміни вхідних змінних. Згідно до рис. 3.10, значення $G = 2$, тобто $P = \{p_1, p_2\}$.

Сформуємо таблицю заміни вхідних змінних $x_1 - x_5$ на змінні p_1, p_2 (табл. 3.5), для чого застосуємо методику, викладену в [36].

Далі виконаємо алгебраїчний синтез автомата за ГСА, що зображена на рис. 3.10, у якій замість змінних x_1, x_3, x_4 використовується змінна p_1, замість змінних x_2, x_5 – змінна p_2.

Закодуємо стани $a_0 - a_9$ проміжними кодами K_I із множини цілих чисел у діапазоні [0 – 15]. Нехай $K_I(a_0) = 3$, $K_I(a_1) = 8$, $K_I(a_2) = 4$, $K_I(a_3) = 0$, $K_I(a_4) = 13$, $K_I(a_5) = 11$, $K_I(a_6) = 5$, $K_I(a_7) = 2$, $K_I(a_8) = 6$, $K_I(a_9) = 7$. Структурні коди станів, що представляються двійковими векторами $< T_1, T_2, T_3, T_4 >$, виберемо такими, що дорівнюють відповідним проміжним кодам, представленим у форматі чотирирозрядних двійкових беззнакових чисел: $K_S(a_0) = 0011$, $K_S(a_1) = 1000$, $K_S(a_2) = 0100$, $K_S(a_3) = 0000$, $K_S(a_4) = 1101$, $K_S(a_5) = 1011$, $K_S(a_6) = 0101$, $K_S(a_7) = 0010$, $K_S(a_8) = 0110$, $K_S(a_9) = 0111$.

Задамо наступні операції переходів, визначені на множині проміжних кодів станів:

$$O_1: K_I(a^{t+1}) = K_I(a^t) + 5, \tag{3.39}$$

$$O_2: K_I(a^{t+1}) = K_I(a^t) + 11, \tag{3.40}$$

$$O_3: K_I(a^{t+1}) = K_I(a^t) \times 4, \tag{3.41}$$

$$O_4: K_I(a^{t+1}) = K_I(a^t) / 2. \tag{3.42}$$

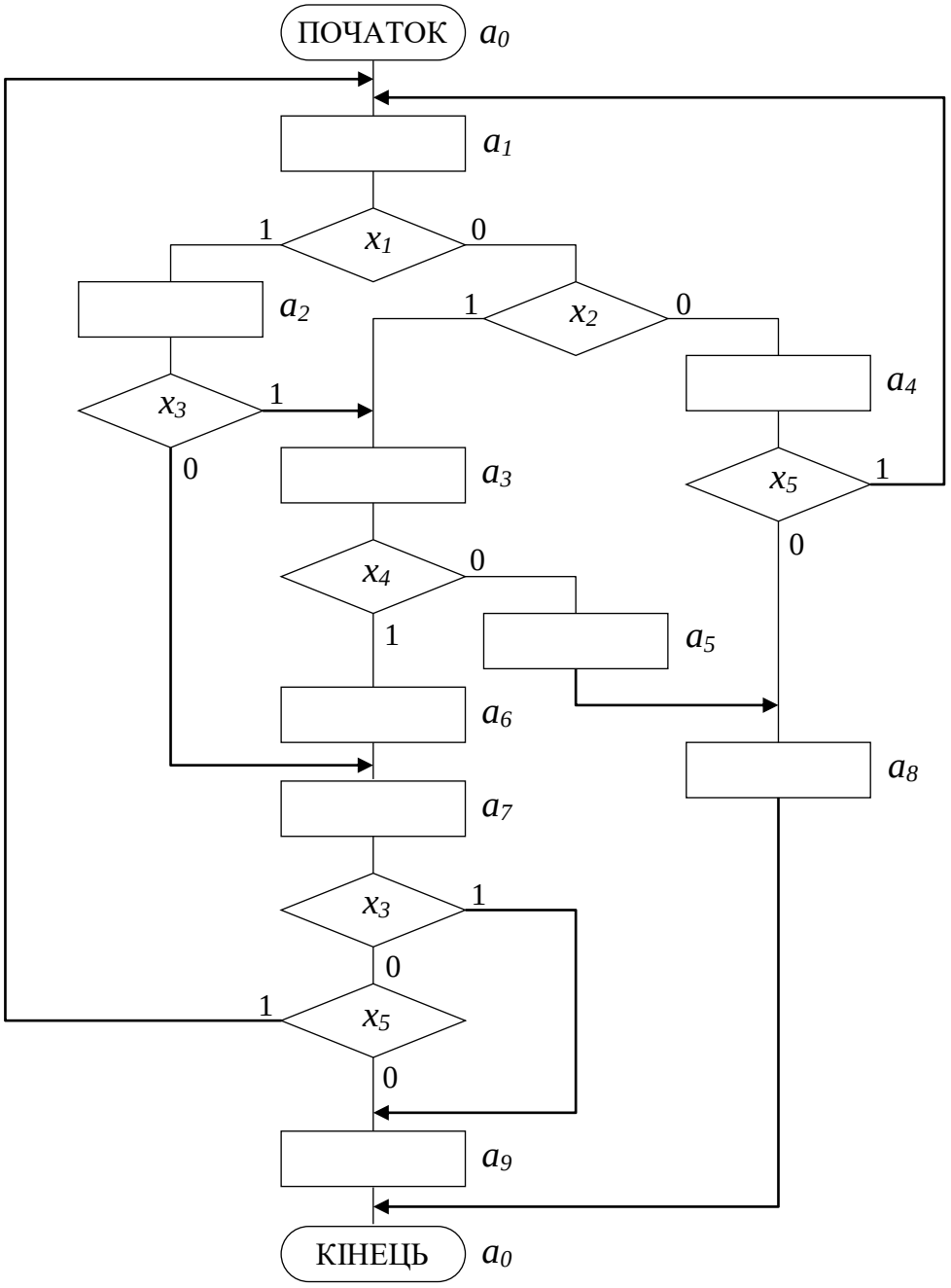


Рисунок 3.10 – Граф-схема алгоритму Γ_{3-2}

Таблиця 3.5

Таблиця заміни вхідних змінних (ГСА Γ_{3-2})

a_i	p_1	p_2	a_i	p_1	p_2
a_0	–	–	a_5	–	–
a_1	x_1	x_2	a_6	–	–
a_2	x_3	–	a_7	x_3	x_3
a_3	x_4	–	a_8	–	–
a_4	–	x_5	a_9	–	–

При використанні даних ОП будемо вважати, що результат кожної ОП завжди приводиться до діапазону [0, 15], що виражається операцією «mod 16». При схемній реалізації операцій переходів дія «mod 16» реалізується шляхом побудови функціональних блоків для ОП $O_1 - O_4$ із розрахунку обробки чотирирозрядних двійкових кодів.

Закодуємо операції $O_1 - O_4$ унікальними дворозрядними кодами $< z_1, z_2 >$: $K_Z(O_1) = 00$, $K_Z(O_2) = 01$, $K_Z(O_2) = 10$, $K_Z(O_3) = 11$. Зіставимо кожному переходу автомата одну з ОП O_1 - O_4 так, як показано в операційній таблиці переходів (табл. 3.6).

Відзначимо, що ОТП для МПА зі структурою U_3 відрізняється від ОТП для МПА зі структурою U_2 тим, що замість стовпця X_h використовується стовпець P_h, який містить змінні з множини *P*, що ставляться у відповідність змінним із множини *X*. У графічній формі результати алгебраїчного синтезу показані на рис. 3.11.

За табл. 3.6 будується система рівнянь (3.34), кожне рівняння якої, за аналогією з (3.19), визначається наступним виразом:

$$z_i = \bigvee_{h=1}^{H} C_{ih} A_m^h P_h . \qquad (3.43)$$

Тут C_{ih} – булева змінна, що набуває одиничне значення, якщо і тільки якщо в рядку *h* операційної таблиці переходів у стовпці Z_h записана змінна z_i; P_h – кон'юнктивний терм, утворений змінними з множини *P*, вказаними в стовпці P_h рядка *h* ОТП.

Для синтезу М-підсхеми треба побудувати систему рівнянь (3.36) за табл. 3.5. Методика синтезу М-підсхеми, що враховує особливості обраного елементного базису, докладно викладена в [36] і в даній роботі не розглядається. Там же дана методика синтезу схеми СФМО у випадку автомата Мілі із заміною вхідних змінних.

Таблиця 3.6

Операційна таблиця переходів МПА зі структурою U_3 (ГСА Γ_{3-2})

a_m	$K_S(a_m)$	a_s	$K_S(a_s)$	P_h	O_h	Z_h	h
a_0	0011	a_1	1000	1	O_1	–	1
a_1	1000	a_2	0100	p_1	O_4	$z_1\ z_2$	2
		a_3	0000	$\overline{p}_1 p_2$	O_3	z_1	3
		a_4	1101	$\overline{p}_1 \overline{p}_2$	O_1	–	4
a_2	0100	a_3	0000	p_1	O_3	z_1	5
		a_7	0010	$\overline{p}_1$	O_4	$z_1\ z_2$	6
a_3	0000	a_5	1011	p_1	O_2	z_2	7
		a_6	0101	$\overline{p}_1$	O_1	–	8
a_4	1101	a_1	1000	p_2	O_2	z_2	9
		a_8	0110	$\overline{p}_2$	O_4	$z_1\ z_2$	10
a_5	1011	a_8	0110	1	O_2	z_2	11
a_6	0101	a_7	0010	1	O_4	$z_1\ z_2$	12
a_7	0010	a_9	0111	p_1	O_1	–	13
		a_1	1000	$\overline{p}_1 p_2$	O_3	z_1	14
		a_9	0111	$\overline{p}_1 \overline{p}_2$	O_1	–	15
a_8	0110	a_0	0011	1	O_4	$z_1\ z_2$	16
a_9	0111	a_0	0011	1	O_4	$z_1\ z_2$	17

Метод заміни вхідних змінних може також бути застосований до структури U_2, що зображена на рис. 3.6 [16, 38]. Оскільки в U_2 сигнали логічних умов надходять тільки до схеми ОАП, додавання М-підсхеми дозволяє одержати структуру МПА, подібну до структури U_3, але з відсутнім зв'язком між М- та Z-підсхемами (рис. 3.12). Позначимо дану структуру символом U_4.

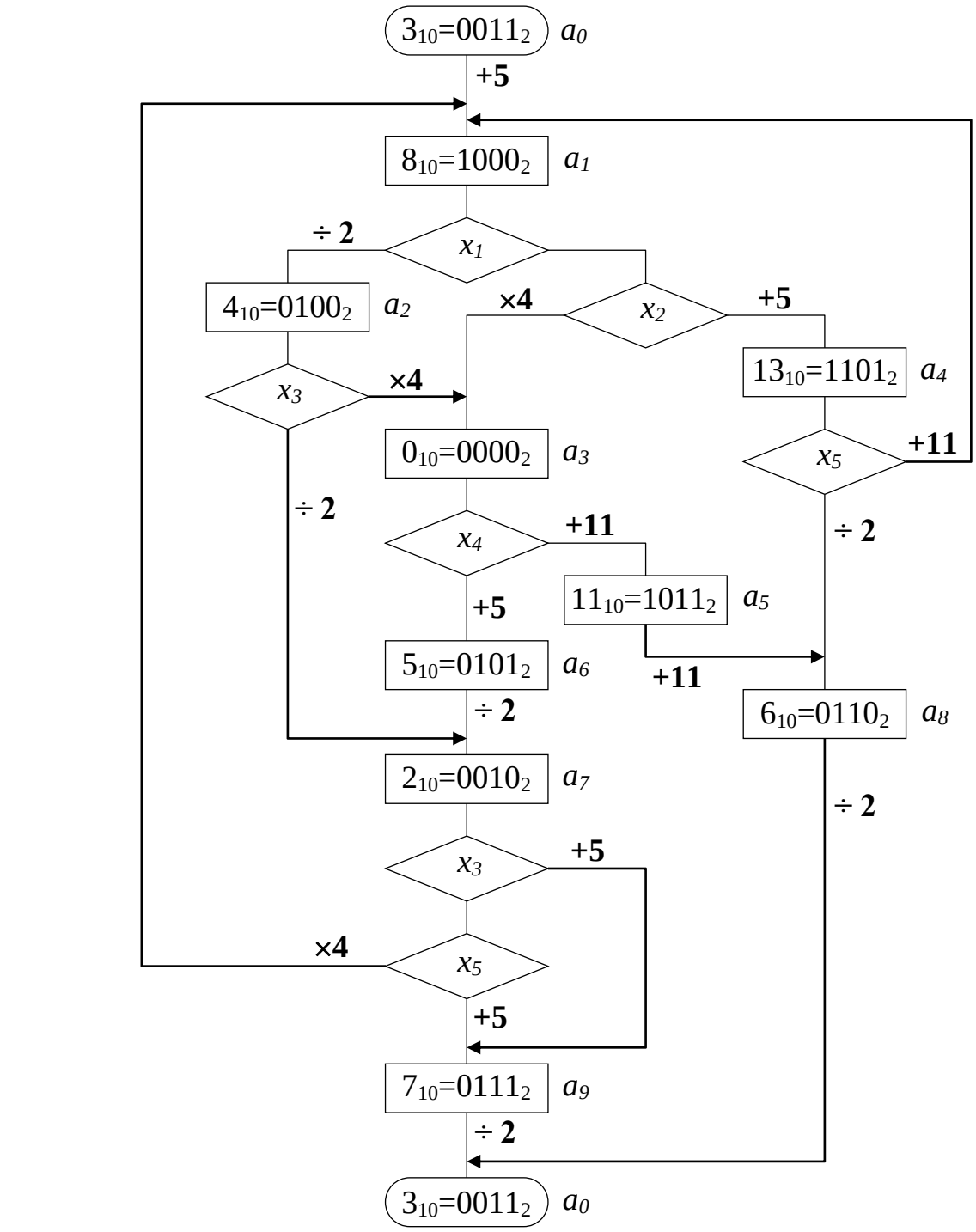


Рисунок 3.11 – Графічне представлення результатів алгебраїчного синтезу МПА з ОАП (ГСА Γ_{3-2})

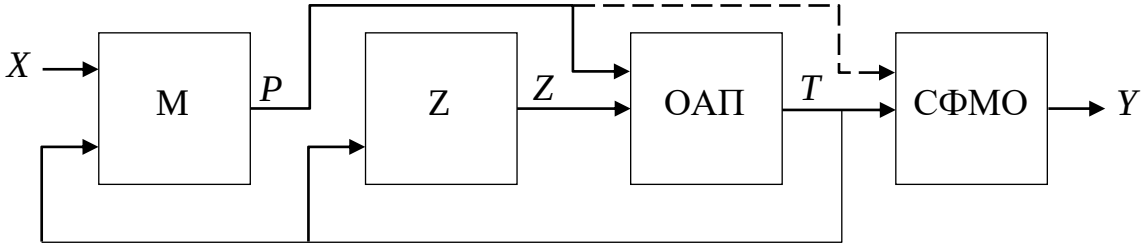


Рисунок 3.12 – Структурна схема МПА з ОАП U_4

У даній структурі Z-підсхема реалізує функцію (3.24) і синтезується за системою рівнянь (3.23). M-підсхема реалізує функцію (3.35) і синтезується за системою рівнянь (3.36). Схема формування мікрооперацій синтезується за системою рівнянь (3.37) у випадку автомата Мілі та за системою (1.14) у випадку автомата Мура.

Порівняння структур U_2 та U_4 показує, що в них Z-підсхеми мають однакову кількість вхідних і вихідних сигналів. При цьому в U_4 присутня M-підсхема, що збільшує апаратурні витрати в логічній схемі МПА зі структурою U_4 в порівнянні зі структурою U_2. Також в автоматі Мілі зі структурою U_4 можливе зниження апаратурних витрат у схемі СФМО в порівнянні з автоматом Мілі зі структурою U_2 за рахунок зменшення числа входів СФМО.

У той же час застосування методу заміни вхідних змінних до структури U_2 впливає на схему операційного автомата переходів, що полягає в наступному. Нехай деяка ГСА Γ_{3-3}, позначена станами автомата Мура, містить фрагмент, зображений на рис. 3.13.

Припустимо, що загальна кількість станів ГСА Γ_{3-3} така, що для їхнього кодування достатньо $R=4$ двійкових розряди. Задамо проміжні й структурні коди станів так, як показано в табл. 3.7. Помітимо, що якби у всіх умовних вершинах даного фрагмента була записана одна й та саме логічна умова (наприклад, x_1), то всі переходи в рамках даного фрагмента могли бути реалізовані шляхом зіставлення станам $a_1 - a_3$ операції переходів (3.44).

$$O_1: \quad K_I(a^{t+1}) = \begin{cases} K_I(a^t) + 5, \text{ якщо } x_1 = 0; \\ K_I(a^t)/2 + 5, \text{ якщо } x_1 = 1, \end{cases} \tag{3.44}$$

Схемна реалізація даної операції відповідає рис. 2.12. Результат алгебраїчного синтезу для ГСА Γ_{3-3} із використанням ОП (3.44) за умови знаходження в умовних вершинах однакової логічної умови x_1 графічно показаний на рис. 3.14.

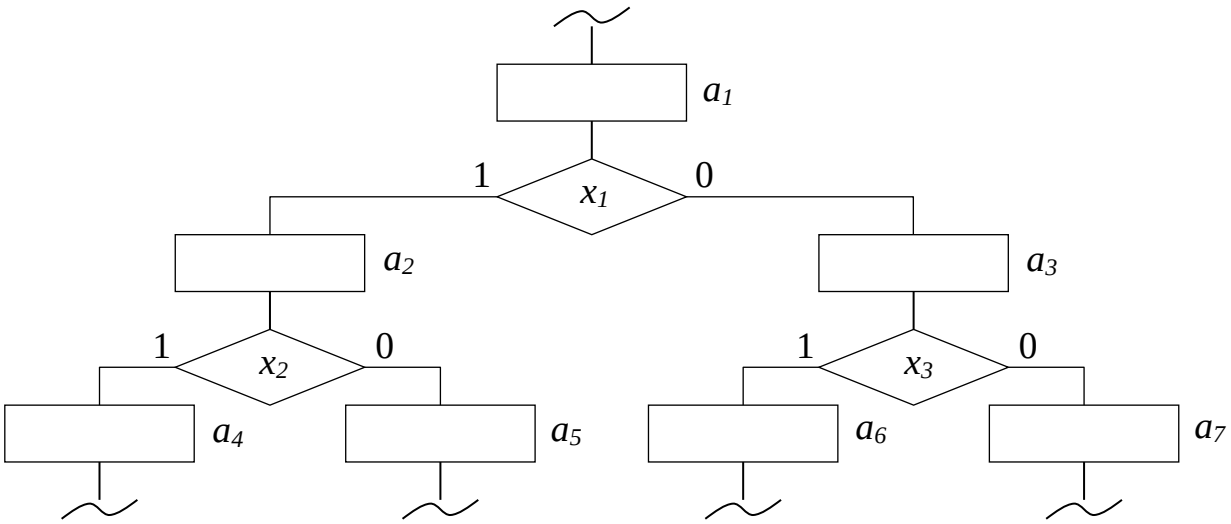


Рисунок 3.13 – Фрагмент ГСА Γ_{3-3}

Таблиця 3.7

Кодування станів (фрагмент ГСА Γ_{3-3})

a_i	a_1	a_2	a_3	a_4	a_5	a_6	a_7
$K_I(a_i)$	1	6	5	11	8	10	7
$K_S(a_i)$	0001	0110	0101	1011	1000	1010	0111

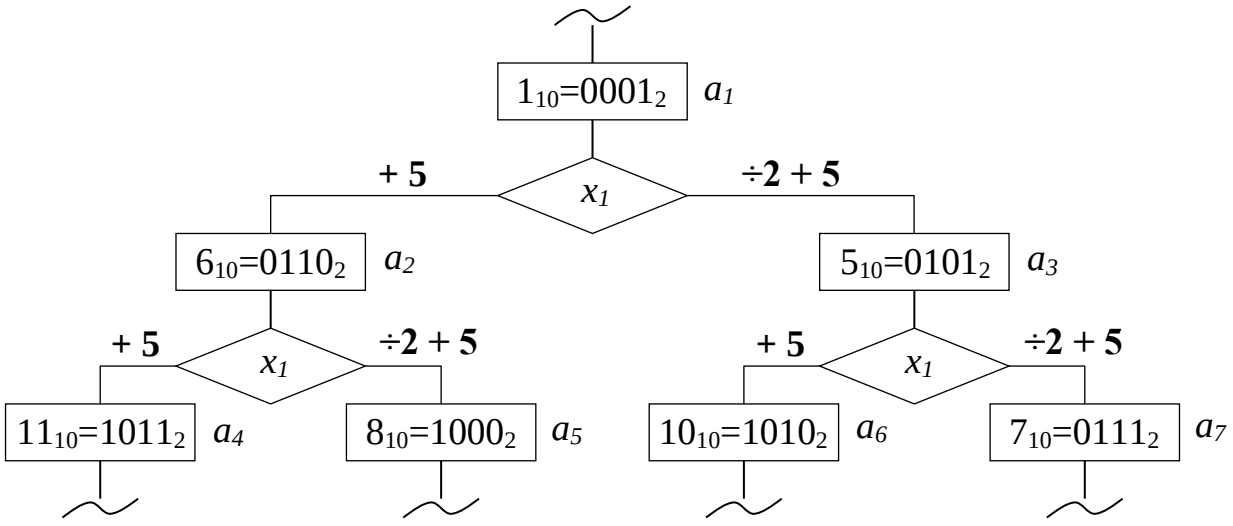


Рисунок 3.14 – Графічне представлення реалізації переходів у фрагменті ГСА Γ_{3-3} за допомогою ОП (3.44) при використанні в усіх умовних вершинах ЛУ x_1

Той факт, що на рис. 3.13 у кожній умовній вершині записані різні ЛУ, у випадку МПА зі структурою U_2 не дозволяє зіставити операцію (3.44) станам a_2 та a_3. Дана ОП може бути зіставлена тільки стану a_1, тоді як для стану a_2 повинна бути задана окрема ОП O_2, що використовує у своїй формулі ЛУ x_2, для a_3 – ОП O_3, що використовує ЛУ x_3. Дані ОП можуть бути відповідно задані виразами (3.45) та (3.46), подібними (3.44):

$$O_2: \quad K_I(a^{t+1}) = \begin{cases} K_I(a^t) + 5, \text{ якщо } x_2 = 0; \\ K_I(a^t)/2 + 5, \text{ якщо } x_2 = 1, \end{cases} \tag{3.45}$$

$$O_3: \quad K_I(a^{t+1}) = \begin{cases} K_I(a^t) + 5, \text{ якщо } x_3 = 0; \\ K_I(a^t)/2 + 5, \text{ якщо } x_3 = 1. \end{cases} \tag{3.46}$$

Незважаючи на загальну схожість виразів (3.45) та (3.46) з виразом (3.44), у схемі ОАП структури U_2 даним ОП будуть відповідати окремі комбінаційні схеми, які подібні до схеми ОП O_1, але використовують різні сигнали ЛУ для керування мультиплексором МХ (рис. 2.12). При цьому кожна ОП буде мати унікальний код, що враховується при побудові мультиплексора результату в операційній частині ОАП. Фрагмент функціональної схеми ОЧ, що відповідає операціям переходів $O_1 - O_3$, наведений на рис. 3.15.

У випадку МПА зі структурою U_4 змінні $x_1 - x_3$ в рамках фрагменту ГСА на рис. 3.13 заміняються на змінну p_1. Це дозволяє одержати результати алгебраїчного синтезу, наведені на рис. 3.14, при використанні єдиної ОП виду

$$O_4: \quad K_I(a^{t+1}) = \begin{cases} K_I(a^t) + 5, \text{ якщо } p_1 = 0; \\ K_I(a^t)/2 + 5, \text{ якщо } p_1 = 1. \end{cases} \tag{3.47}$$

Схемна реалізація даної ОП відповідає рис. 2.12, але замість сигналу x_1 на керуючий вхід мультиплексора МХ подається сигнал p_1. Оскільки дана відмінність не впливає на апаратурні витрати в комбінаційній схемі операції O_4, порівняння рис. 2.12 та 3.15 дозволяє стверджувати, що для розглянутого фрагмента ГСА Γ_{3-3} витрати апаратури в блоці ОАП МПА зі структурою U_4

будуть нижче, ніж у випадку МПА зі структурою U_2 за рахунок меншої кількості комбінаційних схем в блоці ОЧ.

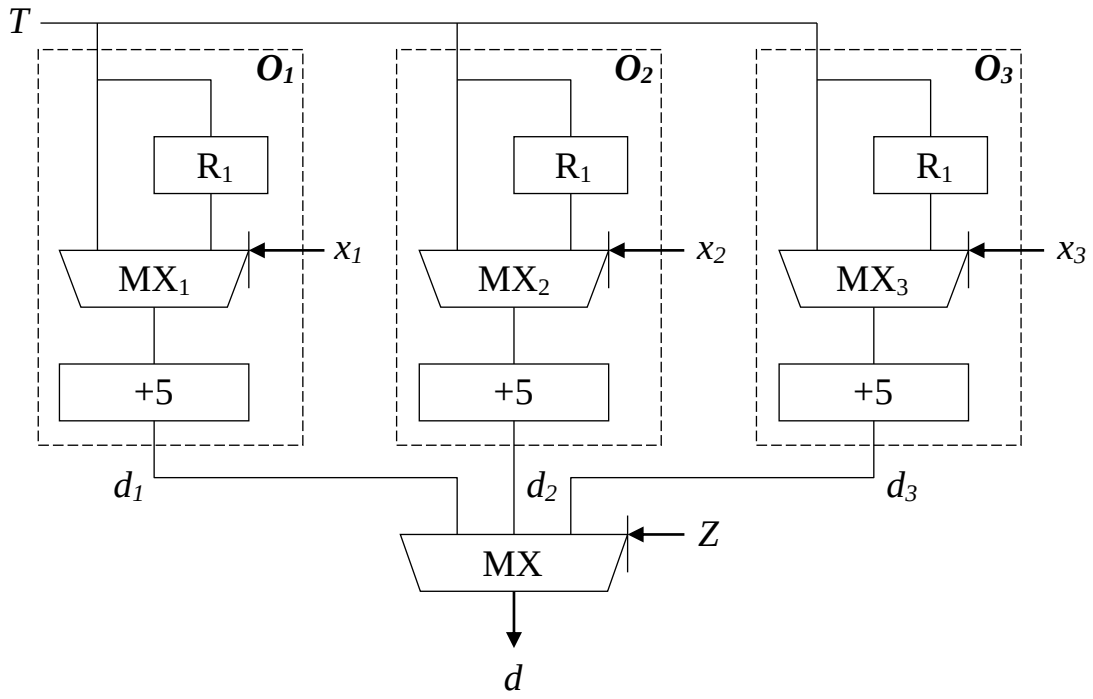


Рисунок 3.15 – Фрагмент функціональної схеми МПА з ОАП зі структурою U_2, що відповідає операціям переходів (3.44) – (3.46)

Також в Z-підсхемі та ОАП матимуть місце наступні зміни: зменшується число виходів Z-підсхеми, що приводить до зниження апаратурних витрат в Z-підсхемі; зменшується число входів мультиплексора результату ОАП, що приводить до зниження апаратурних витрат у схемі ОАП.

Таким чином, у структурі U_4 в порівнянні зі структурою U_2 можливі як приріст апаратурних витрат за рахунок додавання М-підсхеми, так і їх зменшення в Z-підсхемі і блоці ОЧ операційного автомата переходів. Складемо, за аналогією з виразом (2.37), нерівність, при виконанні якої логічна схема МПА U_4 матиме менші апаратурні витрати в порівнянні зі схемою МПА U_2:

$$H_M^{U_4} + H_Z^{U_4} + H_{ОЧ}^{U_4} + H_{РП}^{U_4} + H_{СФМО}^{U_4} < H_Z^{U_2} + H_{ОЧ}^{U_2} + H_{РП}^{U_2} + H_{СФМО}^{U_2} \ . \qquad (3.48)$$

У випадку $H_{РП}^{U_2} = H_{РП}^{U_4}$ нерівність (3.48) набуває наступного вигляду:

$$H_M^{U_4} + H_Z^{U_4} + H_{ОЧ}^{U_4} + H_{СФМО}^{U_4} < H_Z^{U_2} + H_{ОЧ}^{U_2} + H_{СФМО}^{U_2} \ . \qquad (3.49)$$

3.3.4 Зменшення максимальної кількості істотних вхідних змінних

Існує ряд методів оптимізації логічної схеми МПА, в основі яких лежать припустимі перетворення вихідної ГСА [29]. Одним з таких перетворень є введення у вихідну ГСА додаткових станів [35, 36, 151]. Ця дія супроводжується збільшенням часу виконання алгоритму, а також можливим зростанням апаратурних витрат, викликаним збільшенням розрядності структурного коду стану. Якщо наслідки введення додаткових станів не суперечать критеріям проектування, даний підхід може бути використаний для зниження витрат апаратури в схемі мікропрограмного автомата з операційним автоматом переходів [16, 47].

Як було відзначено в п. 3.3.3, застосування до МПА з ОАП методу заміни вхідних змінних, що ототожнюються з логічними умовами ГСА, дозволяє в деяких випадках знизити витрати апаратури в логічній схемі автомата за рахунок зменшення числа входів Z-підсхеми та операційного автомата переходів. Так, при використанні структури U_3 замість U_1 і структури U_4 замість U_2 число входів ОАП знижується до значення $(R+G)$, де R – розрядність структурного коду стану, G – максимальна кількість вхідних змінних, що суттєво впливають на автоматні переходи. До того ж значення знижується число входів Z-підсхеми в структурі U_3 в порівнянні зі структурою U_1.

Подальше зниження апаратурних витрат можливе за рахунок зменшення величини G. Для цього може бути використаний відомий підхід, що полягає в наступному [36, 151]:

1. Порожня операторна вершина може бути додана до будь-якої гілки, що з'єднує вихід однієї умовної вершини із входом іншої умовної вершини.

2. Кількість вершин, що додаються, має бути мінімально достатньою для зниження величини G до необхідного значення.

У граничному випадку G може дорівнювати одиниці, тобто будь-який автоматний перехід залежить не більш ніж від однієї логічної умови.

Розглянемо застосування даного підходу до структури U_4. Нехай МПА заданий ГСА Γ_{3-4}, що позначена станами автомата Мілі. Її фрагмент наведений на рис. 3.16 і визначає переходи зі станів $a_1 - a_4$. Виконаємо заміну вхідних змінних відповідно до табл. 3.8 за методикою, що викладена в [36]. Після заміни логічних умов фрагмент ГСА Γ_{3-4} матиме вигляд відповідно до рис. 3.17.

Таблиця 3.8

Таблиця заміни вхідних змінних (фрагмент ГСА Γ_{3-4})

a_i	p_1	p_2	p_3
a_1	x_1	x_2	x_3
a_2	x_4		x_3
a_3	–	–	–
a_4	x_1		

Використання структури МПА U_4 припускає, що операції переходів будуть зіставлені не окремим переходам, а станам автомата. На рис. 3.17 переходи зі станів $a_1 - a_4$ залежать від різної кількості вхідних змінних. Це не дозволяє в рамках даного фрагмента зіставити ту саму ОП більш ніж одному стану: стану a_1 повинна бути зіставлена ОП, що використовує в якості аргументів три змінні $p_1 - p_3$, стану a_2 – ОП, що використовує дві змінні p_1 та p_3, стану a_3 – ОП, що не використовує вхідні змінні в якості аргументів, стану a_4 – ОП, що використовує одну змінну p_1. Таким чином, усі чотири ОП будуть різними незалежно від того, яким чином буде здійснена заміна змінних *X* змінними *P*.

Перетворимо фрагмент ГСА, зображений на рис. 3.16, таким чином, щоб будь-який перехід у рамках фрагмента залежав не більш ніж від однієї логічної умови. Для цього додамо порожні операторні вершини в кожну гілку, яка з'єднує умовні вершини, що приведе до появи трьох додаткових станів $a_{10} - a_{12}$.

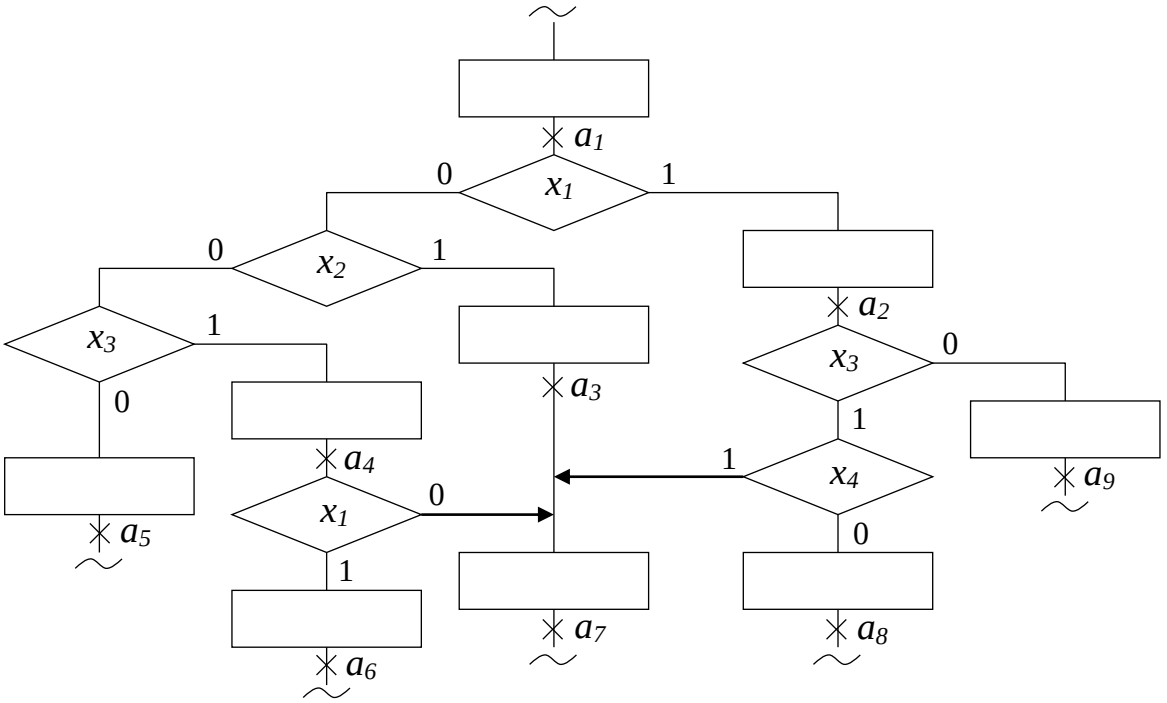


Рисунок 3.16 – Фрагмент ГСА Γ_{3-4}

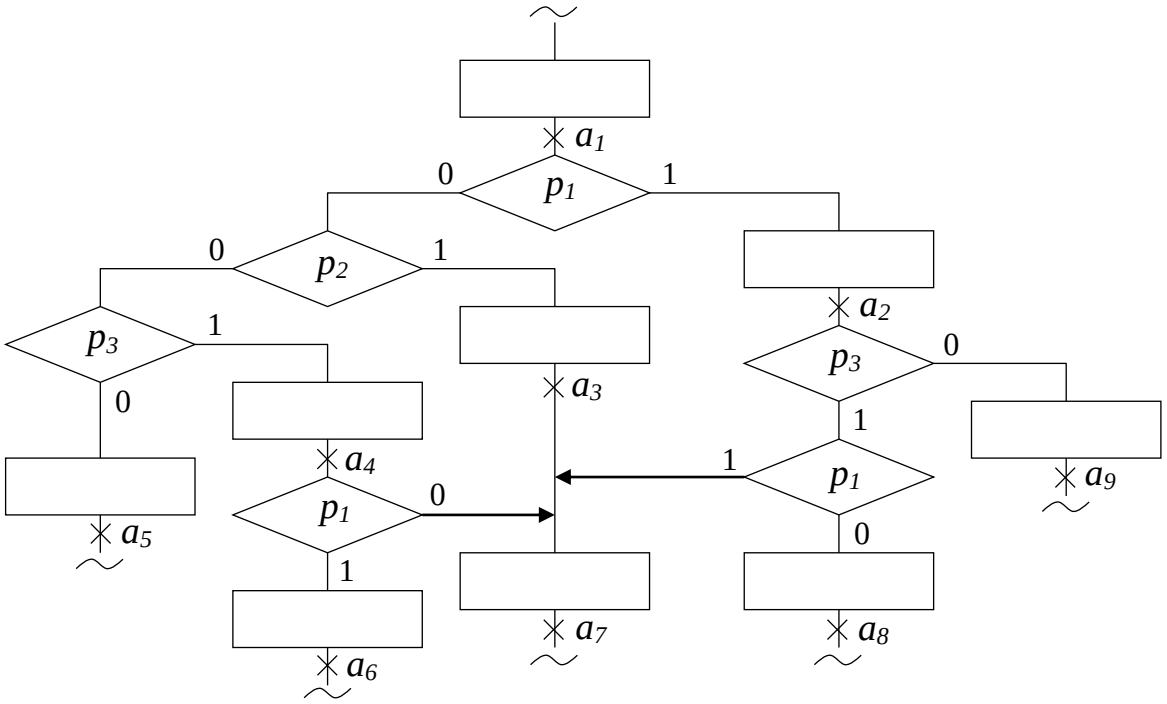


Рисунок 3.17 – Фрагмент ГСА Γ_{3-4} після заміни вхідних змінних

Заміна вхідних змінних у фрагменті Γ_{3-4} приводить до того, що в кожній умовній вершині буде записана змінна p_1 (рис. 3.18). У випадку структури U_4 зроблені перетворення дають формальну можливість зіставити станам, виходи яких з'єднані зі входом умовної вершини (станам a_1, a_2, a_4, a_{10}, a_{11}, a_{12}) однакову операцію переходів, що використовує в якості аргументів код поточного стану і значення змінної p_1. При цьому стану a_3, як і раніше, має бути зіставлена окрема ОП, що реалізує безумовний перехід, не залежний від p_1. Таким чином, мінімально припустима кількість ОП для даного фрагмента дорівнює двом.

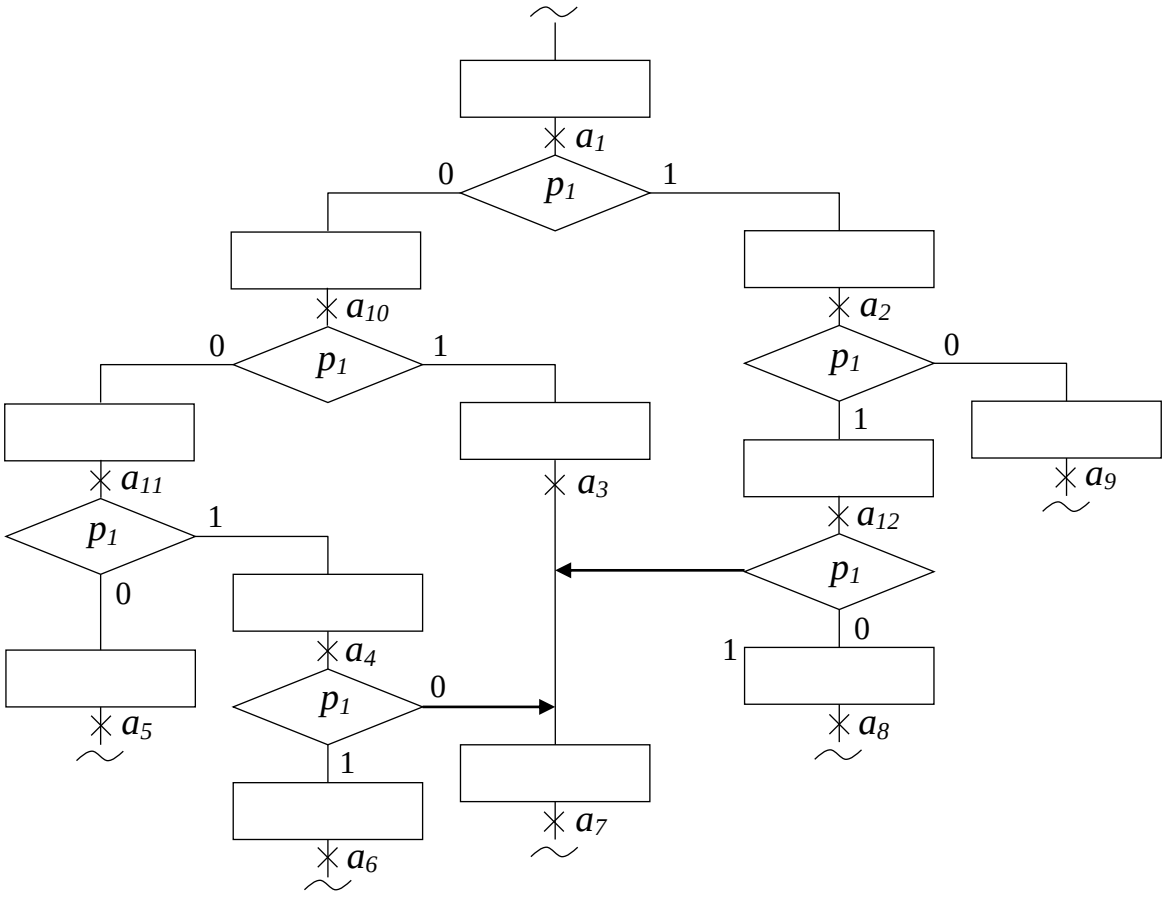


Рисунок 3.18 – Перетворений фрагмент ГСА Γ_{3-4}

Розглянемо приклад, що демонструє можливість використання двох ОП для реалізації всіх переходів у рамках фрагмента, зображеного на рис. 3.18. Даний фрагмент містить три типи переходів: безумовний перехід, перехід за умовою $p_1 = 0$ та перехід за умовою $p_1 = 1$. В абстрактному автоматі ці переходи

виконуються під впливом абстрактних вхідних сигналів $z_0 - z_2$, структурні коди яких визначаються значеннями змінної p_1, де $K_S(z_0) = <->$, $K_S(z_1) = <0>$, $K_S(z_2) = <1>$.

Проведемо алгебраїчний синтез для фрагмента ГСА на рис. 3.18. Задамо дві абстрактні підалгебри переходів, що визначаються виразами (3.50) та (3.51), перша з яких визначає всі умовні переходи, друга – єдиний безумовний перехід зі стану a_3 до стану a_7.

$$\begin{cases} G_{\delta_1} = <\{A_{\delta_1}, Z_{\delta_1}\}, \{\delta_1\}>; \\ A_{\delta_1} = \{a_1, ..., a_{12}\}; \\ Z_{\delta_1} = \{z_1, z_2\}; \\ \delta_1 = \{< a_1, z_1, a_{10} >, < a_1, z_2, a_2 >, < a_2, z_1, a_9 >, < a_2, z_2, a_{12} >, \\ \quad < a_{10}, z_1, a_{11} >, < a_{10}, z_2, a_3 >, < a_{11}, z_1, a_5 >, < a_{11}, z_2, a_4 >, \\ \quad < a_{12}, z_1, a_8 >, < a_{12}, z_2, a_7 >, < a_4, z_1, a_7 >, < a_4, z_2, a_6 >\}. \end{cases} \tag{3.50}$$

$$\begin{cases} G_{\delta_2} = <\{A_{\delta_2}, Z_{\delta_2}\}, \{\delta_2\}>; \\ A_{\delta_2} = \{a_3, a_7\}; \\ Z_{\delta_2} = \{z_0\}; \\ \delta_2 = \{< a_3, z_0, a_7 >\}. \end{cases} \tag{3.51}$$

Сформуємо проміжну алгебру переходів G_{I_1}, ізоморфну підалгебрі G_{δ_1}. Проміжні коди станів виберемо з безлічі натуральних чисел, проміжні коди вхідних сигналів будемо інтерпретувати як логічні величини, обумовлені на безлічі {ІСТИНА, ХИБНІСТЬ}. Нехай $K_{I_1}(z_1) = $ ХИБНІСТЬ, $K_{I_1}(z_2) = $ ІСТИНА.

Припустимо, що загальна кількість станів у ГСА Γ_{3-4} така, що мінімально достатня розрядність структурного коду стану $R = 5$. Задамо проміжні коди станів у діапазоні [0; 31] так, як показано в табл. 3.9.

Зіставимо станам $a_1, a_2, a_4, a_{10}, a_{11}, a_{12}$ наступну ОП:

$$O_1: \quad K_{I_1}(a^{t+1}) = \begin{cases} (K_{I_1}(a^t) \oplus 10100_2 + 2_{10}) \bmod 32, \text{ якщо } p_1 = 0; \\ (K_{I_1}(a^t) \oplus 10100_2 + 5_{10}) \bmod 32, \text{ якщо } p_1 = 1. \end{cases} \tag{3.52}$$

У цій інтервальній функції проміжний код поточного стану $K_{I_1}(a^t)$ спочатку приводиться до форми п’ятирозрядного двійкового вектора, над яким виконується порозрядна операція суми за модулем 2 із вектором-константою 10100_2. Отриманий результат інтерпретується як двійкове ціле без знака і підсумується із цілочисельною константою, значення якої залежить від значення змінної p_1. Над результатом додавання виконується операція «mod 32», що приводить результат ОП до діапазону [0; 31].

Таблиця 3.9

Кодування станів (фрагмент ГСА Γ_{3-4})

a_i	$K_I(a_i)$	a_i	$K_I(a_i)$
a_1	21	a_7	8
a_2	6	a_8	5
a_3	28	a_9	20
a_4	18	a_{10}	3
a_5	15	a_{11}	25
a_6	11	a_{12}	23

Операція O_1 дозволяє реалізувати всі переходи зі станів $a_1, a_2, a_4, a_{10}, a_{11}, a_{12}$ за умови використання проміжних кодів станів з таблиці 3.9. Наприклад, перехід зі стану a_{10} за умовою $p_1 = 0$ виконується в такий спосіб:

1) $K_{I_1}(a_{10}) = 3_{10} = 00011_2$;

2) $00011_2 \oplus 10100_2 = 10111_2$;

3) $10111_2 = 23_{10}$;

4) $23 + 2 = 25$;

5) $25 \bmod 32 = 25 = K_{I_1}(a_{11})$.

З урахуванням табл. 3.9 і виразу (3.52) проміжна алгебра G_{I_1} представляється системою (3.53).

Сформуємо структурну підалгебру переходів G_{d_1}, ізоморфну алгебрам (3.50) та (3.53). Для цього задамо станам із множини A_{δ_1} структурні коди, еквівалентні п'ятирозрядній двійковій формі відповідних проміжних кодів із множини A_{I_1}. Одержувана в результаті структурна підалгебра має вигляд (3.54).

$$\begin{cases} G_{I_1} = < A_{I_1}, F_{I_1} > = < \{K_{I_1}(A_{\delta_1}), K_{I_1}(Z_{\delta_1})\}, \{O_1\} >; \\ A_{I_1} = \{3, 5, 6, 8, 11, 15, 18, 20, 21, 23, 25, 28\}; \\ Z_{I_1} = \{\text{ІСТИНА, ХИБНІСТЬ}\}; \\ O_1 = \{< 21, \text{ ХИБНІСТЬ, } 35 >, < 21, \text{ ІСТИНА, } 6 >, \\ \quad < 6, \text{ ХИБНІСТЬ, } 20 >, < 6, \text{ ІСТИНА, } 23 >, \\ \quad < 3, \text{ ХИБНІСТЬ, } 25 >, < 3, \text{ ІСТИНА, } 28 >, \\ \quad < 25, \text{ ХИБНІСТЬ, } 15 >, < 25, \text{ ІСТИНА, } 18 >, \\ \quad < 23, \text{ ХИБНІСТЬ, } 5 >, < 23, \text{ ІСТИНА, } 8 >, \\ \quad < 18, \text{ ХИБНІСТЬ, } 8 >, < 18, \text{ ІСТИНА, } 11 >\}. \end{cases} \quad (3.53)$$

$$\begin{cases} G_{d_1} = < \{K_S(A_{d_1}), K_S(Z_{d_1})\}, \{d_1\} >; \\ K_S(A_{d_1}) = \{K_S(a_1), ..., K_S(a_{12})\} = \\ \quad = \{00011, 00101, 00110, 01000, 01011, 01111, \\ \quad 10010, 10100, 10101, 10111, 11001, 11100\}; \\ K_S(Z_{d_1}) = \{K_S(z_1), K_S(z_2)\} = \{0, 1\}; \\ d_1 = \{< 10101, 0, 00011 >, < 10101, 1, 00110 >, \\ \quad < 00110, 0, 10100 >, < 00110, 1, 10111 >, \\ \quad < 00011, 0, 11001 >, < 00011, 1, 11100 >, \\ \quad < 11001, 0, 01111 >, < 11001, 1, 10010 >, \\ \quad < 10111, 0, 00101 >, < 10111, 1, 01000 >, \\ \quad < 10010, 0, 01000 >, < 10010, 1, 01011 >\}. \end{cases} \quad (3.54)$$

Сформуємо проміжну алгебру переходів G_{I_2}, ізоморфну підалгебрі G_{δ_2}. Оскільки $A_{\delta_2} = \{a_3, a_7\} \subseteq A_{\delta_1}$, структурні коди $K_S(a_3)$ та $K_S(a_7)$ у підалгебрі G_{d_2} мають збігатися з відповідними структурними кодами в підалгебрі G_{d_1}. Знаючи

структурні коди станів і вхідних сигналів, можна сформувати структурну підалгебру G_{d_2}, ізоморфну підалгебрі (3.51):

$$\begin{cases} G_{d_2} = <\{K_S(A_{d_2}), K_S(Z_{d_2})\}, \{d_2\}>; \\ K_S(A_{d_2}) = \{K_S(a_3), K_S(a_7)\} = \{11100, 01000\}; \\ K_S(Z_{d_2}) = \{K_S(z_0)\} = \{-\}; \\ d_2 = \{<11100, -, 01000>\}. \end{cases} \tag{3.55}$$

Помітимо, що перетворення $K_S(a_3)$=11100 у $K_S(a_7)$=01000, обумовлене функцією d_2, може бути реалізоване шляхом виконання порозрядної операції додавання за модулем 2 коду $K_S(a_3)$ із двійковим вектором 10100. Це дозволяє задати операцію O_2 проміжної алгебри G_{I_2} в наступному вигляді:

$$O_2: \quad K_{I_2}(a^{t+1}) = K_{I_2}(a^t) \oplus 10100_2. \tag{3.56}$$

Виходячи з (3.56), проміжні коди станів, що утворюють носій $K_{I_2}(A_{\delta_2})$ алгебри G_{I_2}, є двійкові вектори, що збігаються зі структурними кодами відповідних станів підалгебри (3.55): $K_{I_2}(a_3) = 11100$, $K_{I_2}(a_7) = 01000$. Оскільки сигнал z_0 не бере участь в операції O_2, для проміжної алгебри G_{I_2} він є формальним, і $K_{I_2}(Z_{\delta_2}) = Z_{\delta_2} = \{z_0\}$. Тепер проміжна алгебра переходів G_{I_2} може бути задана наступною системою:

$$\begin{cases} G_{I_2} = <A_{I_2}, F_{I_2}> = <\{K_{I_2}(A_{\delta_2}), K_{I_2}(Z_{\delta_2})\}, \{O_2\}>; \\ K_{I_2}(A_{\delta_2}) = \{11100, 01000\}; \\ K_{I_2}(Z_{\delta_2}) = Z_{\delta_2} = \{z_0\}; \\ O_2 = \{<11100, z_0, 01000>\}. \end{cases} \tag{3.57}$$

Підкреслимо, що проміжна алгебра (3.57) ізоморфна підалгебрам (3.51) і (3.55). Існування ізоморфізмів $G_{\delta_1} \leftrightarrow G_{I_1} \leftrightarrow G_{d_1}$ та $G_{\delta_2} \leftrightarrow G_{I_2} \leftrightarrow G_{d_2}$ підтверджує можливість реалізації всіх переходів у фрагменті ГСА на рис. 3.18 за допомогою двох операцій переходів. Графічно дана можливість показана на рис. 3.19, де в кожній операторній вершині вказані проміжні й структурні коди стану,

що закріплений за даною вершиною, а кожна дуга графа відзначена операцією, що реалізує даний перехід.

Таким чином, зменшення максимальної кількості істотних вхідних змінних при використанні структури U_4 може в деяких випадках приводити до зменшення кількості операцій переходів. Наслідком цього є зменшення кількості комбінаційних схем в операційній частині ОАП, що приводить до зниження апаратурних витрат у схемі автомата. У тих випадках, коли зменшення кількості ОП супроводжується зменшенням розрядності коду ОП, має місце додаткове зниження апаратурних витрат в ОАП за рахунок спрощення мультиплексора результату, а також в Z-підсхемі за рахунок зменшення числа виходів.

Можна помітити, що операція порозрядного додавання за модулем 2 з константою 10100_2, що є у виразі (3.52) внутрішньою функцією стосовно ОП O_1, в точності збігається з ОП O_2. Це дозволяє синтезувати комбінаційні схеми даних ОП так, як показано на рис. 3.20.

У даній схемі всі лінії зв'язку, за винятком однорозрядної лінії, по якій подається сигнал p_1, є п’ятирозрядними, а в якості констант 2_{10} та 5_{10} подаються їхні двійкові форми 00010_2 та 00101_2 відповідно. Безумовно, можливість використання частини $КС_{d_1}$ в якості $КС_{d_2}$ є в даному прикладі сугубо окремим випадком, однак подібні ситуації слід завжди мати на увазі при оптимізації логічної схеми операційного автомата переходів.

Відзначимо, що у випадку структури U_3 (рис. 3.9) розглянутий вище ефект зменшення кількості ОП за рахунок застосування методу зменшення максимальної кількості вхідних змінних неможливий. Це пов'язане з тим, що в структурі U_3 операції переходів зіставляються не станам автомата, а окремим переходам, причому не має значення, від скількох вхідних змінних залежить той або інший перехід. Якщо у випадку структури U_4 кількість ОП для фрагмента на рис. 3.17 не може бути менше чотирьох, то у випадку структури U_3 такого обмеження немає.

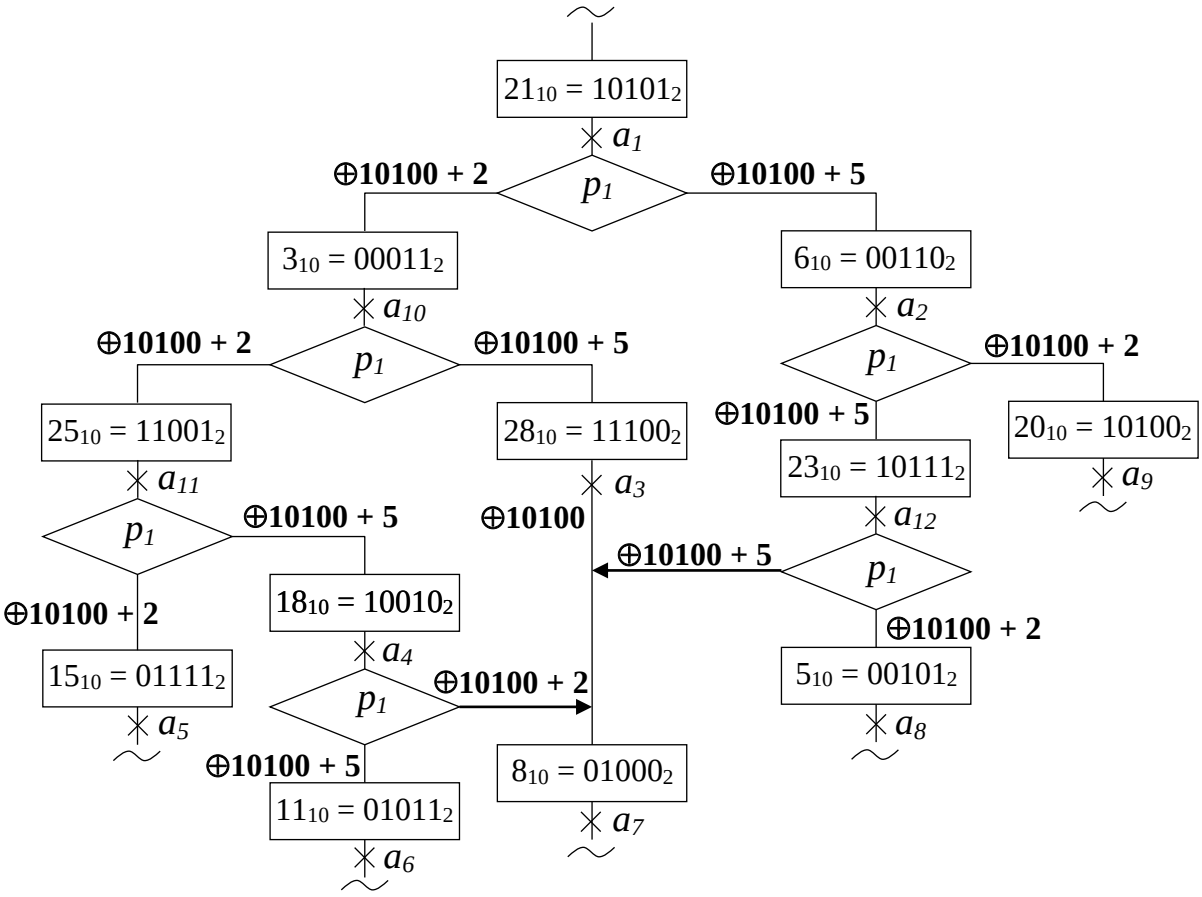


Рисунок 3.19 – Графічне представлення результатів алгебраїчного синтезу (фрагмент ГСА Γ_{3-4})

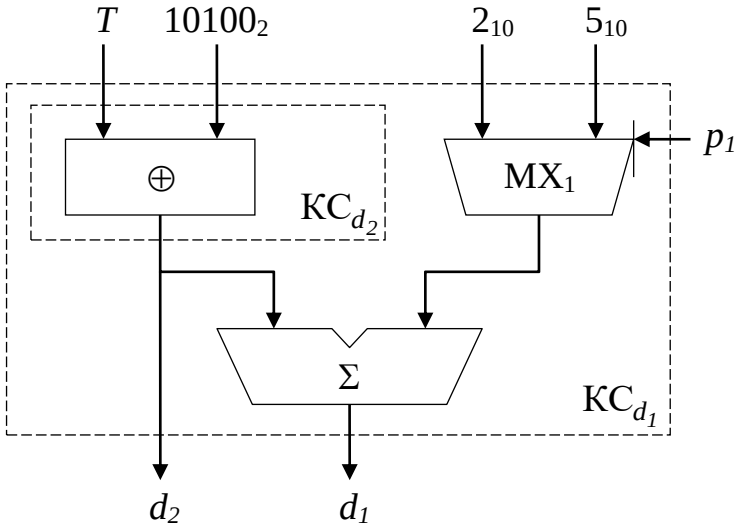


Рисунок 3.20 – Фрагмент функціональної схеми блоку ОЧ, що відповідає операціям переходів O_1 та O_2

У той же час слід враховувати, що зменшення величини G неминуче супроводжується збільшенням числа станів автомата і у деяких випадках може приводити до збільшення розрядності R структурного коду стану. Збільшення R приводить до зростання апаратурних витрат у всіх блоках логічної схеми МПА незалежно від його структурної організації.

3.4 Реалізація функції виходів мікропрограмного автомата з операційним автоматом переходів

3.4.1 Операційна реалізація функції виходів

Синтез схеми формування мікрооперацій, що реалізує функцію виходів мікропрограмного автомата, є частиною структурного синтезу МПА. У канонічному МПА функції переходів і виходів реалізуються у вигляді систем булевих рівнянь, що робить методи їх синтезу схожими. У МПА з ОАП для побудови функції переходів використовується принцип операційного перетворення кодів станів, відповідно до якого перетворення кодів станів здійснюється за допомогою певної множини операцій переходів.

Розглянемо можливість поширення принципу операційного перетворення кодів станів на функцію виходів автомата [39]. У цьому випадку перетворення кодів станів також буде здійснюватися за допомогою певної множини операцій (назвемо їхніми *операціями виходів*, ОВ), однак код поточного стану буде перетворюватися не в код наступного стану, а в мікрокоманду (набір мікрооперацій), відповідну до аргументів функції виходів автомата.

На рис. 3.21 *а* та *б* зображені структурні схеми блоку СФМО для автоматів Мілі й Мура відповідно. Формування вихідних сигналів МПА здійснюється за допомогою множини ОВ $\{O_1, ..., O_W\}$, у якій кожна операція O_i представлена у вигляді окремої комбінаційної схеми і реалізує функцію $Y(O_i)$, значеннями якої є деяка підмножина мікрокоманд вихідної ГСА. Аргументами операцій виходів є, у випадку автомата Мілі, код поточного стану T та сигнали логічних умов X, у випадку автомата Мура – тільки код поточного стану.

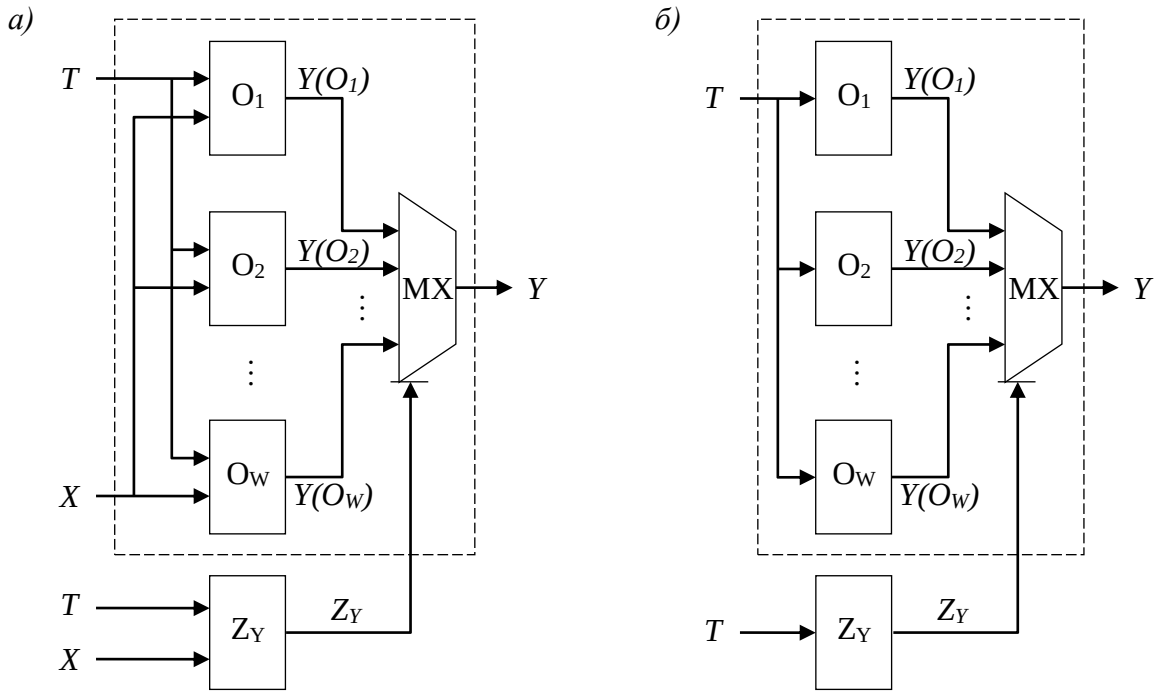


Рисунок 3.21 – Структура блоку СФМО при операційній реалізації функції виходів для автоматів Мілі (*а*) і Мура (*б*)

Для вибору операції виходів використовується мультиплексор результату MX, що керується кодом операції виходів Z_Y, який формується спеціальною Z_Y-підсхемою. На вхід Z_Y-підсхеми надходять сигнали T та X (у випадку автомата Мілі) або тільки сигнали T (у випадку автомата Мура). Таким чином, у випадку автомата Мілі операція виходів зіставляється завжди окремому переходу автомата, у випадку автомата Мура – завжди окремому стану автомата. На виході мультиплексора формується мікрокоманда, що надходить в об'єкт керування мікропрограмного автомата.

Проаналізуємо структури на рис. 3.21.

1. Як відомо, при синтезі МПА мікрокоманди, закріплені за операторними вершинами ГСА, вважаються заданими і не можуть вибиратися довільно. У той же час структурні коди станів не визначаються вихідною ГСА і можуть вибиратися в будь-який зручний спосіб.

При операційній реалізації функції переходів коди станів та операції переходів є варіативними. Це дозволяє не тільки вибирати операції переходів на підставі кодів станів, але й задавати коди станів з урахуванням операцій

переходів. Єдиним обмеженням у цьому випадку є система переходів заданої ГСА.

При операційній реалізації функції виходів коди станів і формовані мікрокоманди є інваріантними: коди станів є обраними в процесі синтезу ОАП, а вміст мікрокоманд заданий вихідною ГСА. Єдиним варіативним елементом є множина операцій, що використовуються для формування мікрокоманд.

Задача формування множини операцій схеми СФМО тривіально вирішується для випадку, коли кожна мікрокоманда формується за допомогою окремої операції, тобто коли число операцій W у СФМО дорівнює числу операторних вершин ГСА. Однак у цьому випадку слід очікувати, що результуюча логічна схема не буде мати переваг в апаратурних витратах у порівнянні з канонічною реалізацією функції виходів у вигляді системи булевих рівнянь. Це пов'язане в тому числі з необхідністю використання в складі схеми СФМО мультиплексора результату та Z_Y-підсхеми, апаратурні витрати в яких, при великій розрядності мікрокоманди і максимально можливому числі операцій виходів, можуть бути порівняні або перевищувати витрати в комбінаційних схемах операцій виходів.

Зниження апаратурних витрат у схемі СФМО при операційному формуванні функції виходів можливе у випадку, якщо деякі операції формують кілька мікрокоманд (за аналогією з тим, як одна операція переходів реалізує кілька переходів автомата). При заданих кодах станів (аргументах функції) і вмісті операторних вершин (значеннях функції) пошук таких операцій зводиться до математичного завдання інтерполяції за базовими крапками [138] на елементах деякої розбивки множини операторних вершин. Для кожної з можливих розбивок може бути отримана множина функцій інтерполяції, що відповідає операціям $O_1 - O_W$ на рис. 3.21. Мінімізація апаратурних витрат у СФМО полягає в пошуку такої розбивки множини операторних вершин і відповідної до цієї розбивки множини операцій виходів, при яких сумарні апаратурні витрати в СФМО (з урахуванням Z_Y-підсхеми) виявляються якомога можливими.

