

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Комп'ютерні системи та мережі

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему Розробка спецобчислювача на FPGA
з комбінованою операційною частиною

Виконав: студент 4 курсу, групи КНТ-520
Спеціальності 123 «Комп'ютерна інженерія»

(код і найменування спеціальності)

Освітня програма (спеціалізація)

«Комп'ютерна інженерія»

БУТКО В. О.

(прізвище та ініціали)

Керівник ЗЕЛЕНЬОВА І. Я.

(прізвище та ініціали)

Рецензент КОЗІНА Г. Л.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування вищого навчального закладу)

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

ЗАТВЕРДЖУЮ

Завідувач кафедри КУДЕРМЕТОВ Р.К.

“ ” 2024 року

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

БУТКО Владислава Олексійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка спецобчислювача на FPGA з комбінованою операційною частиною

керівник проєкту (роботи) доцент, ЗЕЛЕНЬОВА Ірина Яківна
 (науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом вищого навчального закладу від “01” квітня 2024 року № 105

2. Строк подання студентом проєкту (роботи) 5 червня 2024 року

3. Вихідні дані до проєкту (роботи) теорія проєктування операційного автомата з комбінованою структурою, теорія проєктування керуючого автомата Мура, програмне середовище Vivado, віртуальна модель цільової FPGA

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Огляд структур операційних автоматів в процесорних ядрах;

2) Класифікація операційних автоматів розглянутих процесорних ядер;

3) Теоретичне проєктування операційного автомата з комбінованою структурою;

4) Розробка керуючого автомата

5) Проєктування операційного автомата та розробка спецобчислювача

6) Апаратна реалізація спецобчислювача на базі віртуальної моделі FPGA

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

Плакат 1 – Функціональна граф-схема алгоритму спецобчислювача

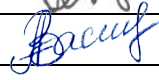
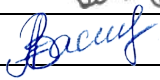
Плакат 2 – Функціональна схема операційного автомата

Плакат 3 – Схематичне зображення типів операційних автоматів згідно з базовою класифікацією

Плакат 4 – Результат тестування спецобчислювача (ліва гілка алгоритму)

Плакат 5 – Результат тестування спецобчислювача (права гілка алгоритму)

6. Консультанти розділів проєкту (роботи)

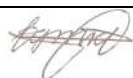
Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5	ЗЕЛЕНЬОВА І. Я., доцент		
нормоконтроль	ПОЛЬСЬКА О. В., ст. викл.		

7. Дата видачі завдання 01.03.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Побудова функціональної граф-схеми алгоритму	01.03.2024-10.03.2024	
2	Проектування операційного автомата з комбінованою структурою	10.03.2024-18.03.2024	
3	Огляд структур операційних автоматів в процесорних ядрах та їх класифікація	18.03.2024-17.05.2024	
4	Розробка керуючого автомата та проектування спецобчислювача	17.05.2024-19.05.2024	
5	Тестування та апаратна реалізація спецобчислювача на базі віртуальної моделі FPGA	19.05.2024-03.06.2024	
6	Оформлення пояснювальної записки	03.06.2024-05.06.2024	

Студент


 (підпис)
Владислав БУТКО

(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)


 (підпис)
Ірина ЗЕЛЕНЬОВА

(Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

ПЗ: 79 с.; 43 рис.; 17 табл.; 40 джерел.

ОПЕРАЦІЙНИЙ АВТОМАТ, КЕРУЮЧИЙ АВТОМАТ, ПРОЦЕСОРНЕ ЯДРО, FPGA, VIVADO, XILINX.

Об'єкт дослідження – операційний автомат з комбінованою структурою.

Предмет дослідження – структури операційних автоматів.

Мета роботи – апаратна реалізація спецобчислювача з комбінованою операційною частиною на базі віртуальної моделі FPGA.

Матеріали, методи та технічні засоби: теорія проєктування операційного автомата з комбінованою структурою, теорія проєктування керуючого автомата Мура, програмне середовище Vivado v2023.2, віртуальна модель FPGA (xc7k70tfbv676-1), персональний комп'ютер з процесором AMD Ryzen 7 5825U під управлінням операційної системи Microsoft Windows 10.

