

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка

до дипломного проекту (роботи)

магістра

(ступінь вищої освіти)

на тему СИСТЕМА ПАРАЛЕЛЬНОГО ФУНКЦІОНУВАННЯ
МІКРОПРОГРАМНИХ ПРИСТРОЇВ КЕРУВАННЯ НА FPGA

Виконав: студент 2 курсу, групи КНТ-612м
спеціальності _____

123 Комп'ютерна інженерія

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Спеціалізовані комп'ютерні системи

ШАДРІН М. А.

(ПРИЗВИЩЕ та ініціали)

Керівник ЗЕЛЕНЬОВА І.Я.

(ПРИЗВИЩЕ та ініціали)

Рецензент КОЗИНА Г.Л.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти магістерський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) «Спеціалізовані комп'ютерні системи»
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Зав. кафедри Кудерметов Р.К.

“ ” 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

ШАДРИНА Михайла Анатолійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проекту (роботи) Система паралельного функціонування мікропрограмних пристроїв керування на FPGA

керівник проекту (роботи) к. т. н., доцент, ЗЕЛЕНЬОВА Ірина Яківна

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом вищого навчального закладу від “24” жовтня 2023 р. № 400

2. Строк подання студентом проекту (роботи) 10 грудня 2023 року

3. Вихідні дані до проекту (роботи) три алгоритми за якими необхідно розробити схеми керуючих пристроїв

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Аналіз предметної області. Аналіз таких сфер як: вбудовані системи, керуючі пристрої, ПЛІС, сучасні мікросхеми, середовища розробки;

2) Синтез пристроїв керування різних структур на основі вихідних алгоритмів;

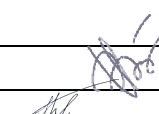
3) Моделювання синтезованих автоматів у середовищі Quartus II;

4) Розробка паралельних схем;

5) Експериментальні дослідження;

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень).

6. Консультанти розділів проекту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	ЗЕЛЕНЬОВА І.Я., доцент		
нормоконтроль	ЩЕРБАК Н.В., ст. викл.		

7. Дата видачі завдання 01.10.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Підготовка вихідних алгоритмів	10.10.2023 р.	
2	Розробка базової структури пристрою керування	15.10.2023 р.	
3	Розробка схеми КМПК на розподілених ресурсах	20.10.2023 р.	
4	Розробка схеми КМПК із вбудованою пам'яттю	05.10.2023 р.	
5	Експериментальні випробування розроблених схем	20.11.2023 р.	
6	Розробка паралельних схем	25.11.2023 р.	
7	Експериментальні випробування паралельних Схем	01.12.2023 р.	

Студент  Михайло ШАДРІН
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник проекту (роботи)  Ірина ЗЕЛЕНЬОВА
(підпис) (ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

ПЗ: 129 с., 88 рис., 37 таблиць, 16 посилань.

FPGA, QUARTUS, КЕРУЮЧИЙ АВТОМАТ, АВТОМАТ МУРА, КОМПОЗИЦІЙНИЙ МІКРОПРОГРАМНИЙ АВТОМАТ, ВБУДОВАНА ПАМ'ЯТЬ, ПАРАЛЕЛЬНИЙ ПРИСКОРЮВАЧ

Метою роботи є підвищення ефективності реалізації мікропрограмного керування в базисі FPGA. Особливої актуальності це набуває стосовно вбудованих систем.

Об'єктом дослідження є засоби апаратного прискорювання алгоритмів керування на ПЛІС.

Предметом дослідження є паралельний запуск декількох керуючих автоматів на одному кристалі FPGA, що може значно зменшувати габарити, вагу та енергоспоживання вбудованого обчислювального пристрою.

У ході роботи розроблені мікропрограмні пристрої керування трьох різних структур для реалізації декількох алгоритмів. Структури відрізняються за способом використання архітектурних властивостей кристалу FPGA. За результатами тестувань різних структур виконано їх порівняння з точки зору продуктивності у часі та ефективного використання ресурсів FPGA, зроблені висновки та визначено найпродуктивнішу структуру.

Наукова новизна роботи полягає в теоретичні пропозиції та експериментальному дослідженні паралельної роботи декількох незалежних пристроїв керування на одному кристалі. Практична цінність результатів роботи полягає в розробці паралельного прискорювача, який поліпшує продуктивність, енергоефективність та зменшує габарити обчислювального пристрою.

ABSTRACT

Explanatory note to the master's work: 129 p., 88 figures, 37 tables, 16 sources.

FPGA, QUARTUS, CONTROL MACHINE, MOORE CONTROL MACHINE, COMPOSITIONAL MICROPROGRAM CONTROL MACHINE, EMBADDED MEMORY, PARALLEL ACCELERATOR

The aim of the work is to increase the efficiency of implementing microprogram control in the FPGA base. This is especially relevant for embedded systems.

The object of research is means of hardware acceleration of control algorithms on FPGAs.

The subject of the research is the parallel launch of several control machines on one FPGA chip, which can significantly reduce the dimensions, weight and power consumption of the embedded computing device.

In the course of the work, firmware control devices of three different structures were developed for the implementation of several algorithms. The structures differ in the way they use the architectural properties of the FPGA crystal. Based on the results of testing various structures, their comparison was made in terms of time performance and effective use of FPGA resources, conclusions were drawn and the most productive structure was determined.

The scientific novelty of the work consists in theoretical proposals and experimental research of the parallel operation of several independent control devices on one crystal. The practical value of the work results lies in the development of a parallel accelerator that improves performance, energy efficiency and reduces the dimensions of the computing device.

ЗМІСТ

Вступ.....	7
1 Аналіз предметної області	9
1.1 Вбудовані системи	9
1.2 Огляд базових структур пристроїв керування.....	13
1.3 Аналіз базису ПЛІС.....	15
1.4 Огляд мікросхем які використовуються у роботі.....	20
1.5 Огляд середовища розробки Quartus II	26
1.6 Постановка задачі розробки	27
2 Розробка моделей та синтез схем композиційних мікропрограмних пристроїв керування.....	29
2.1 Граф-схеми алгоритмів керування	29
2.2 Синтез базової структури за першим алгоритмом	41
2.3 Реалізація проекту базової структури за першим алгоритмом та його тестування в EDA Quartus II.....	43
2.4 Синтез композиційного мікропрограмного пристрою керування за першим алгоритмом	43
2.5 Реалізація проекту композиційного мікропрограмного пристрою керування за першим алгоритмом та його тестування в EDA Quartus II.....	49
2.6 Реалізація проекту композиційного мікропрограмного пристрою керування з вбудованою пам'яттю за першим алгоритмом та його тестування в EDA Quartus II.....	50
3 Розробка паралельних моделей	53
3.1 Компоновка паралельної схеми базових структур першого, другого та третього алгоритму	54
3.2 Компоновка паралельної схеми композиційних автоматів першого, другого та третього алгоритму.....	55

3.3 Компоновка паралельної схеми композиційних автоматів структур з вбудованою пам'яттю першого, другого та третього алгоритму	56
4 Результати експериментів.....	57
4.1 Тестування розроблених схем різних структур які працюють за першим алгоритмом	57
4.2 Тестування розроблених схем різних структур які працюють за другим алгоритмом	66
4.3 Тестування розроблених схем різних структур які працюють за третім алгоритмом	74
4.4 Тестування розроблених паралельних схем різних структур які працюють за першим та другим алгоритмом	83
4.5 Тестування розроблених паралельних схем різних структур які працюють за першим, другим та третім алгоритмом	92
Висновки.....	101
Перелік джерел посилання	103
Додаток А	105
Додаток Б.....	113
Додаток В	121
Додаток Г	129

ВСТУП

Вбудована система визначається сьогодні, як спеціалізована обчислювальна система, яка в силу вирішуваного завдання безпосередньо взаємодіє з фізичними об'єктами та процесами. Сфера вбудованих систем дуже швидко розвивається сьогодні. Вбудовані системи використовуються у повсякденному житті та у багатьох сферах таких як бізнес сектор, сектор оборони, аерокосмічний, аграрний сектор і т.д. Вбудовані системи присутні скрізь. Цифрові годинники, іграшки, телевізори, цифрові телефони, пральна машина, калькулятори, фотоапарати – це все вбудовані системи.

Вбудована система працює на основі мікропроцесора, або мікроконтролера, або FPGA. Сьогодні створення вбудованих систем на базі FPGA набуває актуальності. FPGA це тип інтегральної схеми (IC) або мікросхеми, яка відрізняється тим, що вона може налаштовуватися за допомогою масиву конфігурованих логічних блоків, тобто схема FPGA може перепрограмовуватися, що дає значну гнучкість при налаштуванні складних вбудованих систем.

Пристрій керування ж є дуже важливою частиною вбудованої системи та взагалі більшості цифрових пристроїв. Керуючий автомат може мати різноманітну структуру. Це може бути автомат Мура або автомат Мілі або композиційний мікропрограмний керуючий автомат і т.д. Залежно від структури автомата може зменшуватися або збільшуватись кількість займаних розподілених ресурсів на кристалі, кількість вбудованої пам'яті, часові затримки та розподіленість ресурсів на кристалі. Тобто одна структура може бути продуктивнішою за іншу.

Отже необхідно розробити автомати трьох різних структур для декількох алгоритмів. За результатами тестувань автоматів різних структур необхідно порівняти всі ці автомати з точки зору продуктивності у часі та ресурсах та виділити найпродуктивнішу структуру.

Головною метою роботи є розробка схем для паралельного запуску на одному кристалі декількох автоматів однієї і тієї самої структури, які працюють за різними алгоритмами. Необхідно довести, що паралельний запуск на одному кристалі декількох автоматів можливий.

Практична цінність результатів роботи полягає у тому, що розроблений паралельний прискорювач поліпшує продуктивність, енергоефективність та зменшує габарити обчислювального пристрою.

У першому розділі надаються теоретичні відомості щодо вбудованих систем, принципів роботи FPGA, сучасних FPGA мікросхем, пристроїв керування та обраних засобів розробки.

У другому розділі здійснюється синтез та розробка моделей у пакеті Quartus II керуючих автоматів різних структур різних алгоритмів.

У третьому розділі здійснюється розробка паралельних схем у пакеті Quartus II керуючих автоматів різних структур різних алгоритмів.

У третьому розділі здійснюється експериментальне дослідження розроблених схем окремо, а також їхньої паралельної роботи на кристалі FPGA Altera\Intel. Тестування виконано на різних мікросхемах – дешевого, середнього і кошовного сегментів. Наводяться графіки стосовно основних характеристик продуктивності.

У висновках сформульовані основні результати експериментальних досліджень, а також рекомендації щодо вибору ефективної структури і організації паралельного прискорення алгоритмів керування на FPGA.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Вбудовані системи

Вбудована система — це спеціалізована система, що складається як з апаратного, так і з програмного забезпечення та яка працює на основі мікропроцесора, або мікроконтролера. Її призначення полягає у виконанні певних функцій у ширшій механічній або електричній системі.

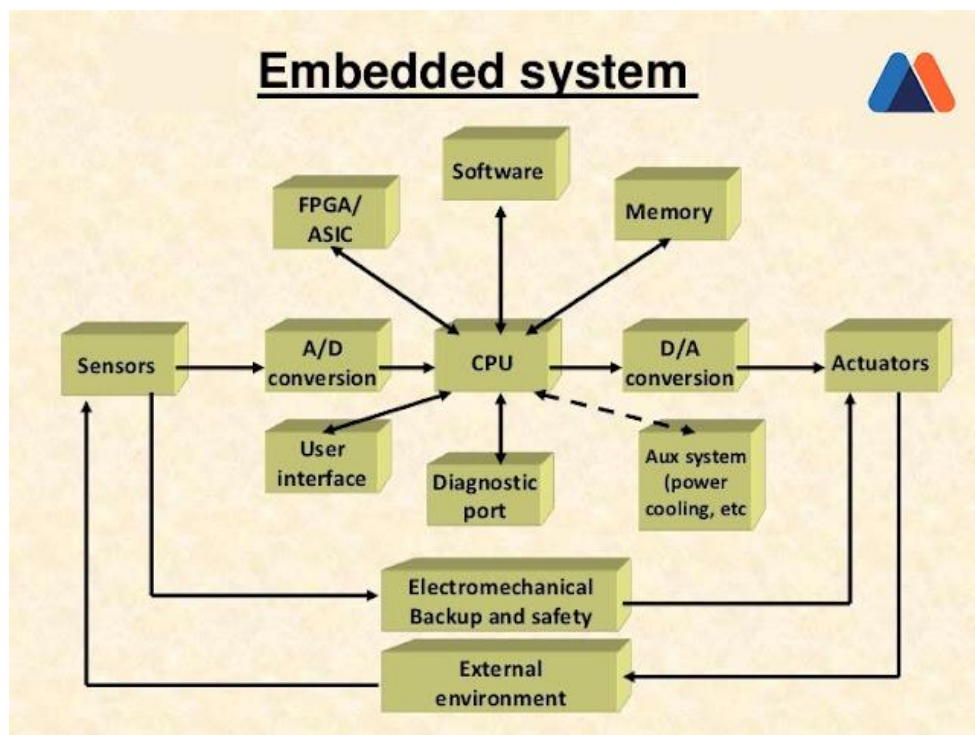


Рисунок 1.1 – Структурна схема типової вбудованої системи [1]

Першою сучасною вбудованою обчислювальною системою в режимі реального часу був навігаційний комп'ютер Apollo, розроблений у 1960-х роках доктором Чарльзом Старком Дрейпером з Массачусетського технологічного інституту для програми Apollo. Навігаційний комп'ютер Apollo був розроблений для автоматичного збору даних і забезпечення критично важливих обчислень для командного модуля Apollo та місячного модуля [1].

У 1971 році Intel випустила перший комерційно доступний мікропроцесор - Intel 4004 - ранній мікропроцесор, який все ще вимагав підтримки мікросхем і зовнішньої пам'яті; у 1978 році Національна асоціація виробників машинобудування випустила стандарт для програмованих мікроконтролерів, покращуючи дизайн вбудованої системи; і на початку 1980-х років пам'ять, компоненти системи введення та виведення були інтегровані в той самий чіп, що й процесор, утворюючи мікроконтролер [1].

Деякі приклади вбудованих систем: цифрові годинники, пральна машина, іграшки, телевізори, цифрові телефони, лазерний принтер, фотоапарати, промислові машини, електронні калькулятори, автомобілі [3].

В основному вбудовані системи присутні скрізь. Ми використовуємо їх в повсякденному житті несвідомо, оскільки в більшості випадків вони інтегровані у більші системи. Отже, ось деякі сфери застосування вбудованих систем: побутова техніка, транспорт, охорона здоров'я, бізнес сектор та офіси, сектор оборони, аерокосмічна, аграрний сектор [3].

1.1.1 Базова структура вбудованої системи

Наступна ілюстрація показує основну структуру вбудованої системи.

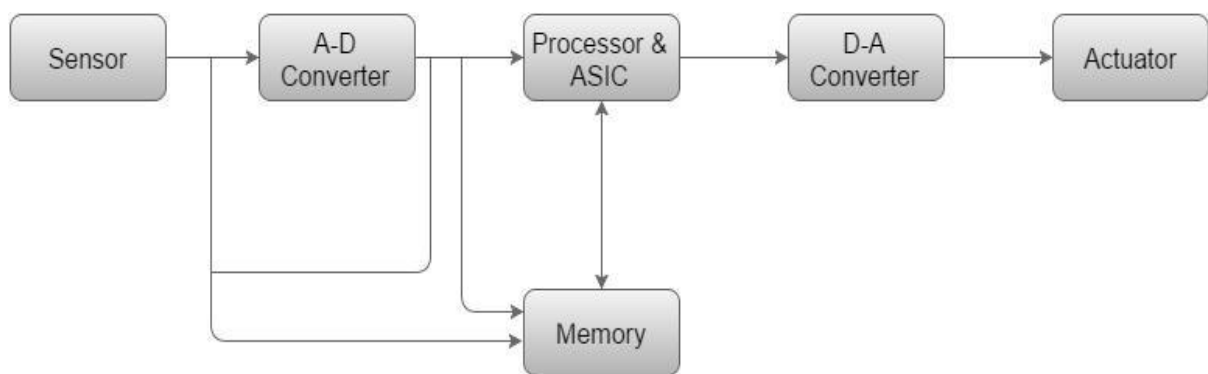


Рисунок 1.2 – Спрощена структура вбудованої системи [2]

Датчик –вимірює фізичну величину та перетворює її на електричний сигнал, який може бути зчитаний спостерігачем або будь-яким електронним інструментом, таким як перетворювач A2D. Датчик зберігає виміряну величину в пам'яті [2].

Аналого-цифровий перетворювач (АЦП) — це пристрій, який перетворює безперервний аналоговий сигнал, отриманий від датчика, на дискретний цифровий сигнал.

Після аналого-цифрового перетворення процесори та спеціальні інтегральні схеми (ASIC) аналізують і обробляють цифрові дані. Вони виконують обчислення для вимірювання бажаного результату, а потім зберігають цю інформацію в пам'яті для подальшого використання.

Цифро-аналоговий перетворювач (DAC) використовується для перетворення цифрових даних, створених процесором, назад в аналоговий формат. Потім цей аналоговий сигнал можна використовувати для різних програм або передати зовнішнім системам.

Актуатор – виконавчий механізм порівнює вихідні дані цифрово-аналогового перетворювача з фактичними (очікуваними) вихідними сигналами, які зберігаються в ньому, і зберігає затверджені вихідні дані [2].

1.1.2 Особливості вбудованої системи

Серед головних особливостей вбудованої системи можна виділити:

- однофункціональність. Вбудована система зазвичай виконує спеціальну операцію та повторює її неодноразово. Наприклад: пейджер завжди працює як пейджер [2];

- жорсткі обмеження. Усі обчислювальні системи мають обмеження щодо проектних показників, але ті, що стосуються вбудованої системи, можуть бути особливо жорсткими. Показники дизайну – це вимірювання характеристик впровадження, таких як його вартість, розмір, потужність і продуктивність. Він має бути такого розміру, щоб поміститися на один чіп, повинен працювати достатньо швидко, щоб обробляти дані в режимі реального часу, і споживати мінімальну енергію, щоб подовжити термін служби батареї;

- реактивний і реальний час. Багато вбудованих систем повинні постійно реагувати на зміни в системному середовищі та повинні обчислювати певні результати в реальному часі без будь-яких затримок. Розглянемо приклад автомобільного круїз-контролера; він постійно контролює та реагує на датчики

швидкості та гальма. Він повинен багаторазово обчислювати прискорення або де прискорення протягом обмеженого часу; запізніле обчислення може призвести до відмови керування автомобілем [2];

- на основі мікропроцесорів. Повинна бути на основі мікропроцесора або мікроконтролера;

- пам'ять. Повинна мати пам'ять, оскільки її програмне забезпечення зазвичай вбудовано в ПЗУ. Не потребує додаткової пам'яті в комп'ютері;

- підключення. Для підключення пристроїв введення та виводу необхідно підключити периферійні пристрої;

- апаратно-програмні системи. Програмне забезпечення використовується для збільшення можливостей і гнучкості. Апаратне забезпечення використовується для продуктивності та безпеки [2].

Плюси вбудованих систем:

- легко налаштовується: ця технологія пропонує високу гнучкість налаштування, що дозволяє користувачам адаптувати її до своїх конкретних потреб і вимог;

- низьке енергоспоживання: однією з ключових переваг є енергоефективність, що призводить до зниження енергоспоживання. Це може бути вигідно як з екологічних причин, так і з міркувань економії;

- низька вартість: впровадження цієї технології є економічно ефективним, що робить її бюджетним варіантом для різних застосувань;

- підвищена продуктивність: технологія забезпечує високу продуктивність, сприяючи більшій ефективності і продуктивності великих систем.

Мінуси вбудованих систем:

- високі зусилля щодо розробки: недолік передбачає значні інвестиції часу та ресурсів на етапі розробки. Для цього може знадобитися спеціальна команда та ретельне планування для подолання потенційних проблем;

- більший час виходу на ринок: процес розробки технології може призвести до довшого часу виходу на ринок. Ця затримка може вплинути на

своєчасну доступність продукту чи послуги, вимагаючи стратегічних міркувань у управлінні проектом.

1.1.3 Майбутні тенденції у вбудованих системах

Сфера вбудованих систем швидко розвиватиметься завдяки постійним досягненням у сфері штучного інтелекту, доповненої реальності, віртуальної реальності і т.д. Вбудовані системи відіграватимуть центральну роль у таких ключових тенденціях, як підключення до хмари, сітчастих мережах, зниження споживання енергії, програми глибокого навчання і т.д.

Відповідно до звіту за 2018 рік, опублікованого QYResearch, світовий ринок індустрії вбудованих систем оцінювався в 68,9 мільярдів доларів у 2017 році та, як очікується, зросте до 105,7 мільярдів доларів до кінця 2025 року [1].

1.2 Огляд пристроїв керування

Залежно від розроблюваної вбудованої системи обирається той чи інший тип керуючого автомату для керування цією вбудованою системою. У цій роботі створюються пристрої керування двох типів: автомат Мура та композиційний мікропрограмний пристрій керування.

1.2.1 Автомат Мура

Автомати Мура є кінцевими автоматами з вихідним значенням, і його вихід залежить лише від поточного стану. Його можна визначити як:

$$(Q, q_0, \Sigma, O, \delta, \lambda) \quad (1.1)$$

де Q — скінченна множина станів;

q_0 — початковий стан;

Σ — алфавіт введення;

O — вихідний алфавіт;

δ — функція переходу, яка відображає $Q \times \Sigma \rightarrow Q$;

λ є вихідною функцією, яка відображає $Q \rightarrow O$ [5].

Структурну схему автомата Мура можна побачити на рисунку 1.3 нижче.

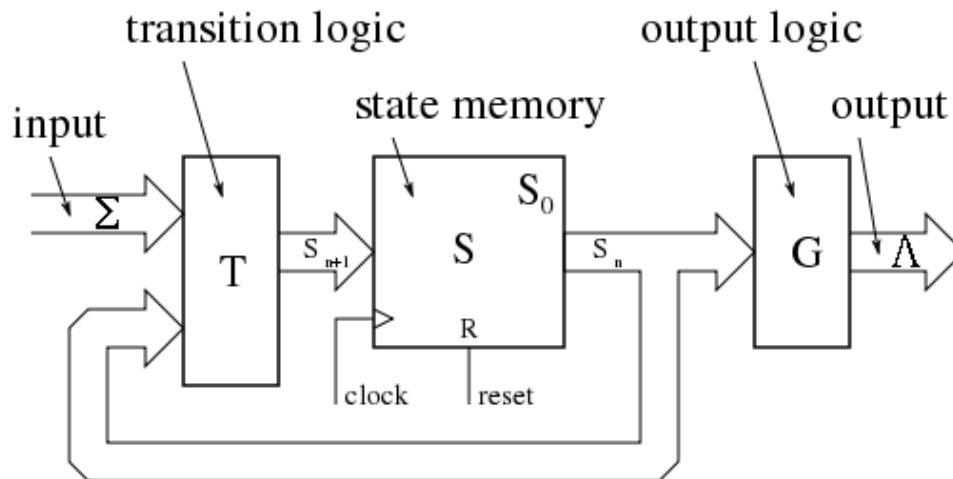


Рисунок 1.3 – Спрощена структура автомату Мура [4]

У першому блоці T на рисунку 1.3 містяться функції для переведення автомату Мура у наступний стан. Автомат має стан за рахунок блоку S. На його виході стан тримається впродовж деякого часу доки цей стан не змінюється із надходженням тактового імпульсу. За тактовим сигналом clock у блок S передається стан переходу сформований у блоці T на основі вхідних сигналів Σ та попереднього стану який тримався на виході S до надходження тактового сигналу. Стан з S передається у блок G який на основі цього стану формує вихідні сигнали схеми

1.2.2 Композиційний мікропрограмний пристрій керування

Композиційний мікропрограмний пристрій керування найкраще підходить для пристроїв алгоритм роботи яких має багато розгалужень тобто логічних умов та містить довгі ланцюги операторних вершин.

Структурна схема композиційного мікропрограмного автомата наведена на рисунку 1.4 нижче.

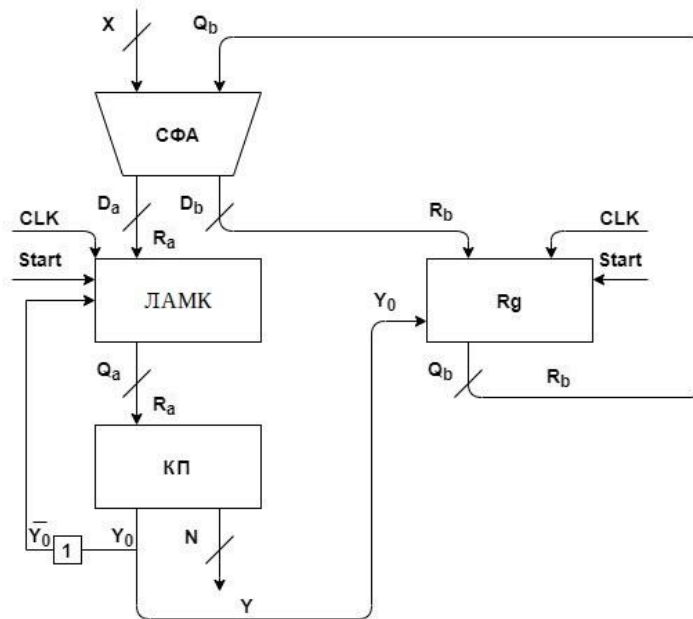


Рисунок 1.4 – Структура КМПК

На рисунку 1.4 наявні такі елементи:

СФА – керуючий автомат;

ЛАМК – лічильник станів вихідних сигналів;

Rg – регістр здвигу пам'яті;

КП – керуюча пам'ять (ROM).

Функції переходу у стани у ланцюгах зберігати у СФА немає необхідності завдяки лічильнику (ЛАМК). Завдяки ньому, щоб перейти у наступний стан у ланцюжку можна просто додати одиницю до попереднього стану автомата. Очевидно, що завдяки цьому можна звільнити багато розподілених ресурсів оскільки функції переходу на новий стан зберігаються саме на них.

1.3 Аналіз базису ПЛІС

Програмована вентиля матриця (FPGA) — це тип інтегральної схеми (IC) або мікросхеми, яка налаштовується за допомогою масиву конфігурованих

логічних блоків (CLB) і може бути переконфігурована після виготовлення [8].

Компоненти ПЛІС:

- CLB - це фундаментальна логічна одиниця FPGA, яка зазвичай містить комбінацію тригерів і таблиць пошуку (LUT);

- тригери забезпечують можливість зберігати та маніпулювати даними в послідовних логічних схемах;

- LUT використовуються для реалізації комбінаційної логіки.

ПЛІС можуть містити додаткові компоненти, такі як:

- інтерконнекти - ресурси маршрутизації, які з'єднують різні логічні блоки;

- блоки введення/виведення (IO) - компоненти, які забезпечують інтерфейс між зовнішнім світом і внутрішньою логікою FPGA;

- блоки обробки цифрових сигналів (блоки DSP) - блоки, які полегшують виконання спеціальних функцій, таких як фільтрація або множення [8].

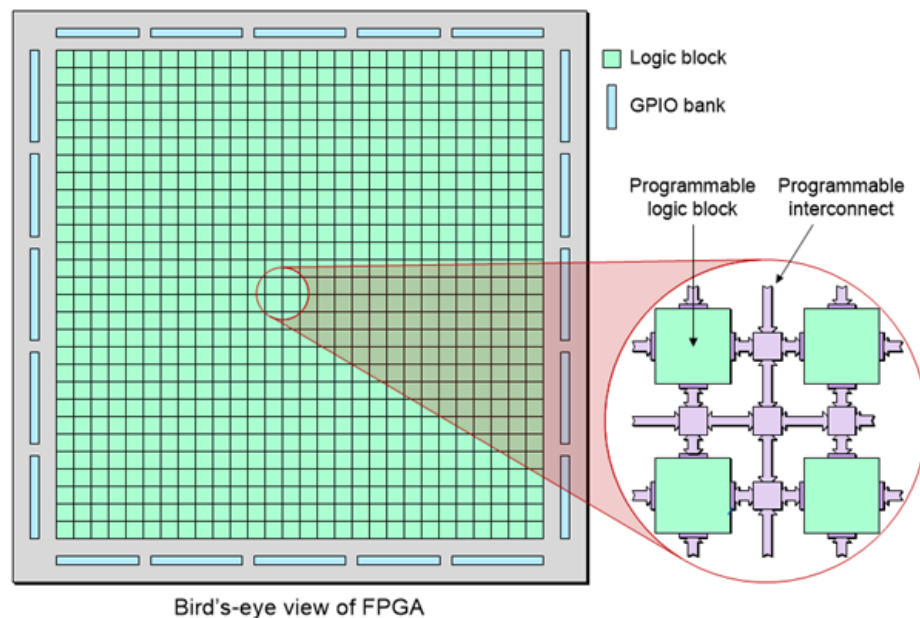


Рисунок 1.5 - Внутрішня структура FPGA [7]

Виробники FPGA можуть впроваджувати власні цифрові схеми та змінювати конфігурацію компонентів відповідно до потреб кінцевого користувача.

До популярних постачальників у сфері FPGA належать Xilinx (придбана AMD у 2022 році), Altera (придбана Intel у 2015 році), Lattice Semiconductor, Microchip (раніше Microsemi) і QuickLogic Corporation [8].

1.3.1 Різниця FPGA vs ASIC vs CPLD

Разом із FPGA, ASIC та CPLD також є важливими компонентами сучасних обчислень, які часто порівнюють у галузі. Знати відмінності між ними та знати, який із них вибрати для певних програм, важливо як для початківців, так і для досвідчених ІТ-фахівців, які приймають рішення щодо обчислень [8].

У наведеній нижче таблиці розбиті переваги та недоліки кожного типу схеми, щоб допомогти вам вирішити, яка найкраще підходить для ваших застосувань [8].

What's the Difference Between FPGA, ASIC and CPLD?			
	Field-Programmable Gate Array	Application-Specific Integrated Circuit	Complex Programmable Logic Device
Acronym	FPGA	ASIC	CPLD
Best Used For	Prototyping, development, and applications that require frequent design changes	High-volume consumer electronics, specialized hardware, mission-critical systems	Glue logic, simple state machines, interfacing tasks, and control applications that don't require high complexity
Customization	Highly customizable and reconfigurable for a wide range of applications	Highly customizable for a specific application	Less flexible and less customizable than FPGAs
Configurability	Reconfigurable after manufacturing	Not reconfigurable after manufacturing; permanent	May offer limited partial reconfiguration
Performance	Good performance, but may not match the performance of ASICs due to the configurability overhead	Highest level of performance due to its tailored design	Can typically be faster than FPGA
Cost	Lower upfront costs compared to ASICs	Requires high upfront costs	Lower upfront costs compared to ASICs
Development Time	Faster development time compared to ASICs	Requires longer development times	Faster development time compared to ASICs
Power Consumption	Higher power consumption due to the configurability overhead	Lowest power consumption because the design is optimized for specific tasks	Typically consumes less power than FPGAs

Рисунок 1.6 – Різниця між FPGA, ASIC та CPLD [8]

FPGA і ASIC є типами інтегральних схем зі схожою схемою, але вони відрізняються за призначенням і функціями. Загалом, ASIC спеціально розроблені для конкретного застосування, тоді як FPGA є більш багатоцільовими і можуть бути переобладнані для багатьох програм. CPLD менш складні, ніж FPGA та ASIC, і часто використовуються для простіших логічних програм [8].

1.3.2 Переваги використання FPGA

FPGA забезпечують багато переваг, включаючи, але не обмежуючись:

- універсальність. ПЛІС дуже універсальні завдяки можливості програмування та реконфігурації, що дозволяє використовувати їх у широкому діапазоні програм;
- швидкість і ефективність. ПЛІС підтримують паралельну обробку, що дозволяє їм запропонувати вищу швидкість, вищу потужність обробки, кращий час відгуку та загальну покращену продуктивність порівняно з іншими типами обладнання;
- вартість. Порівняно з ASIC, FPGA забезпечують нижчі початкові витрати через їх здатність перепрограмувати знову і знову. Це особливо корисно для малих і середніх обсягів виробництва, де початкові витрати на тривалий цикл проектування та розробки ASIC можуть бути невиправданими;
- конфігурація. Основна перевага FPGA полягає в тому, що вони перепрограмовані. Ця якість дозволяє швидко й легко модифікувати FPGA відповідно до потреб користувача порівняно з ASIC, які не можна переконфігурувати після виготовлення;
- інтеграція. Завдяки своїй гнучкій природі FPGA можна легко інтегрувати та швидко оновлювати у більші системи, ніж ASIC, для оновлення яких можуть знадобитися додаткові інженерні витрати [8].

1.3.3 Проблеми, пов'язані з використанням FPGA

Деякі проблеми використання FPGA включають, але не обмежуються:

- вартість. Незважаючи на те, що FPGA вимагають нижчої початкової вартості, ніж ASIC, через короткий час розробки, FPGA можуть бути дорожчими в довгостроковій перспективі для великого виробництва;

- енергоспоживання. Накладні витрати на реконфігурацію FPGA призводять до того, що вони споживають більше енергії, ніж ASIC, які споживають менше електроенергії, оскільки їх конструкція сильно налаштована та оптимізована для конкретних завдань;

- складність. ПЛІС вимагають складного процесу програмування, що включає спеціалізовані знання про дизайн апаратного забезпечення, HDL, інструменти синтезу та розміщення та маршрутизації, особливо для великих і складних проектів;

- обмеження за часом. Дані конфігурації необхідно завантажувати щоразу, коли вмикається FPGA, що може призвести до довшого часу запуску порівняно з ASIC [8].

1.3.5 Мови програмування FPGA і основні виробники мікросхем

Verilog і VHDL (мова опису обладнання VHSIC) є найбільш часто використовуваними мовами програмування FPGA [8].

Verilog є простою, легкою для вивчення текстовою мовою, яка широко визнана як промисловий стандарт. Він був розроблений, щоб нагадувати мову програмування C, частіше використовується в США та характеризується гнучким синтаксисом [8].

VHDL відомий своєю строго типізованою мовою опису обладнання, яка найкраще підходить для складних цифрових систем. Він має синтаксис, що нагадує мову програмування Ada, більш широко використовується в Європі та характеризується суворою логікою та розсудливістю [8].

Інші мови програмування FPGA включають lucid, C і C++, python і фреймворки AI, такі як TensorFlow і PyTorch [8].

1.4 Огляд мікросхем які використовуються у роботі

У цій роботі здійснюються тестування розроблюваних схем на трьох основних сімействах мікросхем Cyclone, Stratix та Arria. У таблиці 1.1 здійснено невеликий огляд головних поколінь цих мікросхем.

Таблиця 1.1 - Таблица поколінь семейств ПЛИС фірми Altera

Сімейство	анонс	ТП (техпроцес), нм	Реком. до застосування у нових розр.	Еквів. лог. вентилів*, тис. шт.
Stratix	2002	130	-	-
Cyclone	2002	130	+	-
Straix GX	2003	130	-	-
Stratix II	2004	90	+	15.6-179.4
Cyclone II	2004	90	+	-
Stratix II GX	2005	90	+	33.88-90.96
Stratix III	2006	65	+	47.5-338
Arria GX	2007	90	+	21.58-90.22
Cyclone III	2007	65	+	-
Stratix IV	2008	40	+	73-813
Arria II GX	2009	40	+	43-244
Cyclone IV	2009	60	+	-
Arria II GZ	2010	40	+	224-349
Stratix V	2010	28	+	236-952
Arria II GZ	2010	40	+	-
Cyclone V	2011	28	+	25-301
Arria V	2011	28	+	-
Arria 10	2013	20	+	160-1150
Stratix 10	2013	14		

* Еквівалентний вентиль (ЕВ) - група елементів БМК, що відповідає можливості реалізації логічної функції вентиля (зазвичай це двовходовий елемент І-НЕ або АБО-НЕ).

Висновки:

- FPGA демонструє значне збільшення внутрішніх ресурсів у різних сімействах;
- спостерігається безперервна еволюція з представленням нового сімейства кристалів майже щорічно, що відображає постійний прогрес у технології;
- у виробництві мікросхем спостерігається постійне зменшення технологічних стандартів;
- сімейства кристалів поділяються на три категорії: бюджетні кристали Cyclone, Arria середнього класу та дорогі та високоефективні кристали Stratix;
- примітно, що сімейство Stratix лідирувало у впровадженні нових технічних стандартів, за ними йшли Cyclone та Arria;
- важливо те, що більшість кристалів підтримуються та рекомендуються для включення в нові проекти, що підтверджує їх постійну актуальність і доступність.

1.4.1 Новітні мікросхеми сімейства Stratix

Altera Stratix 10 – найпродуктивніші FPGA.

Щороку спостерігається постійне зростання обсягу переданих даних. Для задоволення потреб відносно цього докладаються спільні зусилля для будівництва нових центрів обробки даних, дослідження методів збільшення пропускної здатності каналу зв'язку та оптимізації швидкості обробки інформації. Водночас реалізуються ініціативи щодо вирішення невідкладної проблеми пом'якшення зростання споживання енергії, пов'язаного з передачею та обробкою даних. Оскільки цифровий ландшафт розвивається, пошук ефективних і стійких рішень залишається ключовим аспектом цієї динамічної сфери.

У Stratix 10 реалізовані:

- технологія 14-нм Tri-Gate виготовлення чіпа від Intel, яка дозволила знизити енергоспоживання на 70% порівняно із Stratix V;
- збільшена у 10 разів у порівнянні з попереднім поколінням продуктивність ядра, що включає до 5,5 мільйонів логічних елементів (LEs);
- вбудований 64-бітовий процесор ARM Cortex-A53;
- підтримка технології Hybrid Memory Cube;
- блоки обробки сигналів (DSP), сумісні з IEEE 754 і працюють зі швидкістю понад 9 ТФлопс (80 ГФлопс/Вт);
- нова архітектура HyperFlex, що забезпечує дворазовий приріст продуктивності;
- підтримка програм, створених за допомогою фреймворку OpenCL;
- безліч інших технічних рішень, що в сумі дають високопродуктивну архітектуру зі зниженим енергоспоживанням.

Виробник рекомендує застосування FPGA у таких сферах:

- бездротовий зв'язок – для реалізації пристроїв перспективних мереж 5G;
- пристрої обробки даних ЦОД, серверних кластерів і хмарних сховищ;
- маршрутизатори 1 Тб, 400 Гб, 100 Гб-мереж для обробки та керування трафіком;
- пристрої потокової обробки аудіо та відео;
- складні медичні пристрої – сканери, томографи;
- апаратура військового призначення – зв'язки, пеленгації та радіоелектронної боротьби;
- інтелектуальні контрольно-вимірювальні прилади та лабораторне обладнання.

Найбільш продуктивні мікросхеми, що входять до сімейства Stratix 10 – сімейство GX/SX, розглянемо його детальніше. Призначені для високопродуктивних пристроїв, забезпечуючи пропускну здатність до 10 ТФлопс, підтримку обчислень з плаваючою точкою та роботу приймачів зі швидкістю до 28,3 Гбіт.

Таблиця 1.2 - Таблиця характеристик мікросхем ПЛІС сімейства Stratix 10

10 Product Line	GX 160	GX 220	GX 270	GX 320	GX 480	GX 570	GX 660	GX 900	GX 1150	GT 900	GT 1150
Part # Reference	10AX016	10AX022	10AX027	10AX032	10AX048	10AX057	10AX066	10AX090	10AX115	10AT090	10AT115
Equivalent Les	160K	220K	270K	320K	480K	570K	660K	900K	1150K	900K	1150K
Adaptive logic modules (ALMs)	61,510	83,730	101,620	118,730	181,790	217,080	250,540	339,620	427,200	339,620	427,200
Registers	246,040	334,920	406,480	474,920	727,160	868,320	1,002,160	1,358,480	1,708,800	1,358,480	1,080,800
Maximum transceiver count	12	12	24	24	36	48	48	96	96	96	96
M20K memory blocks	440	588	750	891	1,438	1,800	2,133	2,423	2,713	2,423	2,713
M20K memory (Mb)	9	11	15	17	28	35	42	47	53	47	53
MLAB memory (Mb)	1.0	1.8	2.4	2.8	4.3	5.0	5.7	9.2	12.7	9.2	12.7
Variable-precision digital signal processing (DSP) blocks	156	191	830	985	1,368	1,523	1,688	1,518	1,518	1,518	1,518
18x19 multipliers	312	382	1,660	1,970	2,736	3,046	3,376	3,036	3,036	3,036	3,036
Peak GMACS	343	420	1,826	2,167	3,010	3,351	3,714	3,340	3,340	3,340	3,340
Single-precision floating-point multipliers	156	191	830	985	1,368	1,523	1,688	1,518	1,518	1,518	1,518
Single-precision floating-point adders	156	191	830	985	1,368	1,523	1,688	1,518	1,518	1,518	1,518
Peak giga floating-point operation per second (GFLOPs)	140	172	747	887	1,231	1,371	1,519	1,366	1,366	1,366	1,366
PCI Express® (PCIe®) hardened intellectual property (IP) blocks	1	1	2	2	2	2	2	4	4	4	4

Висновки:

- ПЛІС мають вбудовані помножувачі 18×19 від 312 штук у молодшому кристалі, до дуже значної кількості 3036 у старшому кристалі сімейства.

Вбудовані помножувачі можуть утворювати DSP-блоки, що дозволяють проводити обчислення з точкою, що плаває;

- ПЛІС мають вбудовані блоки оперативної пам'яті від 440 до 2713;
- ПЛІС мають високошвидкісні трансівери від 12 до 96.

1.4.2 Новітні мікросхеми сімейства Arria

Arria V Submarine виділяється в галузі своїм 28-нм NVIS, який має найвигіднішу еквівалентну вартість вентелів і демонструє найнижче енергоспоживання серед мікросхем середнього класу. Це сімейство програмованих логічних схем (PLC) представляє передові рішення, включаючи інноваційний апаратний процесор на основі двоядерного процесора ARM Cortex A9. Крім цього, сімейство Arria V включає високошвидкісні трансівери, апаратні блоки IP, такі як контролер PCI Express і контролер зовнішньої синхронної пам'яті, блоки обробки цифрових сигналів із змінною точністю, блоки вбудованої пам'яті M10K і блоки синхронізації.

Сімейство ж FPGA Arria® 10, створене відповідно до технологічних стандартів 20 нм, досягає максимальної продуктивності, перевершуючи конкурентів у своєму класі. Ці нові ПЛІС забезпечують дивовижне зниження енергоспоживання на 40% у порівнянні з попереднім поколінням, утверджуючи себе як лідера галузі з енергоефективності. Примітно, що сімейство Arria 10 виділяється, як єдина програмована логічна схема з апаратно реалізованими блоками цифрової обробки сигналів з плаваючою комою (DSP), що досягає вражаючого рівня продуктивності 1500 GFLOPS (мільйон операцій з плаваючою комою в секунду). Ці досягнення зміцнюють позиції цієї мікросхеми як революційного гравця в ПЛІС технології.

Сімейство ж FPGA Arria® 10:

- на один клас вищий за швидкодією, ніж конкуруючі рішення;
- пропускну здатність інтегрованих приймачів до 28.3 Гбіт/с;
- високопродуктивний інтерфейс зовнішньої пам'яті з пропускну здатністю 2666 Мбіт/с;

- апаратні блоки DSP-обчислень з плаваючою точкою, що відповідають вимогам специфікації IEEE 754;
- до 96 каналів приймачів забезпечують тракт послідовної передачі даних з пропускнуою здатністю до 3.6 Тбіт/с.