2. При операційній реалізації функції переходів розрядність структурних кодів аргументу і результату будь-якої операції переходів однакова і дорівнює розрядності R структурного коду стану. Дана особливість дозволяє використовувати стандартні R-розрядні функціональні блоки, що мають, як правило, регулярну внутрішню структуру.

Мікрокоманда на виході СФМО утворена N структурними сигналами, відповідними до мікрооперацій $y_1 - y_N$. Оскільки величина N визначається вихідною ГСА та у загальному випадку не залежить від розрядності коду стану R і кількості логічних умов L, при операційній реалізації функції виходів розрядність аргументів операцій виходів, що дорівнює R у випадку автомата Мура та $(R + L)$ у випадку автомата Мілі, може не збігатися з розрядністю N результату. Це вимагає використання для реалізації операцій виходів нестандартних функціональних блоків з нерегулярною структурою.

3. У схемі мультиплексора результату блоку ОЧ (рис. 3.1) розрядність кожного каналу дорівнює розрядності R структурного коду стану автомата, а число каналів відповідає числу використовуваних операцій переходів. У схемі на рис. 3.21 число виходів кожної схеми O_i дорівнює числу мікрооперацій N, що формуються керуючим автоматом. Якщо, наприклад, $N = 100$ (що типово, наприклад, для МПА АЛП [98]), а $R = 10$, то при однаковій кількості операцій переходів у блоці ОЧ ОАП і операцій виходів у схемі СФМО складність мультиплексора результату схеми СФМО буде у 10 разів вища за складність мультиплексора результату блоку ОЧ. Якщо додатково припустити, що кількість W операцій виходів у СФМО буде значно більша за кількість операцій переходів N_d у блоці ОЧ ОАП (що цілком очікуване з урахуванням формування множини операцій виходів за заданими структурними кодами станів і вмістом операторних вершин), то складність мультиплексора на рис. 3.21 на тлі складності інших блоків логічної схеми МПА може виявитися неприйнятно великою.

Розглянемо приклад операційної реалізації функції виходів МПА з ОАП, заданого ГСА Γ_{2-1} (рис. 2.2), що позначена станами автомата Мура. Нехай для

автомата виконаний алгебраїчний синтез функції переходів, відповідно до якого сформовані проміжні й структурні коди станів, наведені в табл. 2.1 та 2.2 відповідно. Аналіз рис. 2.2 показує, що автомат формує $N = 5$ мікрооперацій $y_1 - y_5$.

Поставимо у відповідність кожній мікрокоманді п'ятирозрядний двійковий вектор виду $< y_1, y_2, y_3, y_4, y_5 >$, у якому кожний компонент y_i дорівнює одиниці в тому випадку, якщо в даної МК мікрооперація y_i формується, і нулю, якщо ні. Складемо таблицю виходів МПА (табл. 3.10), що містить наступні стовпці:

a_i – стан автомата;

$K_I(a_i)$ – проміжний код стану a_i відповідно до табл. 2.1;

$K_S(a_i)$ – структурний код стану a_i відповідно до табл. 2.2;

Y – двійковий вектор виду $< y_1, y_2, y_3, y_4, y_5 >$, що відповідає операторній вершині, яка позначена станом a_i.

Таблиця 3.10

Таблиця виходів МПА (ГСА Γ_{2-1})

a_i	$K_I(a_i)$	$K_S(a_i)$	Y
a_0	3	0 1 1	0 0 0 0 0
a_1	2	0 1 0	1 1 0 0 0
a_2	1	0 0 1	0 0 1 0 0
a_3	0	0 0 0	0 0 0 1 1
a_4	5	1 0 1	1 0 1 0 0
a_5	4	1 0 0	0 0 1 1 1

Зіставимо станам a_0, a_1, та a_5 наступну операцію виходів:

$$Y = 00.K_S(a_i) \oplus 00011_2. \qquad (3.58)$$

У даному виразі структурний код стану $K_S(a_i)$ приводиться до п'ятирозрядного формату шляхом додавання двох старших розрядів, рівних нулю. Отриманий

вектор підсумовується за модулем 2 з вектором-константою 00011_2. Результатом операції є п'ятирозрядний вектор, що ототожнюється з формованою мікрокомандою. Наприклад, формування мікрокоманди для стану a_5 здійснюється в наступному порядку:

1) $00.K_S(a_5) = 00100$;

2) $00100 \oplus 00011 = 00111$;

3) $00111 = \{y_3, y_4, y_5\}$.

Станам a_1, a_2, та a_4 поставимо у відповідність наступну ОВ:

$$Y = (\mathrm{L}_2(K_S(a_i)) + 10_{10}) \oplus 01010_2 . \qquad (3.59)$$

Тут структурний код $K_S(a_i)$ спочатку приводиться до п'ятирозрядного двійкового формату шляхом логічного зсуву ліворуч на два розряди (операція L_2). Отриманий п'ятирозрядний двійковий вектор інтерпретується як ціле число та підсумовується з константою 1010. Результат додавання розглядається як п'ятирозрядний двійковий вектор і підсумовується за модулем 2 з вектором-константою 01010_2. Наприклад, формування мікрокоманди для стану a_1 відбувається в наступному порядку:

1) $\mathrm{L}_2(K_S(a_1)) = 01000$;

2) $01000_2 = 8_{10}$;

3) $8_{10} + 10_{10} = 18_{10}$;

4) $18_{10} = 10010_2$;

5) $10010 \oplus 01010 = 11000$;

6) $11000 = \{y_1, y_2\}$.

Таким чином, для реалізації функції виходів МПА, заданого ГСА Γ_{2-1}, достатньо $W = 2$ операції виходів. Закодуємо дані операції двійковим кодом розрядності $\log_2 W = 1$, який у схемі на рис. 3.21, *б* будемо представляти змінною z_1. Нехай $K(O_1) = 0$, $K(O_2) = 0$. Складемо таблицю істинності ZY-підсхеми (табл. 3.11), що включає наступні стовпці:

a_i – стан автомата;

$K_S(a_i)$ – структурний код стану a_i (табл. 2.2);

z_1 – структурний код операції виходів для стану a_i.

Таблиця 3.11

Таблиця істинності Zy-Підсхеми (ГСА Γ_{2-1})

a_i	$K_S(a_i)$	z_1
a_0	0 1 1	0
a_1	0 1 0	1
a_2	0 0 1	1
a_3	0 0 0	0
a_4	1 0 1	1
a_5	1 0 0	0

Згідно з даною таблицею, функції z_1 відповідає наступна ДНФ: $z_1 = \overline{T}_1 T_2 \overline{T}_3 \vee \overline{T}_1 \overline{T}_2 T_3 \vee T_1 \overline{T}_2 T_3$. При реалізації Z_Y-підсхеми в базисі запам'ятовувальних пристроїв таблиця істинності тривіально переформатується в таблицю вмісту ЗП.

Відзначимо, що розглянутий принцип операційної реалізації функції виходів не залежить від способу реалізації функції переходів автомата. Це дозволяє застосовувати його не тільки в МПА з ОАП, але й у МПА з канонічною структурою. В останньому випадку синтез схеми формування мікрооперацій можливий як після, так і до формування функції переходів автомата. Якщо до моменту синтезу СФМО синтез схеми СФП виконаний, то структурні коди станів визначені і у процесі синтезу СФМО виступають у якості вхідних даних. Якщо ж синтез СФМО виконується перед синтезом схеми формування переходів, структурні коди станів можуть бути обрані в процесі синтезу схеми СФМО таким чином, щоб мінімізувати витрати встаткування в схемі СФМО. При цьому реалізація функції переходів буде здійснюватися при заданих структурних кодах станів, що для випадку канонічної структури МПА зазвичай не є критичним.

Викладені вище міркування, що стосуються застосування принципу операційного перетворення кодів станів при реалізації функції виходів мікропрограмного автомата, демонструють теоретичну можливість використання даного підходу, але не дозволяють однозначно судити про його ефективність. Питання доцільності побудови схеми формування мікрооперацій зі структурою, наведеною на рис. 3.21, вимагає вирішення того ж кола задач, яке вирішується в даній роботі відносно функції переходів. Складність і обсяг потенційних досліджень у цьому напрямку не дозволяють провести їх у рамках даної дисертаційної роботи.

3.4.2 Канонічна реалізація функції виходів в автоматі Мілі

Як відомо, у МПА Мілі з канонічною структурою вихідні сигнали залежать від R-розрядного коду поточного стану автомата та L сигналів логічних умов. Величина $(R+L)$ багато в чому визначає елементний базис, що може бути використаний для синтезу схеми СФМО, а також методику синтезу. Для МПА середньої складності $R=6$, $L=30$ [29]. При таких значеннях використання базису ЗП із числом адресних входів $(R+L)$, виявляється нераціональним через велику необхідну ємність ЗП при значній розрідженості використовуваних рядків [31]. Із цієї причини в автоматі Мілі схему СФМО прийнято реалізовувати в базисі, некритичному до великої кількості вхідних сигналів. У якості такого базису можуть використовуватися, наприклад, дискретні комбінаційні елементи, ПЛМ, КЛБ FPGA, CPLD або замовлені великі інтегральні схеми (ASIC).

При канонічному синтезі логічної схеми автомата Мілі традиційно використовується т.зв. спільна реалізація функцій переходів і виходів, що полягає в об'єднаній побудові та спільному використанні булевих термів, що відповідають рядкам структурної таблиці автомата [29, 31]. При цьому досягається значне зменшення сумарних апаратурних витрат у логічній схемі МПА при збереженні швидкодії [29].

Наведені у [29] значення площ матриць ПЛМ, отримані для автомата середньої складності, показують, що при спільній реалізації функцій переходів і

виходів з $26{,}8 \cdot 10^3$ біт сумарної площі матриць ПЛМ, необхідної для реалізації всієї логічної схеми автомата, на функцію виходів припадає $24{,}4 \cdot 10^3$ біт, що становить близько 90 % сумарної площі. При цьому реалізація логічної схеми функції переходів складає близько 10 % від сумарної площі, або ж 1/9 від площі реалізації функції виходів. Таким чином, канонічна реалізація однієї лише функції виходів за апаратурними витратами може бути порівнянна з реалізацією всієї логічної схеми автомата.

При операційній реалізації функції переходів булеві терми, відповідні до рядків ПСТ, не формуються. Даний факт унеможливлює спільну реалізацію функцій переходів і виходів. Позначимо символом $H_{ОАП}$ чисельно виражені витрати апаратури на реалізацію функції переходів у вигляді ОАП, символом H_Y – витрати апаратури на реалізацію функції виходів МПА Мілі у вигляді системи канонічних рівнянь. Будемо спрощено вважати, що при спільній реалізації функцій переходів і виходів апаратурні витрати на реалізацію частини схеми, що належить тільки до функції переходів, становлять близько 1/9 частини витрат у схемі функції виходів. Використаємо це співвідношення для МПА Мілі з ОАП:

$$H_{ОАП} = \frac{1}{9} H_Y . \qquad (3.60)$$

При виконанні даної рівності витрати апаратури в МПА Мілі з ОАП та канонічною реалізацією функції виходів будуть порівнянні з витратами в схемі канонічного МПА Мілі. Ситуація, при якій логічна схема МПА Мілі з ОАП і канонічною реалізацією функції виходів буде вигравати за апаратурними витратами у схеми МПА Мілі з канонічною структурою та спільною реалізацією функцій переходів і виходів, визначається нерівністю

$$H_{ОАП} < \frac{1}{9} H_Y . \qquad (3.61)$$

Таким чином, канонічна реалізація функції виходів у мікропрограмному автоматі Мілі з операційною реалізацією функції переходів може виявитися доцільною у випадку, коли витрати апаратури на реалізацію функції виходів

багаторазово (в 10 і більше разів) перевищують витрати на реалізацію функції переходів у вигляді ОАП.

Підкреслимо, що умову (3.61) слід розглядати лише у випадку спільної реалізації функцій переходів і виходів у МПА Мілі. Разом з тим існує ряд методів оптимізації логічної схеми автомата Мілі, що використовують роздільну реалізацію даних функцій. Можуть бути виділені наступні три групи методів [36]:

– методи структурної редукції, при яких використовується кілька рівнів логічного перетворення інформації в схемі автомата;

– методи гетерогенної реалізації, що полягають у використанні різного елементного базису для реалізації схем різних рівнів;

– алгоритмічні методи, засновані на оптимальному кодуванні логічних об'єктів у схемі автомата.

Застосування відомих методів оптимізації логічної схеми МПА уможливлює розробку нових структур і методів синтезу мікропрограмних автоматів з операційним автоматом переходів. Дані питання утворюють окремий напрямок досліджень і в дисертації не розглядаються.

3.4.3 Канонічна реалізація функції виходів в автоматі Мура

Як відомо, у канонічній структурі МПА Мура вихідні сигнали залежать тільки від коду поточного стану, що припускає, у загальному випадку, роздільну реалізацію функцій переходів і виходів. Це дозволяє вважати кількість апаратурних витрат у схемі СФМО величиною, однаковою для канонічного МПА та МПА з ОАП, і порівнювати дані структури за кількістю апаратурних витрат у схемі, що реалізує функцію переходів. Спрощена умова виграшу в апаратурних витратах схеми МПА Мура з ОАП у порівнянні з канонічним МПА Мура виражається наступною нерівністю:

$$H_{ОАП} < H_{СФП}, \tag{3.62}$$

де $H_{СФП}$ – чисельно виражені витрати апаратури в схемі формування переходів МПА з канонічною структурою.

Як і для автомата Мілі, для автомата Мура відомий ряд методів оптимізації логічної схеми, що ставлять метою зменшення апаратурних витрат [35]. Умови можливості і доцільності їх застосування в мікропрограмному автоматі з операційним автоматом переходів не є очевидними і вимагають окремих досліджень.

3.5 Висновки до третього розділу

1. В третьому розділі вирішене наукове завдання розробки структури і математичної моделі мікропрограмного автомата, заснованих на запропонованому принципі перетворення кодів станів за допомогою кінцевої множини операцій і орієнтованих на зменшення апаратурних витрат у логічній схемі автомата, а також завдання модифікації розробленої структури мікропрограмного автомата із застосуванням відомих методів оптимізації апаратурних витрат.

2. При використанні в мікропрограмному автоматі принципу операційного перетворення кодів станів структура схеми формування переходів становить собою сукупність окремих комбінаційних схем, що реалізують часткові структурні функції переходів. Існування проміжних алгебр, ізоморфних структурним підалгебрам переходів, дозволяє ототожнювати дані комбінаційні схеми з операціями переходів. За аналогією з операційним автоматом, сукупність комбінаційних схем, що реалізують множину операцій переходів, названа операційною частиною (рис. 3.1) і утворює, разом з регістром пам'яті МПА, операційний автомат переходів (рис. 3.2).

3. У кожному такті роботи МПА операційний автомат переходів виконує завжди тільки одну операцію із множини операцій, що реалізуються операційною частиною. Задача вибору потрібної операції вирішується шляхом мультиплексування виходів комбінаційних схем під керуванням спеціального коду операції переходів Z (рис. 3.1). Код Z формується спеціальною Z-підсхемою на підставі коду поточного стану і сигналів логічних умов. Таким чином,

реалізація функції переходів МПА забезпечується спільною роботою операційного автомата переходів і Z-підсхеми.

4. Уведення в структуру МПА операційного автомата переходів і Z-підсхеми приводить до нової структури МПА, яка в дисертаційній роботі названа мікропрограмним автоматом з операційним автоматом переходів (структура U_1, рис. 3.4). Математична модель МПА з ОАП визначається виразом (3.9) і становить собою систему ізоморфізмів часткових абстрактних і структурних підалгебр та проміжних алгебр переходів.

5. Однією з особливостей операційного автомата переходів є використання в його структурі єдиного регістру, що одночасно виконує функцію регістру пам'яті мікропрограмного автомата. У будь-якій операції ОАП регістр пам'яті виступає одночасно в якості регістру вихідних даних і регістру результату. Дана особливість дозволяє класифікувати операційний автомат переходів одночасно як автомат з індивідуальними (I-автомат) та узагальненими (M-автомат) мікрооперaціями. Також ОАП може структурно розглядатися як операційний автомат з канонічною структурою (K-автомат), що має максимально можливу продуктивність. Разом з тим, ОАП не може бути класифікований як IM-автомат, оскільки можливість виконання в кожному такті лише однієї операції не припускає наявності у реалізованих операцій функціональної або структурної сумісності.

6. У структурі МПА U_1 операції переходів зіставляються кожному переходу автомата. Зіставлення операцій переходів станам автомата приводить до структури U_2 (рис. 3.6). У даній структурі досягається зниження апаратурних витрат в Z-підсхемі за рахунок виключення сигналів логічних умов із числа її вхідних сигналів. Разом з тим, зіставлення операцій переходів станам автомата може приводити до ускладнення та збільшення кількості комбінаційних схем операцій переходів в операційній частині ОАП.

7. Застосування до структури U_1 відомого методу заміни вхідних змінних дозволяє зменшити число вхідних сигналів в ОАП і Z-підсхемі та приводить до

структури U_3 (рис. 3.9). Застосування даного методу до структури U_2 тягне зменшення числа вхідних сигналів схеми ОАП і приводить до структури U_4 (рис. 3.12). В обох випадках у структуру МПА з ОАП додається спеціальна М-підсхема, що перетворює множину логічних умов X у множину змінних P меншої потужності. Ефективність даного підходу залежить від кількості вхідних змінних, що суттєво впливають на автоматні переходи в заданій ГСА.

8. Підвищення ефективності використання методу заміни вхідних змінних можливе за рахунок зменшення кількості вхідних змінних, що суттєво впливають на автоматні переходи. Це можливе за рахунок уведення в ГСА додаткових станів, розташовуваних у гілках, що з'єднують вихід однієї умовної вершини зі входом іншої умовної вершини. Позитивний ефект від цього підходу полягає у зниженні числа входів ОАП і Z-підсхеми, але супроводжується збільшенням часу виконання мікропрограми. Також збільшення кількості станів може приводити до збільшення розрядності структурного коду стану, що збільшує апаратурні витрати в усіх блоках логічної схеми мікропрограмного автомата.

9. Функція виходів мікропрограмного автомата може бути реалізована в канонічний спосіб або за допомогою множини операцій виходів за аналогією з операційною реалізацією функції переходів. При цьому спосіб реалізації функції переходів автомата не залежить від способу реалізації функції виходів і може бути як канонічним, так і операційним. Комбінація канонічної реалізації обох функцій відповідає канонічній структурі МПА. Інші комбінації приводять до нових структур МПА, методам їх синтезу та оцінки ефективності.

10. Запропонований в дисертаційній роботі принцип операційного перетворення кодів станів визначає ряд науково-дослідних напрямків:

– операційна реалізація функції виходів МПА;

– застосування відомих методів оптимізації мікропрограмних автоматів до структур МПА з операційним автоматом переходів;

– використання принципу операційного перетворення кодів станів в інших відомих класах пристроїв керування.

4 СТРУКТУРНИЙ СИНТЕЗ МІКРОПРОГРАМНОГО АВТОМАТА З ОПЕРАЦІЙНИМ АВТОМАТОМ ПЕРЕХОДІВ

4.1 Основні етапи структурного синтезу

Запропонований В. М. Глушковим канонічний метод структурного синтезу полягає в синтезі його комбінаційної логічної схеми [87]. При цьому відносно канонічної структури МПА В. М. Глушковим зроблене уточнення, що полягає в завданні функцій переходів і виходів автомата у формі систем канонічних булевих рівнянь. Дані системи прийнято формувати згідно до структурної таблиці автомата, методика побудови якої для заданої ГСА викладена в [29]. Присутні в канонічній структурі МПА схема формування переходів, що реалізує функцію переходів, і схема формування мікрооперацій, що реалізує функцію виходів, синтезуються відповідно до систем канонічних рівнянь відомим способом [87].

Застосування канонічного методу синтезу зводить задачу структурного синтезу МПА з ОАП до задачі синтезу його комбінаційної логічної схеми. Під синтезом логічної схеми автомата будемо розуміти синтез логічних схем усіх блоків, що утворюють структуру автомата. Для структур $U_1 - U_4$ такими блоками є операційний автомат переходів, Z-підсхема та схема формування мікрооперацій. Структури U_3 і U_4 додатково містять M-підсхему, що реалізує кодування вхідних сигналів автомата.

Нехай деякий абстрактний автомат заданий абстрактною алгеброю (2.3), сигнатура (2.5) якої утворена функцією переходів (2.11) і функцією виходів, яка задана виразом (2.15) у випадку автомата Мілі або виразом (2.17) у випадку автомата Мура. Нехай також деякий структурний автомат заданий структурною алгеброю (2.54), сигнатура якої утворена структурною функцією переходів (2.51) і структурною функцією виходів, яка задана виразом (2.52) у випадку автомата Мілі або виразом (2.53) у випадку автомата Мура.

У п. 3.1 даної роботи показано, що еквівалентність даних автоматів випливає з існування системи ізоморфізмів (3.9). Розглянемо процес побудови логічної схеми МПА з ОАП зі структурою U_1 (рис. 3.4) за відомою системою (3.9).

1. Ізоморфізм $G_\delta \leftrightarrow G_d$ системи (3.9) визначає структурні коди станів і їх розрядність, яка враховується при синтезі всіх вузлів логічної схеми МПА з ОАП.

2. Кожна трійка ізоморфізмів $G_{\delta_i} \leftrightarrow G_{I_i}$, $G_{I_i} \leftrightarrow G_{d_i}$ та $G_{\delta_i} \leftrightarrow G_{d_i}$, представлена в системі (3.9) єдиним ізоморфізмом $G_{\delta_i} \leftrightarrow G_{I_i} \leftrightarrow G_{d_i}$, виступає в якості вхідних даних для побудови комбінаційної схеми КС_{d_i}, яка входить до блоку ОЧ операційного автомата переходів (рис. 3.1). При цьому, у загальному випадку, можливі різні способи схемної реалізації КС_{d_i}. При виборі тієї або іншої реалізації слід керуватися як критеріями оптимальності, встановленими до логічної схеми МПА, так і наявними технічними можливостями.

3. Величина N_I в системі (3.9) визначає кількість комбінаційних схем, що реалізують операції переходів проміжних алгебр. Як відзначено в п. 3.1, кількість N_d часткових структурних функцій переходів може як дорівнювати значенню N_I (якщо всі переходи автомата реалізуються в рамках наявного безлічі операцій переходів), так і бути на одиницю більшою (якщо частина переходів автомата реалізується канонічним способом). Значення N_d однозначно визначається з аналізу системи (3.9): якщо існують переходи, які присутні у вигляді кортежів (2.12) у функції δ, але відсутні у всіх часткових функціях δ_i, то $N_d = N_I + 1$, інакше $N_d = N_I$.

Величина N_d визначає розрядність R_Z коду операцій переходів як величину $R_Z = \lceil \log_2(N_d) \rceil$. Значення N_d та R_Z впливають на процес формування множини $K(O)$ кодів операцій переходів, що не є істотним для системи (3.9). Множина $K(O)$, а також величини N_d та R_Z використовуються при синтезі Z-підсхеми і мультиплексора результату (рис. 3.1).

4. Ізоморфізм $G_\lambda \leftrightarrow G_l$ не торкається функції переходів автомата, у зв'язку із чим не має відношення до операційного автомата переходів. Із цього погляду система ізоморфізмів (3.9) не накладає яких-небудь обмежень на спосіб реалізації функції виходів автомата. Проте, об'єднання ізоморфізмів $G_\lambda \leftrightarrow G_l$ та $G_\delta \leftrightarrow G_d$ в одній системі вимагає, щоб синтез схеми формування мікрооперацій виконувався для значень структурних кодів станів і структурних кодів вхідних сигналів, обумовлених ізоморфізмом $G_\delta \leftrightarrow G_d$.

Зроблені міркування дозволяють стверджувати, що побудова системи ізоморфізмів (3.9) передує синтезу логічної схеми МПА з ОАП. Назвемо процес побудови для заданого автомата системи (3.9) *алгебраїчним синтезом* МПА з ОАП. Тоді структурний синтез МПА з ОАП може бути представлений *двома етапами*:

1. Алгебраїчний синтез автомата.

2. Синтез логічної схеми автомата.

Дані етапи приводять до двох наступних задач структурного синтезу МПА з ОАП:

– *задача алгебраїчного синтезу автомата*;

– *задача синтезу логічної схеми автомата відповідно до результатів алгебраїчного синтезу*.

4.2 Алгебраїчний синтез мікропрограмного автомата з операційним автоматом переходів

4.2.1 Постановка задачі алгебраїчного синтезу

Під *задачею алгебраїчного синтезу мікропрограмного автомата з операційним автоматом переходів* будемо розуміти задачу побудови системи ізоморфізмів виду (3.9). Її суть полягає у виявленні та встановленні закономірностей у перетворенні структурних кодів станів і вхідних сигналів при реалізації автоматних переходів [11].

Зроблені в п. 2.3 узагальнення 1 і 2 показують, що для одного і того ж автомата може бути сформовано, у загальному випадку, кілька різних систем ізоморфізмів (3.9). Назвемо будь-яку побудовану для заданого автомата систему виду (3.9), у якій $N_I > 0$, *формальним розв'язком задачі алгебраїчного синтезу МПА з ОАП*, а множину усіх можливих систем – *множиною формальних розв'язків* даної задачі для заданого автомата [24]. Система (3.9), у якій $N_I = 0$, не є формальним розв'язком, оскільки в ній усі переходи реалізуються канонічним способом без використання операційного перетворення кодів станів, а розглянуті вище структури МПА з ОАП вироджуються в канонічну структуру.

Виключимо із множини формальних розв'язків ті системи, при використанні яких апаратурні витрати в логічній схемі МПА з ОАП більші або дорівнюють витратам у логічній схемі еквівалентного МПА з канонічною структурою. Також виключимо системи, при яких результуюча схема МПА з ОАП не задовольняє яким-небудь додатковим критеріям проектування. Отриману в результаті множину систем ізоморфізмів виду (3.9) назвемо *множиною ефективних розв'язків задачі алгебраїчного синтезу МПА з ОАП*, розуміючи під ефективністю виграш в апаратурних витратах.

Можна стверджувати, що в множині ефективних розв'язків існує така система виду (3.9), для якої логічна схема МПА з ОАП має мінімальні витрати апаратури серед усіх систем даної множини. Назвемо таку систему *оптимальним розв'язком задачі алгебраїчного синтезу МПА з ОАП*. Теоретично є припустимим існування декількох оптимальних розв'язків, що утворюють *множину оптимальних розв'язків задачі алгебраїчного синтезу МПА з ОАП*.

У загальному випадку в процесі алгебраїчного синтезу МПА з ОАП формування множини оптимальних розв'язків не є обов'язковим. Не є обов'язковим і формування всіх елементів множини ефективних розв'язків. Проте, збільшення кількості знайдених ефективних розв'язків сприяє підвищенню результативності структурного синтезу МПА з ОАП, оскільки кожний черговий знайдений розв'язок може виявитися кращим за усі розв'язки, знайдені раніше.

Будь-який формальний розв'язок задачі алгебраїчного синтезу МПА з ОАП припускає деякі вхідні дані і результати. Нехай абстрактний автомат заданий двійкою (2.9). Тоді елементи G_δ та G_λ відомі до початку формування системи ізоморфізмів (3.9) і при алгебраїчному синтезі можуть розглядатися в якості вхідних даних. У деяких випадках до вхідних даних можуть бути також віднесені наступні фактори:

1. Критерій оптимізації логічної схеми автомата.

Крім основного критерію оптимізації, яким у даній роботі виступають витрати апаратури в логічній схемі МПА, в процесі проектування можуть бути встановлені різні обмеження до характеристик результуючої логічної схеми автомата, наприклад:

– мінімально достатня швидкодія;

– максимально припустимий час виконання алгоритму;

– максимально припустиме енергоспоживання;

– сумісність із тим або іншим елементним базисом;

– надійність.

Критерій оптимізації і встановлені обмеження можуть впливати як на алгоритмічну реалізацію окремих етапів структурного синтезу МПА з ОАП, так і на їхню послідовність.

2. Множина операцій переходів.

У деяких випадках множина ОП або її частина не формується в процесі алгебраїчного синтезу автомата, а заданою і незмінною на початку синтезу. Прикладом може бути ситуація, коли в якості операційного автомата переходів виступає деякий стандартний АЛП, що містить фіксований набір функціональних блоків. У цьому випадку множина операцій переходів не може вибиратися довільно і береться із множини операцій даного АЛП. Подібні обмеження, що накладаються на множину операцій переходів, не можуть привести до неможливості синтезу МПА з ОАП – ті переходи, які не можуть бути реалізовано однією з ОП, завжди можуть бути реалізовані системою канонічних рівнянь у рамках окремої підалгебри переходів. Проте, наслідком обмежень може бути те,

що результуюча схема автомата не буде задовольняти заданим критеріям проектування.

3. Структурні коди станів і вхідних сигналів.

З теоретичної точки зору припустима ситуація, коли структурні коди станів задані до початку алгебраїчного синтезу автомата. Це можливо, наприклад, у тому випадку, якщо схема формування мікрооперацій з якихось причин синтезована заздалегідь у вигляді окремого модуля. Така схема розрахована на використання у якості кодів станів строго визначених двійкових векторів. Якщо схема формування мікрооперацій синтезована для автомата Мілі, то, разом зі структурними кодами станів, заданими виявляються і структурні коди вхідних сигналів. У випадку автомата Мура вхідні сигнали не беруть участь у формуванні вихідних сигналів, і їхні структурні коди не визначаються схемою СФМО.

Що стосується результатів вирішення задачі алгебраїчного синтезу, то в загальному випадку до них можуть бути віднесені все без винятку елементи системи (3.9). При цьому окремо слід виділити можливість модифікації елементів G_δ та G_λ, що є еквівалентним модифікації заданого абстрактного автомата. Дана можливість обумовлена існуванням ряду відомих методів перетворення вхідної ГСА [29, 120], результатом яких є збільшення кількості станів і переходів автомата. Якщо для алгоритму, що імплементується МПА з ОАП, такі перетворення припустимі, їх використання в процесі алгебраїчного синтезу автомата може виявитися доцільним і сприяти підвищенню ефективності методів структурного синтезу МПА.

4.2.2 Методологія алгебраїчного синтезу

У системі ізоморфізмів (3.9) зручно виділяти три алгебраїчні рівні: рівень абстрактних алгебр, проміжних алгебр і структурних алгебр.

На рівні абстрактних алгебр розташовані елементи, що належать до абстрактного автомата: абстрактна алгебра переходів $G_\delta = <\{A, Z\}, \{\delta\}>$ і абстрактна алгебра виходів $G_\lambda = <\{A, Z, W\}, \{\lambda\}>$.

Елементами рівня проміжних алгебр є множина проміжних кодів станів $K_I(A)=\{K_{I_1}(A_{I_1}),...,K_{I_{N_I}}(A_{I_{N_I}})\}$, множина операцій переходів $O=\{O_1,...,O_{N_I}\}$ і множина $K_I(Z)=\{K_{I_1}(Z_{I_1}),...,K_{I_{N_I}}(Z_{I_{N_I}})\}$ проміжних кодів вхідних сигналів. Підкреслимо, що в загальному випадку $|K_I(A)|>|A|$ та $|K_I(Z)|>|Z|$. Дані елементи утворюють множину проміжних алгебр переходів $G_I=\{G_{I_1},...,G_{I_{N_I}}\}$.

Елементами рівня структурних алгебр є множина $K_S(A)$ структурних кодів станів, множина $K_S(Z)$ структурних вхідних сигналів і структурна функція переходів *d*, що утворюють структурну алгебру переходів $G_d=<\{K_S(A),K_S(Z)\},\{d\}>$. На цьому ж рівні перебуває структурна алгебра виходів $G_l=<\{K_S(A),K_S(Z),K_S(W)\},\{l\}>$.

Під *методом алгебраїчного синтезу МПА з ОАП* будемо розуміти певну послідовність етапів, в результаті виконання якої може бути отриманий формальний розв'язок задачі алгебраїчного синтезу МПА з ОАП, що виражається системою ізоморфізмів (3.9) [11]. Результатом окремого етапу будемо вважати один або кілька сформованих елементів системи (3.9).

При розробці методу алгебраїчного синтезу МПА з ОАП слід мати на увазі, що для окремого етапу припустимо, у загальному випадку, кілька різних реалізацій. Наприклад, якщо потрібно вибрати проміжні коди станів при заданій множині операцій переходів, можна скористатися такими способами:

– повний перебір варіантів;

– частковий перебір варіантів;

– алгоритми, що не використовують перебір варіантів;

– алгоритми, що допускають перетворення ГСА

та інші. Подібні реалізації можуть різнитися як часом виконання, так і якістю результату.

Також при розробці методу алгебраїчного синтезу повинні враховуватися фактори, що виступають в якості вхідних даних. Вони можуть впливати як на реалізацію етапів синтезу, так і на їхню послідовність. Наприклад, у випадку

заданих структурних кодів станів етап формування множини $K_S(A)$ буде передувати етапу формування множини проміжних кодів станів $K_I(A)$.

Таким чином, можна говорити про потенційне існування *множини методів алгебраїчного синтезу МПА з ОАП*, що характеризуються різною ефективністю з позицій витрат часу і результативності. Методи можуть відрізнятися один від одного:

– вихідними даними;

– множиною етапів синтезу;

– послідовністю етапів;

– алгоритмічною реалізацією окремих етапів.

Варіативність цих складових приводить до *задачі розробки методів алгебраїчного синтезу МПА з ОАП*. У рамках даної задачі повинні розглядатися такі питання, як:

– пошук принципів і підходів до організації та побудови методів алгебраїчного синтезу МПА з ОАП;

– розробка методів, прийомів, способів і процедур, що використовуються при реалізації окремих етапів алгебраїчного синтезу;

– розробка формалізованих методів алгебраїчного синтезу МПА з ОАП;

– аналіз ефективності методів алгебраїчного синтезу і визначення області їхнього застосування

та інші. Кожне з цих питань не має однозначного вирішення і утворює окремий напрямок досліджень.

Назвемо наукову область, що охоплює будь-які теоретичні і практичні питання, що мають відношення до алгебраїчного синтезу МПА з ОАП, *методологією алгебраїчного синтезу мікропрограмних автоматів з операційним автоматом переходів* [14, 16]. Тут термін «методологія» будемо трактувати як сукупність принципів і підходів до виконання певної діяльності (теоретичний аспект), а також як сукупність методів, прийомів, способів і процедур відповідної діяльності (практичний аспект) [9 135, 165, 167, 168].

У рамках методології алгебраїчного синтезу МПА з ОАП у дисертаційній роботі пропонується ряд методів, використання яких може виявитися доцільним при розробці методів синтезу даного класу автоматів. До них належать:

1. Структурне представлення процесу синтезу МПА з ОАП.
2. Алгебраїчний синтез методом повного перебору.
3. Додавання станів.
4. Урахування імовірностей істинності логічних умов.
5. Збільшення розрядності структурного коду стану.

4.2.3 Структурне представлення процесу алгебраїчного синтезу

Неоднозначність у послідовності і способах формування елементів системи ізоморфізмів (3.9) ускладнює процес розробки формальних методів алгебраїчного синтезу МПА з ОАП. Розумінню цього процесу можуть сприяти прийоми, запозичені із системного аналізу. Прикладом є когнітивний аналіз, що припускає представлення деякого процесу у формі т.зв. когнітивних карт, що представляють структуру взаємозалежних явищ досліджуваної ситуації [108, 231]. Суть даного методу полягає у представленні деякої системи у вигляді орієнтованого графа, вершинами якого є окремі елементи системи, а ребра виражають причинно-наслідкові зв'язки між ними [179].

Нехай у якості вхідних даних для синтезу МПА з ОАП задана множина $O = \{O_1, ..., O_{N_I}\}$ операцій переходів. У цьому випадку проміжні коди станів $K_I(A)$ і проміжні коди вхідних сигналів $K_I(Z)$ формуються на підставі множини O і абстрактної алгебри переходів G_δ. Вплив множини O і алгебри G_δ на елементи $K_I(A)$ та $K_I(Z)$ відобразимо графічно так, як показано на рис. 4.1, *а*. Тут елементи системи (3.9) показані у вигляді вершин графа, з'єднаних дугами. Кожна дуга є ознакою того, що елемент системи (3.9), вказаний в «вихідній» вершині, деяким чином бере участь у формуванні елемента, вказаного у «вхідній» вершині [16, 46, 195].

Якщо множина O не є «жорстко» заданою, воно формується, виходячи, у тому числі, із заданої ГСА. У системі ізоморфізмів (3.9) структура ГСА формально представлена абстрактною алгеброю переходів. Її вплив на множину операцій переходів можна показати, додавши дугу від вершини G_δ до вершини O (рис. 4.1, *б)*.

Якщо для алгоритму, що імплементується абстрактним автоматом, припустимі які-небудь модифікації, вони можуть бути графічно виражені дугами, спрямованими до вершини G_δ від інших вершин графа. Наприклад, на рис. 4.1, *в* ребро від вершини O до вершини G_δ відбиває вплив на абстрактну алгебру множини операцій переходів. Одночасне існування ребра від вершини G_δ до вершини O говорить про те, що множина O не є заданою і формується в процесі алгебраїчного синтезу. У загальному випадку, наявність у якої-небудь вершини вхідного ребра говорить про можливість модифікації елемента, відповідного до даної вершини, у процесі алгебраїчного синтезу МПА з ОАП.

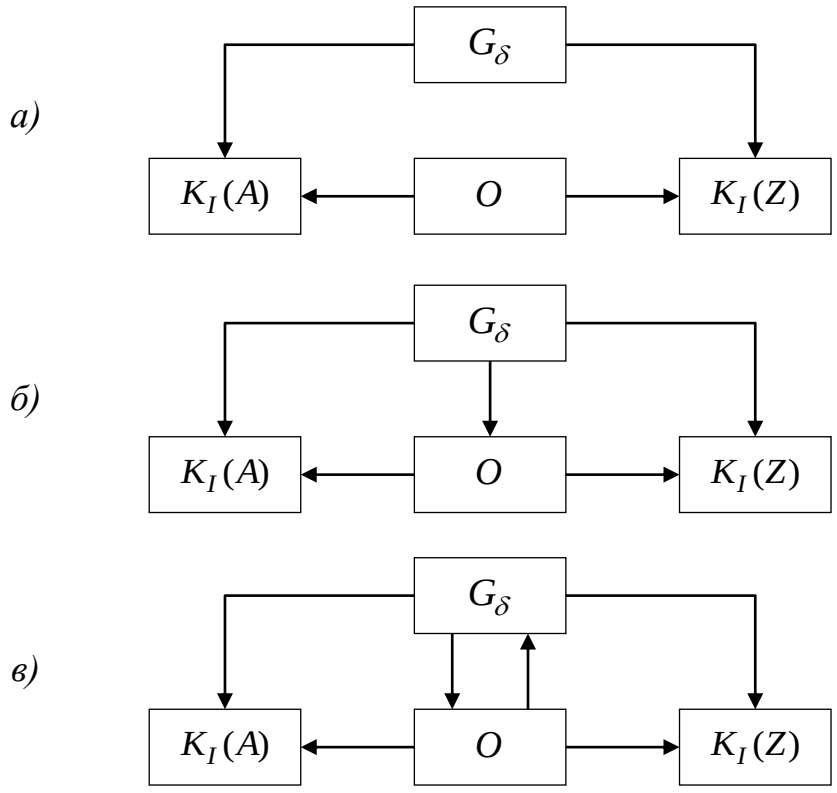


Рисунок 4.1 – Приклади графічного представлення взаємного впливу елементів системи ізоморфізмів (3.9) в процесі алгебраїчного синтезу

Топологія графа, зображеного на рис. 4.1, *в*, дозволяє виділити в ньому визначені в п. 4.2.2 рівень абстрактних алгебр (вершина G_δ) і рівень проміжних алгебр (вершини $K_I(A)$, O та $K_I(Z)$). Додамо в граф рівень структурних алгебр, представивши його вершинами $K_S(A)$, δ та $K_S(Z)$. Додані вершини зв'яжемо з іншими вершинами графа та між собою так, як показано на рис. 4.2. Отриманий граф дозволяє затверджувати про наступне:

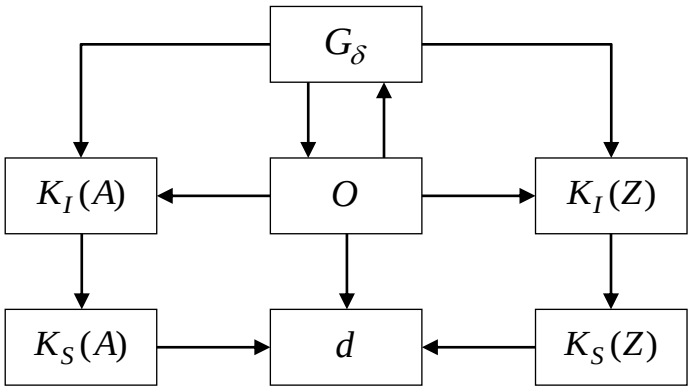


Рисунок 4.2 – Приклад графічного представлення взаємного впливу елементів трьох алгебраїчних рівнів системи ізоморфізмів (3.9)

1. Множина операцій переходів O не є «жорстко» заданою і формується в процесі алгебраїчного синтезу. Про це свідчить ребро, що входить до вершини O.

2. Проміжні й структурні коди станів і вхідних сигналів не є «жорстко» заданими й формуються в процесі алгебраїчного синтезу. Про це свідчать ребра, що входять у відповідні вершини.

3. Для алгоритму, реалізованого синтезованим МПА, припустимі певні модифікації. Про це свідчить ребро, що входить у вершину G_δ.

4. Формування множини операцій переходів є первинним стосовно формування множин структурних кодів станів і вхідних сигналів. Про це говорить недосяжність вершини O з вершин $K_S(A)$ та $K_S(Z)$.

5. Кінцевим етапом алгебраїчного синтезу є формування структурної функції переходів d. Про це говорить відсутність для вершини d вихідних ребер.

Граф, наведений на рис. 4.2, не є відображенням якогось конкретного методу синтезу, оскільки не містить інформації про етапи синтезу, їх послідовність і реалізацію. Разом з тим, граф містить інформацію наступного роду:

– які елементи системи ізоморфізмів (3.9) не формуються в процесі алгебраїчного синтезу і виступають лише в якості вхідних даних (відповідні їм вершини не мають вхідних дуг);

– які елементи можуть формуватися в процесі алгебраїчного синтезу (відповідні їм вершини мають хоча б одну вхідну дугу);

– які елементи можуть безпосередньо впливати на формування того або іншого елемента системи (3.9) (визначається дугами графа);

– які елементи можуть впливати на формування того або іншого елемента системи (3.9) (визначається досяжністю відповідної вершини з інших вершин);

– формування яких елементів може бути віднесене до заключного етапу алгебраїчного синтезу (відповідні їм вершини не мають вихідних дуг).

Дана інформація може бути використана при розробці і класифікації методів алгебраїчного синтезу МПА з ОАП. Варіативність у підходах до формування множини етапів синтезу, їх послідовності і реалізації дозволяє зв'язати з розглянутим графом потенційну множину методів алгебраїчного синтезу.

Помітимо, що граф на рис. 4.2 містить не всі елементи системи ізоморфізмів (3.9). Додамо до графу вершини, відповідні до елементів G_λ та G_l, зв'язавши їх з іншими вершинами графа в такий спосіб:

1. Дуга, спрямована від вершини *O* до вершини G_δ, припускає модифікацію абстрактної алгебри переходів G_δ. Оскільки носії *A* та *Z* алгебри G_δ є одночасно носіями абстрактної алгебри виходів G_λ, їх зміна в рамках алгебри G_δ тягне відповідні зміни алгебри G_λ. Дана властивість може бути виражена дугою, спрямованою від вершини G_δ до вершини G_λ.

2. Структурна алгебра виходів G_l формується відповідно до абстрактної алгебри виходів G_λ. Також при формуванні G_l повинні враховуватися структурні коди станів $K_S(A)$, а у випадку автомата Мілі – також і структурні коди вхідних сигналів $K_S(Z)$. Вплив даних елементів на елемент G_l виражається на графові відповідними дугами. Одержуваний в результаті граф зображений на рис. 4.3.

У контексті алгебраїчного синтезу немає істотної потреби розглядати даний граф як математичний об'єкт через незастосування до нього яких-небудь методів аналізу або перетворення графів. Виділення на графові трьох алгебраїчних рівнів – абстрактного, проміжного і структурного – дозволяє розглядати граф як деяку *структуру*, розуміючи під даним терміном сукупність взаємного розташування і стійких зв'язків своїх складових частин (блоків структури). Назвемо граф, зображений на рис. 4.3, *структурою процесу алгебраїчного синтезу МПА з ОАП зі змінюваною абстрактною алгеброю переходів* і умовимось позначати її в рамках дисертаційної роботи символом M_1.

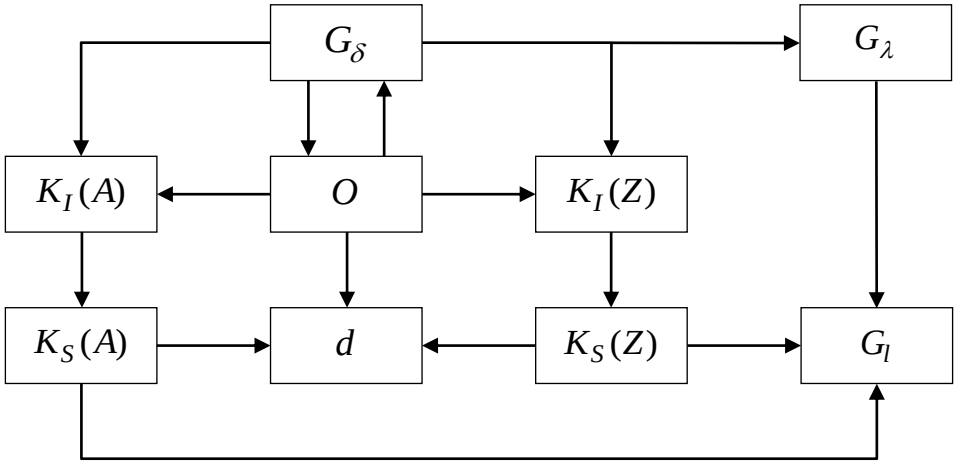


Рисунок 4.3 – Структура процесу алгебраїчного синтезу МПА з ОАП зі змінюваною абстрактною алгеброю переходів (структура M_1)

Говорячи про множину методів, що можуть бути позначені структурою M_1, слід підкреслити наявність у них деяких загальних властивостей, обумовлених даною структурою. Так, відсутність блоків без вхідних зв'язків говорить про те,

що жоден з елементів системи (3.9) (у тому числі абстрактні алгебри переходів і виходів) не є «жорстко» заданим і формується в процесі алгебраїчного синтезу.

Вимоги до процесу алгебраїчного синтезу, обумовлені структурою, що зображена на рис. 4.3, не є єдино можливими. Нехай одним з вимог є неприпустимість будь-яких змін абстрактної алгебри переходів при одночасній можливості зміни інших елементів системи (3.9). В цьому випадку структура процесу алгебраїчного синтезу буде відповідати рис. 4.4.

У даній структурі в блоці G_δ відсутні вхідні зв'язки, що говорить про неможливість модифікації відповідної алгебри в процесі алгебраїчного синтезу. Незмінність G_δ тягне незмінність абстрактної алгебри виходів G_λ, що виражається у відсутності зв'язку від блоку G_δ до блоку G_λ. В іншому структура процесу синтезу схожа зі структурою на рис. 4.3. Назвемо структуру, зображену на рис. 4.4, *структурою алгебраїчного синтезу МПА з ОАП з фіксованою абстрактною алгеброю переходів* і позначимо її символом M_2.

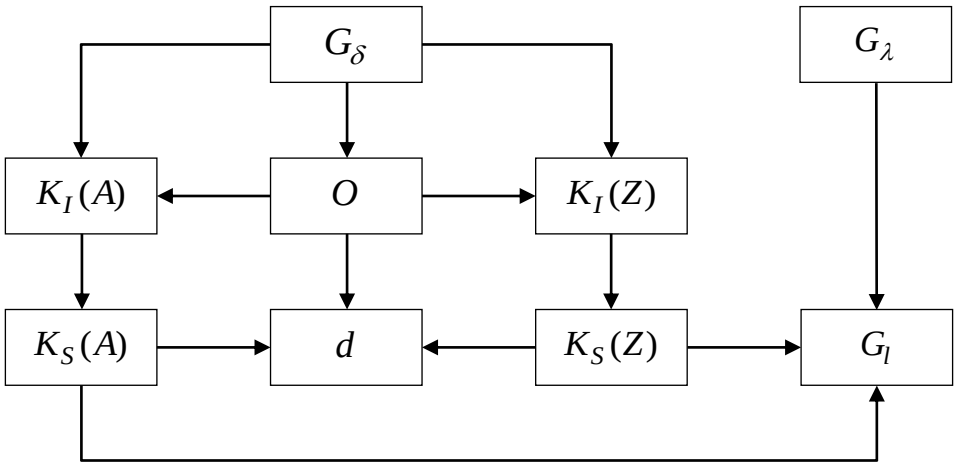


Рисунок 4.4 – Структура процесу алгебраїчного синтезу МПА з ОАП з фіксованою абстрактною алгеброю переходів (структура M_2)

Нехай до алгебраїчного синтезу МПА з ОАП висуваються наступні вимоги:

– множина O є заданою і незмінною;

– будь-які зміни алгоритму, що імплементується мікропрограмним автоматом, неприпустимі.

У цьому випадку структура процесу алгебраїчного синтезу набуває вигляд рис. 4.5. Назвемо її *структурою алгебраїчного синтезу МПА з ОАП із фіксованою абстрактною алгеброю переходів і заданою множиною операцій переходів* і позначимо символом M_3.

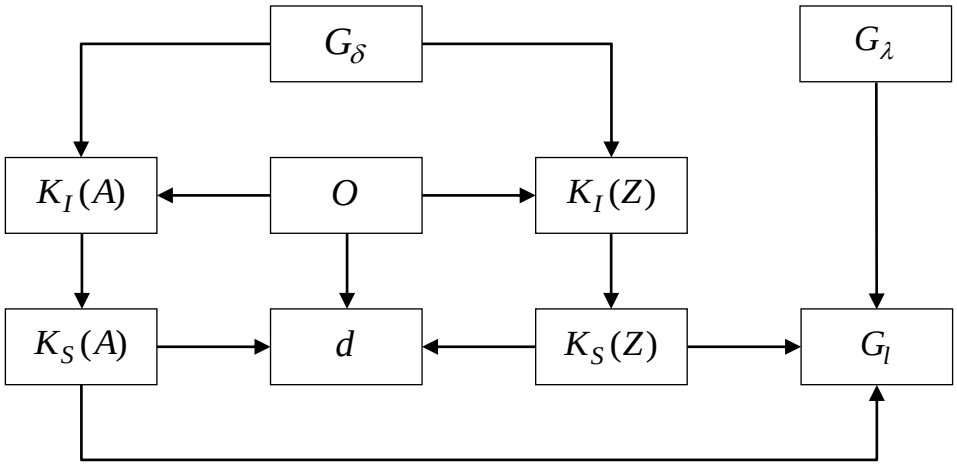


Рисунок 4.5 – Структура процесу алгебраїчного синтезу МПА з ОАП із фіксованою абстрактною алгеброю переходів та заданою множиною операцій переходів (структура M_3)

Єдиною відмінністю даної структури від структури M_2 є відсутність зв'язку від блоку G_δ до блоку O. Результатом буде відсутність у складі методів алгебраїчного синтезу, що визначаються даною структурою, етапу формування множини операцій переходів. Алгебраїчний синтез у цьому випадку буде зводитися до вибору таких проміжних і структурних кодів станів, при яких результуюча схема МПА з ОАП буде мати менші витрати апаратури в порівнянні з її реалізацією канонічним способом.

Можливість модифікації алгебри G_δ при фіксованій множині ОП породжує структуру процесу алгебраїчного синтезу, в якій, у порівнянні зі структурою M_3, додані зв'язки від блоку O до блоку G_δ та від блоку G_δ до блоку G_λ (за аналогією зі структурою M_1, рис. 4.3). Позначимо дану структуру, зображену на рис. 4.6, символом M_4 і назвемо її структурою процесу алгебраїчного синтезу МПА з

ОАП зі змінюваною абстрактною алгеброю переходів і заданою множиною операцій переходів.

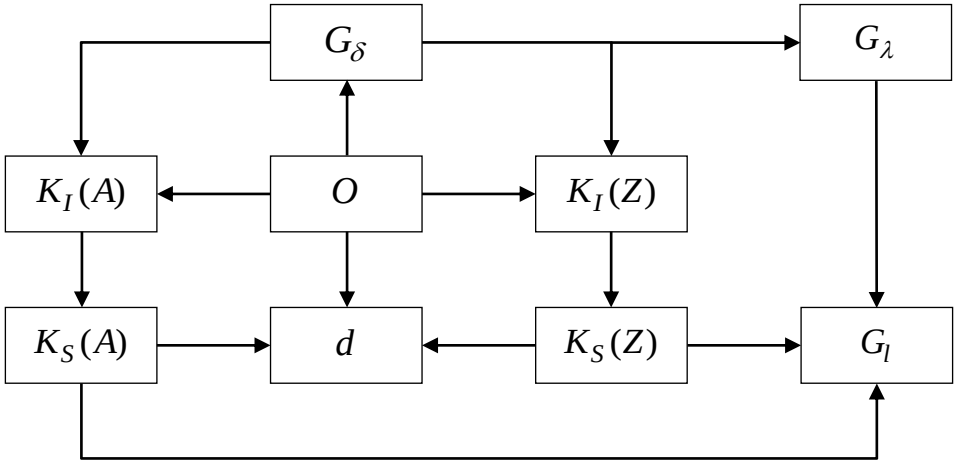


Рисунок 4.6 – Структура процесу алгебраїчного синтезу МПА з ОАП зі змінюваною абстрактною алгеброю переходів та заданою множиною операцій переходів (структура M_4)

Нехай у якості вхідних даних для процесу алгебраїчного синтезу виступають структурні коди станів та вхідних сигналів. Подібна вимога виключає можливість будь-яких модифікацій алгоритму, що приводять до зміни кількості станів і вхідних сигналів. Структура процесу алгебраїчного синтезу, що відповідає даним вимогам, наведена на рис. 4.7. Позначимо дану структуру символом M_5 і назвемо її *структурою процесу алгебраїчного синтезу із заданими множинами структурних кодів станів і вхідних сигналів*.

Пояснимо зміст двоспрямованих зв'язків між блоками $K_I(A)$ та O. Нехай задана ГСА містить перехід зі стану a_i із заданим структурним кодом $K_S(a_i) = 0101_2$ у стан a_j із заданим структурним кодом $K_S(a_j) = 1010_2$. В рамках проміжної алгебри, що реалізує перехід з a_i до a_j, для даних структурних кодів припустимі різні інтерпретації, деякі з яких наведені в табл. 4.1. В останньому стовпці таблиці показані деякі з можливих операцій, виконання яких над відповідними інтерпретаціями структурного коду $K_S(a_i)$ буде відповідати переходу зі стану a_i до стану a_j.

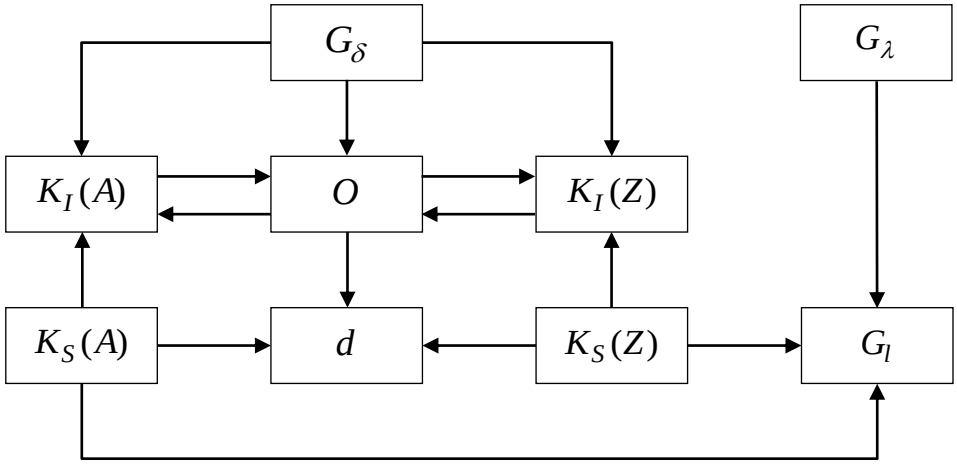


Рисунок 4.7 – Структура процесу алгебраїчного синтезу МПА з ОАП із заданими множинами структурних кодів станів і вхідних сигналів (структура M_5)

Таблиця 4.1

Способи інтерпретації структурних кодів станів $K_I(a_i)$ і $K_I(a_j)$ при $K_S(a_i) = 0101_2$, $K_S(a_j) = 1010_2$

Спосіб інтерпретації	Проміжні коди станів		Операції переходів
	$K_I(a_i)$	$K_I(a_j)$	
Цілі числа без знака	5	10	$K_I(a_i) + 5$ $K_I(a_i) * 2$
Цілі числа у прямому коді	+ 5	– 2	$K_I(a_i) - 7$ $K_I(a_i) / 2 * (-1)$
Цілі числа в доп. коді	+ 5	– 6	$(K_I(a_i) + 1) * (-1)$ $K_I(a_i) * (-1) - 1$
Цілі числа в оберненому коді	+ 5	– 5	$K_I(a_i) * (-1)$ $K_I(a_i) / 4 - 6$
Двійкові вектори	0 1 0 1	1 0 1 0	інверсія, зсув ліворуч

Наявність зв'язку від блоку $K_I(a_i)$ до блоку *O* визначає можливість вибору операцій переходів на основі значень проміжних кодів станів. При цьому спочатку вибирається спосіб інтерпретації структурних кодів станів, потім формуються значення проміжних кодів станів, після чого вибирається ОП для реалізації потрібного переходу. Наприклад, якщо для заданих вище структурних кодів $K_S(a_i)=0101_2$ та $K_S(a_j)=1010_2$ обрана інтерпретація у вигляді цілих чисел без знака, для реалізації переходу зі стану a_i в стан a_j може бути використана або операція додавання з константою 5, (схемна реалізація – суматор), або операція множення на 2 (схемна реалізація – схема зсуву).

Наявність зв'язку від блоку *O* до блоку $K_I(A)$ робить припустимим вибір способу інтерпретації та значень проміжних кодів станів на підставі заданих структурних кодів і множини операцій переходів. Наприклад, якщо множина ОП містить операцію інверсії, то для заданих вище структурних кодів $K_S(a_i)=0101_2$ та $K_S(a_j)=1010_2$ реалізація переходу з a_i до a_j за допомогою даної ОП можлива при інтерпретації $K_S(a_i)$ та $K_S(a_j)$ як двійкових векторів.

Як було відзначено вище, зі структурою, що зображена на рис. 4.7, може ототожнюватися, в загальному випадку, множина методів алгебраїчного синтезу, які у своїй алгоритмічній реалізації спираються як на зв'язок від $K_I(A)$ до *O*, так і на зв'язок від *O* до $K_I(A)$. У даних методах для реалізації переходу зі стану a_i в стан a_j може бути використаний наступний підхід.

1. Якщо в множині операцій переходів вже присутня ОП, яка підходить для реалізації переходу з a_i до a_j при відповідній їй інтерпретації структурних кодів $K_S(a_i)$ та $K_S(a_j)$, то доцільно реалізувати перехід за допомогою даної операції. Це дозволяє зберегти незмінними множину операцій переходів і, як наслідок, апаратурні витрати в схемі операційного автомата переходів. Даному прийому на рис. 4.7 відповідає зв'язок від блоку *O* до блоку $K_I(A)$.

2. Якщо жодна операція із множини ОП не підходить для реалізації переходу з a_i до a_j при відповідній їй інтерпретації структурних кодів цих станів, у множину ОП слід додати нову операцію. ОП, що додається, повинна відповідати наступним критеріям:

– можливість використання для реалізації переходу з a_i до a_j при відповідній інтерпретації структурних кодів цих станів (зв'язок від блоку $K_I(A)$ до блоку *O* на рис. 4.7);

– можливість використання для реалізації якомога більшого числа переходів у рамках інтерпретованої ГСА при заданій множині $K_S(A)$;

– апаратурні витрати в комбінаційній схемі, що відповідає даній ОП, повинні бути менші за витрати у випадку канонічної реалізації множини переходів, що реалізуються даною ОП у рамках імплементованої ГСА.

Як видно з табл. 4.1, для реалізації переходу з a_i в a_j у загальному випадку може бути використано кілька різних операцій переходів. При цьому доцільно вибирати таку ОП, використання якої дає максимальний виграш в апаратурних витратах у порівнянні з реалізацією підмножини переходів, що реалізуються даною ОП, в канонічний спосіб.

Способи структурного представлення процесу алгебраїчного синтезу МПА з ОАП не обмежуються структурами, наведеними на рис. 4.3–4.7. Якщо якийсь метод алгебраїчного синтезу не може бути класифікований однією з розглянутих структур, завжди є можливість побудови нової структури, що відбиває причинно-наслідкові зв'язки між елементами системи ізоморфізмів (3.9), характерні для даного методу.

Запропонований метод представлення процесу алгебраїчного синтезу МПА з ОАП у вигляді структури переслідує наступні цілі:

1. Графічне представлення розподілу елементів системи ізоморфізмів (3.9) за алгебраїчними рівнями.

2. Визначення відносної послідовності формування елементів системи (3.9).

3. Використання у якості узагальнюючої моделі для певної множини методів алгебраїчного синтезу МПА з ОАП.

4.2.4 Алгебраїчний синтез методом повного перебору

Розглянемо МПА з ОАП зі структурою U_1 (рис. 3.4). Характерною рисою даної структури є можливість зіставлення операції переходів окремому переходу автомата.

Позначимо через K_S^R множину усіх структурних (двійкових) кодів розрядності R, через O – множину операцій переходів. Зіставимо кожному стану автомата унікальний структурний код із множини K_S^R, а кожному переходу – операцію переходу із множини O. Якщо зіставлення зроблене відповідно до принципу операційного перетворення кодів станів, викладеному в п. 2.4, то результатом є формальний розв'язок системи ізоморфізмів (3.9). А якщо ні, то зіставлення не приводить до формального розв'язку системи (3.9) і є «даремним».

Нехай M – число станів автомата, B – число переходів, N_I – кількість операцій переходів. Якщо множини K_S^R та O кінцеві, то кінцеве і число різних способів зіставлення елементів даних множин. Будемо виходити з того, що у випадку повного перебору зіставлення структурних кодів станам і зіставлення ОП автоматним переходам виконуються незалежно одне від іншого [16, 21, 22]. Тоді кількість N_1 способів зіставлення структурних кодів станам автомата визначається як число розміщень із 2^R елементів по M елементів:

$$N_1 = \mathrm{A}_{2^R}^{M} = \frac{(2^R)!}{(2^R - M)!}. \qquad (4.1)$$

Визначимо кількість способів зіставлення N_I операцій переходів B переходам автомата:

$$N_2 = (N_I + 1)^B. \qquad (4.2)$$

Одиниця, що додається тут до N_I, враховує можливість того, що будь-якому переходу автомата може бути не зіставлена ніяка ОП, внаслідок чого даний перехід реалізується канонічним способом.

Тоді кількість варіантів взаємонезалежного зіставлення структурних кодів станів і операцій переходів буде дорівнювати:

$$N = N_1 \cdot N_2 = \frac{(2^R)!}{(2^R - M)!} \cdot (N_I + 1)^B. \qquad (4.3)$$

Після того, як обраний один з варіантів зіставлення, слід перевірити – чи є він «даремним» або дає формальний розв'язок задачі алгебраїчного синтезу. Позначимо через t_c час, що витрачається на вибір чергового варіанту зіставлення і перевірку на одержання формального розв'язку. Тоді час *t*, необхідний для одержання множини формальних розв'язків методом повного перебору, визначається виразом (4.4).

$$t = N \cdot t_c = \frac{(2^R)!}{(2^R - M)!} \cdot (N_I + 1)^B \cdot t_c. \qquad (4.4)$$

Нехай $t_c = 0{,}001$ секунди. Тоді час, що витрачається на повний перебір варіантів у випадку ГСА середньої складності ($M = 50$, $R = 6$, $B = 100$ [29]) і $(N_I + 1) = 10$, становить $1{,}45 \cdot 10^{175}$ *с*, що, очевидно, є неприйнятним.

У випадку, наприклад, $M = 10$, $R = 4$, $B = 20$, $(N_I + 1) = 3$ та $t_c = 0{,}001$ *с*, значення $t = 10^{17}$ *с*. При зменшенні значення t_c до 1 наносекунди значення *t* дорівнюватиме 10^{11} *с*, що становить 3200 років і також є неприйнятним.

Вирази (4.1) – (4.4) справедливі також і для структури МПА U_3 (рис. 3.9). На відміну від структур U_1 та U_3, структури U_2 (рис. 3.6) та U_4 (рис. 3.12) дозволяють зіставляти операції переходів не окремим переходам, а станам автомата. При цьому вирази (4.2) – (4.4) приймуть вигляд (4.5) – (4.7) відповідно.

$$N_2 = (N_I + 1)^M. \qquad (4.5)$$

$$N = N_1 \cdot N_2 = \frac{(2^R)!}{(2^R - M)!} \cdot (N_I + 1)^M. \qquad (4.6)$$

$$t = N \cdot t_c = \frac{(2^R)!}{(2^R - M)!} \cdot (N_I + 1)^M \cdot t_c . \qquad (4.7)$$

У випадку ГСА середньої складності ($M = 50$, $R = 6$), $(N_I + 1) = 10$ і $t_c = 0{,}001\ c$ час, що витрачається на повний перебір варіантів, складе близько $1{,}45 \cdot 10^{125}\ c$, що також не дозволяє розв'язати задачу методом повного перебору. При $M = 10$, $R = 4$, $(N_I + 1) = 3$ та $t_c = 0{,}001\ c$ одержуємо $t = 1{,}7 \cdot 10^{12}\ c$. При зменшенні значення t_c до 1 наносекунди величина *t* складе близько $1{,}7 \cdot 10^6\ c$ або близько 20 діб. Таке значення, з одного боку, вимагає для досягнення величини $t_c = 1\ нс$ значних обчислювальних ресурсів і ефективної алгоритмічної реалізації, а з іншого сторони є нераціональним для синтезу автоматів настільки малої складності.