Результати. Власна класифікація операційних автоматів. Апаратна реалізація спецобчислювача на базі віртуальної моделі FPGA. Функціональна схема операційного автомата з комбінованою структурою.

ЗМІСТ

Вступ.....	7
1 Аналіз предметної області.....	9
1.1 Базова класифікація операційних автоматів	9
1.2 Огляд будови операційних автоматів в процесорних ядрах.....	11
1.2.1 Ядро процесора Intel 8008	11
1.2.2 Ядро процесора Elbrus 2000	12
1.2.3 Ядро серії процесорів TMS320C6x.....	12
1.2.4 Ядро Intel Synny Cove	13
1.2.5 Ядро ARM Cortex A8	14
1.2.6 Ядро ARM Cortex M0+	15
1.2.7 Інші ядра.....	17
1.3 Огляд елементної бази та інструментів проєктування.....	17
1.4 Постановка задачі.....	19
2 Порядок класифікації структур операційних автоматів.....	19
2.1 Пропозиція класифікації операційних автоматів.....	19
2.2 Критерії класифікації.....	20
2.3 Класифікація операційних автоматів розглянутих процесорних ядер.....	22
3 Теоретичне проєктування операційного автомата з комбінованою структурою.....	26
3.1 Формування пам'яті.....	27
3.2 Розбиття множини операцій на двомісні та одномісні.....	28
3.3 Розподіл регістрів по шинах.....	28
3.4 Формування операторів.....	29
3.5 Декомпозиція операцій граф-схеми алгоритму.....	32
3.6 Створення функціональної схеми операційного автомата.....	33
4 Розробка керуючого автомата Мура	34
4.1 Теоретичне проєктування.....	34

4.2 Проєктування.....	40
4.3 Тестування.....	43
5 Розробка спецобчислювача.....	45
5.1 Проєктування.....	45
5.2 Тестування.....	48
5.2.1 Перевірка гілки алгоритму, де умова виконується.....	48
5.2.2 Перевірка гілки алгоритму, де умова не виконується.....	56
5.3 Апаратна реалізація.....	61
Висновки	70
Перелік джерел посилання	71

ВСТУП

Спецобчислювач – це пристрій, що складається з двох частин: операційної та керуючої [1, с. 269]. Операційна частина (операційний автомат, ОА) – це обчислювач, який виконує певні операції згідно з керуючими сигналами. Керуюча частина (керуючий автомат, КА) – це пристрій керування, який під дією зовнішніх впливів (вхідних умов) генерує певну послідовність керуючих сигналів. Тож, певній вхідній умові відповідає певна послідовність керуючих сигналів. Порядок генерації такої послідовності ґрунтується на алгоритмі, що закладено при проєктуванні керуючого автомата. Таким чином, у відповідь на зовнішні впливи спецобчислювач виконує відповідні операції згідно з закладеним алгоритмом.

Спецобчислювачі покликані автоматизувати обчислення на апаратному рівні [2, с. 5]. Це дозволяє полегшити роботу програміста або користувача певного обчислювача без зниження швидкодії спецобчислювача.

Програміст може працювати на більш високому рівні абстракції, використовуючи високорівневі інструкції, які при виконанні декодуються (на апаратному рівні) в послідовність більш простих інструкцій. Для користувача це корисно з огляду на зменшення кількості команд, які він повинен віддати системі керування для виконання певної дії.

У порівнянні з автоматизацією на програмному рівні, при апаратній автоматизації унікається етап трансляції, що потребує більше часу, аніж процес декодування. Адже, при трансляції необхідно виконати дві програми: програму-транслятор та виконавчу програму на машинній мові (результат трансляції). Отже, при трансляції виконуються додаткові повільні команди доступу до пам'яті, які застосовує як програма-транслятор для завантаження високорівневої програми, так і виконавча програма.

Тому, апаратна автоматизація застосовується до найчастіше використовуваних дій в будь-якому сучасному процесорному ядрі, що

підтверджується наявністю в них блоку декодування інструкцій в частині «front end».