1.4.3 Новітні мікросхеми сімейства Cyclone

Altera, провідний виробник програмованих логічних інтегральних схем, пропонує ряд моделей FPGA, і моделі Altera Cyclone виділяються оптимальним поєднанням функціональності та доступності. Ці мікросхеми знаходять застосування в різноманітних конструкціях, де ефективне споживання енергії має вирішальне значення в поєднанні з економічною ефективністю проекту.

Остання серія Altera може похвалитися чудовим рівнем інтеграції, забезпечуючи високу продуктивність при економічній експлуатації. Включення ретельного додаткового програмного забезпечення (прошивки) ще більше спрощує обслуговування та встановлення.

У сімействі Altera Cyclone, яке охоплює від першого до п'ятого покоління, більш сучасні лінії часто використовуються в передових технологічних процесах та інженерних починаннях. Серед них варто відзначити серії Altera Cyclone III і Cyclone III LS. Ці мікросхеми досягають оптимального балансу між функціональністю та ціною, що робить їх ідеальними для таких застосувань, як аудіо-відео обладнання та спеціальна електроніка. Примітно, що модулі серії LS розроблені для структур резервування, що дозволяє створювати проекти з багаторівневими системами безпеки. Типи корпусів схем включають EQFP, FBGA, UBGA, MBGA та PQFP.

Ще одним важливим гравцем у сімействі Altera Cyclone є серія Cyclone V. Ці логічні модулі мають адаптивну структуру, містять вбудовану оперативну пам'ять, цифрову обробку сигналів і апаратні IP-блоки. Крім того, вони включають заходи захисту проекту від несанкціонованого відтворення. Зі швидкістю передачі в діапазоні від 3,12 до 5 Гбіт/с, пристрої Cyclone V поставляються в корпусах UBGA і FBGA.

Різноманітні серії пристроїв Altera відрізняються ємністю оперативної пам'яті, номерами портів, швидкістю обміну даними та іншими параметрами. Ця універсальність гарантує наявність відповідного варіанту для вирішення майже будь-якого технічного завдання.

1.5 Огляд середовища розробки Quartus II

Quartus II це передовий пакет програмного забезпечення, розроблений корпорацією Altera, служить універсальною IDE, призначеною для проектування та програмування пристроїв FPGA (Field-Programmable Gate Array) — ключового компонента багатьох цифрових схем. Це програмне забезпечення не лише надає комплексне середовище розробки для розробників, але й об'єднує основні інструменти для компіляції, моделювання та синтезу складних цифрових схем.

За допомогою Quartus II можливо проектувати цифрові схеми на різноманітних мовах програмування, включаючи Verilog, VHDL і AHDL, пропонуючи розробникам гнучкість у використанні бажаної методології кодування. Quartus II має широку функціональність, програмне забезпечення містить велику бібліотеку попередньо розроблених цифрових блоків. Ці попередньо налаштовані модулі сприяють спрощеному процесу проектування схем, оскільки їх можна легко інтегрувати у проекти.

Однією з ключових особливостей Quartus II є його здатність виконувати аналіз синхронізації, що допомагає розробникам переконатися, що їхні схеми відповідають вимогам синхронізації та працюють правильно. Програмне забезпечення також містить функцію моделювання, яка дозволяє дизайнерам тестувати свої проекти перед програмуванням їх на фізичному пристрої [6].

Quartus II підтримує широкий спектр пристроїв Altera FPGA, включаючи сімейства Cyclone, Stratix, Arria та MAX. Програмне забезпечення доступне для операційних систем Windows і Linux [6].

Ключові особливості Quartus II:

- Комплексне середовище розробки для проектування та програмування пристроїв FPGA;
- Підтримка різних мов програмування, таких як Verilog, VHDL і AHDL;
- Велика бібліотека попередньо розроблених IP-блоків для прискорення процесу проектування схеми;
- Функція аналізу синхронізації, щоб переконатися, що схема відповідає вимогам синхронізації;
- Функція моделювання для тестування дизайну перед програмуванням на фізичному пристрої;
- Підтримує широкий спектр пристроїв Altera FPGA [6].

Підводячи підсумок, можна сказати, що Quartus II — це потужний програмний пакет, призначений для проектування схем FPGA, який надає розробникам великий набір інструментів для спрощення процесу розробки [6].

1.6 Постановка задачі розробки

Сформулюємо задачі, які необхідно вирішити в ході виконання роботи:

- створити три різні алгоритми за якими будуть працювати керуючі автомати;
- здійснити синтез та створити три схеми у Quartus II базової структури (автомат Мура). Перша схема працює за першим алгоритмом, друга за другим та третя за третім;

- здійснити синтез та створити три схеми у Quartus II композиційної структури. Перша схема працює за першим алгоритмом, друга за другим та третя за третім;

- здійснити синтез та створити три схеми у Quartus II композиційної структури з використанням вбудованої пам'яті. Перша схема працює за першим алгоритмом, друга за другим та третя за третім;

- протестувати створені схеми та впевнитися, що часові діаграми розроблених схем відповідають алгоритмам за якими вони працюють;

- зафіксувати у вигляді графіків апаратні витрати;

- зафіксувати у вигляді графіків часові затримки t_{su} , t_{co} та t_{pd} ;

- зафіксувати топології мікросхем на яких здійснювались тестування;

- створити три паралельні схеми у Quartus II, де на одному кристалі повинні бути розміщені попередньо створені схеми. Перша паралельна схема повинна містити схеми базової структури які працюють за першим та другим алгоритмом, друга повинна містити схеми композиційної структури які працюють за першим та другим алгоритмом та третя повинна містити схеми композиційної структури з використанням вбудованої пам'яті які працюють за першим та другим алгоритмом;

- створити ще три паралельні схеми у Quartus II, де на одному кристалі повинні бути розміщені попередньо створені схеми. Перша паралельна схема повинна містити схеми базової структури які працюють за першим, другим та третім алгоритмом, друга повинна містити схеми композиційної структури які працюють за першим, другим та третім алгоритмом та третя повинна містити схеми композиційної структури з використанням вбудованої пам'яті які працюють за першим, другим та третім алгоритмом;

- протестувати створені паралельні схеми та впевнитися, що часові діаграми розроблених схем відповідають алгоритмам за якими вони працюють;

- зафіксувати у вигляді графіків апаратні витрати;

- зафіксувати у вигляді графіків часові затримки t_{su} , t_{co} та t_{pd} ;

- зафіксувати топології мікросхем на яких здійснювались тестування.

З усіх проведених тестувань зробити висновки, щодо апаратних витрат, часових затримок, топологій та головне щодо можливості створення паралельної схеми для запуску на одному кристалі декількох пристроїв. Також необхідно зробити висновки, щодо швидкості виконання алгоритмів на паралельних схемах порівняно з запуском їх послідовно.

2 СИНТЕЗ ТА РОЗРОБКА МОДЕЛЕЙ КЕРУЮЧИХ АВТОМАТІВ РІЗНИХ СТРУКТУР

2.1 Граф-схеми алгоритмів роботи керуючих автоматів

У ході роботи будуть розглядатися керуючі автомати трьох різних структур: автомата Мура (базова структура), композиційного автомата на розподілених ресурсах та композиційного автомата з використанням вбудованої пам'яті.

Для паралельного запуску на одному кристалі трьох автоматів які працюють за різними алгоритмами необхідно побудувати граф-схеми цих алгоритмів (ГСА) за якими здійснюється синтез автоматів та за якими працюють автомати. Граф-схем алгоритмів загалом три (ГСА1, ГСА2, ГСА3), вони побудовані на основі фрагментів реального алгоритму керування вбудованою системою. Задача експериментальних досліджень даної роботи – протестувати паралельну роботу автоматів саме трьох різних алгоритмів на одному кристалі. За кожним алгоритмом були розроблені три структури автомата, тобто автомат Мура (базова структура), композиційний автомат на розподілених ресурсах та композиційний автомат з використанням вбудованої пам'яті, тобто для першого алгоритму три структури, для другого алгоритму три структури та для третього алгоритму три структури.

У цьому розділі наведений синтез структур тільки для першого алгоритму через занадто великий обсяг таблиць та оскільки синтез автоматів для інших двох алгоритмів повністю аналогічний першому.

Нижче на рисунках 2.1, 2.2, 2.3 наведена перша граф-схема алгоритму (ГСА1).

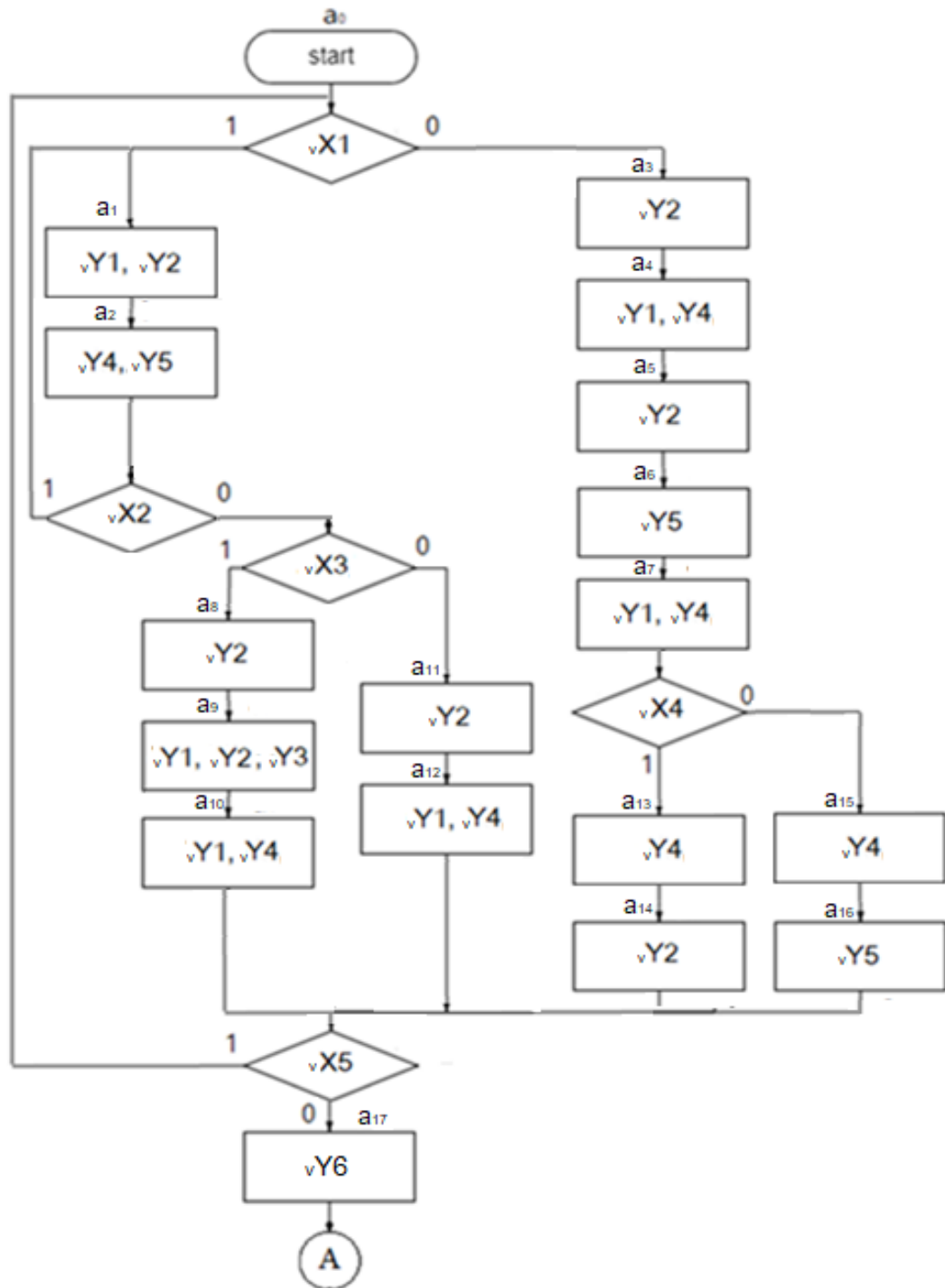


Рисунок 2.1 – Перша граф-схема алгоритму (ГСА1)

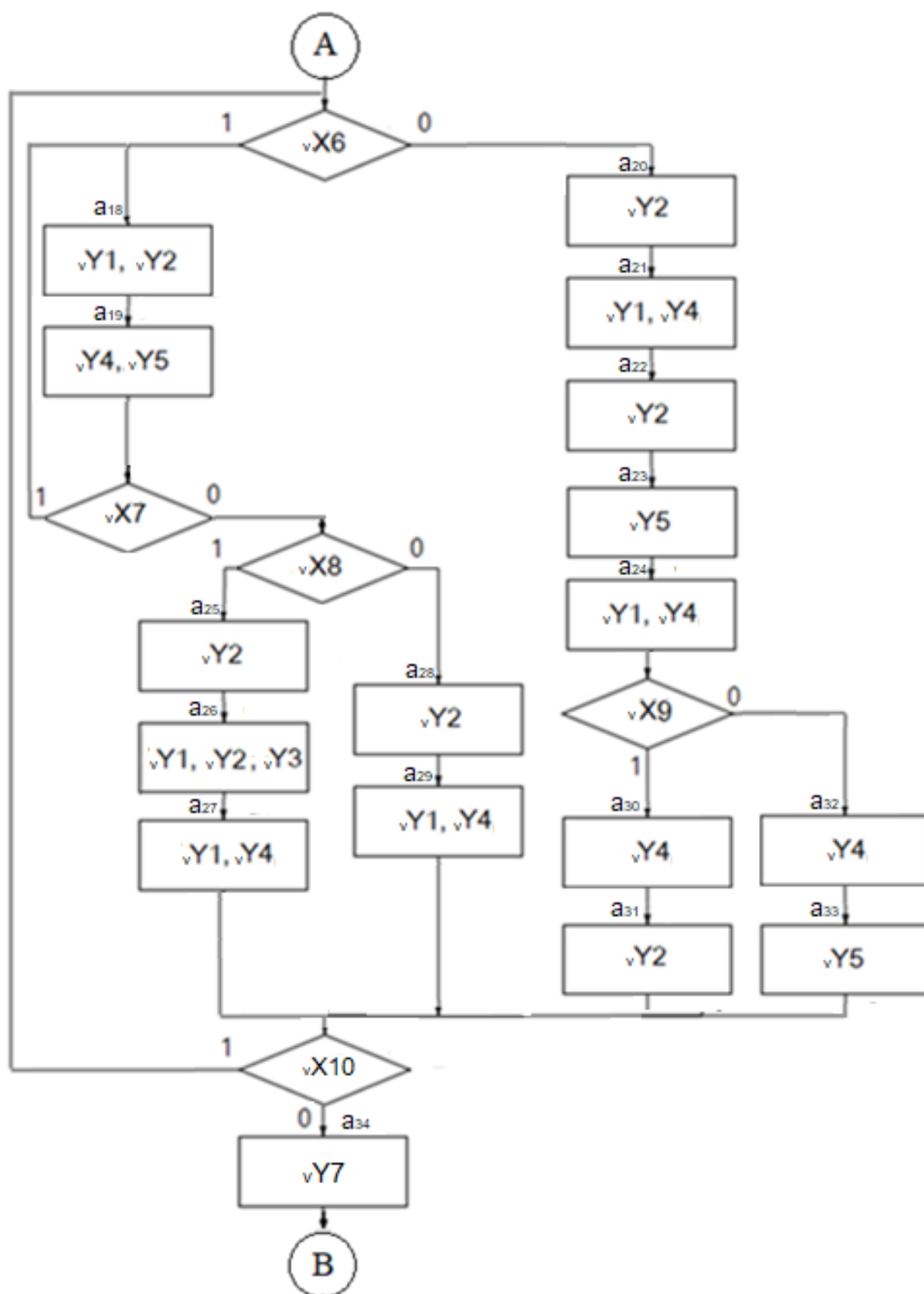


Рисунок 2.2 – Продовження першої граф-схеми алгоритму (ГСА1)

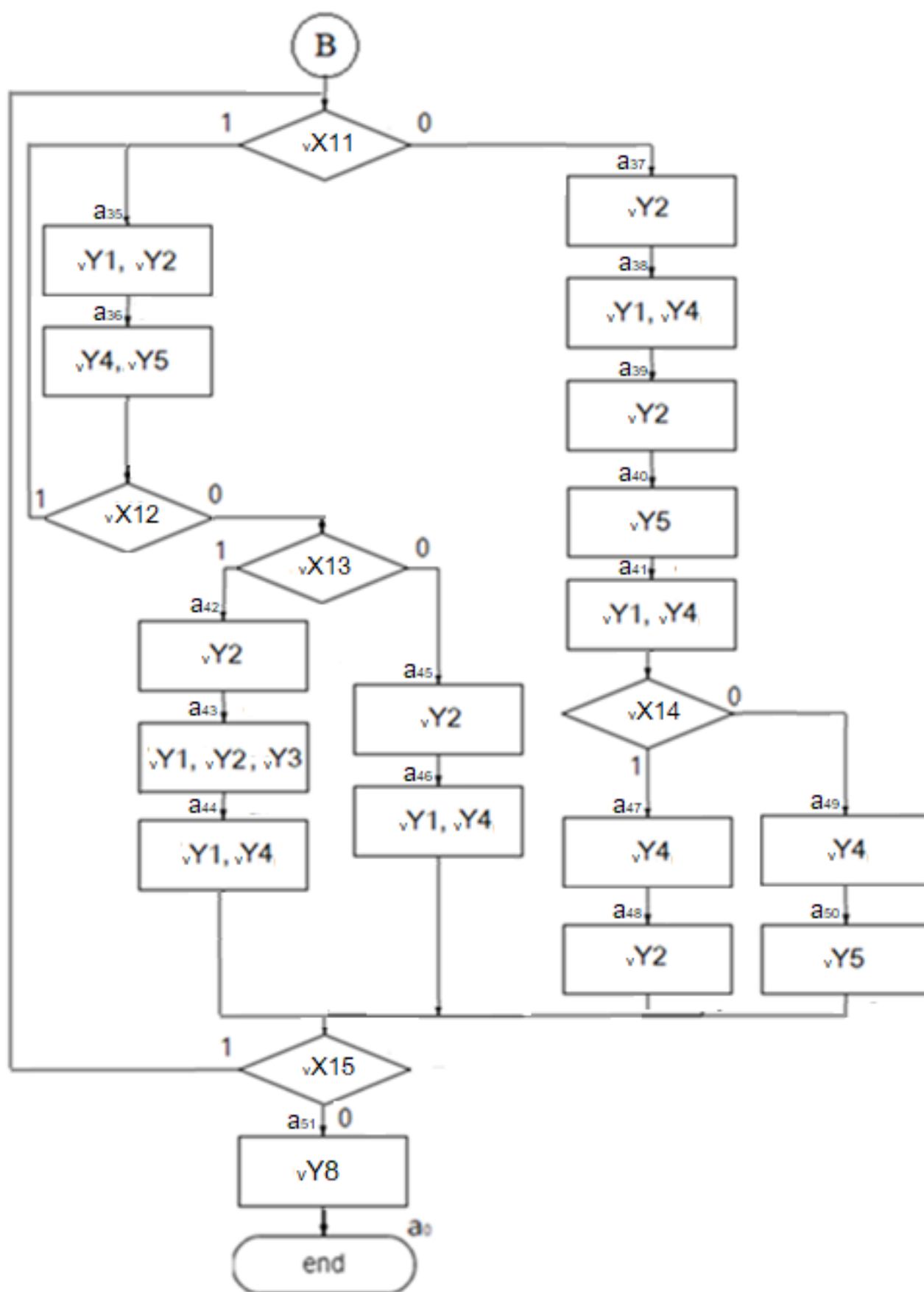


Рисунок 2.3 – Продовження першої граф-схеми алгоритму (ГСА1)

На рисунках 2.4, 2.5, 2.6 наведена друга ГСА2.

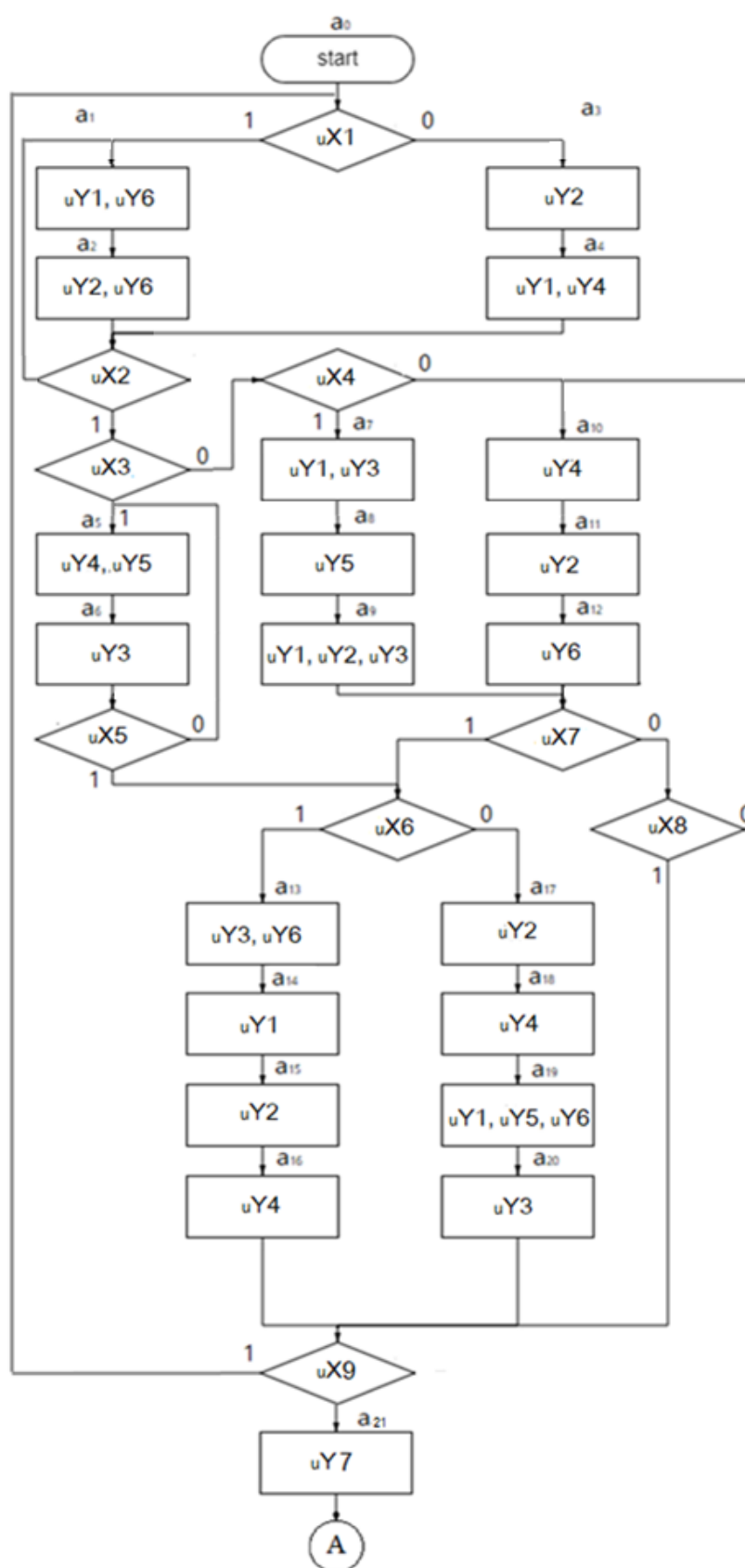


Рисунок 2.4 – Друга граф-схема алгоритму (ГСА2)

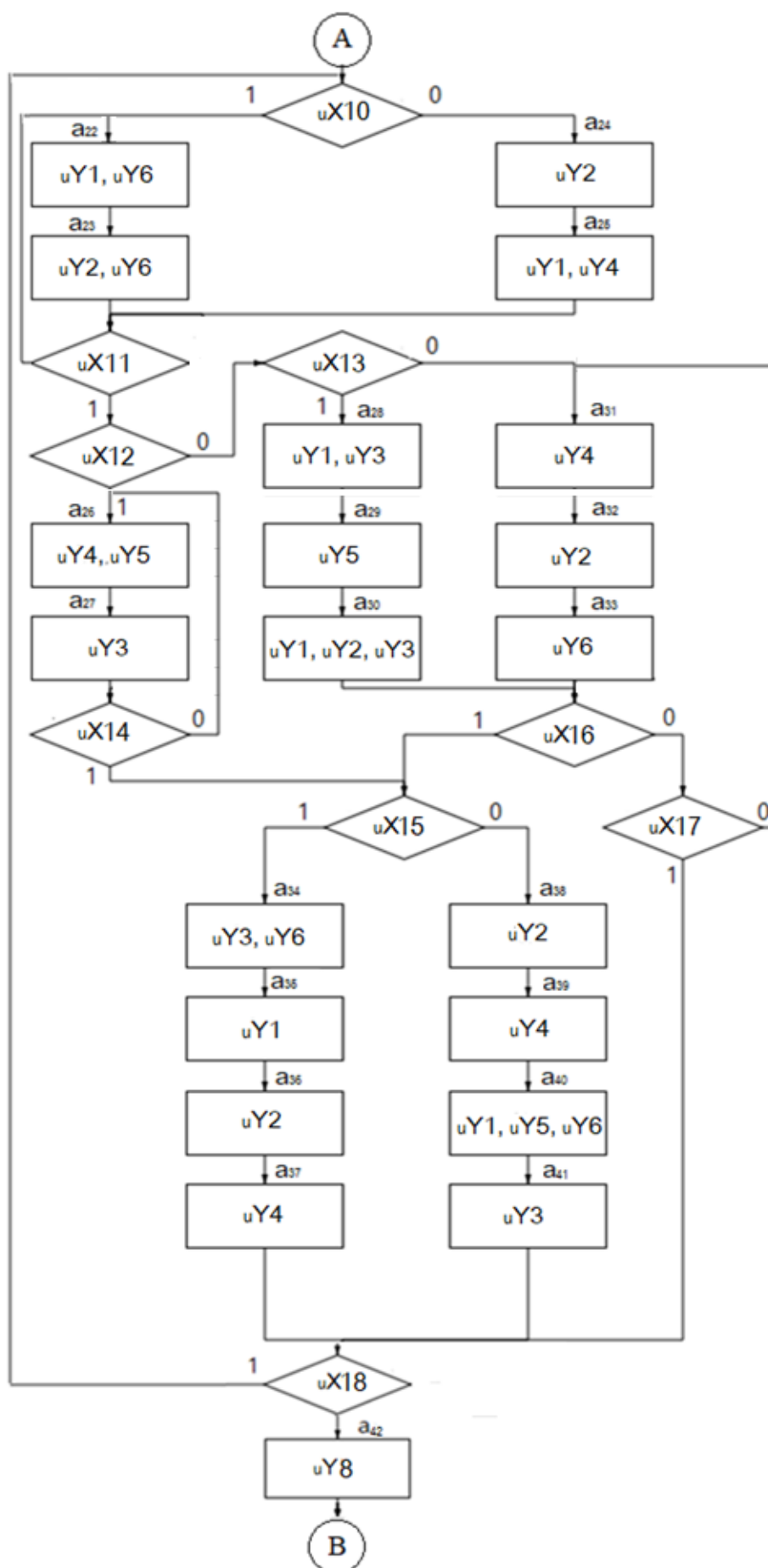


Рисунок 2.5 – Продовження другої граф-схеми алгоритму (ГСА2)

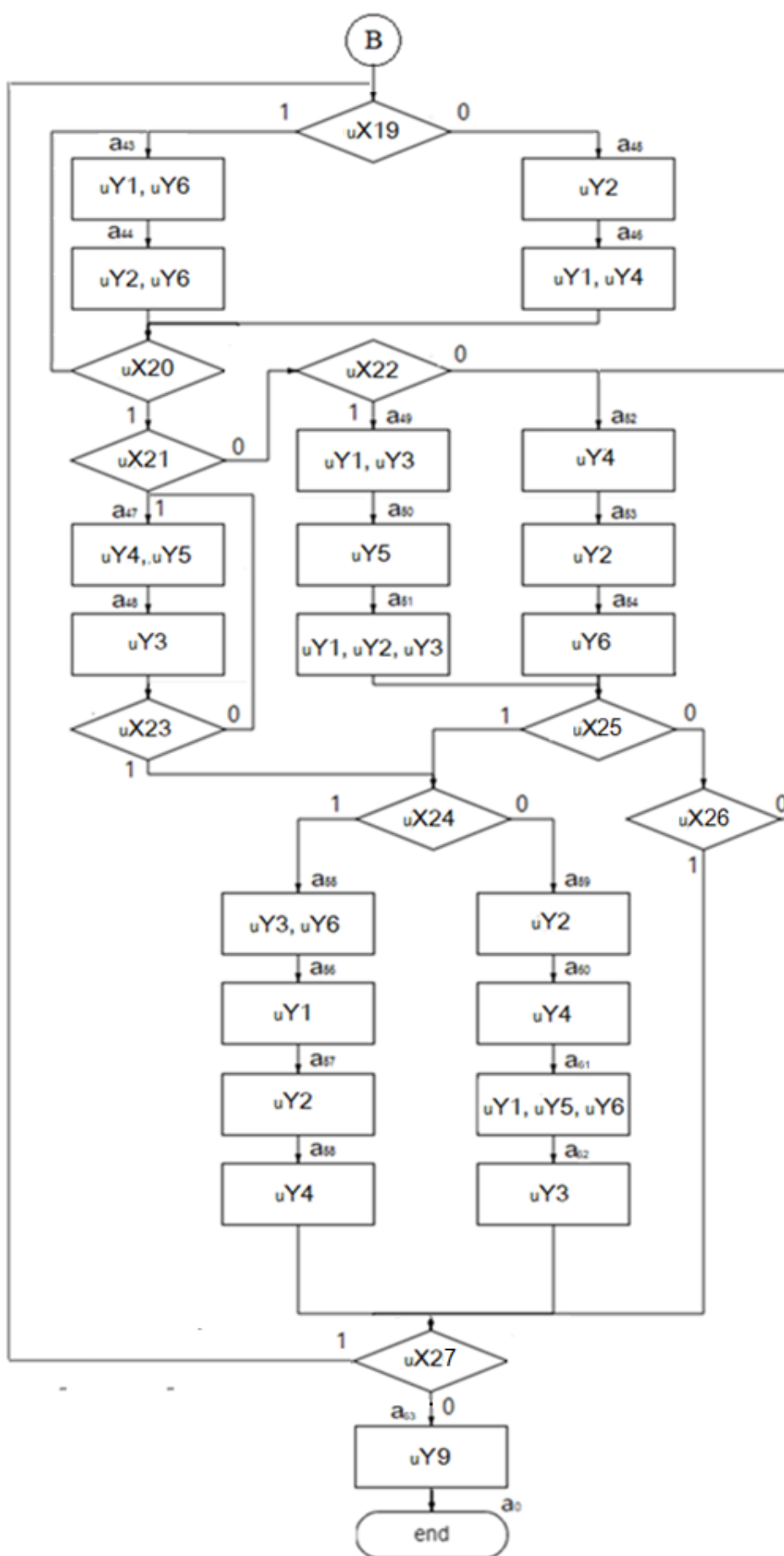


Рисунок 2.6 – Продовження другої граф-схеми алгоритму (ГСА2)

На рисунках 2.7, 2.8, 2.9, 2.10, 2.11 наведена третя ГСАЗ.

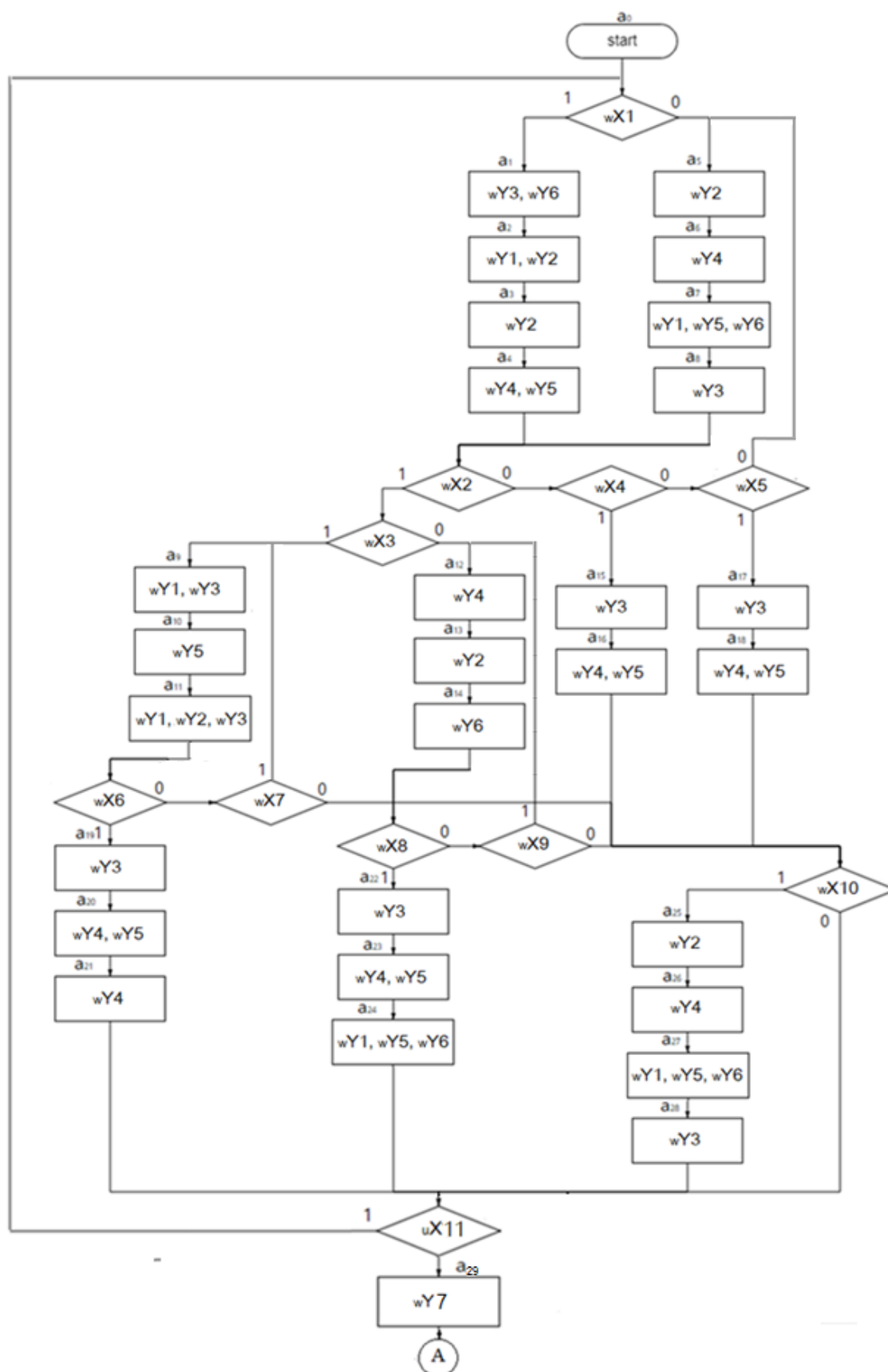


Рисунок 2.7 – Третя граф-схема алгоритму (ГСАЗ)

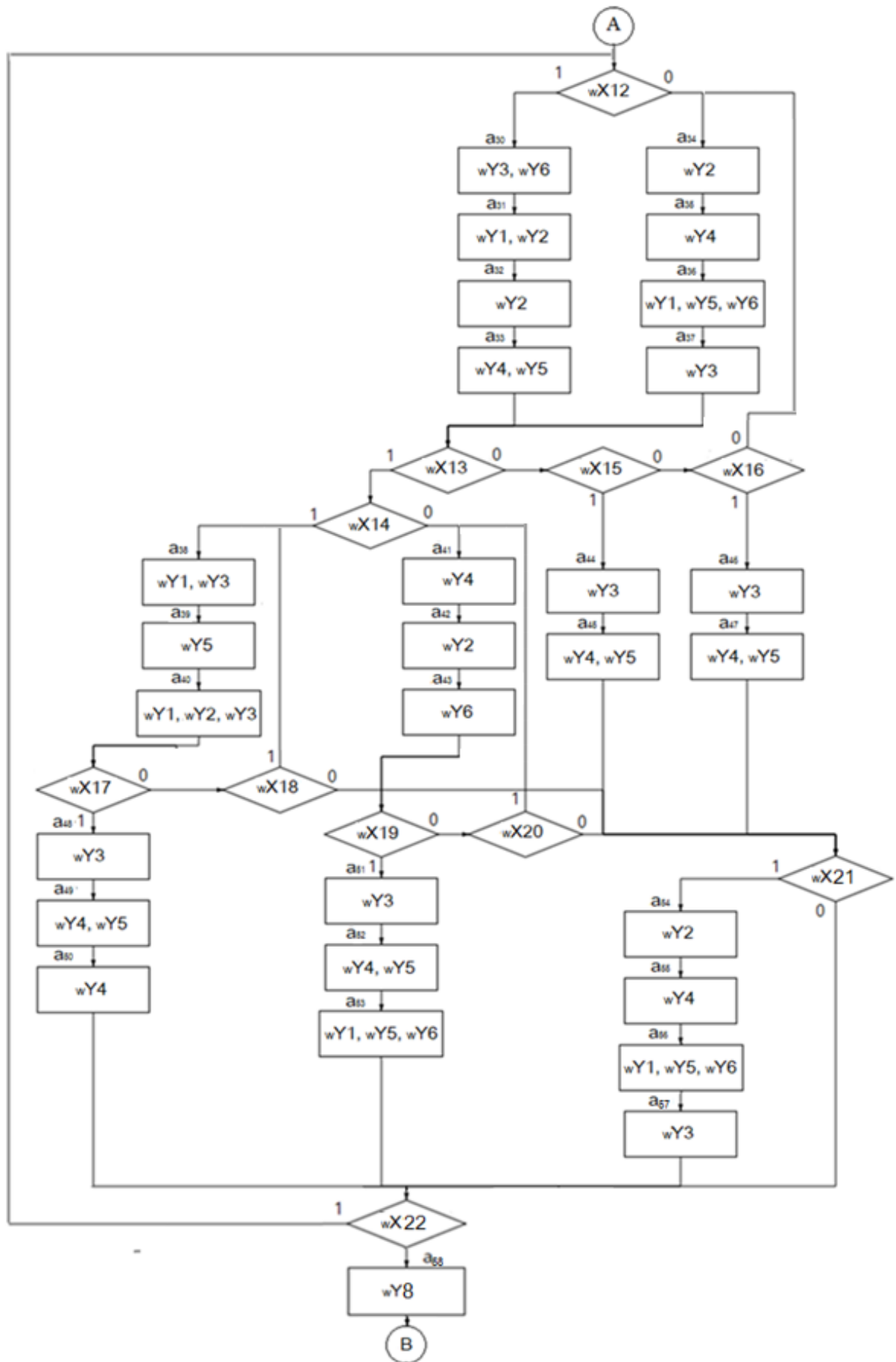


Рисунок 2.8 – Продовження другої граф-схеми алгоритму (ГСА3)

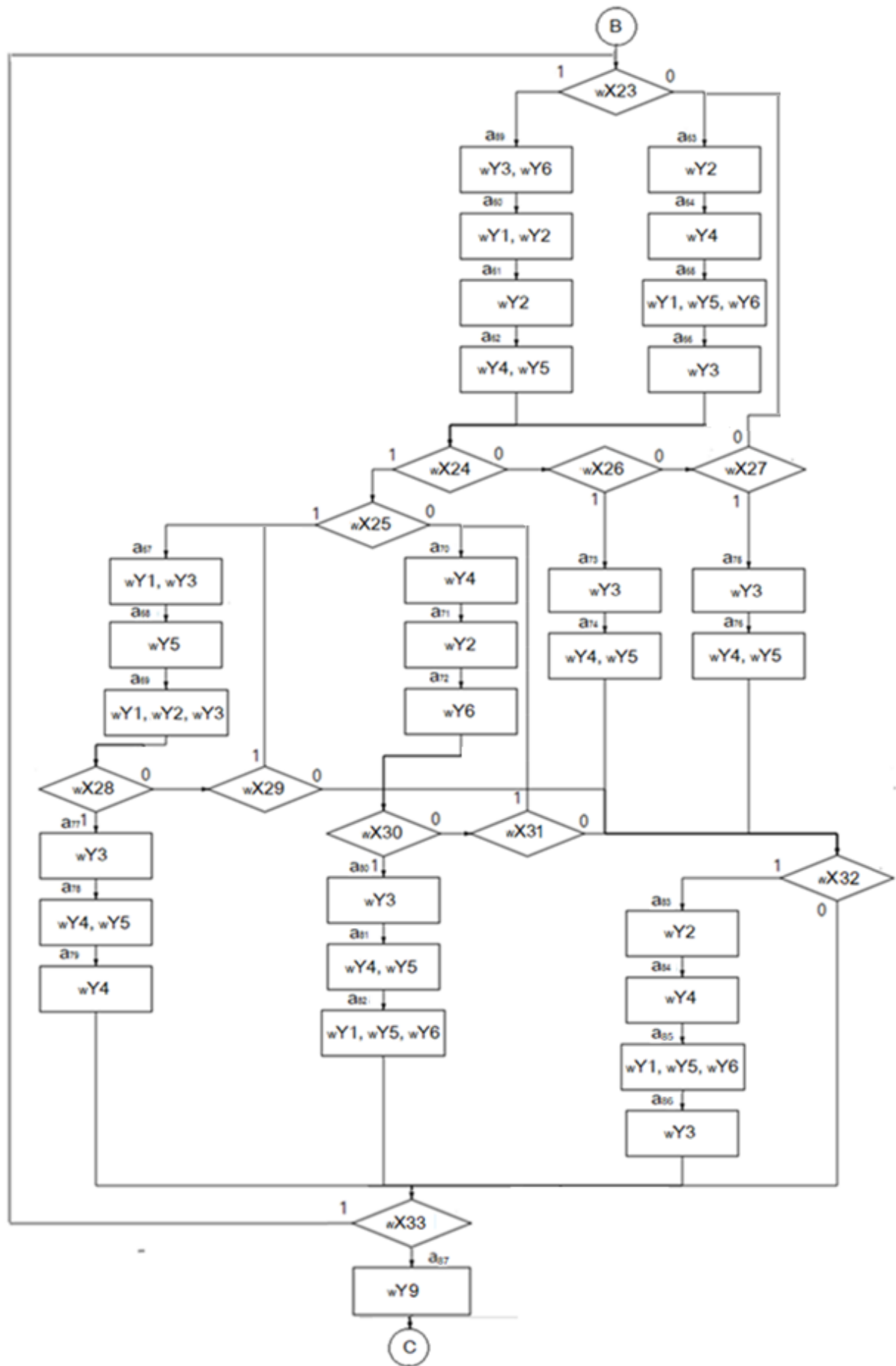


Рисунок 2.9 – Продовження другої граф-схеми алгоритму (ГСА3)