Наведені розрахунки дозволяють зробити висновок про практичну неможливість вирішення задачі алгебраїчного синтезу МПА з ОАП методом повного перебору. При цьому не виключена можливість вирішення даної задачі методом часткового перебору варіантів, однак це вимагає розробки спеціальних методів і алгоритмів.

4.2.5 Використання транзитних станів

Нехай у процесі алгебраїчного синтезу МПА з ОАП необхідно реалізувати перехід зі стану a_m до стану a_n, причому до моменту його реалізації структурні коди $K_S(a_m)$ та $K_S(a_n)$ відомі (наприклад, внаслідок попередньої реалізації інших переходів за участю даних станів). Припустимо також, що жодна операція із множини операцій переходів не підходить для реалізації переходу з a_m у a_n при заданих значеннях $K_S(a_m)$ та $K_S(a_n)$.

У цьому випадку для вирішення задачі реалізації переходу з a_m до a_n є два найбільш очевидні способи:

1. Реалізація переходу за допомогою додаткової ОП.

2. Реалізація переходу канонічним способом.

Перший спосіб приводить до додавання в схему ОЧ додаткової комбінаційної схеми і підключення її виходу до мультиплексора результату. Недоліком тут є збільшення апаратурних витрат у схемі ОАП, перевагою – можливість використання ОП, що додається, для реалізації інших переходів автомата.

Другий спосіб приводить до модифікації комбінаційної схеми, яка відповідає за реалізацію частини переходів автомата канонічним способом. Недоліком є збільшення апаратурних витрат у даній схемі і необхідність її повторного синтезу, перевагою – відсутність змін у мультиплексорі результату.

В дисертаційній роботі пропонується метод вирішення даної задачі, який дозволяє, у деяких випадках, реалізувати такий перехід за допомогою наявних операцій переходів [16, 26]. Метод полягає в наступному:

1. Додамо в задану ГСА, що містить M станів $a_0 - a_{M-1}$, послідовність із k операторних вершин, у яких не формуються ніякі мікрооперації. Послідовність включимо в розрив гілки, яка відповідає переходу зі стану a_m до стану a_n, і відзначимо вершини послідовності станами a_M, ..., a_{M+k-1}, де $k \geq 1$.

2. Закодуємо стани a_M, ..., a_{M+k-1} структурними кодами $K_S(a_M)$, ..., $K_S(a_{M+k-1})$ розрядності $R = \lceil \log_2 M \rceil$ таким чином, щоб:

– коди $K_S(a_M)$, ..., $K_S(a_{M+k-1})$ були унікальні й вільні (не використовувалися для кодування інших станів заданої ГСА);

– переходи $a_m \to a_M$, $a_M \to a_{M+1}$, ..., $a_{M+k-2} \to a_{M+k-1}$, $a_{M+k-1} \to a_n$ могли бути реалізовані за допомогою операцій переходів із заданої множини ОП.

Назвемо стани, що додаються у вихідний автомат з метою реалізації переходів автомата за допомогою наявної множини операцій переходів, *транзитними станами*. Сенс даного терміну визначається допоміжною роллю цих станів у процесі алгебраїчного синтезу автомата.

Нехай, наприклад, $K_S(a_m) = 0101_2$, $K_S(a_n) = 1111_2$. Обмежимо множину операцій переходів наступними ОП:

$$O_1 : K_{I_1}(a^{t+1}) = (K_{I_1}(a^t) + 13) \bmod 16, \qquad (4.8)$$

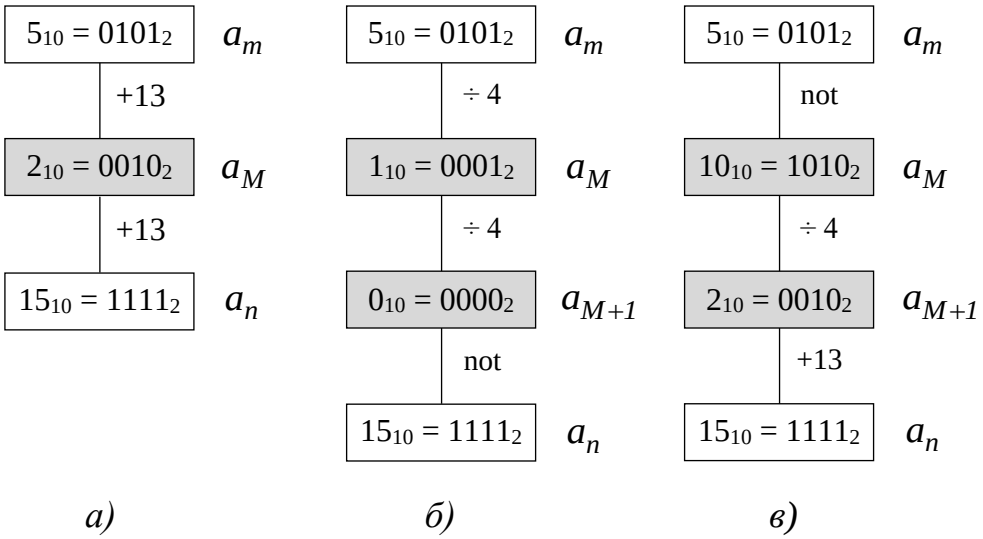
$$O_2 : K_{I_2}(a^{t+1}) = K_{I_2}(a^t) \div 4, \tag{4.9}$$

$$O_3 : K_{I_3}(a^{t+1}) = \overline{K_{I_3}(a^t)}, \tag{4.10}$$

де a^t – поточний стан автомата, a^{t+1} – стан переходу.

Для операцій O_1, O_2 будемо інтерпретувати структурні коди станів як беззнакові цілі в діапазоні $[0;15]$: $K_{I_1}(a_m) = K_{I_2}(a_m) = 5_{10}$, $K_{I_1}(a_n) = K_{I_2}(a_n) = 15_{10}$. Для операції O_3 будемо розглядати структурні коди станів як чотирирозрядніе двійкові вектори, що збігаються з відповідними структурними кодами: $K_{I_3}(a_m) = K_S(a_m) = 0101_2$, $K_{I_3}(a_n) = K_S(a_n) = 1111_2$.

За даних умов реалізація переходу зі стану a_m в стан a_n за допомогою транзитних станів можлива різними способами, деякі з яких показані на рис. 4.8. Тут в операторних вершинах, позначених станами автомата Мура, записані відповідні до станів проміжні коди $K_{I_1}(a_i)$ та $K_{I_2}(a_i)$, причому код $K_{I_2}(a_i)$ збігається з відповідним структурним кодом. Ребра позначені операціями переходів, що виконуються над одним із проміжних кодів поточного стану: позначення «+13» відповідає операції O_1, «÷4» – операції O_2, «not» – операції O_3. Додані вершини позначені затемненим фоном.



а) *б)* *в)*

Рисунок 4.8 – Приклади реалізації автоматного переходу із використанням додаткових станів

На рис. 4.8, *а* для переходу зі стану a_m в стан a_n використаний один транзитний стан a_M. Обраний для нього структурний код $K_S(a_M) = 0010_2$ дозволяє реалізувати переходи $a_m \to a_M$ та $a_M \to a_n$ за допомогою однієї і тієї ж операції O_1. Відзначимо, що дія «mod 16» у виразі (4.8) на рівні логічної схеми реалізується шляхом відкидання переносу зі старшого розряду суми.

На рис. 4.8, *б* та *в* для переходу з a_m до a_n використані два транзитні стани. Їхнє виконання займає два такти роботи схеми автомата замість одного такту у випадку рис. 4.8, *а*. Таким чином, при реалізації переходу за допомогою транзитних станів слід прагнути до зменшення їх кількості в послідовності, що додається.

У деяких випадках використання транзитних станів дозволяє добитися реалізації всіх переходів автомата за допомогою заданої множини операцій переходів. Нехай МПА заданий ГСА Γ_{4-1} (рис. 4.9). Оцінка ГСА станами автомата Мура дозволяє виділити $M = 9$ станів $a_0 - a_8$, для яких $R = 4$. Нехай множина ОП утворена операціями $O_1 - O_3$, що задаються виразами (4.8) – (4.10) відповідно.

При використанні структури МПА з ОАП U_1, що дозволяє зіставляти операції переходів окремим переходам автомата, результат алгебраїчного синтезу з використанням транзитних станів може відповідати рис. 4.10. Позначення на рис. 4.10 зроблені за аналогією з рис. 4.9.

Можна помітити, що в деяких випадках при обраних структурних кодах станів один і той самий перехід може бути реалізований різними ОП. Наприклад, у ГСА Γ_{4-2} та Γ_{4-3} перехід $a_6 \to a_8$ може бути реалізований як операцією O_1, так і операцією O_3. Також у ГСА Γ_{4-3} перехід $a_3 \to a_4$ може бути реалізований операцією O_2 або операцією O_3.

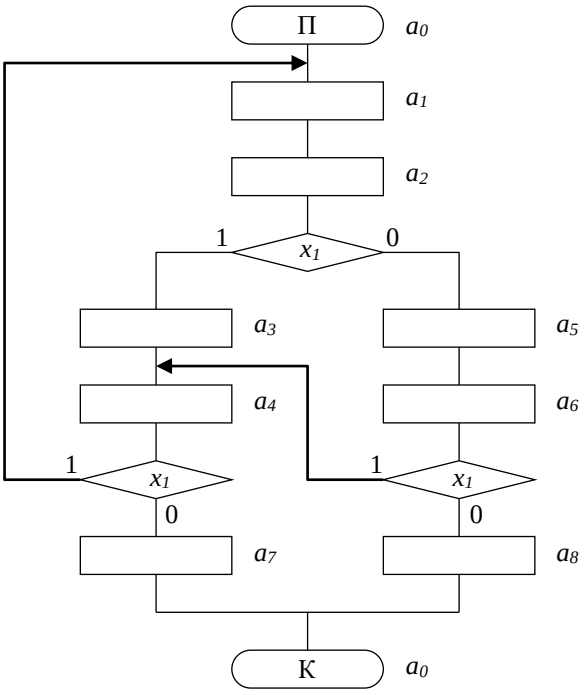
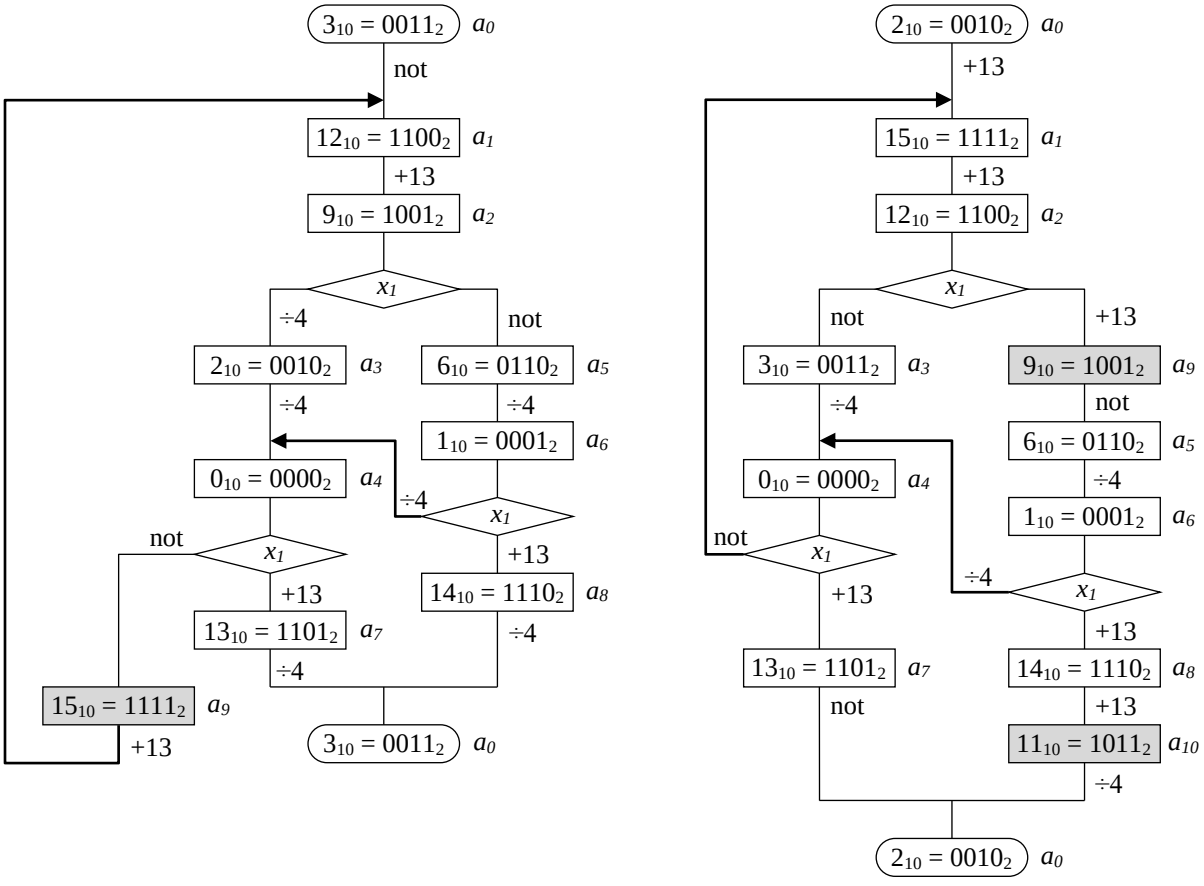


Рисунок 4.9 – Граф-схема алгоритму Γ_{4-1}



а) *б)*

Рисунок 4.10 – Результати алгебраїчного синтезу МПА з ОАП зі структурою U_1 по ГСА Γ_{4-1}: ГСА Γ_{4-2} с одним транзитним станом (*а*), ГСА Γ_{4-3} з двома транзитними станами (*б*)

Вибір тієї або іншої ОП зазвичай не має принципового значення, оскільки апаратурні витрати в комбінаційних схемах, що відповідають ОП $O_1 - O_3$, не залежать від кількості переходів, реалізованих за допомогою даної ОП. Вибір ОП для реалізації конкретних переходів враховується лише при синтезі Z-підсхеми.

Приклад на рис. 4.10, *б* показує, що в одній ГСА припустиме використання декількох послідовностей транзитних станів, довжини яких у загальному випадку можуть бути різними. Пошук найкращого варіанта використання транзитних станів вимагає застосування спеціальних методів, розробка яких виходить за рамки даної роботи.

Використання транзитних станів у процесі алгебраїчного синтезу МПА з ОАП можливе за умови припустимості перетворення заданої ГСА і приводить, у загальному випадку, до наступних результатів:

1. Середній час виконання алгоритму, що імплементується МПА, збільшується за рахунок тактів, що витрачаються на переходи в транзитні стани.

2. Які-небудь зміни в логічній схемі ОАП відсутні, оскільки не змінюються ані розрядність структурного коду стану, ані множина операцій переходів.

3. Z-підсхема модифікується таким чином, щоб формувати потрібні коди ОП у межах послідовностей транзитних станів.

4. Схема формування мікрооперацій модифікується таким чином, щоб при знаходженні автомата в транзитних станах виключити формування будь-яких мікрооперацій. При цьому знаходження МПА в транзитних станах не повинне позначатися на функціонуванні об'єкта, який керується МПА.

4.2.6 Врахування імовірностей істинності логічних умов

У деяких випадках для кожного вхідного сигналу $z_i \in Z,\ i \in [1;\ F]$, відомі ймовірності $p(z_i)$ появи цього сигналу на вході автомата в довільний момент часу. При завданні автомата граф-схемою алгоритму вхідні сигнали кодуються векторами, що утворені значеннями структурних змінних логічних умов $x_1, ..., x_L$. У цьому випадку ймовірності появи вхідних сигналів можуть бути

визначені, виходячи з імовірностей $p(x_i)$ істинності сигналів ЛУ в кожний момент часу.

Нехай для всіх логічних умов $x_1, ..., x_L$, що присутні в заданій ГСА, відомі ймовірності $p(x_1), ..., p(x_L)$ істинності відповідних ЛУ в будь-який момент часу роботи автомата. Це дає можливість визначити для кожної операторної вершини середню кількість тактів, що затрачається на її виконання за один прохід ГСА. Оскільки в МПА стани пов'язані з операторними вершинами або їх виходами, середня кількість тактів, що витрачається на виконання операторних вершини, буде відповідати середній кількості переходів автомата у відповідний стан за один прохід алгоритму [16, 23].

Позначимо через $q(a_i)$ середню кількість переходів у стан a_i, що здійснюються за один прохід алгоритму. Тоді середня кількість тактів Q, що витрачається на виконання мікропрограми, визначається сумою значень q усіх станів заданої ГСА:

$$Q = \sum_{i=0}^{M-1} q(a_i). \qquad (4.11)$$

Для визначення Q при відомих значеннях $p(x_1), ..., p(x_L)$ можна скористатися методикою, що наведена у [125]. Її застосування до ГСА Γ_{4-1} (рис. 4.9), що містить єдину ЛУ x_1, дозволяє визначити величини q для різних значень імовірності $p(x_1)$. Результати відповідних розрахунків наведені в табл. 4.2.

Розглянемо ГСА Γ_{4-2} (рис. 4.10, *а*). У ній транзитний стан a_9 доданий після стану a_4, причому перехід $a_4 \to a_9$ є для a_9 єдиним вхідним переходом. Це дозволяє визначити величину $q(a_9)$ наступним виразом:

$$q(a_9) = q(a_4) \cdot p(x_1). \qquad (4.12)$$

Дане значення відповідає збільшенню ΔQ_{10-1} величини Q при використанні ГСА Γ_{4-2} замість ГСА Γ_{4-1}.

Таблиця 4.2

Значення $q(a_i)$ для різних імовірностей $p(x_1)$

$p(x_1)$	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
$q(a_0)$	1	1	1	1	1	1	1	1	1
$q(a_1)$	1,02	1,08	1,18	1,34	1,60	2,02	2,75	4,31	9,17
$q(a_2)$	1,02	1,08	1,18	1,34	1,60	2,02	2,75	4,31	9,17
$q(a_3)$	0,1	0,22	0,35	0,54	0,80	1,21	1,93	3,45	8,26
$q(a_4)$	0,19	0,39	0,6	0,86	1,20	1,69	2,51	4,14	9,08
$q(a_5)$	0,92	0,86	0,83	0,81	0,80	0,81	0,83	0,86	0,92
$q(a_6)$	0,92	0,86	0,83	0,81	0,80	0,81	0,83	0,86	0,92
$q(a_7)$	0,17	0,31	0,42	0,52	0,60	0,68	0,75	0,83	0,91
$q(a_8)$	0,83	0,69	0,58	0,48	0,40	0,32	0,25	0,17	0,09
Q	6,17	6,49	6,97	7,7	8,8	10,56	13,6	19,93	39,52

За аналогією з (4.12) сформуємо вирази (4.13) та (4.14) для транзитних станів ГСА Γ_{4-3}.

$$q(a_9) = q(a_2) \cdot (1 - p(x_1)). \tag{4.13}$$

$$q(a_{10}) = q(a_8) \cdot 1 = q(a_8). \tag{4.14}$$

При цьому

$$\Delta Q_{10-2} = q(a_9) + q(a_{10}). \tag{4.15}$$

Порівняємо величини ΔQ_{4-2} та ΔQ_{4-3} при різних значеннях імовірності $p(x_1)$, для чого складемо табл.4.3. Розділивши ΔQ_{4-2} та ΔQ_{4-3} на відповідні значення Q з табл. 4.2, одержимо відносні значення ΔQ_{4-2} та ΔQ_{4-3}, представлені в табл. 4.4 у процентному вимірі.

Аналіз табл. 4.2 – 4.4 дозволяє зробити наступні висновки.

1. Вплив імовірностей істинності логічних умов на величину середньої кількості тактів виконання мікропрограми залежить від структури ГСА. Застосування відомих методів дозволяє визначити величину Q до та після

додавання транзитних станів, що дає можливість оцінити доцільність додавання транзитних станів з погляду збільшення часу виконання алгоритму.

Таблиця 4.3

Значення ΔQ_{4-2} та ΔQ_{4-3} при різних імовірностях $p(x_1)$

$p(x_1)$		0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
Γ_{4-2}	$q(a_9)$	0,02	0,08	0,18	0,34	0,6	1,01	1,76	3,31	8,17
	ΔQ_{4-2}	**0,02**	**0,08**	**0,18**	**0,34**	**0,6**	**1,01**	**1,76**	**3,31**	**8,17**
Γ_{4-3}	$q(a_9)$	0,92	0,86	0,83	0,81	0,80	0,81	0,83	0,86	0,92
	$q(a_{10})$	0,83	0,69	0,58	0,48	0,40	0,32	0,25	0,17	0,09
	ΔQ_{4-3}	**1,75**	**1,55**	**1,41**	**1,29**	**1,2**	**1,13**	**1,08**	**1,03**	**1,01**

Таблиця 4.4

Відносні значення ΔQ_{4-2} та ΔQ_{4-3}

$p(x_1)$	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
ΔQ_{4-2}, %	0,3	1,2	2,6	4,4	6,8	9,6	12,9	16,6	20,7
ΔQ_{4-3}, %	28,3	23,9	20,2	16,8	13,6	10,7	7,9	5,1	2,6

2. На прикладі ГСА Γ_{4-2} і Γ_{4-3} можна бачити, що менша кількість використовуваних транзитних станів у випадку Γ_{4-2} в порівнянні з Γ_{4-3} не визначає однозначно більш високу ефективність за часом виконання алгоритму. Структура ГСА Γ_{4-1} така, що при малих значеннях $p(x_1)$ величина ΔQ_{4-3} перевищує величину ΔQ_{4-2}, у той час як при $p(x_1) > 0{,}6$ ситуація змінюється на протилежну.

Таким чином, розглянутий метод врахування імовірностей істинності логічних умов заданої ГСА дозволяє вибрати більш ефективний варіант розміщення транзитних станів, а також оцінити збільшення середнього часу виконання алгоритму і доцільність використання транзитних станів у кожному конкретному випадку.

4.2.7 Збільшення розрядності структурних кодів станів

Якщо синтезований автомат містить M станів, мінімально достатня розрядність структурного коду стану $R_{\min} = \lceil \log_2 M \rceil$. У МПА з канонічною структурою використання для кодування станів структурних кодів розрядності $R > R_{\min}$ не має сенсу, оскільки приводить лише до збільшення апаратурних витрат у схемі автомата.

У випадку МПА з ОАП збільшення значення R значно впливає на величину N_1 у виразі (4.1). Так, для ГСА середньої складності ($M = 50$, $R = 6$, [29]) збільшення R на одиницю збільшує N_1 у $2{,}3 \cdot 10^{22}$ разів. У стільки ж разів зростає число варіантів повного перебору, яке для структур МПА U_1 та U_3 визначається виразом (4.4), для структур U_2 та U_4 – виразом (4.6).

У той же час збільшення варіантів повного перебору може приводити до збільшення кількості формальних розв'язків задачі алгебраїчного синтезу МПА з ОАП, а отже – до збільшення кількості ефективних і оптимальних розв'язків. Якщо при $R = R_{\min}$ обраний метод алгебраїчного синтезу не дозволяє знайти жодного ефективного розв'язку, не виключено, що такий розв'язок може бути знайдене при більшому значенні R [16, 22].

Розглянемо приклад. Нехай автомат заданий ГСА Γ_{4-4}, що складається з початкової вершини, трьох послідовних операторних вершин та кінцевої вершини (рис. 4.11). Дана ГСА може бути позначена станами автомата Мілі $a_0 – a_2$ або станами автомата Мура $b_0 – b_3$.

Нехай обмеженням алгебраїчного синтезу є використання єдиної операції переходів, яка задається виразом (4.16). Це припускає проміжну інтерпретацію структурних кодів станів як цілих чисел без знака.

$$O_1: \quad K_{I_1}(a^{t+1}) = (K_{I_1}(a^t) + 2) \bmod 4. \qquad (4.16)$$

Використання трьох станів у випадку автомата Мілі та чотирьох станів у випадку автомата Мура визначає значення $R_{\min} = 2$. Отже, для кодування станів

можуть бути використані дворазрядні структурні коди $00_2, 01_2, 10_2, 11_2$, що мають цілочисельну інтерпретацію $0_{10}, 1_{10}, 2_{10}, 3_{10}$ відповідно.

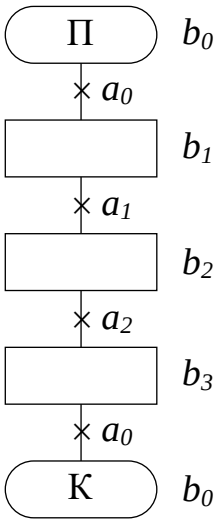


Рисунок 4.11 – Граф-схема алгоритму Γ_{4-4}

Для того, щоб ОП O_1 могла реалізовувати три (для автомата Мілі) або чотири (для автомата Мура) послідовних переходи, вона повинна дозволяти задавати на множині припустимих кодів хоча б одне транзитивне замикання, що складається із трьох або чотирьох елементів відповідно. На множині кодів $\{0, 1, 2, 3\}$ операція O_1 дозволяє задати чотири двоелементні транзитивні замикання: $<0, 2>$, $<1, 3>$, $<2, 0>$, $<3, 1>$. При цьому задати трьохелементне або чотириелементне замикання виявляється неможливим. Таким чином, при будь-якому способі зіставлення структурних кодів станам автомата неможливо реалізувати всі переходи ГСА Γ_{11} за допомогою заданої ОП O_1.

Збільшимо R до величини $R_{\min}+1=3$. В результаті одержимо можливість кодування станів структурними кодами із множини $K_S^R=\{000, 001, ..., 111\}$, які інтерпретуються в цілочисельному діапазоні $[0; 7]$. На даній множині операція O_1 дозволяє задати вісім транзитивних замикань:

$<0, 2, 4, 6>, <2, 4, 6, 0>, <4, 6, 0, 2>, <6, 0, 2, 4>,$

$<1, 3, 5, 7>, <3, 5, 7, 1>, <3, 5, 7, 1>, <7, 3, 1, 5>.$

При використанні кожного з цих кортежів для завдання проміжних кодів станів автомата (Мілі або Мура), заданого ГСА Γ_{4-4}, усі переходи можуть бути реалізовані за допомогою лише ОП O_1.

Розглянутий приклад показує, що збільшення розрядності структурного коду стану збільшує область визначення і область значень операцій переходів. У відношенні окремо взятої ОП це дає потенційну можливість використовувати її більшу кількість разів при реалізації автоматних переходів. У відношенні всієї множини ОП збільшення розрядності структурних кодів станів дозволяє збільшити кількість переходів, що реалізуються за допомогою наявних ОП, зменшивши тим самим кількість переходів, що реалізуються в канонічний спосіб.

Відзначимо також наступне. Якщо обраний метод алгебраїчного синтезу припускає використання транзитних станів, збільшення розрядності структурного коду стану дає можливість підвищити ефективність даного підходу.

Нехай заданий автомат містить $M = 16$ станів. Використання структурних кодів розрядності $R = 4$ приводить до відсутності «вільних» кодів, які могли б бути зіставлені транзитним станам, і отже – до неможливості використання транзитних станів у даному автоматі. Якщо ж автомат містить $M = 15$ станів, то при $R = 4$ припустиме використання лише одного транзитного стану. Це дає можливість реалізувати «транзитним» способом тільки один автоматний перехід, а послідовність, що додається, буде містити лише один стан.

Збільшення розрядності R на одиницю при $M = 16$ приводить до подвоєння множини K_S^R, дозволяючи використовувати до 16 транзитних станів. В результаті виявляється можливим формування декількох послідовностей транзитних станів різної довжини і реалізація «транзитним» способом кількох автоматних переходів.

Безумовно, збільшення R приводить до збільшення апаратурних витрат у всіх структурних блоках МПА з ОАП. Однак зменшення кількості «канонічних» переходів, що досягається при цьому, може давати компенсуючий ефект, що виражається у зменшенні апаратурних витрат в комбінаційній схемі, що реалізує

підмножину переходів автомата канонічним способом. Оскільки ідея операційного перетворення кодів станів полягає у відході від канонічної реалізації автоматних переходів, метод штучного збільшення величини R може сприяти зменшенню апаратурних витрат у логічній схемі МПА з ОАП і використовуватися в різних методах алгебраїчного синтезу даного класу автоматів.

4.2.8 Метод алгебраїчного синтезу мікропрограмного автомата з операційним автоматом переходів

У п. 4.2.4 розглянута можливість пошуку розв'язків задачі алгебраїчного синтезу МПА з ОАП шляхом повного перебору варіантів зіставлення станам автомата структурних кодів при деякому зіставленні переходам або станам автомата операцій переходів. При цьому була відзначена недоцільність застосування даного підходу навіть до автоматів малої складності.

У даній дисертаційній роботі пропонується метод алгебраїчного синтезу МПА з ОАП, що враховує підходи, викладені в п. 4.2.4 – 4.2.7 [11, 16]. Метод має наступні особливості:

– пошук розв'язків виконується шляхом часткового перебору;

– допускається використання транзитних станів;

– враховується середня кількість переходів у стани автомата за один прохід алгоритму;

– множина структурних кодів станів є підмножиною множини K_S^R, де $\left|K_S^R\right| = 2^R \geq M$;

– множина операцій переходів є фіксованою і не може змінюватися в процесі алгебраїчного синтезу.

Перераховані особливості дозволяють співвіднести розроблений метод зі структурою процесу алгебраїчного синтезу M_4 (рис. 4.6). При цьому абстрактний автомат може задаватися як таблицями переходів і виходів, так і граф-схемою алгоритму, що позначена станами автоматів Мілі або Мура.

Метод включає наступні основні кроки:

1. Вибір розрядності структурних кодів станів і завдання множини K_S^R.

2. Завдання множини операцій переходів.

3. Визначення для кожного стану величини q.

4. Вибір стану $a(q_{\max})$ із максимальним значенням q.

5. Прийняття множини формальних розв'язків задачі алгебраїчного синтезу МПА з ОАП рівній порожній.

6. Перебір (у довільному порядку) усіх структурних кодів із множини K_S^R.

7. Кодування стану $a(q_{\max})$ черговим обраним структурним кодом.

8. Спроба реалізації всіх переходів зі стану $a(q_{\max})$ за допомогою операцій переходів із заданої множини ОП з можливістю використання транзитних станів. Якщо реалізація можлива, здійснюється перехід до кроку 9, інакше – до кроку 11.

9. Реалізація інших переходів автомата з урахуванням результатів, отриманих у п. 8, і можливістю використання транзитних станів.

10. Представлення результатів кроків 7–9 у вигляді чергового формального розв'язку задачі алгебраїчного синтезу і додавання його до множини формальних розв'язків.

11. Якщо перебрані не всі елементи множини K_S^R, здійснюється перехід до кроку 7, інакше – до кроку 12.

12. Аналіз сформованої множини формальних розв'язків задачі алгебраїчного синтезу МПА з ОАП.

Пояснимо дані кроки.

Крок 1. Як було показано в п. 4.2.7, розрядність R структурних кодів станів має бути більшою або дорівнювати мінімально достатній розрядності $R_{\min} = \lceil \log_2 M \rceil$, де M – кількість станів заданого автомата. Множина K_S^R припустимих структурних кодів станів завжди включає 2^R двійкових векторів розрядності R.

Зазвичай не можна в кожному конкретному випадку заздалегідь визначити оптимальне значення R. Оскільки саме по собі збільшення R приводить до

неминучого збільшення апаратурних витрат у всіх блоках схеми МПА з ОАП, кращим є наступний підхід: спочатку алгебраїчний синтез виконується для $R = R_{\min}$, потім для $R = R_{\min} + 1$ і так далі. При цьому слід пам'ятати, що збільшення множини K_S^R веде, як правило, до збільшення витрат часу на пошук варіантів кодування.

Крок 2. При формуванні множини операцій переходів до початку алгебраїчного синтезу звичайно слід виходити з вимог до проектування логічної схеми МПА з ОАП. Такими вимогами можуть бути: доступний елементний базис, мінімальна швидкодія, енергоспоживання тощо. Не слід керуватися критерієм універсальності схеми ОАП через неуніверсальність схеми МПА, частиною якої вона є. На підставі множини K_S^R, заданої на кроці 1, для кожної ОП вибирається спосіб інтерпретації структурних кодів станів і задаються область визначення і область припустимих значень, об'єднання яких дає множину проміжних кодів станів даної ОП.

Крок 3. Дії, що виконуються на даному кроці, викладені в п. 4.2.6. Відзначимо, що визначення для станів автомата значень q зводиться до розв'язку системи лінійних алгебраїчних рівнянь порядку M, де M – число станів автомата. На сучасному персональному комп'ютері процес розв'язку системи для $M = 1000$ займає менше секунди і не є джерелом значних витрат часу.

Крок 4. Якщо станів з максимальним значенням q декілька, має бути обраний один з них. Оскільки вибір стану кардинальним образом впливає на кінцевий результат, можна запропонувати три підходи:

– вибір довільного стану;

– вибір стану за якими-небудь критеріями;

– послідовний перебір станів з максимальним значенням q.

Крок 5. Розглянутий метод алгебраїчного синтезу дозволяє одержати для заданого автомата кілька формальних розв'язків задачі алгебраїчного синтезу, множина яких на даному кроці оголошується порожньою, може поповнюватися на кроці 10 і аналізується на кроці 12.

Крок 6. В основі методу, що пропонується, лежить послідовне використання всіх кодів із множини K_S^R для кодування стану $a(q_{\max})$. Оскільки формальний розв'язок задачі алгебраїчного синтезу може бути отриманий при кожному способі кодування, максимально можлива кількість формальних розв'язків дорівнює $\left|K_S^R\right| = 2^R$. Ця величина визначає кількість ітерацій основного циклу, тіло якого утворене кроками 7-9.

Крок 7. Зіставлення стану $a(q_{\max})$ чергового структурного коду є «відправною крапкою», що визначає результати виконання кроків 8 та 9.

Крок 8. На даному кроці вживається спроба реалізувати всі переходи зі стану $a(q_{\max})$ за допомогою операцій із множини *O*. Дана спроба викликана прагненням реалізувати якомога більшу кількість переходів автомата за допомогою ОП. Слід, однак, враховувати, що якщо при будь-якому зіставленні наявних ОП переходам зі стану $a(q_{\max})$ виникає порушення унікальності структурних кодів станів, реалізація всіх переходів із $a(q_{\max})$ за допомогою ОП із заданої множини *O* неможлива.

Як приклад розглянемо фрагмент ГСА Γ_{4-5}, що позначена станами автомата Мілі $a_1 - a_4$ (рис. 4.12).

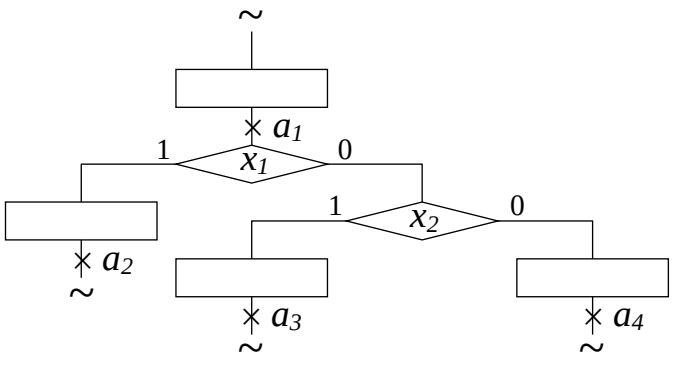


Рисунок 4.12 – Фрагмент ГСА Γ_{4-5}

Для кодування станів умовимося використовувати структурні коди розрядності $R = 4$. Нехай також множина операцій переходів утворена наступними ОП:

$$O_1 : K_{I_1}(a^{t+1}) = (K_{I_1}(a^t) + 5) \bmod 16, \quad (4.17)$$

$$O_2 : K_{I_2}(a^{t+1}) = (K_{I_2}(a^t) \div 2) \bmod 16, \quad (4.18)$$

$$O_3 : K_{I_3}(a^{t+1}) = \overline{K_{I_3}(a^t)}, \quad (4.19)$$

де a^t – поточний стан, a^{t+1} – стан переходу. Для ОП O_1 та O_2 будемо інтерпретувати структурні коди станів як беззнакові цілі в діапазоні [0; 15]. Для операції O_3 будемо розглядати структурні коди станів як чотирирозрядні двійкові вектори, що збігаються з відповідними структурними кодами.

Припустимо, що стан a_1 має максимальне значення q серед усіх станів ГСА Γ_{4-5}. Нехай на кроці 7 йому зіставлений структурний код $K_S(a_1) = 0101_2$. Реалізуємо три вихідні переходи зі стану a_1 за допомогою операцій $O_1 - O_3$, зіставивши кожному переходу деяку ОП, а стану переходу – структурний код, що є результатом виконання даної ОП над кодом $K_S(a_1)$.

Як випливає з виразів (4.17) – (4.19), в операціях переходів $O_1 - O_3$ результат не залежить від значень логічних умов, і єдиним їхнім аргументом є відповідний проміжний код стану a_1. Якщо ту саму ОП зіставити декільком переходам зі стану a_1, то декільком станам будуть привласнені однакові структурні коди, що унеможливить синтез логічної схеми автомата. Із цієї причини для реалізації трьох вихідних переходів зі стану a_1 повинні бути використані всі три ОП.

При $K_S(a_1) = 0101_2$ виконання над даним кодом операцій переходів O_1 і O_3 дає в результаті однаковий структурний код 0101_2, який повинен бути зіставлений двом із трьох станів переходів. В результаті будь-який спосіб зіставлення ОП вихідним переходам приводить до порушення унікальності структурних кодів станів в рамках імплементованої ГСА. Таким чином, при

$K_S(a_1)=0101_2$ реалізація всіх переходів зі стану a_1 за допомогою заданої множини ОП неможлива, і даний структурний код не може використовуватися для кодування стану a_1.

При $K_S(a_1)=0000_2$ результат операції O_2 дорівнює 0000_2 і збігається з кодом $K_S(a_1)$. Зіставлення операції O_2 кожному з переходів зі стану a_1 приводить до порушення унікальності структурних кодів станів у заданій ГСА. Отже, структурний код 0000_2 також не може бути використаний для кодування стану a_1.

При $K_S(a_1)=1001_2$ операції O_1-O_3 дають структурні коди 1110_2, 0100_2 та 0110_2 відповідно. Дані коди не збігаються як між собою, так і з кодом $K_S(a_1)=1001_2$, і можуть бути надані станам a_2-a_4 у будь-якому порядку. Таким чином, структурний код 1001_2 при використанні ОП (4.17) – (4.19) є припустимим для кодування стану a_1 у фрагменті ГСА на рис. 4.12.

Причина того, що розглянута спроба реалізації всіх вихідних переходів операційним способом виконана саме для стану $a(q_{\max})$, полягає в тому, що для реалізації переходів із цього стану не потрібне використання транзитних станів. Структурний код, що надається стану переходу, є або вільним, або зайнятим іншим станом переходу або станом $a(q_{\max})$. Якщо код вільний, то перехід у нього здійснюється безпосередньо за допомогою обраної ОП без використання транзитних станів. Якщо ж код зайнятий, процес алгебраїчного синтезу для поточного значення $K_S(a(q_{\max}))$ припиняється. Таким чином, відсутність транзитних станів при реалізації переходів зі $a(q_{\max})$ сприяє зниженню середнього числа тактів роботи автомата, що витрачається на один прохід алгоритму.

Порядок реалізації вихідних переходів, так само, як і порядок перебору ОП для реалізації кожного переходу, можуть бути різними. Наприклад, перебір переходів може здійснюватися в порядку убування їх імовірностей, а перебір операцій переходів – у порядку збільшення апаратурних витрат у їхніх

комбінаційних схемах. Різні способи перебору можуть приводити до різних варіантів кодування станів переходів, що, у свою чергу, позначається на результатах кроку 9 зокрема й на результатах методу алгебраїчного синтезу в цілому. При цьому вибір порядку реалізації переходів і порядку перебору ОП, що приводять до найкращого кінцевого результату, не є очевидним і гарантовано можливий лише шляхом повного перебору варіантів.

Оскільки метод, що пропонується, не ставить метою пошук оптимального розв'язку задачі алгебраїчного синтезу, на даному кроці припустимо виконувати перебір переходів і операцій переходів у довільному порядку.

Крок 9. На кроці 7 стан $a(q_{\max})$ був закодований структурним кодом $K_S(a(q_{\max}))$. На кроці 8 структурні коди були привласнені станам, у які є переходи зі стану $a(q_{\max})$. Таким чином, частина станів вже одержала структурні коди, які повинні залишатися незмінними до завершення алгебраїчного синтезу.

На кроці 9 необхідно виконати наступне:

– зіставити незакодованим станам унікальні структурні коди із множини K_S^R;

– реалізувати якомога більшу кількість переходів за допомогою ОП із множини *O*;

– реалізувати переходи, що залишилися, в канонічний спосіб.

Ця задача вирішується як шляхом повного перебору варіантів зіставлення, так і за допомогою алгоритмів, які тією чи іншою мірою скорочують кількість переборів. Один з таких алгоритмів представлений наступними кроками 9.1-9.7:

9.1. Формування множини неопрацьованих станів $A' = A \setminus a(q_{\max})$ і множини оброблених станів $A'' = A \setminus A' = \{a(q_{\max})\}$.

9.2. Вибір стану $a_i \in A'$ з максимальним значенням q. Якщо таких станів декілька, обирається будь-який. Такий вибір є евристичним і сприяє реалізації переходів, що більш часто виконуються, в операційний спосіб в умовах максимально можливого числа вільних структурних кодів у множині K_S^R. Це в

загальному випадку дозволяє скоротити кількість доданих транзитних станів та зменшити середнє число Q тактів роботи автомата.

9.3. Реалізація якомога більшого числа переходів зі стану a_i за допомогою операцій переходів із множини O. Припускається використання транзитних станів.

Виконання даного кроку полягає в послідовній реалізації вихідних переходів стану a_i в довільному порядку. При цьому слід враховувати наступне:

1. Якщо існують переходи в стан a_i зі станів, що належать множині A'', то для стану a_i структурний код $K_S(a_i)$ на даний момент вже відомий, оскільки був наданий при реалізації зазначених переходів.

2. Якщо переходи зі станів, що належать множині A'', у стан a_i відсутні, то $K_S(a_i)$ на даний момент невідомий і має бути обраний серед вільних структурних кодів. Пропонований метод алгебраїчного синтезу не передбачає в цьому випадку який-небудь спеціальний алгоритм і дозволяє довільний вибір $K_S(a_i)$.

3. Для деяких станів, у які виконуються переходи зі стану a_i, структурні коди можуть бути відомі, для деяких – невідомі.

Будь-яка структура МПА з ОАП припускає три способи реалізації довільного мікропрограмного переходу:

1) операційна реалізація без використання транзитних станів;

2) операційна реалізація з використанням транзитних станів;

3) канонічна реалізація.

Враховуючи, що критерієм ефективності структур МПА з ОАП стосовно канонічної структури є зниження апаратурних витрат, найбільш кращим є перший спосіб, найменш кращим – третій. Оскільки реалізація переходу першим і другим способом може виявитися неможливою, умовимося до кожного переходу застосовувати дані способи послідовно в порядку їх перерахування вище.

Нехай перехід зі стану a_i веде в деякий стан a_j. Розглянемо для даного переходу застосування кожного із трьох способів реалізації.

Перший спосіб (операційна реалізація без використання транзитних станів) полягає в пошуку такої ОП із множини *O*, результатом виконання якої над проміжним кодом, відповідним до коду $K_S(a_i)$, буде проміжний код, відповідний до коду $K_S(a_j)$.

Якщо до моменту реалізації переходу код $K_S(a_j)$ відомий, достатньо послідовно перебрати наявні ОП і вибрати будь-яку підходящу. Якщо $K_S(a_j)$ невідомий, слід спробувати вибрати таке його значення із множини вільних структурних кодів, при якому в множині *O* може бути знайдена підходяща ОП.

Другий спосіб (операційна реалізація з використанням транзитних станів) для випадку, коли $K_S(a_j)$ відомий, розглянута в п. 4.2.5.

Якщо $K_S(a_j)$ невідомий, він повинен бути обраний в якийсь спосіб. З одного боку, можна перебрати всі структурні коди, припустимі для кодування a_j, і вибрати такий, при якому послідовність транзитних станів має мінімальну довжину. З іншого боку, використання більш короткої транзитної послідовності при реалізації даного переходу може приводити до більш довгих послідовностей або навіть неможливості транзитної реалізації інших переходів, що знизить загальну ефективність одержуваного розв'язку задачі алгебраїчного синтезу. Складність у прогнозуванні наслідків вибору конкретного значення $K_S(a_j)$ визначає доцільність його довільного вибору із множини вільних структурних кодів.

Третій спосіб (канонічна реалізація) застосовується при неможливості реалізації переходу першим або другим способами. Якщо прийняте рішення реалізовувати перехід $a_i \to a_j$ в канонічний спосіб, то на поточному кроці запам'ятовується лише спосіб його реалізації, а на кроці 10 даний перехід включається в абстрактну й структурну часткові функції переходів, що поєднують усі «канонічні» переходи ГСА.

Канонічна реалізація переходу можлива лише при відомому значенні структурного коду $K_S(a_j)$ стану переходу. Якщо $K_S(a_j)$ невідомий, він повинен

бути обраний в якийсь спосіб. Пропонований метод алгебраїчного синтезу не передбачає в цьому випадку який-небудь спеціальний алгоритм і дозволяє довільний вибір $K_S(a_j)$.

9.4. Перенесення стану a_i із множини A' до множини A''.

9.5. Якщо множина A' не порожня, здійснюється перехід до кроку 9.2, інакше – до кроку 10.

Крок 10. Тут слід лише зазначити, що після того, як на кроці 8 усі переходи зі стану $a(q_{\max})$ вдалося реалізувати операційним способом, формальний розв'язок задачі алгебраїчного синтезу буде отриманий навіть у тому випадку, якщо на кроці 9 усі інші переходи були реалізовані канонічним способом.

Крок 11. У пропонованому методі кількість ітерацій «головного» циклу дорівнює 2^R, а вибір кращого результату серед отриманих виконується на кроці 12 із множини знайдених формальних розв'язків.

Ніщо, однак, не заважає проводити аналіз кожного формального розв'язку відразу після його одержання, вибираючи кращий результат у міру виконання головного циклу. Такий підхід зручний тим, що «головний» цикл може бути перерваний у будь-який момент, якщо знайдений *ефективний* розв'язок задачі алгебраїчного синтезу, що задовольняє встановленим критеріям. З іншого боку, переривання циклу може приводити до того, що більш ефективні розв'язки не будуть знайдені.

Крок 12. Достатньою умовою доцільності використання МПА з ОАП замість канонічного МПА є існування хоча б одного ефективного розв'язку задачі алгебраїчного синтезу. Виконуваний на даному кроці пошук ефективних розв'язків на множині формальних розв'язків може приводити до наступних результатів:

1. Ефективні розв'язки відсутні.

2. Знайдений єдиний ефективний розв'язок.

3. Знайдено кілька ефективних розв'язків.

В останньому випадку серед множини ефективних розв'язків потрібно вибрати найкращий відповідно до встановлених критеріїв оптимальності та вимогам до проектування.

Зіставлення операцій переходів окремим переходам автомата при виконанні кроків 8 і 9 дозволяє використовувати запропонований метод алгебраїчного синтезу як частину методу структурного синтезу МПА з ОАП зі структурою U_1 або U_3. Для застосування даного методу до структур U_2 та U_4 його треба модифікувати з урахуванням того, що в зазначених структурах окрема операція переходів може бути зіставлена стану автомата, реалізуючи усі без винятку переходи з даного стану.

Приклад алгебраїчного синтезу МПА з ОАП з використанням запропоноваго методу наведений в додатку Б.

4.3 Синтез логічної схеми мікропрограмного автомата з операційним автоматом переходів

На кроці 12 методу алгебраїчного синтезу МПА з ОАП, запропонованого в п. 4.2.8, визначається ефективність знайдених формальних розв'язків, яка виражається меншим числом апаратурних витрат в логічній схемі МПА з ОАП у порівнянні зі схемою МПА з канонічною структурою. Для цього очевидною є необхідність синтезу логічної схеми МПА з ОАП відповідно до результатів алгебраїчного синтезу.

Задача синтезу логічної схеми автомата відповідно до результатів алгебраїчного синтезу є, згідно з п. 4.1, другою задачею структурного синтезу МПА з ОАП. Розглянемо розв'язок даної задачі на прикладі формального розв'язку задачі алгебраїчного синтезу, наведеного в додатку Б [16, 20].

Структура U_1 МПА з ОАП (рис. 3.4) включає наступні блоки:

– операційна частина;

– Z-підсхема;

– регістр пам'яті;

– схема формування мікрооперацій.

Розглянемо синтез даних блоків.

Операційна частина.

У загальному випадку даний блок містить комбінаційні схеми усіх задіяних операцій переходів, та комбінаційну схему, що реалізує частину переходів канонічним способом. Кожна КС формує часткову структурну функцію переходів d_i, значення якої надходить до мультиплексора результату. Для побудови мультиплексора всі функції d_i слід закодувати двійковим кодом $K(d_i)$ розрядності $R_Z = \lceil log_2 N_d \rceil$.

Для розглянутого прикладу комбінаційна схема ОП O_1, що формує функцію d_1, являє собою трирозрядний суматор, на одне плече якого подається код поточного стану T, на друге – двійкова константа 101, що відповідає значенню 5 у проміжній алгебрі G_{I_1}.

Комбінаційна схема ОП O_2, що формує функцію d_2, складається із трирозрядного елемента «XOR», що виконує дану операцію порозрядно між кодом поточного стану T та двійковою константою 100.

Для синтезу комбінаційної схеми, що реалізує підмножину автоматних переходів в канонічний спосіб, слід дотримуватися канонічного підходу, що включає наступні етапи [29, 87]:

– побудова структурної таблиці переходів;

– формування і мінімізація системи булевих функцій переходів;

– синтез комбінаційної схеми, вихід якої відповідає частковій структурній функції переходів d_3.

Підкреслимо, що дані дії виконуються лише для тієї підмножини переходів, які, згідно з результатами алгебраїчного синтезу, вирішено реалізовувати в канонічний спосіб.

У розглянутому прикладі канонічним способом реалізується єдиний перехід $a_0 \to a_1$. Оскільки такий перехід один, комбінаційна схема функції d_3 є

формувачем константи, яка дорівнює структурному коду стану переходу $K_S(a_1)$=000.

Закодуємо функції $d_1 - d_3$ дворозрядним двійковим кодом $< z_1, z_2 >$. Нехай $K(d_1)$=00, $K(d_2)$=01, $K(d_3)$=10. Дані коди використовуються при синтезі мультиплексора результату.

На рис. 4.13 показана функціональна схема операційної частини для розглянутого прикладу.

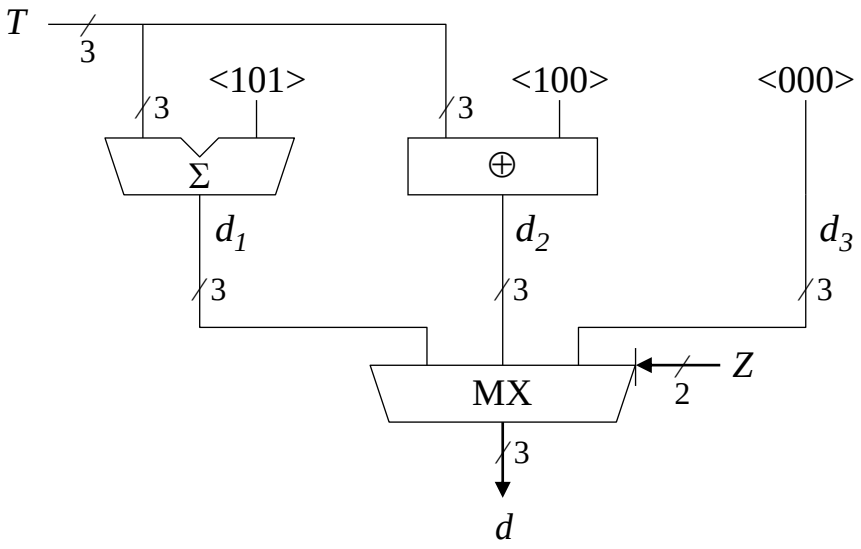


Рисунок 4.13 – Функціональна схема операційної частини

Схема формування кодів операцій переходів.

Дана схема, що позначена на рис. 3.4 як Z-підсхема, може бути реалізована або в базисі логічних елементів, або в базисі ЗП (ПЗП, ППЗП тощо). Незалежно від елементного базису, синтез Z-підсхеми проводиться за операційною таблицею переходів, структура якої викладена в п. 3.3.1 і показана на прикладі табл. 3.1. Для розглянутого прикладу операційна таблиця переходів відповідає табл. 4.4.

Представляючи код *T* поточного стану двійковими змінними $T_1 - T_3$, можна записати наступні рівняння для функцій z_1 і z_2 (у вигляді ДНФ):

$$z_1 = T_1 T_2 \overline{T}_3;\ z_2 = \overline{T}_1 \overline{T}_2 \overline{T}_3 \overline{x}_1 \vee \overline{T}_1 \overline{T}_2 T_3 x_1 \vee T_1 T_2 T_3.$$

Таблиця ЗП Z-підсхеми будується за аналогією з табл. 3.4 і для розглянутого прикладу представлена табл. 4.5.

Таблиця 4.4

Операційна таблиця переходів (ГСА Γ'_{4-6})

a_m	$K_S(a_m)$	a_s	$K_S(a_s)$	X_h	O_h	z_h	h
a_0	110	a_1	000	1	–	z_1	1
a_1	000	a_2	101	x_1	O_1	–	2
		a_3	100	$\bar{x}_1$	O_2	z_2	3
a_2	101	a_5	010	1	O_1	–	4
a_3	100	a_4	001	1	O_1	–	5
a_4	001	a_2	101	x_1	O_2	z_2	6
		a_0	110	$\bar{x}_1$	O_1	–	7
a_5	010	a_6	111	1	O_1	–	8
a_6	111	a_7	011	1	O_2	z_2	9
a_7	011	a_2	000	1	O_1	–	10

Таблиця 4.5

Вміст ЗП Z-підсхеми (ГСА Γ'_{4-6})

$T_1\ T_2\ T_3\ x_1$	$z_1\ \ z_2$
0 0 0 0	0 1
0 0 0 1	0 0
0 0 1 0	0 0
0 0 1 1	0 1
0 1 0 0	0 0
0 1 0 1	0 0
0 1 1 0	0 0
0 1 1 1	0 0
1 0 0 0	0 0
1 0 0 1	0 0
1 0 1 0	0 0
1 0 1 1	0 0
1 1 0 0	1 0
1 1 0 1	1 0
1 1 1 0	0 1
1 1 1 1	0 1

Регістр пам'яті.

В якості регістру пам'яті може бути обраний стандартний регістр необхідної розрядності, організований на базі тригерів D-типу. У такому випадку окремий синтез регістру пам'яті не потрібний.

Схема формування мікрооперацій.

Структури МПА з ОАП, що пропонуються в дисертаційній роботі, припускають різні способи реалізації схеми формування мікрооперацій, деякі з яких розглянуті в п. 3.4. Зазвичай доцільною є канонічна реалізація схеми СФМО за системою булевих рівнянь [29, 87].

Відзначимо, що особливості синтезу структури U_2 викладені в п. 3.3.2, структур U_3 та U_4 – в п. 3.3.3. У порівнянні з алгебраїчним синтезом МПА з ОАП, синтез його логічної схеми є однозначним, оскільки використовує відомі підходи до побудови окремих структурних елементів.

4.4 Висновки до четвертого розділу

1. В четвертому розділі вирішено наукове завдання розробки методології синтезу запропонованих структур мікропрограмних автоматів, що основані на принципі операційного перетворення кодів станів.

2. Процес структурного синтезу мікропрограмного автомата з операційним автоматом переходів може бути розділений на два етапи: алгебраїчний синтез і синтез логічної схеми автомата відповідно до результатів алгебраїчного синтезу. Складність даних етапів дозволяє розглядати розробку кожного з них як окрему наукову задачу.

3. Задача алгебраїчного синтезу мікропрограмного автомата з операційним автоматом переходів зводиться до формування системи ізоморфізмів (3.9). При цьому для заданого автомата можливе формування, у загальному випадку, множини різних систем ізоморфізмів, кожна з яких становить собою формальний розв'язок задачі алгебраїчного синтезу. Підмножина формальних розв'язків, що дозволяють одержати логічну схему МПА з ОАП, більш ефективну за

апаратурними витратами, ніж схема еквівалентного МПА з канонічною структурою, є множиною ефективних розв'язків задачі алгебраїчного синтезу. Підмножина ефективних розв'язків, що приводять до максимально можливого виграшу в апаратурних витратах, є множиною оптимальних розв'язків задачі алгебраїчного синтезу МПА з ОАП.

4. Під методом алгебраїчного синтезу МПА з ОАП розуміється деяка послідовність етапів, що дозволяє одержати формальний розв'язок задачі алгебраїчного синтезу. Припустимі відмінності в реалізації та порядку проходження окремих етапів дозволяють говорити про потенційну можливість існування множини методів алгебраїчного синтезу і приводять до задачі розробки методів алгебраїчного синтезу МПА з ОАП.

5. Множина теоретичних і практичних питань, що стосуються алгебраїчного синтезу МПА з ОАП, може бути виділена в окрему наукову область, яку названо методологією алгебраїчного синтезу мікропрограмних автоматів з операційним автоматом переходів.

6. Процес алгебраїчного синтезу МПА з ОАП може бути представлений у вигляді структури, що відбиває причинно-наслідкові зв'язки між елементами системи ізоморфізмів (3.9). У дисертаційній роботі розроблений ряд структур, що характеризують даний процес із погляду вхідних даних та результатів. Будь-який метод алгебраїчного синтезу МПА з ОАП припускає певну послідовність у формуванні елементів системи (3.9), у зв'язку із чим може бути зіставлений до однієї зі структур процесу алгебраїчного синтезу. У загальному випадку одна структура може характеризувати множину методів, що володіють загальними рисами, властивими даній структурі. Запропоноване в роботі структурне представлення процесу алгебраїчного синтезу може використовуватися при розробці нових і класифікації існуючих методів алгебраїчного синтезу МПА з ОАП.

7. Алгебраїчний синтез мікропрограмного автомата з операційним автоматом переходів може бути виконаний шляхом повного перебору варіантів зіставлення структурних кодів станам автомата та операцій переходів окремим

автоматним переходам. Однак навіть для автоматів малої складності час, що витрачається на повний перебір варіантів, є неприйнятно великим. У даному аспекті практично доцільною представляється розробка методів алгебраїчного синтезу, заснованих на скороченому переборі варіантів.

8. Одним зі способів реалізації автоматних переходів із використанням наявного набору операцій переходів є використання так званих транзитних станів. Перевагою даного підходу є можливість зниження апаратурних витрат у схемі операційного автомата переходів, недоліком – збільшення середньої кількості тактів роботи автомата, що витрачається на один прохід алгоритму.

9. Якщо відомі ймовірності появи вхідних сигналів, може бути визначена середня кількість переходів у кожний зі станів автомата, що відбувається за один прохід алгоритму. Знання даних значень дозволяє враховувати в процесі алгебраїчного синтезу доцільність використання транзитних станів з погляду збільшення середнього часу виконання алгоритму, що імплементується мікропрограмним автоматом.

10. Структурна організація МПА з ОАП припускає використання структурних кодів станів, розрядність яких перевищує мінімально припустиме для наявної кількості станів значення. Результатом є, з одного боку, збільшення кількості варіантів перебору кодів станів і операцій переходів, з іншого боку – збільшення кількості формальних розв'язків задачі алгебраїчного синтезу. Враховуючи практичну неможливість розв'язку задачі алгебраїчного синтезу шляхом повного перебору, даний прийом слід використовувати як частину методів алгебраїчного синтезу, що реалізують скорочений перебір варіантів.

11. Запропоновано метод алгебраїчного синтезу МПА з ОАП зі структурою U_1 за граф-схемі алгоритму. Основу методу становить перебір структурних кодів для кодування стану з максимальним значенням середньої кількості переходів у даний стан за один прохід алгоритму, причому на кожній ітерації перебору може бути отриманий формальний розв'язок задачі алгебраїчного синтезу. Метод припускає використання фіксованої множини операцій переходів і модифікацію

заданої ГСА шляхом додавання порожніх операторних вершин, відповідних до транзитних станів.

12. Синтез логічної схеми МПА з ОАП відповідно до результатів алгебраїчного синтезу є другим етапом структурного синтезу МПА з ОАП і полягає в послідовному синтезі всіх елементів структурної схеми автомата.

5 ДОСЛІДЖЕННЯ МІКРОПРОГРАМНОГО АВТОМАТА З ОПЕРАЦІЙНИМ АВТОМАТОМ ПЕРЕХОДІВ

5.1 Загальна методологія досліджень

У розділі 3 дисертаційної роботи пропонуються структури мікропрограмного автомата з операційним автоматом переходів, метою розробки яких є зменшення апаратурних витрат в логічній схемі автомата у порівнянні з іншими структурними реалізаціями. Вибір апаратурних витрат у якості критерію оптимальності ставить задачу проведення порівняльної оцінки ефективності запропонованих структур МПА з ОАП з відомими структурами МПА [12, 43, 181].

Мікропрограмний автомат з операційним автоматом переходів не є модифікацією якої-небудь відомої структури МПА і позиціонується в даній роботі як окрема структурна гілка мікропрограмних автоматів. Відсутність структури-прототипу приводить до необхідності вибору структури МПА, порівняння з якою дозволить найбільш повно охарактеризувати запропоновані структури МПА з ОАП з погляду обраного критерію оптимальності.

Аналіз ефективності багатьох відомих структур МПА, для яких критерієм оптимальності виступають апаратурні витрати, проводиться у порівнянні з канонічною структурою [36]. Це дозволяє порівнювати структури між собою відносно канонічної структури, не проводячи окремих досліджень, спрямованих на порівняння неканонічних структур між собою. Умовимось досліджувати ефективність структур МПА з ОАП, що пропонуються в дисертаційній роботі, щодо канонічної структури МПА. Ефективність E^{U_i} структури U_i МПА з ОАП визначимо наступним виразом:

$$E^{U_i} = \frac{H^{U_K}}{H^{U_i}}, \qquad (5.1)$$

де H^{U_K}, H^{U_i} – чисельно виражені витрати апаратури в МПА з канонічною структурою та в еквівалентному МПА з ОАП зі структурою U_i відповідно. Під терміном «еквівалентний» слід розуміти такий МПА з ОАП, який може замінити заданий МПА з канонічною структурою при задоволенні вимог проектування і експлуатації об'єкту керування.

У загальному випадку еквівалентний МПА з ОАП може відрізнятися від канонічного МПА рядом параметрів, наприклад, кількістю станів, середнім часом виконання імплементованого алгоритму керування, розрядністю та значеннями структурних кодів станів. Подібні відмінності можуть бути викликані, наприклад, застосованим методом алгебраїчного синтезу МПА з ОАП, який передбачає можливість використання транзитних станів (див. п. 4.2.5).

Структура U_i МПА з ОАП є більш ефективною за апаратурними витратами у порівнянні з канонічною структурою МПА у випадку $E^{U_i} > 1$, причому економія апаратурних витрат становить $(E^{U_i} - 1)/E^{U_i}$ відсотків від витрат у канонічній структурі. При $E^{U_i} = 1$ порівнювані структури рівносильні за критерієм ефективності. При $E^{U_i} < 1$ структура U_i менш ефективна, ніж канонічна структура, і її використання з погляду апаратурних витрат є недоцільним.

Кожна з величин H^{U_K} та H^{U_i} є сумою чисельно виражених апаратурних витрат у всіх блоках відповідної структури. Для канонічного МПА H^{U_K} визначається виразом (5.2), для структур $U_1 - U_4$ – виразами (5.3) – (5.6) відповідно.

$$H^{U_K} = H_{СФП}^{U_K} + H_{РП}^{U_K} + H_{СФМО}^{U_K}; \quad (5.2)$$

$$H^{U_1} = H_{ОЧ}^{U_1} + H_{РП}^{U_1} + H_{Z}^{U_1} + H_{СФМО}^{U_1}; \quad (5.3)$$

$$H^{U_2} = H_{ОЧ}^{U_2} + H_{РП}^{U_2} + H_{Z}^{U_2} + H_{СФМО}^{U_2}; \quad (5.4)$$

$$H^{U_3} = H_{M}^{U_3} + H_{ОЧ}^{U_3} + H_{РП}^{U_3} + H_{Z}^{U_3} + H_{СФМО}^{U_3}; \quad (5.5)$$

$$H^{U_4} = H_M^{U_4} + H_{ОЧ}^{U_4} + H_{РП}^{U_4} + H_Z^{U_4} + H_{СФМО}^{U_4}. \qquad (5.6)$$

Величина E^{U_i} у виразі (5.1) не має одиниці виміру, однак чисельник і знаменник є іменованими і повинні бути виражені в однакових одиницях виміру. Отже, у тих самих одиницях мають бути виражені всі доданки у виразах (5.2) – (5.6). Хоча структури МПА з ОАП, що пропонуються в дисертаційній роботі, не орієнтовані на якийсь конкретний елементний базис, від його вибору залежить вибір одиниці виміру апаратурних витрат.