Тобто, даний блок виконує функцію КА в процесорному ядрі. Функцію ОА виконує тракт даних (англ. data path), що розміщується в частині «execution engine» та складається з регістрового файлу загального призначення, обчислювальних блоків (англ. execution unit) та шини результату.

Обчислювальний блок – апаратний блок, призначений для обчислення деякої множини операцій. Блок може мати один або більше виходів, на яких формується результат операцій. При наявності декількох виходів, вони можуть підтримувати послідовний або паралельний способи роботи. Підтримка паралельного способу свідчить про можливість паралельного виконання кількох операцій.

Отже, спецобчислювач є прикладом процесорного ядра з обмеженим набором мікрооперацій. Тому аналіз структури, класифікація, розробка є актуальними задачами, що дозволять набути практичних навичок з проєктування основних компонентів будь-якого сучасного процесорного ядра.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Базова класифікація операційних автоматів

Базова класифікація [3, с. 216] виконана на основі відношення швидкодії до ресурсомісткості ОА. Далі наведено опис типів ОА, що виділяє дана класифікація:

- М-автомат (mutual micro-operations; рис. 1.1, а): виконує операції послідовно. Характеристика автомата: найменша ресурсомісткість за рахунок використання лише однієї шини запису результату до регістрів та двох шин читання операндів з регістрів; найменше енергоспоживання за рахунок використання лише одного порту запису та до двох портів читання на регістр; найменша швидкодія через послідовне виконання операцій;

- ІМр-автомат (individual-mutual parallel micro-operations; рис. 1.1, б): виконує до двох операцій паралельно. Характеристика автомата відносно М-автомата: більша ресурсомісткість через використання двох шин результату та трьох шин читання операндів з регістрів; більше енергоспоживання через використання до двох портів запису та трьох портів читання з регістрів; більша швидкодія за рахунок паралельного виконання операцій;

- ІМс-автомат (individual-mutual sequential micro-operations; рис. 1.1, в): виконує операції послідовно в конвеєрі. Характеризовано автомат відносно ІМр-автомата. Незважаючи на меншу кількість використаних шин запису результату та читання операндів, при наявності конвеєра з кількістю етапів більше двох, ресурсомісткість стає більшою через виокремлення по одному апаратному блоку на один етап конвеєра. Більше енергоспоживання через наявність внутрішніх портів запису/читання між блоками, що реалізують етапи конвеєра. Більша швидкодія за рахунок збільшення швидкої в n разів при збільшенні кількості етапів конвеєра в n рази.

- І-автомат (individual micro-operations; рис. 1.1, г): може виконувати максимальну можливу кількість операцій паралельно. Характеристика автомата: найбільша ресурсомісткість та енергоспоживання через досягнення максимального

паралелізму на апаратному рівні; найбільша швидкодія за рахунок досягнення максимального паралелізму при виконанні операцій.