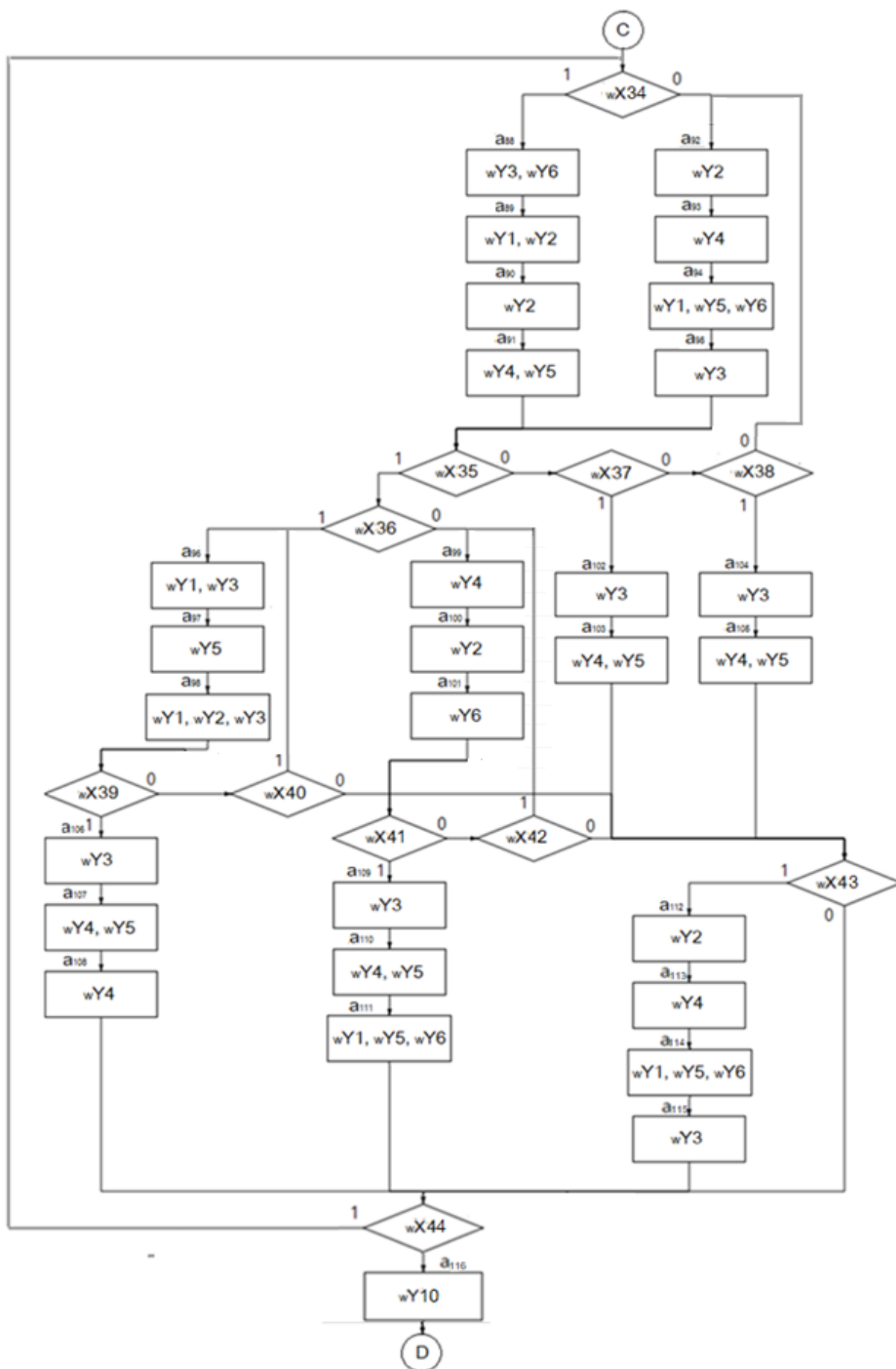


Рисунок 2.10 – Продовження другої граф-схеми алгоритму (ГСА3)

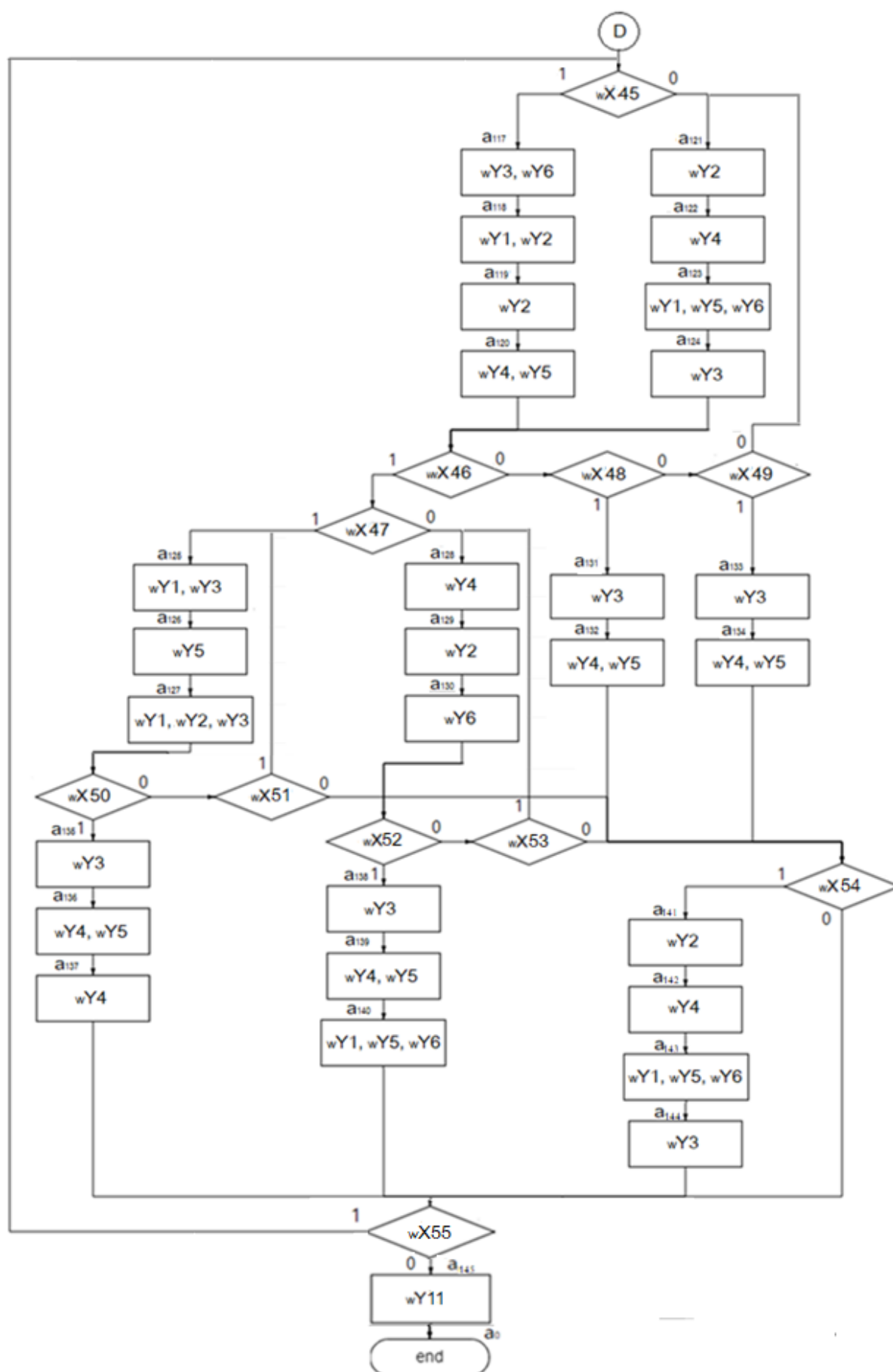


Рисунок 2.11 – Продовження другої граф-схеми алгоритму (ГСА3)

2.2 Синтез базової структури за першим алгоритмом

За першим алгоритмом синтезуємо автомат Мура (базову структуру).

Для автомата Мура на етапі одержання зазначеної ГСА1 розмітка проводиться відповідно до таких правил:

- символом a_0 відзначається початкова та кінцева вершини;
- різні операторні вершини відзначаються різними символами;
- всі операторні вершини мають бути відзначені.

Параметри вихідного алгоритму - множина мікрооперацій:

$$Y = \{vY_1, vY_2, vY_3, vY_4, vY_5, vY_6, vY_7, vY_8\}, N = |Y| = 5$$

$$X = \{uX_1, uX_2, uX_3, uX_4, uX_5, uX_6, uX_7, uX_8, uX_9, uX_{10}, uX_{11}, uX_{12}, uX_{13}, uX_{14}, uX_{15}\}, L = |X| = 15$$

Спочатку, необхідно відмітити стани автомата Мура ($a_0, \dots, a_n$)

$$\text{Множина станів Мура: } A = \{a_0 \dots a_{51}\}, M = |A| = 51$$

Знаходимо кількість елементів пам'яті $Ra = \log_2 M = 6$ (M - число станів абстрактного автомата)

У кожному такті автомат Мура, що інтерпретує мікропрограму, переходить з одного стану в інший і видає відповідні вихідні сигнали керуючі Y_i . Порядок видачі вихідних сигналів Y_i визначається значеннями вхідних сигналів X_i . Так, за наявності вхідного сигналу $X_1 = 0$ автомат із стану a_0 перейде у стан a_1 і видасть вихідний сигнал Y_1 . У наступному такті роботи під впливом вхідного сигналу Y_1 автомат зі стану a_1 перейде у стан a_2 з видачею вихідних сигналів Y_2 та Y_4 .

Наступним кроком сформуємо таблицю функцій виходів для автомата Мура, де:

$$Q_a = \{Q_5, Q_4, Q_3, Q_2, Q_1, Q_0\} - \text{адреса } a_m$$

$$vY_a = \{vY_8, vY_7, vY_6, vY_5, vY_4, vY_3, vY_2, vY_1\} - \text{мікрооперації } vY(a_m),$$

$$N=10$$

Таблиця 2.1 – Адресація станів ЛПС

A_m	Q_5	Q_4	Q_3	Q_2	Q_1	Q_0	vY_1	vY_2	vY_3	vY_4	vY_5	uY_6	uY_7	uY_8
a_0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a_1	0	0	0	0	0	1	1	1	0	0	0	0	0	0
a_2	0	0	0	0	1	0	0	0	0	1	1	0	0	0
a_3	0	0	0	0	1	1	0	1	0	0	0	0	0	0
a_4	0	0	0	1	0	0	1	0	0	1	0	0	0	0
...	...						...							

Повна таблиця наведена у додатку А.

Далі визначимо таблицю переходів і збудження. Функція збудження залежить від вхідних сигналів та стану автомата, з якого був переключений цей тригер.

Побудуємо структурну таблицю переходів - виходів автомата Мура.

Таблиця 2.2 - Структурна таблиця переходів – виходів автомата Мура

	Початковий стан						Умови пере- ходу		Подальший стан						Функції збудження					
Міт -ка	Код							Міт -ка	Код											
	Q ₀	Q ₁	Q ₂	Q ₃	Q ₄	Q ₅			Q ₀	Q ₁	Q ₂	Q ₃	Q ₄	Q ₅	D ₀	D ₁	D ₂	D ₃	D ₄	D ₅
a ₀	0	0	0	0	0	0	uX_1	a ₁	0	0	0	0	0	1	0	0	0	0	0	1
	0	0	0	0	0	0	$\overline{uX_1}$	a ₃	0	0	0	0	1	1	0	0	0	0	1	1
a ₁	0	0	0	0	0	1	1	a ₂	0	0	0	0	1	0	0	0	0	0	1	0
a ₂	0	0	0	0	1	0	uX_2	a ₁	0	0	0	0	0	1	0	0	0	0	0	1
	0	0	0	0	1	0	$\overline{uX_2}uX_3$	a ₈	0	0	1	0	0	0	0	0	1	0	0	0
	0	0	0	0	1	0	$\overline{uX_2}uX_3$	a ₁₁	0	0	1	0	1	1	0	0	1	0	1	1
a ₃	0	0	0	0	1	1	1	a ₄	0	0	0	1	0	0	0	0	0	1	0	0
...																				

Повна таблиця наведена у додатку А.

Далі програмуємо автомат Мура в Quartus II мовою VHDL. Лістинг VHDL опису блоку пристрою керування базової структури автомата можна знайти у додатку Б.

2.3 Реалізація проекту базової структури за першим алгоритмом та його тестування в EDA Quartus II

Схема проекту базового пристрою керування наведена на рисунку 2.12.

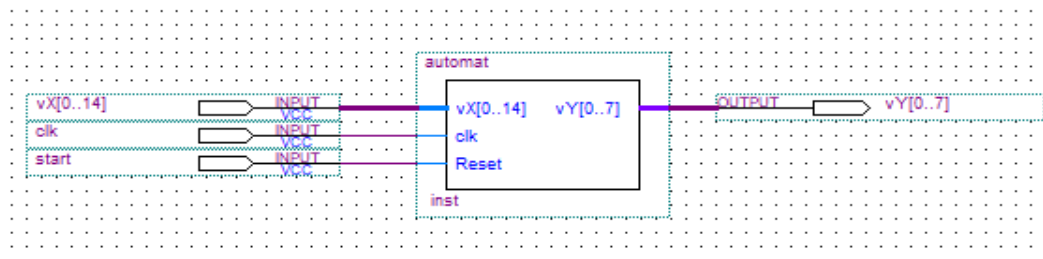


Рисунок 2.12– Схема базового пристрою керування – автомат МУРА

На схемі можна побачити 15 вхідних сигналів логічних операторів та 8 вихідних мікрооперацій, що відповідає ГСА1. Також можна побачити тактовий сигнал та сигнал скидання.

2.4 Синтез композиційного мікропрограмного пристрою керування за першим алгоритмом

Тепер синтезуємо другу структуру тобто композиційний мікропрограмний автомат за першим алгоритмом.

Перший етап синтезу починається з початкових даних, тобто зі граф-схеми автомата Мура (той самий алгоритм що і для базового автомату)

Параметри вихідного алгоритму - множина мікрооперацій:

$$Y = \{vY_1, vY_2, vY_3, vY_4, vY_5, vY_6, vY_7, vY_8\}, N = |Y| = 5$$

$$X = \{uX_1, uX_2, uX_3, uX_4, uX_5, uX_6, uX_7, uX_8, uX_9, uX_{10}, uX_{11}, uX_{12}, uX_{13}, uX_{14}, uX_{15}\}, L = |X| = 15$$

Спочатку, необхідно відмітити стани автомата Мура ($a_0, \dots, a_n$)

Множина станів Мура: $A = \{a_0 \dots a_{51}\}$, $M = |A| = 51$

Знаходимо кількість елементів пам'яті $Ra = \log_2 M = 6$ (M - число станів абстрактного автомата)

$Qa = \{ Q_5, Q_4, Q_3, Q_2, Q_1, Q_0 \}$, $Da = \{ D_5, D_4, D_3, D_2, D_1, D_0 \}$ – функції активації D-тригерів.

Наступним кроком необхідно сформувати лінійні послідовності станів (ЛПС). Додамо мікрооперацію uY_0 "стоп-лічильник" в останню операторну вершину в лінійному ланцюзі.

Формування лінійних послідовностей станів:

$$\alpha_1 = \{a_1, a_2\},$$

$$\alpha_2 = \{a_3, a_4, a_5, a_6, a_7\},$$

$$\alpha_3 = \{a_8, a_9, a_{10}\},$$

$$\alpha_4 = \{a_{11}, a_{12}\},$$

$$\alpha_5 = \{a_{13}, a_{14}\},$$

$$\alpha_6 = \{a_{15}, a_{16}\},$$

$$\alpha_7 = \{a_{17}\},$$

$$\alpha_8 = \{a_{18}, a_{19}\},$$

$$\alpha_9 = \{a_{20}, a_{21}, a_{22}, a_{23}, a_{24}\},$$

$$\alpha_{10} = \{a_{25}, a_{26}, a_{27}\},$$

$$\alpha_{11} = \{a_{28}, a_{29}\},$$

$$\alpha_{12} = \{a_{30}, a_{31}\},$$

$$\alpha_{13} = \{a_{32}, a_{33}\},$$

$$\alpha_{14} = \{a_{34}\},$$

$$\alpha_{15} = \{a_{35}, a_{36}\},$$

$$\alpha_{16} = \{a_{37}, a_{38}, a_{39}, a_{40}, a_{41}\},$$

$$\alpha_{17} = \{a_{42}, a_{43}, a_{44}\},$$

$$\alpha_{18} = \{a_{45}, a_{46}\},$$

$$\alpha_{19} = \{a_{47}, a_{48}\},$$

$$\alpha_{20} = \{a_{49}, a_{50}\},$$

$$\alpha_{21} = \{a_{51}\},$$

Наступний крок, це адресація станів в ЛПС та формування таблиці керуючої пам'яті.

$$Q_a = \{Q_5, Q_4, Q_3, Q_2, Q_1, Q_0\} - \text{адреса } a_m$$

$$vY_a = \{vY_8, vY_7, vY_6, vY_5, vY_4, vY_3, vY_2, vY_1, vY_0\} - \text{мікрооперації } vY(a_m), N=10$$

Таблиця 2.3 – Адресація станів ЛПС

ЛПС α_i	I_{bx}	A_m	Q_5	Q_4	Q_3	Q_2	Q_1	Q_0	vY_0	vY_1	vY_2	vY_3	vY_4	vY_5	uY_6	uY_7	uY_8
α_1	I_1	a_1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
		a_2	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0
α_2	I_2	a_3	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0
		a_4	0	0	0	0	1	1	0	1	0	0	1	0	0	0	0
		a_5	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
		a_6	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0
		a_7	0	0	0	1	1	0	1	1	0	0	1	0	0	0	0
α_3	I_3	a_8	0	0	0	1	1	1	0	0	1	0	0	0	0	0	0
		a_9	0	0	1	0	0	0	0	1	1	1	0	0	0	0	0
		a_{10}	0	0	1	0	0	1	1	1	0	0	1	0	0	0	0
...			...						...	...							

Повна таблиця наведена у додатку А.

Стан a_0 не входить у таблицю керуючого пристрою, оскільки відноситься до автомата з жорсткою логікою, позначається як Start і не містить керуючих слів (мікрооперації uY_i).

Мітки a та α можуть бути довільними, головне – адресувати стани всередині ЛПС у натуральному порядку.

Кодування операцій Y_i виконано унітарно. Такий спосіб кодування забезпечує найвищу швидкодію, оскільки не вимагає додаткових схем на розшифрування.

Адреси входів в ЛПС:

$$I_1 = 000000 (a_1),$$

$$I_2 = 000010 (a_3),$$

$$I_3 = 000111 (a_8),$$

$$I_4 = 001010 (a_{11}),$$

$$I_5 = 001100 (a_{13}),$$

$$I_6 = 001110 (a_{15}),$$

$$I_7 = 010000 (a_{17}),$$

$$I_8 = 010001 (a_{18}),$$

$$I_9 = 010011 (a_{20}),$$

$$I_{10} = 011000 (a_{25}),$$

$$I_{11} = 011011 (a_{28}),$$

$$I_{12} = 011101 (a_{30}),$$

$$I_{13} = 011111 (a_{32}),$$

$$I_{14} = 100001 (a_{34}),$$

$$I_{15} = 100010 (a_{35}),$$

$$I_{16} = 100100 (a_{37}),$$

$$I_{17} = 101001 (a_{42}),$$

$$I_{18} = 101100 (a_{45}),$$

$$I_{19} = 101110 (a_{47}),$$

$$I_{20} = 110000 (a_{49}),$$

$$I_{21} = 110010 (a_{51})$$

Наступний крок – перетворення ГСА1 к компактному вигляду. Кожна операторна вершина перетвореної ГСА1 містить адреси входу відповідної ЛПС, а точніше містить функцію активації тригерів, необхідних для формування адресації входу. Перетворена ГСА1 містить лише умовні переходи.

Для організації паралельного аналізу всіх логічних умов застосуємо модель автомата Мілі, і таким чином збільшимо швидкодію та знизимо кількість циклів алгоритму, оскільки Милі має менше станів.

Для автомата Мілі відмітка граф-схеми проводиться в такому порядку:

- символом b_0 відзначається вихід початкової та вхід кінцевої вершини;

- символами $b_1, b_2, \dots, b_n$ відзначаються входи вершин, що йдуть за операторними вершинами;

- входи двох різних вершин не можуть бути позначені однаковими символами;

- входи вершини можуть бути позначені лише одним символом стану.

Якщо вершина має кілька входів, то станом відзначається їх підмножина, що складається з входів, що йдуть тільки за початковою або за операторними вершинами.

Побудова граф-схеми алгоритму роботи КА Мілі на D-тригерах першого основного боку без резервних блоків (резервні за аналогією) наведена на рисунку 2.13.

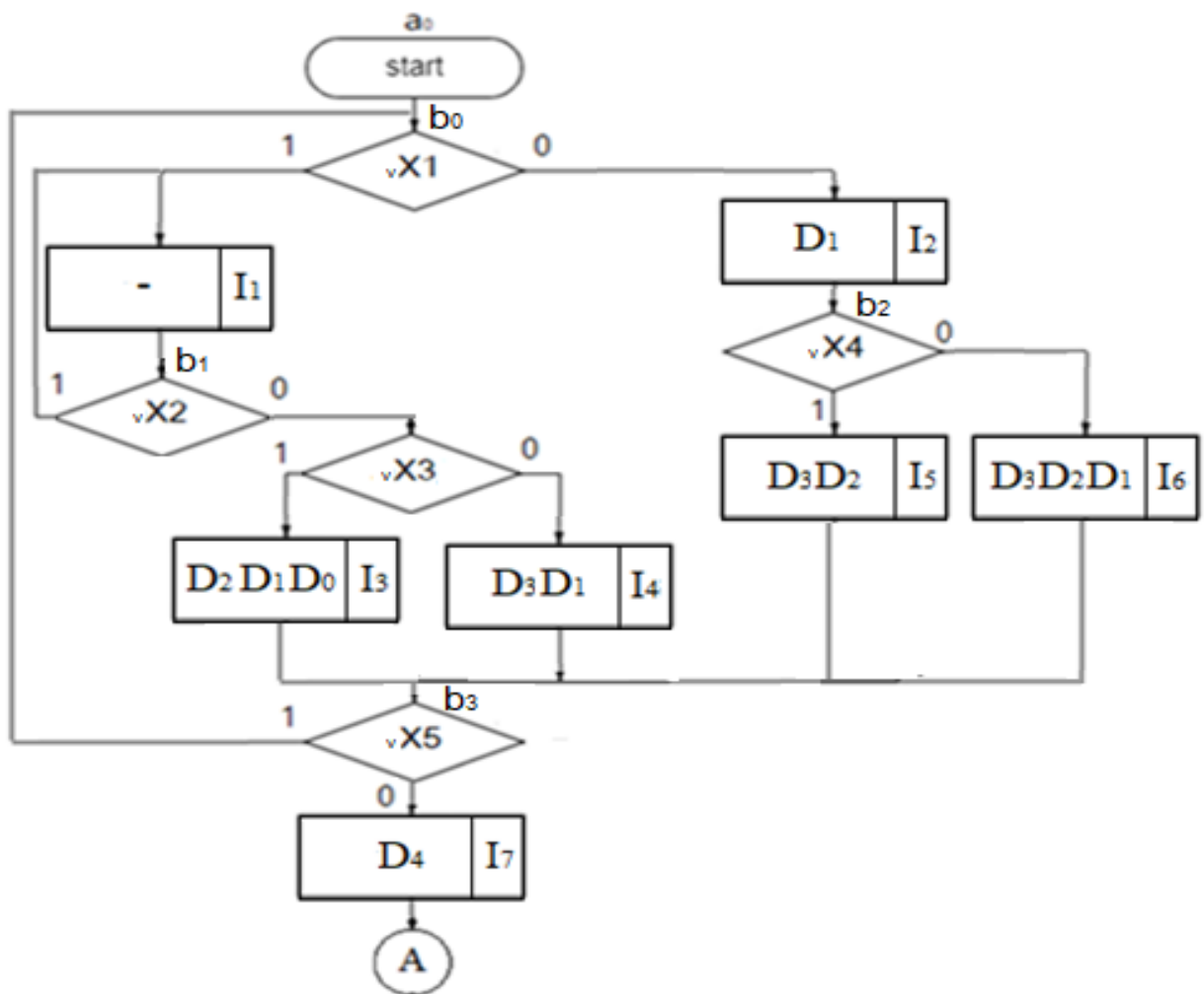


Рисунок 2.13 – Граф-схема роботи КА Мілі на D-тригерах

Відмітка станів та формування таблиці переходів УА Мілі:

$B = \{b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8, b_9, b_{10}, b_{11}\}$, $M_b = 11$ – множина станів КА Мілі.

$R_b = \log_2 M_b = 11 \rightarrow Q_b = \{Q_9, Q_8, Q_7, Q_6\}$ – розрядність кодів станів с 6 по 9.

Для кодування стану b_m необхідно 4 розряди, тобто 4 D-тригери. Тому позначимо їх наступними номерами після тих, що вже використовувалися – $D_b = D_6, D_7, D_8, D_9$.

Функції збудження пам'яті D-тригерів наведено у таблиці 2.

Таблиця 2.4 – Функції збудження пам'яті

K(b _i)				b
Q ₉	Q ₈	Q ₇	Q ₆	
0	0	0	0	b ₀
0	0	0	1	b ₁
0	0	1	0	b ₂
0	0	1	1	b ₃
0	1	0	0	b ₄
0	1	0	1	b ₅
0	1	1	0	b ₆
0	1	1	1	b ₇
1	0	0	0	b ₈
1	0	0	1	b ₉
1	0	1	0	b ₁₀
1	0	1	1	b ₁₁

Таблиця 2.5 – Таблиця переходів УА Мілі

b_m	$K(b_m)$	b_s	$K(b_s)$	X_n	D_a	D_b	H
b_0	0 0 0 0	b_1	0 0 0 1	uX_1	1	D_6	1
		b_2	0 0 1 0	$\overline{uX_1}$	D_1	D_7	2
b_1	0 0 0 1	b_1	0 0 0 1	uX_2	1	D_6	3
		b_3	0 0 1 1	$\overline{uX_2}uX_3$	D_3D_1	D_7D_6	4
		b_3	0 0 1 1	$\overline{uX_2}uX_3$	$D_2D_1D_0$	D_7D_6	5
b_2	0 0 1 0	b_3	0 0 1 1	uX_4	D_3D_2	D_7D_6	6
		b_3	0 0 1 1	$\overline{uX_4}$	$D_3D_2D_1$	D_7D_6	7
b_3	0 0 1 1	b_4	0 1 0 0	$\overline{uX_5}$	D_4	D_8	8
		b_1	0 0 0 1	uX_5uX_1	1	D_6	9
		b_2	0 0 1 0	$uX_5\overline{uX_1}$	D_1	D_6	10
...							

Повна таблиця наведена у додатку А.

Програмування КМПК на мові VHDL включає написання програм для описаних вище компонентів: керуючий автомат, лічильник, регістр та ROM. VHDL опис усіх цих блоків можна знайти у додатку В.

2.5 Реалізація проекту композиційного мікропрограмного пристрою керування за першим алгоритмом та його тестування в EDA Quartus II

Схема проекту композиційного мікропрограмного пристрою керування на розподілених ресурсах наведено на рисунку 2.14.

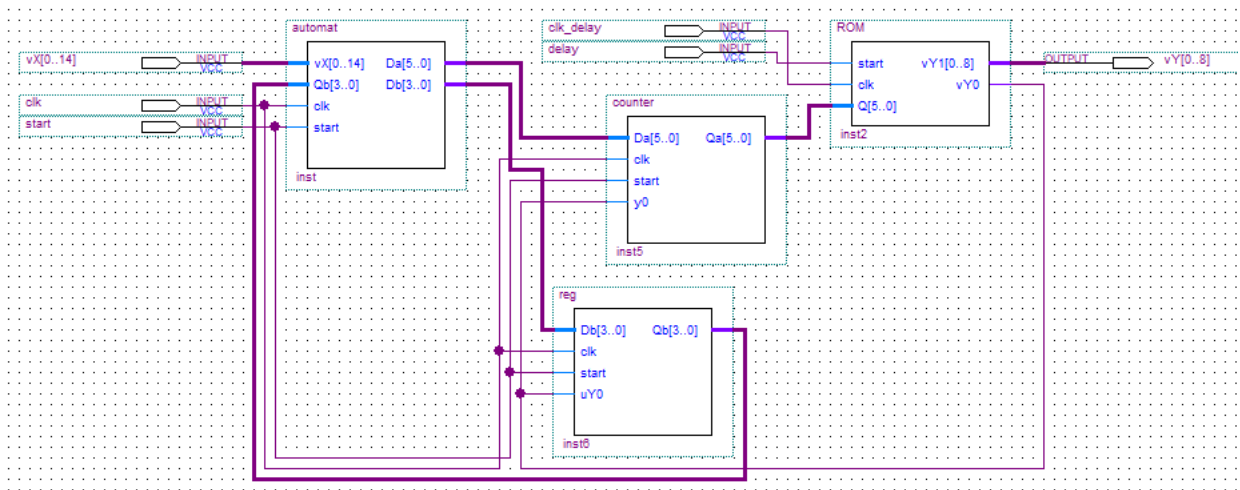


Рисунок 2.14– Схема композиційного мікропрограмного автомату керування

Для відокремлення сигналу uY0 із загальної шини uY(0:9) стало рішення відокремити сигнал у кодї програми та створити окремий вихідний порт.

Створений додатковий дозволяючий сигнал delay з затримкою в один такт, для правильного формування вихідних сигналів та їх ліку в лічильнику. Для активації роботи лічильника, обов'язково необхідно виставити в кодї програми початкове значення uY0=1

Для правильного відображенні станів на часових діаграмах, які будуть наведені у розділі з результатами експериментів, було введено додатковий вхідний сигнал CLK_delay для ROM для виконання поступаючих операцій з затримкою в півтакту.

2.6 Реалізація проекту композиційного мікропрограмного пристрою керування з вбудованою пам'яттю за першим алгоритмом та його тестування в EDA Quartus II

Для реалізації структури з використанням вбудованої пам'яті, необхідно додатково створити .mif-файл, в який буде записуватись дані, що необхідні для роботи ПЗП.

Файл має вигляд таблиці, в якій в заголовках рядків вказаний адрес чарунки в пам'яті, а в заголовках стовпців – зміщення відносно цього адресу (рисунк 2.15).

Addr	+000	+001	+010	+011	+100	+101	+110	+111
000000	0011000000	0100011000	0001000000	0010010000	0001000000	0000001000	0110010000	0001000000
001000	0011100000	0110010000	0001000000	0110010000	0000010000	0101000000	0000010000	0100001000
010000	0100000100	0011000000	0100011000	0001000000	0010010000	0001000000	0000001000	0110010000
011000	0001000000	0011100000	0110010000	0001000000	0110010000	0000010000	0101000000	0000010000
100000	0100001000	0100000010	0011000000	0100011000	0001000000	0010010000	0001000000	0000001000
101000	0110010000	0001000000	0011100000	0110010000	0001000000	0110010000	0000010000	0101000000
110000	0000010000	0100001000	0100000001					

Рисунок 2.15 - Mif-файл

Далі необхідно створити сам блок пам'яті за допомогою «MegaWizard Plug-In Manager» (рис. 2.16).

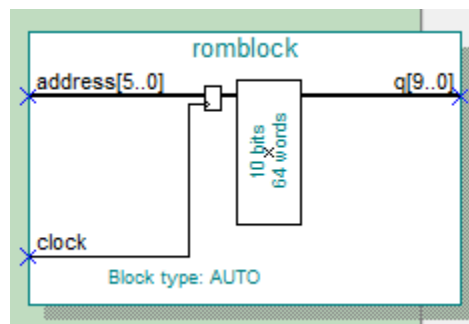


Рисунок 2.16- Блок вбудованої пам'яті

Для додавання сигналу start та відмежування сигналу uY0 від шини uY створимо обгортку створеного блоку вбудованої пам'яті - блок EMB_uY. Цей блок буде замість ROM у попередній структурі. У наступному лістингу описаний цей блок мовою VHDL

Лістинг 2.1 – Опис EMB_Y на VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity EMB_uY is
    port(
        start, clk : in STD_LOGIC;
        Qa : in STD_LOGIC_VECTOR(5 downto 0);
```

```

        uY : out STD_LOGIC_VECTOR(0 to 9);
        uY0 : out STD_LOGIC
    );
end EMB_uY;
architecture EMB_uY of EMB_uY is
    COMPONENT romEmb
        PORT(clock : IN STD_LOGIC;
            address : IN STD_LOGIC_VECTOR(5 DOWNTO 0);
            q : OUT STD_LOGIC_VECTOR(9 DOWNTO 0)
        );
    END COMPONENT;

    SIGNAL    NET1 : STD_LOGIC_VECTOR(9 DOWNTO 0);