У рамках досліджень пропонованих структур МПА умовимось використовувати елементний базис ПЛІС FPGA і ті одиниці виміру апаратурних витрат, у яких представляються результати синтезу для конкретної моделі ПЛІС при використанні обраної САПР (наприклад, LUT). Хоча синтез МПА можливий у базисі ПЛІС різних виробників, для проведення експериментів достатньо дослідити реалізацію схем автоматів у базисі ПЛІС якогось одного виробника. У якості такого виробника в даній роботі обрана компанія Xilinx, для якої особливості використання продукції та спеціалізованих засобів проектування цифрових систем докладно освітлені в літературі [101, 162]. Таким засобом проектування є, наприклад, САПР Xilinx ISE версії 9.2i – вільно розповсюджуваний пакет, що має, у тому числі, наступні можливості [102]:

– реалізація проектів мовою опису апаратури VHDL;

– підтримка всіх основних серій ПЛІС фірми Xilix на момент випуску САПР;

– моделювання процесу синтезу проекту на кристалі ПЛІС із вираженням апаратурних витрат на реалізацію проекту в LUT-елементах.

У цифровій схемотехніці складні обчислювальні структури представляються складеними із типових функціональних вузлів, до яких належать [125, 164, 255, 262]:

– комбінаційні логічні схеми (КЛС), що задаються таблицею істинності і синтезуються з використанням канонічного методу синтезу за системою булевих рівнянь [125];

– комутаційні елементи (мультиплексори, дешифратори тощо);

– елементи з пам'яттю (тригери, регістри, лічильники);

– операційні елементи АЛП, що реалізують різні арифметичні та логічні операції над двійковими векторами;

– запам'ятовувальні пристрої (ОЗП, ПЗП, ППЗП)

та інші. До теперішнього часу для всіх типів вузлів розроблені ефективні VHDL-описи з використанням синтезованої підмножини операторів [74, 75, 126, 139, 158].

Окремі блоки в канонічній структурі МПА та структурах $U_1 - U_4$ мають схемну архітектуру, відповідну до архітектури типових функціональних вузлів. Наприклад, в канонічному МПА схеми СФП та СФМО реалізуються за системою булевих рівнянь у вигляді КЛС, а схема РП становить собою стандартний *R*-розрядний регістр. Виключенням можуть бути блоки, що входять в операційну частину ОАП та реалізують якісь нестандартні операції переходів. Однак у більшості випадків розбивка таких блоків на більш прості не викликає складностей.

Відповідність елементів досліджуваних структур типам функціональних вузлів показана в табл. 5.1. В останньому стовпці таблиці наведені посилання на літературні джерела, у яких наводяться синтезовані VHDL-моделі відповідних функціональних блоків. Дані моделі мають схожий вигляд в різних джерелах і легко перетворяться в VHDL-описи блоків досліджуваних структур МПА.

Як видно з таблиці, одному типовому функціональному блоку відповідають, у загальному випадку, декілька блоків у досліджуваних структурах МПА. Це дає можливість провести дослідження апаратурних витрат не для структурних блоків МПА, а для відповідних їм функціональних блоків, поширивши отримані результати на структурні блоки МПА.

Визначимо наступний порядок проведення досліджень:

1. Дослідження апаратурних витрат у типових функціональних блоках при синтезі їх VHDL-моделей в базисі ПЛІС фірми Xilinx з використанням САПР Xilinx ISE 9.2i.

2. Ототожнення результатів, отриманих для типових функціональних блоків, з результатами для структурних елементів МПА.

3. Дослідження ефективності структур мікропрограмного автомата з операційним автоматом переходів.

Таблиця 5.1

Відповідність між елементами досліджуваних структур МПА і типовими функціональними блоками

Типовий функц. блок	Структура МПА	Елемент структури	Синтезовані VHDL-моделі
КЛС, що реалізує систему БФ	канонічна	СФП, СФМО	[75], стор. 258 [94], стор. 364 [150], стор. 109
	U_1 - U_4	Z-підсхема, СФМО	
регістр	усі структури	РП	[75], стор. 333 [94], стор. 372
мультиплексор	U_1 - U_4	мультиплексор в блоці ОЧ	[75], стор. 292 [162], стор. 122 [166], стор. 464
ЗП	канонічна	СФМО	[75], стор. 297 [162], стор. 124
	U_1 - U_4	Z-підсхема, СФМО	
суматор	U_1 - U_4	елемент ОЧ	[75], стор. 297 [94], стор. 379 [166], стор. 500
інкрементор	U_1 - U_4	елемент ОЧ	[75], стор. 312
схема множення	U_1 - U_4	елемент ОЧ	[75], стор. 309 [166], стор. 518
схема зсуву	U_1 - U_4	елемент ОЧ	[166], стор. 588
паралельний логіч. елемент	U_1 - U_4	елемент ОЧ	[75], стор. 161 [139], стор. 57

5.2 Дослідження апаратурних витрат у типових функціональних блоках цифрових пристроїв

5.2.1 Комбінаційна схема, що реалізує систему булевих функцій

Нехай система булевих функцій, представлених у формі ДНФ, містить s аргументів, t функцій і q проміжних термів. Умовимось розглядати систему, в

якій кожне рівняння містить у правій частині рівно l булевих термів. Будемо також вважати, що число кон'юнктивних компонентів у кожному термі є однаковим і дорівнює s. Також будемо враховувати, що величина l не може перевищувати загальна кількість q проміжних термів.

Витрати апаратури $H_{КЛС}$ на реалізацію системи булевих функцій, представлених у формі ДНФ, є сумою двох величин: витрат $H_{\&}(q,s)$ на реалізацію кон'юнктивних термів і витрат $H_{\vee}(t,l)$ на їхнє диз'юнктивне об'єднання в окремі рівняння:

$$H_{КЛС} = H_{\&} + H_{\vee}. \tag{5.7}$$

Оскільки функції $H_{\&}$ та $H_{\vee}$, згідно з аргументами, є взаємно незалежними, кожна з них може бути досліджена окремо.

1. Дослідження апаратурних витрат на реалізацію термів.

Апаратурні витрати $H_{\&}$, затрачувані на реалізацію множини термів, що використовуються у системі булевих функцій, залежать від кількості q термів та кількості s змінних у кожному термі.

Дослідимо залежність величини $H_{\&}$ від параметрів q та s. Для цього скористаємося VHDL-моделлю, у якій кожний терм пов'язаний з окремим вихідним сигналом. Приклад моделі для системи БФ (5.8) представлений лістингом А.7.

$$\begin{cases} y_1 = \bar{x}_1 x_2 x_3 x_4 \vee x_1 \bar{x}_2 \bar{x}_3 x_4 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4; \\ y_2 = x_1 x_2 \bar{x}_3 x_4 \vee x_1 x_2 x_3 x_4 \vee x_1 \bar{x}_2 \bar{x}_3 x_4. \end{cases} \tag{5.8}$$

Позначимо експериментально обумовлену функцію $H_{\&}$ символом $H_{\&}^{E}$. Визначимо наступні умови проведення експерименту:

1. Величини q та s незалежно змінюються в діапазоні [10; 100] із кроком 10.

2. У кожному термі кількість кон'юнктивних компонентів однакова і дорівнює s. Компонентами термів можуть виступати як самі вхідні змінні, так і їх інверсні значення.

3. Вибір термів із множини всіх можливих термів виконується довільним способом. При цьому функція мінімізації схеми (наприклад, спільна реалізація фрагментів термів) припадає на САПР.

4. Визначення експериментальних значень $H_{\&}^{E}$ здійснюється за допомогою САПР Xilinx ISE 9.2i для ПЛІС типу FPGA фірми Xilinx, у яких апаратурні витрати вимірюються в LUT-елементах. Як показали дослідження, серії ПЛІС типу CPLD (Coolrunner-2, XC9500XL), що існують у фірми Xilinx, мають недостатню ресурсну ємність для реалізації використовуваних VHDL-моделей. При цьому можливість використання декількох CPLD для реалізації одного проекту в рамках даних досліджень не розглядається.

У табл. 5.2 наведені вимірювані в LUT-елементах значення функції $H_{\&}^{E}(q,s)$ для ПЛІС марки XC3S1500 серії Spartan 3. Дослідження показали, що для ПЛІС марки XC4VLX15 серії Virtex 4 та для ПЛІС більш нових серій результати виявляються схожими.

Зазначені мікросхеми є типовими представниками ПЛІС типу FPGA і мають достатню ресурсну ємність для реалізації VHDL-моделі при значеннях $q = s = 100$. Використовувана VHDL-модель не має прив'язки до конкретних марок ПЛІС, що дозволяє у разі необхідності повторити експеримент для ПЛІС інших серій.

На рис. 5.2, *а* вміст табл. 5.2 показаний у вигляді сімейства кривих $H_{\&}^{E}(s)$ для різних значень q (кожна крива відповідає одному рядку табл. 5.2). На рис. 5.2, *б* наведене сімейство кривих $H_{\&}^{E}(q)$ для різних значень s (кожна крива відповідає одному стовпцю табл. 5.2).

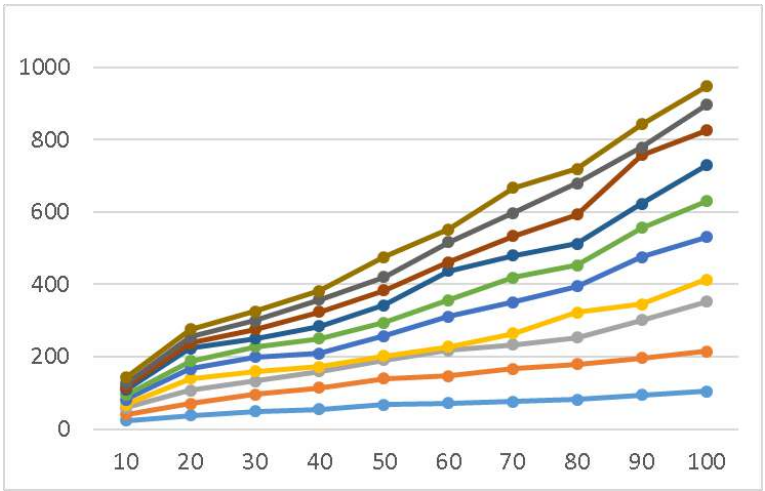
Загальний характер кривих, близький до лінійного, є наслідком регулярності внутрішньої структури ПЛІС типу FPGA разом із з регулярністю VHDL-моделі. Різке зниження апаратурних витрат, що спостерігається при значеннях $q = 10$ та $s = 10$, є результатом оптимізації логічної схеми засобами САПР.

Таблиця 5.2

Значення функції $H_{\&}^{E}(q,s)$ для ПЛІС XC3S1500, LUT

$q \setminus s$	10	20	30	40	50	60	70	80	90	100
10	23	40	59	68	82	94	108	115	126	142
20	38	70	107	140	167	188	224	238	255	276
30	49	96	133	160	199	227	251	276	300	325
40	54	114	159	173	209	251	284	324	358	382
50	67	139	191	202	257	294	343	383	420	476
60	71	147	218	228	311	357	438	462	517	552
70	76	167	234	264	351	419	480	534	598	667
80	82	180	253	322	395	453	513	593	680	720
90	94	196	301	346	476	557	624	758	779	843
100	104	215	352	414	531	630	729	826	897	948

а)



б)

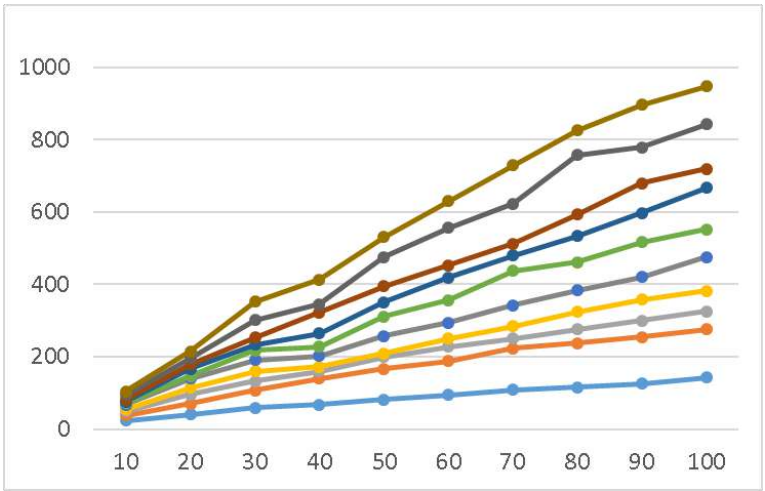


Рисунок 5.1 – Сімейства кривих для функцій $H_{\&}^{E}(s)$ $(а)$ та $H_{\&}^{E}(q)$ $(б)$

Не дивлячись на те, що у табл. 5.2 значення $H_{\&}^{E}$ дані для $q \geq 10$ та $s \geq 10$, функція $H_{\&}(q,s)$ формально існує і при нульових значеннях своїх аргументів. Оскільки у разі відсутності термів ($q = 0$) або при їх нульовому розмірі ($s = 0$) система булевих рівнянь не існує, витрати апаратури на її реалізацію відсутні, тобто $H_{\&}(q,s) = 0$. Зрозуміло, що при $q < 0$ або $s < 0$ функція не існує.

Додаткові дослідження показують, що заміна одних булевих термів іншими (такого ж розміру) незначно впливає на кількість апаратурних витрат. У табл. 5.3 значення $H_{\&}^{E1}(q,s)$ відповідають значенням з останнього рядка табл. 5.2. Значення $H_{\&}^{E2}(q,s)$ отримані за тих самих умов, але з використанням термів, відмінних від тих, що використовувалися при побудові табл. 5.2. Значення $\Delta H_{\&}^{E}$ відповідають модулю різниці величин $H_{\&}^{E2}(q,s)$ та $H_{\&}^{E1}(q,s)$, вираженому у відсотках від значення $H_{\&}^{E1}(q,s)$.

Таблиця 5.3

Порівняння значень функції $\Delta H_{\&}^{E}(q,s)$
при використанні різних термів для q=100

s	10	20	30	40	50	60	70	80	90	100
$H_{\&}^{E1}(q,s)$	104	215	352	414	531	630	729	826	897	948
$H_{\&}^{E2}(q,s)$	110	222	334	394	488	651	711	811	913	933
$\Delta H_{\&}^{E}$, %	5,7	3,3	5,1	4,8	8,1	3,3	2,5	1,8	1,8	1,6

З табл. 5.3 видно, що відхилення $\Delta H_{\&}^{E}$ не перевищує 10 %, становлячи в середньому близько 4 %. Даний факт, а також аналогічні дослідження, проведені для інших рядків табл. 5.2, дозволяють у рамках даної роботи узагальнити результати, зібрані в табл. 5.2, для довільних систем булевих рівнянь із відповідними параметрами s та q.

Відзначимо наступне. Табл. 5.2 містить значення функції $H_{\&}(q,s)$ для аргументів, що кратні 10 і знаходяться у діапазоні $[10; 100]$. Одержання значень

функції для q та s, не кратних 10 або лежачих за межами зазначеного діапазону, можливе двома способами: за допомогою VHDL-моделювання або шляхом аналітичної апроксимації вмісту табл. 5.2.

Перший спосіб аналогічний способу формування табл. 5.2 і вимагає побудови і дослідження окремої VHDL-моделі для кожної пари значень s і q.

Другий спосіб дозволяє приблизно оцінити значення функції $H_{\&}(q,s)$ у точках, не досліджених експериментально. Апроксимуємо вміст табл. 5.2 наступним виразом:

$$H_{\&}^{A} = \frac{q \cdot s}{10}, \tag{5.9}$$

де $H_{\&}^{A}$ – функція $H_{\&}$, що задана в аналітичний спосіб.

Табл. 5.4 містить відхилення $\Delta H_{\&}(q,s)$ між аналітичними та експериментальними значеннями функції $H_{\&}$.

Таблиця 5.4

Функція $\Delta H_{\&}(q,s)$, LUT

$q \setminus s$	10	20	30	40	50	60	70	80	90	100
10	-13	-20	-29	-28	-32	-34	-38	-35	-36	-42
20	-18	-30	-47	-60	-67	-68	-84	-78	-75	-76
30	-19	-36	-43	-40	-49	-47	-41	-36	-30	-25
40	-14	-34	-39	-13	-9	-11	-4	-4	2	18
50	-17	-39	-41	-2	-7	6	7	17	30	24
60	-11	-27	-38	12	-11	3	-18	18	23	48
70	-6	-27	-24	16	-1	1	10	26	32	33
80	-2	-20	-13	-2	5	27	47	47	40	80
90	-4	-16	-31	14	-26	-17	6	-38	31	57
100	-4	-15	-52	-14	-31	-30	-29	-26	3	52

Значення в кожній комірці вимірюється в LUT-елементах і обчислюється за формулою

$$\Delta H_{\&} = H_{\&}^{A} - H_{\&}^{E}, \qquad (5.10)$$

де $H_{\&}^{E}$ – значення з відповідної комірки табл. 5.2. У табл. 5.5 вміст табл. 5.4 виражений у відсотках від відповідних значень $H_{\&}^{E}$ з табл. 5.2.

Таблиця 5.5

Функція $\Delta H_{\&}(q,s)$, %

$q \setminus s$	10	20	30	40	50	60	70	80	90	100
10	-57	-50	-49	-41	-39	-36	-35	-30	-29	-30
20	-47	-43	-44	-43	-40	-36	-38	-33	-29	-28
30	-39	-38	-32	-25	-25	-21	-16	-13	-10	-8
40	-26	-30	-25	-8	-4	-4	-1	-1	1	5
50	-25	-28	-21	-1	-3	2	2	4	7	5
60	-15	-18	-17	5	-4	1	-4	4	4	9
70	-8	-16	-10	6	0	0	2	5	5	5
80	-2	-11	-5	-1	1	6	9	8	6	11
90	-4	-8	-10	4	-5	-3	1	-5	4	7
100	-4	-7	-15	-3	-6	-5	-4	-3	0	5

Проаналізуємо табл. 5.5.

1. В інтервалі $40 \le q \le 100$, $40 \le s \le 100$ функція (5.10) досить точно апроксимує функцію $H_{\&}(q, s)$, даючи середнє відхилення у +1 %.

2. При $q < 40$ або $s < 40$ функція (5.10) дає значення, менші за експериментальні. Даний факт пояснюється наступним.

При синтезі VHDL-моделей, побудованих за аналогією з моделлю на лістингу А.7, частина LUT-елементів обраної ПЛІС витрачається на формування інверсних значень вхідних сигналів *x* та на їхнє розгалуження при реалізації термів (назвемо ці витрати апаратури допоміжними). Якщо при більших значеннях *s* та *q* допоміжні витрати є незначними у порівнянні з витратами на реалізацію термів, то зі зменшенням *s* та *q* частка допоміжних витрат у загальних витратах апаратури зростає і при значеннях *s* та *q*, менших за 40, стає порівнянною з витратами на реалізацію термів.

Оскільки функція (5.10) добре підходить для великих значень s та q, при яких частка допоміжних витрат невисока, можна умовно вважати, що вона визначає тільки витрати на реалізацію термів і не враховує допоміжні апаратурні витрати. Спроба апроксимувати за допомогою (5.10) частину табл. 5.2 для значень $q < 40$, $s < 40$ приводить до того, що функція (5.10), обчислюючи витрати тільки на реалізацію термів без урахування допоміжних витрат, дає значення помітно менші за експериментальні. Так, при $q = 10$, $s = 10$ функція (5.10) дає значення 10, що на 13 одиниць менше за відповідне експериментальне значення (табл. 5.2). Різницю в 13 LUT-елементів слід вважати приблизно рівною допоміжним апаратурним витратам при синтезі використовуваної VHDL-моделі.

3. При значеннях s або q, менших за 30, значення $\Delta H_{\&}$ в табл. 5.5 становлять 30-50 %, що у відносному вимірі досить багато. Однак, оскільки загальні апаратурні витрати при даних значеннях аргументів невеликі, відхилення $\Delta H_{\&}$, виражене в LUT-елементах (табл. 5.4) також невелике. Наприклад, при $q = 10$, $s = 100$ маємо $H_{\&}^{E} = 142$ LUT, $H_{\&}^{A} = 100$ LUT, що дає відхилення (за модулем) $\Delta H_{\&} = 30\,\% = 42$ LUT. У той же час при $q = 100$, $s = 100$ відхилення $\Delta H_{\&} = 5\,\%$, що в абсолютних величинах становить 52 LUT-елементи, перевищуючи абсолютне значення $\Delta H_{\&}$ при $q = 10$, $s = 100$.

Відзначимо, що формула (5.9) теоретично може бути модифікована таким чином, щоб враховувати допоміжні апаратурні витрати. Однак з аналізу табл. 5.3 випливає, що відхилення величини $H_{\&}^{E}$, пов'язане зі структурою реалізованих термів, може нівелювати відхилення величини $\Delta H_{\&}$, викликане допоміжними витратами апаратури. Із цієї причини уточнення формули (5.9) для урахування допоміжних апаратурних витрат не гарантує підвищення точності обчислення функції $H_{\&}(q,s)$ і бачиться недоцільним. Будемо вважати, що формула (5.9) апроксимує вміст табл. 5.2 з достатньою точністю і може бути використана для обчислення параметра $H_{\&}$ у виразі (5.7).

2. Дослідження апаратурних витрат на диз'юнктивне об'єднання термів.

Перетворимо модель, представлену лістингом А.7, так, як показано на лістингу А.8. В отриманій моделі булеві терми *q* не підключені до вихідних портів, а реалізовані внутрішніми сигналами. Замість них до вихідних портів підключені функції *y*, що формуються як диз'юнкції відповідних термів згідно із системою (5.8).

VHDL-модель, представлена лістингом А.8, дозволяє експериментально визначити кількість апаратурних витрат на реалізацію всієї системи булевих рівнянь, тобто величину $H_{КЛС}$ відповідно до виразу (5.7).

Тоді, згідно з (5.7), величина $H_{\vee}$ буде визначатися як

$$H_{\vee} = H_{КЛС} - H_{\&}. \qquad (5.11)$$

Однак такий підхід виявляється незручним. Слід враховувати, що система булевих рівнянь характеризується, крім параметрів *s* і *q*, кількістю рівнянь *t* та кількістю термів у кожному рівнянні *l*. Отже, у виразі (5.11) $H_{\vee}$ є функцією не двох, а чотирьох аргументів, що ускладнює її дослідження.

Умовимося розглядати величину $H_{\vee}$ як функцію двох аргументів: $H_{\vee}(t, l)$. Для її експериментального дослідження скористаємося VHDL-моделлю, приклад якої представлений лістингом А.9.

У даній моделі вхідні порти *q* відповідають булевим змінним, що ототожнюються з кон'юнктивними термами ДНФ. Сигнали *y*, підключені до вихідних портів, відповідають булевим функціям реалізованої системи. У лістингу А.9 система включає $t = 5$ рівнянь по $l = 7$ термів у кожному, причому в рівняннях системи задіяні усі $q = 20$ булевих термів.

Позначимо експериментально визначену функцію $H_{\vee}$ символом $H_{\vee}^{E}$ і дослідимо її залежність від параметрів *t* та *l*. Визначимо наступні умови проведення експерименту:

1. Величина *t* змінюється в діапазоні [1;10] із кроком 1.

2. Величина *l* змінюється в діапазоні [10;100] із кроком 10.

3. У кожному рівнянні кількість термів однакова і дорівнює *l*.

4. Визначення чисельних значень $H_{\vee}^{E}$ здійснюється за допомогою САПР Xilinx ISE для ПЛІС типу FPGA фірми Xilinx, у яких апаратурні витрати вимірюються в LUT-елементах.

У табл. 5.6 наведені вимірювані в LUT-елементах значення функції $H_{\vee}^{E}(t,l)$ для ПЛІС XC3S1500 серії Spartan 3. На рис. 5.2, *а* вміст табл. 5.6 показаний у вигляді сімейства кривих $H_{\vee}^{E}(l)$ для різних значень t (кожна крива відповідає одному рядку табл. 5.6). На рис. 5.2, *б* наведене сімейство кривих $H_{\vee}^{E}(t)$ для різних значень l (кожна крива відповідає одному стовпцю табл. 5.6). Спостережуваний у межах досліджуваного діапазону аргументів лінійний (або близький до такого) характер кривих є наслідком регулярності внутрішньої структури ПЛІС FPGA і використовуваних VHDL-моделей.

Апроксимуємо вміст табл. 5.6 виразом

$$H_{\vee}^{A} = \frac{t \cdot l}{4}, \tag{5.12}$$

де $H_{\vee}^{A}$ – функція $H_{\vee}$, задана аналітично.

Табл. 5.7 містить відхилення $\Delta H_{\vee}(t,l)$ між аналітичними та експериментальними значеннями функції $H_{\vee}$. Значення в кожній комірці вимірюється в LUT-елементах і обчислюється за формулою

$$\Delta H_{\vee} = H_{\vee}^{A} - H_{\vee}^{E}, \tag{5.13}$$

де $H_{\vee}^{E}$ – значення з відповідної комірки табл. 5.6. У табл. 5.8 вміст табл. 5.7 виражений у відсотках від відповідних значень $H_{\vee}^{E}$ із табл. 5.6.

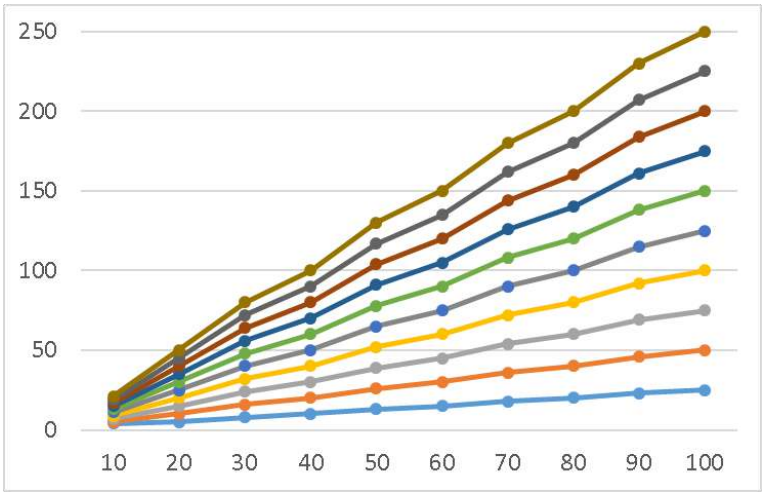
З даних таблиць видно, що формула (5.12) досить точно апроксимує функцію $H_{\vee}^{E}$, даючи відхилення не більш ніж у 5 LUT-елементів або (за винятком першого стовпця табл. 5.8) не більш ніж у 7%. Враховуючи, що значення в першому стовпці табл. 5.6 відносно невеликі, відхилення в межах 20% для них також є прийнятним. Умовимося використовувати формулу (5.12) для обчислення параметра $H_{\vee}$ у виразі (5.7).

Таблиця 5.6

Значення функції $H_{\vee}^{E}(t,l)$ для ПЛІС XC3S1500, LUT

$t \setminus l$	10	20	30	40	50	60	70	80	90	100
1	4	5	8	10	13	15	18	20	23	25
2	5	10	16	20	26	30	36	40	46	50
3	7	15	24	30	39	45	54	60	69	75
4	9	20	32	40	52	60	72	80	92	100
5	11	25	40	50	65	75	90	100	115	125
6	13	30	48	60	78	90	108	120	138	150
7	15	35	56	70	91	105	126	140	161	175
8	17	40	64	80	104	120	144	160	184	200
9	19	45	72	90	117	135	162	180	207	225
10	21	50	80	100	130	150	180	200	230	250

а)



б)

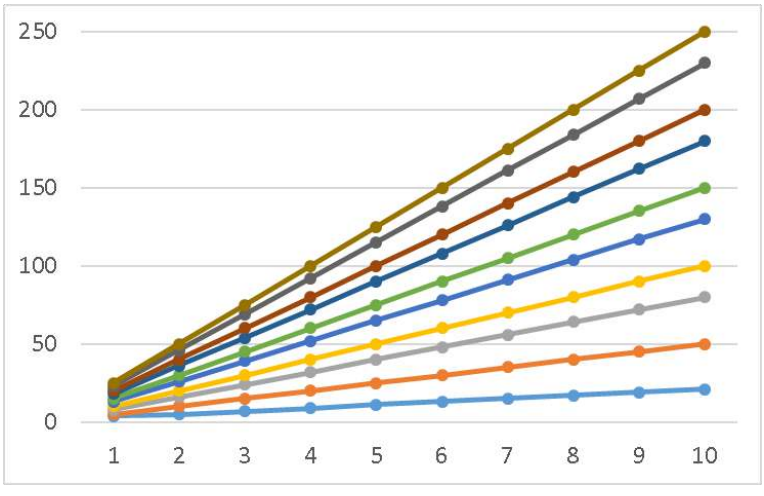


Рисунок 5.2 – Сімейства кривих для функцій $H_{\vee}(l)$ (*а*) та $H_{\vee}(t)$ (*б*)

Таблиця 5.7

Функція $\Delta H_{\vee}(t,l)$, LUT

$t \setminus l$	10	20	30	40	50	60	70	80	90	100
1	-1,5	0	-0,5	0	-0,5	0	-0,5	0	-0,5	0
2	0	0	-1	0	-1	0	-1	0	-1	0
3	0,5	0	-1,5	0	-1,5	0	-1,5	0	-1,5	0
4	1	0	-2	0	-2	0	-2	0	-2	0
5	1,5	0	-2,5	0	-2,5	0	-2,5	0	-2,5	0
6	2	0	-3	0	-3	0	-3	0	-3	0
7	2,5	0	-3,5	0	-3,5	0	-3,5	0	-3,5	0
8	3	0	-4	0	-4	0	-4	0	-4	0
9	3,5	0	-4,5	0	-4,5	0	-4,5	0	-4,5	0
10	4	0	-5	0	-5	0	-5	0	-5	0

Таблиця 5.8

Функція $\Delta H_{\vee}(t,l)$, %

$t \setminus l$	10	20	30	40	50	60	70	80	90	100
1	-37,5	0,0	-6,3	0,0	-3,8	0,0	-2,8	0,0	-2,2	0,0
2	0,0	0,0	-6,3	0,0	-3,8	0,0	-2,8	0,0	-2,2	0,0
3	7,1	0,0	-6,3	0,0	-3,8	0,0	-2,8	0,0	-2,2	0,0
4	11,1	0,0	-6,3	0,0	-3,8	0,0	-2,8	0,0	-2,2	0,0
5	13,6	0,0	-6,3	0,0	-3,8	0,0	-2,8	0,0	-2,2	0,0
6	15,4	0,0	-6,3	0,0	-3,8	0,0	-2,8	0,0	-2,2	0,0
7	16,7	0,0	-6,3	0,0	-3,8	0,0	-2,8	0,0	-2,2	0,0
8	17,6	0,0	-6,3	0,0	-3,8	0,0	-2,8	0,0	-2,2	0,0
9	18,4	0,0	-6,3	0,0	-3,8	0,0	-2,8	0,0	-2,2	0,0
10	19,0	0,0	-6,3	0,0	-3,8	0,0	-2,8	0,0	-2,2	0,0

Таким чином, з урахуванням (5.9) і (5.12), вираз (5.7) набуває наступного вигляду:

$$H_{КЛС} = \frac{q \cdot s}{10} + \frac{t \cdot l}{4}. \qquad (5.14)$$

Тут $H_{КЛС}$ вимірюється в LUT-елементах, параметри q, s, t, l є безрозмірними величинами.

У процесі дослідження функцій $H_{\&}^{E}$ та $H_{\vee}^{E}$ однією з умов було те, що кожний терм складається рівно з s кон'юнктивних компонентів, а кожне рівняння містить рівно l термів. Для реальних систем булевих рівнянь дані вимоги, у загальному випадку, не виконуються. Щоб мати можливість оцінки апаратурних витрат у довільній системі булевих рівнянь за допомогою виразу (5.14), параметр s повинен бути взятий рівним середньому арифметичному значень s усіх наявних термів, а параметр l – рівним середньому арифметичному значень l усіх рівнянь системи. Доведемо дане твердження експериментально.

Перетворимо VHDL-модель, що використана для дослідження функції $H_{\&}^{E}$ (лістинг А.7), таким чином, щоб число s кон'юнктивних компонентів у термах було різним, але його середнє значення для всіх термів дорівнювало 80. Побудуємо дві моделі: у першій моделі для однієї половини термів $s_1 = 60$, для другої половини $s_2 = 100$; у другій моделі для однієї половини термів $s_1 = 70$, для другої половини $s_2 = 90$. При цьому в обох моделях середнє значення $s = 80$ незалежно від кількості термів.

Дослідимо за допомогою даних моделей залежність $H_{\&}^{E}(q)$, порівнявши результати із вмістом стовпця $s = 80$ табл. 5.2. У табл. 5.9 перший стовпець містить значення q, для яких проведені дослідження; другий стовпець відповідає стовпцю $s = 80$ табл. 5.2; третій стовпець містить значення $H_{\&}^{E}(q)$, отримані за допомогою VHDL-моделі з параметрами $s_1 = 60$, $s_2 = 100$; четвертий стовпець містить значення $H_{\&}^{E}(q)$, отримані за допомогою VHDL-моделі з параметрами $s_1 = 70$, $s_2 = 90$.

Графічно вміст табл. 5.9 показаний на рис. 5.3. Близькість кривих дозволяє стверджувати, що усереднення значень s не приводить у рамках дослідженого діапазону аргументів до істотних відхилень функції $H_{\&}^{E}$ і дозволяє використовувати вираз (5.9) для оцінки апаратурних витрат на реалізацію кон'юнктивної частини систем булевих рівнянь із термами неоднакового розміру.

Таблиця 5.9

Залежності $H_{\&}^{E}(q)$ для систем рівнянь із усередненим значенням s, LUT

q	$H_{\&}^{E}$ $(s=80)$	$H_{\&}^{E}$ $(s_1=60,\ s_2=100)$	$H_{\&}^{E}$ $(s_1=70,\ s_2=90)$
10	115	109	122
20	238	248	237
30	276	270	281
40	324	315	329
50	383	371	376
60	462	482	446
70	534	516	544
80	593	579	581
90	758	688	685
100	826	825	805

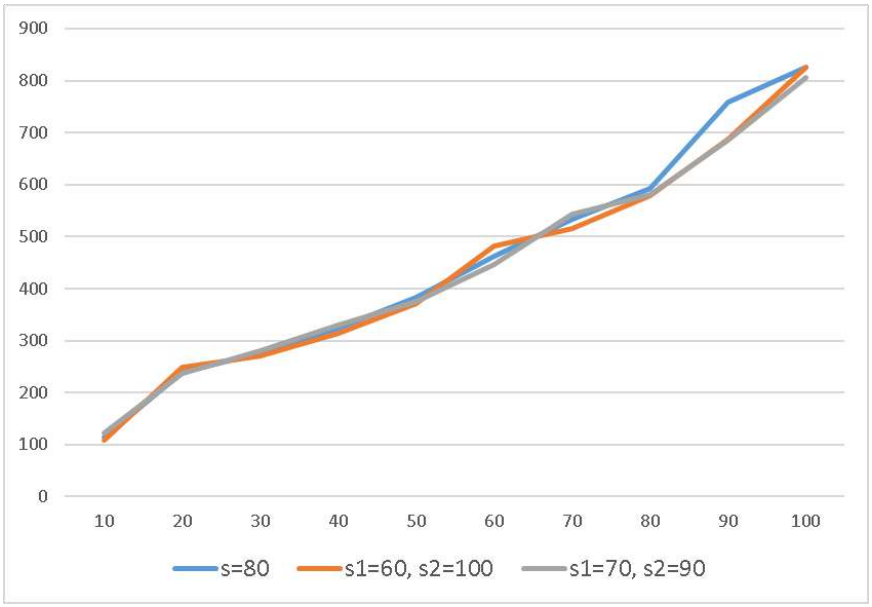


Рисунок 5.3 – Графічне представлення вмісту табл. 5.9

Проведемо аналогічні дослідження для параметра l. Перетворимо VHDL-модель, що використана для дослідження функції $H_{\vee}^{E}$ (лістинг А.9), таким чином, щоб число l термів у рівняннях було різним, але його середнє значення для всіх рівнянь дорівнювало 80. Побудуємо дві моделі: у першій моделі для однієї

половини рівнянь $l_1 = 60$, для другої половини $l_2 = 100$; у другій моделі для однієї половини рівнянь $l_1 = 70$, для другої половини $l_2 = 90$.

Дослідимо за допомогою даних моделей залежність $H_{\vee}^{E}(t)$, порівнявши результати із вмістом стовпця $l = 80$ табл. 5.6. У табл. 5.10 перший стовпець містить значення *t*, для яких проведені дослідження; другий стовпець відповідає стовпцю $l = 80$ табл. 5.6; третій стовпець містить значення $H_{\vee}^{E}(q)$, отримані за допомогою VHDL-моделі з параметрами $l_1 = 60$, $l_2 = 100$; четвертий стовпець містить значення $H_{\vee}^{E}(t)$, отримані за допомогою VHDL-моделі з параметрами $l_1 = 70$, $l_2 = 90$. Відзначимо, що для значень $t = 1$ дослідження не проводилися, оскільки при одному рівнянні в системі усереднення числа термів не має практичного сенсу.

Таблиця 5.10

Залежності $H_{\vee}^{E}(t)$ для систем рівнянь із усередненим значенням *l*, LUT

t	$H_{\vee}^{E}$ ($l = 80$)	$H_{\vee}^{E}$ ($l_1 = 60$, $l_2 = 100$)	$H_{\vee}^{E}$ ($l_1 = 70$, $l_2 = 90$)
2	40	40	41
3	60	55	59
4	80	80	82
5	100	95	100
6	120	120	123
7	140	135	141
8	160	160	164
9	180	175	182
10	200	200	205

Графічно вміст табл. 5.10 показаний на рис. 5.4. Близькість кривих дозволяє стверджувати, що усереднення значень *l* не приводить у рамках дослідженого діапазону аргументів до істотних відхилень функції $H_{\vee}^{E}$ і дозволяє використовувати вираз (5.12) для оцінки апаратурних витрат на реалізацію

диз'юнктивної частини систем булевих рівнянь, рівняння яких містять неоднакову кількість термів.

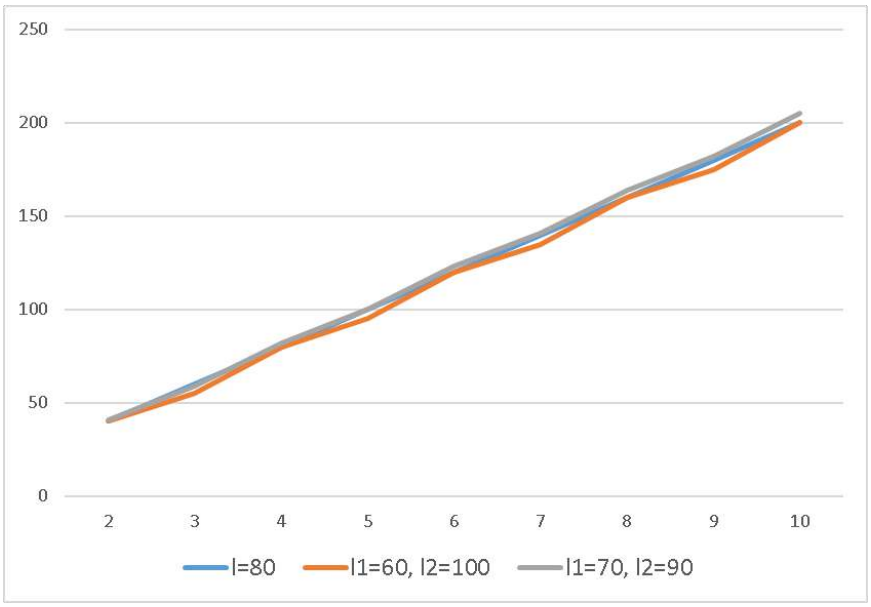


Рисунок 5.4 – Графічне представлення вмісту табл. 5.10

Відзначимо також, що для систем булевих рівнянь із параметрами q, s, l, t фактичні витрати апаратури можуть виявитися меншими за обчислені за формулою (5.14) за рахунок проведення мінімізації. Для урахування даного фактора введемо у вираз (5.14) спеціальний коефіцієнт $k_1 \in (0;1]$:

$$H_{КЛС} = k_1\left(H_{\&} + H_{\vee}\right) = k_1\left(\frac{q \cdot s}{10} + \frac{t \cdot l}{4}\right). \qquad (5.15)$$

При $k_1 = 1$ мінімізація не приводить до зниження апаратурних витрат; при $k_1 \to 0$ значення $H_{КЛС}$ прямує до нуля.

У подальших дослідженнях для визначення витрат апаратури на реалізацію комбінаційної схеми, що реалізує систему булевих рівнянь із параметрами q, s, t, l використовується функція (5.15).

5.2.2 Мультиплексор

Згідно до табл. 5.1, мультиплескор є частиною структур МПА з ОАП $U_1 - U_4$ і відсутній у канонічній структурі МПА. У МПА з ОАП мультиплексор

використовується в операційній частині для мультиплексування результату (рис. 3.1), а також може зустрічатися в комбінаційних схемах, що реалізують операції переходів (наприклад, рис. 2.12, 3.15, 3.20). Помітимо, що в обох випадках виходом мультиплексора є структурний код стану, розрядність якого зазвичай знаходиться у діапазоні від 4 біт (для МПА малої складності) до 10 і вище для МПА великої і надвеликої складності) [29]. При цьому число мультиплексованих напрямків для мультиплексора результату в операційній частині ОАП відповідає кількості комбінаційних схем в блоці ОЧ і може досягати декількох десятків.

Виконаємо дослідження апаратурних витрат H_{MX} на реалізацію r-розрядного мультиплексора з d напрямків, для чого скористаємося VHDL-моделлю, приклад якої представлений лістингом А.10. Модифікація моделі для різних параметрів r та d здійснюється шляхом зміни відповідних констант у розділі package.

Позначимо експериментально визначувану функцію H_{MX} символом H_{MX}^{E} і дослідимо її залежність від параметрів r і d. Визначимо наступні умови проведення експерименту:

1. Величини r і d змінюються незалежно одна від іншої в діапазоні $[2;20]$ із кроком 2.

2. Визначення чисельних значень $H_{\vee}^{E}$ здійснюється за допомогою САПР Xilinx ISE для ПЛІС типу FPGA фірми Xilinx, у яких апаратурні витрати виміряються в LUT-елементах.

У табл. 5.11 наведені значення функції $H_{MX}^{E}(r,d)$ для ПЛІС XC3S1500. Апроксимуємо вміст табл. 5.11 наступним виразом:

$$H_{MX}^{A} = r\left(d/2 + \lceil \log_2 d \rceil - 2\right), \tag{5.16}$$

де H_{MX}^{A} – функція H_{MX}, задана аналітично.

Відхилення між функціями H_{MX}^{A} та H_{MX}^{E} відображене в табл. 5.12, де значення кожної комірки визначається виразом

$$\Delta H_{MX} = H_{MX}^{A} - H_{MX}^{E}. \tag{5.17}$$

Таблиця 5.11

Значення функції $H_{MX}^{E}(r,d)$ для ПЛІС XC3S1500, LUT

$r \setminus d$	2	4	6	8	10	12	14	16	18	20
2	2	4	9	10	15	17	22	20	26	28
4	4	8	17	20	32	36	44	40	50	57
6	6	12	25	30	52	54	65	60	74	85
8	8	16	33	40	66	76	86	80	98	128
10	10	20	41	50	80	92	104	100	150	170
12	12	24	49	60	94	108	122	120	131	144
14	14	28	57	70	108	124	140	140	148	164
16	16	32	65	80	122	140	158	160	166	184
18	18	36	73	90	136	156	176	160	184	205
20	20	40	81	100	150	172	194	176	202	225

Таблиця 5.12

Функція $\Delta H_{MX}(r,d)$, LUT

$r \setminus d$	2	4	6	8	10	12	14	16	18	20
2	-2	0	-1	0	-1	-1	-4	0	-2	-2
4	-4	0	-1	0	-4	-4	-8	0	-2	-5
6	-6	0	-1	0	-10	-6	-11	0	-2	-7
8	-8	0	-1	0	-10	-12	-14	0	-2	-24
10	-10	0	-1	0	-10	-12	-14	0	-30	-40
12	-12	0	-1	0	-10	-12	-14	0	13	12
14	-14	0	-1	0	-10	-12	-14	0	20	18
16	-16	0	-1	0	-10	-12	-14	0	26	24
18	-18	0	-1	0	-10	-12	-14	20	32	29
20	-20	0	-1	0	-10	-12	-14	24	38	35

Аналіз табл. 5.12 дозволяє зробити наступні висновки.

1. У діапазоні $4 \le d \le 16$ функція (5.16) у більшості випадків дає невелике відхилення від експериментальних значень, значення якого при заданому d є фіксованим і майже не залежить від r.

2. При $d = 2$ функція (5.16) набуває нульового значення незалежно від r. Це пов'язане з недосконалістю формули (5.16) і може бути виправлене шляхом її модифікації (наприклад, приведенням до інтервального вигляду). Однак істотної

потреби в цьому немає, оскільки значення функції H_{MX}^{E} при $d=2$ відносно невеликі (див. табл. 5.11) і ними в більшості випадків можна зневажити. При необхідності точної оцінки апаратурних витрат при $d=2$ можна скористатися експериментальними значеннями з відповідного стовпця табл. 5.11.

3. При $d>18$ та $r>10$ значення функції H_{MX}, отримані аналітично, починають перевищувати експериментальні значення, що, імовірно, є результатом оптимізації схеми мультиплексора засобами САПР. У рамках табл. 5.12 таке перевищення становить близька 15 %, що дозволяє вважати використання функції (5.16) у даному діапазоні аргументів прийнятним.

Відзначимо, що в САПР Xilinx ISE є можливість налаштування параметрів синтезу, керуючих синтезом окремих об'єктів VHDL-описів. Зокрема, у властивостях процесу синтезу в розділі «HDL Options» присутній параметр «Mux Extraction». Установка даного параметра у значення «Yes» приводить до того, що САПР виявляє в VHDL-описі фрагменти, логіка роботи яких відповідає логіці роботи мультиплексора. При виявленні таких фрагментів САПР генерує якісь стандартизовані макроси (фрагменти VHDL-коду), що дозволяють реалізувати алгоритми мультиплексування засобами LUT-елементів найбільш оптимально (з погляду САПР).

Повторимо описаний вище експеримент із дослідження функції $H_{MX}^{E}(r,d)$ при включеній опції «Mux Extraction», залишивши інші умови експерименту незмінними. Результати, аналогічні наведеним у табл. 5.11, показані в табл. 5.13.

Аналіз обох таблиць показує, що в більшості випадків включення опції «Mux Extraction» дозволяє знизити витрати апаратури на реалізацію мультиплексора, хоча й незначно (в середньому на 12 % для дослідженого діапазону аргументів). На цій підставі умовимось далі використовувати експериментальні результати з табл. 5.13 замість табл. 5.11.

Таблиця 5.13

Значення функції $H_{MX}^{E}(r,d)$ при включеній опції «Mux Extraction»

$r \setminus d$	2	4	6	8	10	12	14	16	18	20
2	2	4	8	8	12	14	18	16	20	22
4	4	8	16	16	24	28	36	32	40	44
6	6	12	24	24	36	42	54	48	60	66
8	8	16	32	32	48	56	72	64	80	88
10	10	20	40	40	60	70	90	80	100	110
12	12	24	48	48	72	84	108	96	120	132
14	14	28	56	56	84	98	126	112	140	154
16	16	32	64	64	96	112	144	128	160	176
18	18	36	72	72	108	126	162	144	180	198
20	20	40	80	80	120	140	180	160	200	220

Апроксимуємо вміст табл. 5.13 наступним виразом:

$$\begin{aligned} H_{MX}^{A} &= \frac{r\,d}{2} + r - r\left(\lceil \log_2(d+1) \rceil - \lceil \log_2 d \rceil\right) = \\ &= r\left(\frac{d}{2} + 1 - \lceil \log_2(d+1) \rceil + \lceil \log_2 d \rceil\right). \end{aligned} \tag{5.18}$$

Відхилення (5.17) між функцією H_{MX}^{A}, що задається виразом (5.18), і функцією H_{MX}^{E}, що задається табл. 5.13, ілюструється табл. 5.14.

Таблиця 5.14

Функція $\Delta H_{MX}(r,d)$ при включеній опції «Mux Extraction», LUT

$r \setminus d$	2	4	6	8	10	12	14	16	18	20
2	0	0	0	0	0	0	-2	0	0	0
4	0	0	0	0	0	0	-4	0	0	0
6	0	0	0	0	0	0	-6	0	0	0
8	0	0	0	0	0	0	-8	0	0	0
10	0	0	0	0	0	0	-10	0	0	0
12	0	0	0	0	0	0	-12	0	0	0
14	0	0	0	0	0	0	-14	0	0	0
16	0	0	0	0	0	0	-16	0	0	0
18	0	0	0	0	0	0	-18	0	0	0
20	0	0	0	0	0	0	-20	0	0	0

Як можна бачити, результати функції (5.18) дають незначну розбіжність із експериментальними значеннями лише при $d = 14$, що пояснюється оптимізацією схеми вбудованими засобами САПР. Умовимось в подальших дослідженнях для визначення апаратурних витрат на реалізацію r-розрядного мультиплексора з d напрямків використовувати функцію (5.18).

5.2.3 Арифметико-логічні операції

Проведемо дослідження апаратурних витрат на реалізацію арифметико-логічних операцій, які можуть бути використані в якості операцій переходів. Арифметичні операції в рамках досліджень будемо вважати цілочисельними. Розглянемо наступні найбільш типові операції:

– додавання константи, віднімання константи, інкремент, декремент;

– множення на константу;

– ділення на константу;

– логічний зсув;

– інверсія;

– кон’юнкція з константою, диз'юнкція з константою, сума за модулем 2 з константою.

Витрати апаратури в даних вузлах будемо розглядати як функції від розрядності операнда r. Визначимо ті ж умови проведення експериментів, що й раніше: САПР Xilinx ISE 9.2i, ПЛІС XC3S1500 серії Spartan 3.

Додавання константи, віднімання константи, інкремент, декремент.

VHDL-модель для дослідження даних операцій представлена лістингом А.11. У даній моделі використані дві налагоджувальні константи: r, що дорівнює розрядності операнда і результату, та c, що дорівнює константі. Експериментально визначені апаратурні витрати на реалізацію функціональних блоків на основі даної моделі, виражені в LUT-елементах, наведені в табл. 5.15.

Пояснимо причину того, що значення апаратурних витрат на реалізацію операцій додавання з константою та інкремента, а також операцій вирахування константи і декремента об'єднані в одному рядку таблиці.

Таблиця 5.15

Витрати апаратури на реалізацію операцій на основі додавання, LUT

Операція \ r	2	4	6	8	10	12	14	16	18	20
Додавання константи, інкремент	1	3	6	9	12	12	14	16	18	20
Віднімання константи, декремент	1	3	6	9	13	12	14	16	18	20

Як показали дослідження, існує зв'язок між витратами апаратури на реалізацію операції підсумовування з константою і значенням константи. Справа в тому, що САПР не реалізує апаратно підсумовування з молодшими розрядами константи, які дорівнюють нулю. Наприклад, якщо $r = 20$ і константа $c = 240_{10} = 11110000_2$, у якій молодші чотири розряди дорівнюють нулям, то розрядність синтезованого суматора буде на 4 розряди меншою, ніж значення *r*. При $r = 10$ та $c = 512_{10} = 1000000000_2$ існує необхідність у підсумовуванні тільки найстарших розрядів операндів, внаслідок чого витрати апаратури на реалізацію VHDL-моделі становлять один LUT-елемент. З іншого боку, при $c = 1$ існує необхідність у підсумовуванні всіх розрядів операнда незалежно від їхньої кількості. В результаті при $r = 10$ та $c = 1_{10} = 0000000001_2$ витрати апаратури виявляються максимальними і становлять 12 LUT-елементів. Усе сказане стосується і операції віднімання: витрати на її реалізацію виявляються максимальними у випадку, коли в константі, що віднімається, молодший двійковий розряд дорівнює одиниці.

Помітимо, що при $c = 1$ суматор з константою реалізує функцію інкременту, а схема віднімання константи – функцію декременту. Внаслідок цього в рамках проведених досліджень найбільш зручним виявляється використання константи, що дорівнює одиниці. Одержувані в результаті VHDL-моделювання значення апаратурних витрат, по-перше, є максимально можливими для схеми підсумовування з константою, а по-друге, збігаються з витратами на реалізацію інкрементора. Те ж саме стосується операцій декременту і віднімання константи.

Таким чином, можна стверджувати, що витрати апаратури на реалізацію зазначених операцій *не перевищують* значення, наведені в табл. 5.15.

Аналіз вмісту табл. 5.15 дозволяє вважати, що в рамках дослідженого діапазону аргументів апаратурні витрати на реалізацію операцій додавання константи, віднімання константи, інкременту і декременту однакові і приблизно дорівнюють значенню *r*. Позначимо витрати на реалізацію даних операцій символом $H_{+,-}$ і апроксимуємо їх залежністю

$$H_{+,-} = r. \tag{5.19}$$

Множення на константу.

VHDL-модель, що реалізує операцію множення на константу, представлена лістингом А.12. Модель принципово відрізняється від моделі на лістингу А.11 тим, що константа-множник відсутня у моделі в явному вигляді і надходить до блоку множення через *r*-розрядний вхід «x2».

Подібний підхід припускає синтез операції множення в два способи: на базі LUT-елементів і на базі блоків множення, вбудованих у ПЛІС. Архітектура використовуваної при проведенні експериментів ПЛІС XC3S1500 включає 32 блоки для множення 18-розрядних операндів. Їхнім використанням при синтезі VHDL-проектів керує опція «Multiplier style» з розділу «HDL options» у властивостях процесу синтезу САПР Xilinx ISE, що дозволяє вибрати базис для реалізації операцій множення.

Позначимо витрати апаратури на реалізацію операції множення на константу символом H_{mul}. У табл. 5.16 наведені значення функції $H_{mul}(r)$ для різних базисів. При цьому в таблиці зазначені лише витрати LUT-елементів, які у випадку базису вбудованих блоків множення використовуються в незначній кількості для різних допоміжних цілей. Кількість самих блоків множення в таблиці не вказана.

Очевидно, що при наявності можливості використання вбудованих блоків множення їх використання зводить використання LUT-елементів ПЛІС до нуля. При реалізації операції множення в базисі LUT-елементів (у ПЛІС, що не мають

вбудованих блоків множення) витрати апаратури відповідно до першого рядка табл. 5.16 можуть бути апроксимовані наступним виразом:

$$H_{mul} = r^2 + \frac{r}{2}. \qquad (5.20).$$

Таблиця 5.16

Витрати апаратури на реалізацію операції множення, LUT

Базис \ r	2	4	6	8	10	12	14	16	18	20
LUT-елементи	3	16	39	72	113	164	220	287	362	447
Вбудовані блоки множення	3	0	0	0	0	0	0	0	35	49

Ділення на константу.

У мові VHDL операція ділення на довільне ціле число стандартними засобами мови не є синтезовною. Оператор « / » може бути синтезований лише у випадку, коли правий операнд є ступенем двійки. В роботах [75, 166], де приділяється значна увага VHDL-синтезу типових функціональних блоків цифрових пристроїв, питання синтезу операції ділення на довільне число обходиться стороною. Це говорить про те, що для даної операції в мові VHDL немає готових рішень, і її реалізація має проводитися розроблювачем самостійно.

Можливим рішенням є реалізація одного з відомих ітераційних алгоритмів ділення на базі деякого АЛП [104, 134]. Даний спосіб характеризується відносно низькою швидкодією, яка в ряді випадків може бути оптимізована застосуванням ряду спеціальних алгоритмів, що зазвичай приводять до збільшення апаратурних витрат [133, 134]. У той же час ділення на число, що є ступенем двійки, реалізується шляхом зсуву діленого на необхідну кількість розрядів праворуч і має примітивну схемну реалізацію з мінімальними витратами апаратури.

Усе це сприяє тому, що при проектуванні цифрових пристроїв замість операції ділення на довільне число намагаються використовувати операцію ділення на число, що є ступенем двійки. Цього підходу доцільно дотримуватися при формуванні множини операцій переходів в процесі алгебраїчного синтезу

МПА з ОАП. Дослідження апаратурних витрат на реалізацію операції цілочисельного ділення на число, що є ступенем двійки, збігається з дослідженням схем зсуву (див. нижче).

Логічний зсув.

Позначимо витрати апаратури на реалізацію операції логічного зсуву символом H_{sh}. Для дослідження функції $H_{sh}(r)$, де r – розрядність операнда, скористаємося VHDL-моделлю, що відповідає лістингу А.13.

Результати синтезу даної моделі показують, що в базисі ПЛІС операція логічного зрушення реалізується шляхом комутації сигналів без використання LUT-елементів незалежно від розрядності операнда і напрямку зсуву. Таким чином, функцію $H_{sh}(r)$, а також функцію $H_{div2}(r)$, яка виражає витрати апаратури на реалізацію r-розрядного цілочисельного ділення на константу, що дорівнює ступеню двійки, будемо вважати рівними нулю при будь-якому значенні та розрядності аргументу.

Інверсія.

VHDL-модель для дослідження апаратурних витрат на реалізацію r-розрядного інвертора аналогічна моделі на лістингу А.13, однак архітектурне тіло містить єдиний оператор

```
y <= not x;
```

Дослідження показали, що синтез інвертора в базисі ПЛІС XC3S1500 здійснюється без використання LUT-елементів, що дозволяє вважати апаратурні витрати H_{not} на його реалізацію рівними нулю.

Кон’юнкція з константою, диз'юнкція з константою, сума за модулем 2 з константою.

Позначимо витрати апаратури на реалізацію даних операцій символами H_{and}, H_{or} та H_{xor} відповідно. При синтезі VHDL-моделі, представленої листингом А.14, кожна із цих величин виявляється рівною r LUT-елементам, де r – розрядність результату.

Однак дана модель не є, строго кажучи, моделлю логічної операції з константою, оскільки використовує два операнди. Щодо цього більш правильною слід вважати модель, представлену лістингом А.15.

Дослідження показали, що при синтезі останньої моделі САПР Xilinx ISE не використовує LUT-елементи незалежно від використовуваної логічної операції (*and, or* або *xor*) і значення константи. Вбудований в САПР засіб «View Technology Schematic» дозволяє побачити, що САПР при синтезі використовує лише вбудовані інвертори разом із комутацією сигналів.

Проте, таку відмову від використання LUT-елементів можна віднести до особливостей використовуваної САПР і обраної ПЛІС. Будемо вважати, що в загальному випадку витрати на реалізацію розглянутих логічних операцій однакові і становлять *r* LUT-елементів:

$$H_{and} = H_{or} = H_{xor} = r. \qquad (5.21)$$

Проведені дослідження дозволяють зробити наступні висновки:

1. Витрати апаратури на реалізацію *r*-розрядних операцій додавання константи, віднімання константи, інкременту й декременту можна вважати однаковими і рівними *r* LUT-елементів.

2. Витрати на реалізацію *r*-розрядної операції множення на константу при використанні вбудованих у ПЛІС блоків множення дорівнюють нулю.

3. Витрати на реалізацію *r*-розрядної операції розподілу на константу, що є ступенем двійки, дорівнюють нулю.

4. Витрати на реалізацію *r*-розрядної операції інверсії дорівнюють нулю.

5. Витрати на реалізацію *r*-розрядних операцій кон’юнкції з константою, диз'юнкції з константою та суми за модулем 2 з константою однакові й становлять *r* LUT-елементів.

Узагальнимо дані результати, прийнявши вимірювані в LUT-елементах витрати апаратури $H_{ОП}$ на реалізацію будь-якої операції переходів в операційній частині ОАП рівними розрядності операнда *r*. Для забезпечення можливості аналізу апаратурних витрат у випадку ускладнення використовуваних операцій

переходів (наприклад, при використанні в складі однієї ОП декількох арифметико-логічних операцій) додамо коефіцієнт $k_2 \in (0;\infty)$, що дозволяє масштабувати значення $H_{ОП}$. Таким чином, функцію $H_{ОП}(r)$, результат якої вимірюється в LUT-елементах, задамо наступним виразом:

$$H_{ОП} = k_2 r . \tag{5.22}$$

5.2.4 Регістр

Регістр пам'яті є єдиною регістровою схемою як у канонічній структурі МПА, так і в пропонованих структурах МПА з ОАП. Його розрядність, що дорівнює розрядності R структурного коду стану, а також функціональні можливості однакові для кожної з даних структур. Відповідна VHDL-модель представлена лістингом А.16. Дослідження показали, що при використанні даної моделі витрати апаратури H_{RG} на реалізацію r-розрядного синхронного регістру з функцією скидання, виражені в LUT-елементах, чисельно дорівнюють розрядності регістру.

$$H_{RG} = r . \tag{5.23}$$

Відзначимо, що крім LUT елементів, при синтезі регістру використовуються тригери, що входять, поряд з LUT-елементами, до складу функціональних блоків ПЛІС типу FPGA.

5.3 Параметризація досліджуваних структур мікропрограмних автоматів

При формуванні аналітичних залежностей апаратурних витрат у досліджуваних структурах МПА пропонується використовувати ряд параметрів, зібраних у табл. 5.17. Пояснимо деякі з них.

k_1 – коефіцієнт мінімізації апаратурних витрат, уведений в п. 5.2.1. Ступінь зменшення апаратурних витрат на реалізацію КЛС індивідуальна в кожному конкретному випадку, однак у проведених дослідженнях значення k_1 вибирається

однаковим для всіх КЛС досліджуваної структури МПА. Таким чином, даний коефіцієнт слід розглядати як коефіцієнт мінімізації сумарних витрат апаратури в усіх схемах досліджуваного МПА, що будуються за системою булевих рівнянь. При необхідності проведення більш точних досліджень подібні коефіцієнти можуть бути введені окремо для кожної КЛС.

Таблиця 5.17

Параметри МПА

Позначення	Опис
M	кількість станів
R	розрядність структурного коду стану
B	кількість переходів
N	кількість мікрооперацій
L	кількість логічних умов
N_d	кількість комбінаційних схем у блоці ОЧ операційного автомата переходів
k_1	коефіцієнт мінімізації КЛС, $k_1 \in (0;1]$
k_2	коефіцієнт складності схем КС у блоці ОЧ, $k_2 \in (0;\infty)$
k_3	частка переходів, що реалізуються за допомогою операцій переходів, від загального числа переходів B, $k_3 \in [0;1]$
k_4	частка умовних переходів від загального числа переходів B, $k_4 \in [0;1]$
k_5	середня частка термів в одному рівнянні системи булевих рівнянь від загального числа термів, використовуваних у системі, $k_5 \in (0;1)$
k_6	коефіцієнт ефективності використання базису блокової пам'яті ПЛІС FPGA стосовно базису LUT-елементів, $k_5 \in (0;\infty)$

R – розрядність адреси мікрокоманди, яка в загальному випадку може бути різною для канонічного МПА та МПА з ОАП за рахунок використання транзитних станів (див. п. 4.2.5) або методу примусового збільшення розрядності кодів станів (див. п. 4.2.7). Однак, оскільки додавання транзитних станів не

обов'язково приводить до збільшення *R*, в рамках проведених досліджень дана величина вважається однаковою для всіх досліджуваних структур.

k_2 – коефіцієнт масштабування апаратурних витрат на реалізацію однієї операції переходів, уведений в п. 5.2.3. При дослідженні структур МПА з ОАП значення k_2 береться однаковим для кожної ОП, що входить в операційний автомат переходів, будучи, таким чином, усереднюючим коефіцієнтом масштабування. При необхідності проведення більш точних досліджень для кожної ОП може бути задане своє значення k_2.

k_3 – коефіцієнт, який дорівнює відношенню числа переходів МПА з ОАП, реалізованих за допомогою операцій переходів, до загального числа *B* переходів автомата. Відповідно, число переходів автомата, реалізованих канонічним способом, дорівнює $(1-k_3)B$. При $k_3=1$ всі переходи автомата реалізуються за допомогою множини операцій переходів. При $k_3=0$ всі переходи реалізуються в канонічний спосіб, і МПА з ОАП вироджується в МПА з канонічною структурою.

k_4 – коефіцієнт, який дорівнює відношенню кількості умовних переходів до загального числа *B* переходів автомата. При цьому число безумовних переходів дорівнює $(1-k_4)B$. При $k_4=1$ в автоматі відсутні безумовні переходи, при $k_4=0$ – умовні.

k_5 – коефіцієнт, який дорівнює відношенню середньої кількості термів в одному рівнянні системи булевих рівнянь до загального числа термів, використовуваних у системі. Загальне число термів залежить від того, що виступає в якості аргументів системи рівнянь. Наприклад, у системі булевих рівнянь, що реалізує схему СФМО, загальне число термів у випадку автомата Мілі не перевищує числа переходів *B*, у випадку автомата Мура – числа станів *M*. Даний коефіцієнт дозволяє визначити для заданої КЛС значення параметра *l* у виразі (5.15). При $k_5 \to 1$ диз'юнктивна частина системи кожного рівняння є максимально складною, при $k_5 \to 0$ – максимально простою. Відзначимо, що при $k_5=1$ кожне рівняння містить усі можливі терми системи, вироджуючись у

булеву функцію «константа 1». Аналогічно при $k_5 = 0$ в рівняннях системи відсутні булеві терми, що дозволяє зіставити кожному рівнянню булеву функцію «константа 0».

У загальному випадку для різних систем булевих рівнянь значення k_5 різне. Однак у рамках проведених досліджень даний коефіцієнт береться однаковим для кожного структурного блоку, синтезованого за системою булевих рівнянь, будучи, таким чином, усередненням відповідних коефіцієнтів для різних КЛС. При необхідності проведення більш точних досліджень кожній КЛС досліджуваної структури МПА може бути зіставлений власний коефіцієнт.

k_6 – коефіцієнт, що дозволяє виразити в LUT-елементах апаратурні витрати $H_{КЛС}^{БП}$ на реалізацію КЛС (блоку, синтезованого по системі булевих рівнянь), при використанні базису блокової пам'яті (Block RAM, BRAM) ПЛІС типу FPGA [94, 126, 162]. Нехай $H_{КЛС}$ – витрати апаратури, що визначаються виразом (5.15) і вимірюються в LUT-елементах. Тоді

$$H_{КЛС}^{БП} = k_6 H_{КЛС} . \qquad (5.24)$$

Значення $k_6 < 1$ слід вибирати в тому випадку, якщо використовувана ПЛІС містить велику кількість вбудованих блоків пам'яті, що не використовуються в рамках поточного проекту. У цьому випадку реалізація КЛС у базисі блокової пам'яті виявляється більш кращою, ніж на LUT-елементах, оскільки відносна цінність використовуваних при цьому ресурсів ПЛІС виявляється нижчою. При $k_6 = 1$ цінність даних базисів для використовуваної ПЛІС вважається однаковою. При $k_6 > 1$ реалізація КЛС у базисі LUT-елементів вважається більш кращою у порівнянні з базисом блокової пам'яті.