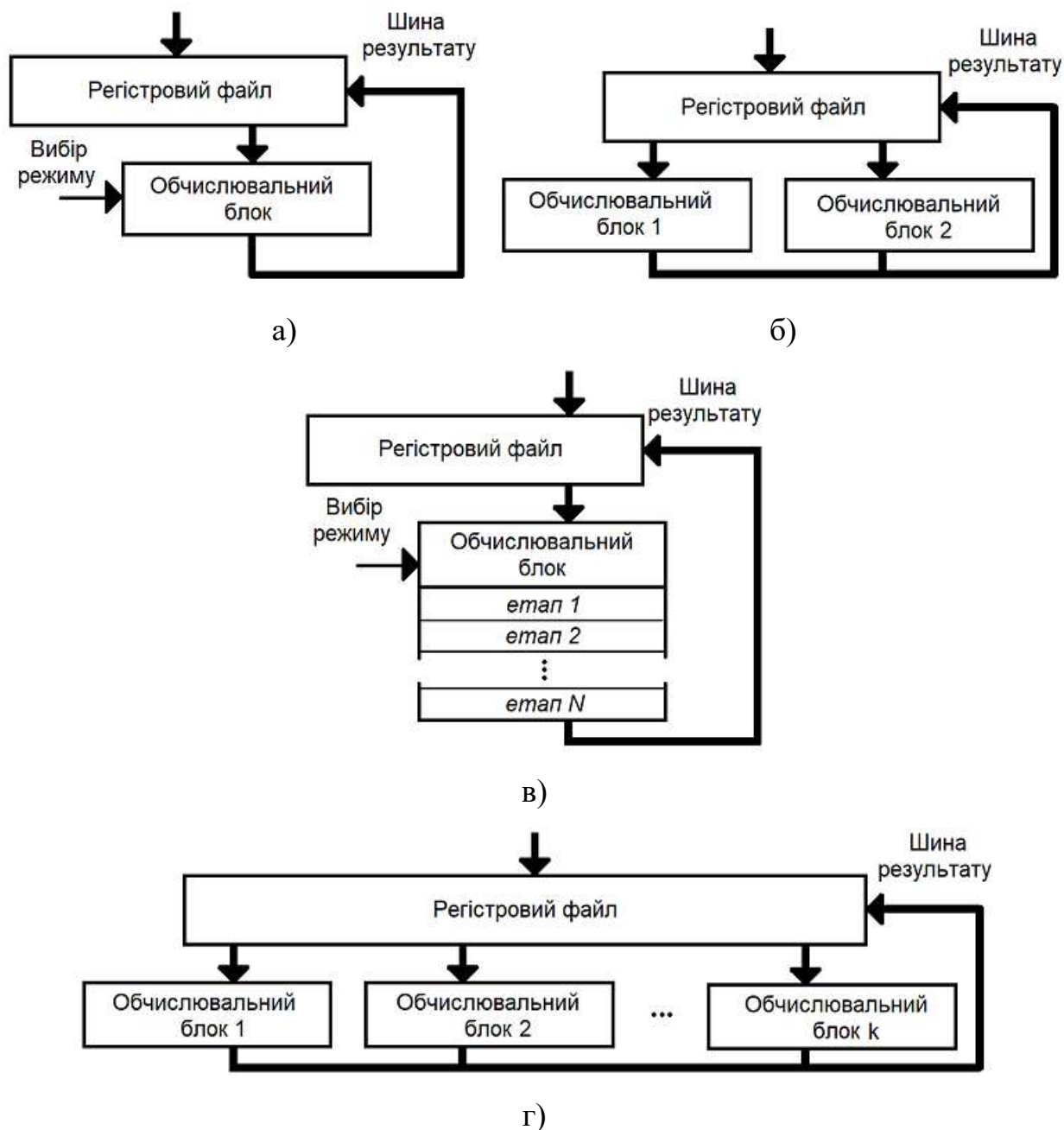


Рисунок 1.1 – Схематичне зображення типів ОА:

а) М-автомат; б) ІМр-автомат; в) ІМs-автомат; г) І-автомат

Далі, задля перевірки актуальності даної класифікації, виконується огляд будови ОА, зокрема, сучасних процесорних ядер.

1.2 Огляд будови операційних автоматів в процесорних ядрах

1.2.1 Ядро процесора Intel 8008 [4, с. 554]

Процесорне ядро складається з компонентів (рис. 1.2): регістри загального призначення, арифметико-логічний пристрій (АЛП), АЛП регістри, шина даних, блок декодування інструкцій, лічильник команд (блок «Shifter»), блок прискорення перенесення.

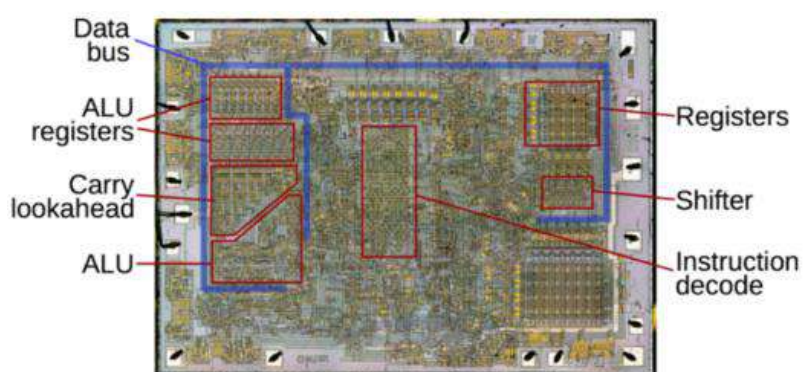


Рисунок 1.2 – Компоненти ядра процесора Intel 8008 [5]