BEGIN

    U1 : romEmb
        PORT MAP(clock => CLK,
            address => Qa,
            q => NET1);
    process(start, clk)
        begin
            if (start = '0') then
                uY <= "1000000000";
                uY0 <= '1';
            else
                uY <= NET1(9 DOWNTO 0);
                uY0 <= NET1(9);
            end if;
        end process;
end EMB_uY;

```

Схема проекту композиційного мікропрограмного пристрою керування з вбудованою пам'яттю наведено на рисунку 2.17.

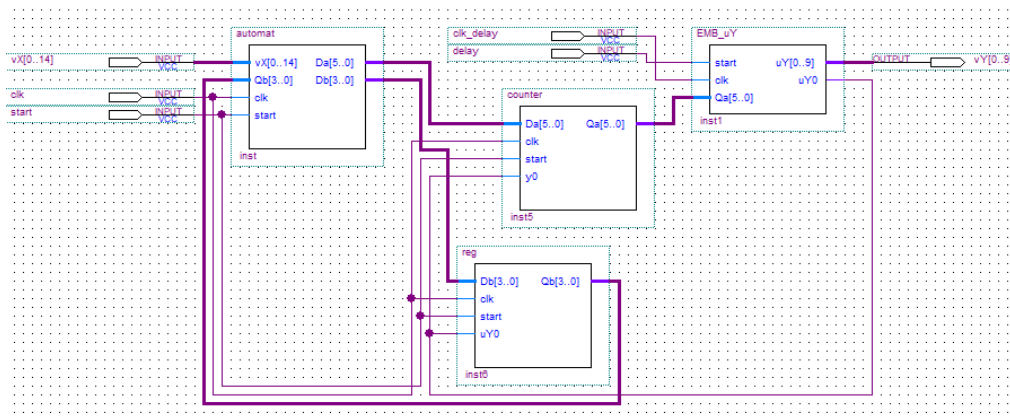


Рисунок 2.17– Схема композиційного мікропрограмного автомату керування з вбудованою пам'яттю

Призначення сигналів схеми та принцип її побудови та функціонування аналогічний структурі на розподілених ресурсах.

Синтез структур за іншими двома алгоритмами як вже було сказано не наведений через занадто великий розмір таблиць та через те, що їх синтез цих алгоритмів здійснюється аналогічним чином.

3 РОЗРОБКА ПАРАЛЕЛЬНИХ МОДЕЛЕЙ

У цьому розділі здійснюється розробка паралельних схем у пакеті Quartus II для паралельного запуску розроблених у попередньому розділі структур та алгоритмів на одному кристалі.

Спочатку була здійснена розробка паралельної схеми першого та другого алгоритму, потім паралельної схеми першого, другого та третього алгоритму.

У цьому розділі наведена розробка паралельної схеми першого, другого та третього алгоритму. Розробка паралельної схеми першого та другого алгоритму у цьому розділі не наведена оскільки їх розробка повністю аналогічна розробці паралельної схеми першого, другого та третього алгоритму за виключенням того, що на кристалі два автомати, а не три.

3.1 Компоновка паралельної схеми базових структур першого, другого та третього алгоритму

Необхідно розробити схеми для паралельного запуску на одному кристалі трьох автоматів однієї і тієї самої структури, які працюють за першим, другим та третім алгоритмами.

Для здійснення паралельного запуску на одному кристалі трьох автоматів базової структури, які працюють за першим, другим та третім алгоритмами все, що потрібно це власно розмістити їх на одному кристалі і тактувати їх одним і тим самим тактовим імпульсом.

Отже на рисунку 3.1 можна побачити на одній схемі автомат базової структури який працює за першим ГСА, автомат базової структури який працює за другим ГСА та автомат базової структури який працює за третім ГСА.

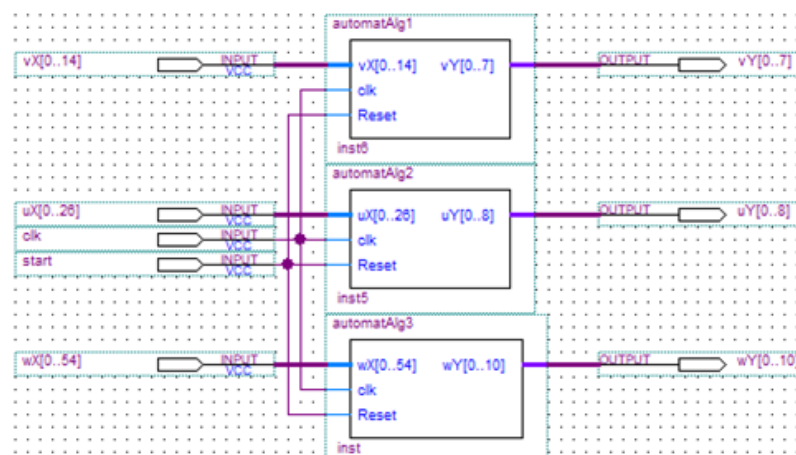


Рисунок 3.1 – Паралельна схема базових структур

Можна побачити, що всі автомати тактуються одним і тим самим тактовим сигналом clk.

3.2 Компоновка паралельної схеми композиційних автоматів першого, другого та третього алгоритму

Подібним чином розмістимо на єдиній схемі автомати композиційних структур які працюють за першим, другим та третім ГСА.

На рисунку 3.2 можна побачити на одній схемі автомат композиційної структури який працює за першим ГСА, автомат композиційної структури який

працює за другим ГСА та автомат композиційної структури який працює за третім ГСА.

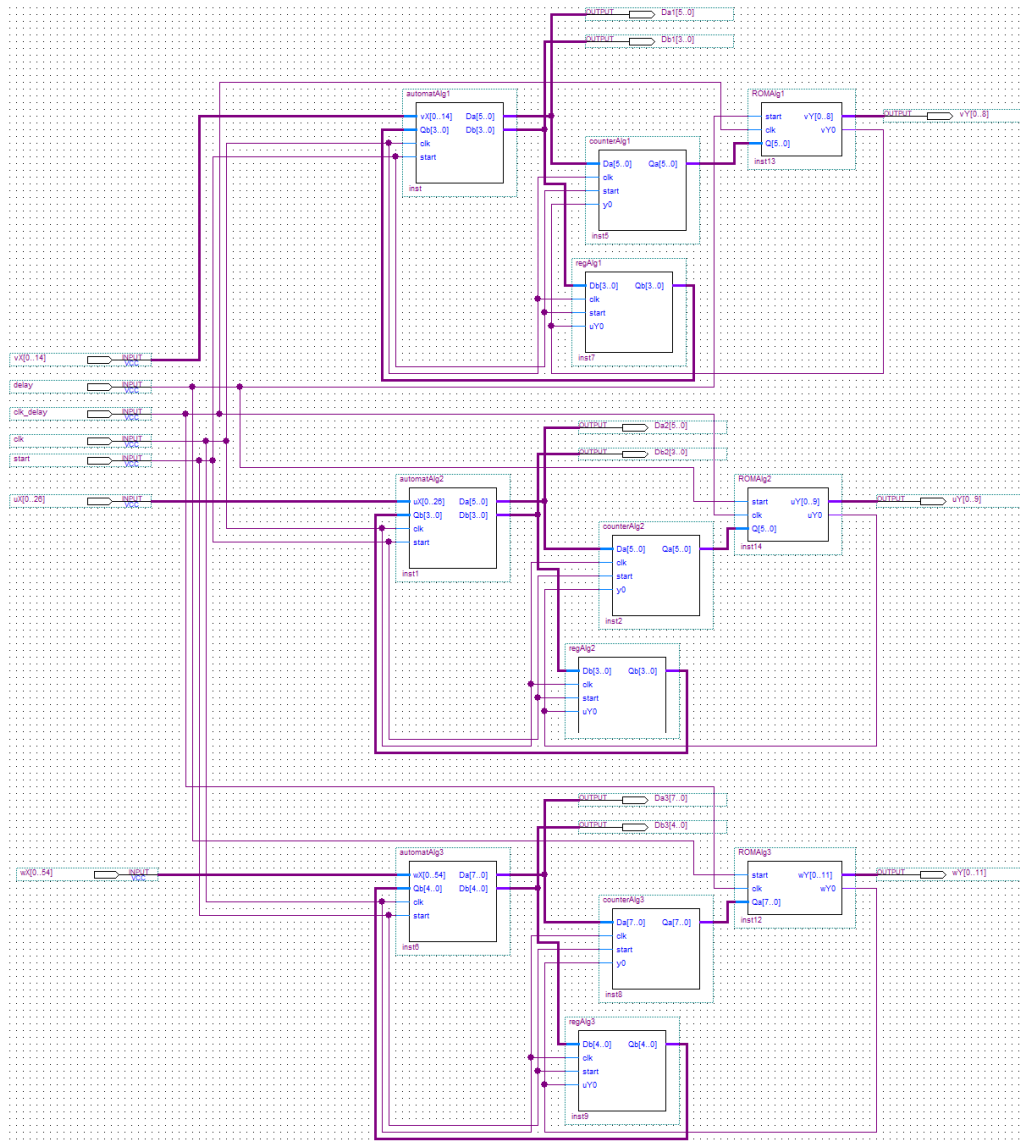


Рисунок 3.2 – Паралельна схема композиційних автоматів 1, 2 та 3 алгоритмів

3.3 Компоновка паралельної схеми композиційних автоматів структур з вбудованою пам'яттю першого, другого та третього алгоритму

Подібним чином розмістимо на єдиній схемі автомати композиційних структур з вбудованою пам'яттю які працюють за першим, другим та третім ГСА.

На рисунку 3.3 можна побачити на одній схемі автомат композиційної структури з вбудованою пам'яттю, який працює за першим ГСА, автомат композиційної структури з вбудованою пам'яттю який працює за другим ГСА та автомат композиційної структури з вбудованою пам'яттю який працює за третім ГСА.

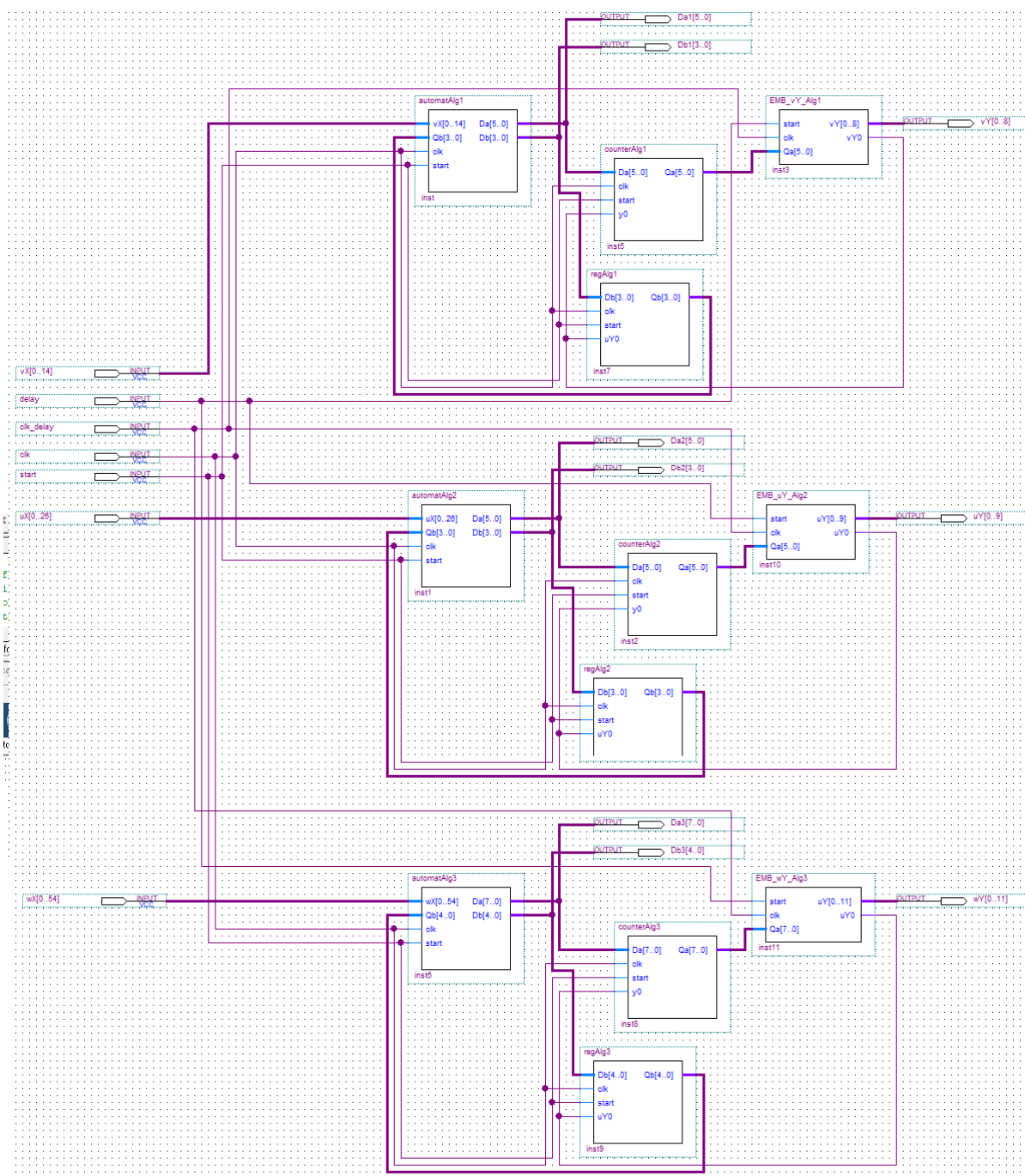


Рисунок 3.3 – Паралельна схема структур з вбудованою пам'яттю

4 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТІВ

4.1 Тестування розроблених схем різних структур які працюють за першим алгоритмом

Для перевірки працездатності зібраної схеми автомата базової структури який працює за першим алгоритмом (рисунок 4.1) запустимо симуляцію часових діаграм в Quartus II.

Встановимо значення входних сигналів vX в «001010000001000» та перевіримо за ГСА1.

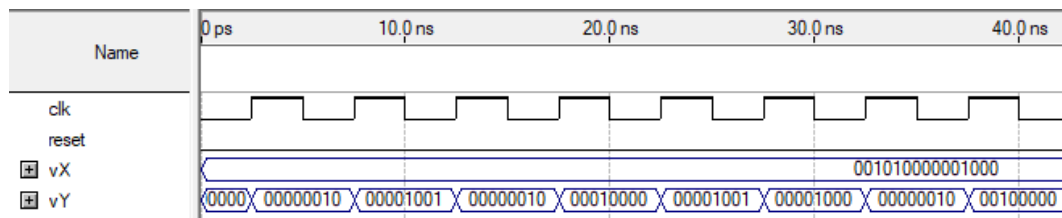


Рисунок 4.1 - Часова діаграма роботи першої частини ГСА1 рис. 2.1 (основного блоку) зі значенням входів vX в «001010000001000»

Як бачимо частина результуючої часової діаграми працює відповідно до ГСА1. Інші частини часової діаграми можна знайти у додатку Г.

Для перевірки працездатності зібраної схеми автомата композиційної структури на розподілених ресурсах який працює за першим алгоритмом (рисунок 4.2) запустимо симуляцію часових діаграм в Quartus II.

Встановимо значення входних сигналів vX в «001010000001000» та перевіримо за ГСА1

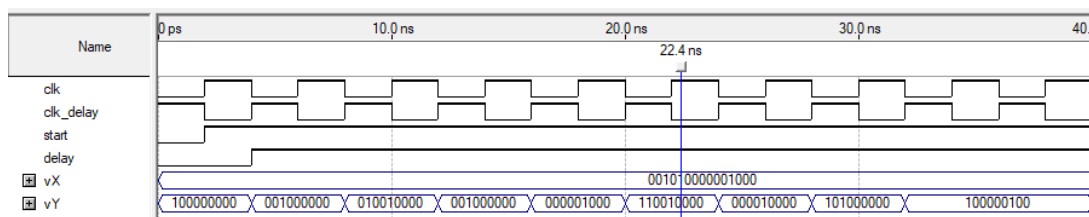


Рисунок 4.2 - Часова діаграма роботи першої частини ГСА1 рис. 2.1 (основного блоку) зі значенням входів vX в «001010000001000»

Як бачимо на рисунку 4.2 частина результуючої часової діаграми працює відповідно до ГСА1. Інші частини часової діаграми можна знайти у додатку Г.

Для перевірки працездатності зібраної схеми автомата композиційної структури з вб. пам'яттю який працює за першим алгоритмом (рисунок 4.3) запусимо симуляцію часових діаграм в Quartus II.

Встановимо значення вхідних сигналів vX в «001010000001000» та перевіримо за ГСА1

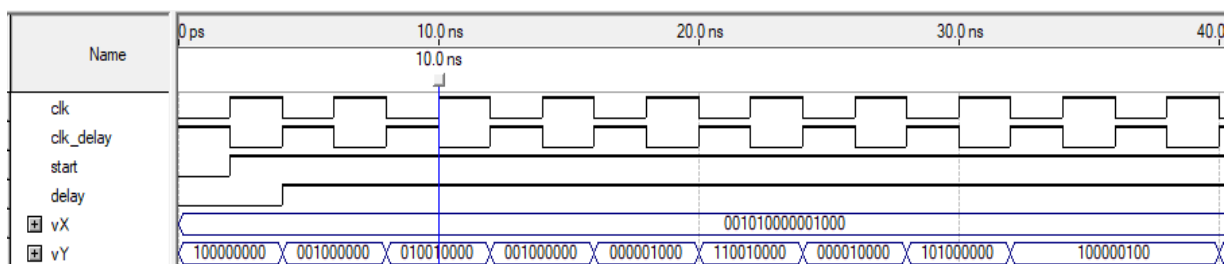


Рисунок 4.3 - Часова діаграма роботи першої частини ГСА1 рис. 2.1 (основного блоку) зі значенням входів vX в «001010000001000»

Як бачимо на рисунку 4.3 частина результуючої часової діаграми працює відповідно до ГСА1. Інші частини часової діаграми можна знайти у додатку Г.

Із діаграм можна зробити висновок, що зібрані схеми працюють правильно, оскільки послідовність вихідних схем сигналів відповідає першій ГСА.

4.1.1 Тестування структур першого алгоритму на різних мікросхемах та порівняння апаратних витрат

У таблицях 4.1, 4.2 та 4.3 наведені апаратні затрати на різних мікросхемах для різних структур першого алгоритму та для більш наочного порівняння на графіках на рисунках 4.4, 4.5 та 4.6 наведені дані з цих таблиць.

Таблиця 4.1 - Апаратурні затрати тестування базової структури на різних мікросхемах.

Family	Les	Rg	Pins	Mem (bits)
Cyclone IV GX (EP4CGX50CF23C^)	140	6	25	0
Startix III (EP3SL70F484C2)	103	6	25	0
Arria GX (EP1AGX60CF484C6)	89	6	25	0

Таблиця 4.2 - Апаратурні затрати тестування композиційної структури з лічильником на різних мікросхемах.

Family	Les	Rg	Pins	Mem (bits)
Cyclone IV GX (EP4CGX50CF23C^)	88	23	28	0
Startix III (EP3SL70F484C2)	56	23	28	0
Arria GX (EP1AGX60CF484C6)	53	23	28	0

Таблиця 4.3 - Апаратурні затрати тестування композиційної структури з лічильником та вбудованою пам'яттю на різних мікросхемах.

Family	Les	Rg	Pins	Mem (bits)
Cyclone IV GX (EP4CGX50CF23C^)	67	14	28	640
Startix III (EP3SL70F484C2)	53	14	28	640
Arria GX (EP1AGX60CF484C6)	52	14	28	640

На графіках нижче можна побачити зміну кількості регістрів та логічних елементів залежно від структури та мікросхеми.

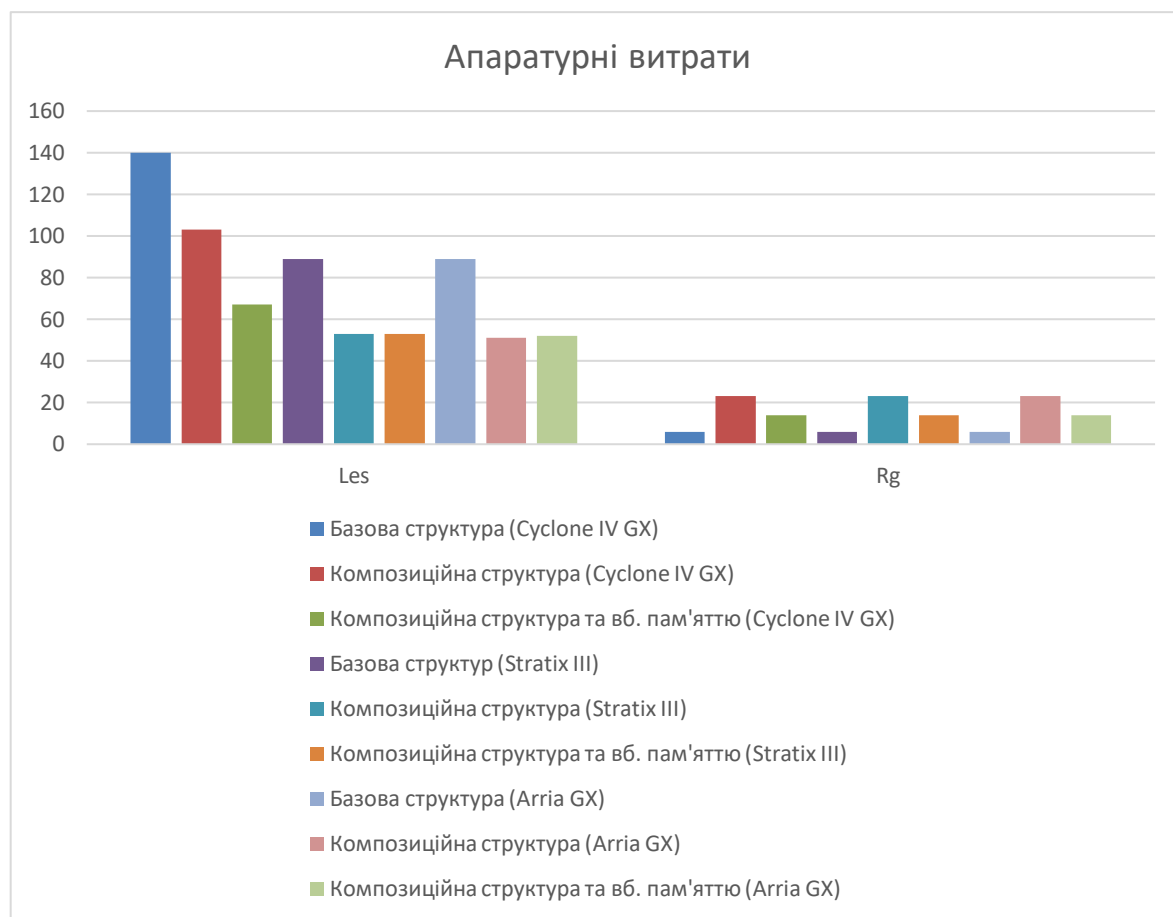


Рисунок 4.4 - Апаратурні затрати на різних структурах та мікросхемах.

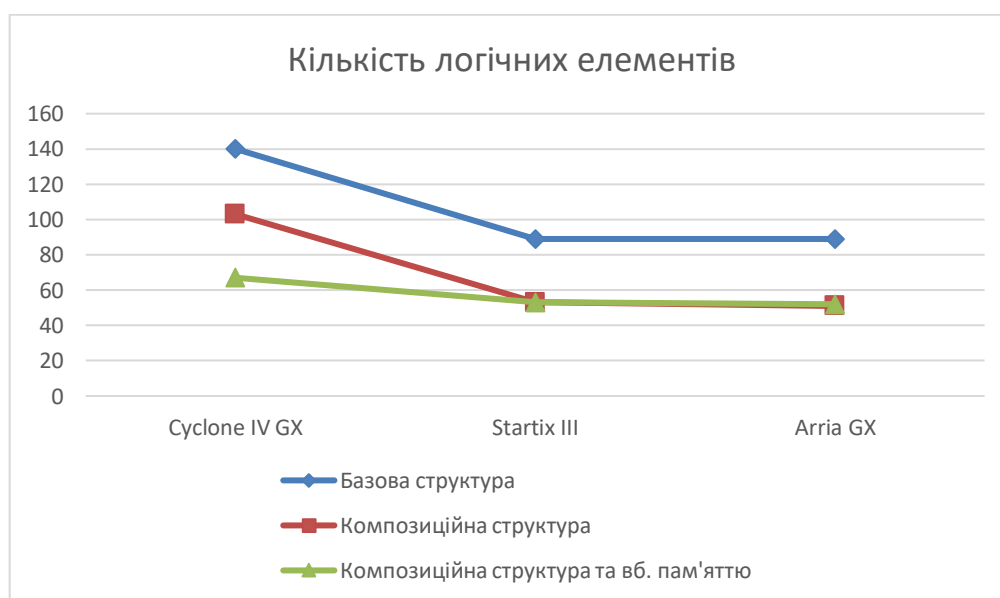


Рисунок 4.5 – Затрати логічних елементів на різних структурах та мікросхемах.



Рисунок 4.6 – Затрати регістрів на різних структурах та мікросхемах.

З графіків ресурсів можна зробити деякі висновки. Можна побачити, що на мікросхемі Cyclone кількість логічних елементів змінюється залежно від структури. На мікросхемах Stratix та Arria кількість логічних елементів композиційних структур майже однакова, але порівняно з базовою структурою кількість займаних ними розподілених ресурсів значно менша. Хоч у композиційних структур і майже однакова кількість логічних елементів, але як можна побачити на графіку у композиційної структури з вб. пам'яттю кількість регістрів значно зменшується порівняно з композиційною структурою на розподілених ресурсах.

Також необхідно помітити, що кількість займаних ресурсів однією і тією ж схемою змінюється залежно від мікросхеми на якій проводяться тестування. Тобто в ході тестувань можна побачити, що на Stratix III та Arria GX кількість займаних ресурсів однією і тією ж самою схемою менші ніж на Cyclone. Це зумовлено різними структурними особливостями та характеристиками різних мікросхем таких як площа мікросхеми від якої залежить щільність упаковки активних логічних елементів на кристалі та інці.

4.1.2 Порівняння часових характеристик

У таблицях 4.4, 4.5 та 4.6 наведені часові характеристики на різних мікросхемах для різних структур та для більш наочного порівняння на графіках на рисунках 4.7, 4.8, 4.9 наведені дані з цих таблиць.

Таблиця 4.4 - Часові характеристики при тестуванні базової структури на різних мікросхемах.

Family	Tsu(ns)	Tco(ns)	Tpd(ns)
Cyclone IV GX (EP4CGX50CF23C [^])	5,238	4,805	7,908
Startix III (EP3SL70F484C2)	4,943	5,617	7,292
Arria GX (EP1AGX60CF484C6)	5,097	4,707	7,96

Таблиця 4.5 - Часові характеристики при тестуванні композиційної структури на різних мікросхемах.

Family	Tsu(ns)	Tco(ns)	Tpd(ns)
Cyclone IV GX (EP4CGX50CF23C [^])	14,987	11,697	15,036
Startix III (EP3SL70F484C2)	11,943	10,348	14,7
Arria GX (EP1AGX60CF484C6)	12,51	9,986	15,6

Таблиця 4.6 - Часові характеристики при тестуванні композиційної структури з вбудованою пам'яттю на різних мікросхемах.

Family	Tsu(ns)	Tco(ns)	Tpd(ns)
Cyclone IV GX (EP4CGX50CF23C [^])	14,377	11,668	17,608
Startix III (EP3SL70F484C2)	12,474	10,4	18,1
Arria GX (EP1AGX60CF484C6)	13,126	10,724	18,403

Для наочності зобразимо на графіках часові затримки структур на різних мікросхемах.

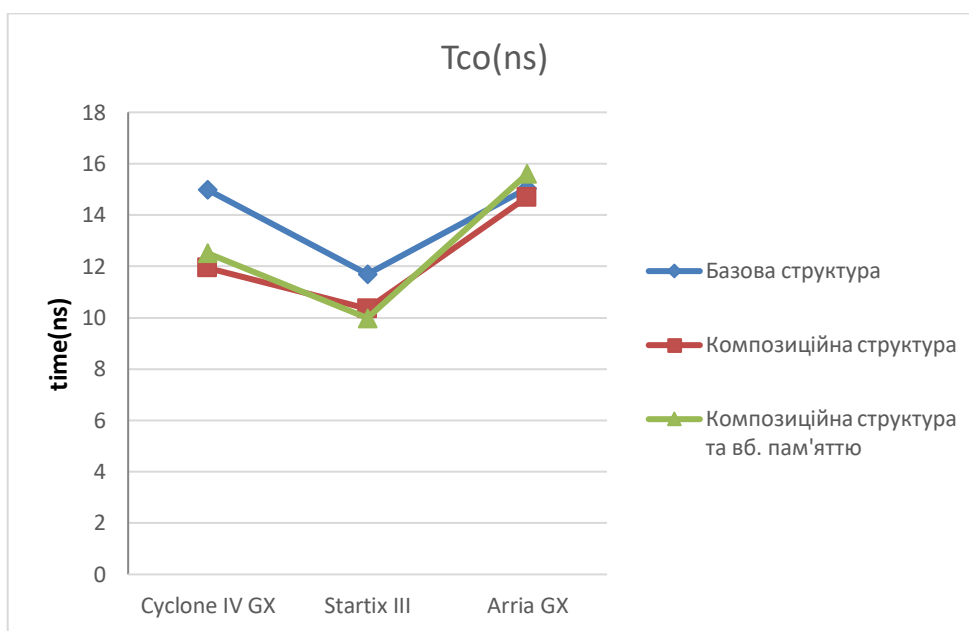


Рисунок 4.7 – Затримка tco при тестуванні різних структур автомату який працює за першою ГСА на різних мікросхемах



Рисунок 4.8 – Затримка tpd при тестуванні різних структур автомату який працює за першою ГСА на різних мікросхемах

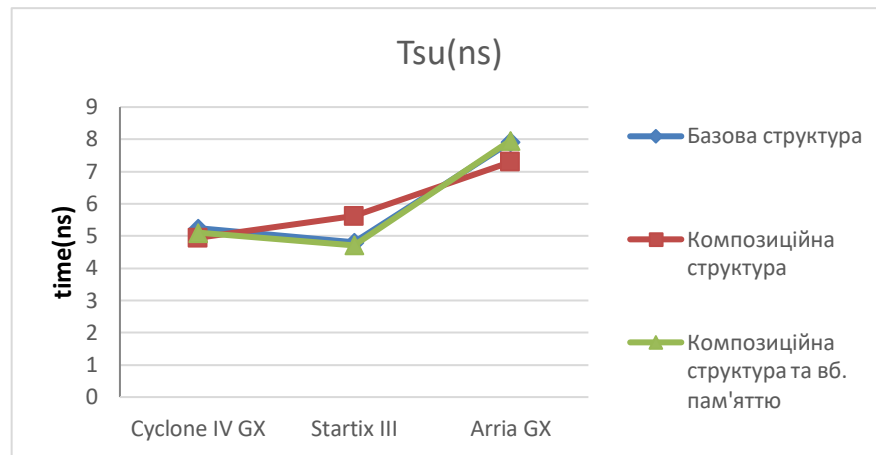


Рисунок 4.9 – Затримка tsu при тестуванні різних структур автомату який працює за першою ГСА на різних мікросхемах

Із графіків можна помітити, що на мікросхемах Cyclone та Stratix затримки tco та trd композиційних структур нижче ніж у базової структури. Для Arria ці затримки майже рівні.

4.1.3 Топології мікросхем

На рисунках 4.10, 4.11, 4.12 можна бачити топології мікросхем на яких проводилися тестування. На них можна побачити різні структури автомата який працює за першою ГСА.

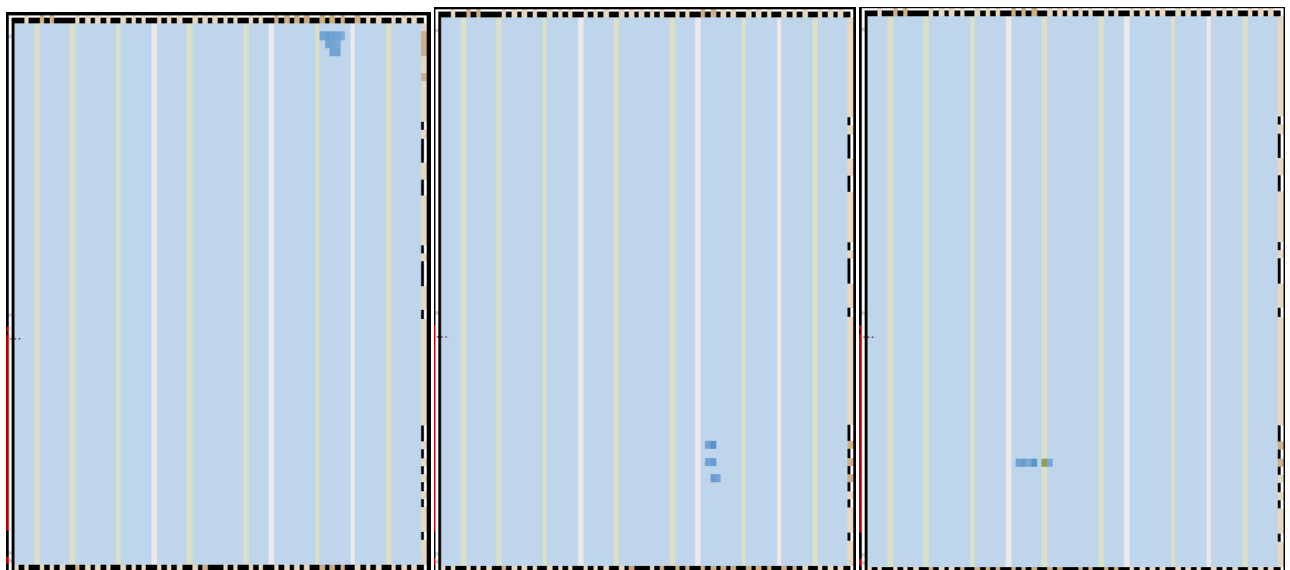


Рисунок 4.10 – Розміщення базової структури (зліва), композиційної структури (посередині) та композиційної структури з вбудованою пам'яттю (справа) на мікросхемі Cyclone IV GX.

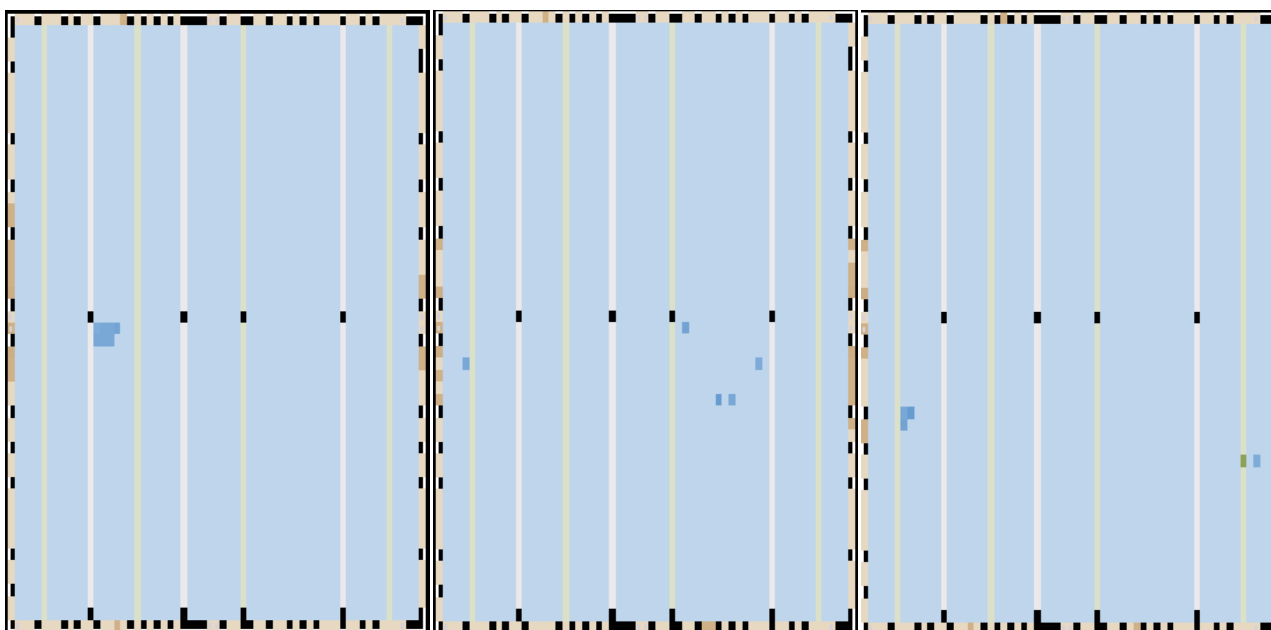


Рисунок 4.11 – Розміщення базової структури (зліва), композиційної структури (посередині) та композиційної структури з вбудованою пам'яттю (справа) на мікросхемі Startix III.

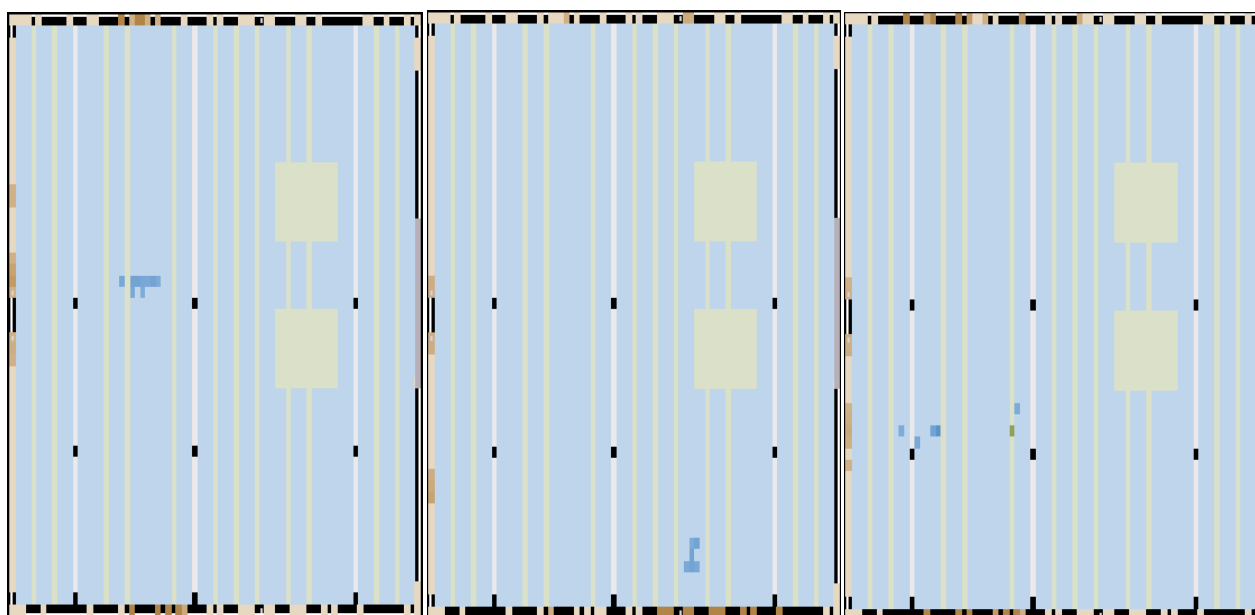


Рисунок 4.12 – Розміщення базової структури (зліва), композиційної структури (посередині) та композиційної структури з вбудованою пам'яттю (справа) на мікросхемі Arria GX.

За результатами тестувань можна прослідкувати зменшення логічних елементів на схемі залежно від структури.

4.2 Тестування розроблених схем різних структур які працюють за другим алгоритмом

Для перевірки працездатності зібраної схеми автомата базової структури який працює за другим алгоритмом (рисунок 4.13) запустимо симуляцію часових діаграм в Quartus II.

Встановимо значення вхідних сигналів uX в «111110010010110010111110000» та перевіримо за ГСА2

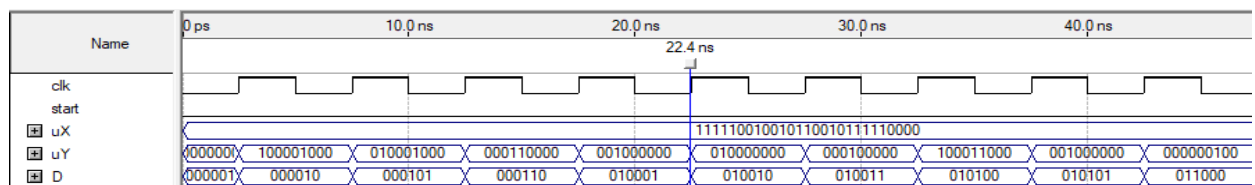


Рисунок 4.13 - Часова діаграма роботи першої частини ГСА2 рис. 2.4 (основного блоку) зі значенням входів uX в «111110010010110010111110000»

Як бачимо на рисунку 4.13 частина результуючої часової діаграми працює відповідно до ГСА2. Інші частини часової діаграми можна знайти у додатку Г.

Для перевірки працездатності зібраної схеми автомата композиційної структури на розподілених ресурсах який працює за другим алгоритмом (рисунок 4.14) запустимо симуляцію часових діаграм в Quartus II.

Встановимо значення вхідних сигналів uX в «111110010010110010111110000» та перевіримо за ГСА2

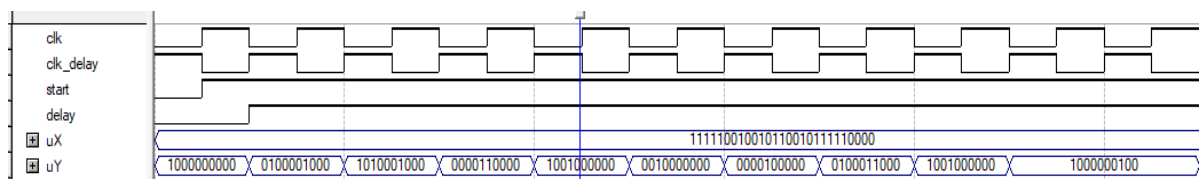


Рисунок 4.14 - Часова діаграма роботи першої частини ГСА2 рис. 2.4 (основного блоку) зі значенням входів uX в «111110010010110010111110000»

Як бачимо на рисунку 4.14 частина результуючої часової діаграми працює відповідно до ГСА2. Інші частини часової діаграми можна знайти у додатку Г.

Для перевірки працездатності зібраної схеми автомата композиційної структури з вб. пам'яттю який працює за другим алгоритмом (рисунок 4.15) запусимо симуляцію часових діаграм в Quartus II.

Встановимо значення вхідних сигналів uX в «111110010010110010111110000» та перевіримо за ГСА2.

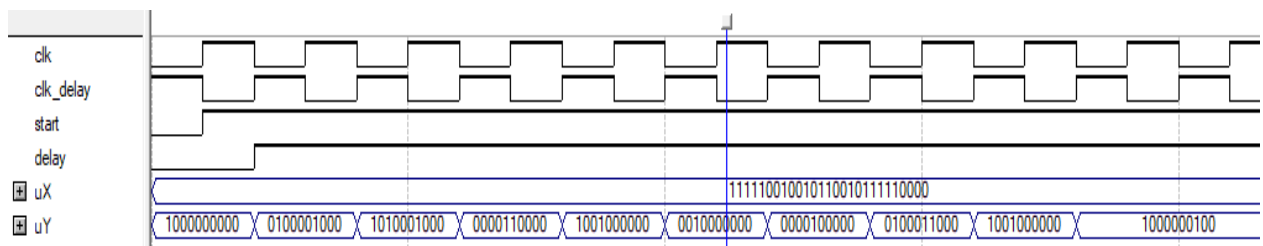


Рисунок 4.15 - Часова діаграма роботи першої частини ГСА2 рис. 2.4 (основного блоку) зі значенням входів uX в «111110010010110010111110000»

Як бачимо на рисунку 4.15 частина результуючої часової діаграми працює відповідно до ГСА2. Інші частини часової діаграми можна знайти у додатку Г.

Із діаграм можна зробити висновок, що зібрані схеми працюють правильно, оскільки послідовність вихідних схем сигналів відповідає другій ГСА.

4.2.1 Тестування структур другого алгоритму на різних мікросхемах та порівняння апаратних витрат

У таблицях 4.7, 4.8 та 4.9 наведені апаратні затрати на різних мікросхемах для різних структур та для більш наочного порівняння на графіках на рисунках 4.16, 4.17, 4.18 наведені дані з цих таблиць.

Таблиця 4.7 - Апаратурні затрати тестування базової структури на різних мікросхемах.

Family	Les	Rg	Pins	Mem (bits)
Cyclone IV GX (EP4CGX50CF23C^)	215	6	38	0
Startix III (EP3SL70F484C2)	156	6	38	0
Arria GX (EP1AGX60CF484C6)	155	6	38	0

Таблиця 4.8 - Апаратурні затрати тестування композиційної структури з лічильником на різних мікросхемах.

Family	Les	Rg	Pins	Mem (bits)
Cyclone IV GX (EP4CGX50CF23C^)	148	24	41	0
Startix III (EP3SL70F484C2)	101	24	41	0
Arria GX (EP1AGX60CF484C6)	101	24	41	0

Таблиця 4.9 - Апаратурні затрати тестування композиційної структури з лічильником та вбудованою пам'яттю на різних мікросхемах.

Family	Les	Rg	Pins	Mem (bits)
Cyclone IV GX (EP4CGX50CF23C^)	113	14	41	640
Startix III (EP3SL70F484C2)	100	14	41	640
Arria GX (EP1AGX60CF484C6)	100	14	41	640

Для наочності зобразимо на графіках ресурси на різних мікросхемах.

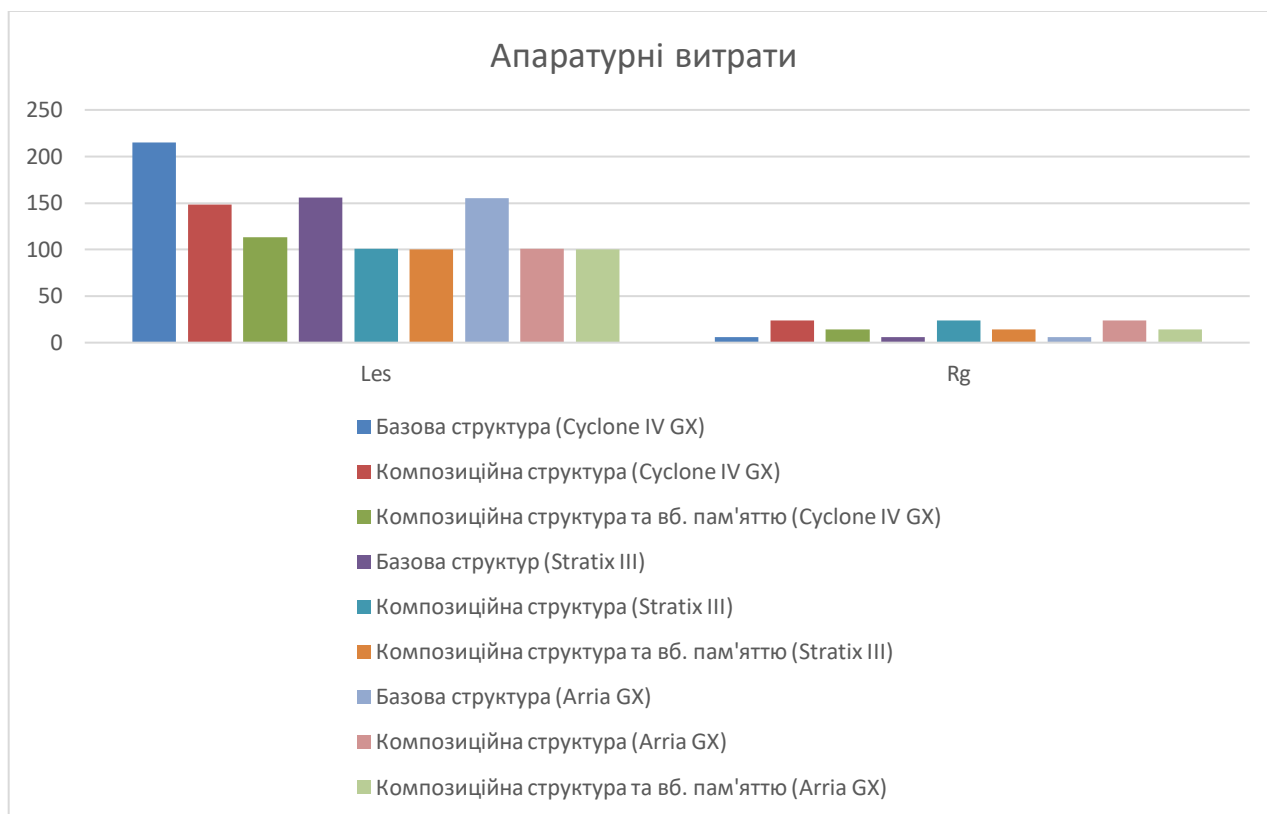


Рисунок 4.16 - Апаратні затрати на різних структурах та мікросхемах.

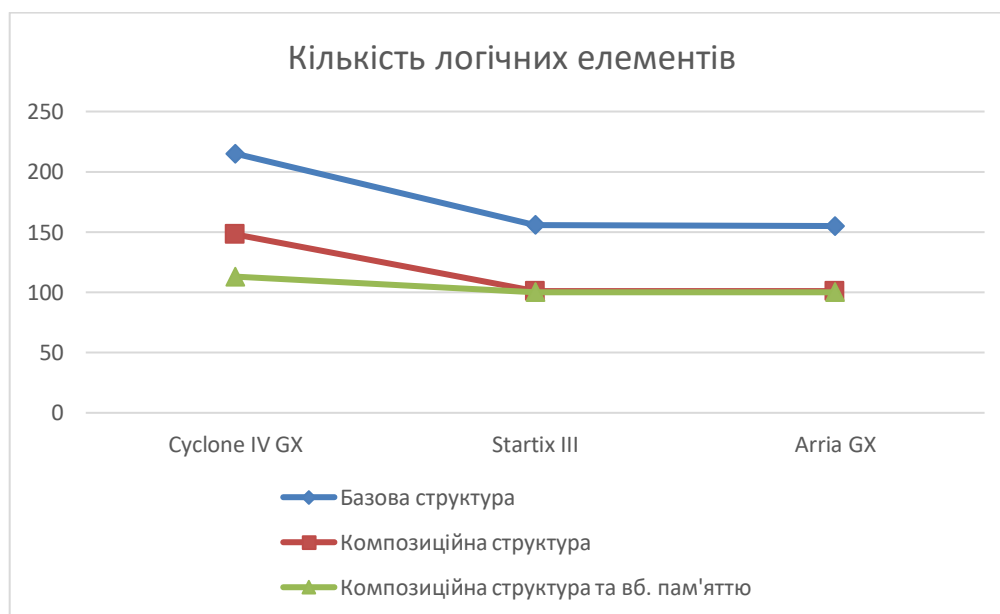


Рисунок 4.17 – Затрати логічних елементів на різних структурах та мікросхемах.



Рисунок 4.18 – Затрати регістрів на різних структурах та мікросхемах.

Стосовно апаратних витрат із графіків можна зробити ті ж самі висновки, що і до попереднього алгоритму.

4.2.2 Порівняння часових характеристик

У таблицях 4.10, 4.11 та 4.12 наведені часові характеристики на різних мікросхемах для різних структур та для більш наочного порівняння на графіках на рисунках 4.19, 4.20, 4.21 наведені дані з цих таблиць.

Таблиця 4.10 - Часові характеристики при тестуванні базової структури на різних мікросхемах.

Family	Tsu(ns)	Tco(ns)	Tpd(ns)
Cyclone IV GX (EP4CGX50CF23C^)	6,942	5,516	10,495
Startix III (EP3SL70F484C2)	6,529	5,057	9,42
Arria GX (EP1AGX60CF484C6)	7,69	4,7	9,4

Таблиця 4.11 - Часові характеристики при тестуванні композиційної структури на різних мікросхемах.

Family	Tsu(ns)	Tco(ns)	Tpd(ns)
Cyclone IV GX (EP4CGX50CF23C^)	14,454	9,29	19,611
Startix III (EP3SL70F484C2)	14,9	8,6	18,012
Arria GX (EP1AGX60CF484C6)	14,673	8,61	17,919

Таблиця 4.12 - Часові характеристики при тестуванні композиційної структури з вбудованою пам'яттю на різних мікросхемах.

Family	Tsu(ns)	Tco(ns)	Tpd(ns)
Cyclone IV GX (EP4CGX50CF23C^)	15,891	11,555	23,092
Startix III (EP3SL70F484C2)	15,266	10,451	21,179
Arria GX (EP1AGX60CF484C6)	16,1	10,384	18,708

Для наочності зобразимо на графіках часові затримки структур на різних мікросхемах.

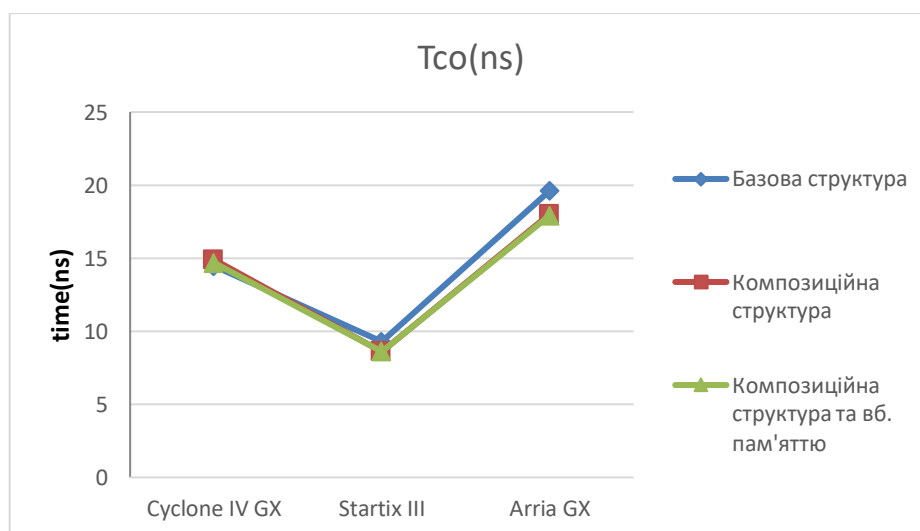


Рисунок 4.19 – Затримка tco при тестуванні різних структур автомату який працює за другою ГСА на різних мікросхемах

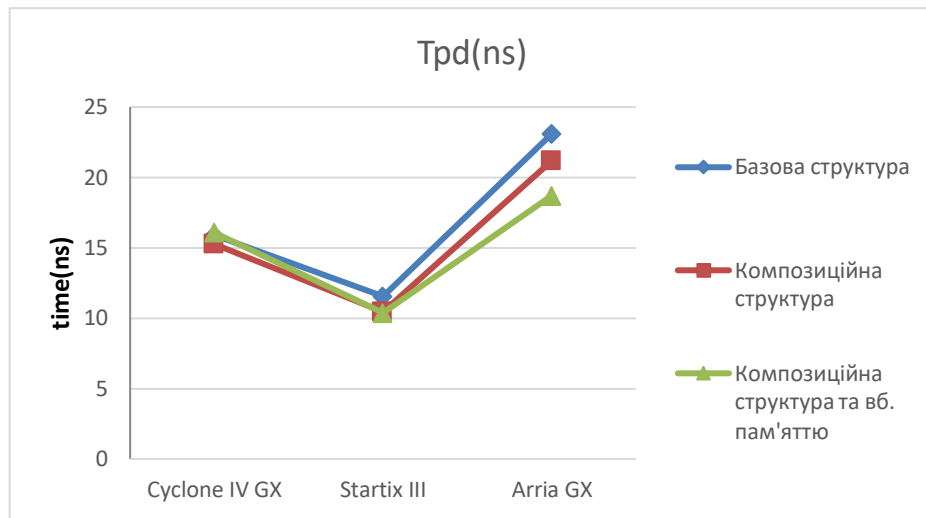


Рисунок 4.20 – Затримка tpd при тестуванні різних структур автомату який працює за другою ГСА на різних мікросхемах

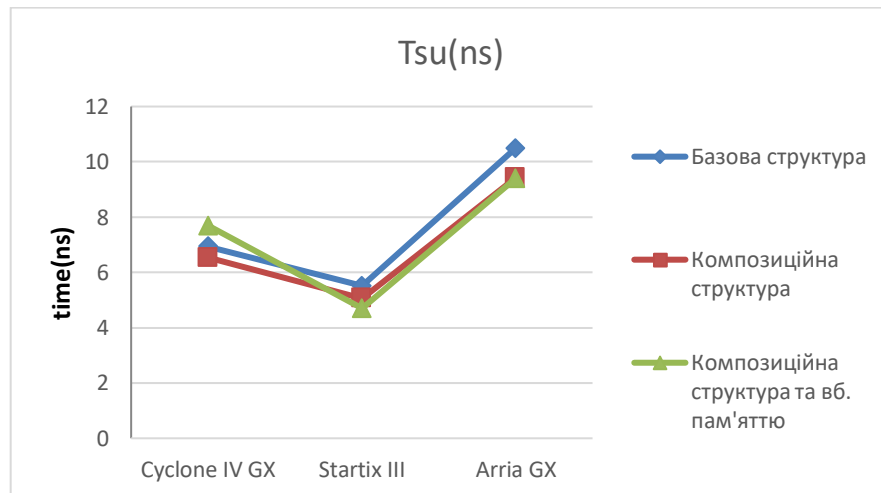


Рисунок 4.21 – Затримка tsu при тестуванні різних структур автомату який працює за другою ГСА на різних мікросхемах

На графіках можна побачити, що найбільш значущі затримки tco та tpd у мікросхеми Cyclone в цілому рівні для всіх структур, але на мікросхемах Stratix та Arria композиційні структури мають нижчі затримки ніж базова структура.

4.2.3 Топології мікросхем

На рисунках 4.22, 4.23, 4.24 можна бачити топології мікросхем на яких проводилися тестування. На них можна побачити різні структури автомата який працює за другою ГСА.

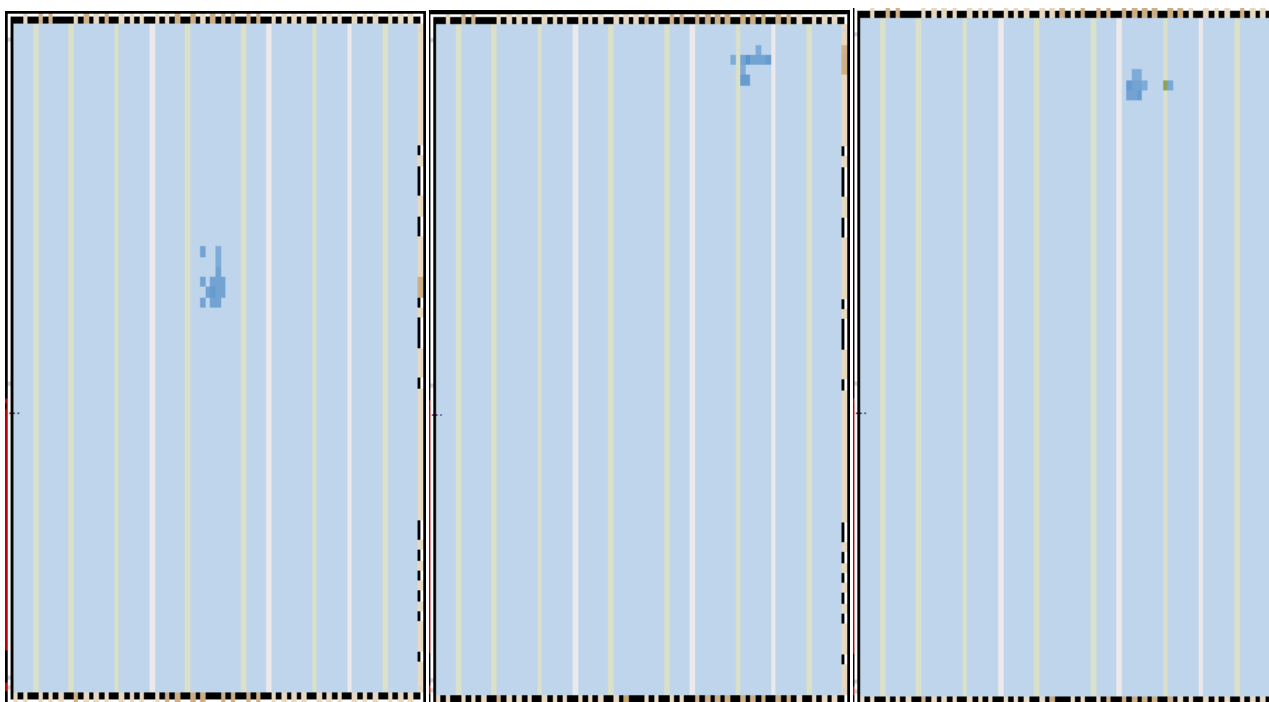


Рисунок 4.22 – Розміщення базової структури (зліва), композиційної структури (посередині) та композиційної структури з вбудованою пам'яттю (справа) на мікросхемі Cyclone IV GX.

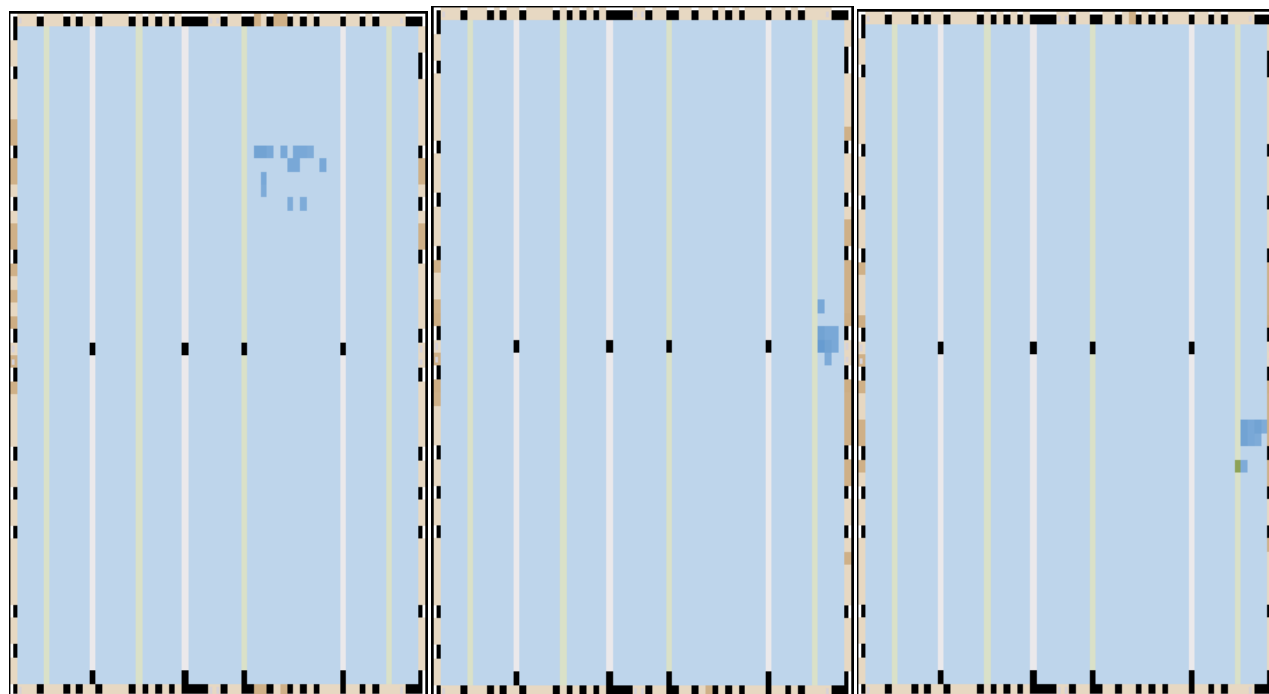


Рисунок 4.23 – Розміщення базової структури (зліва), композиційної структури (посередині) та композиційної структури з вбудованою пам'яттю (справа) на мікросхемі Startix III.

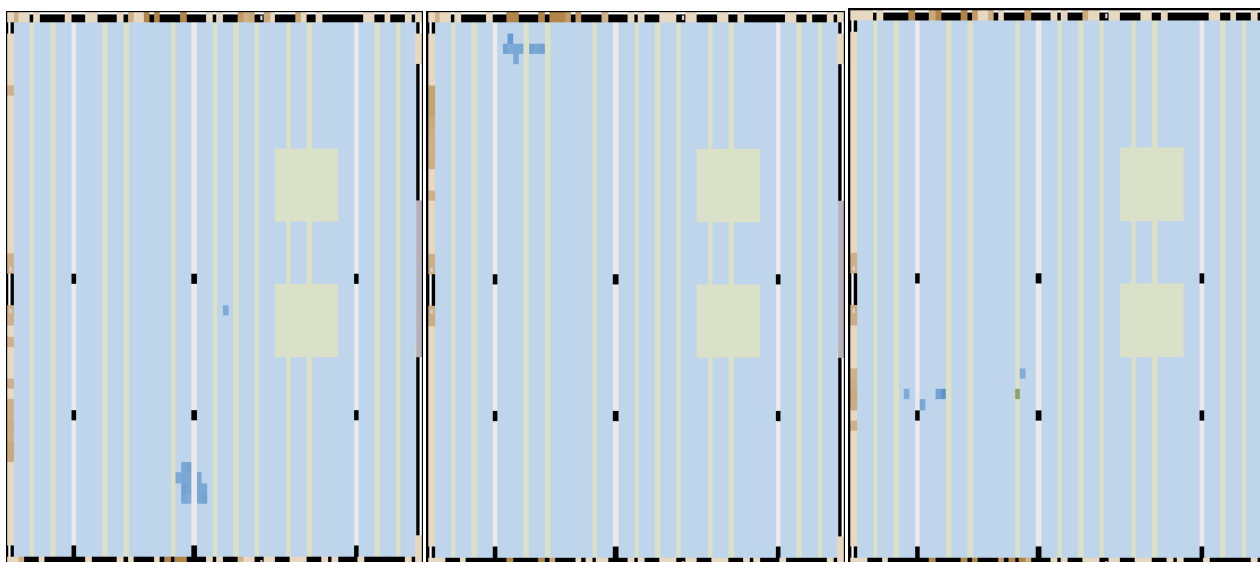


Рисунок 4.24 – Розміщення базової структури (зліва), композиційної структури (посередині) та композиційної структури з вбудованою пам'яттю (справа) на мікросхемі Arria GX.

За результатами тестувань можна прослідкувати зменшення логічних елементів на схемі залежно від структури.

4.3 Тестування розроблених схем різних структур які працюють за третім алгоритмом