Застосування коефіцієнту k_6 можливе в тому випадку, якщо:

1. Використовувана ПЛІС містить достатню кількість блокової пам'яті.

2. Число вхідних сигналів синтезованої КЛС не перевищує числа адресних входів блоку пам'яті, яке в сучасних ПЛІС FPGA фірми Xilinx не перевищує 16 [162]. Наприклад, у МПА Мура схема СФМО має число вхідних сигналів, що

дорівнює розрядності структурного коду стану R, і при $R \le 16$ (а також при достатній розрядності рядка даних для реалізації множини мікрооперацій) може бути синтезована в базисі блокової пам'яті. У МПА Мілі на входи схеми СФМО додатково подаються L сигналів логічних умов, що при середніх значеннях L ($L = 30$, [29]) приводить до неможливості використання блокової пам'яті ПЛІС для синтезу СФМО.

5.4 Формування аналітичних залежностей апаратурних витрат для мікропрограмного автомата з канонічною структурою та мікропрограмного автомата з операційним автоматом переходів

Обмежимось в рамках дисертаційної роботи аналізом ефективності структур МПА з ОАП U_1 та U_2, провівши дослідження окремо для автоматів Мілі і Мура.

Канонічний МПА

Апаратурні витрати H^{U_K} в МПА з канонічною структурою визначаються виразом (5.2). Виразимо доданки, що розташовані у правій частині (5.2), через параметри, зазначені в табл. 5.16.

Схема СФП є комбінаційною схемою, на входи якої подаються R розрядів структурного коду поточного стану та L розрядів логічних умов, а виходами є R розрядів структурного коду стану переходу. Це дозволяє використовувати для визначення величини $H^{U_K}_{СФП}$ вираз (5.15), у якому кількість різних термів q дорівнює числу переходів автомата B, число рівнянь t дорівнює числу вихідних сигналів R, середнє число термів у рівнянні

$$l = k_5 q = k_5 B. \qquad (5.25)$$

Величина s, що дорівнює середньому числу кон’юнктивних компонентів в одному термі, визначається виразом

$$s = (1 - k_4)R + k_4(R + L_1), \qquad (5.26)$$

де L_1 – середнє число логічних умов, що перевіряються в одному умовному переході автомата. Визначимо приблизно L_1 як половину від середнього числа умовних переходів B_1, здійснюваних з одного стану автомата. Значення B_1, у свою чергу, визначимо як відношення загальної кількості умовних переходів, яка дорівнює k_4B, до кількості станів, з яких здійснюються умовні переходи, яка дорівнює $M-(1-k_4)B$:

$$B_1 = \frac{k_4 B}{M-(1-k_4)B}. \tag{5.27}$$

Тоді

$$L_1 = \frac{B_1}{2} = \frac{k_4 B}{2(M-(1-k_4)B)}, \tag{5.28}$$

$$s = (1-k_4)R + k_4\left(R + \frac{k_4 B}{2(M-(1-k_4)B)}\right). \tag{5.29}$$

В результаті вираз (5.15) для визначення величини $H_{СФП}^{U_K}$ набуває наступного вигляду:

$$H_{СФП}^{U_K} = k_1\left(\frac{B\left((1-k_4)R + k_4\left(R + \frac{k_4 B}{2(M-(1-k_4)B)}\right)\right)}{10} + \frac{k_5 BR}{4}\right). \tag{5.30}$$

Параметр $H_{РП}^{U_K}$, що виражає витрати LUT-елементів на реалізацію регістру пам'яті, визначається виразом (5.23), у якому аргумент r дорівнює розрядності структурного коду стану R:

$$H_{РП}^{U_K} = R. \tag{5.31}$$

Схема формування мікрооперацій, як і схема СФП, являє собою КЛС, що дозволяє використовувати для визначення величини $H_{СФМО}^{U_K}$ вираз (5.15). Якщо синтезований МПА є автоматом Мура, число вхідних сигналів СФМО дорівнює R, число вихідних сигналів дорівнює N. Оскільки в автоматі Мура вихідні сигнали залежать тільки від поточного стану, число q внутрішніх термів схеми СФМО у

виразі (5.15) дорівнює числу станів *M*, а довжина *s* кожного терма постійна і дорівнює *R*. Число *t* рівнянь системи дорівнює числу мікрооперацій *N*, а середнє число термів в одному рівнянні $l = k_5 q = k_5 M$. При цьому вираз для визначення $H_{СФМО}^{U_K}$ матиме наступний вигляд:

$$H_{СФМО}^{U_K} = k_1 \left(\frac{MR}{10} + \frac{k_5 MN}{4} \right). \tag{5.32}$$

Той факт, що в автоматі Мура число вхідних сигналів схеми СФМО відносно невелике, дозволяє синтез СФМО в базисі блокової пам'яті ПЛІС FPGA. Дана можливість може бути врахована шляхом множення виразу (5.32) на коефіцієнт k_6:

$$H_{СФМО}^{U_K} = k_1 k_6 \left(\frac{MR}{10} + \frac{k_5 MN}{4} \right). \tag{5.33}$$

Якщо синтезований МПА є автоматом Мілі, у ньому може бути застосований принцип спільного використання булевих термів [29, 31], відповідно до якого схема СФМО не має власної кон’юнктивної частини, використовуючи в диз'юнктивній частині множину термів, реалізовану в схемі СФП. Внаслідок цього у (5.15) для $H_{СФМО}^{U_K}$ перший доданок, відповідний до витрат апаратури на реалізацію кон’юнктивної частини, дорівнює нулю, а в другому доданку аргумент *t* дорівнює числу мікрооперацій *N*, аргумент *l* визначається виразом (5.25):

$$H_{СФМО}^{U_K} = k_1 \frac{k_5 BN}{4}. \tag{5.34}$$

З урахуванням (5.30), (5.31) та (5.34) вираз (5.2) для випадку автомата Мілі набуває наступного вигляду:

$$\begin{aligned} H^{U_K} &= H_{СФП}^{U_K} + H_{РП}^{U_K} + H_{СФМО}^{U_K} = \\ &= k_1 \left(\frac{B\left((1-k_4)R + k_4 \left(R + \frac{k_4 B}{2(M - (1-k_4)B)} \right) \right)}{10} + \frac{k_5 BR}{4} \right) + \\ &+ R + k_1 \frac{k_5 BN}{4}. \end{aligned} \tag{5.35}$$

Аналогічний вираз для автомата Мура, з урахуванням (5.30), (5.31) та (5.33), має вигляд:

$$H^{U_K} = H_{СФП}^{U_K} + H_{РП}^{U_K} + H_{СФМО}^{U_K} =$$
$$= k_1\left(\frac{B\left((1-k_4)R + k_4\left(R + \frac{k_4 B}{2(M-(1-k_4)B)}\right)\right)}{10} + \frac{k_5 BR}{4}\right) + \tag{5.36}$$
$$+ R + k_1 k_6\left(\frac{MR}{10} + \frac{k_5 MN}{4}\right).$$

МПА з ОАП U_1

Розглянемо МПА з ОАП зі структурою U_1 (рис. 3.4, *а*), для якого витрати апаратури визначаються виразом (5.3). У даній структурі, незалежно від того, чи є МПА автоматом Мілі або автоматом Мура, на вхід Z-підсхеми надходять сигнали логічних умов і код поточного стану. Отже, кон'юнктивна частина Z-підсхеми збігається з кон'юнктивною частиною схеми СФП канонічного МПА, для якої витрати апаратури визначаються першим доданком виразу (5.30). У диз'юнктивній частині Z-підсхеми параметр *l*, як і для СФП канонічного МПА, визначається виразом (5.25), а кількість формованих рівнянь *t* дорівнює розрядності коду ОП $R_Z = \lceil \log_2 N_d \rceil$. Таким чином, вираз для визначення величини $H_Z^{U_1}$ матиме наступний вигляд:

$$H_Z^{U_1} = k_1\left(\frac{B\left((1-k_4)R + k_4\left(R + \frac{k_4 B}{2(M-(1-k_4)B)}\right)\right)}{10} + \frac{k_5 B\lceil \log_2 N_d \rceil}{4}\right). \tag{5.37}$$

Витрати апаратури в блоці ОЧ $H_{ОЧ}^{U_1}$ визначимо як суму наступних величин:

$$H_{ОЧ}^{U_1} = H_{ОП}^{ОЧ} + H_{КЛС}^{ОЧ} + H_{МХ}^{ОЧ}, \tag{5.38}$$

де $H_{ОП}^{ОЧ}$ – витрати на реалізацію множини операцій переходів; $H_{КЛС}^{ОЧ}$ – витрати на реалізацію підмножини переходів, реалізованих канонічним способом, $H_{МХ}^{ОЧ}$ – витрати на реалізацію мультиплексора результату.

Величину $H_{ОП}^{ОЧ}$, з урахуванням (5.22), визначимо так:

$$H_{ОП}^{ОЧ} = k_2 R(N_d - 1). \tag{5.39}$$

Якщо в МПА з ОАП усі без винятку переходи реалізуються за допомогою операцій переходів, віднімання одиниці з N_d не потрібне. Однак, оскільки витрати на одну ОП відносно невеликі, умовимось використовувати вираз (5.39) незалежно від того, чи є в МПА з ОАП переходи, реалізовані в канонічний спосіб.

При визначенні $H_{КЛС}^{ОЧ}$ врахуємо, що на входи КЛС, яка реалізує в МПА з ОАП підмножину переходів за системою булевих рівнянь, подаються ті ж сигнали, що й на входи Z-підсхеми. Отже, у якості кон'юнктивної частини тут у загальному випадку може бути використана множина термів з кон'юнктивної частини Z-підсхеми. Таким чином, власні витрати на реалізацію кон'юнктивної частини даної КЛС (перший доданок у виразі (5.15)) дорівнюють нулю. Витрати на реалізацію диз'юнктивної частини визначаються другим доданком у виразі (5.15), де число рівнянь t дорівнює R, а параметр l визначається за формулою (5.25) з тим застереженням, що число переходів береться рівним не значенню B, а значенню $(1-k_3)B$ (числу переходів, реалізованих у МПА з ОАП канонічним способом). В результаті одержуємо наступний вираз для визначення $H_{КЛС}^{ОЧ}$:

$$H_{КЛС}^{ОЧ} = k_1(1-k_3)k_5\frac{BR}{4}. \tag{5.40}$$

Відзначимо, що якщо всі переходи автомата реалізуються за допомогою множини операцій переходів, коефіцієнт $k_3 = 1$, що робить величину $H_{КЛС}^{ОЧ}$ рівною нулю.

Витрати $H_{МХ}^{ОЧ}$ на реалізацію мультиплексора результату визначаються виразом (5.18), у якому аргументу r відповідає розрядність структурного коду стану R, аргументу d – параметр N_d:

$$H_{MX}^{ОЧ} = R\left(\frac{N_d}{2} + 1 - \lceil \log_2(N_d + 1) \rceil + \lceil \log_2 N_d \rceil\right). \quad (5.41)$$

Таким чином, з урахуванням (5.39)-(5.41), вираз (5.38) набуває наступного вигляду:

$$H_{ОЧ}^{U_1} = k_2 R(N_d - 1) + k_1(1 - k_3)k_5 \frac{BR}{4} + R\left(\frac{N_d}{2} + 1 - \lceil \log_2(N_d + 1) \rceil + \lceil \log_2 N_d \rceil\right). \quad (5.42)$$

Витрати на реалізацію регістру пам'яті, як і у випадку канонічного МПА, визначаються виразом (5.23), у якому $r = R$:

$$H_{РП}^{U_1} = R. \quad (5.43)$$

Витрати в блоці СФМО визначаються аналогічно канонічному МПА. Якщо в канонічному МПА Мілі кон’юнктивна частина СФМО є загальною зі схемою СФП, то в автоматі Мілі зі структурою U_1 вона є загальною із Z- підсхемою. Отже, величина $H_{СФМО}^{U_1}$ визначається тільки витратами в диз'юнктивній частині СФМО і дорівнює величині $H_{СФМО}^{U_K}$, яка визначається виразом (5.34). У МПА Мура зі структурою U_1 схема СФМО також ідентична однойменному блоку канонічного МПА Мура, що дозволяє визначати витрати апаратури на її реалізацію виразом (5.33).

З урахуванням отриманих виразів витрати апаратури H^{U_1} для автоматів Мілі і Мура задаються виразами (5.44) та (5.45) відповідно.

$$\begin{aligned} H^{U_1} &= H_{ОЧ}^{U_1} + H_{РП}^{U_1} + H_Z^{U_1} + H_{СФМО}^{U_1} = \\ &= \left(k_2 R(N_d - 1) + k_1(1 - k_3)k_5 \frac{BR}{4} + R\left(\frac{N_d}{2} + 1 - \lceil \log_2(N_d + 1) \rceil + \lceil \log_2 N_d \rceil\right)\right) + \\ &+ R + k_1\left(\frac{B\left((1 - k_4)R + k_4\left(R + \frac{k_4 B}{2(M - (1 - k_4)B)}\right)\right)}{10} + \frac{k_5 B \lceil \log_2 N_d \rceil}{4}\right) + \\ &+ k_1 \frac{k_5 BN}{4}; \end{aligned} \quad (5.44)$$

$$H^{U_1} = H_{ОЧ}^{U_1} + H_{РП}^{U_1} + H_Z^{U_1} + H_{СФМО}^{U_1} =$$
$$= \left(k_2 R(N_d - 1) + k_1(1-k_3)k_5 \frac{BR}{4} + R\left(\frac{N_d}{2} + 1 - \lceil \log_2(N_d + 1) \rceil + \lceil \log_2 N_d \rceil \right) \right) +$$
$$+ R + k_1 \left(\frac{B\left((1-k_4)R + k_4\left(R + \frac{k_4 B}{2(M - (1-k_4)B)} \right) \right)}{10} + \frac{k_5 B \lceil \log_2 N_d \rceil}{4} \right) + \quad (5.45)$$
$$+ k_1 k_6 \left(\frac{MR}{10} + \frac{k_5 MN}{4} \right).$$

МПА з ОАП U_2

Апаратурні витрати в МПА з ОАП зі структурою U_2 (рис. 3.6), визначаються виразом (5.4).

У МПА Мілі зі структурою U_2 в кон'юнктивній частині СФМО формується повна множина термів, відповідних до переходів автомата. Це визначає витрати в кон'юнктивній частині СФМО рівними витратам у кон'юнктивній частині СФП канонічного МПА (перший доданок у виразі (5.30)). Диз'юнктивна частина СФМО, як і в канонічному МПА Мілі, визначається виразом (5.34). Тоді:

$$H_{СФМО}^{U_2} = k_1 \left(\frac{B\left((1-k_4)R + k_4\left(R + \frac{k_4 B}{2(M - (1-k_4)B)} \right) \right)}{10} + \frac{k_5 BN}{4} \right). \quad (5.46)$$

Витрати в блоці ОЧ дорівнюють аналогічним витратам у структурі U_1 і визначаються виразом (5.42). Відзначимо, що КЛС, яка реалізує підмножину переходів канонічним способом, не має власної кон'юнктивної частини, використовуючи множину термів, що формуються в схемі СФМО.

Витрати на реалізацію регістру пам'яті, як і у випадку структури U_1, визначаються виразом (5.43).

Z-підсхема в структурі U_2 має R вхідних розрядів, що дозволяє виконати її синтез як у базисі LUT-елементів, так і в базисі блокової пам'яті. Для Z-підсхеми у виразі (5.15) $q = M$, $s = R$, $l = k_5 M$, $t = \lceil \log_2 N_d \rceil$:

$$H_Z^{U_2} = k_1 k_6 \left(\frac{MR}{10} + \frac{k_5 M \lceil \log_2 N_d \rceil}{4} \right). \quad (5.47)$$

Таким чином, у випадку автомата Мілі вираз (5.4) матиме наступний вигляд:

$$\begin{aligned} H^{U_2} &= H_{ОЧ}^{U_2} + H_{РП}^{U_2} + H_Z^{U_2} + H_{СФМО}^{U_2} = \\ &= \left(k_2 R(N_d - 1) + k_1 (1 - k_3) k_5 \frac{BR}{4} + R \left(\frac{N_d}{2} + 1 - \lceil \log_2 (N_d + 1) \rceil + \lceil \log_2 N_d \rceil \right) \right) + \\ &+ R + k_1 k_6 \left(\frac{MR}{10} + \frac{k_5 M \lceil \log_2 N_d \rceil}{4} \right) + \\ &+ k_1 \left(\frac{B\left((1 - k_4) R + k_4 \left(R + \frac{k_4 B}{2(M - (1 - k_4) B)} \right) \right)}{10} + \frac{k_5 BN}{4} \right); \end{aligned} \quad (5.48)$$

В автоматі Мура зі структурою U_2 на входи СФМО і Z-підсхеми надходить тільки *R*-розрядний структурний код стану, в наслідок чого дані блоки мають загальну кон'юнктивну частину. Отже, витрати в диз'юнктивній частині СФМО можуть бути визначені виразом (5.33), у якому перший доданок, відповідний до витрат у кон'юнктивній частині, дорівнює нулю:

$$H_{СФМО}^{U_2} = k_1 k_6 \frac{k_5 MN}{4}. \quad (5.49)$$

Схема КЛС у складі блоку ОЧ реалізує канонічним способом підмножину переходів, число яких дорівнює $(1 - k_3)B$ (див. п. 5.3). На її входи надходить, крім коду поточного стану, множина сигналів логічних умов. Внаслідок цього дана КЛС не може використовувати терми з кон'юнктивної частини СФМО та Z-підсхеми і має власні кон'юнктивну та диз'юнктивну частини, сумарні витрати в яких є частиною сумарних витрат $H_{СФП}^{U_K}$ у схемі СФП канонічного МПА (вираз (5.30)), визначеною коефіцієнтом $(1 - k_3)$:

$$H_{КЛС}^{ОЧ} = (1-k_3)H_{СФП}^{U_K} =$$

$$= k_1(1-k_3)\left(\frac{B\left((1-k_4)R + k_4\left(R + \frac{k_4 B}{2(M-(1-k_4)B)}\right)\right)}{10} + \frac{k_5 BR}{4} \right). \quad (5.50)$$

З урахуванням (5.50), у випадку автомата Мура витрати апаратури в блоці ОЧ визначаються за аналогією з (5.42) в такий спосіб:

$$H_{ОЧ}^{U_2} = H_{ОП}^{ОЧ} + H_{МХ}^{ОЧ} + H_{КЛС}^{ОЧ} =$$

$$= k_2 R(N_d - 1) + R\left(\frac{N_d}{2} + 1 - \lceil \log_2(N_d + 1)\rceil + \lceil \log_2 N_d \rceil\right) + \quad (5.51)$$

$$+ k_1(1-k_3)\left(\frac{B\left((1-k_4)R + k_4\left(R + \frac{k_4 B}{2(M-(1-k_4)B)}\right)\right)}{10} + \frac{k_5 BR}{4} \right).$$

Вважаючи, що витрати в регістрі пам'яті $H_{РП}^{U_2} = R$, одержуємо вираз для визначення величини H^{U_2} у випадку автомата Мура:

$$H^{U_2} = H_{ОЧ}^{U_2} + H_{РП}^{U_2} + H_Z^{U_2} + H_{СФМО}^{U_2} =$$

$$= k_2 R(N_d - 1) + R\left(\frac{N_d}{2} + 1 - \lceil \log_2(N_d + 1)\rceil + \lceil \log_2 N_d \rceil\right) +$$

$$+ k_1(1-k_3)\left(\frac{B\left((1-k_4)R + k_4\left(R + \frac{k_4 B}{2(M-(1-k_4)B)}\right)\right)}{10} + \frac{k_5 BR}{4} \right) + \quad (5.52)$$

$$+ R + k_1 k_6\left(\frac{MR}{10} + \frac{k_5 M\lceil \log_2 N_d \rceil}{4}\right) + k_1 k_6 \frac{k_5 MN}{4}.$$

Відзначимо, що у виразах (5.35), (5.36), (5.44), (5.45), (5.48) та (5.52) відсутній аргумент *L*, який дорівнює загальній кількості логічних умов, що надходять на вхід МПА. При використанні ПЛІС типу FPGA даний параметр не виявляє безпосереднього впливу на апаратурні витрати, однак він повинен враховуватися при виборі корпусу (package) мікросхеми.

5.5 Дослідження ефективності мікропрограмного автомата з операційним автоматом переходів

5.5.1 Умови проведення досліджень

Ефективність структури U_1 у порівнянні з канонічним МПА, відповідно до (5.1), визначається наступним виразом:

$$E^{U_1} = \frac{H^{U_K}}{H^{U_1}}, \tag{5.53}$$

де H^{U_K} та H^{U_1} визначаються у випадку автомата Мілі виразами (5.35) та (5.44) відповідно, у випадку автомата Мура – виразами (5.36) та (5.45) відповідно. Ефективність структури U_2 визначається виразом

$$E^{U_2} = \frac{H^{U_K}}{H^{U_2}}, \tag{5.54}$$

де H^{U_2} визначається виразами (5.48) та (5.52) для автоматів Мілі й Мура відповідно.

Умовимось розглядати чотири умовні класи складності автоматів – малої, середньої, великої і надвеликої складності, розуміючи під кожним класом деяку сукупність значень аргументів *M*, *R* та *B*, що визначаються функцією переходів автомата (табл. 5.18). Параметр *N* відсутній у табл. 5.18, оскільки не належить до функції переходів. Параметр *L*, хоча й визначається функцією переходів, також не входить у таблицю через його відсутність у виразах (5.35), (5.36), (5.44), (5.45), (5.48) та (5.52).

Таблиця 5.18

Класи складності МПА

Клас складності	*M*	*R*	*B*
1 (мала)	100	7	200
2 (середня)	500	9	1000
3 (велика)	1000	10	2000
4 (надвелика)	2000	11	4000

Для даних класів складності виконаємо дослідження впливу на ефективність E^{U_1} та E^{U_2} окремих аргументів з табл. 5.17, що не є визначальними для класу складності МПА, а саме: N, N_d, $k_1 - k_6$. В процесі досліджень значення незмінних аргументів визначимо в такий спосіб:

– для аргументів M, R, B значення відповідають прийнятим для відповідного класу складності МПА (табл. 5.18);

– $N = 100$;

– $N_d = 6$: для реалізації всієї множини переходів використовуються 5 КС, що відповідають операціям переходів, і комбінаційна схема, що реалізує частину переходів канонічним способом;

– $k_1 = 0{,}8$: передбачається, що в кожній КЛС ефект від мінімізації будь-якої системи булевих рівнянь становить 20 % від розрахункових значень апаратурних витрат за формулою (5.15);

– $k_2 = 1$: передбачається, що в блоці ОЧ використовуються операції переходів, для яких витрати LUT-елементів чисельно дорівнюють розрядності структурного коду стану автомата;

– $k_3 = 0{,}9$: передбачається, що 90 % усіх переходів автомата реалізуються за допомогою операцій переходів, і лише 10 % переходів реалізуються канонічним способом за системою булевих рівнянь;

– $k_4 = 0{,}75$: при даному значенні для всіх класів складності МПА з табл. 5.18 число станів, з яких здійснюються умовні переходи, дорівнює половині від загального числа станів;

– $k_5 = 0{,}5$: кожна булева функція, що формується будь-яким структурним блоком МПА, синтезованим за системою булевих рівнянь, набуває одиничне значення в середньому на половині всієї множини наборів аргументів;

– $k_6 = 0{,}5$: використання базису вбудованих блоків пам'яті ПЛІС типу FPGA замість базису LUT-елементів приводить до економії половини апаратурних витрат у логічній схемі автомата.

Незважаючи на те, що канонічний МПА і еквівалентний йому МПА з ОАП у деяких випадках можуть мати різні значення однойменних параметрів (наприклад, *M, R* та інші), у проведених дослідженнях порівнюються автомати з однаковими значеннями всіх параметрів. У разі необхідності за допомогою аналітичних виразів, отриманих у п. 5.4, можуть бути проведені будь-які додаткові дослідження.

5.5.2 Залежність ефективності від кількості мікрооперацій

Кількість мікрооперацій *N*, що формуються автоматом, визначається імплементованим алгоритмом і не залежить від числа станів та структури ГСА. Результати дослідження залежностей $E^{U_1}(N)$ та $E^{U_2}(N)$ при зміні *N* у діапазоні від 20 до 200 із кроком 20 наведені у табл. 5.19 та графічно показані на рис. 5.5. З аданими результатами можна зробити наступні висновки:

1. Збільшення класу складності МПА у всіх випадках приводить до збільшення ефективності, підвищуючи доцільність використання структур МПА з ОАП U_1 та U_2.

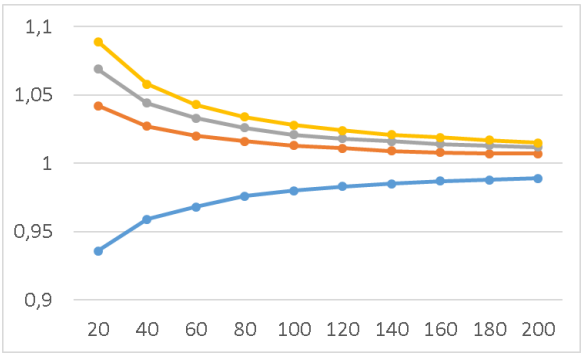
2. Всередині досліджуваного діапазону (при *N*=100) ефективність автоматів Мілі зі структурами U_1 та U_2, а також автомата Мура зі структурою U_2 не перевищує 1,075, що відповідає виграшу в апаратурних витратах близько 7 % у порівнянні з канонічною структурою МПА. Враховуючи можливу погрішність експериментальних досліджень, виконаних у п. 5.2, можна стверджувати, що при обраних значеннях параметрів (п. 5.5.1) перераховані структури в загальному випадку не дають істотного виграшу в апаратурних витратах у порівнянні з канонічним МПА, а в деяких випадках (для структури U_1 першого класу складності) програють МПА з канонічною структурою.

3. Найбільш ефективною серед досліджуваних структур виявляється МПА Мура зі структурою U_2. При $N = 100$ значення ефективності, залежно від класу складності автомата, змінюється від 1,258 до 1,582, забезпечуючи, у максимальному випадку, виграш близько 37 %.

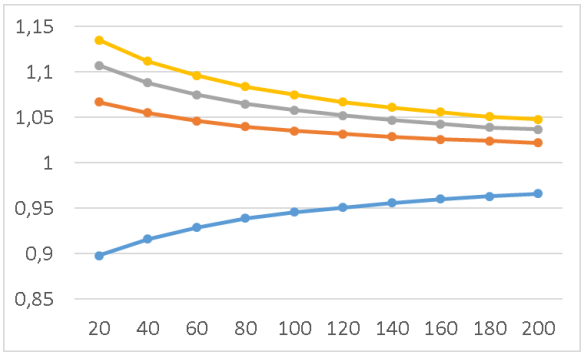
Таблиця 5.19

Залежності $E^{U_1}(N)$ та $E^{U_2}(N)$

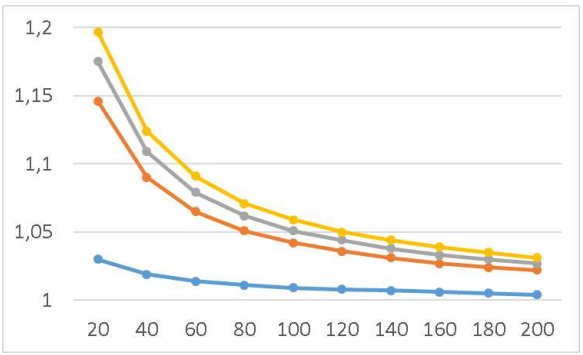
U_i	Тип	Клас склад-ності	N									
			20	40	60	80	100	120	140	160	180	200
U_1	Мілі	1	0,936	0,959	0,968	0,976	0,980	0,983	0,985	0,987	0,988	0,989
		2	1,042	1,027	1,020	1,016	1,013	1,011	1,009	1,008	1,007	1,007
		3	1,069	1,044	1,033	1,026	1,021	1,018	1,016	1,014	1,013	1,012
		4	1,089	1,058	1,043	1,034	1,028	1,024	1,021	1,019	1,017	1,015
	Мура	1	0,898	0,916	0,929	0,939	0,946	0,951	0,956	0,960	0,963	0,966
		2	1,067	1,055	1,046	1,040	1,035	1,032	1,029	1,026	1,024	1,022
		3	1,107	1,088	1,075	1,065	1,058	1,052	1,047	1,043	1,039	1,037
		4	1,135	1,112	1,096	1,084	1,075	1,067	1,061	1,056	1,051	1,048
U_2	Мілі	1	1,030	1,019	1,014	1,011	1,009	1,008	1,007	1,006	1,005	1,004
		2	1,146	1,090	1,065	1,051	1,042	1,036	1,031	1,027	1,024	1,022
		3	1,175	1,109	1,079	1,062	1,051	1,044	1,038	1,033	1,030	1,027
		4	1,197	1,124	1,091	1,071	1,059	1,050	1,044	1,039	1,035	1,031
	Мура	1	1,688	1,485	1,375	1,306	1,258	1,223	1,196	1,175	1,159	1,145
		2	2,361	1,912	1,686	1,549	1,459	1,393	1,344	1,306	1,276	1,251
		3	2,559	2,044	1,785	1,629	1,525	1,450	1,394	1,350	1,315	1,287
		4	2,710	2,150	1,869	1,697	1,582	1,500	1,438	1,389	1,351	1,319



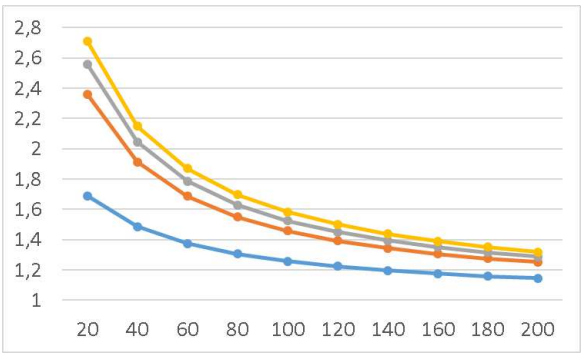
а)



б)



в)



г)

Рисунок 5.5 – Графічне представлення залежностей $E^{U_i}(N)$ для автомата Мілі U_1 (*а*), автомата Мура U_1 (*б*), автомата Мілі U_2 (*в*) та автомата Мура U_2 (*г*)

Настільки висока ефективність даної структури пояснюється тим, що всі КЛС, що входять у структуру, мають відносно невелику кількість вхідних сигналів (що дорівнює R). Це дозволяє реалізувати дані підсхеми в базисі блокової пам'яті ПЛІС, одержавши, з урахуванням $k_6 = 0{,}5$, значну економію апаратурних витрат у порівнянні з канонічною структурою МПА Мура, у якій базис блокової пам'яті може бути використаний тільки для блоку СФМО.

4. Збільшення параметра N приводить, як відомо, до збільшення апаратурних витрат у блоці СФМО. Оскільки використання структур U_1 та U_2 дозволяє одержати економію апаратурних витрат лише в частині функції переходів автомата, збільшення складності функції виходів (збільшення N) при незмінній функції переходів знижує загальну ефективність даних структур. Цим пояснюється гіперболічний характер кривих залежностей $E^{U_1}(N)$ та $E^{U_2}(N)$ (рис. 5.5), значення яких зі збільшенням N прямує до одиниці. На рис. 5.5 по осях абсцис відкладені значення N, по осях ординат – значення ефективності. На кожному рисунку більш висока крива відповідає більш високому рівню складності автомата.

Підкреслимо, що перші три структури, розглянуті в табл. 5.19, не мають достатню ефективність лише при значеннях незмінних параметрів, зазначених у п. 5.5.1. На інших наборах значень ефективність даних структур, можливо, виявиться достатньо високою для їхнього використання у якості альтернативи МПА з канонічною структурою.

5.5.3 Залежність ефективності від кількості комбінаційних схем в операційній частині операційного автомата переходів

При $k_3 < 1$ кількість N_d комбінаційних схем у блоці ОЧ завжди є на одиницю більшою за число використовуваних операцій переходів. Витрати апаратури в комбінаційній схемі, що реалізує в рамках блоку ОЧ частину переходів канонічним способом, залежать від числа таких переходів і можуть багаторазово перевищувати витрати в інших блоках ОЧ. Тому величина N_d сама

собою не характеризує складність блоку ОЧ, маючи безпосередній вплив лише на складність мультиплексора результату.

Проведемо дослідження залежностей $E^{U_1}(N_d)$ та $E^{U_2}(N_d)$ при зміні N_d в діапазоні від 2 до 20 із кроком 2 (табл. 5.20). Графічне представлення вмісту табл. 5.20 дане на рис. 5.6, де по осях абсцис відкладені значення N_d, по осях ординат – значення ефективності. На кожному рисунку більш висока крива відповідає більш високому рівню складності автомата.

Аналіз табл. 5.20 та рис. 5.6 дозволяє затверджувати наступне.

1. У дослідженому діапазоні зміни N_d всі криві мають убутний характер, що пояснюється збільшенням знаменника у виразах (5.53) та (5.54) зі збільшенням N_d за рахунок збільшення апаратурних витрат на реалізацію переходів і мультиплексора результату.

2. Незначні злами, що спостерігаються на графіках, пояснюються ступінчастістю зміни апаратурних витрат у мультиплексорі результату, яка пов'язана зі зміною розрядності коду ОП при досягненні N_d значення чергового ступеню двійки. Подібна ступінчастість спостерігалася експериментально при дослідженні витрат апаратури на реалізацію мультиплексора (табл. 5.13) і врахована у виразі (5.18).

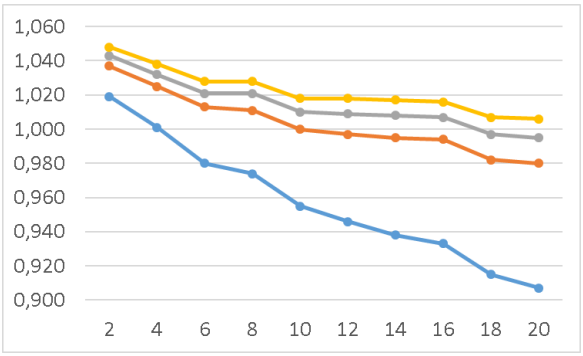
3. При подальшому збільшенні N_d ефективність $E^{U_i}(N_d)$ продовжує знижуватися, наближаючись при $N_d \to \infty$ до нульового значення. Це дозволяє визначити загальну рекомендацію до потенційних методів синтезу МПА з ОАП, яка полягає у прагненні до використання якомога меншої кількості операцій переходів.

4. Збільшення складності МПА сприяє збільшенню ефективності за рахунок більшого виграшу в апаратурних витратах при реалізації функції переходів. Це є наслідком того, що при більшій складності МПА одна й та сама ОП може реалізовувати більшу кількість мікропрограмних переходів при фіксованих витратах апаратури.

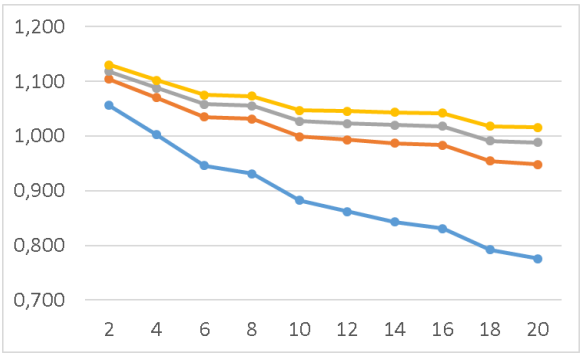
Таблиця 5.20

Залежності $E^{U_1}(N_d)$ та $E^{U_2}(N_d)$

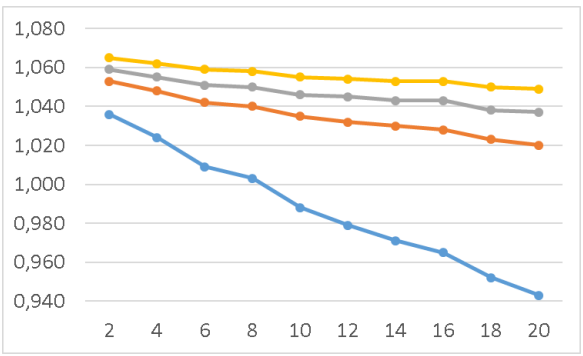
U_i	Тип	Клас склад-ності	N_d									
			2	4	6	8	10	12	14	16	18	20
U_1	Мілі	1	1,019	1,001	0,980	0,974	0,955	0,946	0,938	0,933	0,915	0,907
		2	1,037	1,025	1,013	1,011	1,000	0,997	0,995	0,994	0,982	0,980
		3	1,043	1,032	1,021	1,021	1,010	1,009	1,008	1,007	0,997	0,995
		4	1,048	1,038	1,028	1,028	1,018	1,018	1,017	1,016	1,007	1,006
	Мура	1	1,056	1,002	0,946	0,931	0,882	0,862	0,843	0,831	0,792	0,776
		2	1,104	1,070	1,035	1,031	0,999	0,993	0,987	0,983	0,954	0,948
		3	1,118	1,088	1,058	1,055	1,027	1,023	1,020	1,018	0,991	0,988
		4	1,130	1,102	1,075	1,073	1,047	1,045	1,043	1,042	1,018	1,016
U_2	Мілі	1	1,036	1,024	1,009	1,003	0,988	0,979	0,971	0,965	0,952	0,943
		2	1,053	1,048	1,042	1,040	1,035	1,032	1,030	1,028	1,023	1,020
		3	1,059	1,055	1,051	1,050	1,046	1,045	1,043	1,043	1,038	1,037
		4	1,065	1,062	1,059	1,058	1,055	1,054	1,053	1,053	1,050	1,049
	Мура	1	1,380	1,326	1,258	1,231	1,172	1,137	1,104	1,083	1,037	1,010
		2	1,515	1,489	1,459	1,450	1,421	1,409	1,397	1,389	1,363	1,351
		3	1,569	1,548	1,525	1,520	1,497	1,490	1,483	1,478	1,457	1,450
		4	1,619	1,601	1,582	1,579	1,561	1,556	1,552	1,550	1,532	1,528



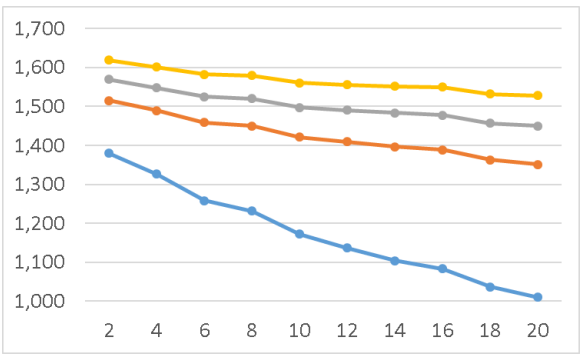
а)



б)



в)



г)

Рисунок 5.6 – Графічне представлення залежностей $E^{U_i}(N_d)$ для автомата Мілі U_1 (*а*), автомата Мура U_1 (*б*), автомата Мілі U_2 (*в*) та автомата Мура U_2 (*г*)

5.5.4 Залежність ефективності від ступеню мінімізації комбінаційних логічних схем

Даний коефіцієнт дає можливість регулювати апаратурні витрати у блоках МПА, які синтезуються за системою булевих рівнянь, при збереженні інших параметрів автомата, таких як кількість станів, розрядність структурного коду стану, число переходів та інші. Збереження даних параметрів, у свою чергу, сприяє збереженню витрат апаратури у вузлах, синтезованих не за системою булевих рівнянь.

Результати дослідження залежностей $E^{U_1}(k_1)$ та $E^{U_2}(k_1)$ при зміні k_1 в діапазоні від 0,1 до 1 із кроком 0,1 наведені в табл. 5.21 і графічно представлені на рис. 5.7, де по осях абсцис відкладені значення k_1, а більш висока крива відповідає автомату більш високого класу складності. Підкреслимо, що отримані результати є наближеними, оскільки однакове значення k_1 застосовується одночасно до всіх схем КЛС у порівнюваних структурах, що для реальних автоматів у загальному випадку неможливо. Проаналізуємо отримані результати.

1. Для всіх досліджуваних структур і класів складності МПА зі збільшенням k_1 спостерігається зростання ефективності, яка при $k_1 = 1$ досягає максимального значення, що визначається також іншими параметрами автоматів.

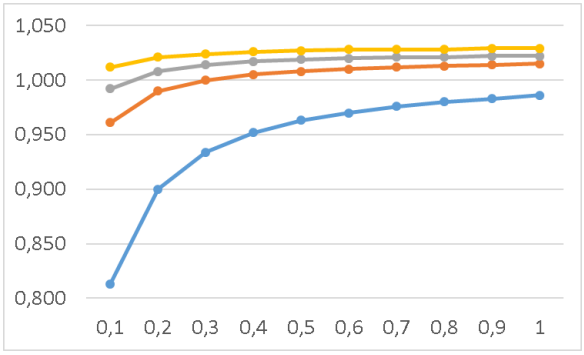
2. Справедливо вважати, що для реальних КЛС $0{,}5 < k_1 < 1$. Зміна k_1 в даному діапазоні приводить до мінімальної зміни ефективності досліджуваних структур. Те, що коефіцієнт k_1 не виявляє істотного впливу на ефективність структур, підкреслює справедливість результатів інших досліджень МПА, що імплементують довільні алгоритми керування.

3. Для МПА більших класів складності структури автоматів Мура демонструють в цілому більшу ефективність у порівнянні зі структурами автоматів Мілі (аж до 50 відсотків для класу складності 4).

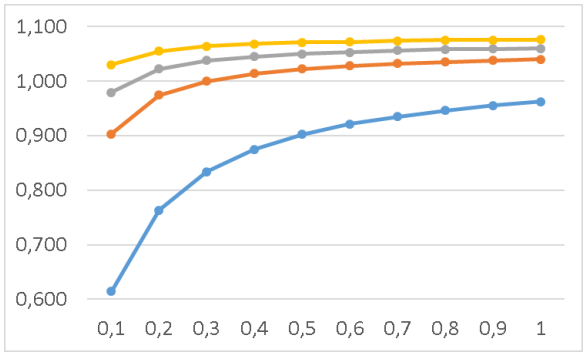
Таблиця 5.21

Залежності $E^{U_1}(k_1)$ та $E^{U_2}(k_1)$

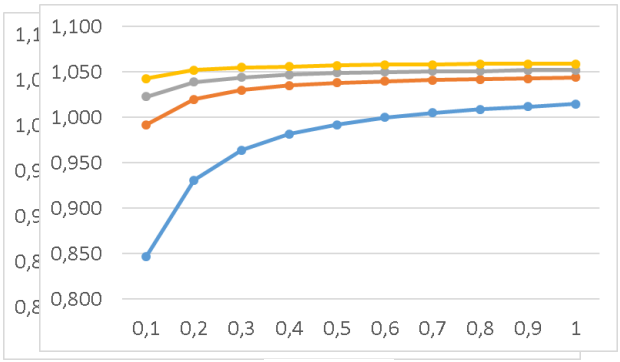
U_i	Тип	Клас склад-ності	k_1									
			0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
U_1	Мілі	1	0,813	0,900	0,934	0,952	0,963	0,970	0,976	0,980	0,983	0,986
		2	0,961	0,990	1,000	1,005	1,008	1,010	1,012	1,013	1,014	1,015
		3	0,992	1,008	1,014	1,017	1,019	1,020	1,021	1,021	1,022	1,022
		4	1,012	1,021	1,024	1,026	1,027	1,028	1,028	1,028	1,029	1,029
	Мура	1	0,614	0,763	0,834	0,875	0,902	0,921	0,935	0,946	0,955	0,962
		2	0,903	0,974	1,000	1,014	1,022	1,028	1,032	1,035	1,038	1,040
		3	0,979	1,022	1,038	1,045	1,050	1,053	1,056	1,058	1,059	1,060
		4	1,030	1,055	1,064	1,068	1,071	1,072	1,074	1,075	1,075	1,076
U_2	Мілі	1	0,847	0,931	0,964	0,982	0,992	1,000	1,005	1,009	1,012	1,015
		2	0,992	1,020	1,030	1,035	1,038	1,040	1,041	1,042	1,043	1,044
		3	1,023	1,039	1,044	1,047	1,049	1,050	1,051	1,051	1,052	1,052
		4	1,043	1,052	1,055	1,056	1,057	1,058	1,058	1,059	1,059	1,059
	Мура	1	0,756	0,972	1,079	1,144	1,187	1,217	1,240	1,258	1,272	1,284
		2	1,224	1,347	1,394	1,419	1,435	1,445	1,453	1,459	1,463	1,467
		3	1,376	1,457	1,486	1,501	1,511	1,517	1,521	1,525	1,527	1,529
		4	1,493	1,543	1,560	1,569	1,574	1,578	1,580	1,582	1,584	1,585



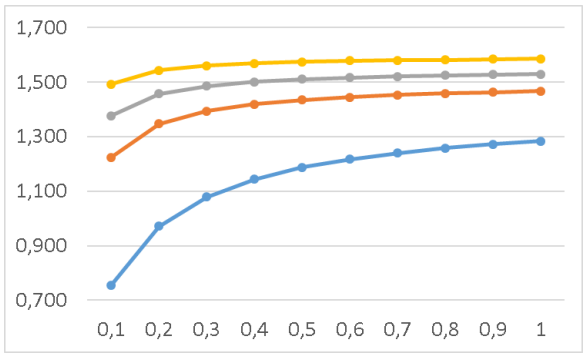
а)



б)



в)



г)

Рисунок 5.7 – Графічне представлення залежностей $E^{U_i}(k_1)$ для автомата Мілі U_1 (*а*), автомата Мура U_1 (*б*), автомата Мілі U_2 (*в*) та автомата Мура U_2 (*г*)

5.5.5 Залежність ефективності від складності функціональних блоків в операційному автоматі переходів

Принцип операційного перетворення кодів станів припускає використання операцій переходів будь-якої складності, яка регулюється у виразах (5.44), (5.45), (5.48), (5.52) коефіцієнтом k_2. Проведемо дослідження впливу даного коефіцієнта на $E^{U_1}(k_2)$ та $E^{U_2}(k_2)$ при його зміні від 1 до 10 із кроком 1. Отримані результати зібрані в табл. 5.22 і графічно показані на рис. 5.8, де значення k_2 відкладені по осях абсцис, а більш висока крива відповідає автомату більш високого класу складності.

Вміст табл. 5.22 та рис. 5.8 дозволяє зробити наступні висновки:

1. Зі збільшенням k_2 зростають витрати апаратури в блоці ОЧ структур U_1 та U_2, що при незмінних інших витратах приводить до зниження їх ефективності, яка при $k_2 \to \infty$ прямує до нуля.

2. Зі збільшенням класу складності МПА зниження ефективності зі зростанням k_2 стає більш плавним. Це пов'язане з тим, що при фіксованому значенні апаратурних витрат на реалізацію операцій переходів їх частка в сумарних витратах будь-якого МПА виявляється тим меншою, чим вищим є клас складності автомата.

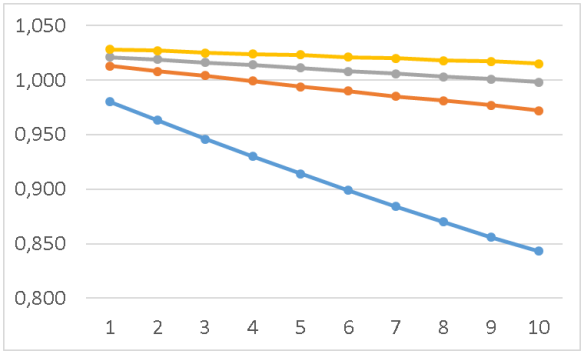
3. МПА Мура U_2 малої складності починає програвати по апаратурних витратах еквівалентному канонічному МПА при $k_2 > 5$. Ефективність даної структури більш високих класів складності при десятикратному збільшенні k_2 зберігається досить високою (не нижче 1,28).

3. Як і у попередньому експерименті, структури автоматів Мура демонструють в цілому більшу ефективність у порівнянні зі структурами автоматів Мілі (особливо у випадку структури U_2).

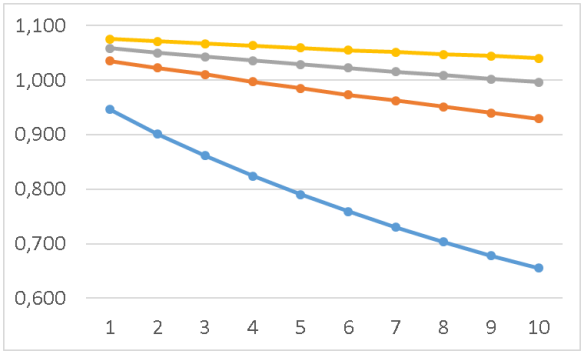
Таблиця 5.22

Залежності $E^{U_1}(k_2)$ та $E^{U_2}(k_2)$

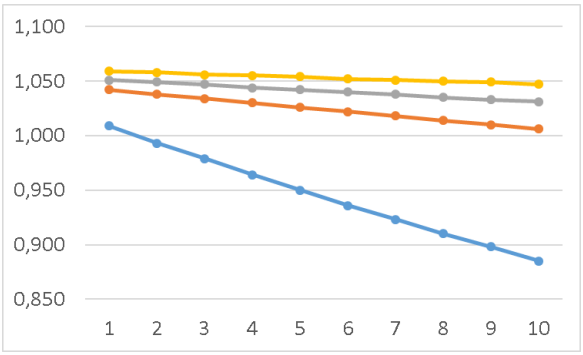
U_i	Тип	Клас склад-ності	k_2									
			1	2	3	4	5	6	7	8	9	10
U_1	Мілі	1	0,980	0,963	0,946	0,930	0,914	0,899	0,884	0,870	0,856	0,843
		2	1,013	1,008	1,004	0,999	0,994	0,990	0,985	0,981	0,977	0,972
		3	1,021	1,019	1,016	1,014	1,011	1,008	1,006	1,003	1,001	0,998
		4	1,028	1,027	1,025	1,024	1,023	1,021	1,020	1,018	1,017	1,015
	Мура	1	0,946	0,901	0,861	0,824	0,790	0,759	0,730	0,703	0,678	0,655
		2	1,035	1,022	1,010	0,997	0,985	0,973	0,962	0,951	0,940	0,929
		3	1,058	1,050	1,043	1,036	1,029	1,022	1,015	1,009	1,002	0,996
		4	1,075	1,071	1,067	1,063	1,059	1,055	1,051	1,047	1,044	1,040
U_2	Мілі	1	1,009	0,993	0,979	0,964	0,950	0,936	0,923	0,910	0,898	0,885
		2	1,042	1,038	1,034	1,030	1,026	1,022	1,018	1,014	1,010	1,006
		3	1,051	1,049	1,047	1,044	1,042	1,040	1,038	1,035	1,033	1,031
		4	1,059	1,058	1,056	1,055	1,054	1,052	1,051	1,050	1,049	1,047
	Мура	1	1,258	1,193	1,134	1,081	1,032	0,988	0,947	0,910	0,875	0,843
		2	1,459	1,437	1,416	1,396	1,376	1,357	1,339	1,321	1,303	1,286
		3	1,525	1,512	1,500	1,488	1,476	1,464	1,452	1,441	1,430	1,419
		4	1,582	1,575	1,568	1,561	1,554	1,547	1,540	1,533	1,527	1,520



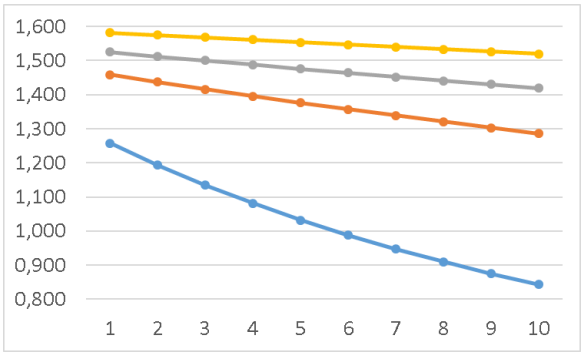
а)



б)



в)



г)

Рисунок 5.8 – Графічне представлення залежностей $E^{U_i}(k_2)$ для автомата Мілі U_1 (*а*), автомата Мура U_1 (*б*), автомата Мілі U_2 (*в*) та автомата Мура U_2 (*г*)

5.5.6 Залежність ефективності від частки переходів, реалізованих операційним способом

Даний коефіцієнт відбиває ефективність використовуваного методу алгебраїчного синтезу: чим вище ефективність алгоритму, тем більша кількість переходів реалізується за допомогою операцій переходів. Принцип операційного перетворення кодів станів, закладений у досліджувані структури МПА з ОАП U_1 та U_2, припускає подібну реалізацію для всіх без винятку переходів, що відповідає значенню $k_3 = 1$. У той же час при $k_3 = 0$ операційне перетворення кодів станів не використовується і організація схеми формування переходів у вигляді ОАП лишається сенсу.

Таким чином, проведемо дослідження функції $E^{U_1}(k_3)$ та $E^{U_2}(k_3)$ при зміні k_3 від 0,1 до 1 з кроком 0,1. Отримані результати зібрані в табл. 5.23 і графічно зображені на рис. 5.9, де значення k_3 відкладені по осях абсцис, а більш високі в крапках $k_3 = 1$ криві відповідають автоматам більш високого класу складності.

Проаналізуємо отримані результати.

1. Збільшення k_3 приводить до зниження апаратурних витрат у комбінаційній схемі блоку ОЧ, яка реалізує «канонічні» переходи автомата. Це приводить до зростання ефективності досліджуваних структур.

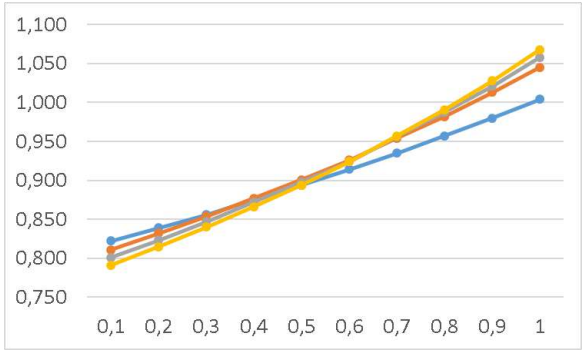
2. При $k_3 = 1$ для МПА Мура зі структурою U_1 середнього класу складності та вище виграш в апаратурних витратах перевищує 10% (рис. 5.9, *б)*, що дозволяє розглядати дану структуру як альтернативу МПА з канонічною структурою. Однак досягнення $k_3 = 1$ при значеннях інших параметрів, обраних в п. 5.5.1, у загальному випадку не завжди можливе і вимагає розробки високоефективних алгоритмів алгебраїчного синтезу даного класу МПА.

3. У випадку МПА Мура зі структурою U_2 будь-якого класу складності ефективність виявляється більшою за одиницю при $k_3 > 0{,}3$. При $k_3 = 1$ максимальний виграш (для автоматів надвисокого класу складності) становить 40 % ($E^{U_2} = 1{,}7$) у порівнянні з канонічним МПА Мура.

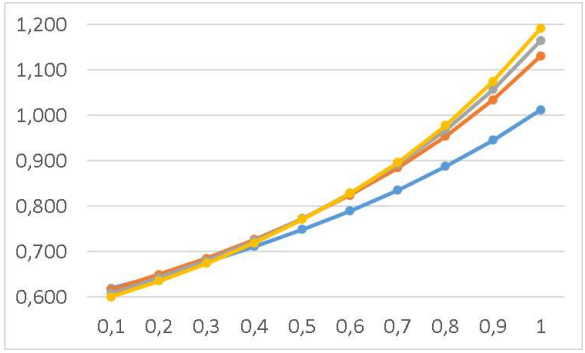
Таблиця 5.23

Залежності $E^{U_1}(k_3)$ та $E^{U_2}(k_3)$

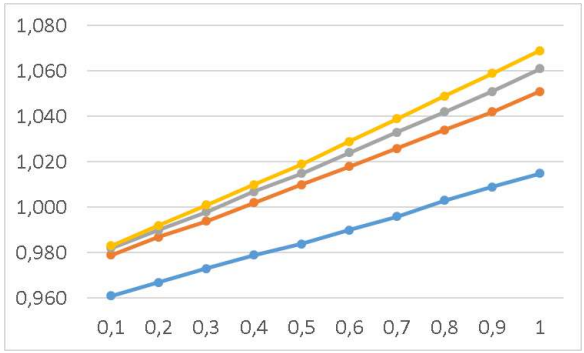
U_i	Тип	Клас склад-ності	k_3									
			0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
U_1	Мілі	1	0,822	0,839	0,856	0,875	0,894	0,914	0,935	0,957	0,980	1,004
		2	0,811	0,832	0,854	0,877	0,901	0,926	0,954	0,982	1,013	1,045
		3	0,801	0,823	0,847	0,872	0,898	0,924	0,956	0,988	1,021	1,058
		4	0,791	0,815	0,840	0,866	0,894	0,924	0,957	0,991	1,028	1,068
	Мура	1	0,620	0,648	0,678	0,712	0,749	0,790	0,836	0,888	0,946	1,013
		2	0,617	0,650	0,686	0,727	0,773	0,825	0,885	0,954	1,035	1,131
		3	0,609	0,643	0,681	0,724	0,773	0,829	0,893	0,968	1,058	1,165
		4	0,601	0,636	0,675	0,720	0,771	0,829	0,897	0,978	1,075	1,192
U_2	Мілі	1	0,961	0,967	0,973	0,979	0,984	0,990	0,996	1,003	1,009	1,015
		2	0,979	0,987	0,994	1,002	1,010	1,018	1,026	1,034	1,042	1,051
		3	0,982	0,990	0,998	1,007	1,015	1,024	1,033	1,042	1,051	1,061
		4	0,983	0,992	1,001	1,010	1,019	1,029	1,039	1,049	1,059	1,069
	Мура	1	0,940	0,971	1,004	1,039	1,076	1,117	1,160	1,207	1,258	1,313
		2	1,003	1,044	1,088	1,136	1,189	1,247	1,310	1,380	1,459	1,546
		3	1,015	1,059	1,108	1,161	1,219	1,283	1,355	1,435	1,525	1,627
		4	1,023	1,070	1,122	1,179	1,242	1,313	1,392	1,481	1,582	1,698



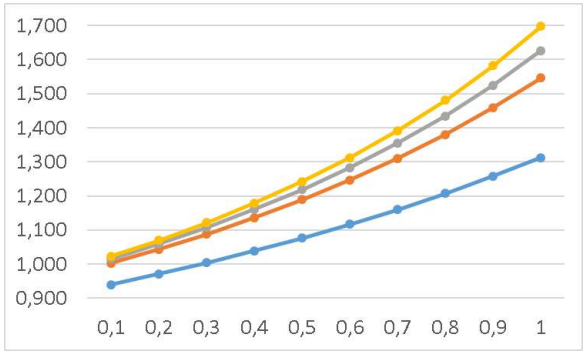
а)



б)



в)



г)

Рисунок 5.9 – Графічне представлення залежностей $E^{U_i}(k_3)$ для автомата Мілі U_1 (*а*), автомата Мура U_1 (*б*), автомата Мілі U_2 (*в*) та автомата Мура U_2 (*г*)

5.5.7 Залежність ефективності від частки умовних переходів

Даний коефіцієнт використовується у виразі (5.29) і дозволяє регулювати середню довжину термів s у булевих рівняннях. При цьому регулювання величини s здійснюється у відносно невеликому діапазоні, виявляючи незначний вплив на апаратурні витрати в кон'юнктивній частині комбінаційних схем.

Особливістю виразу (5.29) є те, що зі зменшенням k_4 значення знаменника може виявитися рівним нулю. При комбінаціях значень M та B, зазначених у табл. 5.18, де $B = 2M$, знаменник дорівнюватиме нулю при $k_4 = 0{,}5$. При $k_4 = 1$ всі булеві терми в системах рівнянь мають однакову довжину $(R + L_1)$, де L_1 при $B = 2M$ дорівнює одиниці. Таким чином, для коефіцієнта k_4 припустимим діапазоном зміни є (0,5; 1], при виході за який чисельні значення, одержувані за допомогою (5.29), не мають сенсу і не можуть бути використані для оцінки ефективності досліджуваних структур.

Результати дослідження залежностей $E^{U_1}(k_4)$ та $E^{U_2}(k_4)$ при зміні k_4 від 0,6 до 1 з кроком 0,1 представлені в табл. 5.24.

Таблиця 5.24

Залежності $E^{U_1}(k_4)$ та $E^{U_2}(k_4)$

U_i	Тип	Клас склад-ності	k_4				
			0,6	0,7	0,8	0,9	1
U_1	Мілі	1	0,980	0,980	0,980	0,980	0,980
		2	1,013	1,013	1,013	1,013	1,013
		3	1,021	1,021	1,021	1,021	1,021
		4	1,028	1,028	1,028	1,028	1,028
	Мура	1	0,947	0,946	0,946	0,946	0,946
		2	1,035	1,035	1,035	1,035	1,035
		3	1,057	1,057	1,058	1,058	1,058
		4	1,074	1,074	1,075	1,075	1,075
U_2	Мілі	1	1,009	1,009	1,009	1,009	1,009
		2	1,042	1,042	1,042	1,042	1,042
		3	1,051	1,051	1,051	1,051	1,051
		4	1,059	1,059	1,059	1,059	1,059
	Мура	1	1,273	1,260	1,257	1,255	1,255
		2	1,474	1,461	1,457	1,456	1,456
		3	1,540	1,527	1,523	1,522	1,522
		4	1,597	1,584	1,581	1,580	1,579

Очевидно, що для досліджуваних автоматів збільшення k_4 на величину 0,1 не виявляє впливу на ефективність, змінюючи апаратурні витрати в максимальному випадку у межах 2 % (МПА Мура зі структурою U_2). Таким чином, зміна середньої довжини термів у невеликому діапазоні не виявляє впливу на ефективність досліджуваних структур МПА з ОАП. Відзначимо, що результати даного дослідження є в значній мірі усередненими, оскільки коефіцієнт k_4 застосовується одночасно до всіх блоків досліджуваних структур, що синтезуються за системами булевих рівнянь.

5.5.8 Залежність ефективності від частки термів системи булевих рівнянь

Як відзначено в п. 5.3, даний коефіцієнт дозволяє змінювати середнє число термів в одному рівнянні кожної системи булевих рівнянь, використовуваної в досліджуваних структурах, здійснюючи дану дію через вираз (5.25). Таким чином, зі зміною k_5 змінюються апаратурні витрати в кон'юнктивній частині КЛС незалежно від диз'юнктивної частини. Подібно до k_4, значення коефіцієнта k_5 застосовується одночасно до всіх блоків типу КЛС у досліджуваних структурах, що не дозволяє відслідковувати зміни в диз'юнктивній частині окремих блоків.

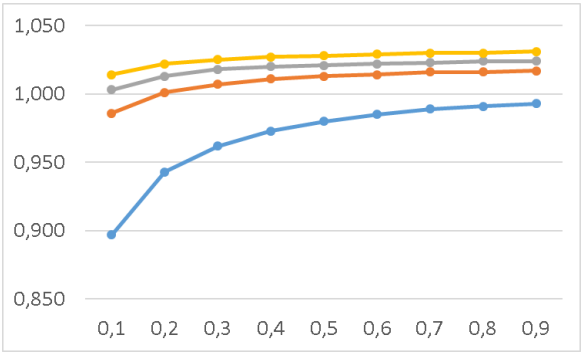
Виконаємо дослідження залежностей $E^{U_1}(k_5)$ та $E^{U_2}(k_5)$ при зміні k_5 від 0,1 до 0,9 із кроком 0,1 (табл. 5.25). Графічне представлення вмісту табл. 5.25 дане на рис. 5.10, де по осях абсцис відкладені значення k_5, а більш висока крива відповідає автомату більш високого класу складності. Аналіз кривих на рис. 5.10 дозволяє зробити наступні висновки.

1. Зростаючий характер кривих на рис. 5.10 *а – в* говорить про те, що для відповідних структур у виразах (5.53) та (5.54) чисельник зі збільшенням k_5 зростає швидше, ніж знаменник, що виражається у зростанні ефективності. Однак навіть при $k_5 = 0{,}9$ виграш в апаратурних витратах для даних структур не перевищує 9 %, що свідчить про загальну низьку ефективность даних структур.