Для перевірки працездатності зібраної схеми автомата базової структури який працює за третім алгоритмом (рисунок 4.25) запустимо симуляцію часових діаграм в Quartus II.

Встановимо значення вхідних сигналів wX в «1101100101001011001010100100100101111111111011010001110» та перевіримо за ГСАЗ (на діаграмі ця послідовність іксів зображена навпаки, тобто wX1 це перше значення справа).

Таблиця 4.17 - Апаратурні затрати тестування композиційної структури з лічильником на різних мікросхемах.

Family	Les	Rg	Pins	Mem (bits)
Cyclone IV GX (EP4CGX50CF23C^)	348	29	71	0
Stratix III (EP3SL70F484C2)	225	29	71	0
Arria GX (EP1AGX60CF484C6)	225	29	71	0

Таблиця 4.18 - Апаратурні затрати тестування композиційної структури з лічильником та вбудованою пам'яттю на різних мікросхемах.

Family	Les	Rg	Pins	Mem (bits)
Cyclone IV GX (EP4CGX50CF23C^)	248	18	71	1740
Stratix III (EP3SL70F484C2)	200	18	71	1740
Arria GX (EP1AGX60CF484C6)	200	18	71	1740



Рисунок 4.28 - Апаратурні затрати на різних структурах та мікросхемах.

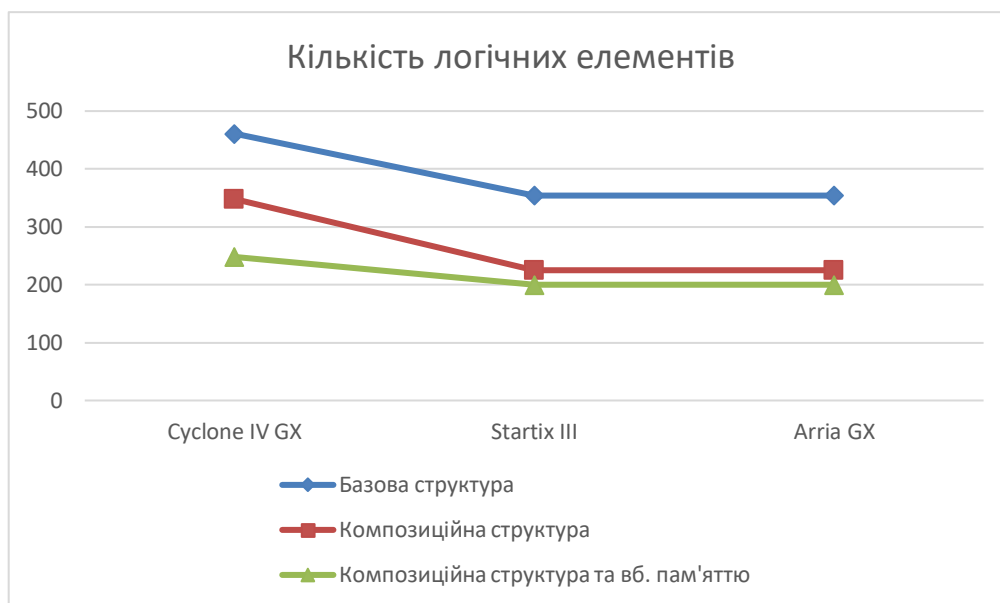


Рисунок 4.29 – Затрати логічних елементів на різних структурах та мікросхемах.



Рисунок 4.30 – Затрати регістрів на різних структурах та мікросхемах.

Стосовно апаратних витрат із графіків можна зробити ті ж самі висновки, що і до попереднього алгоритму.

4.3.2 Порівняння часових характеристик

У таблицях 4.19, 4.20 та 4.21 наведені часові характеристики на різних мікросхемах для різних структур та для більш наочного порівняння на графіках на рисунках 4.31, 4.32, 4.33 наведені дані з цих таблиць.

Таблиця 4.19 - Часові характеристики при тестуванні базової структури на різних мікросхемах.

Family	Tsu(ns)	Tco(ns)	Tpd(ns)
Cyclone IV GX (EP4CGX50CF23C^)	7,358	6,405	11,118
Startix III (EP3SL70F484C2)	8,216	6,167	10,541
Arria GX (EP1AGX60CF484C6)	8,843	6,422	10,417

Таблиця 4.20 - Часові характеристики при тестуванні композиційної структури на різних мікросхемах.

Family	Tsu(ns)	Tco(ns)	Tpd(ns)
Cyclone IV GX (EP4CGX50CF23C^)	14,84	11,22	19,536
Startix III (EP3SL70F484C2)	14,9	10,01	17,811
Arria GX (EP1AGX60CF484C6)	15,189	10,7	17,953

Таблиця 4.21 - Часові характеристики при тестуванні композиційної структури з вбудованою пам'яттю на різних мікросхемах.

Family	Tsu(ns)	Tco(ns)	Tpd(ns)
Cyclone IV GX (EP4CGX50CF23C^)	15,261	12,611	21,9
Startix III (EP3SL70F484C2)	15,662	12,03	20,18
Arria GX (EP1AGX60CF484C6)	16,123	11,9	19,261

Для наочності на графіках зобразимо часові затримки структур на різних мікросхемах.

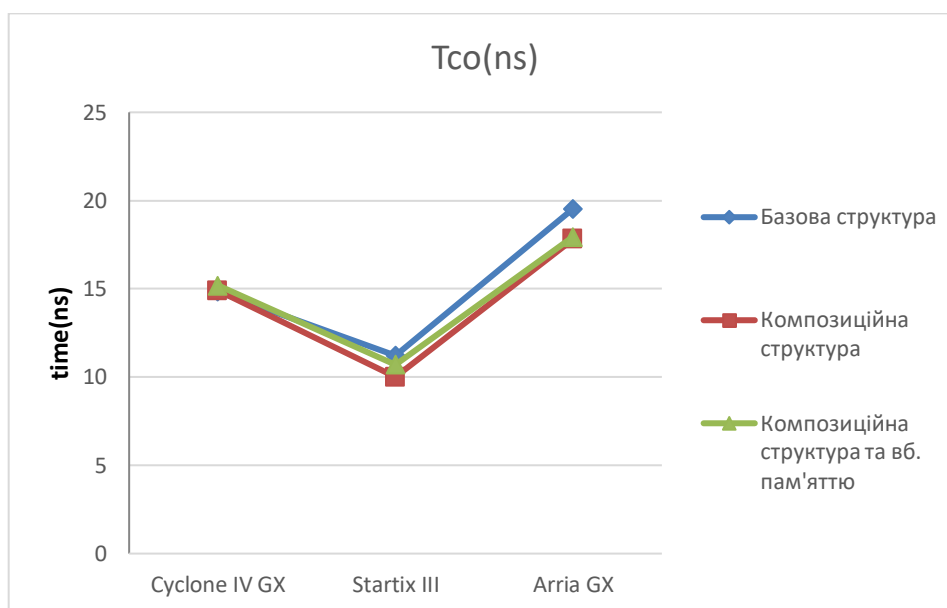


Рисунок 4.31 – Затримка t_{co} при тестуванні різних структур автомату який працює за третьою ГСА на різних мікросхемах



Рисунок 4.32 – Затримка t_{pd} при тестуванні різних структур автомату який працює за третьою ГСА на різних мікросхемах

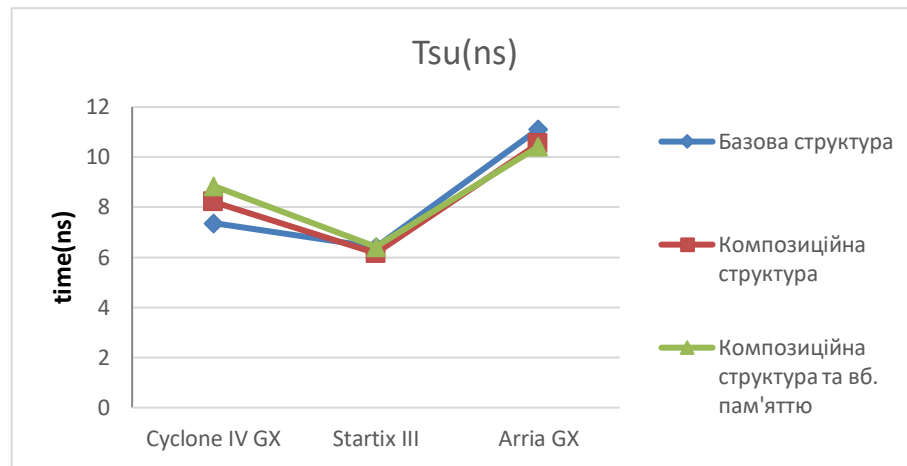


Рисунок 4.33 – Затримка tsu при тестуванні різних структур автомата який працює за третьою ГСА на різних мікросхемах

З графіків можна в цілому зробити ті ж самі висновки, що до попереднього алгоритму. Поки що можна сказати, що композиційні структури виграють перед базовою, не значно, але виграють.

4.3.3 Топології мікросхем

На рисунках 4.34, 4.35, 4.36 можна бачити топології мікросхем на яких проводилися тестування. На них можна побачити різні структури автомата який працює за третьою ГСА.

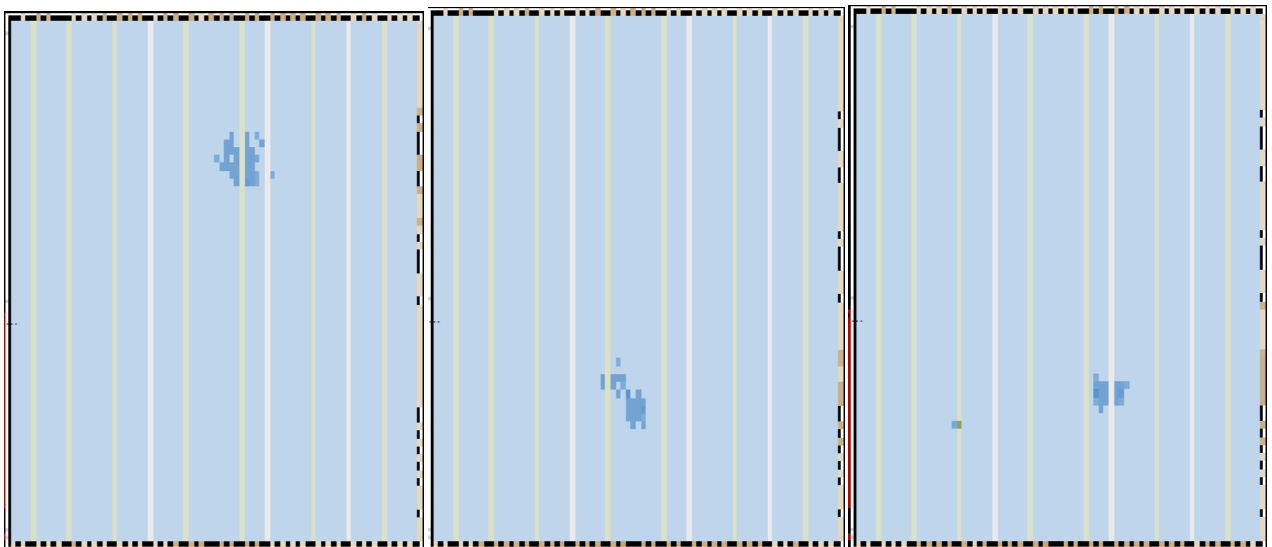


Рисунок 4.34 – Розміщення базової структури (зліва), композиційної структури (посередині) та композиційної структури з вбудованою пам'яттю (справа) на мікросхемі Cyclone IV GX.

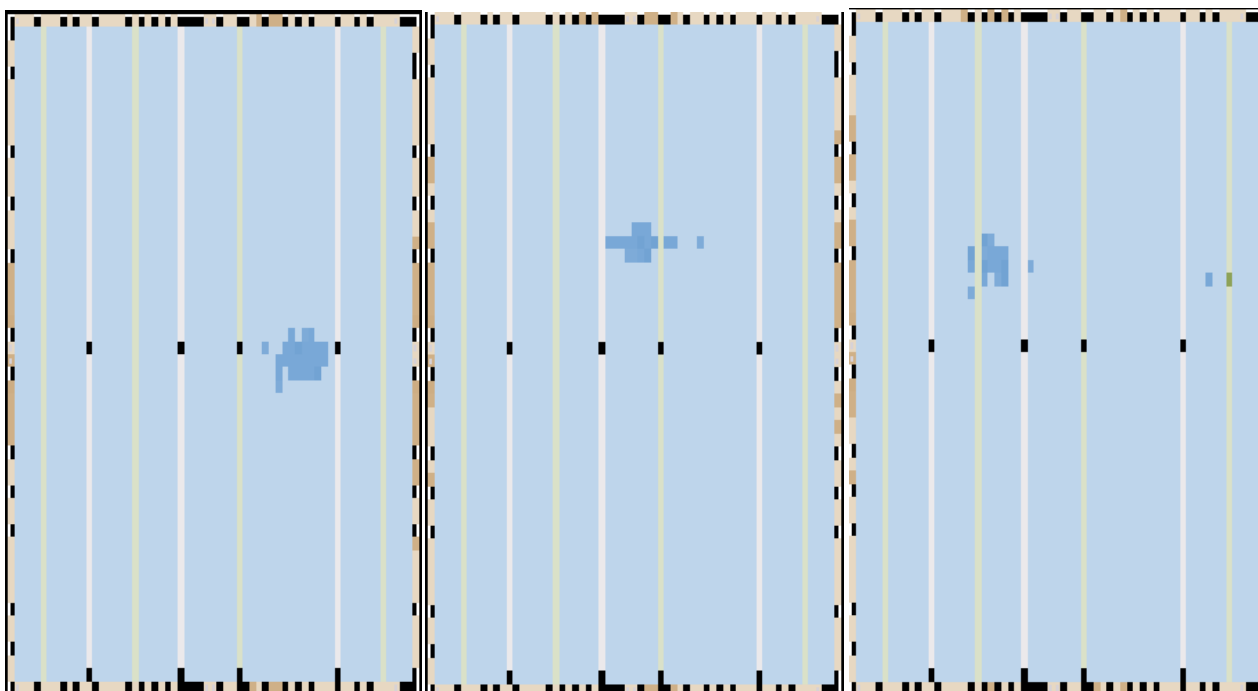


Рисунок 4.35 – Розміщення базової структури (зліва), композиційної структури (посередині) та композиційної структури з вбудованою пам'яттю (справа) на мікросхемі Startix III.

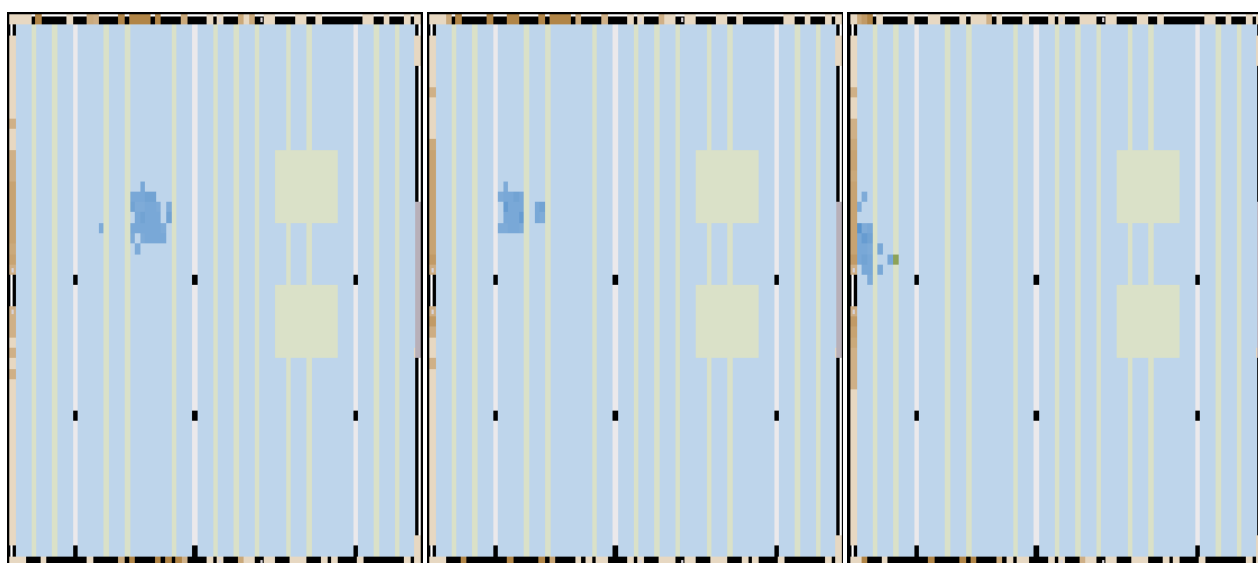


Рисунок 4.36 – Розміщення базової структури (зліва), композиційної структури (посередині) та композиційної структури з вбудованою пам'яттю (справа) на мікросхемі Arria GX.

За результатами тестувань можна прослідкувати зменшення логічних елементів на схемі залежно від структури.

4.4 Тестування розроблених паралельних схем різних структур які працюють за першим та другим алгоритмом

Для перевірки працездатності розроблених паралельних схем різних структур які працюють за першим та другим алгоритмом (рисунки 4.37, 4.38 та 4.39) запусимо симуляцію часових діаграм в Quartus II.

Нижче наведені фрагменти часових діаграм паралельних схем.

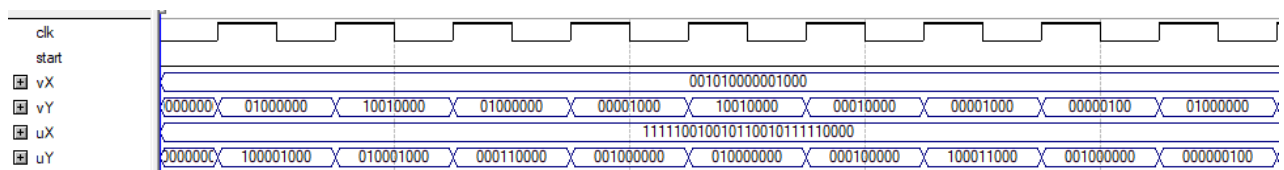


Рисунок 4.37 – Часові діаграми паралельної схеми базових структур

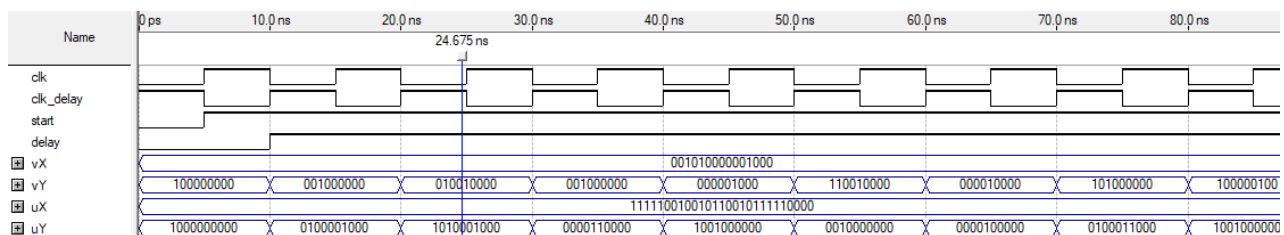


Рисунок 4.38 – Часові діаграми паралельної схеми композиційних автоматів 1 та 2 алгоритмів

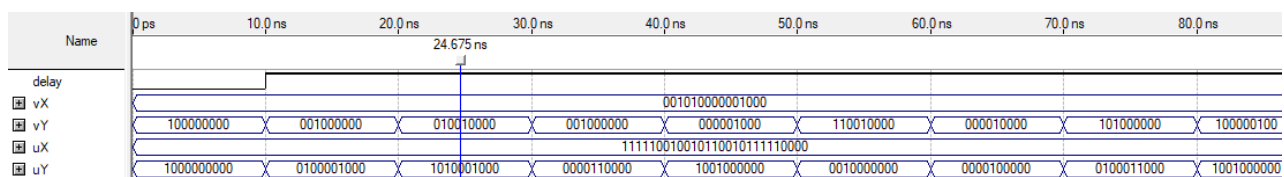


Рисунок 4.39 – Часові діаграми паралельної схеми структур з вбудованою пам'яттю

Із діаграм можна зробити висновок, що зібрані схеми працюють правильно, оскільки послідовності вихідних сигналів схем відповідають першій та другій ГСА.

Також з діаграм можна зробити висновки відносно швидкості роботи паралельної схеми на якій працюють три алгоритми одночасно порівняно з тим якщо запускати ці алгоритми послідовно. На графіку наведено порівняння часу.

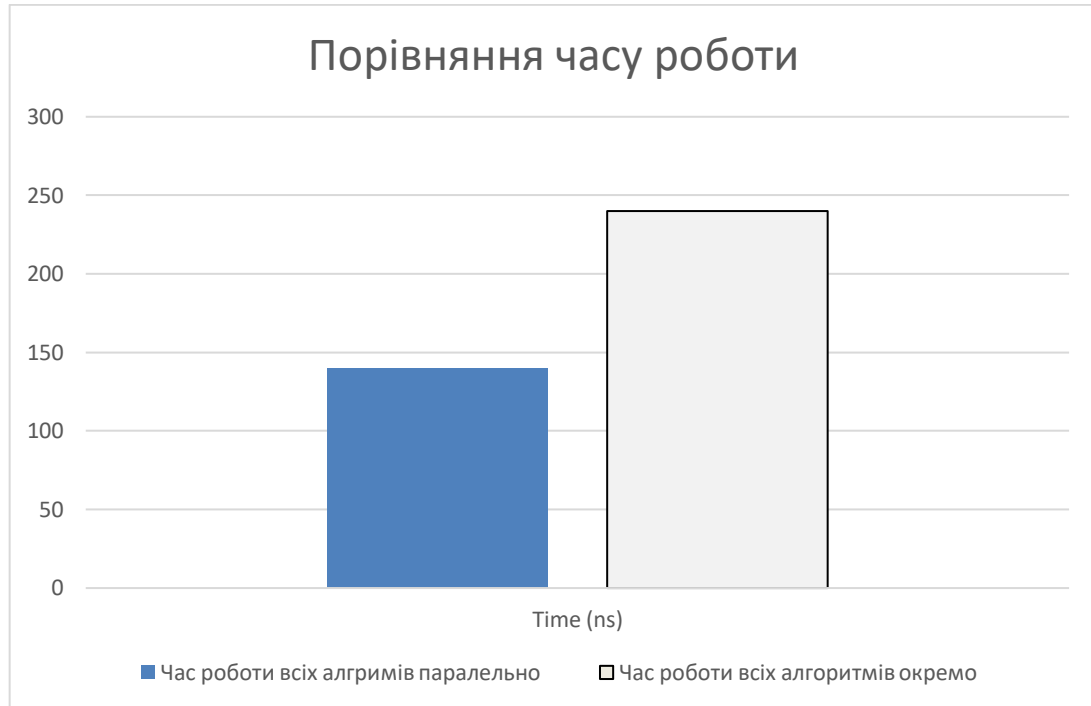


Рисунок 4.40 – Порівняння часу роботи

На графіку часу роботи можна побачити що якщо запускати схеми першого та другого алгоритму окремо та послідовно то загалом час їх виконання буде значно вище часу їх виконання при запуску їх паралельно на одному кристалі. Якщо запускати схеми алгоритмів паралельно то час роботи буде дорівнювати часу найдовшого алгоритму з тих що запускаються паралельно тобто в нашому випадку другому алгоритму.

4.4.1 Тестування паралельної схеми першого та другого алгоритму на різних мікросхемах

У таблицях 4.22, 4.23 та 4.24 наведені апаратні затрати на різних мікросхемах для різних структур та для більш наочного порівняння на графіках на рисунках 4.41, 4.42 та 4.43 наведені дані з цих таблиць.

Таблиця 4.22 - Апаратурні затрати тестування паралельної схеми базової структури першого та другого алгоритму на різних мікросхемах.

Family	Les	Rg	Pins	Mem (bits)
Cyclone IV GX (EP4CGX50CF23C^)	345	12	61	0
Startix III (EP3SL70F484C2)	247	12	61	0
Arria GX (EP1AGX60CF484C6)	247	12	61	0

Таблиця 4.23 - Апаратурні затрати тестування паралельної схеми композиційної структури з лічильником першого та другого алгоритму на різних мікросхемах.

Family	Les	Rg	Pins	Mem (bits)
Cyclone IV GX (EP4CGX50CF23C^)	222	47	77	0
Startix III (EP3SL70F484C2)	154	47	77	0
Arria GX (EP1AGX60CF484C6)	154	47	77	0

Таблиця 4.24 - Апаратурні затрати тестування паралельної схеми композиційної структури з лічильником та вбудованою пам'яттю першого та другого алгоритму на різних мікросхемах.

Family	Les	Rg	Pins	Mem (bits)
Cyclone IV GX (EP4CGX50CF23C^)	176	28	77	1100
Startix III (EP3SL70F484C2)	154	28	77	1100
Arria GX (EP1AGX60CF484C6)	154	28	77	1100

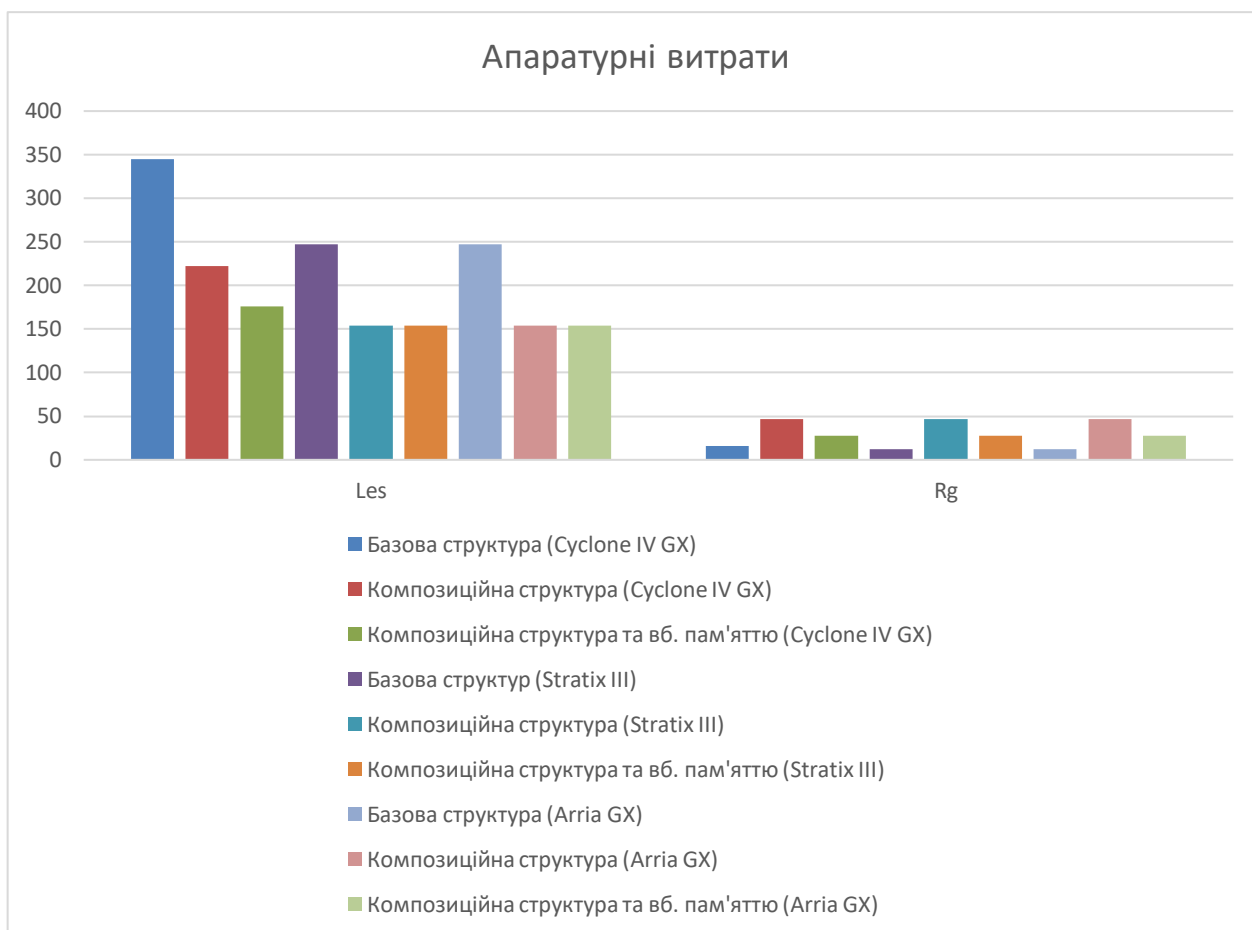


Рисунок 4.41 - Апаратурні затрати на різних структурах та мікросхемах.

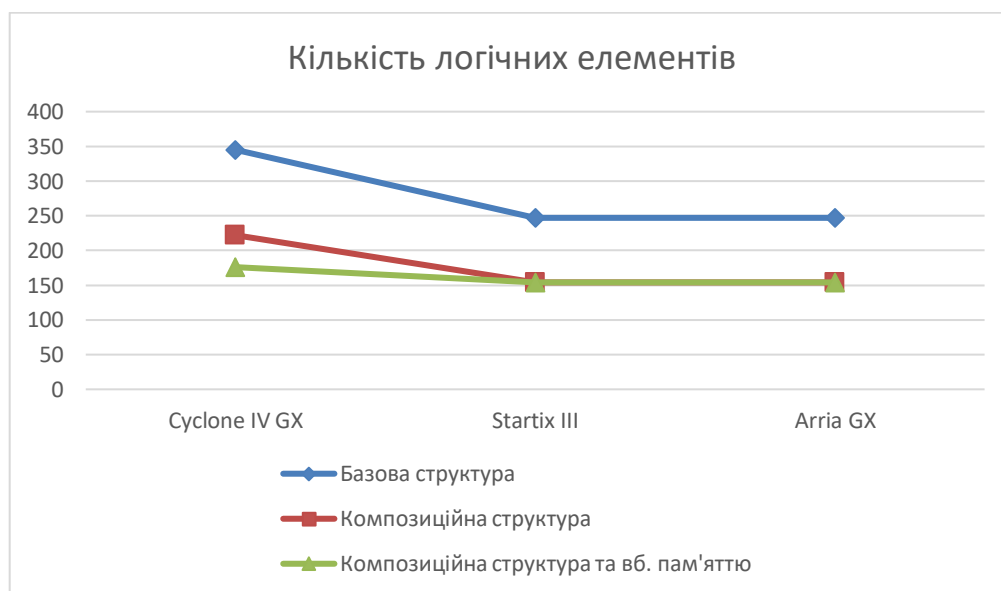


Рисунок 4.42 – Затрати логічних елементів на різних структурах та мікросхемах.

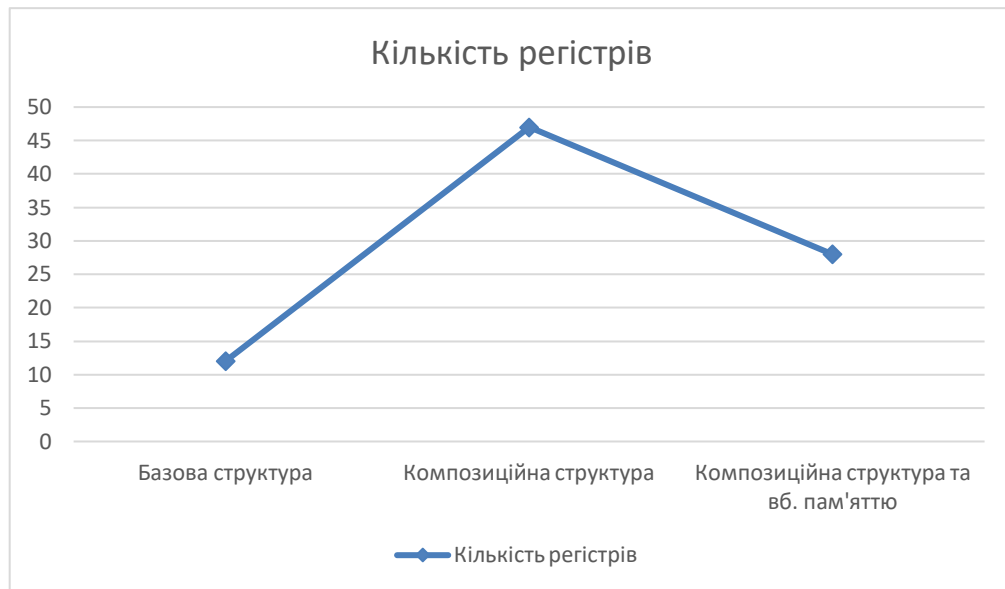


Рисунок 4.43 – Затрати регістрів на різних структурах та мікросхемах.

Стосовно апаратних витрат із графіків можна зробити ті ж самі висновки, що і до попереднього алгоритму.

4.4.2 Порівняння часових характеристик

У таблицях 4.25, 4.26 та 4.27 наведені часові характеристики на різних мікросхемах для різних структур та для більш наочного порівняння на графіках на рисунках 4.44, 4.45 та 4.46 наведені дані з цих таблиць.

Таблиця 4.25 - Часові характеристики тестування паралельної схеми базової структури першого та другого алгоритму на різних мікросхемах.

Family	Tsu(ns)	Tco(ns)	Tpd(ns)
Cyclone IV GX (EP4CGX50CF23C^)	6,676	4,79	9,116
Startix III (EP3SL70F484C2)	6,845	5,891	10,006
Arria GX (EP1AGX60CF484C6)	6,664	5,021	6,786

Таблиця 4.26 - Часові характеристики тестування паралельної схеми композиційної структури з лічильником першого та другого алгоритму на різних мікросхемах.

Family	Tsu(ns)	Tco(ns)	Tpd(ns)
Cyclone IV GX (EP4CGX50CF23C^)	13,845	9,7	18,323
Startix III (EP3SL70F484C2)	13,011	8,692	15,011
Arria GX (EP1AGX60CF484C6)	14,468	9,321	15,288

Таблиця 4.27 - Часові характеристики тестування паралельної схеми композиційної структури з лічильником та вбудованою пам'яттю першого та другого алгоритму на різних мікросхемах.

Family	Tsu(ns)	Tco(ns)	Tpd(ns)
Cyclone IV GX (EP4CGX50CF23C^)	14,114	10,8	18,324
Startix III (EP3SL70F484C2)	14,473	11,4	17,611
Arria GX (EP1AGX60CF484C6)	15,021	11,534	17,8

Для наочності на графіках зобразимо часові затримки структур на різних мікросхемах.

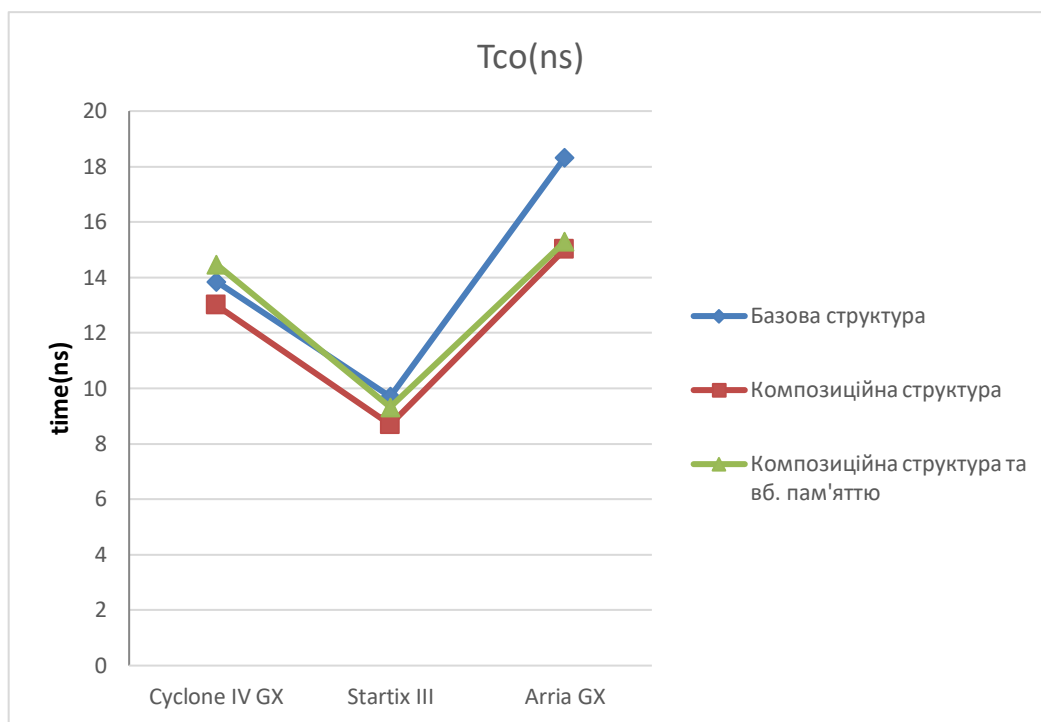


Рисунок 4.44 – Затримка t_{co} при тестуванні різних структур паралельної схеми яка працює за першою та другою ГСА на різних мікросхемах

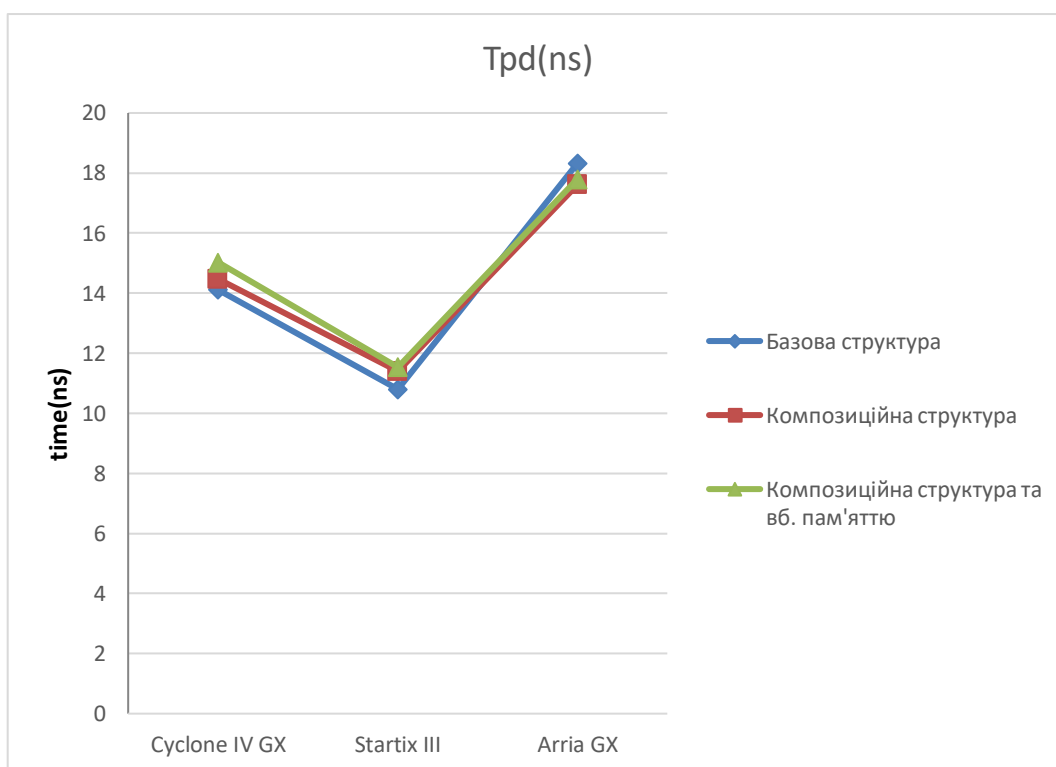


Рисунок 4.45 – Затримка t_{pd} при тестуванні різних структур паралельної схеми яка працює за першою та другою ГСА на різних мікросхемах

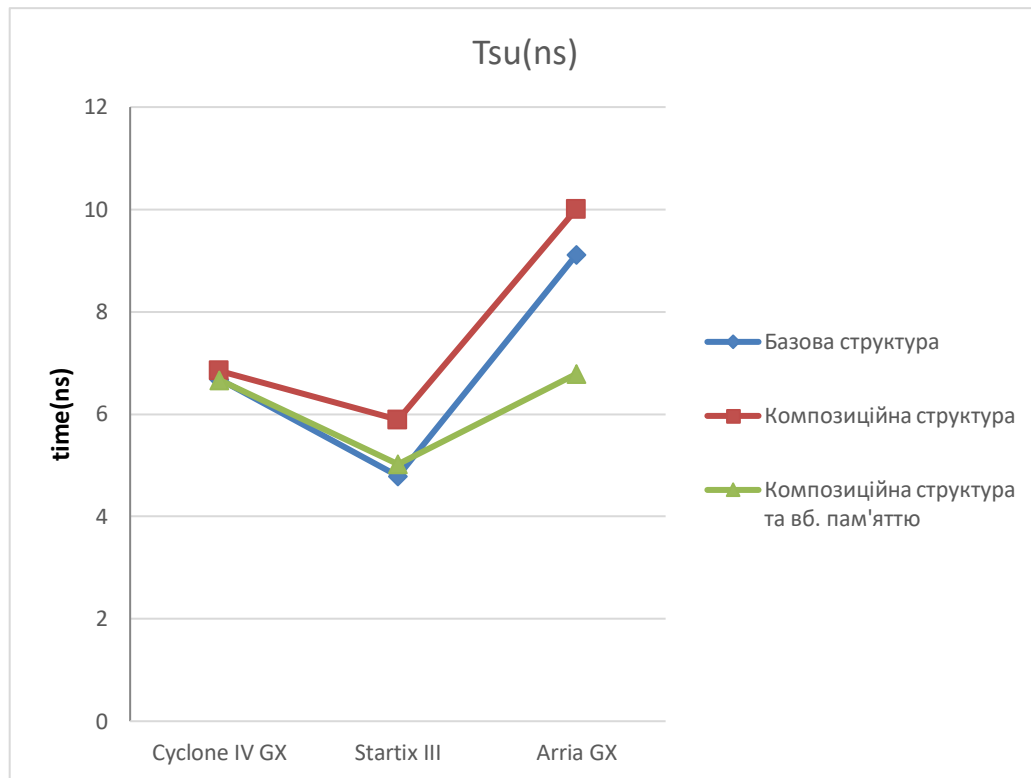


Рисунок 4.46 – Затримка tsu при тестуванні різних структур паралельної схеми яка працює за першою та другою ГСА на різних мікросхемах

Із графіків роботи структур паралельної схеми двох алгоритмів можна зробити висновки, що tso для композиційних структур знову в цілому менший порівняно з базовою особливо на мікросхемі Arria, але затримка tpd у всіх структур на всіх мікросхемах майже не відрізняється. Отже композиційні структури включаючи попередні результати все ще виграють порівняно з базовою.

4.4.3 Топології мікросхем

На рисунках 4.47, 4.48, 4.49 можна бачити топології мікросхем на яких проводилися тестування. На них можна побачити паралельні схеми різних структур які працюють за першим та другим алгоритмом.

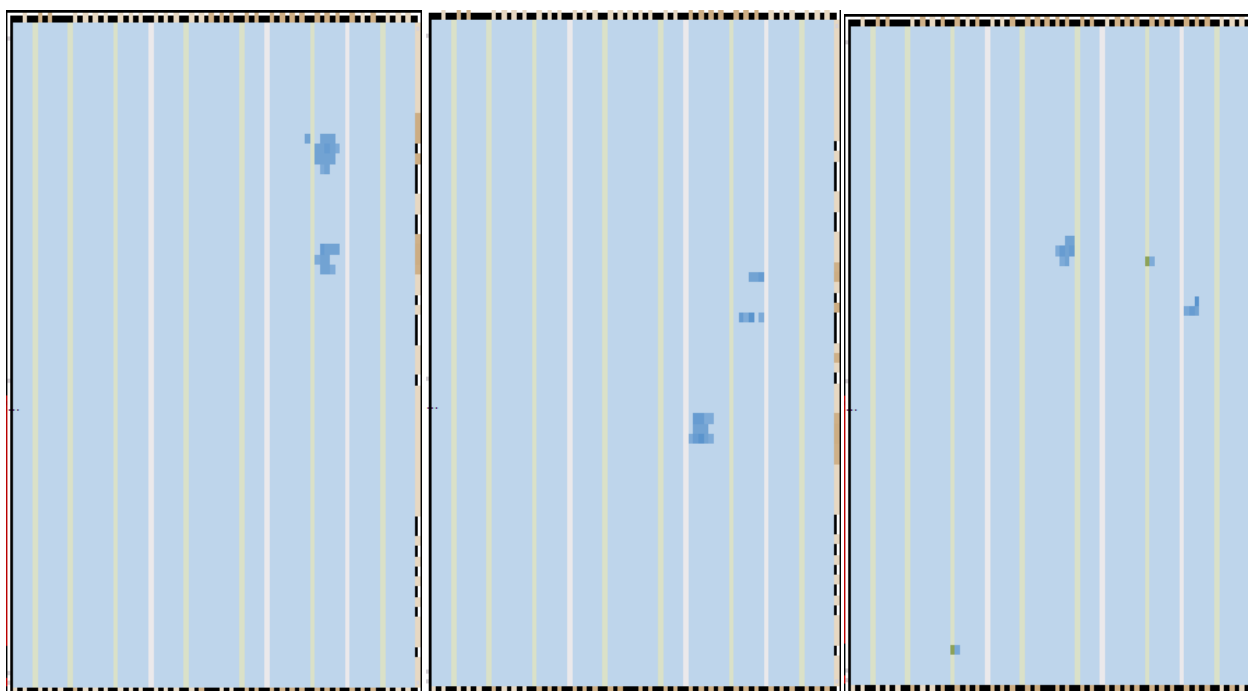


Рисунок 4.47 – Розміщення базової структури (зліва), композиційної структури (посередині) та композиційної структури з вбудованою пам'яттю (справа) на мікросхемі Cyclone IV GX.

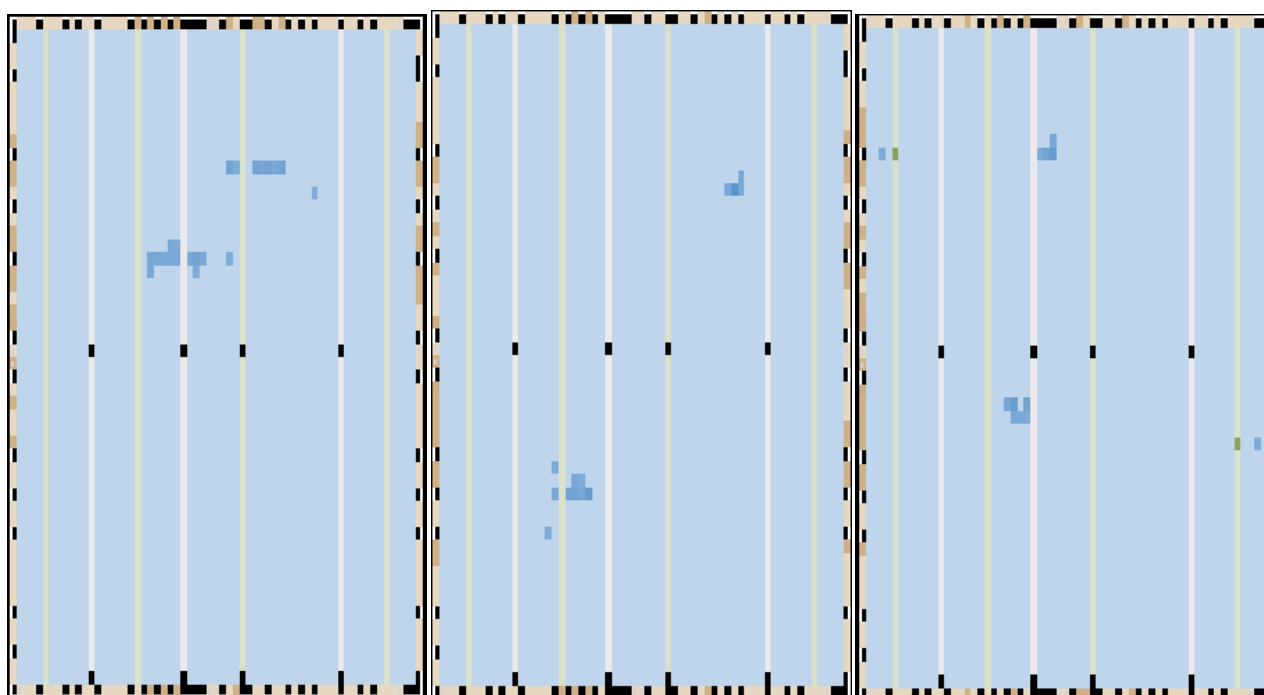


Рисунок 4.48 – Розміщення базової структури (зліва), композиційної структури (посередині) та композиційної структури з вбудованою пам'яттю (справа) на мікросхемі Startix III.

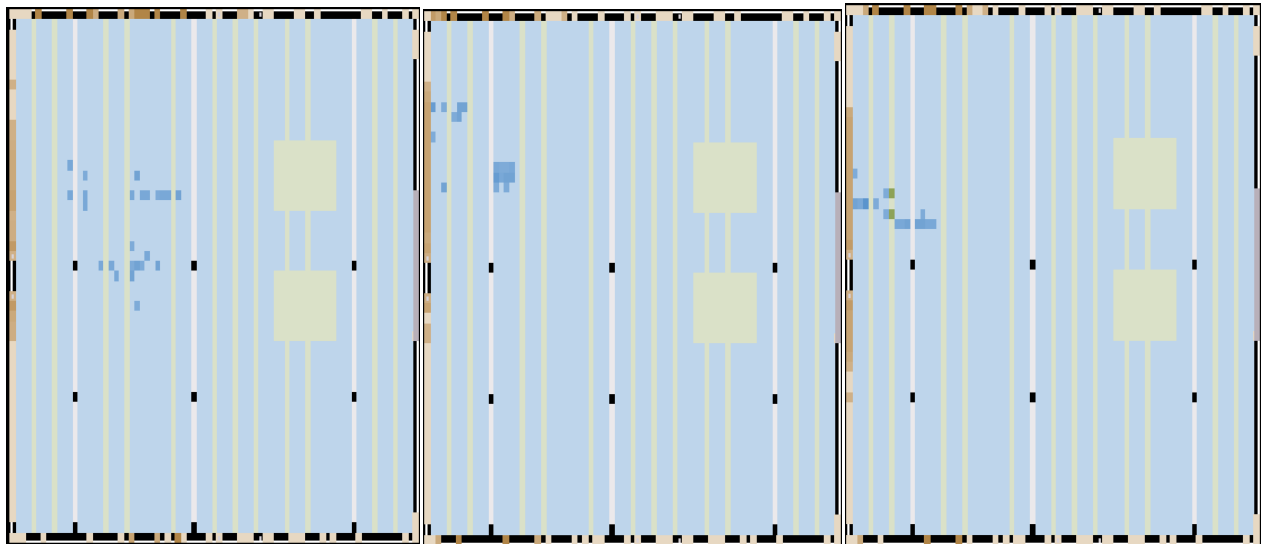


Рисунок 4.49 – Розміщення базової структури (зліва), композиційної структури (посередині) та композиційної структури з вбудованою пам'яттю (справа) на мікросхемі Arria GX.

За результатами тестувань можна прослідкувати зменшення логічних елементів на схемі залежно від структури.

4.5 Тестування розроблених паралельних схем різних структур які працюють за першим, другим та третім алгоритмом

Для перевірки працездатності розроблених паралельних схем різних структур які працюють за першим, другим та третім алгоритмом (рисунок 4.50, 4.51 та 4.52) запусимо симуляцію часових діаграм в Quartus II.

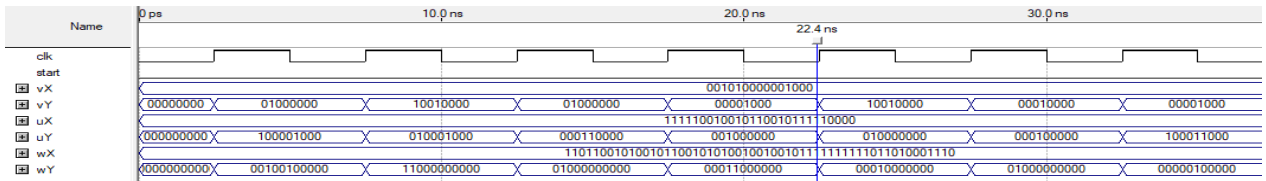


Рисунок 4.50 – Часові діаграми паралельної схеми базових структур

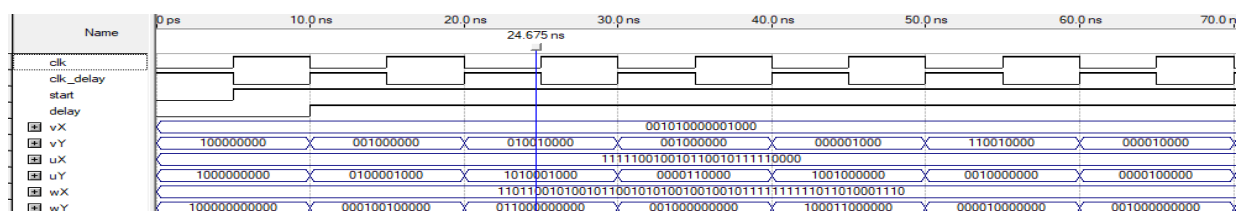


Рисунок 4.51 – Часові діаграми паралельної схеми композиційних автоматів 1, 2 та 3 алгоритму



Рисунок 4.52 – Часові діаграми паралельної схеми структур з вбудованою пам'яттю

Із діаграм можна зробити висновок, що зібрані схеми працюють правильно, оскільки послідовності вихідних сигналів схем відповідають першій, другій та третій ГСА.

Також з діаграм можна зробити висновки відносно швидкості роботи паралельної схеми на якій працюють три алгоритми одночасно порівняно з тим якщо запускати ці алгоритми послідовно. На графіку наведено порівняння часу.

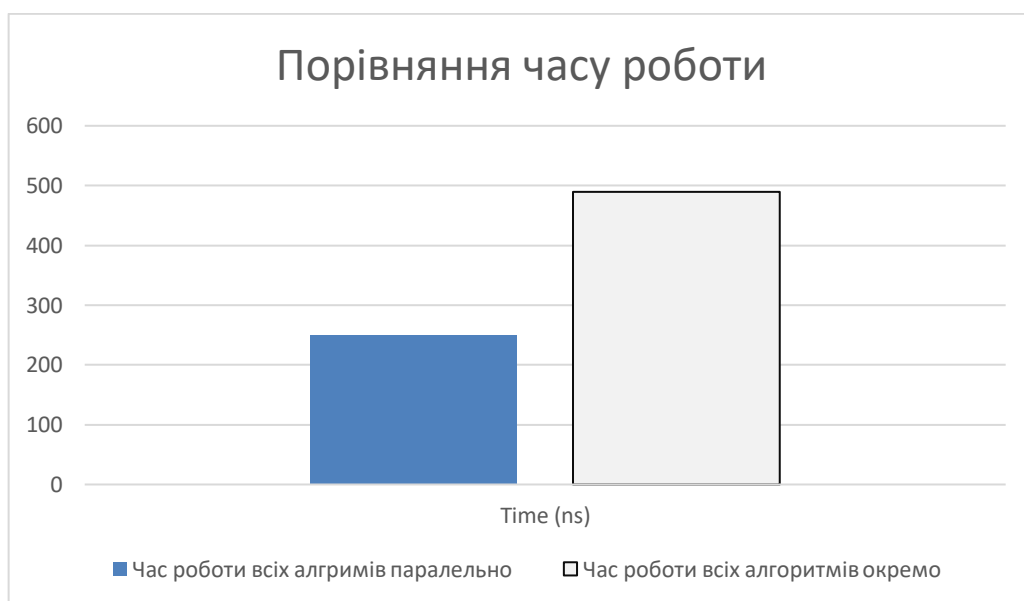


Рисунок 4.53 – Порівняння часу роботи

На графіку часу роботи можна побачити що якщо запускати схеми першого, другого та третього алгоритму окремо та послідовно то загалом час їх виконання буде значно вище часу їх виконання при запуску їх паралельно на одному кристалі. Якщо запускати схеми алгоритмів паралельно то час роботи буде дорівнювати часу найдовшого алгоритму з тих що запускаються паралельно тобто в нашому випадку третьому алгоритму.

4.5.1 Тестування паралельної схеми першого, другого та третього алгоритму на різних мікросхемах

У таблицях 4.28, 4.29 та 4.30 наведені апаратні затрати на різних мікросхемах для різних структур та для більш наочного порівняння на графіках на рисунках 4.54, 4.55 та 4.56 наведені дані з цих таблиць.

Таблиця 4.28 - Апаратні затрати тестування паралельної схеми базової структури першого, другого та третього алгоритму на різних мікросхемах.

Family	Les	Rg	Pins	Mem (bits)
Cyclone IV GX (EP4CGX50CF23C [^])	808	20	127	0
Startix III (EP3SL70F484C2)	618	20	127	0
Arria GX (EP1AGX60CF484C6)	618	20	127	0

Таблиця 4.29 - Апаратні затрати тестування паралельної схеми композиційної структури з лічильником першого, другого та третього алгоритму на різних мікросхемах.

Family	Les	Rg	Pins	Mem (bits)
Cyclone IV GX (EP4CGX50CF23C [^])	571	76	132	0
Startix III (EP3SL70F484C2)	366	76	132	0
Arria GX (EP1AGX60CF484C6)	366	76	132	0

Таблиця 4.30 - Апаратурні затрати тестування паралельної схеми композиційної структури з лічильником та вбудованою пам'яттю першого, другого та третього алгоритму на різних мікросхемах.

Family	Les	Rg	Pins	Mem (bits)
Cyclone IV GX (EP4CGX50CF23C^)	422	46	132	2840
Stratix III (EP3SL70F484C2)	340	46	132	2840
Arria GX (EP1AGX60CF484C6)	340	46	132	2840

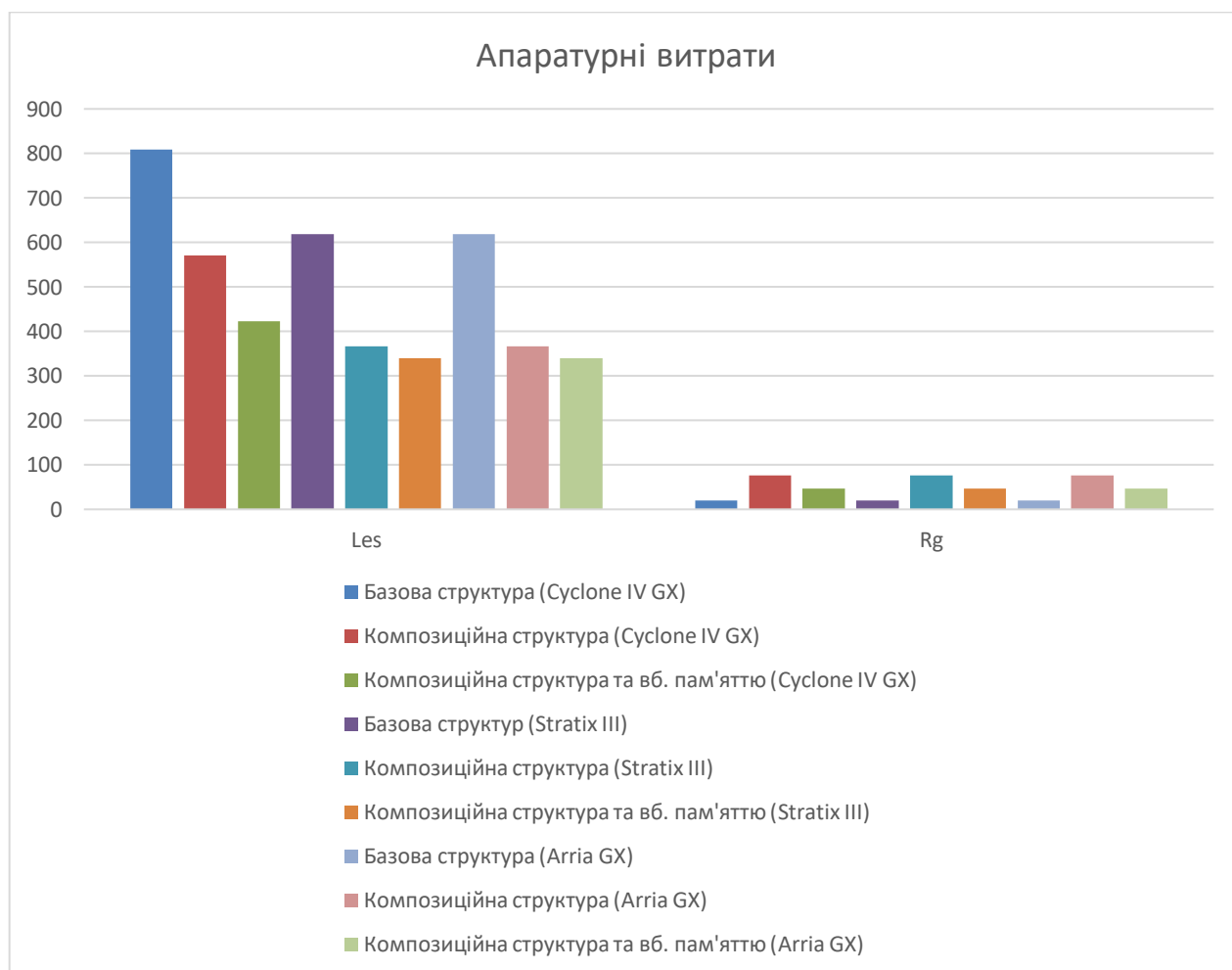


Рисунок 4.54 - Апаратурні затрати на різних структурах та мікросхемах

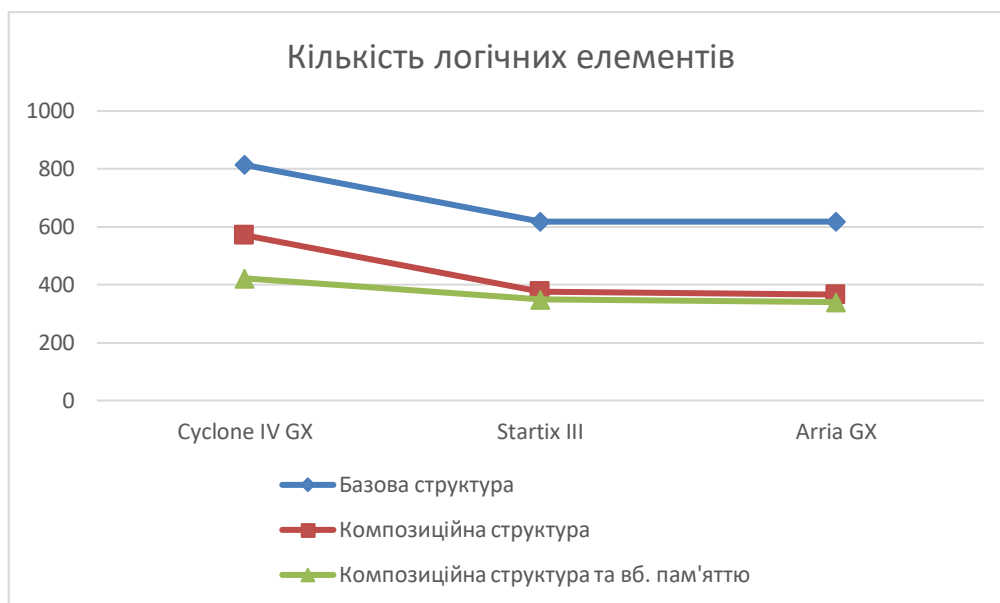


Рисунок 4.55 – Затрати логічних елементів на різних структурах та мікросхемах



Рисунок 4.56 – Затрати регістрів на різних структурах та мікросхемах

Стосовно апаратних витрат із графіків можна зробити ті ж самі висновки, що і до попереднього алгоритму.

4.5.2 Порівняння часових характеристик

У таблицях 4.31, 4.32 та 4.33 наведені часові характеристики на різних мікросхемах для різних структур та для більш наочного порівняння на графіках на рисунках 4.57, 4.58 та 4.59 наведені дані з цих таблиць.

Таблиця 4.31 - Часові характеристики при тестуванні паралельної схеми базової структури першого, другого та третього алгоритму на різних мікросхемах.

Family	Tsu(ns)	Tco(ns)	Tpd(ns)
Cyclone IV GX (EP4CGX50CF23C^)	6,871	6,361	11,478
Startix III (EP3SL70F484C2)	8,08	6,139	11,041
Arria GX (EP1AGX60CF484C6)	7,887	6,821	10,472

Таблиця 4.32 - Часові характеристики тестування паралельної схеми композиційної структури з лічильником першого, другого та третього алгоритму на різних мікросхемах.

Family	Tsu(ns)	Tco(ns)	Tpd(ns)
Cyclone IV GX (EP4CGX50CF23C^)	16,232	12,313	20,48
Startix III (EP3SL70F484C2)	13,948	10,487	17,644
Arria GX (EP1AGX60CF484C6)	14,342	10,122	17,929

Таблиця 4.33 - Часові характеристики при тестуванні паралельної схеми композиційної структури з лічильником та вбудованою пам'яттю першого, другого та третього алгоритму на різних мікросхемах.

Family	Tsu(ns)	Tco(ns)	Tpd(ns)
Cyclone IV GX (EP4CGX50CF23C^)	16,571	12,9	22,67
Startix III (EP3SL70F484C2)	15,14	12,383	20,146
Arria GX (EP1AGX60CF484C6)	15,046	12,5	19,599

Для наочності зобразимо на графіках часові затримки структур на різних мікросхемах.

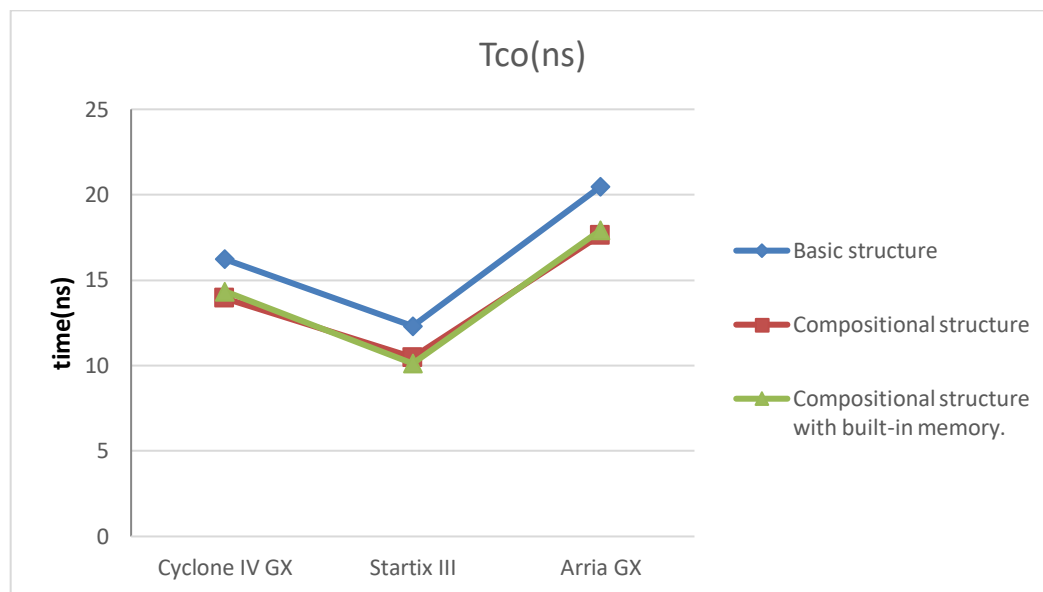


Рисунок 4.57 – Затримка t_{pd} при тестуванні різних структур паралельної схеми, яка працює за першою, другою та третьою ГСА на різних мікросхемах

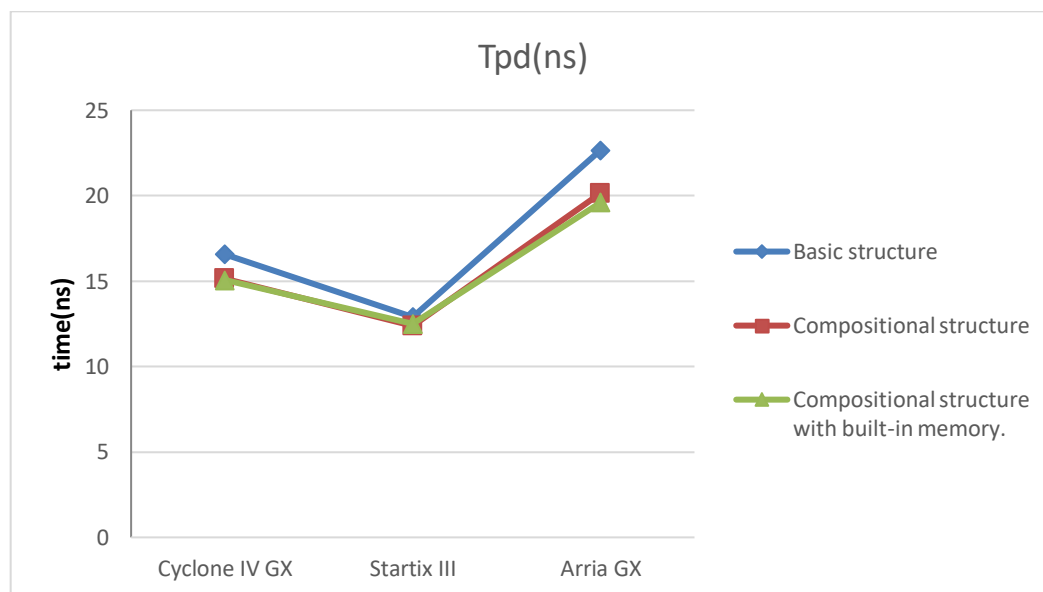


Рисунок 4.58 – Затримка t_{su} при тестуванні різних структур паралельної схеми, яка працює за першою, другою та третьою ГСА на різних мікросхемах

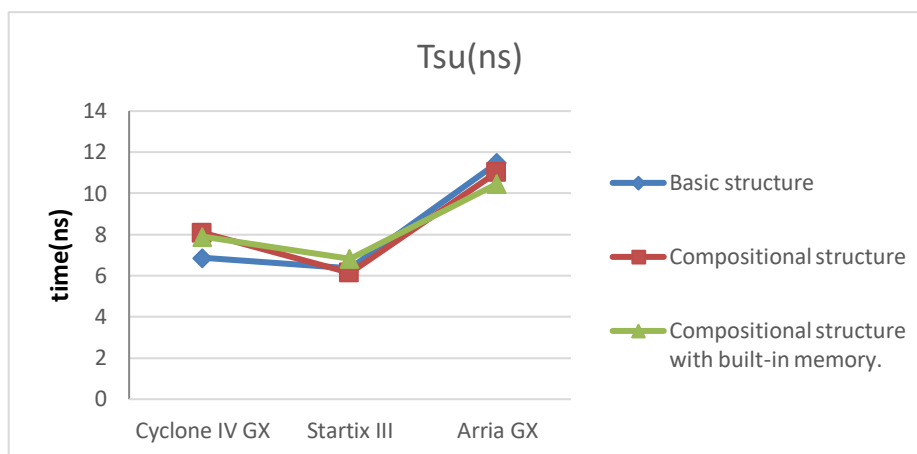


Рисунок 4.59 – Затримка tso при тестуванні різних структур паралельної схеми, яка працює за першою, другою та третьою ГСА на різних мікросхемах

Паралельна схема трьох алгоритмів очевидно враховуючи попередні результати доводить те, що композиційні структури все ж є кращими в плані часу за базову структуру оскільки затримки tpd та tso в них нижче.

4.5.3 Топології мікросхем

На рисунках 4.60, 4.61, 4.62 можна бачити топології мікросхем на яких проводилися тестування. На них можна побачити паралельні схеми різних структур які працюють за першим, другим та третім алгоритмом.

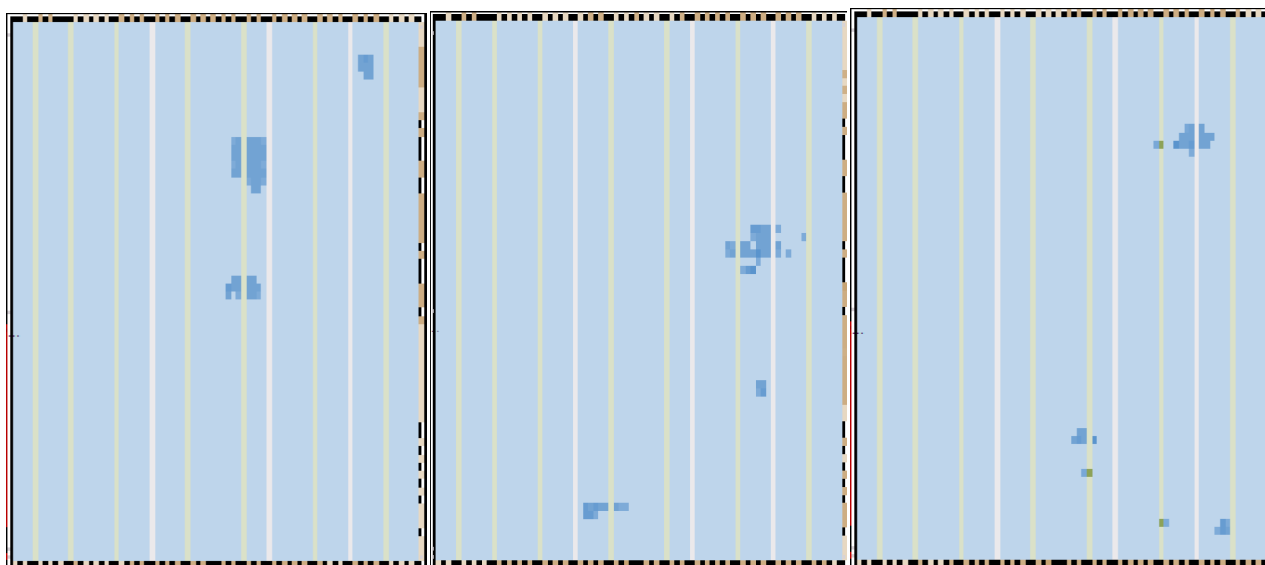


Рисунок 4.60 – Розміщення базової структури (зліва), композиційної структури (посередині) та композиційної структури з вбудованою пам'яттю (справа) на мікросхемі Cyclone IV GX.

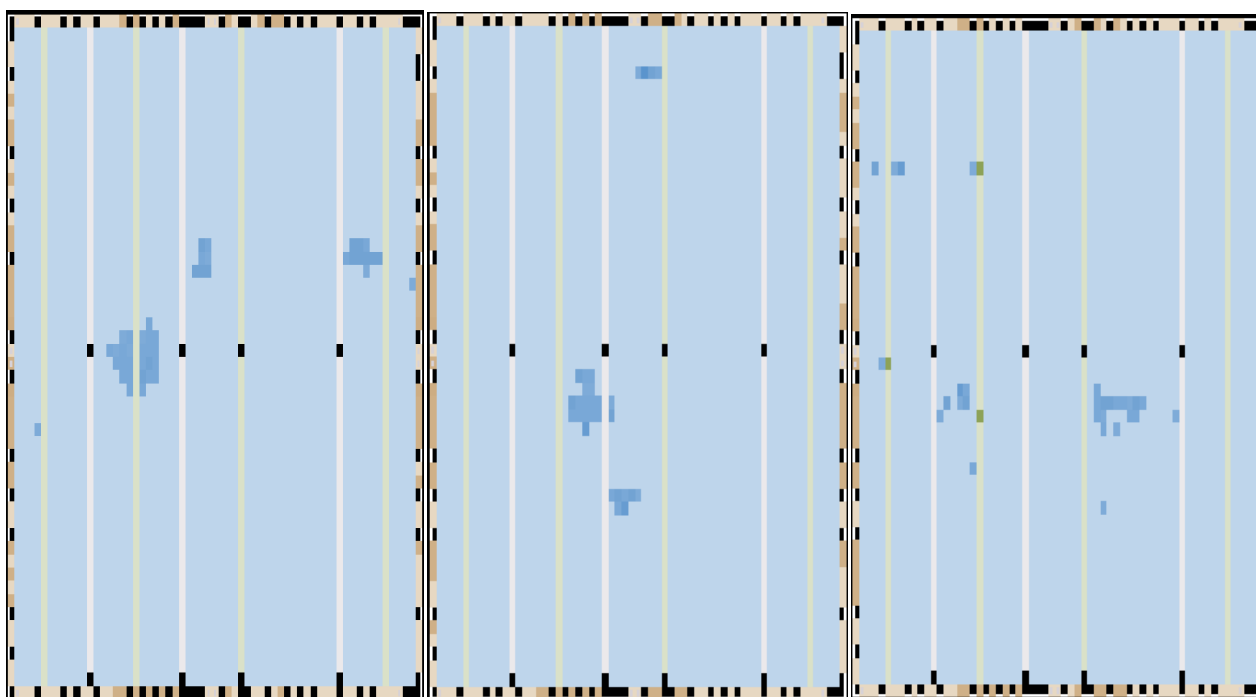


Рисунок 4.61 – Розміщення базової структури (зліва), композиційної структури (посередині) та композиційної структури з вбудованою пам'яттю (справа) на мікросхемі Startix III.

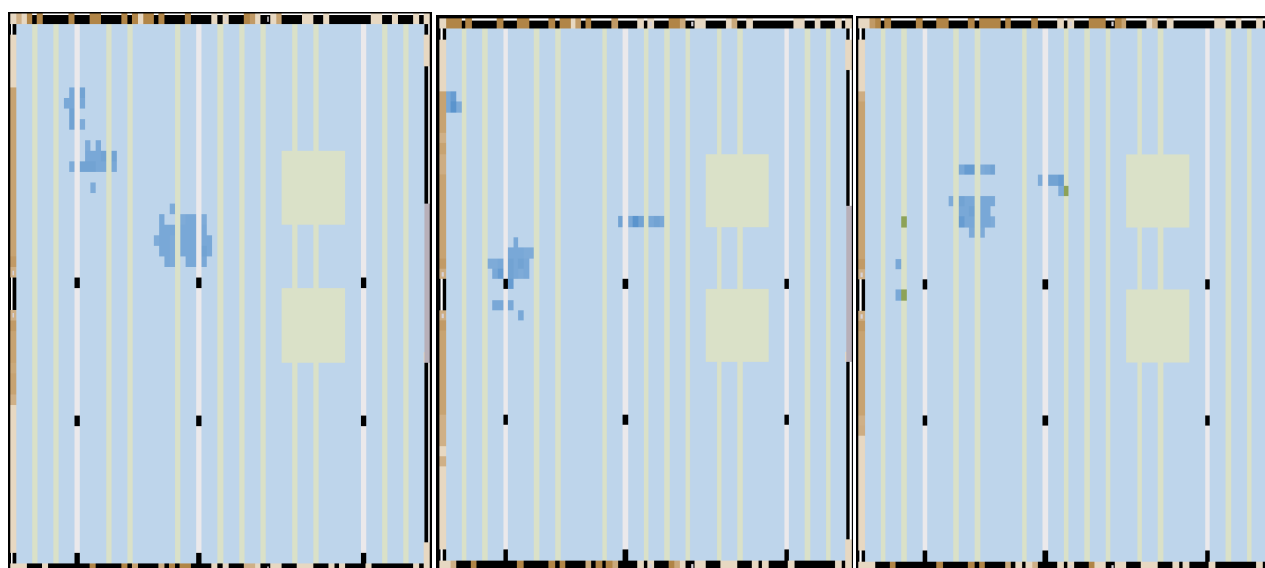


Рисунок 4.62 – Розміщення базової структури (зліва), композиційної структури (посередині) та композиційної структури з вбудованою пам'яттю (справа) на мікросхемі Arria GX.

За результатами тестувань можна прослідкувати зменшення логічних елементів на схемі залежно від структури

ВИСНОВКИ

За підсумками результатів експериментального дослідження можна зробити практично корисні висновки.

По-перше з точки зору економії апаратних ресурсів можна побачити зменшення кількості логічних елементів залежно від структури. Порівняно з базовою структурою, кількість логічних елементів у композиційній структурі зменшилася за рахунок введення лічильника, і як результат, скорочення формул для формування наступного стану автомата.

За рахунок використання вбудованої пам'яті при імплементації композиційного автомата також зменшується кількість логічних елементів та регістрів. Очевидно, такий перерозподіл ресурсів є дуже вигідним, оскільки звільнені логічні елементи потенційно можуть бути задіяні для імплементації інших функціональних вузлів загальної системи.