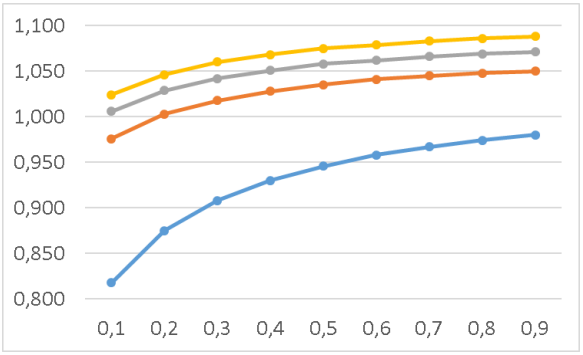
Таблиця 5.25

Залежності $E^{U_1}(k_5)$ та $E^{U_2}(k_5)$

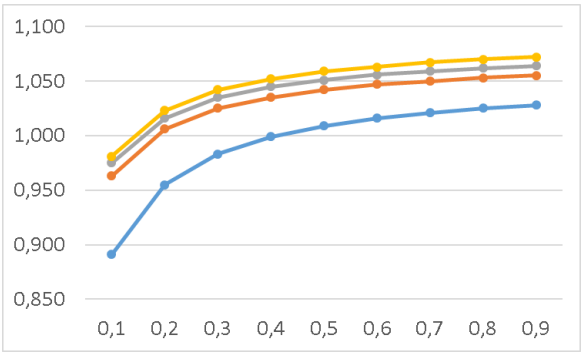
U_i	Тип	Клас слож-ності	k_5								
			0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
U_1	Мілі	1	0,897	0,943	0,962	0,973	0,980	0,985	0,989	0,991	0,993
		2	0,986	1,001	1,007	1,011	1,013	1,014	1,016	1,016	1,017
		3	1,003	1,013	1,018	1,020	1,021	1,022	1,023	1,024	1,024
		4	1,014	1,022	1,025	1,027	1,028	1,029	1,030	1,030	1,031
	Мура	1	0,818	0,875	0,908	0,930	0,946	0,958	0,967	0,974	0,980
		2	0,976	1,003	1,018	1,028	1,035	1,041	1,045	1,048	1,050
		3	1,006	1,029	1,042	1,051	1,058	1,062	1,066	1,069	1,071
		4	1,024	1,046	1,060	1,068	1,075	1,079	1,083	1,086	1,088
U_2	Мілі	1	0,891	0,955	0,983	0,999	1,009	1,016	1,021	1,025	1,028
		2	0,963	1,006	1,025	1,035	1,042	1,047	1,050	1,053	1,055
		3	0,975	1,016	1,035	1,045	1,051	1,056	1,059	1,062	1,064
		4	0,981	1,023	1,042	1,052	1,059	1,063	1,067	1,070	1,072
	Мура	1	1,351	1,305	1,282	1,267	1,258	1,251	1,246	1,242	1,239
		2	1,899	1,665	1,558	1,498	1,459	1,431	1,411	1,395	1,383
		3	2,054	1,771	1,664	1,571	1,525	1,492	1,468	1,449	1,435
		4	2,170	1,856	1,715	1,634	1,582	1,546	1,519	1,498	1,482



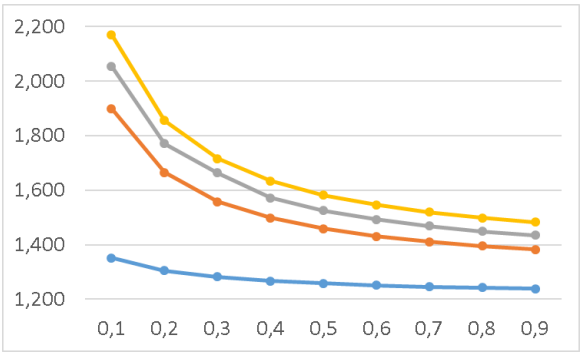
а)



б)



в)



г)

Рисунок 5.10 – Графічне представлення залежностей $E^{U_i}(k_5)$ для автомата Мілі U_1 (*а*), автомата Мура U_1 (*б*), автомата Мілі U_2 (*в*) та автомата Мура U_2 (*г*)

2. Ефективність МПА Мура зі структурою U_2 (рис. 5.10, *г)* зі збільшенням k_5 нелінійно зменшується, забезпечуючи в найгіршому випадку (для автомата малої складності) виграш у 18 % ($E^{U_2} = 1{,}24$). Убутний характер кривих говорить про те, що зі збільшенням k_5 відносний приріст апаратурних витрат у МПА Мура зі структурою U_2 виявляється вищим, ніж у канонічному МПА. Наприклад, для автомата третього класу складності величина H^{U_K} при зміні k_5 з 0,3 до 0,4 збільшується на 22,4 % (з 6069 до 7429 LUT-елементів), а величина H^{U_2} – на 27,9 % (з 3810 до 4876 LUT-елементів). При зміні k_5 з 0,7 до 0,8 величина H^{U_K} збільшується на 11,8 % (з 11509 до 12869 LUT-елементів), у той час як величина H^{U_2} збільшується на 13,2 % (з 8074 до 9140 LUT-елементів).

5.5.9 Залежність ефективності від використання блокової пам’яті FPGA

Значення коефіцієнта k_6 залежить як від використовуваного елементного базису, так і від проектованого пристрою, що реалізується на ПЛІС типу FPGA. На практиці доцільно розглядати два крайні значення даного коефіцієнта: $k_6 = 0$ і $k_6 = 1$. При $k_6 = 0$ блокова пам'ять ПЛІС може використовуватися без обмежень і її витрати в реалізованому проекті не є частиною загальних витрат апаратури, обумовлених тільки LUT-елементами. При $k_6 = 1$ використання блокової пам'яті для реалізації окремих блоків МПА неможливе (наприклад, через її відсутність або недостатню кількість). Проміжні значення k_6 виражають певну перевагу до одного з цих базисів.

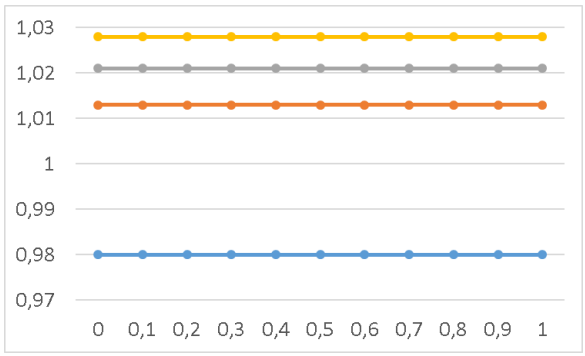
У табл. 5.26 наведені значення функцій $E^{U_1}(k_6)$ та $E^{U_2}(k_6)$ при зміні k_6 від 0 до 1 із кроком 0,1. У графічній формі вміст табл. 5.26 показаний на рис. 5.11, де по осях абсцис відкладені значення k_6, а більш висока крива відповідає автомату більш високого класу складності.

Проаналізуємо отримані результати.

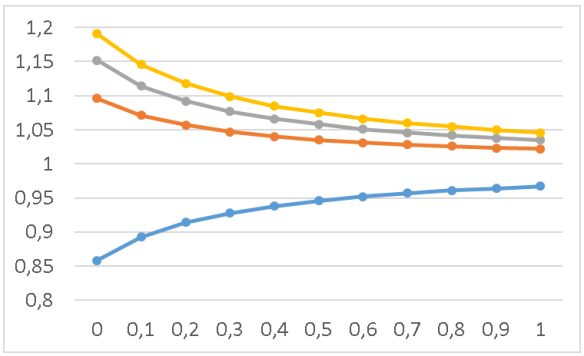
Таблиця 5.26

Залежності $E^{U_1}(k_6)$ та $E^{U_2}(k_6)$

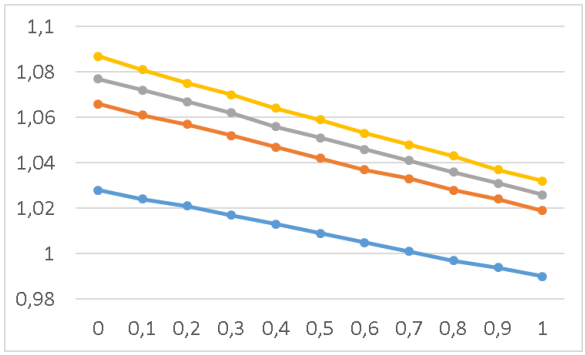
Ui	Тип	Клас склад-ності	k_6										
			0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
U_1	Мілі	1	0,980	0,980	0,980	0,980	0,980	0,980	0,980	0,980	0,980	0,980	0,980
		2	1,013	1,013	1,013	1,013	1,013	1,013	1,013	1,013	1,013	1,013	1,013
		3	1,021	1,021	1,021	1,021	1,021	1,021	1,021	1,021	1,021	1,021	1,021
		4	1,028	1,028	1,028	1,028	1,028	1,028	1,028	1,028	1,028	1,028	1,028
	Мура	1	0,858	0,893	0,914	0,928	0,938	0,946	0,952	0,957	0,961	0,964	0,967
		2	1,096	1,071	1,057	1,047	1,040	1,035	1,031	1,028	1,026	1,023	1,022
		3	1,152	1,114	1,092	1,077	1,066	1,058	1,051	1,046	1,042	1,038	1,035
		4	1,191	1,146	1,118	1,099	1,085	1,075	1,066	1,060	1,055	1,050	1,046
U_2	Мілі	1	1,028	1,024	1,021	1,017	1,013	1,009	1,005	1,001	0,997	0,994	0,990
		2	1,066	1,061	1,057	1,052	1,047	1,042	1,037	1,033	1,028	1,024	1,019
		3	1,077	1,072	1,067	1,062	1,056	1,051	1,046	1,041	1,036	1,031	1,026
		4	1,087	1,081	1,075	1,070	1,064	1,059	1,053	1,048	1,043	1,037	1,032
	Мура	1	2,856	1,861	1,554	1,404	1,316	1,258	1,216	1,185	1,162	1,142	1,127
		2	6,586	2,777	2,048	1,738	1,567	1,459	1,384	1,329	1,287	1,253	1,227
		3	7,929	3,067	2,205	1,846	1,649	1,525	1,439	1,376	1,328	1,291	1,260
		4	8,839	3,298	2,337	1,938	1,720	1,582	1,487	1,418	1,365	1,323	1,290



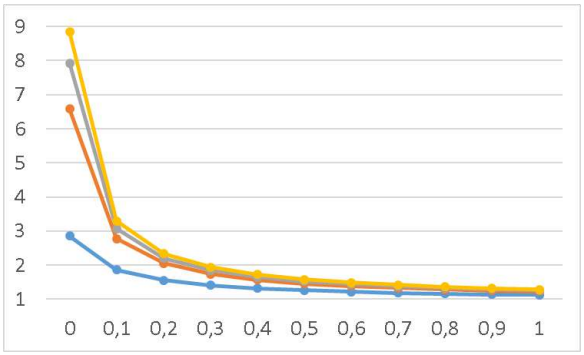
а)



б)



в)



г)

Рис. 5.11. Графічне представлення залежностей $E^{U_i}(k_6)$ для автомата Мілі U_1 (*а*), автомата Мура U_1 (*б*), автомата Мілі U_2 (*в*) та автомата Мура U_2 (*г*)

1. Ефективність МПА Мілі зі структурою U_1 не залежить від k_6 через відсутність у порівнюваних структурах блоків, які в загальному випадку можуть бути реалізовані в базисі блокової пам'яті.

2. У канонічному МПА Мура зі збільшенням k_6 витрати апаратури збільшуються тільки в блоці СФМО. В МПА Мура зі структурою U_1 відбувається збільшення витрат як у СФМО, так і в Z-підсхемі, яка в даній структурі також може бути реалізована в базисі блокової пам'яті. В результаті ефективність МПА Мура U_1 убуває зі збільшенням k_6, за винятком автомата малої складності, де при обраних значеннях параметрів зростання витрат у канонічній структурі виявляється швидшим, ніж у структурі U_1 (рис. 5.11, *б)*.

3. У МПА Мілі зі структурою U_2 збільшення k_6 приводить до лінійного зростання величини H^{U_2}, що викликане збільшенням апаратурних витрат у структурних блоках, реалізованих у базисі блокової пам'яті ПЛІС. Оскільки в канонічному МПА Мілі такі блоки відсутні, величина H^{U_K} для автомата Мілі зі зміною k_6 залишається незмінною. Як наслідок, у виразі (5.54) лінійне зростання знаменника при незмінному чисельнику приводить до лінійного убування величини E^{U_2}, що видно на рис. 5.11, *в.*

4. У випадку МПА Мура зі структурою U_2 (рис. 5.11, *г*) загальний характер кривих схожий із МПА Мура U_1. Примітними тут є значення ефективності при $k_6 = 0$, яке дорівнює, залежно від класу складності автомата, від 2,85 до 8,85. Дані значення показують, у скільки разів знижуються апаратурні витрати в блоках, для яких єдиним можливим базисом реалізації є базис LUT-елементів. На практиці такі значення досяжні при використанні ПЛІС із обсягом блокової пам'яті, що повністю покриває потреби реалізованого проекту.

У той же час при $k_6 = 1$ ефективність МПА Мура U_2 залишається досить високою, становлячи, залежно від класу складності автомата, від 1,12 до 1,29. Це робить дану структуру більш кращою, ніж канонічний МПА, незалежно від обсягу блокової пам'яті у використовуваній ПЛІС.

5.6 Визначення області ефективного застосування мікропрограмного автомата з операційним автоматом переходів

Дослідження, проведені в п. 5.5, дозволяють виявити фактори, що сприяють підвищенню ефективності пропонованих структур МПА з ОАП у порівнянні із МПА з канонічною структурою. Узагальнені результати зібрані в табл. 5.27, де в рядках зазначені досліджувані структури, у стовпцях – параметри, досліджені в п. 5.5.2 – 5.5.9. У комірках таблиці вказані умови підвищення ефективності відповідної структури: «+» – ефективність підвищується із зростанням значення відповідного параметра, «–» – ефективність підвищується зі зменшенням значення параметра, «0» – зміна значення параметра не впливає на ефективність структури.

Таблиця 5.27

Фактори, що сприяють підвищенню ефективності структур МПА з ОАП U_1 та U_2

U_i	Тип	Клас склад-ності	Параметр							
			N	N_d	k_1	k_2	k_3	k_4	k_5	k_6
U_1	Мілі	1	+	–	+	–	+	0	+	0
		2	–	–	+	–	+	0	+	0
		3	–	–	+	–	+	0	+	0
		4	–	–	+	–	+	0	+	0
	Мура	1	+	–	+	–	+	0	+	+
		2	–	–	+	–	+	0	+	–
		3	–	–	+	–	+	0	+	–
		4	–	–	+	–	+	0	+	–
U_2	Мілі	1	–	–	+	–	+	0	+	–
		2	–	–	+	–	+	0	+	–
		3	–	–	+	–	+	0	+	–
		4	–	–	+	–	+	0	+	–
	Мура	1	–	–	+	–	+	–	–	–
		2	–	–	+	–	+	–	–	–
		3	–	–	+	–	+	–	–	–
		4	–	–	+	–	+	–	–	–

Підберемо для кожної зі структур МПА з ОАП U_1 та U_2 кілька наборів значень параметрів з табл. 5.27, при яких значення ефективності становить не

менш ніж 1,1 (виграш в апаратурних витратах не менший за 10 %). На підставі таких наборів визначимо для кожної структури область її ефективного застосування у вигляді множини відносин деяких параметрів з табл. 5.27.

МПА Мілі U_1

Область ефективного застосування даної структури МПА з ОАП може бути визначена з табл. 5.28. Для кожного класу складності рядки розташовані в порядку убування значень E^{U_1}.

Таблиця 5.28

Область ефективного застосування МПА Мілі U_1

Клас склад-ності	Параметр								E^{U_1}
	N	N_d	k_1	k_2	k_3	k_4	k_5	k_6	
1	20	4	1	1	1	0,75	0,9	0,5	1,152
	20	4	0,95	1	0,97	0,75	0,9	0,5	1,115
	20	4	1	1	0,99	0,75	0,6	0,5	1,113
	20	4	0,8	0,5	0,99	0,75	0,5	0,5	1,109
2	20	4	1	1	1	0,75	0,9	0,5	1,252
	20	8	0,9	1	0,95	0,75	0,9	0,5	1,123
	20	8	0,9	1	0,95	0,75	0,8	0,5	1,118
	20	4	0,9	1	0,9	0,75	0,8	0,5	1,100
3	20	8	0,9	1	0,95	0,75	0,7	0,5	1,145
	20	8	0,8	1	0,95	0,75	0,5	0,5	1,127
	20	15	0,9	1	0,95	0,75	0,8	0,5	1,104
	30	15	0,9	1	0,97	0,75	0,8	0,5	1,100
4	30	8	0,9	1	0,95	0,75	0,5	0,5	1,122
	20	30	0,9	1	0,97	0,75	0,7	0,5	1,114
	30	15	0,9	1	0,95	0,75	0,8	0,5	1,104
	50	8	0,95	1	0,97	0,75	0,5	0,5	1,100

З табл. 5.28. можна зробити наступні основні висновки:

1. Для всіх класів складності частка переходів, реалізована за допомогою операцій переходів (стовпець k_3) повинна становити не менш ніж 95 %, а для МПА малої складності – не менш ніж 97 %. При відносно невеликому числі припустимих операцій переходів, що дорівнює $N_d - 1$, досягнення $k_3 \geq 0{,}95$ є важко виконуваною задачею.

2. Для всіх класів складності значення $E^{U_1} \geq 1{,}1$ досягається при невеликому числі мікрооперацій N (від 20 до 30). Значення $N = 50$ припустиме тільки для МПА надвисокого класу складності при малому числі операцій переходів ($N_d - 1 = 8$) і $k_3 \geq 0{,}97$.

3. У випадку МПА надвисокого класу складності при $N = 20$ може бути задіяна досить велика кількість операцій переходів ($N_d = 30$) при збереженні $E^{U_1} = 1{,}11$. Це в загальному випадку спрощує досягнення значення $k_3 \geq 0{,}97$, що є необхідною вимогою в цьому випадку.

4. Структура ефективна при $k_5 > 0{,}7$, тобто тоді, коли диз'юнктивна частина в реалізованих структурами системах булевих рівнянь має складність вище середньої. При невиконанні даної умови ефективність у більшості випадків виявляється нижчою за 1,1.

5. Останній стовпець показує, що середнє значення E^{U_1} для досліджених класів МПА Мілі U_1 становить близька 1,12, що відповідає виграшу в апаратурних витратах 11 % у порівнянні з канонічним МПА.

Таким чином, область ефективного застосування МПА Мілі зі структурою U_1 визначається наступними основними критеріями: $N < 30$, $N_d \leq 8$, $k_3 \geq 0{,}95$, $k_5 > 0{,}7$. При цьому очевидною є загальна низька ефективність МПА Мілі зі структурою U_1, що дорівнює в середньому 1,12. Справедливо вважати, що в загальному випадку використання даного класу МПА в якості альтернативи канонічному МПА Мілі не приводить до істотної економії апаратурних витрат в логічній схемі автомата.

МПА Мура U_1

Для даної структури область ефективного застосування визначимо, виходячи з табл. 5.29. Відзначимо наступне:

1. Для автоматів малої складності виграш досяжний при числі формованих мікрооперацій, близькому до 30. Для автоматів більш високого класу складності це число може становити 200 і вище.

Таблиця 5.29

Область ефективного застосування МПА Мура U_1

Клас склад-ності	Параметр								E^{U_1}
	N	N_d	k_1	k_2	k_3	k_4	k_5	k_6	
1	50	4	0,8	1	0,95	0,75	0,7	0,1	1,168
	30	4	0,9	1	0,95	0,75	0,5	0,1	1,125
	30	4	0,9	1	1	0,75	0,5	1	1,110
	30	4	0,8	1	0,95	0,75	0,5	0,1	1,106
2	30	8	0,9	1	0,95	0,75	0,5	0,1	1,205
	100	8	0,9	1	0,95	0,75	0,5	0,1	1,168
	100	8	0,9	1	0,9	0,75	0,8	0,1	1,103
	90	16	0,9	1	0,97	0,75	0,5	0,1	1,100
3	100	16	0,9	1	0,95	0,75	0,5	0,1	1,137
	200	16	0,5	1	0,95	0,75	0,8	0,1	1,121
	100	8	0,8	1	0,9	0,75	0,5	0,1	1,109
	100	32	0,8	1	0,97	0,75	0,7	0,1	1,106
4	200	16	0,8	1	0,95	0,75	0,5	0,1	1,153
	100	32	0,8	1	0,95	0,75	0,5	0,1	1,105
	100	16	0,8	1	0,91	0,75	0,5	0,1	1,100
	300	8	0,85	1	0,9	0,75	0,5	0,1	1,100

2. Для автоматів 3 і 4 класу складності кількість операцій переходів може досягати 30. Зі зменшенням класу складності це число зменшується.

3. Майже у всіх випадках прийнятне значення ефективності досягається при $k_3 \geq 0{,}95$. Три представлені в таблиці випадки $k_3 = 0{,}9$ можливі за умови $N_d = 8$, тобто 90 % автоматних переходів повинні бути реалізовано за допомогою семи операцій переходів.

4. Неодмінною умовою достатньої ефективності структури є наявність у використовуваній ПЛІС достатнього обсягу блокової пам'яті, що виражається умовою $k_6 \leq 0{,}1$. У табл. 5.29 представлений єдиний випадок, при якому виграш можливий за відсутності блокової пам'яті ($k_6 = 1$), однак супутньою умовою є реалізація всіх без винятку переходів ($k_3 = 1$) за допомогою лише трьох операцій переходів ($N_d = 4$).

5. Середній виграш в апаратурних витратах близький до 10 % і порівняний з виграшем для структури U_1 МПА Мілі (табл. 5.28).

У цілому область ефективного застосування МПА Мура зі структурою U_1 можна охарактеризувати наступним набором відношень: $N \le 200$, $N_d \le 32$, $k_3 \ge 0{,}95$, $k_6 \le 0{,}1$. ефективність, що досягається при цьому, говорить про низьку економію апаратурних витрат при використанні даної структури замість канонічного МПА Мура.

МПА Мілі U_2

Для визначення області ефективного застосування МПА Мілі зі структурою U_2 складемо табл. 5.30. Проаналізуємо вміст таблиці.

Таблиця 5.30

Область ефективного застосування МПА Мілі U_2

Клас склад-ності	Параметр								E^{U_2}
	N	N_d	k_1	k_2	k_3	k_4	k_5	k_6	
1	20	8	0,8	1	0,95	0,75	0,9	0,1	1,158
	20	4	0,8	1	0,75	0,75	0,8	0,1	1,142
	20	8	0,8	1	0,95	0,75	0,7	0,1	1,125
	30	8	0,9	1	0,9	0,75	0,8	0,1	1,103
2	50	16	0,8	1	0,95	0,75	0,7	0,1	1,114
	30	16	0,8	1	0,8	0,75	0,5	0,1	1,107
	30	4	0,8	1	0,6	0,75	0,5	0,1	1,105
	50	8	0,8	1	0,8	0,75	0,7	0,1	1,104
3	50	16	0,8	1	0,75	0,75	0,7	0,1	1,103
	50	8	0,8	1	0,75	0,75	0,5	0,1	1,102
	50	32	0,9	1	0,9	0,75	0,5	0,1	1,100
	50	16	0,8	1	0,8	0,75	0,5	0,1	1,100
4	50	32	0,8	1	0,9	0,75	0,5	0,1	1,126
	50	32	0,8	1	0,8	0,75	0,5	0,1	1,107
	50	16	0,8	1	0,7	0,75	0,6	0,1	1,106
	50	32	0,9	1	0,75	0,75	0,5	0,1	1,100

1. Незалежно від класу складності, необхідна ефективність досягається при $N \le 50$. При подальшому збільшенні N спостерігається різке зниження ефективності, що видно на рис. 5.16, *в.*

2. Можливість реалізувати Z-підсхему в базисі блокової пам'яті ПЛІС, вимагає наявності її достатнього обсягу. Дана вимога виражається в значенні k_6, близькому до нуля. При збільшенні k_6 до значень, близьких до одиниці, витрати

апаратури в блоці ОЧ разом із Z-підсхемою стають порівняними із витратами в блоці СФП канонічного МПА Мілі, що приводить до зменшення E^{U_2} до одиниці.

3. При досить великій кількості використовуваних операцій переходів (16-32) можливе зниження коефіцієнта k_3 до 0,8 і нижче, що може сприяти спрощенню потенційних методів алгебраїчного синтезу даного класу МПА. Проте, навіть для автоматів середньої складності досягнення значення $k_3 = 0{,}6$ можливе лише при $N_d = 3$.

Область ефективного застосування МПА Мілі зі структурою U_2 може бути виражена наступним рядом обмежень: $N \le 50$, $N_d \le 32$, $k_3 \ge 0{,}7$, $k_6 \le 0{,}1$. Ефективність даної структури можна вважати чисельно порівняною з ефективністю двох раніше розглянутих структур.

МПА Мура U_2

Складемо для даної структури табл. 5.31, аналогічну табл. 2.28 – 2.30, та визначимо область ефективного застосування. Відзначимо наступне.

1. Для всіх класів складності припустимий широкий діапазон значень параметра N. При цьому значення інших параметрів залишаються в прийнятному діапазоні.

2. Для всіх класів можливе зниження k_3 до величини 0,3 і нижче, що зазвичай супроводжується відносно невеликим значенням N_d. Це може сприяти спрощенню методів алгебраїчного синтезу і зниженню часу на проектування МПА.

3. Для всіх класів складності можлива реалізація без використання базису блокової пам'яті ПЛІС ($k_6 = 1$). Ефективність структури в цьому випадку забезпечується достатньо високим значенням k_3 (не нижче 0,9).

4. Можливість збільшення зі зростанням класу складності автомата значення N_d до 64 і вище при $k_3 \le 0{,}9$ може сприяти збільшенню кількості ефективних розв'язків задачі алгебраїчного синтезу МПА з ОАП.

Таблиця 5.31

Область ефективного застосування МПА Мура U_2

Клас склад-ності	Параметр								E^{U_2}
	N	N_d	k_1	k_2	k_3	k_4	k_5	k_6	
1	50	8	0,8	1	0,9	0,75	0,5	0,1	1,961
	200	16	0,8	1	0,9	0,75	0,5	0,1	1,193
	1000	8	0,8	1	0,9	0,75	0,5	0,1	1,146
	100	8	0,9	1	0,9	0,75	0,5	1	1,114
	100	6	0,8	5	0,9	0,75	0,5	0,1	1,107
	200	4	0,9	1	0,3	0,75	0,5	0,1	1,100
2	200	32	0,8	1	0,8	0,75	0,5	0,1	1,501
	300	64	0,8	1	0,9	0,75	0,5	0,1	1,251
	100	32	0,8	1	0,9	0,75	0,5	1	1,139
	200	15	0,8	7	0,75	0,75	0,5	0,1	1,124
	100	8	0,8	1	0,2	0,75	0,5	0,1	1,112
	1000	64	0,8	1	0,9	0,75	0,5	0,1	1,107
3	200	64	0,8	1	0,9	0,75	0,5	0,1	1,688
	1000	64	0,8	1	0,9	0,75	0,5	0,1	1,208
	500	128	0,8	1	0,9	0,75	0,5	0,1	1,191
	200	32	0,8	5	0,7	0,75	0,5	0,1	1,178
	200	16	0,8	1	0,25	0,75	0,5	0,1	1,130
	200	64	0,9	1	0,95	0,75	0,4	1	1,103
4	200	32	0,8	1	0,9	0,75	0,5	0,1	2,214
	300	32	0,8	8	0,9	0,75	0,5	0,1	1,444
	500	128	0,8	1	0,9	0,75	0,5	0,1	1,393
	2000	32	0,8	1	0,9	0,75	0,5	0,1	1,164
	200	64	0,8	1	0,9	0,75	0,5	1	1,116
	300	16	0,8	1	0,2	0,75	0,5	0,1	1,101

5. Для кожного класу складності припустимим є використання операцій переходів, складність схемної реалізації яких у 5 і більше раз перевищує складність операцій, розглянутих в п. 5.2.3. Приріст апаратурних витрат, викликаний збільшенням складності операцій переходів, компенсується зменшенням їх кількості, яка дорівнює $N_d - 1$.

Область ефективного застосування МПА Мура зі структурою U_2, з урахуванням зроблених висновків, є досить широкою і в цілому може бути визначена набором наступних відношень: $N \le 1000$, $N_d \in [16;128]$, $k_2 = 1$, $k_3 \le 0{,}9$, $k_6 \le 0{,}1$. Можна відзначити, що для даної структури значення ефективності 1,5 і вище досяжне без істотного обмеження інших параметрів автомата. Одержувана при цьому економія понад 30 % витрат апаратури дозволяє

розглядати МПА Мура зі структурою U_2 в якості високоефективної альтернативи еквівалентному МПА Мура з канонічною структурою.

5.7 Дослідження ефективності синтезу МПА з ОАП в САПР Xilinx Vivado

Внаслідок того, що мікропрограмний автомат є типовою складовою цифрових систем, сучасні САПР компаній-виробників ПЛІС реалізують власні методи синтезу МПА. В п. 1.3 розглянуті основні підходи до опису автоматів мовою VHDL, що підтримуються і рекомендуються компанією Xilinx. В залежності від користувацьких налаштувань САПР Xilinx Vivado виконує синтез МПА для різних критеріїв оптимізації (швидкодія, апаратурні витрати, захищеність від завад тощо. При цьому доцільність застосування з боку проектувальника додаткових методів оптимізації на етапі розробки VHDL-опису є неочевидною та потребує дослідження.

Проведемо дослідження ефективності автоматичного синтезу МПА засобами САПР Xilinx Vivado у порівнянні з синтезом МПА на основі принципу операційного перетворення кодів станів за критерієм апаратурних витрат. За аналогією з виразом (5.1) будемо визначати ефективність МПА з ОАП $E^{U_{ОАП}}$ згідно з виразом

$$E^{U_{ОАП}} = \frac{H^{U_{САПР}}}{H^{U_{ОАП}}}, \tag{5.55}$$

де $H^{U_{САПР}}$ – виражені в LUT-елементах апаратурні витрати, визначені шляхом синтезу заданого МПА засобами САПР Xilinx Vivado за VHDL-описом, розглянутим в п. 1.3; $H^{U_{ОАП}}$ – виражені в LUT-елементах апаратурні витрати, визначені шляхом синтезу еквівалентного МПА з ОАП засобами САПР Xilinx Vivado за VHDL-описом, що відповідає принципу операційного перетворення кодів станів. Таким чином, метою експерименту є обґрунтування доцільності

використання МПА з ОАП в якості альтернативи МПА, синтезованому із використанням вбудованих методів САПР.

В ході дослідження розглянемо п’ять ГСА $\Gamma_{5-1}-\Gamma_{5-5}$, що мають наступні властивості:

1. ГСА відтворює лише функцію переходів автомата і не містить інформації про функцію виходів (усі операторні вершини є пустими). Це дозволяє визначити апаратурні витрати на реалізацію саме функції переходів, хоча й не може вважатися показником ефективності автомата в цілому.

2. ГСА має регулярну структуру, тобто являє собою послідовність однакових фрагментів. Такий підхід, по-перше, спрощує масштабування ГСА шляхом збільшення кількості фрагментів, а по-друге, дозволяє автоматизувати процес генерації VHDL-коду.

Для кожної ГСА розглянемо наступне:

– застосування принципу операційного перетворення кодів станів;

– особливості VHDL-опису МПА з ОАП;

– ефективність використання принципу операційного перетворення кодів станів за виразом (5.55).

ГСА Γ_{5-1}.

ГСА не містить умовних вершин, має цілком лінійну структуру і позначена *M* станами автомата Мура (рис. 5.12).

Найбільш очевидним і простим застосуванням принципу операційного перетворення кодів станів у даному випадку є використання лічильника з функцією інкременту і реалізація МПА з ОАП у вигляді С-автомата. Для ГСА Γ_{5-1} VHDL-опис автомата на лічильнику представлений лістингом А.17.

Згідно з лістингом А.17, в МПА з ОАП використане послідовне кодування станів кодами від 0 до 99 розрядності $R=7$, що дорівнюють значенням індексів станів. Блок архітектури містить два процеси, перший з яких відповідає регістру пам’яті, другий – операційному автомату переходів з єдиною операцією інкременту.

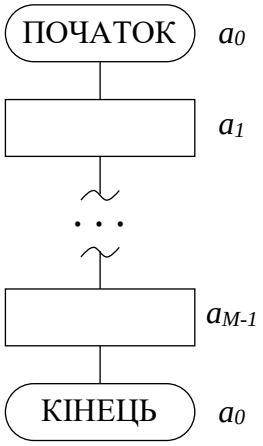


Рисунок 5.12 – Граф-схема алгоритму Γ_{5-1}

В табл. 5.32 наведені результати синтезу МПА, заданого ГСА виду Γ_{5-1} для різної кількості станів M. Стовпчики «one-hot», «sequential», «johnson», «gray», «auto» відповідають однойменним значенням параметру «fsm_extraction» САПР Xilinx Vivado (див. п. 1.3) і містять значення $H^{U_{САПР}}$, отримані із використанням VHDL-опису МПА, аналогічного до лістингу А.6. Стовпчик $H^{U_{ОАП}}$ містить значення апаратурних витрат, отримані для VHDL-опису МПА з ОАП, наведеному на лістингу А.17.

Таблиця 5.32

Результати дослідження ГСА Γ_{5-1}

M	$H^{U_{САПР}}$, LUT					$H^{U_{ОАП}}$, LUT	$E^{U_{ОАП}}$
	one-hot	sequential	johnson	gray	auto		
100	60	9	273	18	60	6	1,5
200	116	15	457	20	15	8	1,88
500	292	18	979	29	292	11	1,63
1000	570	23	–	38	570	12	1,92
2000	1181	27	–	40	1181	13	2,08

Значення в табл. 5.32 вимірюються в LUT-елементах і отримані для ПЛІС марки xc7s6cpga196-2 серії Spartan-7. Усі налаштування Xilinx Vivado, окрім змінюваного параметру «fsm_extraction», обрані за замовченням. Символом «–»

позначені значення, які не були отримані внаслідок неможливого або занадто довгого процесу синтезу. При розрахунку $E^{U_{ОАП}}$ для кожного рядка таблиці використане мінімальне в даному рядку значення $H^{U_{САПР}}$ (позначене затемненим фоном). Відзначимо, що згідно із узагальненням 1 (п. 2.3) для реалізації автоматних переходів припустиме використання будь-яких операцій (не лише інкременту). З огляду на обраний критерій оптимізації, яким є витрати апаратури, доцільно використовувати таку операцію, для якої витрати на реалізацію логічної схеми є якомога меншими.

Перевагою операції інкременту, використаної в VHDL-моделі на лістингу А.17, є можливість формування безперервної послідовності кодів станів максимальною довжиною 2^R. Таку ж можливість надає використання лінійного регістру зсуву зі зворотним зв'язком на базі операції XOR [172]. Такий регістр дозволяє генерувати послідовність з $(2^R - 1)$ унікальних бітових векторів, які можуть бути використані для кодування станів автомата. Для цього зворотний зв'язок має бути організований відповідно до примітивного поліному довжини *R*, де *R* – розрядність коду стану автомата.

Розглянемо фрагмент VHDL-моделі МПА з ОАП для ГСА Γ_{5-1} у випадку $M = 100$, в якій замість операції інкременту використовується зсув зі зворотним зв'язком (лістинг А.18). Блок «entity» в даній моделі співпадає із відповідним блоком в лістингу А.17. Для генерації послідовності кодів станів розрядності $R = 7$ використовується поліном $x^6 + 1$, який в кожному такті роботи автомата формує молодший розряд коду наступного стану як суму за модулем 2 значень шостого та нульового розрядів поточного коду стану [172]:

```
next_state <= state(R-2 downto 0) & (state(6) xor state(0));
```

Для забезпечення генерації послідовності кодів потрібної довжини в якості коду початкового стану a_0 обраний код 1111111_2. Розрахунок показав, що код

кінцевого стану a_{99} в цій безперервній послідовності дорівнюватиме 11110000_2. Дані значення вказані в якості констант у моделі на рис. 5.25.

В табл. 5.33 наведені порівняльні результати синтезу еквівалентних МПА з ОАП для VHDL-моделей, представлених лістингами А.17 (стовбець $H_1^{U_{ОАП}}$) та А.18 (стовбець $H_2^{U_{ОАП}}$). Вміст стовпця $H_1^{U_{ОАП}}$ співпадає із вмістом стовпця $H^{U_{ОАП}}$ в табл. 5.32. Для різних значень *M* регістр зсуву будується на основі різних примітивних поліномів, зазначених у стопці «Поліном».

Таблиця 5.33

Витрати апаратури на реалізацію МПА з ОАП на базі операції інкременту ($H_1^{U_{ОАП}}$) та регістру зсуву ($H_2^{U_{ОАП}}$) для ГСА Γ_{5-1}

M	*R*	Поліном	$H_1^{U_{ОАП}}$, LUT	$H_2^{U_{ОАП}}$, LUT
100	7	x^6+1	6	5
200	8	x^7+x^6+x+1	8	5
500	9	x^8+x^3	11	4
1000	10	x^9+x^2	12	4
2000	11	$x^{10}+x$	13	3

Як можна бачити, витрати апаратури на реалізацію операції переходів на базі примітивного поліному є меншими, ніж у випадку операції інкременту, і суттєво меншими, ніж в результаті синтезу МПА стандартними засобами САПР (табл. 5.32). Порівняно невеликі абсолютні значення апаратурних витрат у даному прикладі є наслідком лінійності ГСА Γ_{5-1}.

ГСА Γ_{5-2}.

ГСА відповідає автомату Мура і має наступну структуру (рис. 5.13):

– після кожної операторної вершини йде умовна вершина;

– в усіх умовних вершинах перевіряється одна й та сама ЛУ x_1;

– при $x_1 = 0$ перехід веде в наступний стан автомата, при $x_1 = 1$ – на десять станів уперед;

– для останніх десяти станів перехід при $x_1 = 1$ веде в стан a_0.

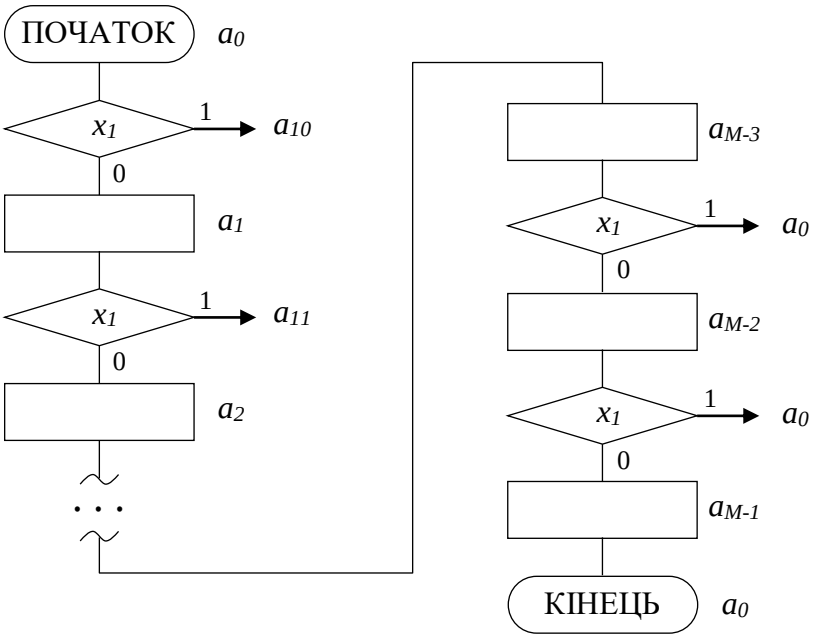


Рисунок 5.13 – Граф-схема алгоритму Γ_{5-2}

В даному випадку очевидним є природна послідовність кодування станів і використання наступних операцій переходів: при $x_1 = 0$ виконується інкремент, при $x_1 = 1$ – операція « + 10 ». Фрагмент VHDL-моделі МПА з ОАП для ГСА Γ_{5-2}, що відображує блок архітектури, представлений лістингом А.19. При цьому опис блоку «entity» збігається із відповідним блоком в лістингах А.17 і в лістингу А.19 не показаний.

Результати досліджень наведені в табл. 5.34, яка за структурою аналогічна табл. 5.32. В кожному рядку комірки з мінімальним значенням $H^{U_{САПР}}$, яке використовується при розрахунку $E^{U_{ОАП}}$ в даному рядку, показані затемненим фоном. Можна відзначити, що у випадку ГСА Γ_{5-2} ефективність МПА з ОАП, що описується лістингом А.19, складає в середньому 1,20, але абсолютні значення виграшу апаратурних витрат є несуттєвими у порівнянні із ресурсоємністю використовуваної мікросхеми ПЛІС. Це свідчить про те, що витрати апаратури на

реалізацію операцій переходів порівняні із витратами на реалізацію МПА табличним способом на базі LUT-елементів.

Таблиця 5.34

Результати дослідження ГСА Γ_{5-2}

M	$H^{U_{САПР}}$, LUT					$H^{U_{ОАП}}$, LUT	$E^{U_{ОАП}}$
	one-hot	sequential	johnson	gray	auto		
100	109	18	267	26	109	13	1,38
200	215	18	759	38	215	15	1,20
500	539	23	3047	56	539	19	1,21
1000	1489	27	–	68	1489	20	1,35
2000	3091	22	–	66	3091	21	1,05

ГСА Γ_{5-3}.

ГСА за структурою схожа на ГСА Γ_{5-2}, але номер стану, в який здійснюється перехід за умовою $x_1 = 1$, генерується у псевдовипадковий спосіб в межах $[0; M-1]$. Загальний вигляд ГСА Γ_{5-3}, відзначеної станами автомата Мура, наведений на рис. 5.14. Застосуємо принцип операційного перетворення кодів станів в наступний спосіб:

– перетворення коду стану за умовою $x_1 = 0$ здійснюється за допомогою регістра зсуву зі зворотнім зв'язком на основі примітивного поліному, як у прикладі для ГСА Γ_{5-1};

– за умовою $x_1 = 1$ переходи реалізуються за допомогою VHDL-оператора «case», як у лістингу А.6.

Фрагмент VHDL-моделі МПА з ОАП для ГСА Γ_{5-3} представлений лістингом А.20. Опис блоку «entity» збігається із відповідним блоком у лістингу А.17. Оператор «case» показаний частково і в дійсності містить M операторів «when». Поліноми, що використовуються в регістрі зсуву при різних значеннях M, наведені в табл. 5.33. При синтезі даної VHDL-моделі параметр «fsm_extraction»

встановлений в «off», що унеможливлює автоматичне застосування будь-яких методів оптимізації МПА з боку САПР Xilinx Vivado.

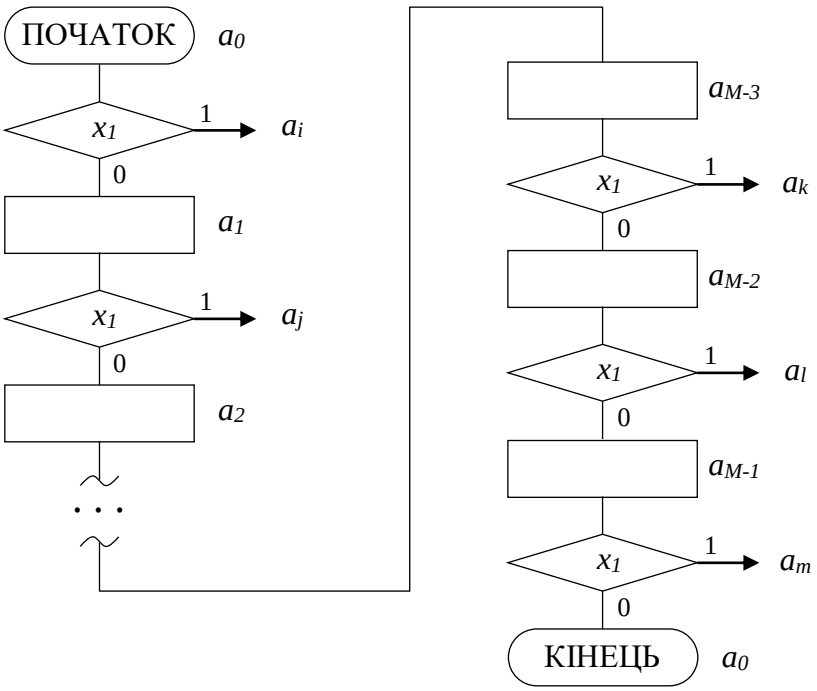


Рисунок 5.14 – Граф-схема алгоритму Γ_{5-3}

Результати дослідження ГСА Γ_{5-3} наведені в табл. 5.35. Значення $H^{U_{САПР}}$ отримані для моделі, представленої лістингом А.6, значення $H^{U_{ОАП}}$ – для моделі, представленої лістингом А.20.

Таблиця 5.35

Результати дослідження ГСА Γ_{5-3}

M	$H^{U_{САПР}}$, LUT					$H^{U_{ОАП}}$, LUT	$E^{U_{ОАП}}$
	one-hot	sequential	johnson	gray	auto		
100	112	44	419	43	112	21	2,05
200	250	94	1642	92	250	41	2,24
500	668	239	–	239	668	99	2,41
1000	1454	511	–	510	1454	190	2,68
2000	3334	1177	–	1166	3334	407	2,86

Як видно з табл. 5.35, у випадку ГСА Γ_{5-3}, при використанні автоматичного синтезу МПА засобами САПР найменші апаратурні витрати отримані із для методу кодування «gray» (показані затемненим фоном), тоді як для ГСА Γ_{5-1} та Γ_{5-2} таким методом є «sequential». Таким чином, значення $E^{U_{OAП}}$ в табл. 5.35 отримані як відношення значень $H^{U_{САПР}}$ зі стовпця «gray» до відповідних значень зі стовпця $H^{U_{OAП}}$.

ГСА Γ_{5-4}.

ГСА за структурою схожа на ГСА Γ_{5-2}, але в кожній умовній вершині перевіряється різна логічна умова, від x_1 до x_{M-1}. Загальний вигляд ГСА Γ_{5-4}, відзначеної станами автомата Мура, наведений на рис. 5.15.

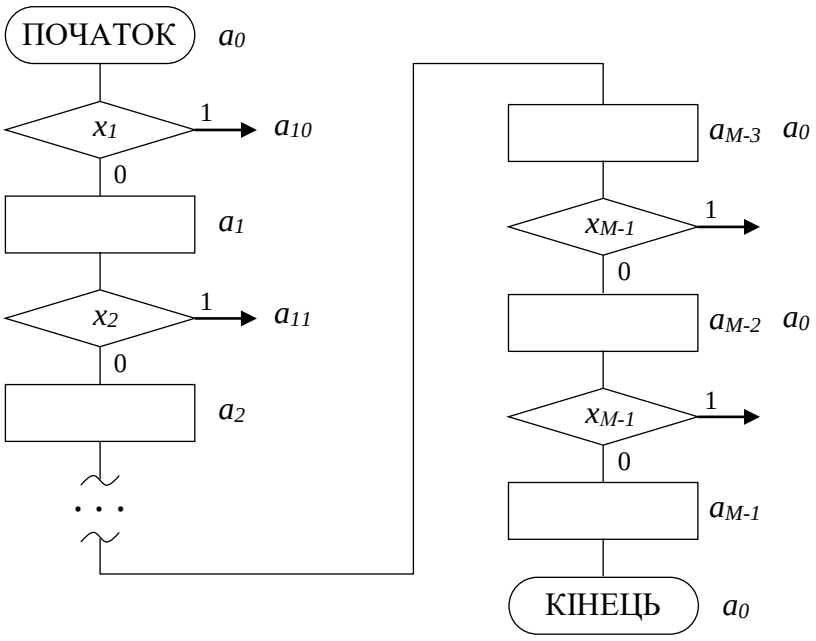


Рисунок 5.15 – Граф-схема алгоритму Γ_{5-4}

Організуємо МПА з ОАП відповідно до структури U_3 (рис. 3.9), що передбачає застосування методу заміни вхідних змінних. Відповідно до цієї структури побудуємо VHDL-модель МПА з ОАП у вигляді трьох процесів, що відповідають наступним елементам структури U_3:

– регістр пам’яті;

– M-підсхема;

– операційна частина ОАП та Z-підсхема.

VHLL-модель для випадку $M = 100$ представлена лістингом А.21.

Результати дослідження ГСА Γ_{5-4} наведені в табл. 5.36. Найбільш ефективним вбудованим методом кодування станів виявився метод «gray». Втім, навіть в цьому випадку МПА з ОАП демонструє ефективність від 2,49 до 4,26, тобто кількакратний виграш у витратах апаратури.

Таблиця 5.36

Результати дослідження ГСА Γ_{5-4}

M	$H^{U_{САПР}}$, LUT					$H^{U_{ОАП}}$, LUT	$E^{U_{ОАП}}$
	one-hot	sequential	johnson	gray	auto		
100	155	130	412	127	155	51	2,49
200	311	269	1219	254	311	72	3,54
500	799	656	3978	631	799	155	4,07
1000	1958	1368	–	1240	1958	318	3,90
2000	4051	2767	–	2498	4051	586	4,26

ГСА Γ_{5-5}.

ГСА за структурою схожа на ГСА Γ_{5-3}, але в кожній умовній вершині перевіряється різна логічна умова, від x_1 до x_M. Загальний вигляд ГСА Γ_{5-5} наведений на рис. 5.16.

У VHDL-моделі МПА з ОАП, що імплементує ГСА Γ_{5-5}, комбінуються прийоми, застосовані в моделях для ГСА Γ_{5-3} та Γ_{5-4}. Так, використання методу заміни вхідних змінних додає в модель процес, що реалізує *M*-підсхему (як в моделі на рис. 5.31). В якості операції переходу використовується генерація кодів станів на основі регістру зсуву зі зворотним зв'язком (як в моделі на лістингу А.20). В операторі процесу, що імплементує ОАП, замість змінних із множини *X* використовується єдина змінна *p* (лістинг А.22).

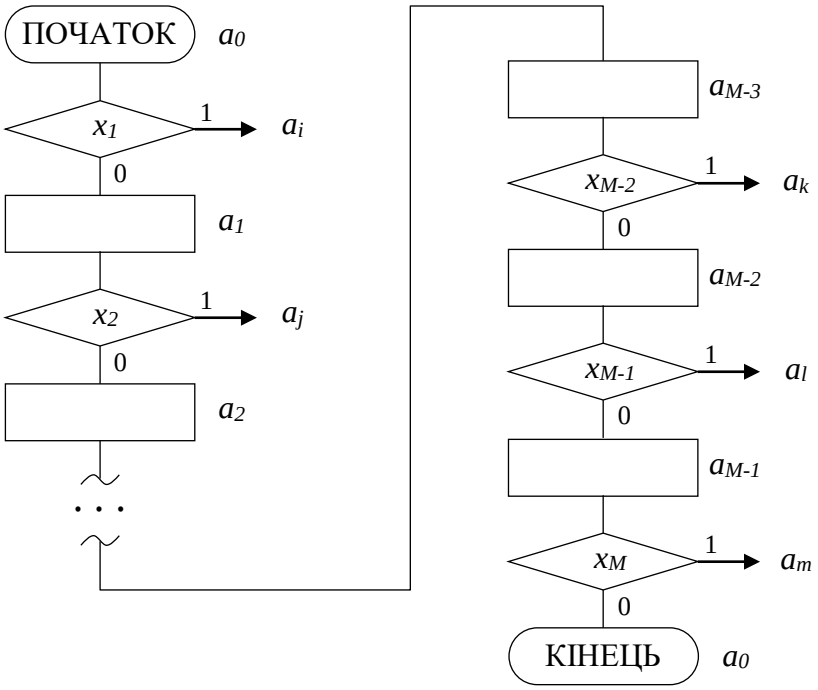


Рисунок 5.16 – Граф-схема алгоритму Γ_{5-5}

Результати дослідження ефективності МПА з ОАП для ГСА Γ_{5-5} наведені в табл. 5.37. Серед вбудованих в САПР методів кодування станів найбільш оптимальним виявився метод «one-hot». Для $M > 200$ не були отримані значення $H^{U_{САПР}}$ для методу «johnson», однак, зважаючи на результати, отримані для попередніх ГСА, слід очікувати суттєве зростання $H^{U_{САПР}}$ із збільшенням *M*, що не надасть переваг у порівнянні із методом «one-hot».

Таблиця 5.37

Результати дослідження ГСА Γ_{5-5}

M	$H^{U_{САПР}}$, LUT					$H^{U_{ОАП}}$, LUT	$E^{U_{ОАП}}$
	one-hot	sequential	johnson	gray	auto		
100	145	166	589	155	145	58	2,50
200	298	357	1967	348	298	117	2,54
500	831	1048	–	995	831	225	3,69
1000	1782	2286	–	2281	1782	454	3,92
2000	4050	5123	–	4922	4050	931	4,35

За результатами дослідження ГСА $\Gamma_{5-1}-\Gamma_{5-4}$ відзначимо наступне.

1. Використання принципу операційного перетворення кодів станів при проектуванні МПА в базисі ПЛІС за допомогою сучасних САПР з метою зменшення апаратурних витрат є доцільним, оскільки в певних випадках дозволяє знизити апаратурні витрати при реалізації функції переодів у порівнянні із використанням методів синтезу МПА, вбудованих в САПР.

2. МПА з ОАП припускає використання додаткових методів оптимізації апаратурних витрат. Наприклад, застосування методу заміни вхідних змінних може бути корисним у випадку великої кількості вхідних сигналів, як у випадках ГСА Γ_{5-4} та Γ_{5-5}.

3. VHDL-моделі, отримані для МПА з ОАП, заданих ГСА $\Gamma_{5-1}-\Gamma_{5-4}$, слід розглядати не як оптимальні рішення задачі алгебраїчного синтезу, а лише як формальні рішення. Не слід виключати, для кожної ГСА може бути знайдене рішення із більш високим значенням ефективності.

4. Окрім методів оптимізації схеми, що реалізує функцію переходів автомата, слід очікувати на ефективність використання методів оптимізації схеми автомата в частині функції виходів. Деякі з цих методів, такі як кодування наборів мікрооперацій, нестандартне представлення термів, використання класів сумісних мікрооперацій та інші, розглянуті в [29, 31, 34, 36, 58, 68, 69, 70].

5.8 Висновки до п’ятого розділу

1. В п’ятому розділі вирішене наукове завдання дослідження та визначення області ефективного застосування розроблених структур мікропрограмних автоматів з оптимізованими витратами апаратури.

2. Ефективність структур мікропрограмного автомата з операційним автоматом переходів, що пропонуються в дисертаційній роботі, може бути визначена як відношення величини апаратурних витрат на реалізацію логічної схеми МПА з канонічною структурою до величини апаратурних витрат на реалізацію еквівалентного МПА з ОАП. Витрати апаратури в кожній

досліджуваній структурі визначаються як сума витрат на реалізацію окремих блоків даної структури.

3. Окремі блоки в канонічній структурі МПА та структурах МПА з ОАП $U_1 - U_4$ мають схемну архітектуру, відповідну до архітектури типових функціональних блоків, до яких належать комбінаційні схеми, регістри, блоки, що реалізують арифметико-логічні операції, модулі пам'яті тощо. Це дає можливість проводити дослідження апаратурних витрат не для структурних блоків МПА, а для відповідних їм функціональних блоків, поширивши отримані результати на структурні блоки МПА.

4. У п. 5.2 за допомогою VHDL-моделювання проведені дослідження апаратурних витрат у типових функціональних блоках цифрових пристроїв. На основі експериментальних даних отримані апроксимуючі аналітичні залежності апаратурних витрат у типових блоках від їхніх параметрів.

5. На підставі аналітичних залежностей, отриманих в п. 5.2, і параметрів МПА, визначених у п. 5.3, в п. 5.4 отримані аналітичні вирази для визначення апаратурних витрат у МПА з канонічною структурою та МПА з ОАП зі структурами U_1 та U_2. Отримані окремі вирази для автоматів Мілі і Мура.

6. З використанням аналітичних виразів, отриманих у п. 5.4, в п. 5.5 проведені дослідження залежності ефективності МПА з ОАП Мілі й Мура зі структурами U_1 та U_2 від різних параметрів МПА. Для всіх досліджуваних структур розглянуті чотири класи складності автоматів, що різняться кількістю станів і переходів (табл. 5.18). У всіх випадках апаратурні витрати у виразах (5.53) і (5.54) визначені при однакових значеннях параметрів порівнюваних структур.

7. При значеннях параметрів, обраних в п. 5.5.1, ефективність МПА Мілі зі структурою U_1 становить близька 1,02 (наприклад, табл. 5.19, стовпець N=100), що дозволяє вважати витрати апаратури на її реалізацію в загальному випадку порівняними з витратами на реалізацію канонічного МПА Мілі. Значення ефективності, що дорівнює 1,1 і вище (табл. 5.28), досягається за умови реалізації всіх автоматних переходів за допомогою операцій переходів ($k_3 \geq 0{,}99$), що може

виявитися неможливим при відносно малому числі операцій переходів (стовпець N_d). Також неодмінною умовою ефективності даної структури є мале число мікрооперацій, формованих автоматом (стовпець *N*).

8. Ефективність МПА Мура зі структурою U_1 при середніх значеннях параметрів МПА (п. 5.5.1) близька до 1,04 (табл. 5.19). З урахуванням можливої погрішності експериментальних досліджень, виконаних у п. 5.2, дане значення говорить про відсутність істотного виграшу в апаратурних витратах у порівнянні з канонічним МПА Мура. Одержання $E^{U_1} \geq 1{,}1$ (табл. 5.20) можливе при числі операцій переходів не більше 30 (стовпець N_d) і операційній реалізації не менш ніж 90 % автоматних переходів (стовпець k_3). Також обов'язковою умовою для досягнення $E^{U_1} \geq 1{,}1$ є використання базису блокової пам'яті ПЛІС, використання якої для реалізації блоків автомата не приводить до росту апаратурних витрат, вимірюваних в LUT-елементах (стовпець k_6). Перевагою даної структури є можливість одержання $E^{U_1} \geq 1{,}1$ при числі формованих мікрооперацій 100 і вище (стовпець *N*), що неможливе у випадку МПА Мілі зі структурою U_1 (табл. 5.28).

9. Структура U_2 у випадку автомата Мілі при середніх значеннях параметрів автомата (п. 5.5.1) має ефективність, порівнянну з ефективністю структури МПА Мура U_1 (табл. 5.19, стовпець *N*=100). Згідно з табл. 5.30, одержання для даної структури значення ефективності вище 1,1 можливе лише при кількості мікрооперацій не більше 50 (стовпець *N*) незалежно від класу складності МПА. На відміну від МПА Мілі і Мура зі структурою U_1, у даній структурі досягнення $E^{U_2} \geq 1{,}1$ можливе при меншій частці переходів, що реалізовані операційним способом (табл. 5.30, стовпець k_3), яка зазвичай не перевищує 90 % від загального числа переходів автомата. Оскільки Z-підсхема в даній структурі може бути реалізована в базисі блокової пам'яті, наявність цього базису у використовуваній ПЛІС є обов'язковою умовою ефективності структури (табл. 5.30, стовпець k_6).

10. МПА Мура зі структурою U_2 при значеннях параметрів МПА, обраних у п. 5.5.1, дозволяє одержати значення ефективності E^{U_2}, залежно від класу складності автомата, в діапазоні від 1,25 до 1,58, що відповідає виграшу в апаратурних витратах від 20 % до 37 % у порівнянні з канонічним МПА Мура. Вміст табл. 5.31 показує, що на певних наборах параметрів задовільні значення ефективності для даної структури можуть бути отримані в наступних випадках:

– при великій кількості формованих мікрооперацій (N >1000);

– при використанні великої кількості операцій переходів (N_d > 100);

– при використанні ресурсномістких операцій переходів ($k_2 \geq 5$);

– при малій частці переходів, реалізованих операційним способом ($k_3 \leq 0{,}3$);

– при відсутності блокової пам'яті на кристалі ПЛІС ($k_6 = 1$).

У деяких випадках (табл. 5.19, 5.25, 5.26) можливе одержання багаторазового виграшу в апаратурних витратах, що для інших структур неможливе в досліджених діапазонах зміни параметрів. Ці та інші результати досліджень дозволяють вважати дану структуру найбільш ефективною серед структур, досліджених в дисертаційній роботі.

11. Дослідження запропонованих у розділі 3 структур МПА з ОАП U_3 та U_4 може бути виконане аналогічно проведеному дослідженню структур U_1 та U_2. З урахуванням методів оптимізації, що лежать в основі структур U_3 та U_4, справедливо припускати більш високу ефективність даних структур за критерієм апаратурних витрат у порівнянні із дослідженими структурами U_1 та U_2.

12. Імплементація принципу операційного перетворення кодів станів МПА засобами сучасних САПР цифрових систем дозволяє отримати більш оптимальні за критерієм апаратурних витрат схеми у порівнянні із використанням методів синтезу МПА, вбудованих в САПР. Це актуалізує науково-практичне завдання автоматизації процесу проектування запропонованих структур МПА як у рамках існуючих САПР, так і у вигляді окремих програмних систем.

ВИСНОВКИ

В дисертаційній роботі отримано нове рішення важливої наукової проблеми розробки, обґрунтування і дослідження теоретичних основ, структур, моделей і методів, спрямованих на зменшення апаратурних витрат в схемі мікропрограмного автомата за рахунок адаптації схеми автомата до характеристик імплементованого алгоритму керування. Відповідно до сформульованої мети в ході досліджень були отримані такі основні наукові та практичні результати:

1. Виконано аналіз сучасних теоретико-методичних та практичних концепцій синтезу і оптимізації цифрових пристроїв керування, який дозволив визначити основні класи пристроїв керування та сформулювати їх переваги та недоліки. Відзначено, що реалізація пристрою керування у вигляді мікропрограмного автомата характеризується максимальною швидкодією і максимальними витратами апаратури у порівняні з іншими класами пристроїв керування. Це, з одного боку, дозволяє вважати мікропрограмний автомат найбільш перспективним способом імплементації пристроїв керування сучасних цифрових систем, для яких швидкодія часто є визначальним фактором ефективності у зв'язку із високою складністю алгоритмів керування, з іншого боку – актуалізує наукову проблему мінімізації апаратурних витрат в логічній схемі МПА з метою зменшення вартості цифрових систем та покращення інших споріднених характеристик.

Проведені дослідження та їх аналіз дозволили визначити основні сучасні напрями оптимізації цифрових пристроїв керування. До них належать багаторівневе перетворення логічних сигналів, спеціальне кодування станів автомата, використання особливостей елементного базису та інші. У зв'язку з цим, зменшення апаратурних витрат в логічній схемі мікропрограмного автомата є важливою проблемою, яка породжує нові напрямки прикладних досліджень зі створення нових структур і методів синтезу мікропрограмних автоматів з оптимізованими витратами апаратури.

2. Вперше запропоновано концепцію перетворення кодів станів МПА за допомогою кінцевої множини арифметико-логічних операцій (принцип операційного перетворення кодів станів). Її застосування приводить до представлення схеми формування переходів автомата у вигляді операційного автомата, кожна операція якого використовується для реалізації окремої підмножини мікропрограмних переходів. Такий підхід сприяє зменшенню апаратурних витрат при реалізації функції переходів за рахунок багаторазового використання функціональних блоків операційного автомата.

3. Вперше виконано формалізацію принципу операційного перетворення кодів станів на основі математичного апарата універсальних алгебр. Функцію переходів МПА запропоновано представляти як кінцеву множину часткових функцій переходів, із кожною з яких ототожнюється власний закон перетворення певної підмножини кодів станів автомата. Враховано різні форми представлення функції переходів: абстрактну, структурну та проміжну, що виражається у завданні відповідних алгебр. Представлення часткових функцій переходів у вигляді підалгебр відповідних алгебр дозволяє визначити тотожність часткових функцій різних рівнів у вигляді ізоморфізмів.

4. На основі формалізованого представлення принципу операційного перетворення кодів станів вперше розроблені:

4.1. Вперше запропоновано структуру мікропрограмного автомата з операційним автоматом переходів, в якому схема формування переходів організована на базі операційного автомата переходів, що реалізує множину часткових функцій переходів. Функцією операційного автомата переходів є формування коду стану переходу автомата в залежності від його поточного стану та вхідних сигналів за допомогою кінцевої множини арифметико-логічних операцій. Такий підхід сприяє виграшу в апаратурних витратах за рахунок того, що витрати апаратури в окремому функціональному блоці операційного автомата переходів не залежать від кількості мікропрограмних переходів, що реалізуються за допомогою даного блока.

4.2. Вперше запропоновано структуру операційної частини як складової операційного автомата переходів у вигляді сукупності операційних блоків, кожний з яких являє собою схемну реалізацію певної часткової функції переходів автомата. Доведено, що принцип операційного перетворення кодів станів припускає використання в якості вхідних сигналів функціональних блоків коду поточного стану та вхідних сигналів автомата.

4.3. Вперше розроблено та досліджено структуру операційного автомата переходів, складовими якої є операційна частина та регістр пам’яті МПА в якості єдиної регістрової схеми операційного автомата. Результатом роботи операційного автомата переходів є код стану переходу МПА, який формується шляхом мультиплексування виходів функціональних блоків операційної части операційного автомата під керуванням коду операції переходів. Той факт, що для формування коду операції переходів використовується код поточного стану разом із вхідними сигналами МПА, свідчить про можливість реалізації кожного мікропрограмного переходу за допомогою окремої операції.

Наявність в операційному автоматі переходів єдиного регістру, який одночасно виступає в якості регістру вхідних даних та регістру результату, дозволила класифікувати цей автомат одночасно як операційний автомат K-, I- та M-типів. Показано, що для зменшення апаратурних витрат в операційному автоматі переходів можуть бути використані відомі методи формування еквівалентних мікрооперацій та узагальнених операторів.

4.4. Вперше запропоновано математичну модель мікропрограмного автомата з операційним автоматом переходів. Модель представляє собою систему изоморфізмів алгебр, складність якої визначається кількістю часткових функцій переходів автомата. Кожний ізоморфізм системи встановлює тотожність різних форм представлення часткової функції переходів, а їх система забезпечує унікальність і однозначність кодів станів, що є обов’язковою умовою синтезу автомата.

4.5. Запропоновано нову структуру мікропрограмного автомата з операційним автоматом переходів, яка передбачає зіставлення операцій переходів

окремим станам автомата на відміну від їх зіставлення окремим мікропрограмним переходам. Особливістю даної структури є менша кількість входів схеми формування кодів операцій переходів, що дозволяє реалізувати дану схему в базисі запам'ятовуючих пристроїв. Це дозволяє при використанні ПЛІС типу FPGA задіяти модулі блокової пам'яти кристалу ПЛІС, вивільнивши частину LUT-елементів для реалізації інших блоків автомата або цифрової системи. У проектах, в яких модульна пам'ять не використовується, даний підхід приводить до суттєвої економії ресурсів кристалу ПЛІС, що рівнозначно зменшенню апаратурних витрат на реалізацію логічної схеми МПА.

4.6. Досліджено особливості реалізації функції виходів в МПА Мілі і Мура з операційним автоматом переходів. Проаналізовані операційний та канонічний способи реалізації функції виходів, для яких сформульовані умови доцільності використання.

5. Сформульовано нову наукову задачу алгебраїчного синтезу мікропрограмного автомата з операційним автоматом переходів. Для її вирішення запропоновано методологічну базу, яка містить наступні складові, що пропонуються вперше:

5.1. Метод структурного представлення процесу синтезу мікропрограмного автомата з операційним автоматом переходів, згідно з яким будь-який метод синтезу представляється у вигляді сукупності окремих етапів та причинно-наслідкових зв'язків між ними. Кожна із запропонованих структур процесу синтезу може відповідати кільком методам синтезу і позиціонується як спосіб їх класифікації.

5.2. Метод використання транзитних станів, що дозволяє за певних умов уникнути застосування додаткових операцій переходів за рахунок додавання проміжних станів, зменшуючи, таким чином, апаратурні витрати в схемі операційного автомата переходів та схемі формування кодів операцій переходів.

5.3. Метод урахування імовірностей істинності логічних умов, використання якого дозволяє оцінити вплив додавання транзитних станів на середню кількість мікрокоманд, що виконуються за один цикл мікропрограми.

5.4. Метод збільшення розрядності кодів станів автомата, що передбачає використання більшої за мінімально достатню кількості двійкових розрядів і дозволяє збільшити кількість способів кодування станів та припустиму кількість транзитних станів при виконанні алгебраїчного синтезу мікропрограмного автомата з операційним автоматом переходів.

6. Отримали подальший розвиток методи заміни вхідних змінних та зменшення кількості істотних вхідних змінних внаслідок їх застосування та адаптації до мікропрограмного автомата з операційним автоматом переходів. В результаті отримані нові модифіковані структури МПА з операційним автоматом переходів, які можуть мати додаткове зменшення апаратурних витрат в схемі формування кодів операцій переходів та в операційному автоматі переходів у порівнянні зі структурами, що не використовують дані методи.

7. Вперше розроблено метод алгебраїчного синтезу мікропрограмного автомата з операційним автоматом переходів, який передбачає пошук формальних розв’язків задачі алгебраїчного синтезу виконується шляхом часткового перебору варіантів. Метод припускає збільшення кількості станів автомата за рахунок транзитних станів, враховує імовірності істинності логічних умов та реалізує Алгоритмічна спрямованість методу дозволяє його використання в системах автоматизованого проектування пристроїв керування цифрових систем.

8. Вперше проведені експериментальні дослідження ефективності запропонованих структур мікропрограмних автоматів з операційним автоматом переходів, які дозволили визначити їх переваги згідно критерію апаратурних витрат у порівнянні з раніше відомими структурами. В якості методології дослідження використано комп’ютерне моделювання із використанням мови опису апаратури VHDL, яке дозволило отримати аналітичні залежності апаратурних витрат в логічних схемах досліджуваних структур мікропрограмних автоматів від параметрів автомата та умов проектування. Для дослідження апаратурних витрат в логічній схемі мікропрограмного автомата використано удосконалений метод, який полягає в тому, що структурні блоки автомату та їх параметри ототожнюються зі стандартними функціональними блоками цифрових

систем, а результати експериментальних досліджень, отримані для стандартних функціональних блоків, розповсюджуються на елементи досліджених структур МПА.

Дослідження показали, що найбільш ефективною серед запропонованих структур виявився мікропрограмний автомат Мура з операційним автоматом переходів, в якому операції переходів зіставляються окремим станам автомата. Визначено, що для цієї структури виграш в апаратурних витратах у порівнянні із мікропрограмним автоматом з канонічною структурою сягає в середньому 20-37 %, а за певних умов може перевищувати 50 %, що еквівалентне двократному зменшенню апаратурних витрат. Також визначено область ефективного застосування запропонованих структур мікропрограмних автоматів як сукупність діапазонів параметрів автомата.

Дослідження можливостей САПР Xilinx Vivado з автоматичного синтезу МПА за VHDL-моделлю показало, що реалізація функції переходів автомата із використанням методів, вбудованих в САПР, може бути менш ефективною за критерієм апаратурних витрат у порівнянні із використанням структур і методів синтезу мікропрограмних автоматів з операційним автоматом переходів. Це свідчить про доцільність та ефективність застосування існуючих методів оптимізації мікропрограмних автоматів при проектування пристроїв за допомогою сучасних САПР цифрових систем.

9. Дисертаційна робота виконана відповідно до планів науково-дослідних робіт Донецького національного університету імені Василя Стуса в рамках таких держбюджетних тем: НДР «Дослідження ефективності мікропрограмного автомата з операційним автоматом переходів» (номер ДР 0117U004097); НДР «Методологічні аспекти синтезу мікропрограмного автомата з операційним автоматом переходів» (номер ДР 0117U004098).

10. Практичні результати, що отримано, підтверджені актами впровадження та доводять коректність теоретичних положень дисертаційної роботи, високу ефективність розроблених структур, моделей, методів та методології. Результати дисертаційної роботи впроваджені: на підприємстві ТОВ «С-інжинірінг»; в

науково-технічному спеціальному конструкторському бюро «Полісвіт» державного науково-виробничого підприємства «Об’єднання "Коммунар"»; в освітньому європейському проекті ERASMUS+ «ALLIOT» – Internet of Things: Emerging Curriculum for Industry and Human Applications; у навчальному процесі кафедри комп’ютерних інтелектуальних систем та мереж Одеського національного політехнічного університету при викладанні дисциплін «Технології проектування комп’ютерних систем» та «Проектування і діагностика систем критичного застосування».

Таким чином, мета і завдання дисертаційної роботи виконані в повному обсязі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Автоматы. Сборник статей под редакцией К. Шеннона и Д. Маккарти. М.: Издательство иностранной литературы, 1956. 404 с.
2. Айзерман М. А., Гусев Л. А., Розоноэр Л. И., Смирнова И. М., Таль А. А. Логика, автоматы, алгоритмы. М.: Физматгиз, 1963. 556 с.
3. Алгебраическая теория автоматов, языков и полугрупп / под ред. М. Арбиба; пер. с англ. М.: Статистика, 1975. 335 с.
4. Алексеенко А. Г., Шагурин И. И. Микросхемотехника: учеб. пособие для вузов. 2-е изд., перераб. и доп. М.: Радио и связь, 1990. 496 с.
5. Алферова З. В. Теория алгоритмов. М.: Статистика, 1973. 164 с.
6. Аляев А. Ю., Тюрин С. Ф. Дискретная математика и математическая логика. М.: Финансы и статистика, 2006. 368 с.
7. Амосов В. В. Схемотехника и средства проектирования цифровых устройств. СПб.: БХВ-Петербург, 2007. 560 с.
8. Ангер С. Асинхронные последовательностные схемы. М.: Наука, 1977. 298 с.
9. Антонов А. П. Язык описания цифровых устройств AlteraHDL. Практический курс. М.: ИП РадиоСофт, 2001. 224 с.
10. Артамонов В. А., Салий В. Н., Скорняков Л. А. Общая алгебра. Т. 2. М.: Наука, Гл. ред. физ.-мат. лит., 1991. 480 с.
11. Бабаков Р. М. Алгебраический синтез микропрограммного автомата с операционным автоматом переходов. *Информационные технологии и компьютерная инженерия*. 2017. № 39. Т. 2. С. 35–41.
12. Бабаков Р. М. Исследование аппаратурных затрат в микропрограммном автомате с операционным автоматом переходов. *Радиоэлектроника, информатика, управление*. 2017. № 4. С. 106–115.
13. Бабаков Р. М. Математическая модель микропрограммного автомата с операционным автоматом переходов. *Наукові праці Донецького національного технічного університету. Серія: Інформатика, кібернетика та обчислювальна техніка*. Красноармійськ: ДВНЗ «ДонНТУ». 2016. № 1

(22). С. 54–57.

14. Бабаков Р.М. Методологические аспекты синтеза микропрограммных автоматов с операционным автоматом переходов. *Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: технічні науки*. 2019. № 2. С. 82–86.
15. Бабаков Р. М. Микропрограммный автомат с операционным автоматом переходов. *Тези доповідей Міжнародної наукової конференції «Сучасна інформатика: проблеми, досягнення та перспективи розвитку», присвяченої 60-річчю заснування Інституту кібернетики імені В.М. Глушкова НАН України* (м. Київ, 13–15 грудня 2017 р.) Київ, 2017. С. 4–6.
16. Бабаков Р. М. Операционное преобразование кодов состояний в микропрограммном автомате: монография. Винница: «ТВОРИ», 2019. 208 с.
17. Бабаков Р. М. Обобщение математической модели микропрограммного автомата на счетчике. *Радиоэлектроника, информатика, управление*. 2018. № 1. С. 100–109.
18. Бабаков Р. М. Организация композиционных микропрограммных устройств управления с полностью ассоциативной кэш-памятью. *Наукові праці Донецького національного технічного університету. Серія: Інформатика, кібернетика та обчислювальна техніка*. Вип. 39. Донецьк: ДонДТУ. 2002. С. 122–127.
19. Бабаков Р. М. Промежуточная алгебра переходов в микропрограммном автомате. *Радиоэлектроника, информатика, управление*. 2016. № 1. С.64–73.
20. Бабаков Р. М. Синтез логической схемы микропрограммного автомата с операционным автоматом переходов. *Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: технічні науки*. 2018. № 4. С. 96–99.
21. Бабаков Р. М. Синтез микропрограммного автомата с операционным автоматом переходов методом полного перебора. *Матеріали Всеукраїнської науково-практичної конференції з міжнародною участю «Інформатика та системні науки (ІСН-2017)* (м. Полтава, 16–18 березня

2017 р.) Полтава: ПУЕТ, 2017. С. 23–25.

22. Бабаков Р. М. Синтез микропрограммного автомата с операционным автоматом переходов методом полного перебора. *Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: технічні науки*. 2018. № 1. С. 70–74.
23. Бабаков Р. М. Урахування імовірності станів у мікропрограмному автоматі з операційним автоматом переходів. *Наукові праці Вінницького національного технічного університету*. 2017. № 2. URL: https://praci.vntu.edu.ua/index.php/praci/article/view/505/500
24. Бабаков Р. М. Формальное решение задачи алгебраического синтеза микропрограммного автомата с операционным автоматом переходов. *Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: технічні науки*. 2018. № 2. С. 103–107.
25. Бабаков Р. М., Самир Нахлави, Зайцев В. В. Синтез двухуровневой схемы автомата Мура на счетчике. *Сборник научных трудов ДонДТУ. Серия «Проблемы моделирования и автоматизации проектирования динамических систем»*. Выпуск 10. Донецк: ДонДТУ, 1999. С. 301–305.
26. Бабаков Р. М., Ярош И. В. Использование транзитных состояний в микропрограммном автомате с операционным автоматом переходов. *Наукові праці Донецького національного технічного університету. Серія: Обчислювальна техніка та автоматизація*. Красноармійськ: ДВНЗ «ДонНТУ». 2016. № 1 (29). С. 56–64.
27. Бабаков Р. М., Ярош И. В. Операционный автомат переходов. *Наукові праці Донецького національного технічного університету. Серія: Обчислювальна техніка та автоматизація*. Красноармійськ: ДВНЗ «ДонНТУ». 2015. № 1 (28). С. 33–40.
28. Бабаков Р. М., Ярош И. В. Формирование кодов операций переходов в микропрограммном автомате с операционным автоматом переходов. *Наукові праці Донецького національного технічного університету. Серія: Інформатика, кібернетика та обчислювальна*

техніка. Красноармійськ: ДВНЗ «ДонНТУ». 2015. Випуск 1 (20). С. 11–16.

29. Баранов С. И. Синтез микропрограммных автоматов. Л.: Энергия, 1979. 232 с.
30. Баранов С. И., Баркалов А. А. Микропрограммирование: принципы, методы, применения. *Зарубежная радиоэлектроника*. 1984, № 5. С. 3–29.
31. Баранов С. И., Скляров В. А. Цифровые устройства на программируемых БИС с матричной структурой. М.: Радио и связь, 1986. 272 с.
32. Баркалов А. А, Бабаков Р. М. Организация устройств управления с операционной адресацией. *Управляющие системы и машины*. 2008. № 6. С. 34–39.
33. Баркалов А. А. Микропрограммное устройство управления как композиция автоматов с программируемой и жесткой логикой. *Автоматика и вычислительная техника*. 1983. № 4. С. 36–41.
34. Баркалов А. А. Разработка формализованных методов структурного синтеза композиционных автоматов: дисс. ... д-ра техн. наук: 05.13.08. Донецк, 1994. 301 с.
35. Баркалов А. А. Синтез операционных устройств. Донецк: РВА ДонНТУ, 2003. 306 с.
36. Баркалов А. А. Синтез устройств управления на программируемых логических устройствах. Донецк: ДонНТУ, 2002. 262 с.
37. Баркалов А. А., Бабаков Р. М. Алгебраическая интерпретация микропрограммного автомата с операционным автоматом переходов. *Кибернетика и системный анализ*. 2016. № 2. С. 22–29.
38. Баркалов А. А., Бабаков Р. М. Модификация микропрограммного автомата с операционным автоматом переходов и заменой входных переменных. *Управляющие системы и машины*. 2017. № 6. С. 35–40.
39. Баркалов А. А., Бабаков Р. М. Операционная реализация функции выходов микропрограммного автомата. *Управляющие системы и машины*. 2017. № 3. С. 57–62.
40. Баркалов А. А., Бабаков Р. М. Операционное формирование кодов состояний в микропрограммных автоматах. *Кибернетика и системный*

анализ. 2011. № 2. С. 21–26.

41. Баркалов А. А., Бабаков Р. М. Операционное формирование переходов в управляющих автоматах. *Матеріали доповідей III Міжнародної науково-практичної конференції молодих учених, аспірантів, студентів «Сучасна інформаційна Україна: інформатика, економіка, філософія»* (м. Донецьк, 14–15 травня 2009 р.) Донецьк, 2009. Т. 1. С. 18–20.
42. Баркалов А. А., Бабаков Р. М. Операционный автомат переходов с дополненным множеством операций переходов. *Наукові праці Донецького національного технічного університету. Серія «Інформатика, кібернетика і обчислювальна техніка»*. Донецьк: ДВНЗ «ДонНТУ». 2011. № 14 (188). С. 80–84.
43. Баркалов А. А., Бабаков Р. М. Определение области эффективного применения микропрограммного автомата с операционным автоматом переходов. *Кибернетика и системный анализ*. 2018. № 3. С. 27–37.
44. Баркалов А. А., Бабаков Р. М. Организация операционной части в управляющем автомате с операционным автоматом переходов. *Наукові праці Донецького національного технічного університету. Серія «Обчислювальна техніка та автоматизація»*. Донецьк: ДВНЗ «ДонНТУ». 2011. № 21 (183). С. 149–156.
45. Баркалов А. А., Бабаков Р. М. Реализация функции переходов микропрограммного автомата на базе операционного автомата. *Управляющие системы и машины*. 2015. № 5. С. 22–29.
46. Баркалов А. А., Бабаков Р. М. Структурная классификация методов синтеза микропрограммного автомата с операционным автоматом переходов. *Кибернетика и системный анализ*. 2019. № 2. С. 3–9.
47. Баркалов А. А., Бабаков Р. М. Уменьшение максимального количества существенных входных переменных в микропрограммном автомате с операционным автоматом переходов. *Управляющие системы и машины*. 2018. № 2. С. 42–50.
48. Баркалов А. А., Бабаков Р. М. Формализация операций переходов в

управляющем автомате с операционным автоматом переходов. *Наукові праці Донецького національного технічного університету. Серія: Інформатика, кібернетика та обчислювальна техніка*. Донецьк: ДВНЗ «ДонНТУ». 2013. № 1 (17). С. 19–23.

49. Баркалов А. А., Джалиашвили З. О., Струнилин В. Н. Оптимизация композиционного микропрограммного устройства управления. *Известия вузов СССР. Приборостроение*. 1989. № 3 . С. 36–39.
50. Баркалов А. А., Зеленева И. Я., Ефименко К. Н. Оптимизация схемы адресации КМУУ с элементарными логическими цепями. *Наукові праці ДонНТУ. Серія: «Обчислювальна техніка та автоматизація»*. Донецьк: ДВНЗ «ДонНТУ». 2013. № 1 (24). С. 214–221.
51. Баркалов А. А., Зеленева И. Я., Татолов Е. Р. Анализ эффективности методов кодирования состояний при синтезе автомата Мили на FPGA. *Наукові праці ДонНТУ. Серія: «Пробеми моделювання та автоматизації проектування»*. Донецьк: ДВНЗ «ДонНТУ». 2011. № 10 (197). С. 183–190.
52. Баркалов А. А., Зеленева И. Я., Цололо С. А., Биайрак Х. Уменьшение площади матричной схемы устройства управления с разделением кодов. *Радиоэлектроника, информатика, управление*. 2011. № 2. С. 101–107.
53. Баркалов А. А., Ковалев С. А., Бабаков Р. М. Определение вероятности кэш-попаданий в композиционных микропрограммных устройствах управления по граф-схеме алгоритма. *Автоматика и вычислительная техника*. 2001. № 4. С. 44–52.
54. Баркалов А. А., Ковалев С. А., Бабаков Р. М. Применение принципов кэширования в композиционных микропрограммных устройствах управления. *Управляющие системы и машины*. 2001. № 5. С. 26–33.
55. Баркалов А. А., Мальчева Р. В., Солдатов К. А. Матричная реализация автомата Мура с расширением кодов состояний перехода. *Научные труды Донецкого национального технического университета. Серия «Информатика, кибернетика и вычислительная техника» (ИКВТ-2010)*. Выпуск 11 (164). Донецк: ГВУЗ «ДонНТУ». 2010. С. 79–83.

56. Баркалов А. А., Мальчева Р.В., Баркалов А. А. Реализация схемы автомата Мили в базисе гибридных FPGA. *Наукові праці ДонНТУ. Серія: «Інформатика, кібернетика та обчислювальна техніка»*. Донецьк: ДВНЗ «ДонНТУ». 2014. № 1 (19). С. 22–26.
57. Баркалов А. А., Матвиенко А. В. Реализация микропрограммного устройства управления композицией автоматов с жесткой и программируемой логикой. *Микропроцессорные средства, разработка и применение*. К.: ИК АН УССР, 1985. С. 38–42.
58. Баркалов А. А., Палагин В. А. Синтез микропрограммных устройств управления. Киев: Институт кибернетики НАН Украины. 1997. 135 с.
59. Баркалов А. А., Саломатин В. А., Струнилин В. Н., Швец А. Г. Синтез композиционного микропрограммного устройства управления с фиксированной областью входов и модифицированной системой микрокоманд. *Управляющие системы и машины*. 1996. № 1–2. С.5–9.
60. Баркалов А. А., Струнилин В. Н. Композиционные микропрограммные устройства управления с максимальным кодированием микрокоманд. *Информатика, кибернетика и ВТ. Сборник трудов ДГТУ*. Выпуск 1. 1996. С. 123–128.
61. Баркалов А. А., Титаренко Л. А., Ефименко К. Н., Зеленева И. Я. Оптимизация КМУУ с разделением кодов. *Наукові праці ДонНТУ. Серія «Інформатика, кібернетика та обчислювальна техніка»*. Донецьк: ДВНЗ «ДонНТУ». 2012. № 15 (203). С. 11–16.
62. Баркалов А. А., Титаренко Л. А., Ефименко К. Н., Зеленева И. Я. Реализация КМУУ с общей памятью на гибридных FPGA. *Наукові праці ДонНТУ. Серія: «Проблеми моделювання та автоматизації проектування».* Донецьк: ДВНЗ «ДонНТУ». 2013. № 1 (12) – 2 (13). С. 33–42.
63. Баркалов А. А., Титаренко Л. А., Ефименко К. Н., Зеленёва И. Я. Разделение схемы адресации в КМУУ с общей памятью. *Известия ТТИ ЮФУ-ДонНТУ. Материалы Четырнадцатого Международного научно-практического семинара «Практика и перспективы развития*

партнерства в сфере высшей школы». В 3х кн. Таганрог: Изд-во ТТИ ЮФУ. 2013. № 13. Кн. 2. С. 11–16.

64. Баркалов А. А., Титаренко Л. А., Ефименко К. Н., Липински Я. М. Оптимизация схемы КМУУ с преобразователем адреса микрокоманд. *Наукові праці ДонНТУ. Серія: «Проблеми моделювання та автоматизації проектування»*. Донецьк: ДВНЗ «ДонНТУ». 2011. № 9 (179). С. 26–35.
65. Баркалов А. А., Титаренко Л. А., Ефименко К. Н., Липински Я. М. Оптимизация схемы КМУУ с общей памятью. *Управляющие системы и машины*. 2011. № 5. С. 47–52.
66. Баркалов А. А., Титаренко Л. А., Ефименко К. Н., Липински Я. М. Реализация КМУУ с элементарными цепями на гибридных FPGA. *Наукові праці ДонНТУ. Серія: «Інформатика, кібернетика та обчислювальна техніка»*. Донецьк: ДВНЗ «ДонНТУ». 2014. № 1 (19). С. 16–21.
67. Баркалов А. А., Титаренко Л. А., Мирошкин А. Н. Реализация композиционных микропрограммных устройств управления на FPGA-микросхемах. *Радиоэлектроника и информатика*. 2011. № 1. С. 52–55.
68. Баркалов А. А., Титаренко Л. А., Цололо С. А. Уменьшение аппаратурных затрат в схеме микропрограммного автомата Мура на CPLD. *Радіоелектронні та комп'ютерні системи*. 2008. № 7 (34). С.118–123.
69. Баркалов А. А., Черкашин В. А., Самир Нахлави. Принципы оптимизации логических схем автоматов на счетчиках. *Сборник научных трудов ДонДТУ. Серия «Вычислительная техника и автоматизация».* Выпуск 12. Донецк: ДонДТУ, 1999. С. 190–196.
70. Баркалов А. А., Швец А. Г. Синтез композиционного микропрограммного устройства управления с преобразователем номера операторной цепи. *Сборник трудов факультета вычислительной техники и информатики*. Выпуск 1. Донецк, ДонГТУ. 1996. С. 83–89.
71. Баркалов А. А., Швец А. Г. Синтез композиционного микропрограммного устройства управления с преобразователем кода. *Автоматика и вычислительная техника*. 1995. № 6. С. 16–24.

72. Баркалов А. А., Юсифов С. И. Минимизация числа ПЛМ в композиционном микропрограммном устройстве управления. *Средства микропроцессорной техники*. К.: ИК АН УССР. 1987. С. 24–28.
73. Берже Ж.-М., Фонкуа А., Мажино С., Руйар Ж. Новые свойства языка описания аппаратуры VHDL / пер. с англ. М.: Радио и связь, 1995. 256 с.
74. Бибило П. Н. Основы языка VHDL. Изд. 3-е, доп. М.: Издательство ЛКИ, 2007. 328 с.
75. Бибило П. Н. Синтез логических схем с использованием языка VHDL. М.: СОЛОН-Р, 2002. 384 с.
76. Биркгоф Г., Барти Т. Современная прикладная алгебра. М.: Мир, 1976. 400 с.
77. Богатырев Е. А., Ларин В. Ю., Лякин А. Е. Энциклопедия электронных компонентов. Большие интегральные схемы. Т. 1. М.: ООО «МАКРО ТИМ», 2006. 224 с.
78. Богомолов А. М., Салий А. М. Алгебраические основы теории дискретных систем. М.: Наука. Физматлит, 1997. 368 с.
79. Большая советская энциклопедия: в 52 т. / гл. ред. Б. А. Введенский. 2-е изд. М.: Государственное научное издательство «Большая советская энциклопедия». Т. 27. 1954. 661 с.
80. Брауэр В. Введение в теорию конечных автоматов / пер. с нем. М.: Радио и связь, 1987. 392 с.
81. Бродин В. Б., Калинин А. В. Системы на микроконтроллерах и БИС программируемой логики. М.: Издательство ЭКОМ, 2002. 400 с.
82. Гаврилов М. А., Девятков В. В., Пупырев Е. И. Логическое проектирование дискретных автоматов. М.: Наука, 1977. 351 с.
83. Гёлль П. Как превратить компьютер в универсальный программатор / пер. с франц. М.: ДМК, 2000. 168 с.
84. Гилл А. Введение в теорию конечных автоматов. М.: Наука, 1966. 272 с.
85. Глушков В. М. Абстрактная теория автоматов. *Успехи математических наук*. 1961. Т. 16, вып. 5 (101). С. 3–62.

86. Глушков В. М. Введение в кибернетику. Киев: Изд-во АН УССР, 1964. 324 с.
87. Глушков В. М. Синтез цифровых автоматов. М.: Физматгиз, 1962. 476 с.
88. Глушков В. М. Теория автоматов и вопросы проектирования структур цифровых устройств. *Кибернетика*. 1965. № 1. С. 1–9.
89. Глушков В. М. Теория автоматов и вопросы проектирования структур цифровых машин. *Кибернетика*. 1965. № 1. С. 3–11.
90. Глушков В. М. Теория автоматов и формальные преобразования микропрограмм. *Кибернетика*. 1965. № 5. С. 1–9.
91. Глушков В. М., Барабанов А. А., Калиниченко Л. А., Махновский С. Д., Рабинович З. Л. Вычислительные машины с развитыми системами интерпретации. Киев: Наукова думка, 1970. 259 с.
92. Глушков В. М., Цейтлин Г. Е., Ющенко Е. Л. Алгебра. Языки. Программирование. Изд. 3-е, перераб. и доп. Киев: Наук. думка, 1989. 376 с.
93. Горбатов В. А. Семантическая теория проектирования автоматов. М.: Энергия, 1979. 264 с.
94. Грушвицкий Р. И., Мурсаев А. Х., Угрюмов Е. П. Проектирование систем на микросхемах программируемой логики. СПб.: БВХ-Петербург, 2002. 608 с.
95. Девятков В. В. Методы реализации конечных автоматов на сдвиговых регистрах. М.: Энергия, 1974. 80 с.
96. Дроздов Е. А. Оптимизация структур цифровых автоматов. М.: Советское радио, 1975. 352 с.
97. Жинтелис Г. Б., Карчаускас Э. К., Мачикенас Э. К. Автоматизация проектирования микропрограммируемых структур. Л.: Машиностроение, 1985. 216 с.
98. Жмалкин А. П. Архитектура ЭВМ. СПб.: БХВ-Петербург, 2006. 320 с.
99. Закревский А. Д. Алгоритмы синтеза дискретных автоматов. М.: Наука, 1971. 511 с.
100. Захаров В. Н., Поспелов Д. А., Хазацкий В. Е. Системы управления. Задание. Проектирование. Реализация. Изд. 2-е, перераб. и доп. М.: Энергия, 1977. 424 с.

101. Зотов В. Ю. Проектирование встраиваемых микропроцессорных систем на основе ПЛИС фирмы Xilinx. М.: Горячая линия – Телеком, 2006. 520 с.
102. Зотов В. Ю. Проектирование цифровых устройств на основе ПЛИС фирмы Xilinx в САПР WebPACK ISE. М.: Горячая линия – Телеком, 2003. 624 с.
103. Зубчук В. И., Сигорский В. П., Шкурко А. Н. Справочник по цифровой схемотехнике. К.: Тэхника, 1990. 448 с.
104. Каган Б. М. Электронные вычислительные машины и системы: учеб. пособие для вузов. 3-е изд., перераб. и доп. М.: Энергоатомиздат, 1991. 592 с.
105. Калужнин Л. А. Об алгоритмизации математических задач. *Проблемы кибернетики*. М.: Физматгиз, 1959. № 2. С. 51–67.
106. Капитонова Ю. В., Кривой С. Л., Летичевский А. А., Луцкий Г. М. Лекции по дискретной математике. СПб.: БХВ-Петербург, 2004. 624 с.
107. Карпов Ю. Г. Теория автоматов. СПб.: Питер, 2003. 208 с.
108. Каталевский Д. Ю. Основы имитационного моделирования и системного анализа в управлении. М.: МГУ, 2011. 312 с.
109. Кирпичников В. М., Скляров В. А. Синтез микропрограммных автоматов по граф-схемам алгоритмов с малым числом условных вершин. *Управляющие системы и машины*. 1978. № 1. С. 77–83.
110. Ковалев С. А., Зеленева И. Я., Татолов Е. Р. Подход к унификации процесса синтеза МПА Мура для FPGA. *Материалы Двенадцатого международного научно-практического семинара «Практика и перспективы развития партнерства в сфере высшей школы» (12–14 апреля 2011 г., Донецк, Украина)*. Донецк: ДонНТУ. 2011. Том 2. С. 45–48.
111. Комолов Д. А., Мыляк Р. А., Зобенко А. А., Филлипов А. С. Системы автоматизированного проектирования фирмы Altera Max plus II и Quartus II. Краткое описание и самоучитель. М.: ИП РадиоСофт, 2002. 352 с.
112. Кон П. Универсальная алгебра. М.: Мир, 1968. 352 с.
113. Кравцов Л. Я., Черницкий Г. И. Проектирование микропрограммных устройств управления. Л.: Энергия, 1976. 152 с.
114. Кудрявцев В. Б., Алешин С. В., Подкозлин А. С. Введение в теорию

автоматов. М.: Наука, 1985. 320 с.

115. Кузнецов О. П. Представление регулярных событий в асинхронных автоматах. *Автоматика и телемеханика*. 1965, Т. XXVI. № 6. С. 1086–1093.
116. Кук Д., Бейз Г. Компьютерная математика / пер. с англ. М.: Наука, Гл. ред. физ.-мат. лит., 1990. 384 с.
117. Курош А. Г. Лекции по общей алгебре, изд. 2. М.: Наука, 1973. 440 с.
118. Лаговиер Б. А. Алгоритм синтеза автоматов со сложным регистром в качестве памяти. *Механизация и автоматизация управления*. 1976. № 2. С. 45–49.
119. Лазарев В. Г. Синтез управляющих автоматов. 3-е изд., перераб. и доп. М.: Энергоатомиздат, 1989. 328 с.
120. Лазарев В. Г., Пийль Е. И. Синтез асинхронных конечных автоматов. М.: Наука, 1965. 320 с.
121. Лазарев В. Г., Пийль Е. И. Синтез управляющих автоматов. М.: Энергия, 1978. 408 с.
122. Лазарев В. Г., Пийль Е. И., Турута Е. Н. Построение программируемых управляющих устройств. М.: Энергоатомиздат, 1984. 192 с.
123. Липский В. Комбинаторика для программистов: Пер. с польск. М.: Мир, 1988. 213 с.
124. Лысиков Б. Г. Арифметические и логические основы цифровых автоматов: учебник для вузов по спец. «Электрон. вычисл. машины». 2-е изд., перераб. и доп. Минск.: Выш. школа, 1980. 336 с.
125. Майоров С. А., Новиков Г. И. Структура электронных вычислительных машин. Л.: Машиностроение, 1979. 384 с.
126. Максфилд К. Проектирование на ПЛИС. Курс молодого бойца. М.: Издательский дом «Додэка-XXI», 2007. 408 с.
127. Мальцев А. И. Алгебраические системы. М.: Наука, 1970. 392 с.
128. Мальцев П. П., Гарбузов Н. И., Шарапов А. П., Кнышев Д. А. Программируемые логические ИМС на КМОП-структурах и их

применение. М.: Энергоатомиздат, 1998. 160 с.

129. Мелихов А. Н. Ориентированные графы и конечные автоматы. М.: Наука, 1974. 416 с.
130. Михайлов С. А. Проектирование матричных сверхбольших интегральных схем. Киев: Техника, 1991. 140 с.
131. Новиков Ф. А. Дискретная математика для программистов. СПб.: Питер, 2000. 304 с.
132. Новиков Ю. В. Основы цифровой схемотехники. Базовые методы и схемы. Методы проектирования. М.: Мир, 2001. 379 с.
133. Опанасенко В. Н. Реализация ускоренных алгоритмов целочисленного деления на ПЛИС. *Проблемы информатизации и управления*. 2008. № 1. С. 114–117.
134. Орлов С. А., Цилькер Б. Я. Организация ЭВМ и систем: Учебник для вузов. 2-е изд. СПб.: Питер, 2011. 688 с.
135. Первый толковый БЭС. Большой энциклопедический словарь / ред. С. М. Снарская. М.; СПб.: Рипол-Норинт, 2006. 2144 с.
136. Плоткин Б. И. Универсальная алгебра, алгебраическая логика и базы данных. М.: Наука. Гл. ред. физ.-мат. лит., 1991. 448 с.
137. Плоткин Б. И., Гринглаз Л. Я., Гварамия А. А. Элементы алгебраической теории автоматов: учеб. пособие для вузов. М.: Высш. школа, 1994. 191 с.
138. Половко А. М., Бутусов П. Н. Интерполяция. Методы и компьютерные технологии их реализации. СПб.: БХВ-Петербург, 2004. 320 с.
139. Поляков А. К. Языки VHDL и VERILOG в проектировании цифровой аппаратуры. М.: СОЛОН-пресс, 2003. 320 с.
140. Поспелов Д. А. Арифметические основы вычислительных машин дискретного действия. Учеб. пособие для втузов. М.: Высш. школа, 1970. 308 с.
141. Поспелов Д. А. Арифметические основы вычислительных машин дискретного действия. Учебное пособие для втузов. М.: Высш. школа, 1970. 308 с.

142. Поспелов Д. А. Логические методы анализа и синтеза схем. Изд. 3-е, перераб. и доп. М.: «Энергия», 1974. 368 с.
143. Прикладная теория цифровых автоматов / К. Г. Самофалов и др. Киев: Вища школа, Головное изд-во, 1987. 375 с.
144. Проектирование реконфигурируемых цифровых систем. А. В. Палагин, А. А. Баркалов, Л. А. Титаренко. Луганск: Издательство ВНУ, 2011. 432 с.
145. Пупырев Е. И. Перестраиваемые автоматы и микропроцессорные системы. М.: Наука, 1984. 192 с.
146. Пухальский Г. И., Новосельцева Т. Я. Цифровые устройства: учебное пособие для втузов. СПб.: Политехника, 1996. 885 с.
147. Савельев А. Я. Арифметические и логические основы цифровых автоматов: учебник. М.: Высш. школа, 1980. 255 с.
148. Савельев А. Я. Основы информатики: учеб. для вузов. М.: Изд-во МГТУ им. Н. Э. Баумана, 2001. 328 с.
149. Савельев А. Я. Прикладная теория цифровых автоматов: учеб. для вузов по спец. ЭВМ. М.: Высш. школа, 1987. 272 с.
150. Сергиенко А. М. VHDL для проектирования вычислительных устройств. Киев: ЧП «Корнейчук», ООО «ТИД «ДС», 2003. 208 с.
151. Скляров В. А. Синтез автоматов на матричных БИС. Минск: Наука и техника, 1984. 256 с.
152. Соловьев В. В. Реализация на ПЛИС граф-схем алгоритмов с повторяющимися фрагментами. Весцi Акадэмii Навук Беларусi. Сер. физ.-техн. наук. 1997. № 3. С. 75–81.
153. Соловьев В. В., Климович А. Логическое проектирование цифровых систем на основе программируемых логических интегральных схем. 2-е изд. М.: Горячая линия – Телеком. 2014. 376 с.
154. Соловьев Г. Н. Арифметические устройства ЭВМ. М.: Энергия, 1978. 176 с.
155. Соловьев Г. Н. Схемотехника ЭВМ: учебник для студентов вузов спец. ЭВМ. М.: Высш. школа, 1985. 391 с.
156. Спивакс Ш. Р. Реализация микропрограммных автоматов. *Автоматика и*

телемеханика. 1970. № 12. С. 133–139.

157. Стешенко Б. В. ПЛИС фирмы Altera: элементная база, система проектирования и языки описания аппаратуры. М.: Издательский дом «Додэка-XXI», 2007. 576 с.
158. Суворова Е. А., Шейнин Ю. Е. Проектирование цифровых систем на VHDL. СПб.: БХВ-Петербург, 2003. 576 с.
159. Судоплатов С. В., Овчинникова Е. В. Элементы дискретной математики: учебник. М.: ИНФРА-М, Новосибирск: Изд-во НГТУ, 2002. 280 с.
160. Таненбаум Э. Архитектура компьютера. 5-е изд. СПб.: Питер, 2007. 844 с.
161. Таненбаум Э., Остин Т. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 816 с.
162. Тарасов И. Е. Разработка цифровых устройств на основе ПЛИС Xilinx с применением языка VHDL. М.: Горячая линия – Телеком, 2005. 252 с.
163. Трахтенброт Б. А., Бардзинь Я. М. Конечные автоматы (поведение и синтез). М.: Наука, 1970. 400 с.
164. Угрюмов Е. П. Цифровая схемотехника. СПб.: БХВ – Санкт-Петербург, 2000. 528 с.
165. Український радянський енциклопедичний словник: в 3-х т. / ред. кол.: Бажан М. П., Білодід І. К., Гуржій І. О. та ін. Київ: Головна редакція Української радянської енциклопедії. Т. 2. 1967. 736 с.
166. Уэйкерли Дж. Ф. Проектирование цифровых устройств. В 2-х т. М.: Постмаркет, 2002.
167. Философский словарь / под ред. И. Т. Фролова. М.: Политиздат, 1991. 560 с.
168. Філософський енциклопедичний словник / ред. Н. В. Хамітов. Київ: Інститут філософії Національної академії наук України, Абрис, 2002. 744 с.
169. Хассон С. Микропрограммное управление. М.: Мир, 1973. 238 с.
170. Шалыто А. А. Логическое управление. Методы аппаратной и программной реализации. СПб.: Наука, 2000. 780 с.
171. Шиханович Ю. А. Введение в современную математику (начальные понятия). М.: Наука, 1965. 376 с.

172. Шнайер Б. Прикладная криптография: протоколы, алгоритмы и исходный код на С, 2-е юбилейное издание. М.: Диалектика-Вильямс, 2016. 1040 с.
173. Энциклопедия кибернетики. В 2-х томах / отв. ред. В. М. Глушков. Киев: Гл. ред. Украинской Советской энциклопедии, 1974.
174. Энциклопедия кибернетики: в 2 т. / под ред. В. М. Глушкова. Киев: Главная редакция Украинской советской энциклопедии, 1973. Т. 2. 576 с.
175. Энциклопедия кибернетики: в 2 т. / Под ред. В. М. Глушкова. Киев: Главная редакция Украинской советской энциклопедии, 1973. Т. 1. 584 с.
176. Якубайтис Э. А. Асинхронные логические автоматы. Рига: Зинатне, 1966. 379 с.
177. Adamski M., Barkalov A. Architectural and Sequential Synthesis of Digital Devices, Zielona Gora: University of Zielona Gora Press, 2006. 199 p.
178. Anand B. P., Saravanan C. G. Development of Research Engine Control Unit Using FPGA-based Embedded Control System. *Journal of KONES Powertrain and Transport*. 2012. Vol. 19. № 3. P. 9–18.
179. Axelford R. Structure of decision: The cognitive maps of political elites. New Jersey: Princeton University Press, 1976. 400 p.
180. Babakov R. M. Using of method of replacement of input variables in microprogram finite-state machine with datapath of transitions. *Технологический аудит и резервы производства*. 2017. № 4/2 (36). С. 18–23.
181. Babakov R., Barkalov A., Titarenko L. Research of Efficiency of Microprogram Final-State Machine with Datapath of Transitions. *Proceedings of 14th International Conference «The Experience of Designing and Application of CAD Systems in Microelectronics» (CADSM)* (Україна, Поляна, 21–25 лютого 2017 р). Львів, 2017. С. 203–206.
182. Bader D. A., Madduri K. A parallel state assignment algorithm for finite state machines. *Proceedings of 11th IEEE International Conference on High-Performance Computing (December 19–22, 2004, Bangalore, India)*. Springer-Verlag Berlin Heidelberg. 2005. P. 297–308.
183. Baranov S. Logic Synthesis for Control Automata. Boston: Kluwer Academic

Publishers. 1994. 393 p.

184. Baranov S., Levin I., Koren O., Karpovski M. Designing fault tolerant FSM by nano-PLA. *In Proceedings of the 15th IEEE International On-Line Testing Symposium (June 24–26, 2009, Sembra-Lisbon, Portugal)*. IEEE Computer Society: Test Technology Technical Council. 2009. P. 229–234.
185. Barkalov A. A., Barkalov A. A., Jr. Design of Mealy finite-state machines with the transformation of object codes. *International Journal of Applied Mathematics and Computer Science*. 2005. № 15 (1). P. 151–158.
186. Barkalov A. A., Bukowiec A. Synthesis of Mealy Finite State Machines for Interpretation of Verticalized Flow-Charts. *Theoretical and Applied Informatics*. 2005. № 9. P. 39–51.
187. Barkalov A. A., Titarenko L. A., Babakov R. M. Compositional Microprogram Control Unit with Operational Automaton of Transitions. *Proceedings of IEEE East-West Design & Test Symposium (EWDTS'2012)* (Kharkov, Ukraine, September 14–17). Kharkov, 2012. P. 151–154.
188. Barkalov A. A., Titarenko L. A., Babakov R. M. Compositional Microprogram Control Unit with Operational Automaton of Transitions. *12th Conference on Programmable Devices and Embedded Systems* (Velke Karlovice, Czech Republic, September 25–27), Velke Karlovice, 2013. P. 239–244.
189. Barkalov A. A., Titarenko L. A., Babakov R. M., Baiev A. V. Logic synthesis for FPGA-based finite state machines: monograph. Vinnitsya: DonNU Vasyl' Stus. 2016. 195 p.
190. Barkalov A. A., Tytarenko L. A., Yefimenko K. N., Zeleneva I. J. Optimization of the Addressing Scheme in Compositional Microprogram Control Unit with Code Sharing. *Наукові праці ДонНТУ. Серія «Інформатика, кібернетика та обчислювальна техніка»*. Донецьк: ДВНЗ «ДонНТУ». 2013. № 1(17). С. 5–10.
191. Barkalov A. A., Tytarenko L. A., Zeleneva I. J., Mielcarek Kamil. Design FSM Mealy with transformation of object codes based on embedded memory blocks. *Наукові праці ДонНТУ. Серія «Інформатика, кібернетика та обчислювальна техніка»*. Красноармійськ: ДВНЗ «ДонНТУ». 2015. № 1

(20). С. 5–10.

192. Barkalov A. A., Wisnievski R. Optimization of Compositional Microprogram Control Unit Implemented on System-on-Chip. *Theoretical and Applied Informatics*. 2005. P. 7–22.
193. Barkalov A. A., Zelenjova I. J., Hrushko S. S. Optimization of combined automaton circuit on base FPGA using the method of input variables changing. *Scientific Herald of Chernivtsi University: Computer systems and components*. 2015. Volume 6. Issue 2. P. 49–54.
194. Barkalov A. A., Zelenyova I. J., Biayrek Hathot, Lavrik A. S. Implementation of logic circuit of compositional microprogramming control unit with elementary operational linearchains on custom-made matrices. *Наукові праці ДонНТУ. Серія «Інформатика, кібернетика та обчислювальна техніка»*. Донецьк: ДВНЗ «ДонНТУ». 2011. № 13 (185). С. 49–53.
195. Barkalov A., Babakov R. Structural representation of synthesis methods of finite state machine with datapath of transitions. *Conference Proceedings of 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT'2018)* (м. Київ, 24–27 травня 2018 р). Київ, 2018. С. 242–246.
196. Barkalov A., Titarenko L., Bieganowski J. Synthesis of control unit with modified operational linear chains. *Pomiary Automatyka Kontrola*. 2007. № 5. P. 5–17.
197. Barkalov A., Titarenko L., Bieganowski J. Synthesis of compositional microprogram control unit with extended microinstruction format. *Mixed Design of Integrated Circuits and Systems – MIXDES'2009: 16th International Conference (25–27 June, 2009, Lodz, Poland)*. 2009.
198. Barkalov A., Titarenko L., Chmielewski S. Optimization of Moore FSM on System-on-Chip. *5th IEEE East-West Design & Test Symposium (EWDTS 2007) (September 7–10, Yerevan, Armenia)*. Kharkov: Kharkov National University of Radio Electronics. 2007. P. 105–109.
199. Barkalov A., Titarenko L., Chmielewski S. Reduction in the number of PAL macrocells in the circuit of a Moore FSM, International Journal of Applied

Mathematics and Computer Science 17 (4), 2007. P. 565–675. DOI: 10.2478/v10006-007-0046-8.

200. Barkalov A., Titarenko L., Kolopienczyk M., Mielcarek K., Bazydlo G. Logic Synthesis for FPGA-Based Finite State Machines. Springer Cham. 2015. 280 p. DOI: 10.1007/978-3-319-24202-6.
201. Barkalov A., Titarenko L., Smolinski L. Optimization of control unit based on construction of CPLD. *Pomiary, Automatyka, Kontrola*. 2012. № 58. P. 93–96.
202. Barkalov A., Titarenko L., Wisniewski R. Synthesis of compositional microprogram control units with sharing codes and address decoder. *Proceedings of the International Conference on Mixed Design of Integrated Circuits and Systems (MIXDES 2006) (22–24 June, Gdynia, Poland, 2006)*. Lodz: Politechnika Lodzka. 2006. P. 397–400.
203. Barkalov A., Titerenko L. Logic synthesis for compositional microprogram control units. Berlin: Springer. 2008. 272 p.
204. Barkalov A., Titerenko L. Logic synthesis for FSM-based control units. Berlin: Springer. 2009. 233 p.
205. Barkalov A., Wegrzyn M. Design of control units with programmable logic. Zielona Gora: University of Zielona Gyra Press. 2006. 150 p.
206. Barkalov A., Wegrzyn M., Wisniewski, R. Optimization of LUT-elements amount in cotrol unit of system-on-chip. *Discrete-Event System Design, DESDes '06: A Proceedings Volume from the 3rd IFAC Workshop*. Rydzyna, Poland, 2006. P. 143–146.
207. Barkalov A., Titarenko L., Barkalov A., Jr. Structural decomposition as a tool for the optimization of an FPGA-based implementation of a Mealy FSM. *Cybernetics and Systems Analysis*. 2012. №48 (2). P. 313–322.
208. Barkalov A., Titarenko L. and Smolinski L. (2011) Optimization of Microprogram Control Unit with Code Sharing. *Proceedings of IEEE East-West Design and Test Symposium (9–12 September, 2011, Sevastopol, Ukraine)*. Kharkov, Ukraine: Kharkov National University of Radio Electronics. 2011. P. 55–59.

209. Bieganowski J. Synthesis of microprogram control units oriented toward decreasing the number of macrocells of addressing circuit. *Lecture Notes in Control and Computer Science*. 2011. Vol. 17. P. 103–109.

210. Bomar B. W. Implementation of microprogrammed control in FPGAs. *IEEE Transactions on Industrial Electronics*. 2002. Vol. 49. № 2. P. 415–422.

211. Borowik G., Fialkowski B., Luba T. Cost-efficient synthesis for sequetnial circuits implemented using embedded memory blocks of FPGA's. *Proceedings of the IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (April 11–13, Krakow, Poland)*. Krakow: IEEE Computer Society. 2007. P. 99–104.

212. Brown S., Vranesic Z. Fundamentals of Digital Logic with VHDL Design, 3rd Ed. New York: McGraw Hill. 2000. 939 p.

213. Champarnaud J.-M., Khorsi A., Paranthoen T. Split and join for minimization: Brzozowski's algorithm. *Proceedings of Prague Stringology Conference (September 23–24, 2002, Prague)*. Prague: Czech Technical University. 2002. P. 96–104.

214. Chang D., Mazek-Sadowska M. Dinamically Reconfigurable FPGA. *IEEE Transaction on Computers*. 1999. № 6. P. 565–578.

215. Chattopadhyay S. Area conscious state assignment with flip-flop and output polarity selection for finite state machines synthesis – a genetic algorithm. *The Computer Journal*. 2005. № 48 (4). P. 443–450.

216. Chyzy M., Kosinsky W. Evolutionary algorithm for state assignment of finite state machines. *Proceedings of Symposium on Methods of Artificial Intelligence AI-METH 2003 (November 5–7, 2003, Gliwice, Poland)*. Gliwice: Silesian University of Technology. 2003. P. 51–52.

217. Cong J., Yan K. Synthesis for FPGAs with embedded memory blocks. *Proceedings of the 2000 ACM/SIGDA 8th International Symposium on FPGAs (Monterey, CA, USA – February 10–11, 2000)*. ACM: New York, NY, USA. 2000. P. 75–82.

218. Czerwinski R., Kania D. Finite State Machine Logic Synthesis for Complex Programmable Logic Devices. Berlin: Springer. 2013. 172 p.

219. Czerwinski R., Kania D. State assignment for PAL-based CPLDs. *Proceedings of 8th Euromicro Symposium on Digital System Design (August 30 to September 3, 2005, Porto, Portugal)*. IEEE Computer Society. 2005. P. 127–134. DOI:10.1109/DSD.2005.71.

220. Czerwinski R., Kania D. State assignment method for high speed FSMs. *Proceedings of IFAC Workshop on Programmable Devices and Systems (November 18–19, 2004, Cracow, Poland)*. Elseveir Science. 2004. Volume 37. Issue 20. P. 216–221.

221. De Micheli G. Synthesis and Optimization of Digital Circuits. McGraw-Hill: NY, 1994. 574 p.

222. Devadas S., Ma H.-K., Newton R., Sangiovanni-Vincentelli A. State Assignment of Finite State Machines Targeting Multilevel Logic Implementations. *IEEE Transactions on Computer-Aided Design*. № 7 (12). 1988. P. 1290–1300.

223. Doligalski M., Adamski M. Hierarchical configurable Petri net modeling in VHDL. *International Journal of Electronics and Telecommunications*. 2012. № 58 (4). P. 397–402

224. Drusinsky D., Harel D. Using statecharts for hardware description and synthesis. *IEEE Transactions on Computer-Aided Design*. 1989. Vol. 8. № 7. P. 798–807.

225. El-Maleh A., Sait S. M., Khan F. N. Finite state machine state assignment for area and power minimization. *Proceedings of IEEE International Symposium on Circuits and Systems (May 21–24, 2006, Island of Kos, Greece)*. New Jersey: IEEE Circuits and Systems Society. 2007. P. 5303–5306.

226. Escherman B. State assignment for hardwired VLSI control units. *ACM Computing Surveys*. 1993. № 25 (4). P. 415–436.

227. Gajski D. D., Abdi S., Gerstlauer A., Schirner G. Embedded System Design: Modeling, Synthesis and Verification. Berlin / Heidelberg: Springer. 2009. 352 p.

228. Gajski D., Kleinsmith J. Principles of Digital Design. New Jersey: Prentice Hall. 1996. 173 p.

229. Garcia-Vargas I., Senhadji-Navarro R. Finite state machines with input multiplexing: A performance study. *IEEE Transactions a Computer-Aided Design of Integrated Circuits and Systems*. 2015. № 34 (5). P. 867–871.

230. Garcia-Vargas I., Senhadji-Navarro R., Jimenez-Moreno G., Civit-Balcells A., Guerra-Gutierrez P. ROM-based finite state machine implementation in low cost FPGAs. *IEEE International Symposium on Industrial Electronics (ISIE) (Vigo, Spain, 4–7 June 2007)*. P. 2342–2347. DOI: 10.1109/ISIE.2007.4374972.

231. Glykas M. Fuzzy Cognitive Maps: Advances in Theory, Methodologies, Tools and Applications. *Studies in Fuzziness and Soft Computing*. Vol. 247. Springer: Berlin, 2010.

232. Golson S. One-hot state machine design for FPGAs. *Proceedings of the 3rd PLD Design Conference (March, 30, 1993, Santa Clara, Canada)*. 1993. P. 1–6.

233. Goren S., Ferguson F. CHESMIN: a heuristic for state reduction in incompletely specified finite state machines. *Proceedings of the Conference on Design, Automation and Test in Europe (March 4–8, 2002, Paris, France)*. IEEE Computer Society Washington, DC, USA. 2002. P. 248–254.

234. Grout I. Digital Systems Design with FPGAs and CPLDs. 1st Edition. Oxford: Elsevier. 2008. 784 p.

235. Gupta B., Narayanan H. and Desai M. A state assignment scheme targeting performance and area. *Proceedings of 12th International Conference on VLSI Design (January 10–13, 1999, Goa, India)*. IEEE Computer Society. 1999. P. 378–383.

236. Hartmanis J., Stearns R. E. Algebraic Structure Theory of Sequential Machines. New Jersey: Prentice-Hall, 1966. 211 p.

237. Higuchi H., Matsunaga Y. A fast state reduction algorithm for incompletely specified finite state machines. *33rd Annual Conference of Design Automation (June 03–07, 1996, Las Vegas, NV, USA)*. New York: ACM. 1996. P. 463–466.

238. Hu H., Xue H.-X., Bian J.-N. HSM2: a new heuristic state minimization algorithm for finite state machine. *Journal of Computer Science and Technology*. 2004. № 19 (5). P. 729–733.

239. Jarvinen K., Tommiska M., Skytta J. Hardware implementation of the MD5 hash algorithm. *HICSS'05 Proceedings of the 38th Annual Hawaii International Conference on System Sciences (January 3–6, 2005, Big Island, Hawaii, USA)*. IEEE Computer Society Washington, DC, USA. 2005. Track 9. P. 298.1. DOI: 10.1109/HICSS.2005.291.

240. Jenkins J. H. Designing with FPGAs and CPLDs. New Jersey, Englewood Cliffs: Prentice-Hall. 1994. 273 p.

241. Kam T., Villa T., Brayton R., Sangiovanni-Vincentelli A. A Synthesis of Finite State Machines: Functional Optimization. Boston, MA, USA: Springer. 1997. 282 p. DOI: 10.1007/978-1-4757-2622-0.

242. Kania D. Synteza logiczna przeznaczona dla matrycowych struktur programowalnych typu PAL. Gliwice: Zeszyty naukowe Politechniki Slaskiej. 2004. 240 p.

243. Kania D. The Logic Synthesis for the PAL-based Complex Programmable Logic Devices, Lecture Notes of the Silesian University of Technology, Gliwice (in Polish). 2004.

244. Kinney Larry L. Decomposition of asynchronous sequential switching circuits. *IEEE Transactions on Computers*. 1970. V. 19. № 6. P. 515-519.

245. Klimowicz A., Salauyou V. The Synthesis of Combined Mealy and Moore Machines Structural Model Using Values of Output Variables as Codes of States. *15th Euromicro Conference on Digital System Design (September 5–8, 2012, Cesme, Izmir Tourkey)*. P. 789–794. DOI: 10.1109/DSD.2012.130.

246. Kolopienczyk M. Application of address converter for decreasing memory size of CMCU with code sharing. *Lecture Notes in Control and Computer Science*. Vol. 12. Zielona Gora: University of Zielona Gora Press. 2008.

247. Kolopienczyk M., Titarenko L., Barkalov A. Design of EMB-based Moore FSMs. *Journal of Circuits, Systems, and Computers*. 2017. № 26 (7). P. 1–23.

248. Kopetz H. Real-time systems: design principles for distributed embedded applications. 2nd Edition. Springer Science+Business Media. 2011. 378 p.

249. Krzywicki K., Andrzejewski G. Hardware implementation of the CloudBus

protocol using FPGA. *In: Proceedings of the Prague Embedded Systems Workshop (June 12–13, 2014, Roztoky u Prahy, Czech Republic)*. P. 11–14. URL: http://pesw.fit.cvut.cz/2014/PESW_2014.pdf

250. Kubatova H., Bevcar M. FEL-Code: FSM internal state encoding method. *Proceedings of 5th International Workshop on Boolean Problems (September 12–20, 2002, Freiberg)*. Freiberg: Freiberg University of Mining and Technology. 2002. P. 109–114.
251. Kubica M., Kania D. Area-oriented technology mapping for LUT-based logic blocks. *International Journal of Applied Mathematics and Computer Science*. 2017. № 27 (1). P. 207–222. DOI: 10.1515/amcs-2017-0015.
252. Lee J. M. VERILOG QuickStart: A Practical Guide to Simulation and Synthesis in VERILOG, 3rd Ed. Norwell: Kluwer Academic Publishers. 1999. 356 p.
253. Luba T. Synthesis of Logic Devices. Warsaw: Warsaw University of Technology Press. 2005.
254. Luba T., Rawski M., Jachna Z. Functional decomposition as a universal method of logic synthesis for digital circuits. *Proceedings of 9th International Conference «Mixed Design of Integrated Circuits and Systems» (June 20–22, 2002, Wroclaw, Poland)*. Lodz: KMiTI. 2002. P. 285–290.
255. Mano M. Digital design (4th Edition). New Jersey: Prentice Hall. 2006. 624 p.
256. Martinez M., Avedillo M. J., Quintana J. M., Huertas J. L. A dynamic model for the state assignment problem. *Proceedings of Design, Automation and Test in Europe (February 23–26, 1998, Paris, France)*. ACM New York, NY, USA. 1998. P. 835–839.
257. Milik A., Hrynkiewicz E. Synthesis and implementation of reconfigurable PLC on FPGA platform. *International Journal of Electronics and Telecommunications*. 2012. № 58 (1). P. 85–94
258. Minns P., Elliot I. FSM-Based Digital Design Using Verilog HDL. New Jersey: J. Wiley & Sons. 2008. 408 p.
259. Monmasson E., Idkhajine L., Cirstea M. N. FPGAs in Industrial Control Applications. *IEEE Transactions on Industrial Informatics*. 2011. Vol. 7. Issue 2.

P. 224–243. DOI: 10.1109/TII.2011.2123908.

260. Navabi Z. Embedded Core Design with FPGAs. New York: McGraw-Hill. 2007. 433 p.
261. Nedjah N., Mourelle L. M. Evolutionary synthesis of synchronous finite state machines. Evolvable Machines. *In: Evolutionary synthesis of synchronous finite state machines. Evolvable Machines, 1st Edn*. Berlin: Springer. 2004. P. 103–128.
262. Nelson V., Nagle H., Irwin J., Carroll B. Digital logic circuit analysis and design. New Jersey: Prentice Hall. 1995. 842 p.
263. Papachristou C. Hardware microcontrol schems using PLAs. *Proceeding of 14th Microprogramming Workshop*. 1981. P. 3–15.
264. Pena J. M., Oliveira A. L. A new algorithm for exact reduction of incompletely specified finite state machines. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 1999. Vol. 18 (11). P. 1619–1632.
265. Popovskij V., Barkalov A., Titarenko L. Control and adaptation in telecommunication systems: Mathematical Foundations. *Lectures Notes in Electrical Engineering*. Berlin: Springer. 2011. № 94. 173 p.
266. Rawski M., Selvaraj H. Luba T. An application of functional decomposition in ROM-based FSM implementation in FPGA devices. *Journal of System Architecture*. 2005. № 51 (6–7). P. 423–434.
267. Rho J.-K., Hatchel G., Somenzi F., Jacoby R. Exact and heuristic algorithms for the minimization of incompletely specified state machines. *IEEE Transactions on Computer-Aided Design*. 1994. Vol. 13. P. 167–177.
268. Salcic Z. VHDL and FPLDs in Digital Systems Design, Prototyping and Customization. New York: Springer. 1998. 548 p.
269. Sasao T. Memory-Based Logic Synthesis. New York: Springer. 2011. 189 p.
270. Sasao T. Switching Theory for Logic Synthesis. Boston: Kluwer Academic Publishers. 1999. 362 p.
271. Scholl C. Functional Decomposition with Application to FPGA Synthesis. Boston: Kluwer. 2011.
272. Sklyarov V. Hierarchical finite-state machines and their use for digital control.

IEEE Transactions on VLSI Systems. 1999. Vol. 7. № 2. P. 222–228.

273. Sklyarov V. Reconfigurable models of finite state machines and their implementation in FPGAs. *Journal of Systems Architecture*. 2002. V. 47. Issue 14–15. P. 1043–1064.
274. Sklyarov V. Synthesis and Implementation of RAM-Based Finite State Machines in FPGAs. *In: Proceedings of 10th International Conference on Field Programmable Logic (August 27–30, 2000, Villach, Austria)*. Villach: Springer. 2000. P. 718–728.
275. Sklyarov V., Skliarova I., Barkalov A., Titarenko L. Synthesis and Optimization of FPGA-Based Systems. Berlin: Springer. 2014. 432 p.
276. Sklyarova I., Sklyarov V., Sudnitson A. Design of FPGA-based Circuits using Hierarchical Finite State Machines. Tallin: TUT Press. 2012. 240 p.
277. Sutter G., Todorovich E., Lypez-Buedo S., Boemo E. Low-power FSMs in FPGA: Encoding alternatives. *Proceedings of the 12th International Workshop on Power and Timing Modelling Optimization and Simulation (September 11–13, 2002, Sevilla, Spain)*. Berlin: Springer. 2002. P. 363–370.
278. Thomas D., Moorby P. The Verilog Hardware Description Language, 5th Edn. Norwell: Kluwer Academic Publishers. 2002. 382 p.
279. Titarenko L., Bieganowski J. Optimization of compositional microprogram control unit by modification of microinstruction format. *Electronics and Telecommunication Quarterly*. 2009. № 55 (2). P. 201–214.
280. Tiwari A., Tomko K. Saving power by mapping finite-state machines into embedded memory blocks in FPGAs. *Proceedings of the Conference on Design, Automation and Test in Europe (February 16–20, 2004, Paris, France)*. Washington, DC, USA: IEEE Computer Society. 2004. P. 916–921.
281. V. Salauyou, T. Grzes. FSM State Assignment Methods for Low-Power Design. *Computer Information Systems and Industrial Management Applications, International Conference (June 28–30, 2007, Elk, Poland)*. Minneapolis, MN, 2007. P. 345–350. DOI: 10.1109/CISIM.2007.32.
282. Wilkes M. The best way to design an automatic calculation machine. *Manchester*

University Computer Inaugural Conf. Proc., 1951. P. 16–28.

283. Wilkes M., Stringer J. Microprogramming and the Design of the Control Circuits in the Electronic Digital Computer. *Proc. of the Cambridge Philosophical Society*, 1953. V. 42. Part 2. P. 230–238.
284. Wisniewska M., Wisniewski R., Adamski M. Usage of hypergraph theory in decomposition of concurrent automata. *Pomiary, Automatyka, Kontrola (Poland)*. 2007. № 7. P. 66–68.
285. Wisniewski R. Synthesis of Compositional Microprogram Control Units for Programmable Devices. Zielona Gora: University of Zielona Gora Press. 2009. 153 p.
286. Wisniewski R., Barkalov A., Titarenko L. Optimization of address circuit of compositional microprogram unit. *Proceedings of the IEEE East-West Design & Test Workshop (EWDTW '06) (Sochi, Russia, September 15–19, 2006)*. Kharkov: Kharkov National University of Radioelectronics. P. 167–170.
287. Xia Y., Almaini A. Genetic algorithm based state assignment for power and area optimization. *IEE Proceedings – Computers and Digital Techniques*. № 149 (4). 2002. P. 128–133. DOI: 10.1049/ip-cdt:20020431.
288. Zielonka W. Notes on finite asynchronous automata. RAIRO, Inf. Theor. Appl. 1987. Y. 21. P. 99–135.
289. Zwolinski M. Digital System Design with VHDL. Boston: Addison-Wesley Longman Publishing Co., Inc. 2000. 416 p.