Отже, за результатами тестувань можна зробити висновок, що композиційна структура пристрою керування із вбудованою пам'яттю є найкращою серед інших структур оскільки займає найменше логічних елементів і регістрів.

За результатами аналізу часу виконання алгоритмів, за всіма експериментами можна помітити, що головні затримки t_{co} та t_{pd} у композиційних структурах в цілому менші, ніж у базовій структурі. Найбільше це видно на мікросхемі Arria. Це свідчить про перевагу композиційних структур перед базовою.

Отже, за результатами заміру часових характеристик можна зробити висновок, що серед усіх мікросхем, на яких проводилися тестування, найкращою з точки зору часу буде Arria. Часові затримки t_{co} та t_{pd} майже за всіма результатами на цій мікросхемі стабільно нижчі, ніж у інших.

Стосовно топологій, за результатами тестувань можна було прослідкувати зменшення логічних елементів на схемі залежно від

структури. Також можна побачити, що як правило, логічні елементи у всіх топологіях, окрім винятків, в цілому розташовані щільно поблизу один одного, що добре з точки зору швидкості роботи.

У ході роботи було доведено, що паралельний запуск декількох пристроїв керування на одному кристалі можливий. Завдяки цьому можна значно зменшувати габарити, вагу та енергоспоживання бортового обчислювального пристрою.

За результатами тестування паралельної схеми з часових діаграм можна зробити висновки, що час виконання схем на одному кристалі дорівнює часу роботи алгоритму, час виконання якого найдовший. Таким чином, загальний час виконання декількох алгоритмів паралельно значно зменшується порівняно з тим, якщо запускати їх всіх послідовно один за одним.

Підсумкові результати експериментальних досліджень дозволяють визначити, що композиційний мікропрограмний пристрій керування доцільно використовувати для вбудованих систем. Паралельна організація роботи декількох незалежних пристроїв керування дозволяє прискорити виконання керуючих алгоритмів, а також розташувати на одному кристалі схеми пристроїв для керування різними блоками складної системи. Таким чином, мета досліджень досягнута і всі поставлені задачі вирішені.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Embedded System Definition. URL: <https://www.heavy.ai/technical-glossary/embedded-systems> (дата звернення: 25.11.2023).
2. Embedded Systems - Overview. URL: https://www.tutorialspoint.com/embedded_systems/es_overview.html (дата звернення: 25.11.2023).
3. Introduction of Embedded Systems | Set-1. URL: <https://www.geeksforgeeks.org/introduction-of-embedded-systems-set-1/#> (дата звернення: 01.12.2023).
4. Moore machine. URL: https://en.wikipedia.org/wiki/Moore_machine (дата звернення: 25.11.2023).
5. Mealy and Moore Machines in TOC. URL: <https://www.geeksforgeeks.org/mealy-and-moore-machines-in-toc/#> (дата звернення: 01.12.2023).
6. Quartus II 7.2. URL: <https://quartus-ii.updatestar.com/> (дата звернення: 25.11.2023).
7. What is an FPGA? URL: <https://www.latticesemi.com/en/What-is-an-FPGA> (дата звернення: 05.12.2023).
8. What is an FPGA? Definition, Types, Programming, and More. URL: <https://www.datacenterknowledge.com/hardware/what-fpga-definition-types-programming-and-more#close-modal> (дата звернення: 25.10.2023).
9. Сайт Altera (Intel). URL: <https://www.intel.com/content/www/us/en/products/programmable.html>.
10. Cyclone® IV GX FPGA. URL: <https://www.intel.com/content/www/us/en/products/details/fpga/cyclone/iv/gx/products.html>. (дата звернення: 25.10.2023).
11. Intel® Stratix® Series FPGAs and SoCs. URL: <https://www.intel.com/content/www/us/en/products/details/fpga/stratix.html>. (дата звернення: 25.10.2023).

12. intel® Arria® Series FPGAs and SoCs. URL: <https://www.intel.com/content/www/us/en/products/details/fpga/arria.html>. (дата звернення: 25.10.2023).

13. Іванець С., Зубань Ю., Казимир В., Литвинов В., Проектування комп'ютерних систем на основі мікросхем програмованої логіки [Текст]: монографія. / Іванець С., Зубань Ю., Казимир В., Литвинов В.; Сумський державний університет, Чернігівський державний технологічний університет. – Суми, Чернігів: СДУ, ЧДТУ, 2013. – 313 с.

14. Шадрін М.С. Зеленьова І.Я., Грушко С.С. Порівняльне дослідження структур суміщеного пристрою керування вбудованою системою в базисі FPGA // Збірник тез доповідей щорічної науково-технічної конференції серед студентів, викладачів, науковців, молодих учених і аспірантів "Тиждень науки-2021. Факультет комп'ютерних наук і технологій", 18–22 квітня 2022 р. - Редкол. : В. В. Наумик (відпов. ред.) Електрон. дані. – Запоріжжя : НУ «Запорізька політехніка», 2022. – 1 електрон. опт. диск (DVD-ROM); 12 см. – Назва з тит. екрана. – С. 812.

15. Synthesis of a Finite State Machine With Elementary Chains of States / Alexander Barkalov, Larysa Titarenko, Irina Zeleneva, Svitlana Hrushko, Tetiana Holub, Mykhailo Shadrin // Problems of Infocommunications. Science and Technology PIC S&T'2022: October 10-12, 2022, – Kharkiv, Ukraine, 2022 – pp. 25-28.

16. Parallel organisation of an embedded system control on a single FPGA chip / Irina Zeleneva, Tetiana Holub, Svitlana Hrushko, Mykhaylo Shadrin // Innovative scientific research: theory and practice: int. conf., 21-24 november 2023 – Stockholm, Sweden, 2023 – P. 453-459.

ДОДАТОК А

Таблиця А.1 – Адресація станів ЛПС

A_m	Q₅	Q₄	Q₃	Q₂	Q₁	Q₀	vY₁	vY₂	vY₃	vY₄	vY₅	uY₆	uY₇	uY₈
a ₀	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a ₁	0	0	0	0	0	1	1	1	0	0	0	0	0	0
a ₂	0	0	0	0	1	0	0	0	0	1	1	0	0	0
a ₃	0	0	0	0	1	1	0	1	0	0	0	0	0	0
a ₄	0	0	0	1	0	0	1	0	0	1	0	0	0	0
a ₅	0	0	0	1	0	1	0	1	0	0	0	0	0	0
a ₆	0	0	0	1	1	0	0	0	0	0	1	0	0	0
a ₇	0	0	0	1	1	1	1	0	0	1	0	0	0	0
a ₈	0	0	1	0	0	0	0	1	0	0	0	0	0	0
a ₉	0	0	1	0	0	1	1	1	1	0	0	0	0	0
a ₁₀	0	0	1	0	1	0	1	0	0	1	0	0	0	0
a ₁₁	0	0	1	0	1	1	0	1	0	0	0	0	0	0
a ₁₂	0	0	1	1	0	0	1	0	0	1	0	0	0	0
a ₁₃	0	0	1	1	0	1	0	0	0	1	0	0	0	0
a ₁₄	0	0	1	1	1	0	0	1	0	0	0	0	0	0
a ₁₅	0	0	1	1	1	1	0	0	0	1	0	0	0	0
a ₁₆	0	1	0	0	0	0	0	0	0	0	1	0	0	0
a ₁₇	0	1	0	0	0	1	0	0	0	0	0	1	0	0
a ₁₈	0	1	0	0	1	0	1	1	0	0	0	0	0	0
a ₁₉	0	1	0	0	1	1	0	0	0	1	1	0	0	0
a ₂₀	0	1	0	1	0	0	0	1	0	0	0	0	0	0
a ₂₁	0	1	0	1	0	1	1	0	0	1	0	0	0	0
a ₂₂	0	1	0	1	1	0	0	1	0	0	0	0	0	0
a ₂₃	0	1	0	1	1	1	0	0	0	0	1	0	0	0
a ₂₄	0	1	1	0	0	0	1	0	0	1	0	0	0	0
a ₂₅	0	1	1	0	0	1	0	1	0	0	0	0	0	0
a ₂₆	0	1	1	0	1	0	1	1	1	0	0	0	0	0
a ₂₇	0	1	1	0	1	1	1	0	0	1	0	0	0	0
a ₂₈	0	1	1	1	0	0	0	1	0	0	0	0	0	0
a ₂₉	0	1	1	1	0	1	1	0	0	1	0	0	0	0
a ₃₀	0	1	1	1	1	0	0	0	0	1	0	0	0	0
a ₃₁	0	1	1	1	1	1	0	1	0	0	0	0	0	0
a ₃₂	1	0	0	0	0	0	0	0	0	1	0	0	0	0
a ₃₃	1	0	0	0	0	1	0	0	0	0	1	0	0	0

Продовження таблиці А.1 - Адресація станів ЛПС

A_m	Q_5	Q_4	Q_3	Q_2	Q_1	Q_0	vY_1	vY_2	vY_3	vY_4	vY_5	uY_6	uY_7	uY_8
a_{34}	1	0	0	0	1	0	0	0	0	0	0	0	1	0
a_{35}	1	0	0	0	1	1	1	1	0	0	0	0	0	0
a_{36}	1	0	0	1	0	0	0	0	0	1	1	0	0	0
a_{37}	1	0	0	1	0	1	0	1	0	0	0	0	0	0
a_{38}	1	0	0	1	1	0	1	0	0	1	0	0	0	0
a_{39}	1	0	0	1	1	1	0	1	0	0	0	0	0	0
a_{40}	1	0	1	0	0	0	0	0	0	0	1	0	0	0
a_{41}	1	0	1	0	0	1	1	0	0	1	0	0	0	0
a_{42}	1	0	1	0	1	0	0	1	0	0	0	0	0	0
a_{43}	1	0	1	0	1	1	1	1	1	0	0	0	0	0
a_{44}	1	0	1	1	0	0	1	0	0	1	0	0	0	0
a_{45}	1	0	1	1	0	1	0	1	0	0	0	0	0	0
a_{46}	1	0	1	1	1	0	1	0	0	1	0	0	0	0
a_{47}	1	0	1	1	1	1	0	0	0	1	0	0	0	0
a_{48}	1	1	0	0	0	0	0	1	0	0	0	0	0	0
a_{49}	1	1	0	0	0	1	0	0	0	1	0	0	0	0
a_{50}	1	1	0	0	1	0	0	0	0	0	1	0	0	0
a_{51}	1	1	0	0	1	1	0	0	0	0	0	0	0	1

Таблиця А.2 - Структурна таблиця переходів – виходів автомата Мура

	Початковий стан						Умови пере- ходу		Подальший стан						Функції збудження					
Mіт -ка	Код							Mіт -ка	Код											
	Q ₀	Q ₁	Q ₂	Q ₃	Q ₄	Q ₅			Q ₀	Q ₁	Q ₂	Q ₃	Q ₄	Q ₅	D ₀	D ₁	D ₂	D ₃	D ₄	D ₅
a ₀	0	0	0	0	0	0	uX_1	a ₁	0	0	0	0	0	1	0	0	0	0	0	1
	0	0	0	0	0	0	$\overline{uX_1}$	a ₃	0	0	0	0	1	1	0	0	0	0	1	1
a ₁	0	0	0	0	0	1	1	a ₂	0	0	0	0	1	0	0	0	0	0	1	0
a ₂	0	0	0	0	1	0	uX_2	a ₁	0	0	0	0	0	1	0	0	0	0	0	1
	0	0	0	0	1	0	$\overline{uX_2}uX_3$	a ₈	0	0	1	0	0	0	0	0	1	0	0	0
	0	0	0	0	1	0	$\overline{uX_2}uX_3$	a ₁₁	0	0	1	0	1	1	0	0	1	0	1	1
a ₃	0	0	0	0	1	1	1	a ₄	0	0	0	1	0	0	0	0	0	1	0	0
a ₄	0	0	0	1	0	0	1	a ₅	0	0	0	1	0	1	0	0	0	1	0	1
a ₅	0	0	0	1	0	1	1	a ₆	0	0	0	1	1	0	0	0	0	1	1	0
a ₆	0	0	0	1	1	0	1	a ₇	0	0	0	1	1	1	0	0	0	1	1	1
a ₇	0	0	0	1	1	1	uX_4	a ₁₃	0	0	1	1	0	1	0	0	1	1	0	1
	0	0	0	1	1	1	$\overline{uX_4}$	a ₁₅	0	0	1	1	1	1	0	0	1	1	1	1
a ₈	0	0	1	0	0	0	1	a ₉	0	0	1	0	0	1	0	0	1	0	0	1
a ₉	0	0	1	0	0	1	1	a ₁₀	0	0	1	0	1	0	0	0	1	0	1	0
a ₁₀	0	0	1	0	1	0	$\overline{uX_5}$	a ₁₇	0	1	0	0	0	1	0	1	0	0	0	1
	0	0	1	0	1	0	uX_5uX_1	a ₁	0	0	0	0	0	1	0	0	0	0	0	1
	0	0	1	0	1	0	$uX_5\overline{uX_1}$	a ₃	0	0	0	0	1	1	0	0	0	0	1	1

Продовження таблиці А.2 - Структурна таблиця переходів – виходів автомата Мура

Міт -ка	Початковий стан						Умови пере- ходу	Міт -ка	Подальший стан						Функції збудження					
	Код								Код											
	Q ₀	Q ₁	Q ₂	Q ₃	Q ₄	Q ₅			Q ₀	Q ₁	Q ₂	Q ₃	Q ₄	Q ₅	D ₀	D ₁	D ₂	D ₃	D ₄	D ₅
a ₁₁	0	0	1	0	1	1	1	a ₁₂	0	0	1	1	0	0	0	0	1	1	0	0
a ₁₂	0	0	1	1	0	0	$\overline{uX_5}$	a ₁₇	0	1	0	0	0	1	0	1	0	0	0	1
	0	0	1	1	0	0	uX_5uX_1	a ₁	0	0	0	0	0	1	0	0	0	0	0	1
	0	0	1	1	0	0	$uX_5\overline{uX_1}$	a ₃	0	0	0	0	1	1	0	0	0	0	1	1
a ₁₃	0	0	1	1	0	1	1	a ₁₄	0	0	1	1	1	0	0	0	1	1	1	0
a ₁₄	0	0	1	1	1	0	$\overline{uX_5}$	a ₁₇	0	1	0	0	0	1	0	1	0	0	0	1
	0	0	1	1	1	0	uX_5uX_1	a ₁	0	0	0	0	0	1	0	0	0	0	0	1
	0	0	1	1	1	0	$uX_5\overline{uX_1}$	a ₃	0	0	0	0	1	1	0	0	0	0	1	1
a ₁₅	0	0	1	1	1	1	1	a ₁₆	0	1	0	0	0	0	0	0	0	1	0	1
a ₁₆	0	1	0	0	0	0	$\overline{uX_5}$	a ₁₇	0	1	0	0	0	1	0	0	0	0	0	0
	0	1	0	0	0	0	uX_5uX_1	a ₁	0	0	0	0	0	1	0	0	1	1	1	1
	0	1	0	0	0	0	$uX_5\overline{uX_1}$	a ₃	0	0	0	0	1	1	0	0	0	1	0	1
a ₁₇	0	1	0	0	0	1	uX_6	a ₁₈	0	1	0	0	1	0	0	1	0	0	1	0
	0	1	0	0	0	1	$\overline{uX_6}$	a ₂₀	0	1	0	1	0	0	0	1	0	1	0	0
a ₁₈	0	1	0	0	1	0	1	a ₁₉	0	1	0	0	1	1	0	1	0	0	1	1
a ₁₉	0	1	0	0	1	1	uX_7	a ₁₈	0	1	0	0	1	0	0	1	0	0	1	0
	0	1	0	0	1	1	$\overline{uX_7}uX_8$	a ₂₅	0	1	1	0	0	1	0	1	1	0	0	1
	0	1	0	0	1	1	$\overline{uX_7}\overline{uX_8}$	a ₂₈	0	1	1	1	0	0	0	1	1	1	0	0
a ₂₀	0	1	0	1	0	0	1	a ₂₁	0	1	0	1	0	1	0	1	0	1	0	1
a ₂₁	0	1	0	1	0	1	1	a ₂₂	0	1	0	1	1	0	0	1	0	1	1	0
a ₂₂	0	1	0	1	1	0	1	a ₂₃	0	1	0	1	1	1	0	1	0	1	1	1
a ₂₃	0	1	0	1	1	1	1	a ₂₄	0	1	1	0	0	0	0	1	1	0	0	0
a ₂₄	0	1	1	0	0	0	uX_9	a ₃₀	0	1	1	1	1	0	0	1	1	1	1	0
	0	1	1	0	0	0	$\overline{uX_9}$	a ₃₂	1	0	0	0	0	0	0	1	1	1	1	1
a ₂₅	0	1	1	0	0	1	1	a ₂₆	0	1	1	0	1	0	0	1	1	0	1	0
a ₂₆	0	1	1	0	1	0	1	a ₂₇	0	1	1	0	1	1	0	1	1	0	1	1
a ₂₇	0	1	1	0	1	1	$\overline{uX_{10}}$	a ₃₄	1	0	0	0	1	0	0	1	0	0	0	1
	0	1	1	0	1	1	$uX_{10}uX_6$	a ₁₈	0	1	0	0	1	0	0	1	0	0	1	0
	0	1	1	0	1	1	$uX_{10}\overline{uX_6}$	a ₂₀	0	1	0	1	0	0	0	1	0	1	0	0
a ₂₈	0	1	1	1	0	0	1	a ₁₂	0	0	1	1	0	0	0	0	1	1	0	0
a ₂₉	0	1	1	1	0	1	$\overline{uX_{10}}$	a ₃₄	1	0	0	0	1	0	0	1	0	0	0	1
	0	1	1	1	0	1	$uX_{10}uX_6$	a ₁₈	0	1	0	0	1	0	0	1	0	0	1	0
	0	1	1	1	0	1	$uX_{10}\overline{uX_6}$	a ₂₀	0	1	0	1	0	0	0	1	0	1	0	0
a ₃₀	0	1	1	1	1	0	1	a ₃₁	0	1	1	1	1	1	0	1	1	1	1	1
a ₃₁	0	1	1	1	1	1	$\overline{uX_{10}}$	a ₃₄	1	0	0	0	1	0	0	1	0	0	0	1
	0	1	1	1	1	1	$uX_{10}uX_6$	a ₁₈	0	1	0	0	1	0	0	1	0	0	1	0
	0	1	1	1	1	1	$uX_{10}\overline{uX_6}$	a ₂₀	0	1	0	1	0	0	0	1	0	1	0	0
a ₃₂	1	0	0	0	0	0	1	a ₃₃	1	0	0	0	0	1	1	0	0	0	0	1
a ₃₃	1	0	0	0	0	1	$\overline{uX_{10}}$	a ₃₄	1	0	0	0	1	0	0	1	0	0	0	1

Продовження таблиці А.2 - Структурна таблиця переходів – виходів автомата Мура

	Початковий стан						Умови пере- ходу		Подальший стан						Функції збудження					
Міт -ка	Код							Міт -ка	Код											
	Q ₀	Q ₁	Q ₂	Q ₃	Q ₄	Q ₅			Q ₀	Q ₁	Q ₂	Q ₃	Q ₄	Q ₅	D ₀	D ₁	D ₂	D ₃	D ₄	D ₅
	1	0	0	0	0	1	$uX_{10}uX_6$	a ₁₈	0	1	0	0	1	0	0	1	0	0	1	0
	1	0	0	0	0	1	$uX_{10}\overline{uX_6}$	a ₂₀	0	1	0	1	0	0	0	1	0	1	0	0
a ₃₄	1	0	0	0	1	0	uX_{11}	a ₃₅	1	0	0	0	1	1	1	0	0	0	1	1
	1	0	0	0	1	0	$\overline{uX_{11}}$	a ₃₇	1	0	0	1	0	1	1	0	0	1	0	1
a ₃₅	1	0	0	0	1	1	1	a ₃₆	1	0	0	1	0	0	1	0	0	1	0	0
a ₃₆	1	0	0	1	0	0	uX_{12}	a ₃₅	1	0	0	0	1	1	1	0	0	0	1	1
	1	0	0	1	0	0	$\overline{uX_{12}}uX_{11}$	a ₄₅	1	0	1	1	0	1	1	0	1	0	1	0
	1	0	0	1	0	0	$\overline{uX_{12}}\overline{uX_{11}}$	a ₄₂	1	0	1	0	1	0	1	0	1	1	0	1
a ₃₇	1	0	0	1	0	1	1	a ₃₈	1	0	0	1	1	0	1	0	0	1	1	0
a ₃₈	1	0	0	1	1	0	1	a ₃₉	1	0	0	1	1	1	1	0	0	1	1	1
a ₃₉	1	0	0	1	1	1	1	a ₄₀	1	0	1	0	0	0	1	0	1	0	0	0
a ₄₀	1	0	1	0	0	0	1	a ₄₁	1	0	1	0	0	1	1	0	1	0	0	1
a ₄₁	1	0	1	0	0	1	uX_{14}	a ₄₇	1	0	1	1	1	1	1	0	1	1	1	1
	1	0	1	0	0	1	$\overline{uX_{14}}$	a ₄₉	1	1	0	0	0	1	1	1	0	0	0	1
a ₄₂	1	0	1	0	1	0	1	a ₄₃	1	0	1	0	1	1	1	0	1	0	1	1
a ₄₃	1	0	1	0	1	1	1	a ₄₄	1	0	1	1	0	0	1	0	1	1	0	0
a ₄₄	1	0	1	1	0	0	$\overline{uX_{15}}$	a ₅₁	1	1	0	0	1	1	1	0	0	0	1	0
	1	0	1	1	0	0	$uX_{15}uX_{11}$	a ₃₅	1	0	0	0	1	1	1	0	0	0	1	1
	1	0	1	1	0	0	$uX_{15}\overline{uX_{11}}$	a ₃₇	1	0	0	1	0	1	1	0	0	1	0	1
a ₄₅	1	0	1	1	0	1	1	a ₄₆	1	0	1	1	1	0	1	0	1	1	1	0
a ₄₆	1	0	1	1	1	0	$\overline{uX_{15}}$	a ₅₁	1	1	0	0	1	1	1	0	0	0	1	0
	1	0	1	1	1	0	$uX_{15}uX_{11}$	a ₃₅	1	0	0	0	1	1	1	0	0	0	1	1
	1	0	1	1	1	0	$uX_{15}\overline{uX_{11}}$	a ₃₇	1	0	0	1	0	1	1	0	0	1	0	1
a ₄₇	1	0	1	1	1	1	1	a ₄₈	1	1	0	0	0	0	1	1	0	0	0	0
a ₄₈	1	1	0	0	0	0	$\overline{uX_{15}}$	a ₅₁	1	1	0	0	1	1	1	0	0	0	1	0
	1	1	0	0	0	0	$uX_{15}uX_{11}$	a ₃₅	1	0	0	0	1	1	1	0	0	0	1	1
	1	1	0	0	0	0	$uX_{15}\overline{uX_{11}}$	a ₃₇	1	0	0	1	0	1	1	0	0	1	0	1
a ₄₉	1	1	0	0	0	1	1	a ₅₀	1	1	0	0	1	0	1	1	0	0	1	0
a ₅₀	1	1	0	0	1	0	$\overline{uX_{15}}$	a ₅₁	1	1	0	0	1	1	0	0	0	0	0	0
	1	1	0	0	1	0	$uX_{15}uX_{11}$	a ₃₅	1	0	0	0	1	1	1	0	0	0	1	1
	1	1	0	0	1	0	$uX_{15}\overline{uX_{11}}$	a ₃₇	1	0	0	1	0	1	1	0	0	1	0	1
a ₅₁	1	1	0	0	1	1	uX_{11}	a ₁	0	0	0	0	0	1	0	0	0	0	0	1
	1	1	0	0	1	1	$\overline{uX_{11}}$	a ₃	0	0	0	0	1	1	0	0	0	0	1	1

Таблиця А.3 – Адресація станів ЛПС

ЛПС α_i	I_{bx}	A_m	Q_5	Q_4	Q_3	Q_2	Q_1	Q_0	vY_0	vY_1	vY_2	vY_3	vY_4	vY_5	uY_6	uY_7	uY_8
α_1	I_1	a_1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
		a_2	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0
α_2	I_2	a_3	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0
		a_4	0	0	0	0	1	1	0	1	0	0	1	0	0	0	0
		a_5	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
		a_6	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0
		a_7	0	0	0	1	1	0	1	1	0	0	1	0	0	0	0
α_3	I_3	a_8	0	0	0	1	1	1	0	0	1	0	0	0	0	0	0
		a_9	0	0	1	0	0	0	0	1	1	1	0	0	0	0	0
		a_{10}	0	0	1	0	0	1	1	1	0	0	1	0	0	0	0
α_4	I_4	a_{11}	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0
		a_{12}	0	0	1	0	1	1	1	1	0	0	1	0	0	0	0
α_5	I_5	a_{13}	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0
		a_{14}	0	0	1	1	0	1	1	0	1	0	0	0	0	0	0
α_6	I_6	a_{15}	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0
		a_{16}	0	0	1	1	1	1	1	0	0	0	0	1	0	0	0
α_7	I_7	a_{17}	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0
α_8	I_8	a_{18}	0	1	0	0	0	1	0	1	1	0	0	0	0	0	0
		a_{19}	0	1	0	0	1	0	1	0	0	0	1	1	0	0	0
α_9	I_9	a_{20}	0	1	0	0	1	1	0	0	1	0	0	0	0	0	0
		a_{21}	0	1	0	1	0	0	0	1	0	0	1	0	0	0	0
		a_{22}	0	1	0	1	0	1	0	0	1	0	0	0	0	0	0
		a_{23}	0	1	0	1	1	0	0	0	0	0	0	1	0	0	0
		a_{24}	0	1	0	1	1	1	1	1	0	0	1	0	0	0	0
α_{10}	I_{10}	a_{25}	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0
		a_{26}	0	1	1	0	0	1	0	1	1	1	0	0	0	0	0
		a_{27}	0	1	1	0	1	0	1	1	0	0	1	0	0	0	0
α_{11}	I_{11}	a_{28}	0	1	1	0	1	1	0	0	1	0	0	0	0	0	0
		a_{29}	0	1	1	1	0	0	1	1	0	0	1	0	0	0	0
α_{12}	I_{12}	a_{30}	0	1	1	1	0	1	0	0	0	0	1	0	0	0	0
		a_{31}	0	1	1	1	1	0	1	0	1	0	0	0	0	0	0
α_{13}	I_{13}	a_{32}	0	1	1	1	1	1	0	0	0	0	1	0	0	0	0
		a_{33}	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0
α_{14}	I_{14}	a_{34}	1	0	0	0	0	1	1	0	0	0	0	0	0	1	0
α_{15}	I_{15}	a_{35}	1	0	0	0	1	0	0	1	1	0	0	0	0	0	0
		a_{36}	1	0	0	0	1	1	1	0	0	0	1	1	0	0	0
α_{16}	I_{16}	a_{37}	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0
		a_{38}	1	0	0	1	0	1	0	1	0	0	1	0	0	0	0
		a_{39}	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0
		a_{40}	1	0	0	1	1	1	0	0	0	0	0	1	0	0	0
		a_{41}	1	0	1	0	0	0	1	1	0	0	1	0	0	0	0
α_{17}	I_{17}	a_{42}	1	0	1	0	0	1	0	0	1	0	0	0	0	0	0
		a_{43}	1	0	1	0	1	0	0	1	1	1	0	0	0	0	0
		a_{44}	1	0	1	0	1	1	1	1	0	0	1	0	0	0	0
α_{18}	I_{18}	a_{45}	1	0	1	1	0	0	0	0	1	0	0	0	0	0	0
		a_{46}	1	0	1	1	0	1	1	1	0	0	1	0	0	0	0

Продовження таблиці А.3 – Адресація станів ЛПС

ЛПС α_i	I_{bx}	A_m	Q_5	Q_4	Q_3	Q_2	Q_1	Q_0	vY_0	vY_1	vY_2	vY_3	vY_4	vY_5	uY_6	uY_7	uY_8
α_{19}	I_{19}	a_{47}	1	0	1	1	1	0	0	0	0	0	1	0	0	0	0
		a_{48}	1	0	1	1	1	1	1	0	1	0	0	0	0	0	0
α_{20}	I_{20}	a_{49}	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0
		a_{50}	1	1	0	0	0	1	1	0	0	0	0	1	0	0	0
α_{21}	I_{21}	a_{51}	1	1	0	0	1	0	1	0	0	0	0	0	0	0	1

Таблиця А.4 – Таблиця переходів УА Мілі

b_m	$K(b_m)$				b_s	$K(b_s)$				X_n	D_a	D_b	H
b_0	0	0	0	0	b_1	0	0	0	1	uX_1	1	D_6	1
					b_2	0	0	1	0	uX_1	D_1	D_7	2
b_1	0	0	0	1	b_1	0	0	0	1	uX_2	1	D_6	3
					b_3	0	0	1	1	$\overline{uX_2}uX_3$	D_3D_1	D_7D_6	4
					b_3	0	0	1	1	$\overline{uX_2}uX_3$	$D_2D_1D_0$	D_7D_6	5
b_2	0	0	1	0	b_3	0	0	1	1	uX_4	D_3D_2	D_7D_6	6
					b_3	0	0	1	1	$\overline{uX_4}$	$D_3D_2D_1$	D_7D_6	7
b_3	0	0	1	1	b_4	0	1	0	0	$\overline{uX_5}$	D_4	D_8	8
					b_1	0	0	0	1	uX_5uX_1	1	D_6	9
					b_2	0	0	1	0	uX_5uX_1	D_1	D_6	10
b_4	0	1	0	0	b_5	0	1	0	1	uX_6	D_4D_0	D_8D_6	11
					b_6	0	1	1	0	$\overline{uX_6}$	$D_4D_1D_0$	D_8D_7	12
b_5	0	1	0	1	b_5	0	1	0	1	uX_7	D_4D_0	D_8D_6	13
					b_7	0	1	1	1	$\overline{uX_7}uX_8$	$D_4D_3D_1D_0$	$D_8D_7D_6$	13
					b_7	0	1	1	1	$\overline{uX_7}uX_8$	D_4D_3	$D_8D_7D_6$	13
b_6	0	1	1	0	b_7	0	1	1	1	uX_9	$D_4D_3D_2D_0$	$D_8D_7D_6$	14
					b_7	0	1	1	1	$\overline{uX_9}$	$D_4D_3D_2D_1D_0$	$D_8D_7D_6$	14
b_7	0	1	1	1	b_8	1	0	0	0	$\overline{uX_{10}}$	D_5D_0	D_9	15
					b_5	0	1	0	1	$uX_{10}uX_6$	D_4D_0	D_8D_6	16
					b_6	0	1	1	0	$uX_{10}\overline{uX_6}$	$D_4D_1D_0$	D_8D_7	17
b_8	1	0	0	0	b_9	1	0	0	1	uX_{11}	D_5D_1	D_9D_6	18
					b_{10}	1	0	1	0	$\overline{uX_{11}}$	D_5D_2	D_9D_7	19
b_9	1	0	0	1	b_9	1	0	0	1	uX_{12}	D_5D_1	D_9D_6	20
					b_{11}	1	0	1	1	$\overline{uX_{12}}uX_{13}$	$D_5D_3D_2$	$D_9D_7D_6$	21
					b_{11}	1	0	1	1	$\overline{uX_{12}}uX_{13}$	$D_5D_3D_0$	$D_9D_7D_6$	22
b_{10}	1	0	1	0	b_{11}	1	0	1	1	uX_{14}	$D_5D_3D_2D_1$	$D_9D_7D_6$	23
					b_{11}	1	0	1	1	$\overline{uX_{14}}$	D_5D_4	$D_9D_7D_6$	24
b_{11}	1	0	1	1	b_0	0	0	0	0	$\overline{uX_{15}}$	$D_5D_4D_1$	1	25
					b_9	1	0	0	1	$uX_{15}uX_{11}$	D_5D_1	D_9D_6	21
					b_{10}	1	0	1	0	$uX_{15}uX_{11}$	D_5D_2	D_9D_7	22

ДОДАТОК Б

Лістинг Б.1 - VHDL опис блоку пристрою керування базової структури
автомата який працює за першим алгоритмом