ДОДАТОК А

Лістинги VHDL-моделей

Лістинг А.1 – Синтезована частина VHDL-опису МПА Мура на основі систем булевих рівнянь

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity FSM is                              -- Мікропрограмний автомат
    port (x1: in std_logic;                        -- Вхідний сигнал
          C: in std_logic;                         -- Clock
          Reset: in std_logic;                     -- Reset
          Y: out std_logic_vector (1 to 3));  -- Мікрооперації
end FSM;

architecture FSM_A of FSM is
signal T, D: std_logic_vector (1 to 2);--Поточний та наступний стани
begin
        process(C)                                 -- Регістр пам'яті
        begin
             if rising_edge(C) then
               if Reset = '1' then
                    T <= "00";
                else
                    T <= D;
                end if;
             end if;
       end process;

       -- Система рівнянь функції переходів
       D(1) <= (not T(1) and T(2)) or (T(1) and not T(2));
       D(2) <= (not T(1) and not T(2)) or (T(1) and not T(2))
                or (not T(1) and T(2) and x1);

       -- Система рівнянь функції виходів
       Y(1) <= not T(1) and T(2);
       Y(2) <= T(2);
       Y(3) <= T(1);
end FSM_A;
```

Лістинг А.2 – Моделююча частина VHDL-опису МПА Мура

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Model is
end Model;

architecture Model_A of Model is

signal C: std_logic;                          -- Синхронізація
signal Reset: std_logic;                      -- Сигнал ініціалізації
signal x1: std_logic;                         -- Логічні умови
signal Y: std_logic_vector (1 to 3);          -- Мікрооперації

component FSM is              -- Мікропрограмний автомат
    port (x1: in std_logic;                     -- Вхідні сигнали
          C: in std_logic;                      -- Clock
          Reset: in std_logic;                  -- Reset
          Y: out std_logic_vector (1 to 3));  -- Мікрооперації
end component FSM;

begin
        process               -- Генерація сінроімпульсу
        begin
           C <= '0'; wait for 40 ns;
           C <= '1'; wait for 20 ns;
        end process;

       Reset <= '0' after 0 ns, '1' after 10 ns, '0' after 80 ns;

        process               -- Генерація x1
        begin
          x1 <= '1'; wait for 25 ns;
             x1 <= '0'; wait for 44 ns;
        end process;

        L1: component FSM
            port map (x1, C, Reset, Y);
end Model_A;
```

Лістинг А.3 – Синтезована VHDL-модель МПА Мілі із використанням одного процесу

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity FSM is           -- Мікропрограмний автомат
    port (x1: in std_logic;                    -- Вхідний сигнал
          C: in std_logic;                     -- Clock
          Reset: in std_logic;                 -- Reset
          Y: out std_logic_vector (1 to 3));  -- Мікрооперації
end FSM;

architecture FSM_A of FSM is
type state_type is (a0, a1, a2);
signal state: state_type;
begin

        process(C)      -- Єдиний процес для всього автомата
        begin
             if rising_edge(C) then
               if Reset = '1' then
                    state <= a0;
                else
                    case state is
                        when a0 => state <= a1;
                                   Y <= "110";
                        when a1 => if x1='0' then
                                          state <= a2;
                                          Y <= "001";
                                   else
                                          state <= a0;
                                          Y <= "011";
                                   end if;
                        when a2 => state <= a0;
                                   Y <= "001";
                    end case;
                end if;
            end if;
        end process;
end FSM_A;
```

Лістинг А.4 – Синтезована VHDL-модель МПА Мура із використанням одного процесу

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity FSM is           -- Мікропрограмний автомат
    port (x1: in std_logic;                    -- Вхідний сигнал
          C: in std_logic;                     -- Clock
          Reset: in std_logic;                 -- Reset
          Y: out std_logic_vector (1 to 3));  -- Мікрооперації
end FSM;

architecture FSM_A of FSM is
type state_type is (b0, b1, b2, b3);
signal state: state_type;
begin
        process(C)      -- Єдиний процес для всього автомата
        begin
             if rising_edge(C) then
               if Reset = '1' then
                    state <= b0;
                else
                    case state is  -- Аналіз стану автомата
                        when b0 => Y <= "000";
                                   state <= b1;
                        when b1 => Y <= "110";
                                   if x1='0' then
                                         state <= b2;
                                   else
                                         state <= b3;
                                   end if;
                        when b2 => Y <= "001";
                                   state <= b3;
                        when b3 => Y <= "011";
                                   state <= b0;
                    end case;
                end if;
            end if;
        end process;
end FSM_A;
```

Лістинг А.5 – Синтезована VHDL-модель МПА Мілі із використанням двох процесів

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity FSM is           -- Мікропрограмний автомат
    port (x1: in std_logic;                    -- Вхідний сигнал
          C: in std_logic;                     -- Clock
          Reset: in std_logic;                 -- Reset
          Y: out std_logic_vector (1 to 3));  -- Мікрооперації
end FSM;

architecture FSM_A of FSM is
type state_type is (a0, a1, a2);
signal state: state_type;
begin

        process(C)      -- Єдиний процес для всього автомата
        begin
             if rising_edge(C) then
               if Reset = '1' then
                    state <= a0;
                else
                    case state is
                        when a0 => state <= a1;
                                   Y <= "110";
                        when a1 => if x1='0' then
                                          state <= a2;
                                          Y <= "001";
                                   else
                                          state <= a0;
                                          Y <= "011";
                                   end if;
                        when a2 => state <= a0;
                                   Y <= "001";
                    end case;
                end if;
            end if;
        end process;
end FSM_A;
```

Лістинг А.6 – Синтезована VHDL-модель МПА Мілі із використанням трьох процесів

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity FSM is           -- Мікропрограмний автомат
    port (x1: in std_logic;                    -- Вхідний сигнал
          C: in std_logic;                     -- Clock
          Reset: in std_logic;                 -- Reset
          Y: out std_logic_vector (1 to 3));  -- Мікрооперації
end FSM;

architecture FSM_A of FSM is
type state_type is (a0, a1, a2);
signal state, next_state: state_type; -- Поточний та наступний стани
begin

        process(C)           -- Процес для регістра пам'яті
        begin
             if rising_edge(C) then
               if Reset = '1' then
                    state <= a0;
                else
                    state <= next_state;
                end if;
            end if;
        end process;

        process (state, x1)  -- Процес для функції переходів
        begin
                   case state is
                        when a0 => next_state <= a1;
                        when a1 => if x1='0' then
                                          next_state <= a2;
                                   else
                                          next_state <= a0;
                                   end if;
                        when a2 => next_state <= a0;
                    end case;
        end process;

        process (state, x1)  -- Процес для функції виходів
        begin
                    case state is
```

```
                        when a0 => Y <= "110";
                        when a1 => if x1='0' then
                                        Y <= "001";
                                   else
                                        Y <= "011";
                                   end if;
                        when a2 => Y <= "001";
                    end case;
        end process;
end FSM_A;
```

Лістинг А.7 – Приклад VHDL-моделі для дослідження апаратурних витрат в кон’юнктивній частині системи булевих рівнянь

```
entity KLS is
  port(x : in  std_logic_vector (1 to 4);
       q : out std_logic_vector (1 to 5));
end KLS;

architecture A_KLS of KLS is
  signal nx : std_logic_vector (1 to 4);
  begin
      nx <= not x;
      q(1) <= nx(1) and x(2) and x(3) and x(4);
      q(2) <= x(1) and nx(2) and nx(3) and x(4);
      q(3) <= nx(1) and nx(2) and nx(3) and nx(4);
      q(4) <= x(1) and x(2) and nx(3) and x(4);
      q(5) <= x(1) and x(2) and x(3) and x(4);
end A_KLS;
```

Лістинг А.8 – Приклад VHDL-моделі, що реалізує систему булевих рівнянь

```
entity KLS is
  port(x : in  std_logic_vector (1 to 4);
       y : out std_logic_vector (1 to 2));
end KLS;

architecture A_KLS of KLS is
    signal nx : std_logic_vector (1 to 4);
    signal q :  std_logic_vector (1 to 5);
begin
     nx <= not x;
```

```
     q(1) <= nx(1) and x(2) and nx(3) and x(4);
     q(2) <= x(1) and nx(2) and nx(3) and x(4);
     q(3) <= nx(1) and nx(2) and nx(3) and nx(4);
     q(4) <= x(1) and x(2) and nx(3) and x(4);
     q(5) <= x(1) and x(2) and x(3) and x(4);

     y(1) <= q(1) or q(2) or q(3);
     y(2) <= q(4) or q(5) or q(2);
end A_KLS;
```

Лістинг А.9 – Приклад VHDL-моделі для дослідження апаратурних витрат в диз'юнктивній частині системи булевих рівнянь

```
entity KLS is
  port(q : in  std_logic_vector (1 to 20);
       y : out std_logic_vector (1 to 5));
end KLS;
architecture A_KLS of KLS is
  begin
     y(1) <= q(1) or q(2) or q(3) or q(4) or q(5) or q(6) or q(7);
     y(2) <= q(8) or q(9) or q(10) or q(11) or q(12)
                  or q(13) or q(14);
     y(3) <= q(1) or q(3) or q(5) or q(7) or q(15)
                  or q(17) or q(19);
     y(4) <= q(2) or q(4) or q(6) or q(8) or q(16)
                  or q(18) or q(20);
     y(5) <= q(5) or q(9) or q(11) or q(15) or q(16)
                  or q(19) or q(20);
end A_KLS;
```

Лістинг А.10 – VHDL-модель для дослідження апаратурних витрат на реалізацію мультиплексора

```
package my is
   constant r: integer := 7;  -- Розрядність виходу
   constant d: integer := 10; -- Число вхідних напрямків
   type Inputbus is array (0 to d-1) of bit_vector (1 to r);
end package;
use my.all;
```

```
Entity MX is
      port (X: in Inputbus;
            Y: out bit_vector (1 to r);
            sel: in natural range 0 to d-1);
end entity MX;

Architecture MX_A of MX is
begin
      Y <= X (sel);
end architecture MX_A;
```

Лістинг А.11 – VHDL-модель, реалізує операції додавання та віднімання

```
Entity OP is
          generic (r: integer :=20;
                   c: integer := 1);
          port (x: in integer range 0 to 2**r-1;
                y: out integer range 0 to 2**r-1);
end entity OP;
Architecture OP_A of OP is
begin
                y <= x + c;
                -- y <= x - c;
end architecture OP_A;
```

Лістинг А.12 – VHDL-модель, реалізує операцію множення на константу

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

Entity OP is
          generic (r: integer :=20);
      port (x1: in UNSIGNED (r-1 downto 0);
            x2: in UNSIGNED (r-1 downto 0);
            y: out UNSIGNED (r*2-1 downto 0));
end entity OP;

Architecture OP_A of OP is
begin
            y <= x1 * x2;
end architecture OP_A;
```

Лістинг А.13 – VHDL-модель, реалізує операції логічного зсуву

```
Entity OP is
          generic (r: integer := 4);
      port (x: in  bit_vector (r-1 downto 0);
            y: out bit_vector (r-1 downto 0));
end entity OP;
Architecture OP_A of OP is
begin
     y <= '0' & x(r-1 downto 1);    -- зсув на 1 розряд вліво
     -- y <= x(r-2 downto 0) & '0'; -- зсув на 1 розряд вправо
end architecture OP_A;
```

Лістинг А.14 – VHDL-модель для дослідження апаратурних витрат на реалізацію логічних операцій

```
Entity OP is
      generic (r: integer := 20);
      port (x1: in  bit_vector (r-1 downto 0);
            x2: in  bit_vector (r-1 downto 0);
            y:  out bit_vector (r-1 downto 0));
end entity OP;
Architecture OP_A of OP is
begin
           y <= x1 and x2; -- and, or або xor
end architecture OP_A;
```

Лістинг А.15 – VHDL-модель для дослідження апаратурних витрат на реалізацію порозрядних логічних операцій з константою

```
Entity OP is
          generic (r: integer := 20);
      port (x: in  bit_vector (r-1 downto 0);
            y: out bit_vector (r-1 downto 0));
end entity OP;
Architecture OP_A of OP is
begin
          y <= x and "11110000111100001111"; -- and, or або xor
end architecture OP_A;
```

Лістинг А.16 – VHDL-модель синхронного регістру з функцією скидання

```
Entity RG is
        generic (r: integer :=40);
      port (D: in bit_vector (1 to r);
            C: in bit;
            rst: in bit;
          Y: out bit_vector (1 to r));
end entity RG;

Architecture RG_A of RG is
begin
      Process (C)
      variable Q: bit_vector (1 to r);
      begin
        if C='1' then
           if rst='1' then
                Q := (others => '0');
           end if;
           if rst='0' then
                  Q := D;
              end if;
            end if;
        Y <= Q;
      end process;
end architecture RG_A;
```

Лістинг А.17 – VHDL-модель МПА з ОАП на лічильнику (ГСА Γ_{5-1}, 100 станів)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity FSM is
    generic (R: integer := 7);              -- Розрядність коду стану
    port (C: in std_logic;
          Reset: in std_logic;
          D: out unsigned (R-1 downto 0));
end entity FSM;

architecture FSM_A of FSM is
signal state, next_state: unsigned (R-1 downto 0);
```

```
begin
    process (C)                                          -- Регістр пам'яті
    begin
        if rising_edge(C) then
           if Reset = '1' then
               state <= (others => '0');
           else
               state <= next_state;
           end if;
        end if;
    end process;

    process(state)                    -- Операційний автомат переходів
    begin
        if state = "1100011" then -- Якщо досягнутий стан з кодом 99
            next_state <= (others => '0');     -- Перехід до стану a0
        else
            next_state <= state + 1;           -- Операція інкременту
        end if;
    end process;

    D <= state;
end architecture FSM_A;
```

Лістинг А.18 –VHDL-модель МПА с ОАП на базі регістру зсуву зі зворотним зв'язком (ГСА Γ_{5-1}, 100 станів)

```
architecture FSM_A of FSM is
signal state, next_state: unsigned (R-1 downto 0);
begin

    process (C)                                          -- Регістр пам'яті
    begin
        if rising_edge(C) then
           if Reset = '1' then
               state <= "1111111";           -- Код початкового стану
           else
               state <= next_state;
           end if;
        end if;
    end process;

    process(state)                    -- Операційний автомат переходів
    begin
```

```
        if state = "1111000" then         -- Якщо досягнутий стан a99
            next_state <= "1111111";      -- Перехід до стану a0
        else

         -- Зсув коду поточного стану state ліворуч із заповненням
         -- молодшого розряду сумою за модулем 2 розрядів 6 та 0
         -- відповідно до поліному x^6+1
        next_state <= state(R-2 downto 0) & (state(6) xor state(0));

        end if;
    end process;

    D <= state;
end architecture FSM_A;
```

Лістинг А.19 – Фрагмент VHDL-моделі МПА з ОАП (ГСА Γ_{5-2}, 100 станів)

```
architecture OAP_A of OAP is
signal state, next_state: unsigned (R-1 downto 0);
begin

    process (C)
    begin
        if rising_edge(C) then
           if Reset = '1' then
               state <= "0000000";
           else
               state <= next_state;
           end if;
        end if;
    end process;

    process (state, x1)                   -- ОАП
    begin

       -- Якщо стан лежить в диапазоні від a0 до A(M-10)
       if state < "1011010" then
          if x1 = '0' then                -- По x1=0
              next_state <= state + 1;    -- Виконуємо інкремент
          else                            -- По x1=1
               next_state <= state + 10; -- Виконуємо операцію «+10»
          end if;
       elsif state = "1100011" then     -- Якщо це саме стан a(M-1)
          next_state <= "0000000";      -- Перехід в початковий стан
```

```
      -- Якщо це один з останніх десяти станів
      else
         if x1 = '0' then              -- По x1=0
            next_state <= state + 1;  -- Перехід в наступний стан
         else                          -- По x1=1
            next_state <= "0000000";  -- Перехід в початковий стан
         end if;
      end if;
   end process;

   D <= state;
end architecture OAP_A;
```

Лістинг А.20 – Фрагмент VHDL-моделі МПА з ОАП (ГСА Γ_{5-3}, 100 станів)

```
architecture FSM_A of FSM is
signal state, next_state: std_logic_vector(R-1 downto 0);
begin

       process(C)                              -- Регістр пам'яті
       begin
            if rising_edge(C) then
              if Reset = '1' then
                   state <= "1111111";
               else
                   state <= next_state;
              end if;
            end if;
       end process;

    process (state, x1)                       -- МПА з ОАП
    begin
            if x1 = '0' then
               next_state <= state(R-2 downto 0) & (state(6)
                             xor state(0));
            else
                   case state is
                        when "1111111" => next_state <= "0101011";
                        when "1111110" => next_state <= "1100011";
                        when "1111101" => next_state <= "0010011";
                        ...
                        when "1001111" => next_state <= "1100100";
                        when "0011110" => next_state <= "1010110";
                        when "0111100" => next_state <= "1011011";
                        when others => next_state <= "1111111";
```

```
                end case;
            end if;
        end process;

        D <= state;
end FSM_A;
```

Лістинг А.21 – Фрагмент VHDL-моделі МПА з ОАП (ГСА Γ_{5-4}, 100 станів)

```
entity FSM is
    generic(R: integer := 7);
    port (X: in std_logic_vector(1 to 100);
          C: in std_logic;
          Reset: in std_logic;
          D: out unsigned(R-1 downto 0));
end FSM;

architecture FSM_A of FSM is
signal state, next_state: unsigned(R-1 downto 0);
signal p: std_logic;                                  -- Вхідна змінна
після заміни
begin
        process (C)                                   -- Регістр пам'яті
        begin
            if rising_edge(C) then
              if Reset = '1' then
                  state <= "0000000";
               else
                  state <= next_state;
              end if;
            end if;
        end process;

     process (state, X)           -- М-підсхема
     begin
                case state is -- Реалізація заміни вхідних змінних
                      when "0000000" => p <= X(1);
                      when "0000001" => p <= X(2);
                      when "0000010" => p <= X(3);
                      ...
                      when "1100010" => p <= X(99);
                      when others => p <= X(100);
                end case;
     end process;
```

```
    process (state, p)                          -- ОАП
    begin
       if state < "1011010" then  -- Якщо це стан від a0 до a(M-10)
            if p = '0' then                     -- Якщо ЛУ хибна
                 next_state <= state + 1;       -- Інкремент
            else                                -- Якщо ЛУ істинна
                 next_state <= state + 10;      -- Операція «+10»
            end if;
       elsif state = "1100011" then  -- Якщо досягнутий стан a(M-1)
            next_state <= "0000000";            -- Перехід до стану a0
       else                          -- Якщо це стан з останніх десяти
            if p = '0' then                     -- Якщо ЛУ хибна
                 next_state <= state + 1;       -- Інкремент
            else                                -- Якщо ЛУ істинна
                 next_state <= "0000000";       -- Перехід до стану a0
            end if;
       end if;
    end process;

    D <= state;
end FSM_A;
```

Лістинг А.22 – Фрагмент VHDL-моделі МПА з ОАП (ГСА Γ_{5-5}, 100 станів)

```
       process (state, p)
       begin
           if p = '0' then
                   next_state <= state(R-2 downto 0) & (state(6)
                                 xor state(0));
             else
                   case state is
                        when "1111111" => next_state <= "0101011";
                        when "1111110" => next_state <= "1100011";
                        ...
                        when "0011110" => next_state <= "1010110";
                        when "0111100" => next_state <= "1011011";
                        when others => next_state <= "1111111";
                   end case;
            end if;
       end process;

       D <= state;
end FSM_A;
```

ДОДАТОК Б

Приклад алгебраїчного синтезу мікропрограмного автомата з операційним атвоматом переходів

Мікропрограмний автомат заданий ГСА $\Gamma_{Б-6}$, що позначена станами автомата Мура (рис. Б.1).

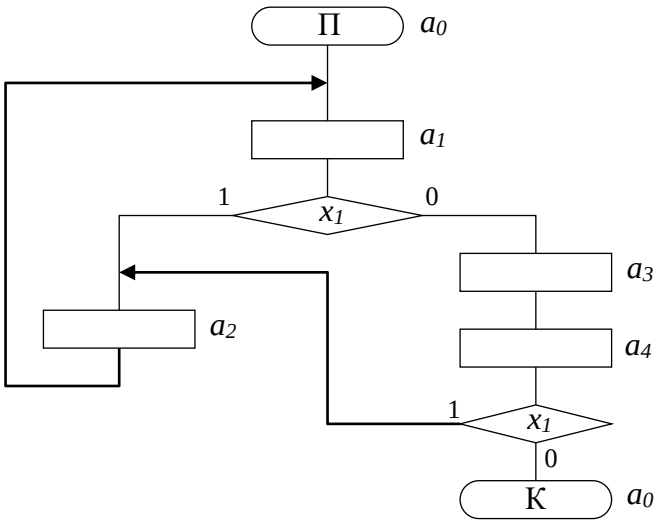


Рисунок Б.1 – Граф-схема алгоритму $\Gamma_{Б-1}$

Крок 1. ГСА $\Gamma_{Б-1}$ містить $M = 5$ станів $a_0 - a_4$. Розрядність структурного коду стану $R = R_{min} = \lceil log_2 M \rceil = 3$. Тоді множина припустимих для використання структурних (двійкових) кодів станів $K_S^R = \{000, 001, 010, 011, 100, 101, 110, 111\}$.

Крок 2. Нехай множина операцій переходів $O = \{O_1, O_2\}$, де

$$O_1: \quad K_{I_1}(a^{t+1}) = (K_{I_1}(a^t) + 5) \bmod 8, \qquad (Б.1)$$

$$O_2: \quad K_{I_2}(a^{t+1}) = K_{I_2}(a^t) \oplus 100_2. \qquad (Б.2)$$

Відповідно до операції O_1 структурні коди станів інтерпретуються як цілі числа без знака в діапазоні [0; 7]. Ця множина чисел збігається з областю визначення та областю припустимих значень операції O_1.

У випадку операції O_2 структурні коди станів інтерпретуються як двійкові вектори. Область визначення і область припустимих значень операції O_2 однакові й збігаються із множиною K_S^R.

Крок 3. Нехай $p(x_1)$=0,2. Для визначення значень $q(a_i)$ може бути використаний метод, викладений у [125]. Результат його застосування до ГСА $\Gamma_{Б-1}$ наведений у табл. Б.1.

Таблиця Б.1

Значення $q(a_i)$ для ГСА Γ_{4-6} при $p(x_1)$=0,2

a_i	a_0	a_1	a_2	a_3	a_4
$q(a_i)$	1	6,25	5,25	5	5

При цьому середнє число тактів, що витрачається на один прохід ГСА, $Q=\sum_{i=0}^{4} q(a_i)$ =22,5.

Крок 4. Згідно табл. Б.1, максимальне значення q має стан a_1.

Крок 5. Множина отриманих формальних розв'язків оголошується порожньою.

Крок 6. Організується перебір структурних кодів з множини K_S^R в послідовному порядку від 000 до 111.

Крок 7. Стану a_1, що обраний на кроці 4, надається структурний код $K_S(a_1)$=000.

Крок 8. Зі стану a_1 є два переходи: $a_1 \to a_2$ по x_1 та $a_1 \to a_3$ по $\bar{x}_1$. Спробуємо реалізувати дані переходи за допомогою наявних операцій переходів.

1. Реалізуємо перехід $a_1 \to a_2$. Перевіримо, чи можлива реалізація даного переходу за допомогою операції O_1. При $K_{I_1}(a_1)$=0 одержуємо $K_{I_1}(a_2)$=5. Даному значенню відповідає структурний код 101, який на даний момент не використовується для кодування інших станів. Отже, перехід $a_1 \to a_2$ може бути

реалізований за допомогою операції O_1. Стану a_2 надаємо структурний код $K_S(a_2)$=101.

2. Реалізуємо перехід $a_1 \to a_3$. Використання операції O_1 неможливе, тому що вона вже використана для реалізації іншого переходу зі стану a_1 і її повторне використання не дозволить надати стану a_3 унікальний структурний код.

Перевіримо, чи можлива реалізація даного переходу за допомогою операції O_2. При $K_{I_2}(a_1)$=000 одержуємо $K_{I_2}(a_3)$=100. Даному значенню відповідає структурний код 100, який на даний момент не використовується для кодування інших станів. Отже, перехід $a_1 \to a_3$ може бути реалізований за допомогою операції O_2. Стану a_3 надаємо структурний код $K_S(a_3)$=100.

Таким чином, усі переходи зі стану з максимальним значенням *q* реалізовані за допомогою операцій переходів з множини *O*. Переходимо до кроку 9.

Крок 9. При виконанні кроку 9 будемо спиратися на наступні результати, отримані на попередніх кроках:

$K_S(a_1)$=000, $K_S(a_2)$=101, $K_S(a_3)$=100.

Крок 9.1. $A' = \{a_0, a_2, a_3, a_4\}$, $A'' = \{a_1\}$.

Крок 9.2 (ітерація 1). З A' вибираємо стан з максимальним значенням *q*. Згідно табл. Б.1, таким станом є a_2. На даний момент структурний код $K_S(a_2)$ відомий і дорівнює 101.

Крок 9.3 (ітерація 1). Реалізуємо для стану a_2 всі вихідні переходи. У заданій ГСА єдиним таким переходом є безумовний перехід $a_2 \to a_1$. При $K_S(a_2)$=101 використання ОП O_1 для реалізації даного переходу приводить до структурного коду 010, використання O_2 – до коду 001. Обидва коди на даний момент не використовуються для кодування інших станів, однак не дорівнюють структурному коду стану переходу $K_S(a_1)$=000. Отже, реалізація переходу $a_2 \to a_1$ за допомогою однієї ОП у цьому випадку неможлива.

Той факт, що структурні коди 010 та 001, одержувані шляхом виконання над кодом $K_S(a_2)$=101 операцій O_1 та O_2 відповідно, не використовуються для кодування інших станів, дозволяє спробувати реалізувати даний перехід із використанням транзитних станів. Дана задача полягає в наступному: знайти таку послідовність станів мінімальної довжини, для якої:

– перехід зі стану a_2 в перший стан послідовності, переходи між станами, а також перехід з останнього стану до стану a_1 можуть бути реалізовані за допомогою операцій із множини *O*;

– структурні коди станів не використовуються на даний момент для кодування інших станів ГСА;

– кількість станів у послідовності не перевищує кількість резервних структурних кодів, яка на даному кроці дорівнює ($2^R - M$) = 3.

Процес формування послідовності транзитних станів можна представити у вигляді дерева, зображеного на рис. Б.2.

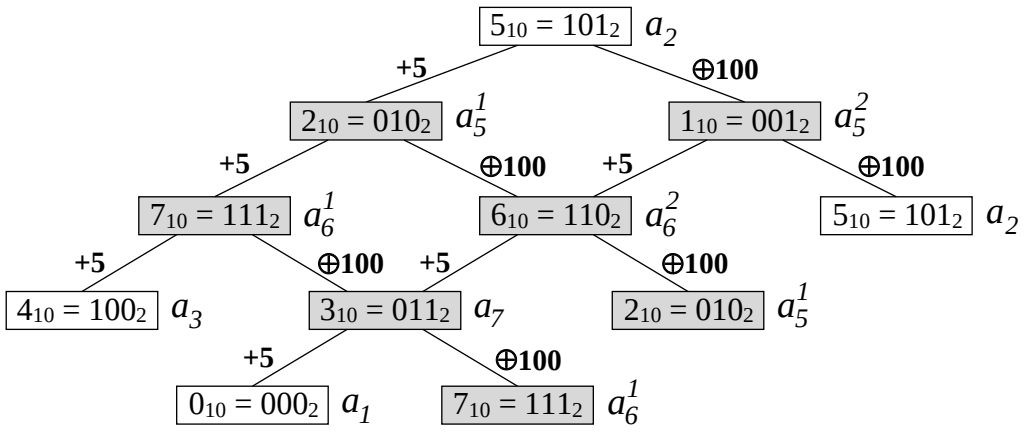


Рисунок Б.2 – Графічне представлення
пошуку послідовності транзитних станів

На верхньому рівні дерева зображена вихідна вершина, відзначена станом a_2. Його структурний код $K_S(a_2)$=101 інтерпретується як $K_{I_1}(a_2)$=5 або $K_{I_2}(a_2)=K_S(a_2)$=101. Дані проміжні коди, розділені знаком рівності, вказані всередині вершини.

Ребра, що ведуть зі стану a_2, відповідають операціям переходів O_1 та O_2, що виконуються над відповідними проміжними кодами стану a_2. Результатом виконання ОП O_1 є структурний код 010, що зіставляється транзитному стану a_5^1; результатом ОП O_2 – структурний код 001, що зіставляється транзитній вершині a_5^2. Відповідно, $K_{I_1}(a_5^1)$=2, $K_{I_2}(a_5^1)$=010, $K_{I_1}(a_5^2)$=1, $K_{I_2}(a_5^2)$=001.

Вершини, позначені станами a_5^1 та a_5^2, перебувають на другому рівні дерева. Нижній індекс «5» даних станів вказує, що кожен з них буде першим транзитним станом в послідовності, що додається, і одержить індекс, що дорівнює значенню *M*. Верхні індекси відповідають двом можливим варіантам побудови послідовності транзитних станів.

Виконання над кодом $K_{I_1}(a_5^1)$=2 операції O_1 дає проміжний код 7, якому відповідає структурний код 111, що не використовується на даний момент для кодування інших станів. Зіставимо даний код транзитному стану a_6^1, який розташований на третьому рівні дерева.

Виконання операції O_2 над кодом $K_{I_2}(a_5^1)$=2 і операції O_1 над кодом $K_{I_1}(a_5^2)$=1 дає в результаті однаковий структурний код 110. Оскільки даний код наразі вільний, закодуємо їм транзитний стан a_6^2. При цьому $K_{I_1}(a_6^2)$=6, $K_{I_2}(a_6^2)$=110.

Виконання операції O_2 над кодом $K_{I_2}(a_5^2)$=001 дає структурний код 101, який вже використаний для кодування стану a_2. Отже, дана гілка не може привести в кінцевий стан a_1 і є тупиковою. Такими ж тупиковими гілками є переходи зі стану a_6^1 за допомогою O_1, зі стану a_6^2 за допомогою O_2 та зі стану a_7 за допомогою O_2.

Виконання операції O_1 над кодом $K_{I_1}(a_7)$=3 стану a_7 дає структурний код 000, що дорівнює структурному коду стану переходу a_1. Таким чином, перехід

$a_2 \to a_1$ може бути реалізований за допомогою операцій із заданої множини ОП із використанням послідовності з трьох транзитних станів. Згідно з рис. Б.2, існують три різні транзитні послідовності:

$$a_2 \to a_5^1 \to a_6^1 \to a_7 \to a_1 ;$$

$$a_2 \to a_5^1 \to a_6^2 \to a_7 \to a_1 ;$$

$$a_2 \to a_5^2 \to a_6^2 \to a_7 \to a_1 .$$

Транзитні стани, що додаються, перебувають в одній гілці зі станом a_2, що робить їхні значення q рівними $q(a_2) = 5{,}25$. Кожна з даних послідовностей складається з трьох транзитних станів і збільшує середнє число Q тактів роботи автомата на величину $\Delta Q = 3 \cdot q(a_2) = 15{,}75$. У відсотковому відношенні до величини $Q = 22{,}5$, що отримана на кроці 3, значення $\Delta Q = 70\%$, тобто середній час виконання автоматом заданого алгоритму збільшується більш ніж у півтора рази.

Іншим недоліком використання кожної з отриманих транзитних послідовностей є те, що кожна послідовність містить максимально можливу для заданої ГСА кількість транзитних станів, обумовлену значеннями *M* та *R*. Це унеможливлює використання транзитних станів при реалізації інших переходів і може приводити до збільшення числа переходів, що реалізуються канонічним способом, і як наслідок – до збільшення апаратурних витрат у логічній схемі автомата.

Проте, метод алгебраїчного синтезу, що використовується, зобов'язує реалізувати перехід транзитним способом, якщо такий спосіб можливий. Виберемо першу з трьох можливих транзитних послідовностей: $a_2 \to a_5^1 \to a_6^1 \to a_7 \to a_1$. Одержуваний результат графічно представлений на рис. Б.3. Стани a_5^1 та a_6^1 у межах ГСА позначені символами a_5 та a_6 відповідно. Вершини, що відповідають транзитним станам, показані затемненим фоном.

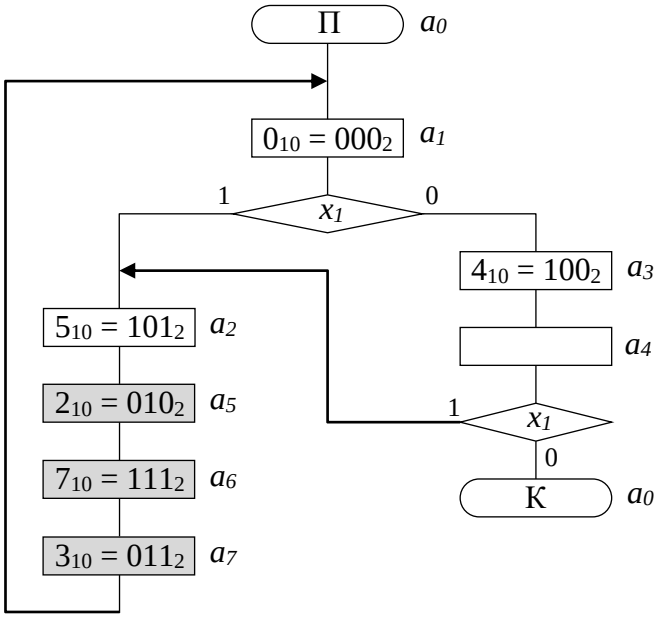


Рисунок Б.3 – Графічне представлення реалізації переходу $a_2 \to a_1$ за допомогою послідовності транзитних станів (ГСА Γ_{4-6})

Крок 9.4 (ітерація 1). $A' = A' \setminus a_2 = \{a_0, a_3, a_4\}$; $A'' = \{a_1, a_2\}$. Відзначимо, що транзитні стани, додані в ГСА на попередньому кроці, не відображаються в множинах A' та A'', оскільки для будь-якого транзитного стану вхідний і вихідний переходи завжди реалізуються в процесі формування транзитної послідовності.

Крок 9.5 (ітерація 1). $A' \neq \varnothing$, переходимо до кроку 9.2.

Крок 9.2 (ітерація 2). Із множини A' вибираємо стан з максимальним значенням q. Згідно з табл. Б.1, такими станами є a_3 та a_4. Вибираємо кожний з них, наприклад, a_3, для якого на даний момент структурний код відомий і дорівнює $K_S(a_3)$=100. Відповідно, $K_{I_1}(a_3)$=4, $K_{I_2}(a_3) = K_S(a_3)$=100.

Крок 9.3 (ітерація 2). Зі стану a_3 існує єдиний вихідний безумовний перехід у стан a_4. Оскільки структурний код стану a_4 на даний момент не заданий, його слід вибрати з множини вільних структурних кодів: {001, 110}. Нехай $K_S(a_4)$=001. Тоді $K_{I_1}(a_4)$=1, $K_{I_2}(a_4) = K_S(a_4)$=001.

Спробуємо реалізувати перехід $a_3 \to a_4$ за допомогою однієї з операцій переходів. Виконання ОП O_1 над кодом $K_{I_1}(a_3)$=4 дає в результаті значення 1, що дорівнює проміжному коду $K_{I_1}(a_4)$. Таким чином, перехід $a_3 \to a_4$ може бути реалізований операційним способом за допомогою ОП O_1 без використання транзитних станів.

Крок 9.4 (ітерація 2). $A' = A' \setminus a_3 = \{a_0, a_4\}$; $A'' = \{a_1, a_2, a_3\}$.

Крок 9.5 (ітерація 2). $A' \neq \varnothing$, переходимо до кроку 9.2.

Крок 9.2 (ітерація 3). Із множини A' вибираємо стан з максимальним значенням q. Згідно з табл. Б.1, таким станом є a_4. На даний момент коди цього стану відомі: $K_{I_1}(a_4)$=1, $K_{I_2}(a_4) = K_S(a_4)$=001.

Крок 9.3 (ітерація 3). Зі стану a_4 існують два переходи: перехід за умовою x_1 в стан a_2 та перехід за умовою $\bar{x}_1$ в стан a_0.

Реалізуємо перехід $a_4 \to a_2$. Помітимо, що при $K_{I_2}(a_4)$=001 та $K_{I_2}(a_2)$=101 даний перехід може бути реалізований за допомогою ОП O_2 без використання транзитних станів.

Реалізуємо перехід $a_4 \to a_0$. Оскільки $K_S(a_0)$ не заданий, його слід вибрати з множини структурних кодів, що наразі не використовуються. На даний момент єдиним вільним кодом є 110. Таким чином, $K_S(a_0)$=110, $K_{I_1}(a_0)$=6, $K_{I_2}(a_0)$=110. Помітимо, що при $K_{I_1}(a_4)$=1 та $K_{I_1}(a_0)$=6 даний перехід може бути реалізований за допомогою ОП O_1 без використання транзитних станів.

Крок 9.4 (ітерація 3). $A' = A' \setminus a_4 = \{a_0\}$; $A'' = \{a_1, a_2, a_3, a_4\}$.

Крок 9.5 (ітерація 3). $A' \neq \varnothing$, переходимо до кроку 9.2.

Крок 9.2 (ітерація 4). Із множини A' вибираємо стан з максимальним значенням q. Таким станом є, очевидно, a_0, для якого $K_{I_1}(a_0)$=6, $K_{I_2}(a_0) = K_S(a_0)$=110.

Крок 9.3 (ітерація 4). Єдиним переходом зі стану a_0 є безумовний перехід до стану a_1, причому $K_{I_1}(a_1)$=0, $K_{I_2}(a_1)=K_S(a_1)$=000. Виконання ОП O_1 над кодом $K_{I_1}(a_0)$=6 дає в результаті значення 3, що не збігається з кодом $K_{I_1}(a_1)$=0. Виконання над кодом $K_{I_2}(a_0)$=110 операції O_2 дає значення 010, що не збігається з кодом $K_{I_2}(a_1)$=000. Таким чином, перехід $a_0 \to a_1$ не можна реалізувати операційним способом за допомогою однієї із заданих ОП.

Також неможлива реалізація даного переходу з використанням транзитних станів, оскільки в множині K_S^R відсутні вільні структурні коди. Отже, єдиним способом реалізації переходу є канонічна реалізація.

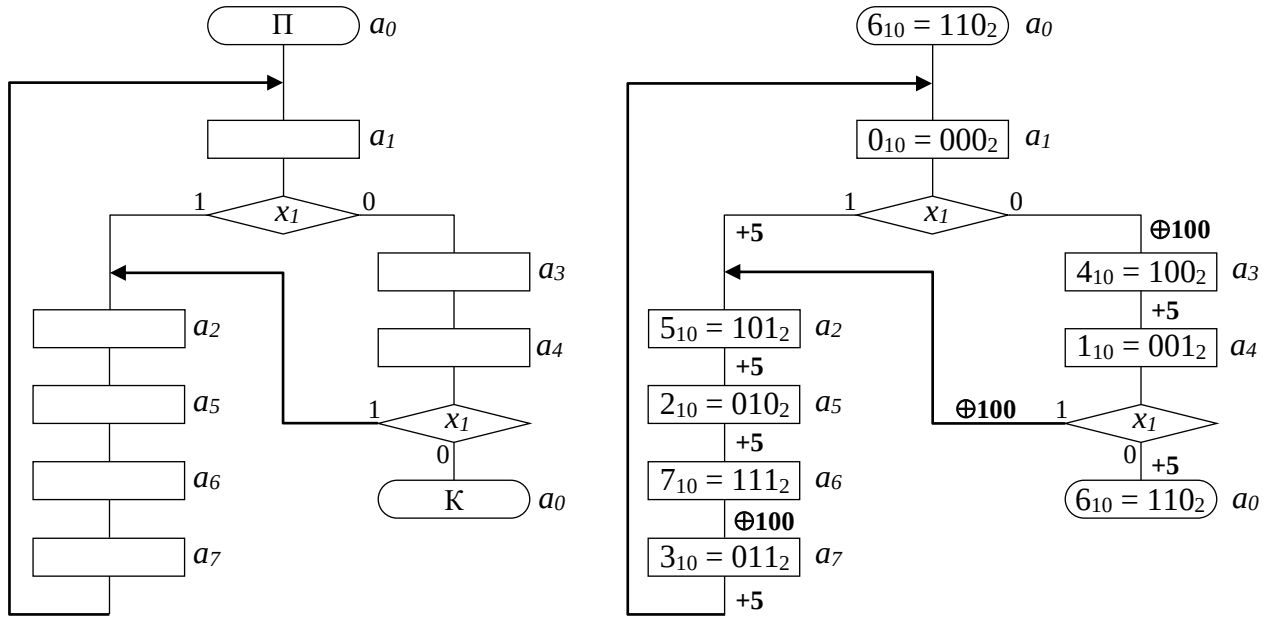
Крок 9.4 (ітерація 4). $A' = A' \setminus a_0 = \varnothing$; $A'' = \{a_1, a_2, a_3, a_4, a_0\}$.

Крок 9.5 (ітерація 4). $A' = \varnothing$, переходимо до кроку 10.

Крок 10.

У процесі алгебраїчного синтезу у вихідну ГСА $\Gamma_{Б-1}$ додано три порожні операторні вершини, відзначені станами $a_5 - a_7$ автомата Мура. Частина переходів реалізована за допомогою ОП O_1, частина – за допомогою O_2, частина – канонічним способом. На рис. Б.4, *а* зображена перетворена ГСА $\Gamma'_{Б-1}$ з доданими транзитними станами, на рис. Б.4, *б* – графічне представлення розв'язку задачу алгебраїчного синтезу, отриманого в результаті кроків 7–9.

На рис. Б.4, *б* у вершинах записані проміжні коди відповідних станів у форматі «$K_{I_1}(a_i) = K_{I_2}(a_i)$». Ребра відзначені операціями переходів, що використовуються для реалізації відповідних їм автоматних переходів (ОП O_1 позначена як «+5», O_2 – як «⊕100»). Перехід $a_0 \to a_1$ реалізується канонічним способом, тому відповідне до нього ребро позначення не має. Сформуємо формальний розв'язок задачі алгебраїчного синтезу, отриманий в результаті виконання кроків 7–9.



а) б)

Рисунок Б.4 – Перетворена ГСА $\Gamma'_{Б-1}$ (а) та графічне представлення розв'язку задачі алгебраїчного синтезу МПА з ОАП (б)

Задамо абстрактну алгебру переходів автомата, що відповідає ГСА $\Gamma'_{Б-1}$. У цій ГСА присутні три типи переходів: безумовний перехід, перехід за умовою x_1 та перехід за умовою $\bar{x}_1$, яким в абстрактному автоматі відповідають абстрактні вхідні сигнали $z_0 - z_2$. Ототожнюючи дане ЛУ з однойменним вхідним структурним сигналом, закодуємо сигнали $z_0 - z_2$, структурними кодами у вигляді векторів $< x_1 >$, у яких x_1 може набувати значення з множини {–, 0, 1}: $K_S(z_0) = <->,\ K_S(z_1) = <1>,\ K_S(z_2) = <0>.$

Абстрактна алгебра переходів задається виразом (Б.3).

$$\begin{cases} G_\delta = <\{A, Z\}, \{\delta\}>; \\ A = \{a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7\}; \\ Z = \{z_0, z_1, z_2\}; \\ \delta = \{<a_0, z_0, a_1>, <a_1, z_1, a_2>, <a_1, z_2, a_3>, <a_2, z_0, a_5>, \\ \quad <a_5, z_0, a_6>, <a_6, z_0, a_7>, <a_7, z_0, a_1>, <a_3, z_0, a_4>, \\ \quad <a_4, z_1, a_2>, <a_4, z_1, a_0>\}. \end{cases} \qquad (Б.3)$$

Структурна алгебра переходів, ізоморфна алгебрі (Б.3), задається виразом (Б.4), у якому стани та вхідні сигнали представлені своїми структурними кодами.

$$\begin{cases} G_S = <\{K_S(A),\ K_S(Z)\}, d>; \\ K_S(A) = \{110, 000, 101, 100, 001, 010, 111, 011\}; \\ K_S(Z) = \{-, 1, 0\}; \\ d = \{<110, -, 000>, <000, 1, 101>, <000, 0, 100>, <101, -, 010>, \\ \quad <010, -, 111>, <111, -, 011>, <011, -, 000>, <100, -, 001>, \\ \quad <001, 1, 101>, <001, 0, 110>\}. \end{cases} \quad (Б.4)$$

Оскільки частина переходів реалізується за допомогою ОП O_1, частина – за допомогою O_2, частина – канонічним способом, абстрактна і структурна функції переходів розбиваються відповідним чином на три часткові функції кожна. На їхній основі будуються три абстрактні підалгебри переходів (вирази (Б.5) – (Б.7)) та три структурні підалгебри переходів (вирази (Б.8) – (Б.10)). Аналіз даних виразів підтверджує існування ізоморфізмів $G_{\delta_1} \leftrightarrow G_{d_1}$, $G_{\delta_2} \leftrightarrow G_{d_2}$ та $G_{\delta_3} \leftrightarrow G_{d_3}$.

$$\begin{cases} G_{\delta_1} = <\{A_{\delta_1}, Z_{\delta_1}\}, \{\delta_1\}>; \\ A_{\delta_1} = \{a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7\}; \\ Z_{\delta_1} = \{z_0, z_1, z_2\}; \\ \delta_1 = \{<a_1, z_1, a_2>, <a_2, z_0, a_5>, <a_5, z_0, a_6>, \\ \quad <a_7, z_0, a_1>, <a_3, z_0, a_4>, <a_4, z_2, a_0>\}. \end{cases} \quad (Б.5)$$

$$\begin{cases} G_{\delta_2} = <\{A_{\delta_2}, Z_{\delta_2}\}, \{\delta_2\}>; \\ A_{\delta_2} = \{a_1, a_2, a_3, a_4, a_6, a_7\}; \\ Z_{\delta_2} = \{z_0, z_1, z_2\}; \\ \delta_2 = \{<a_1, z_2, a_3>, <a_6, z_0, a_7>, <a_4, z_1, a_2>\}. \end{cases} \quad (Б.6)$$

$$\begin{cases} G_{\delta_3} = <\{A_{\delta_3}, Z_{\delta_3}\}, \{\delta_3\}>; \\ A_{\delta_3} = \{a_0, a_0\}; \\ Z_{\delta_3} = \{z_0\}; \\ \delta_3 = \{<a_0, z_0, a_1>\}. \end{cases} \quad (Б.7)$$

$$\begin{cases} G_{d_1} = <\{K_S(A_{d_1}), K_S(Z_{d_1})\}, \{d_1\}>; \\ K_S(A_{d_1}) = \{110, 000, 101, 100, 001, 010, 111, 011\}; \\ K_S(Z_{d_1}) = \{-, 1, 0\}; \\ d_1 = \{<000, 1, 101>, <101, -, 010>, <010, -, 111>, \\ \quad <011, -, 000>, <100, -, 001>, <001, 0, 110>\}. \end{cases} \quad (Б.8)$$

$$\begin{cases} G_{d_2} = <\{K_S(A_{d_2}), K_S(Z_{d_2})\}, \{d_2\}>; \\ K_S(A_{d_2}) = \{000, 101, 100, 001, 111, 011\}; \\ K_S(Z_{d_2}) = \{-, 1, 0\}; \\ d_2 = \{<000, 0, 100>, <111, -, 011>, <001, 1, 101>\}. \end{cases} \quad (Б.9)$$

$$\begin{cases} G_{d_3} = <\{K_S(A_{d_3}), K_S(Z_{d_3})\}, \{d_3\}>; \\ K_S(A_{d_3}) = \{110, 001\}; \\ K_S(Z_{d_3}) = \{-\}; \\ d_3 = \{<110, -, 000>\}. \end{cases} \quad (Б.10)$$

Задамо також дві проміжні алгебри переходів G_{I_1} та G_{I_2}, сигнатури яких утворені операціями переходів O_1 та O_2 відповідно.

$$\begin{cases} G_{I_1} = <\{K_{I_1}(A_{\delta_1}), K_{I_1}(Z_{\delta_1})\}, \{O_1\}>; \\ K_{I_1}(A_{\delta_1}) = \{6, 0, 5, 4, 1, 2, 7, 3\}; \\ K_{I_1}(Z_{\delta_1}) = \{z_0, z_1, z_2\}; \\ O_1 = \{<0, z_1, 5>, <5, z_0, 2>, <2, z_0, 7>, \\ \quad <3, z_0, 0>, <4, z_0, 1>, <1, z_2, 6>\}. \end{cases} \quad (Б.11)$$

$$\begin{cases} G_{I_2} = <\{K_{I_2}(A_{\delta_2}), K_{I_2}(Z_{\delta_2})\}, \{O_2\}>; \\ K_{I_2}(A_{\delta_2}) = \{000, 101, 100, 001, 111, 011\}; \\ K_{I_2}(Z_{\delta_2}) = \{z_0, z_1, z_2\}; \\ O_2 = \{<000, z_2, 100>, <111, z_0, 011>, <001, z_1, 101>\}. \end{cases} \quad (Б.12)$$

У виразах (Б.11) і (Б.12) операції переходів O_1 та O_2 представлені множиною векторів виду $<K_{I_n}(a_i), z_j, K_{I_n}(a_k)>$, яка відповідає множині переходів, що реалізуються даною ОП. Це дозволяє встановити ізоморфізми $G_{\delta_1} \leftrightarrow G_{I_1} \leftrightarrow G_{d_1}$ та $G_{\delta_2} \leftrightarrow G_{I_2} \leftrightarrow G_{d_2}$. Також відзначимо, що оскільки обидві

ОП, згідно з виразами (Б.1) і (Б.2), не використовують у якості аргументів вхідні сигнали автомата, у векторах операцій вхідні сигнали присутні з формальною метою і вказані в абстрактному вигляді.

Таким чином, можна констатувати існування наступної системи ізоморфізмів:

$$\begin{cases} G_{\delta_1} \leftrightarrow G_{I_1} \leftrightarrow G_{d_1}; \\ G_{\delta_2} \leftrightarrow G_{I_2} \leftrightarrow G_{d_2}; \\ G_{\delta} \leftrightarrow G_{d}. \end{cases} \qquad \text{(Б.13)}$$

Дана система разом із виразами (Б.3) – (Б.12) являє собою формальний розв'язок задачі алгебраїчного синтезу МПА з ОАП, що задається ГСА $\Gamma_{Б-1}$, для множини операцій переходів, утвореної операціями (Б.1) та (Б.2), при кодуванні стану a_1 структурним кодом $K_S(a_1)$=000.

Метод алгебраїчного синтезу припускає на кроці 10 додавання отриманого формального розв'язку до множини знайдених формальних розв'язків, після чого на кроці 11 має бути здійснена наступна ітерація головного циклу для чергового значення $K_S(a_1)$. Крім зробленої ітерації для $K_S(a_1)$=000, має бути зроблено сім ітерацій для значень $K_S(a_1)$ від 001 до 111, у кожній з яких може бути отриманий інший формальний розв'язок задачі алгебраїчного синтезу. Розглянутий приклад обмежений лише першою ітерацією головного циклу методу.

ДОДАТОК В

Акти впровадження результатів дисертаційної роботи

Engineering
ИНЖИНИРИНГОВАЯ КОМПАНИЯ

Довідка № 10/07-19
від «10» липня 2019р.

про впровадження результатів дисертаційної роботи
Бабакова Романа Марковича
на здобуття наукового ступеня доктора технічних наук
«Структури і методи синтезу мікропрограмних автоматів
з операційним перетворенням кодів станів»

Довідка видана в тому, що розроблені в дисертаційній роботі структури та методику синтеза мікропрограмного автомата з операційним автоматом переходів впроваджено в практичну діяльність ТОВ «С-інжиніринг» при проектуванні спеціалізованих пристроїв керування. Використання розроблених структур дозволило вирішити актуальну задачу зниження вартості пристроїв керування за рахунок зменшення апаратурних витрат в схемі мікропрограмного автомата.

Результати впровадження довели, що розроблені Бабаковим Р.М. засоби виконані на високому науково-технічному рівні та мають переваги над існуючими рішеннями, що підтверджується досвідом ТОВ «С-інжиніринг».

Заступник генерального директора
по науці ТОВ «С-інжинірінг» О.М. Семенюг

ул. Николая Боровского, 28, корпус 47, г. Одесса, 65031, Украина
тел.: +38 048 730 57 31; 730 57 33; т/ф.: +38 048 730 57 40
info@se.ua www.se.ua

ЗАТВЕРДЖУЮ

Головний конструктор
ДНВП «Об'єднання «Комунар»
Начальник НТ СКБ «Поліграфія»
к.т.н., доцент
заслужений винахідник України
Сидоренко М.Ф.
„27" вересня 2019 року

АКТ
реалізації результатів дисертаційної роботи
Бабакова Романа Марковича
«Структури і методи синтезу мікропрограмних автоматів
з операційним перетворенням кодів станів»,
виконаної на здобуття наукового ступеня доктора технічних наук

Комісія у складі голови – заступника начальника відділу к.т.н. Тарасенка В.В., членів комісії – начальника лабораторії Авраменка І,Є., начальника бюро Жеребкіной Т.Ф. склала даний акт в тому, що при розробленні бортових авіаційних обчислювальних систем та пристроїв було використано наступні нові наукові результати досліджень Бабакова Р.М.:

1) метод перетворення кодів станів мікропрограмного автомата за допомогою спеціалізованого операційного автомата;

2) структурна та математична моделі мікропрограмного автомата з операційним перетворенням кодів станів;

3) структури мікропрограмних автоматів з операційним перетворенням кодів станів та додатковою оптимізацією апаратурних витрат;

4) метод синтезу мікропрограмного автомата з операційним перетворенням кодів станів;

5) результати аналізу ефективності мікропрограмних автоматів з операційним перетворенням кодів станів за критерієм апаратурних витрат.

Впровадження результатів досліджень Бабакова Р.М. дозволяє:

– знизити апаратурні витрати в схемах пристроїв керування цифрових систем на 5-20% при використанні елементного базису ПЛІС FPGA;

– забезпечити вимоги до швидкодії та енергоспоживання проектованих пристроїв керування;

– вдосконалити методи вибору структурної моделі пристрою керування в залежності від встановлених технологічних вимог.

Голова комісії: _____ В.В. Тарасенко

Члени комісії: _____ І.Є.Авраменко

_____ Т.Ф.Жеребкіна

ДОВІДКА

про впровадження у навчальних курсах, розроблених
при виконанні проекту ALIOT за програмою ЄС ERASMUS+,
результатів докторської дисертації Бабакова Романа Марковича
«Структури і методи синтезу мікропрограмних автоматів
з операційним перетворенням кодів станів»

Наукові результати, що одержані в дисертаційній роботі Бабакова Романа Марковича, використано при виконанні освітнього європейського проекту ERASMUS+ «ALIOT» – INTERNET OF THINGS: EMERGING CURRICULUM FOR INDUSTRY AND HUMAN APPLICATIONS, 573818-EPP-1-2016-1-UK-EPPKA2-CBHE-JP (2016 – 2020 р.), спрямованого на розроблення магістерських і аспірантських курсів, а також курсів підвищення кваліфікації фахівців з Інтернету речей. Результати досліджень, виконаних Бабаковим Романом Марковичем, зокрема:

– принцип перетворення кодів станів за допомогою множини арифметико-логічних операцій,

– структура мікропрограмного автомата з операційним перетворенням кодів станів, яка має оптимізовані витрати апаратури у порівнянні із канонічною структурою мікропрограмного автомата,

– постанова та підходи до вирішення задачі алгебраїчного синтезу мікропрограмного автомата з операційним перетворенням кодів станів

увійшли до курсу TM6 «Internet of Things for industrial systems», модуль TMM 6.4 «Development and hardware optimization of control units for IoT devices», який створено і використовується відповідно до завдань проекту ERASMUS+ «ALIOT».

Матеріали досліджень використано для розроблення розділу 55 підручника, який забезпечує лекційну частину курсу: Internet of Things for Human and Industry. In 3 volumes. Volume 3. Assessment and Implementation / Kharchenko V. S. (editor). Ministry of Education and Science of Ukraine, National Aerospace University “Kharkiv Aviation Institute”, 2019, 921 p.

До практичної частини курсу увійшов розроблений Бабаковим Р.М. практикум «Technique of development and hardware optimization of control units for IoT devices», який опубліковано у посібнику Yu.P. Kondratenko, R.M. Babakov, V.S. Kharchenko , O.O. Illiashenko et. al. Internet of Things for Industrial Systems: Trainings / Yu.P. Kondratenko, V.S. Kharchenko (Eds.). Ministry of Education and Science of Ukraine, Petro Mohyla Black Sea National University, Zaporizhzhia National Technical University, National Aerospace University KhAI, 2019. – 143 p.

Результати досліджень, які впроваджено у проекті, надали змогу забезпечити фундаментальність, наочність та практичність матеріалів.

Національний координатор проекту TEMPUS-ALIOT,
завідувач кафедри комп'ютерних систем, мереж і кібербезпеки
Національного аерокосмічного університету ім. М.Є. Жуковського «ХАІ»,
заслужений винахідник України
доктор технічних наук, професор В. С. Харченко
12 серпня 2019 р.

Менеджер проекту TEMPUS-ALIOT,
доцент кафедри комп'ютерних систем, мереж і кібербезпеки
кандидат технічних наук О.О. Іляшенко
12 серпня 2019 р.

Підписи засвідчую.
Декан факультету
радіоелектроніки, комп'ютерних систем та інфокомунікацій
кандидат технічних наук О.В. Одокієнко

ЗАТВЕРДЖУЮ
Проректор з науково-педагогічної
та виховної роботи
Одеського національного
політехнічного університету
_____ С.А. Нестеренко
«16» 07 2019 р.

А К Т

про впровадження в навчальний процес Одеського національного політехнічного університету результатів дисертаційного дослідження на здобуття наукового ступеня доктора технічних наук «Структури і методи синтезу мікропрограмних автоматів з операційним перетворенням кодів станів» доцента кафедри прикладної математики і теорії систем управління Донецького національного університету імені Василя Стуса Бабакова Романа Марковича

Комісія у складі директора інституту комп'ютерних систем, д.т.н., проф. Антощук С.Г., завідувача кафедри комп'ютерних інтелектуальних систем та мереж, к.т.н., доц. Шапоріна Р.О., професора кафедри комп'ютерних інтелектуальних систем та мереж, д.т.н., проф. Дрозда О.В. розглянула матеріали дисертаційного дослідження Бабакова Р.М. і прийшла до наступного висновку.

Запропоновані в дисертаційній роботі структурні моделі мікропрограмних автоматів та методика їхнього синтезу, а саме:

– мікропрограмний автомат з операційним автоматом переходів, який відрізняється від існуючих структур мікропрограмних автоматів архітектурою схеми формування переходів та принципом перетворення кодів станів та характеризуються зменшеними витратами апаратури;

– методика алгебраїчного синтезу мікропрограмного автомата з операційним автоматом переходів, яка дозволяє провести синтез логічної схеми автомата для заданого набору операцій переходів;

впроваджені у навчальний процес кафедри комп'ютерних інтелектуальних систем та мереж ОНПУ та використовуються у наступних дисциплінах:

1. У навчальній дисципліні «Технології проектування комп'ютерних систем» для бакалаврів спеціальності 123 «Комп'ютерна інженерія» у лекційному матеріалі та завданнях для самостійної роботи.

2. У навчальній дисципліні «Проектування і діагностика систем критичного застосування» для магістрів спеціальності 123 «Комп'ютерна інженерія» у лекційному матеріалі та лабораторному практикумі.

Директор ІКС, д.т.н., проф. _____ С.Г. Антощук

Зав. каф. КІСМ, к.т.н., доц. _____ Р.О. Шапорін

Проф. каф. КІСМ, д.т.н., проф. _____ О.В. Дрозд

ДОДАТОК Г

Список публікацій здобувача за темою дисертації

в яких опубліковані основні наукові результати дисертації:

1. Бабаков Р. М. Операционное преобразование кодов состояний в микропрограммном автомате: монография. Винница: «ТВОРИ», 2019. 208 с.

2. Бабаков Р. М. Алгебраический синтез микропрограммного автомата с операционным автоматом переходов. *Информационные технологии и компьютерная инженерия*. 2017. № 39. Т. 2. С. 35–41. (Журнал індексується міжнародною наукометричною базою Google Scholar).

3. Бабаков Р. М. Исследование аппаратурных затрат в микропрограммном автомате с операционным автоматом переходов. *Радиоэлектроника, информатика, управление*. 2017. № 4. С. 106–115. (Входить до міжнародних наукометричних баз **Web of Science**, DOAJ, BASE, Index Copernicus, Google Scholar).

4. Бабаков Р. М. Математическая модель микропрограммного автомата с операционным автоматом переходов. *Наукові праці Донецького національного технічного університету. Серія: Інформатика, кібернетика та обчислювальна техніка*. Красноармійськ: ДВНЗ «ДонНТУ». 2016. № 1 (22). С. 54–57. (Журнал індексується міжнародною наукометричною базою Google Scholar).

5. Бабаков Р. М. Обобщение математической модели микропрограммного автомата на счетчике. *Радиоэлектроника, информатика, управление*. 2018. № 1. С. 100–109. (Входить до міжнародних наукометричних баз **Web of Science**, BASE, Index Copernicus, Google Scholar).

6. Бабаков Р. М. Промежуточная алгебра переходов в микропрограммном автомате. *Радиоэлектроника, информатика, управление*. 2016. № 1. С. 64–73. (Входить до міжнародних наукометричних баз **Web of Science**, DOAJ, BASE, Index Copernicus, Google Scholar).

7. Бабаков Р. М. Синтез логической схемы микропрограммного автомата с операционным автоматом переходов. *Вчені записки Таврійського національного*

університету імені В.І. Вернадського. Серія: технічні науки. 2018. № 4. С. 96–99. (Журнал індексується міжнародною наукометричною базою Google Scholar).

8. Бабаков Р. М. Синтез микропрограммного автомата с операционным автоматом переходов методом полного перебора. *Вчені записки Таврійського національного університету імені В.І. Вернадського*. Серія: технічні науки. 2018. № 1. С. 70–74. (Журнал індексується міжнародною наукометричною базою Google Scholar).

9. Бабаков Р. М. Урахування імовірності станів у мікропрограмному автоматі з операційним автоматом переходів. *Наукові праці Вінницького національного технічного університету*. 2017. № 2. URL: https://praci.vntu.edu.ua/index.php/praci/article/view/505/500. (Входить до міжнародної наукометричної бази РІНЦ).

10. Бабаков Р. М. Формальное решение задачи алгебраического синтеза микропрограммного автомата с операционным автоматом переходов. *Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: технічні науки*. 2018. № 2. С. 103–107. (Журнал індексується міжнародною базою Google Scholar).

11. Babakov R. M. Using of method of replacement of input variables in microprogram finite-state machine with datapath of transitions. *Технологический аудит и резервы производства*. 2017. № 4/2 (36). С. 18–23. (Входить до міжнародних наукометричних баз Index Copernicus, BASE, DOAJ, EBSCO, РІНЦ, Google Scholar).

12. Бабаков Р.М. Методологические аспекты синтеза микропрограммных автоматов с операционным автоматом переходов. *Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: технічні науки*. 2019. № 2. С. 82–86. (Журнал індексується міжнародною базою Google Scholar).

13. Бабаков Р. М., Ярош И. В. Использование транзитных состояний в микропрограммном автомате с операционным автоматом переходов. *Наукові праці Донецького національного технічного університету. Серія: Обчислювальна техніка та автоматизація*. Красноармійськ: ДВНЗ «ДонНТУ». 2016. № 1 (29). С.

56–64. (Журнал індексується міжнародною наукометричною базою Google Scholar).

14. Бабаков Р. М., Ярош И. В. Операционный автомат переходов. *Наукові праці Донецького національного технічного університету. Серія: Обчислювальна техніка та автоматизація*. Красноармійськ: ДВНЗ «ДонНТУ». 2015. № 1 (28). С. 33–40. (Журнал індексується міжнародною наукометричною базою Google Scholar).

15. Бабаков Р. М., Ярош И. В. Формирование кодов операций переходов в микропрограммном автомате с операционным автоматом переходов. *Наукові праці Донецького національного технічного університету. Серія: Інформатика, кібернетика та обчислювальна техніка*. Красноармійськ: ДВНЗ «ДонНТУ». 2015. Випуск 1 (20). С. 11–16. (Журнал індексується міжнародною наукометричною базою Google Scholar).

16. Баркалов А. А., Бабаков Р. М. Алгебраическая интерпретация микропрограммного автомата с операционным автоматом переходов. *Кибернетика и системный анализ*. 2016. № 2. С. 22–29. (Входить до міжнародних наукометричних баз **Scopus**, INSPEC, EBSCO, Google Scholar).

17. Баркалов А. А., Бабаков Р. М. Модификация микропрограммного автомата с операционным автоматом переходов и заменой входных переменных. *Управляющие системы и машины*. 2017. № 6. С. 35–40. (Входить до міжнародних наукометричних баз РІНЦ, Google Scholar).

18. Баркалов А. А., Бабаков Р. М. Операционная реализация функции выходов микропрограммного автомата. *Управляющие системы и машины*. 2017. № 3. С. 57–62. (Входить до міжнародних наукометричних баз РІНЦ, Google Scholar).

19. Баркалов А. А., Бабаков Р. М. Операционное формирование кодов состояний в микропрограммных автоматах. *Кибернетика и системный анализ*. 2011. № 2. С. 21–26. (Входить до міжнародних наукометричних баз **Scopus**, INSPEC, EBSCO, Google Scholar).

20. Баркалов А. А., Бабаков Р. М. Операционный автомат переходов с дополненным множеством операций переходов. *Наукові праці Донецького національного технічного університету. Серія «Інформатика, кібернетика і обчислювальна техніка»*. Донецьк: ДВНЗ «ДонНТУ». 2011. № 14 (188). С. 80–84. (Журнал індексується міжнародною наукометричною базою Google Scholar).

21. Баркалов А. А., Бабаков Р. М. Определение области эффективного применения микропрограммного автомата с операционным автоматом переходов. *Кибернетика и системный анализ*. 2018. № 3. С. 27–37. (Входить до міжнародних наукометричних баз **Scopus**, INSPEC, EBSCO, Google Scholar).

22. Баркалов А. А., Бабаков Р. М. Структурная классификация методов синтеза микропрограммного автомата с операционным автоматом переходов. *Кибернетика и системный анализ*. 2019. № 2. С. 3–9. (Входить до міжнародних наукометричних баз **Scopus**, INSPEC, EBSCO, Google Scholar).

23. Баркалов А. А., Бабаков Р. М. Уменьшение максимального количества существенных входных переменных в микропрограммном автомате с операционным автоматом переходов. *Управляющие системы и машины*. 2018. № 2. С. 42–50. (Входить до міжнародних наукометричних баз РІНЦ, Google Scholar).

які засвідчують апробацію матеріалів дисертації:

24. Бабаков Р. М. Микропрограммный автомат с операционным автоматом переходов. *Тези доповідей Міжнародної наукової конференції «Сучасна інформатика: проблеми, досягнення та перспективи розвитку», присвяченої 60-річчю заснування Інституту кібернетики імені В.М. Глушкова НАН України* (м. Київ, 13–15 грудня 2017 р.) Київ, 2017. С. 4–6.

25. Бабаков Р. М. Синтез микропрограммного автомата с операционным автоматом переходов методом полного перебора. *Матеріали Всеукраїнської науково-практичної конференції з міжнародною участю «Інформатика та системні науки (ІСН-2017)* (м. Полтава, 16–18 березня 2017 р.) Полтава: ПУЕТ, 2017. С. 23–25. (Входить до міжнародної наукометричної бази Google Scholar).

26. Баркалов А. А., Бабаков Р. М. Операционное формирование переходов в управляющих автоматах. *Матеріали доповідей ІІІ Міжнародної науково-практичної конференції молодих учених, аспірантів, студентів «Сучасна інформаційна Україна: інформатика, економіка, філософія»* (м. Донецьк, 14–15 травня 2009 р.) Донецьк, 2009. Т. 1. С. 18–20. (Входить до міжнародної наукометричної бази Google Scholar).

27. Babakov R., Barkalov A., Titarenko L. Research of Efficiency of Microprogram Final-State Machine with Datapath of Transitions. *Proceedings of 14th International Conference «The Experience of Designing and Application of CAD Systems in Microelectronics» (CADSM)* (Україна, Поляна, 21–25 лютого 2017 р). Львів, 2017. С. 203–206. (Входить до міжнародної наукометричної бази **Scopus**).

28. Barkalov A. A., Titarenko L. A., Babakov R. M. Compositional Microprogram Control Unit with Operational Automaton of Transitions. *Proceedings of IEEE East-West Design & Test Symposium (EWDTS'2012)* (Kharkov, Ukraine, September 14–17). Kharkov, 2012. P. 151–154. (Входить до міжнародної наукометричної бази **Scopus**).

29. Barkalov A. A., Titarenko L. A., Babakov R. M. Compositional Microprogram Control Unit with Operational Automaton of Transitions. *12th IFAC Conference on Programmable Devices and Embedded Systems* (Velke Karlovice, Czech Republic, September 25–27), Velke Karlovice, 2013. P. 239–244. (Входить до міжнародних наукометричних баз **Scopus**, Google Scholar).

30. Barkalov A., Babakov R. Structural representation of synthesis methods of finite state machine with datapath of transitions. *Conference Proceedings of 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT'2018)* (м. Київ, 24–27 травня 2018 р). Київ, 2018. С. 242–246. (Входить до міжнародної наукометричної бази **Scopus**).