```

library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity automat is
  port(
    vX: in std_logic_vector(0 to 14);
    vY : out STD_LOGIC_VECTOR(0 to 7);
    D : out std_logic_vector(5 downto 0);
    clk, Reset : in std_logic
  );
end automat;

architecture automat of automat is
  signal Dsig,Q: std_logic_vector(5 downto 0);
  signal
a0,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,a16,a17
,a18,a19,a20,a21,a22,a23,a24,a25,a26,a27,a28,a29,a30,
a31,a32,a33,a34,a35,a36,a37,a38,a39,a40,a41,a42,a43,a44,a45,a
46,a47,a48,a49,a50,a51: std_logic;
begin
  process(clk, Reset)
  begin
    if (Reset = '1') then
      Q <= "000000";
    elsif (clk'event and clk = '1') then
      Q<=Dsig;
    end if;
  end process;
  process(Q)
  begin
    a0 <= '0';
    a1 <= '0';
    a2 <= '0';
    a3 <= '0';
    a4 <= '0';
    a5 <= '0';
    a6 <= '0';
    a7 <= '0';
    a8 <= '0';
    a9 <= '0';
    a10 <= '0';
    a11 <= '0';
    a12 <= '0';
    a13 <= '0';

```

```

a14 <= '0';
a15 <= '0';
a16 <= '0';
a17 <= '0';
a18 <= '0';
a19 <= '0';
a20 <= '0';
a21 <= '0';
a22 <= '0';
a23 <= '0';
a24 <= '0';
a25 <= '0';
a26 <= '0';
a27 <= '0';
a28 <= '0';
a29 <= '0';
a30 <= '0';
a31 <= '0';
a32 <= '0';
a33 <= '0';
a34 <= '0';
a35 <= '0';
a36 <= '0';
a37 <= '0';
a38 <= '0';
a39 <= '0';
a40 <= '0';
a41 <= '0';
a42 <= '0';
a43 <= '0';
a44 <= '0';
a45 <= '0';
a46 <= '0';
a47 <= '0';
a48 <= '0';
a49 <= '0';
a50 <= '0';
a51 <= '0';
case Q is
  when "000000" => a0 <= '1';
  when "000001" => a1 <= '1';
  when "000010" => a2 <= '1';
  when "000011" => a3 <= '1';
  when "000100" => a4 <= '1';
  when "000101" => a5 <= '1';
  when "000110" => a6 <= '1';
  when "000111" => a7 <= '1';

  .....

  when "110010" => a50 <= '1';
  when "110011" => a51 <= '1';
  when others => report "Error in Qb" severity ERROR;

```

```

    end case;
  end process;
  process(vX,
a0,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,a16,a17
,a18,a19,a20,a21,a22,a23,a24,a25,a26,a27,a28,a29,a30,
a31,a32,a33,a34,a35,a36,a37,a38,a39,a40,a41,a42,a43,a44,a45,a
46,a47,a48,a49,a50,a51)
  begin
    Dsig(0)<=(a0 and vX(0)) or ( a0 and (not vX(0))) or (a2
and vX(1)) or ( a2 and (not vX(1))and vX(2)) or (a4) or (a6)
or (a7 and vX(3)) i т.д.
    Dsig(1)<=( a0 and (not vX(0))) or (a1) or ( a2 and (not
vX(1))and vX(2)) or (a5) i т.д.
    Dsig(2)<=(a3) or (a4) or (a5) or (a6) or (a7 and vX(3))
or ( a7 and (not vX(3))) i т.д.
    Dsig(3)<=( a2 and (not vX(1)) and (not vX(2))) or ( a2
and (not vX(1))and vX(2)) i т.д.
    Dsig(4)<=(a10 and (not vX(4))) or (a12 and (not vX(4)))
or (a14 and (not vX(4))) i т.д.
    Dsig(5)<=(a27 and (not vX(9))) or (a29 and ( vX(9))) or
(a31 and (not vX(9))) or (a32) i т.д.
    D<=Dsig;
    vY(0) <= a1 or a4 or a7 or a9 or a10 or a12 or a18 or a21
or a24 or a26 or a27 or a29 or a35 or a38 or a41 or a43 or
a44 or a46;
    vY(1) <= a1 or a3 or a5 or a8 or a9 or a11 or a14 or a18 or
a20 or a22 or a25 or a26 or a28 or a31 or a35 or a37 or a39
or a42 or a43 or a45 or a48;
    vY(2) <= a9 or a26 or a43;
    vY(3) <= a2 or a4 or a7 or a10 or a12 or a13 or a15 or a19
or a21 or a24 or a27 or a29 or a30 or a32 or a36 or a38 or
a41 or a44 or a46 or a47 or a49;
    vY(4) <= a2 or a6 or a16 or a19 or a23 or a33 or a36 or a40
or a50;
    vY(5) <= a17;
    vY(6) <= a34;
    vY(7) <= a51;

    end process;
  end automat;

```

ДОДАТОК В

Лістинг В.1 - VHDL опис блоку пристрою керування композиційної
структури автомата який працює за першим алгоритмом

```

library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity automat is
    port(
        vX: in std_logic_vector(0 to 14);
        Da: out std_logic_vector(5 downto 0);
        Db: out std_logic_vector(3 downto 0);
        Qb: in std_logic_vector(3 downto 0);
        clk, start : in std_logic
    );
end automat;
architecture automat of automat is
    signal Q: std_logic_vector(3 downto 0);
    signal b0, b1, b2, b3, b4, b5, b6, b7, b8, b9, b10, b11:
        std_logic;
begin
    process(clk, start)
    begin
        if (start = '0') then
            Q <= "0000";
        elsif (clk'event and clk = '1') then
            Q<=Qb;
        end if;
    end process;
    process(Q)
    begin
        b0<='0';
        b1<='0';
        b2<='0';
        b3<='0';
        b4<='0';
        b5<='0';
        b6<='0';
        b7<='0';
        b8<='0';
        b9<='0';
        b10<='0';
        b11<='0';
        case Q is
            when "0000" => b0 <= '1';
            when "0001" => b1 <= '1';
            when "0010" => b2 <= '1';
            when "0011" => b3 <= '1';
            when "0100" => b4 <= '1';
        end case;
    end process;
end automat;

```

```

        when "0101" => b5 <= '1';
        when "0110" => b6 <= '1';
        when "0111" => b7 <= '1';
        when "1000" => b8 <= '1';
        when "1001" => b9 <= '1';
        when "1010" => b10 <= '1';
        when "1011" => b11 <= '1';
        when others => report "Error in Qb" severity
ERROR;
        end case;
    end process;
    process(vX, b0, b1, b2, b3, b4, b5, b6, b7, b8, b9,
b10, b11)
    begin
        Db(0)<=(vX(0) and b0) or ((not vX(8)) and b6) or (vX(1)
and b1) or (vX(9) and vX(5) and b7) or (vX(10) and b8) or
(vX(11) and b9) i т.д.
        Db(1)<=((not vX(0)) and b0) or (vX(9) and (not vX(5)) and
b7) or (vX(2) and (not vX(1)) and b1) or (vX(8) and b6) i
т.д.
        Db(2)<=((not vX(6)) and vX(7) and b5) or ((not vX(4)) and
b3) or (vX(5) and b4) i т.д.
        Db(3)<=((not vX(9)) and b7) or ((not vX(10)) and b8) or
(vX(11) and b9) i т.д.
        Da(0)<=((not vX(1)) and vX(2) and b1) or ((not vX(9)) and
b7) or (vX(8) and b6) i т.д.
        Da(1)<=((not vX(0)) and b0) or ((not vX(5)) and b4) or
((not vX(14)) and b11) i т.д.
        Da(2)<=((not vX(1)) and vX(2) and b1) or ((not vX(8)) and
b6) or ((not vX(10)) and b8) i т.д.
        Da(3)<=((not vX(1)) and (not vX(2)) and b1) or ((not
vX(3)) and b2) i т.д.
        Da(4)<=((not vX(4)) and b3) or (vX(9) and (not vX(5))
and b7) or ((not vX(13)) and b10) i т.д.
        Da(5)<=((not vX(9)) and b7) or (vX(10) and b8) or ((not
vX(11)) and (not vX(12)) and b9) i т.д.
        end process;
    end automat;

```

Лістинг В.2 – Опис лічильника на VHDL

```

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;
entity counter is
    port(
        Da : in STD_LOGIC_VECTOR(5 downto 0);
        clk, start, y0 : in STD_LOGIC;
        Qa : out STD_LOGIC_VECTOR(5 downto 0)
    );
end counter;

```

```

architecture counter of counter is
signal csignal: STD_LOGIC_VECTOR(5 downto 0);
begin
    process(clk, start)
    begin
        if (start = '0') then
            csignal <= (others => '0');
        elsif (clk'event and clk = '1') then
            if y0 = '1' then
                csignal <= Da;
            else
                csignal <= csignal + '1';
            end if;
        end if;
    end process;
    Qa<=csignal;
end counter;

```

Лістинг В.3 – Опис регістру на VHDL

```

library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity reg is
    port(
        Db : in STD_LOGIC_VECTOR(3 downto 0);
        clk : in STD_LOGIC;
        start : in STD_LOGIC;
        uY0 : in STD_LOGIC;
        Qb : out STD_LOGIC_VECTOR(3 downto 0)
    );
end reg;
architecture reg of reg is
begin
    process(clk, start)
    begin
        if (start = '0') then Qb <= "0000";
        elsif (clk'event and clk = '1') then
            if uY0 = '1' then Qb<=Db;
            end if;
        end if;
    end process;
end reg;

```

Лістинг В.4 – Опис ROM на VHDL

```

library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity ROM is
    port(
        start, clk : in STD_LOGIC;
        Q : in STD_LOGIC_VECTOR(5 downto 0);
        vY1 : out STD_LOGIC_VECTOR(0 to 8);
    );
end ROM;

```

```

        vY0 : out STD_LOGIC
    );
end ROM;
architecture ROM of ROM is
    signal vY: STD_LOGIC_VECTOR(0 to 8);
begin
    process(start, clk)
    begin
        if (start = '0') then
            vY <= "100000000";
        elsif (clk'event and clk = '1') then
            case Q is
                when "000000" => vY <= "011000000";
                when "000001" => vY <= "100011000";
                when "000010" => vY <= "001000000";
                when "000011" => vY <= "010010000";
                when "000100" => vY <= "001000000";
                when "000101" => vY <= "000001000";
                when "000110" => vY <= "110010000";
                when "000111" => vY <= "001000000";
                when "001000" => vY <= "011100000";
                when "001001" => vY <= "110010000";
                when "001010" => vY <= "001000000";
                when "001011" => vY <= "110010000";
                when "001100" => vY <= "000010000";
                when "001101" => vY <= "101000000";
                when "001110" => vY <= "000010000";
                when "001111" => vY <= "100001000";
                when "010000" => vY <= "100000100";
                when "010001" => vY <= "011000000";
                i т.д. по таблиці
                when others => vY <= "000000000";
            end case;
        end if;
    end process;
    vY1<=vY;
    vY0<=vY(0);
end ROM;

```

ДОДАТОК Г

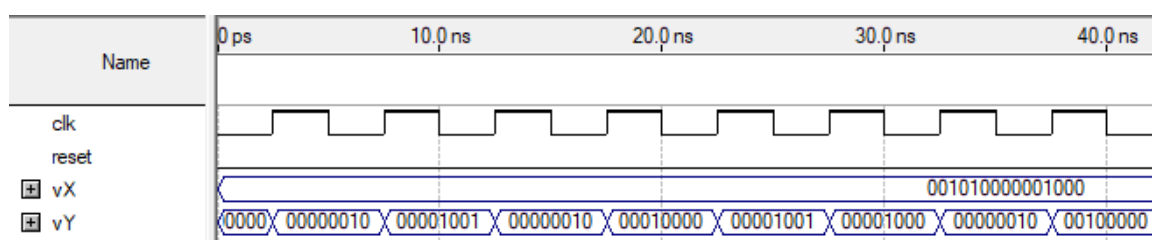


Рисунок Г.1 - Часова діаграма роботи першої частини ГСА рис. 2.1 (основного блоку) зі значенням входів vX в «001010000001000»

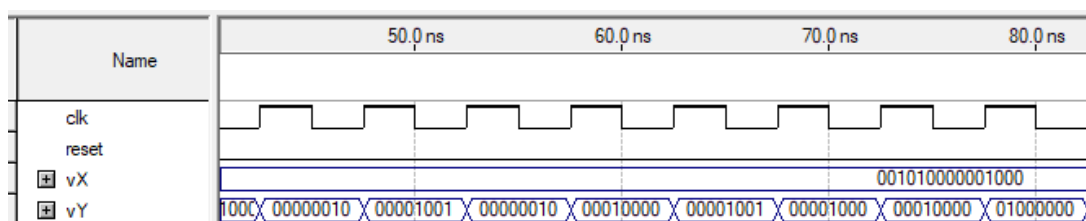


Рисунок Г.2 - Часова діаграма роботи другої частини ГСА рис. 2.2 (першого резервного блоку) зі значенням входів vX в «001010000001000»

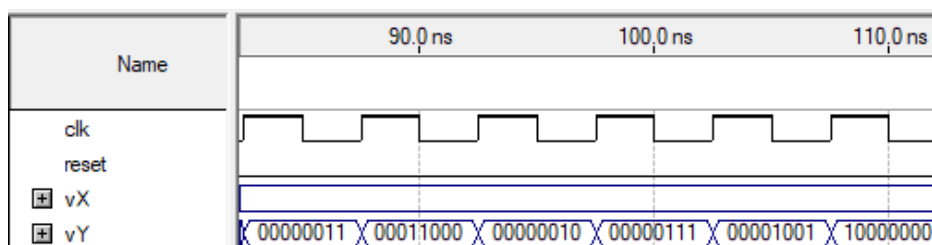


Рисунок Г.3 - Часова діаграма роботи третьої частини ГСА рис. 2.3 (другого резервного блоку) зі значенням входів vX в «001010000001000»

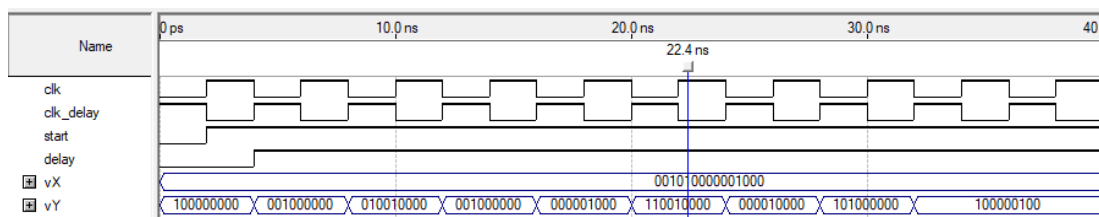


Рисунок Г.4 - Часова діаграма роботи першої частини ГСА рис. 2.1 (основного блоку) зі значенням входів vX в «001010000001000»

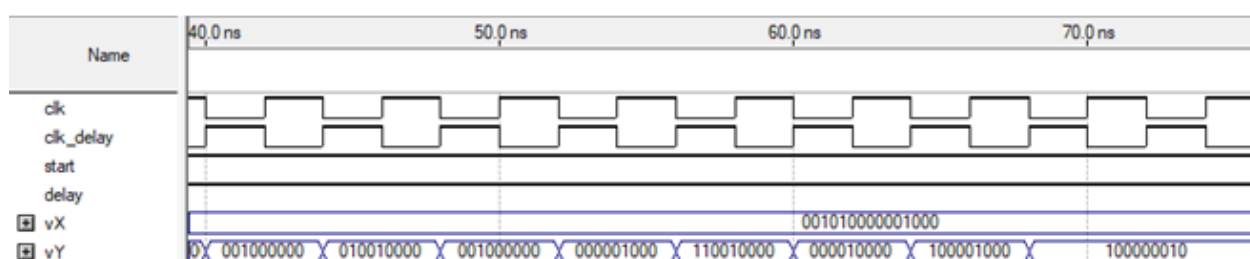


Рисунок Г.5 - Часова діаграма роботи другої частини ГСА рис. 2.2 (першого резервного блоку) зі значенням входів vX в «001010000001000»

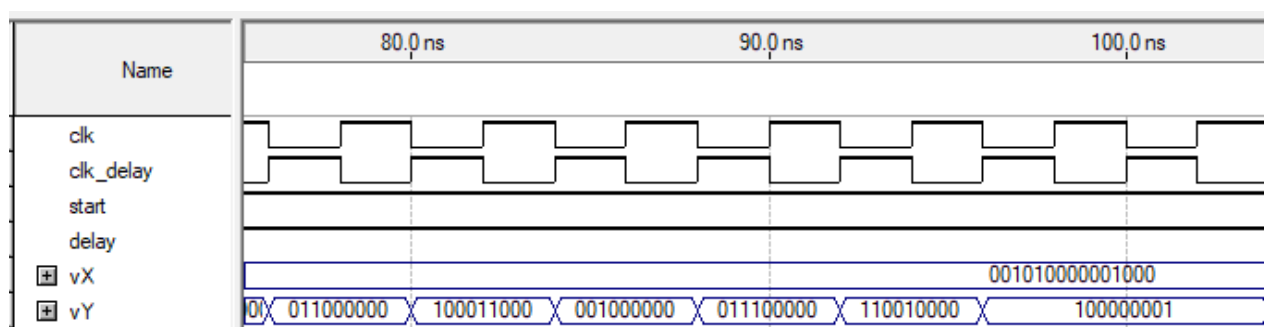


Рисунок Г.6 - Часова діаграма роботи третьої частини ГСА рис. 2.3 (другого резервного блоку) зі значенням входів vX в «001010000001000»

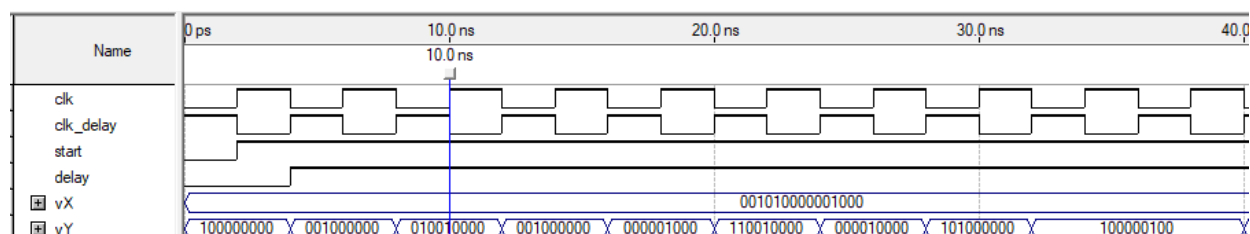


Рисунок Г.7 - Часова діаграма роботи першої частини ГСА рис. 2.1 (основного блоку) зі значенням входів vX в «001010000001000»

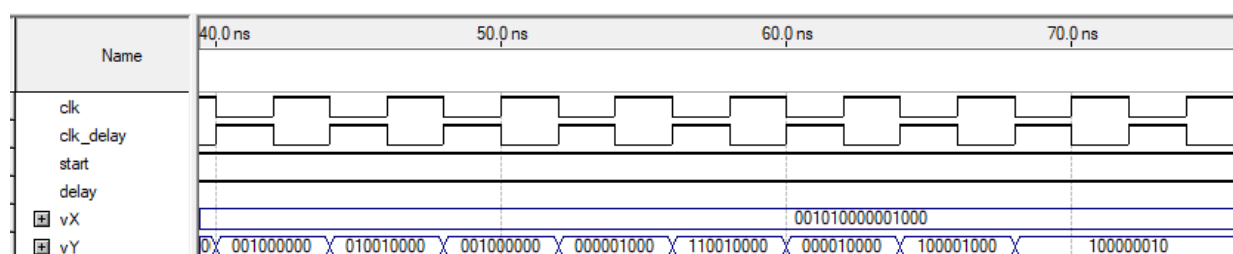


Рисунок Г.8 - Часова діаграма роботи другої частини ГСА рис. 2.2 (першого резервного блоку) зі значенням входів vX в «001010000001000»

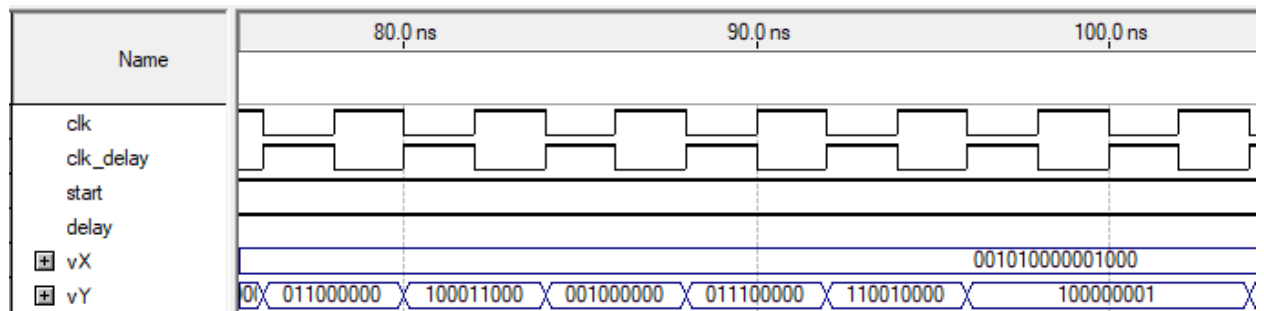


Рисунок Г.9 - Часова діаграма роботи третьої частини ГСА рис. 2.3 (другого резервного блоку) зі значенням входів vX в «001010000001000»

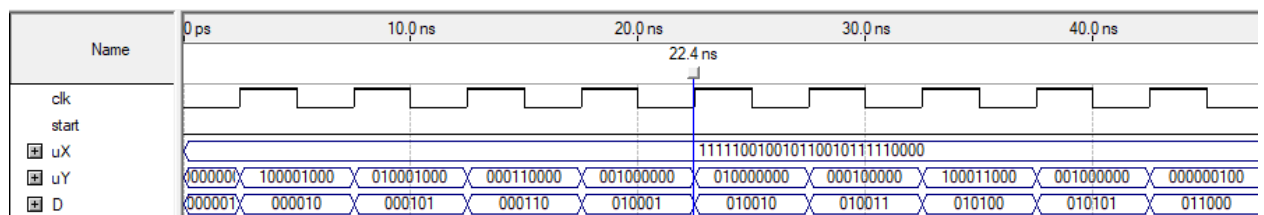


Рисунок Г.10 - Часова діаграма роботи першої частини ГСА рис. 2.4 (основного блоку) зі значенням входів uX в «111110010010110010111110000»

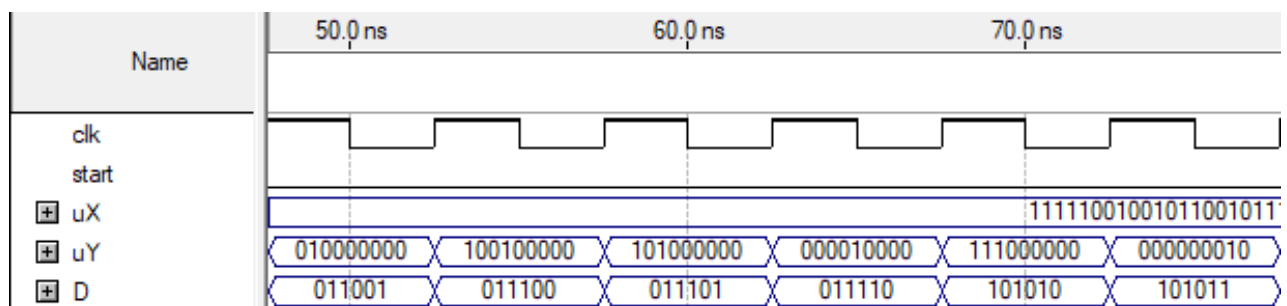


Рисунок Г.11 - Часова діаграма роботи другої частини ГСА рис. 2.5 (першого резервного блоку) зі значенням входів uX в «111110010010110010111110000»

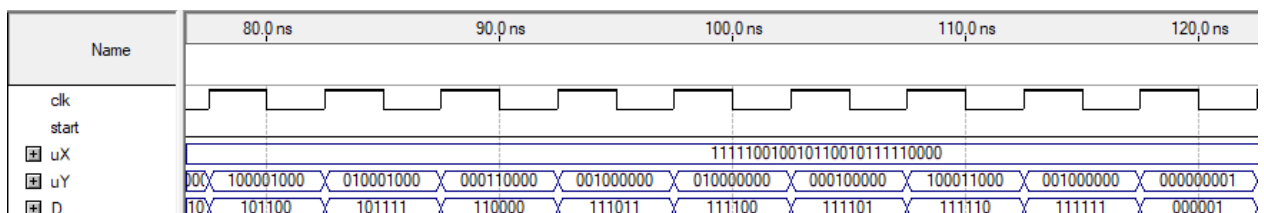


Рисунок Г.12 - Часова діаграма роботи третьої частини ГСА рис. 2.6 (другого резервного блоку) зі значенням входів uX в «111110010010110010111110000»

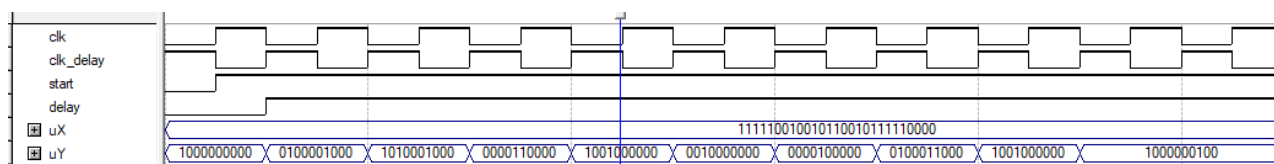


Рисунок Г.13 - Часова діаграма роботи першої частини ГСА рис. 2.4 (основного блоку) зі значенням входів uX в «111110010010110010111110000»

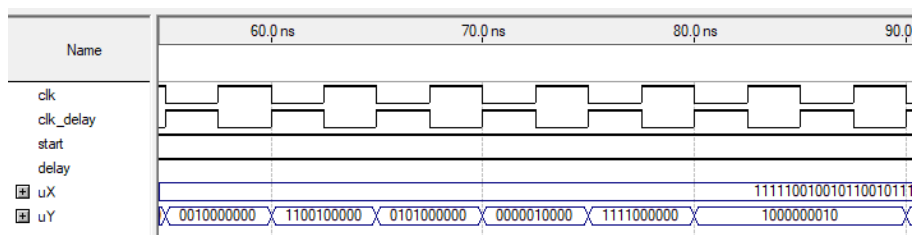


Рисунок Г.14 - Часова діаграма роботи другої частини ГСА рис. 2.5 (першого резервного блоку) зі значенням входів uX в «111110010010110010111110000»

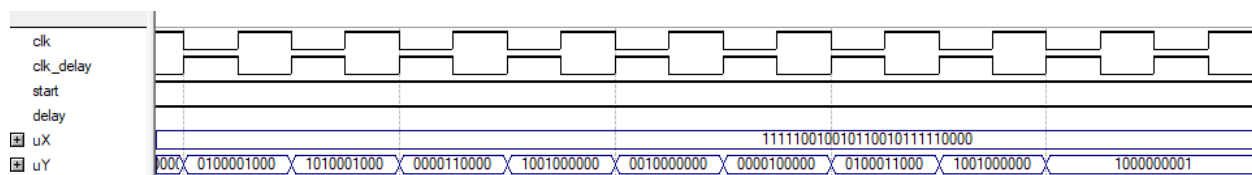


Рисунок Г.15 - Часова діаграма роботи третьої частини ГСА рис. 2.6 (другого резервного блоку) зі значенням входів uX в «111110010010110010111110000»

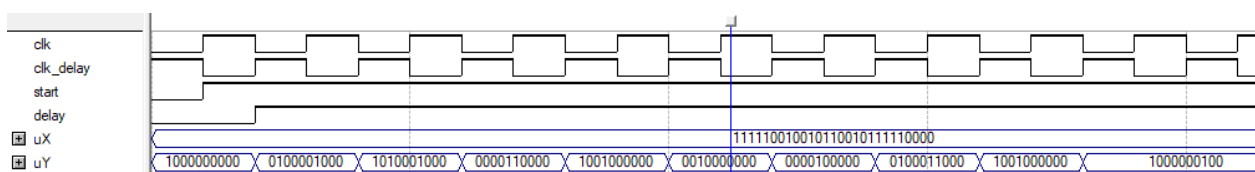


Рисунок Г.16 - Часова діаграма роботи першої частини ГСА рис. 2.4 (основного блоку) зі значенням входів uX в «111110010010110010111110000»

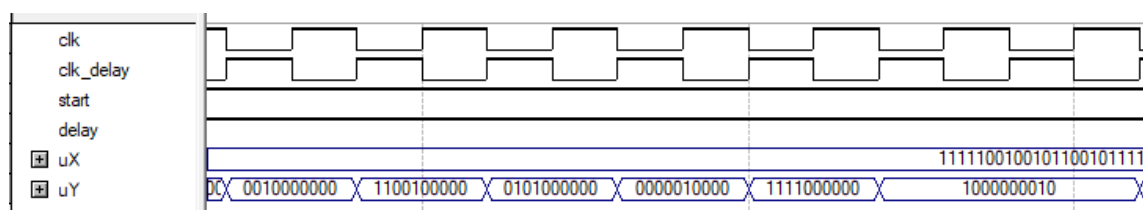


Рисунок Г.17 - Часова діаграма роботи другої частини ГСА рис. 2.5 (першого резервного блоку) зі значенням входів uX в «111110010010110010111110000»

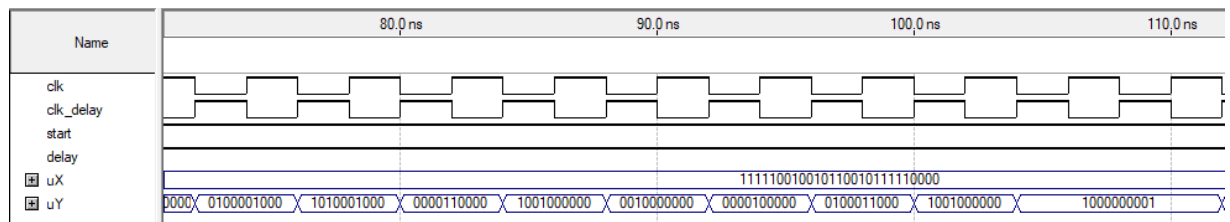


Рисунок Г.18 - Часова діаграма роботи третьої частини ГСА рис. 2.6 (другого резервного блоку) зі значенням входів uX в «111110010010110010111110000»

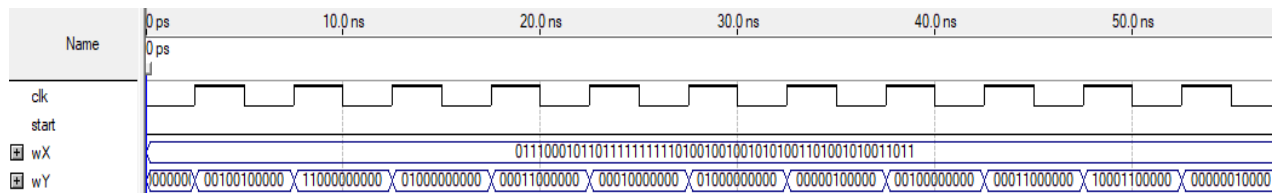


Рисунок Г.19 - Часова діаграма роботи першої частини ГСА рис. 2.7 (основного блоку) зі значенням входів uX в «1101100101001011001010100100100101111111111011010001110»

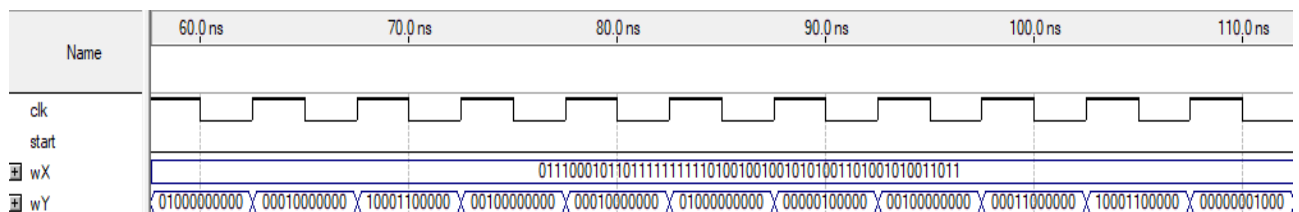


Рисунок Г.20 - Часова діаграма роботи другої частини ГСА рис. 2.8 (першого резервного блоку) зі значенням входів wX в «1101100101001011001010100100100101111111111011010001110»

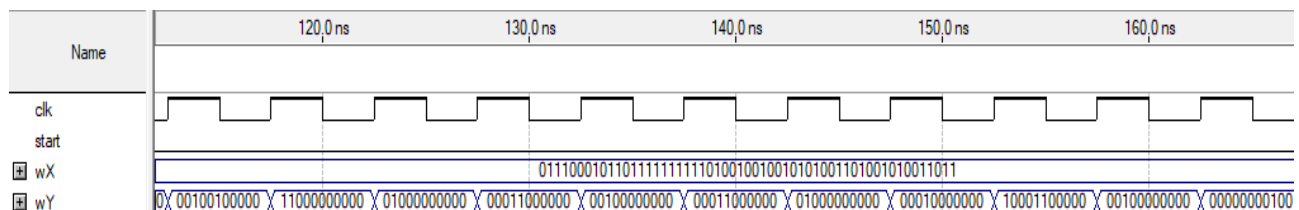


Рисунок Г.21 - Часова діаграма роботи третьої частини ГСА рис. 2.9 (другого резервного блоку) зі значенням входів wX в «1101100101001011001010100100100101111111111011010001110»

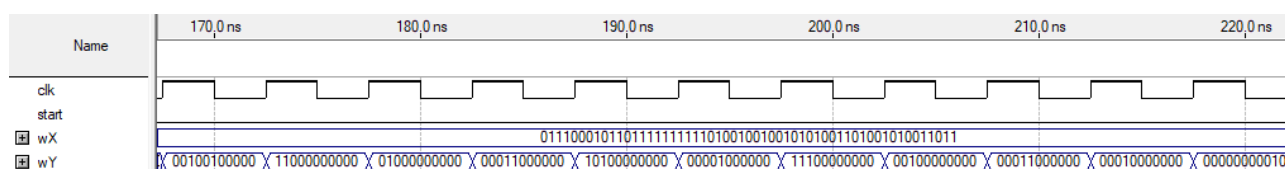


Рисунок Г.22 - Часова діаграма роботи четвертої частини ГСА рис. 2.10
(третього резервного блоку) зі значенням входів wX в
«1101100101001011001010100100100101111111111011010001110»

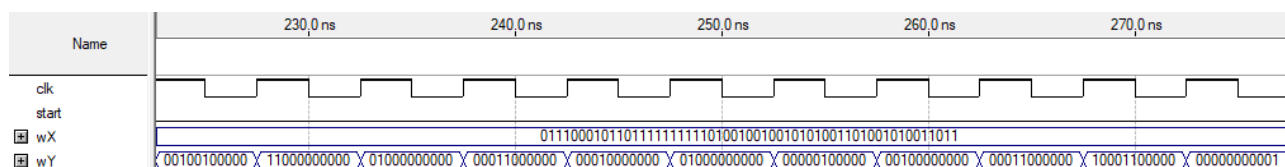


Рисунок Г.23 - Часова діаграма роботи п'ятої частини ГСА рис. 2.11
(четвертого резервного блоку) зі значенням входів wX в
«1101100101001011001010100100100101111111111011010001110»

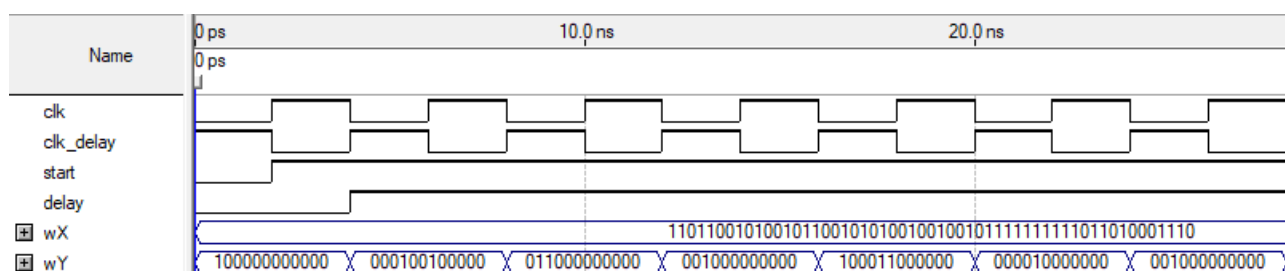


Рисунок Г.24 - Часова діаграма роботи першої частини ГСА рис. 2.7 (основного
блоку) зі значенням входів wX в
«1101100101001011001010100100100101111111111011010001110»

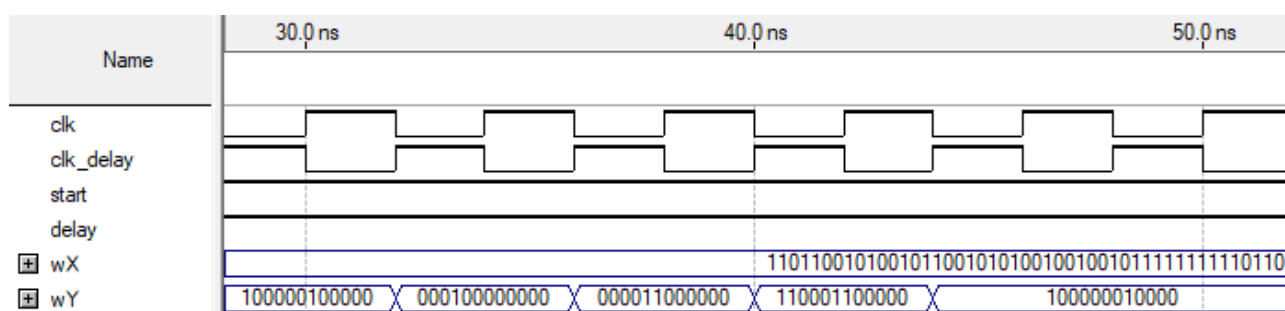


Рисунок Г.25 - Часова діаграма роботи першої частини ГСА рис. 2.7 (основного
блоку) зі значенням входів wX в
«1101100101001011001010100100100101111111111011010001110»

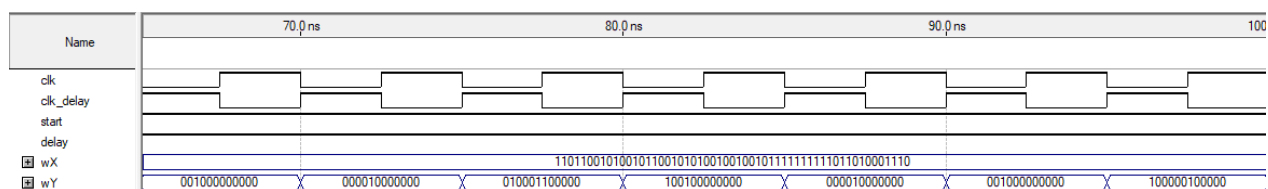


Рисунок Г.26 - Часова діаграма роботи другої частини ГСА рис. 2.8 (першого резервного блоку) зі значенням входів wX в
«11011001010010110010101001001001011111111111011010001110»

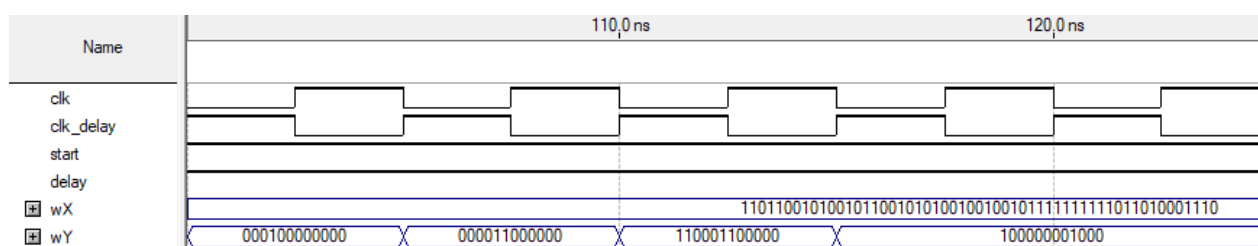


Рисунок Г.27 - Часова діаграма роботи другої частини ГСА рис. 2.8 (першого резервного блоку) зі значенням входів wX в
«11011001010010110010101001001001011111111111011010001110»

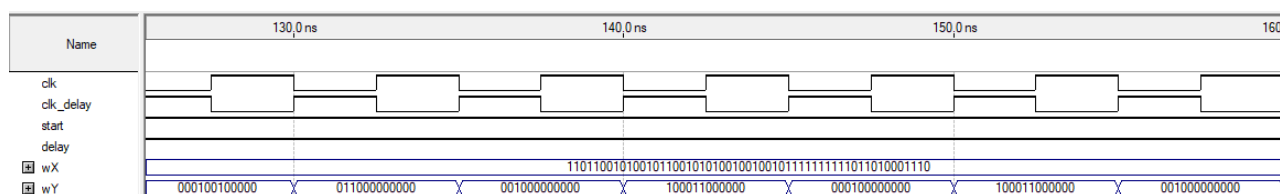


Рисунок Г.28 - Часова діаграма роботи третьої частини ГСА рис. 2.9 (другого резервного блоку) зі значенням входів wX в
«11011001010010110010101001001001011111111111011010001110»

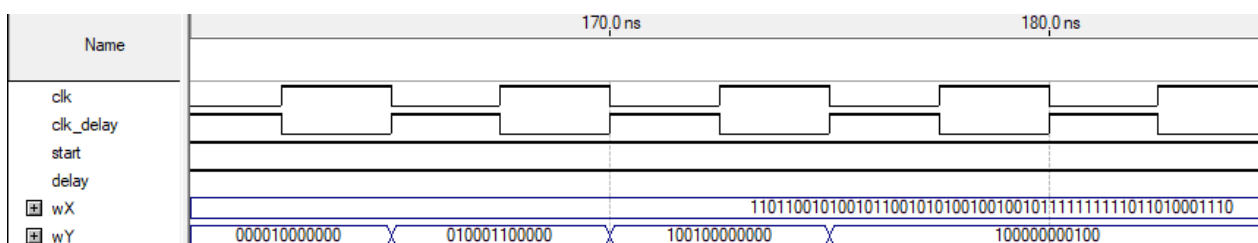


Рисунок Г.29 - Часова діаграма роботи третьої частини ГСА рис. 2.9 (другого резервного блоку) зі значенням входів wX в
«11011001010010110010101001001001011111111111011010001110»

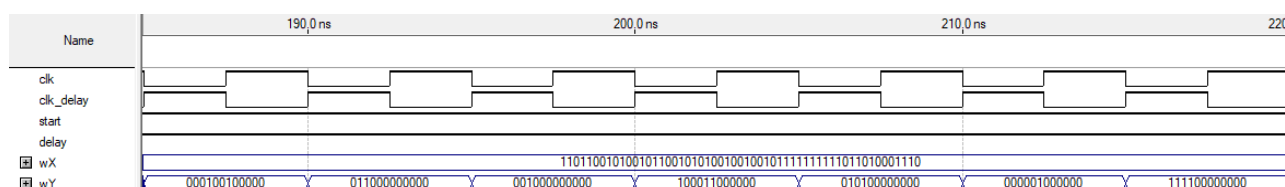


Рисунок Г.30 - Часова діаграма роботи четвертої частини ГСА рис. 2.10
(третього резервного блоку) зі значенням входів wX в
«11011001010010110010101001001001011111111111011010001110»

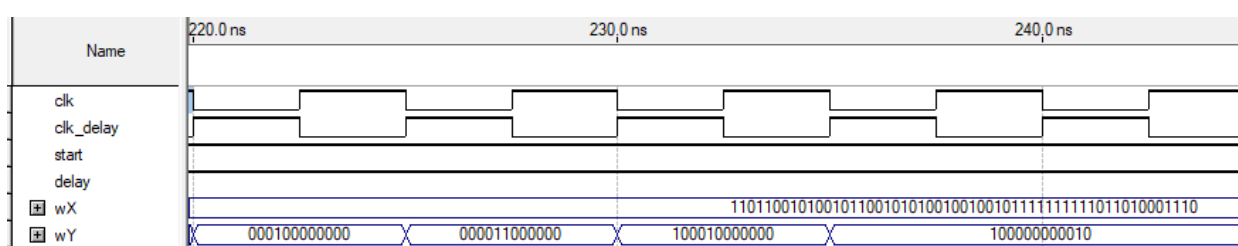


Рисунок Г.31 - Часова діаграма роботи четвертої частини ГСА рис. 2.10
(третього резервного блоку) зі значенням входів wX в
«11011001010010110010101001001001011111111111011010001110»

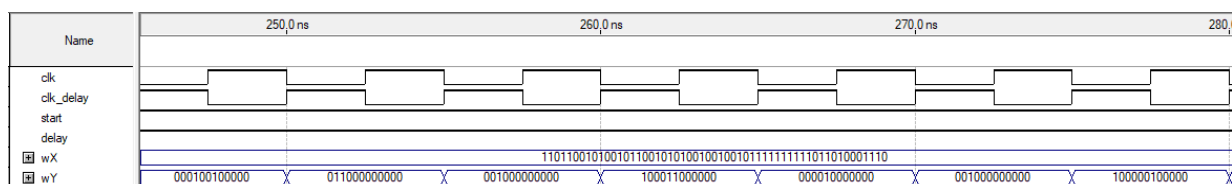


Рисунок Г.32 - Часова діаграма роботи п'ятої частини ГСА рис. 2.11
(четвертого резервного блоку) зі значенням входів wX в
«11011001010010110010101001001001011111111111011010001110»

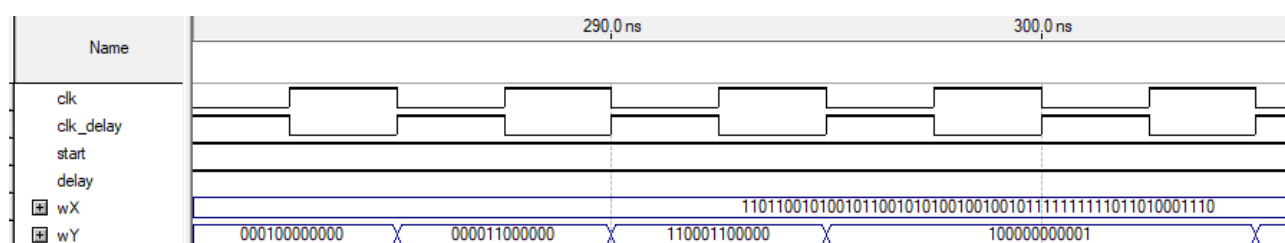


Рисунок Г.33 - Часова діаграма роботи п'ятої частини ГСА рис. 2.11
(четвертого резервного блоку) зі значенням входів wX в
«11011001010010110010101001001001011111111111011010001110»

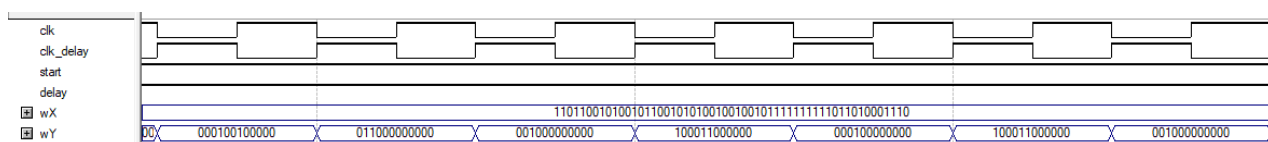


Рисунок Г.38 - Часова діаграма роботи третьої частини ГСА рис. 2.9 (другого резервного блоку) зі значенням входів wX в

«1101100101001011001010100100100101111111111011010001110»

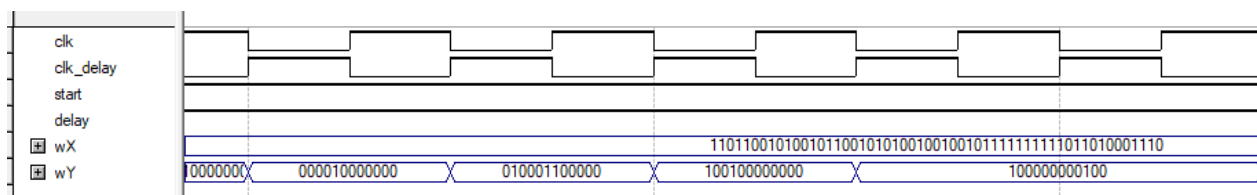


Рисунок Г.39 - Часова діаграма роботи третьої частини ГСА рис. 2.9 (другого резервного блоку) зі значенням входів wX в

«11011001010010110010110011111111111011010001110»

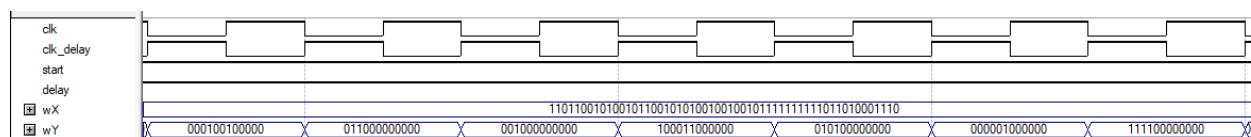


Рисунок Г.40 - Часова діаграма роботи четвертої частини ГСА рис. 2.10

(третього резервного блоку) зі значенням входів wX в

«11011001010010110010110011111111111011010001110»

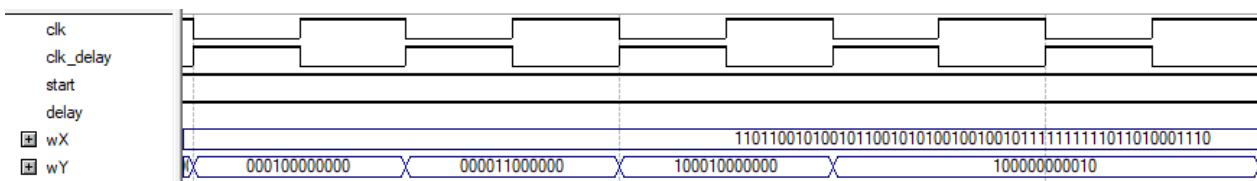


Рисунок Г.41 - Часова діаграма роботи четвертої частини ГСА рис. 2.10

(третього резервного блоку) зі значенням входів wX в

«11011001010010110010110011111111111011010001110»