З перелічених компонентів, лише блок декодування інструкцій та лічильник команд не відносяться до складових частин ОА. Тож, зі складових ОА лише АЛП та блок прискорення перенесення, що працюють взаємоузгоджено, відповідають за обчислення операцій ОА. Отже, ядро Intel 8008 налічує лише один обчислювальний блок. Розглянуто логічну схему АЛП (рис. 1.3).

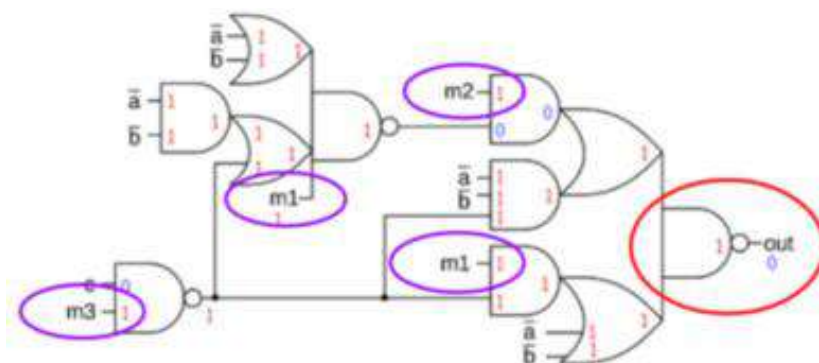


Рисунок 1.3 – Логічна схема АЛП в процесорному ядрі Intel 8008 [5]

Згідно з рис. 1.3, схема має лише один вихід (виділено червоним), що свідчить про послідовний характер виконання операцій в ОА. Серед входів, логічна схема має входи обрання режиму (виділено фіолетовим), що свідчить про відсутність конвеєра – в кожному машинний цикл АЛП виконує лише одну операцію.

1.2.2 Ядро процесора Elbrus 2000 [4, с. 554]

Розглянуто виконавчий конвеєр (рис. 1.4).



Рисунок 1.4 – Виконавчий конвеєр в процесорному ядрі E2K [6, с. 3]

Згідно з рис. 1.4, процесорне ядро має виконавчий конвеєр, що після етапу читання операндів з пам'яті, поділяється на три конвеєри, кожен з яких виконує певний клас операцій: цілочисельні, з плаваючою комою та операції завантаження. Тож, в залежності від виконуваної операції її обробка продовжується з використанням відповідного конвеєра. Кожному конвеєру відповідає окремий обчислювальний блок. Тож, ОА налічує три обчислювальні блоки.

Конвеєр цілочисельних операцій складається з двох етапів: виконання операції та запису результату. Тож, етап виконання в даному конвеєрі не конвеєризований. Конвеєр операцій з плаваючою комою складається з фаз етапу виконання. Тож, етап виконання в даному конвеєрі конвеєризований.

1.2.3 Ядро серії процесорів TMS320C6x [4, с. 555]

Процесорне ядро має виконавчий конвеєр, що складається з трьох етапів [7, с. 6-2]: завантаження (англ. fetch), декодування та виконання інструкції. На етапі виконання інструкції вона передається до одного з конвеєрів на рівні

обчислюваних блоків. Наявні два таких конвеєра [7, с. 6-5]. Один в блоці для обчислення операндів з плаваючою комою (англ. floating point), інший – в блоці для обчислення цілочисельних операндів (англ. fixed point або integer).

Етап виконання в блоці «floating point» поділений на десять фаз, в блоці «integer» – на п'ять фаз. Розглянуто табл. 1.1, що є фрагментом таблиці [7, с. 6-15].

Таблиця 1.1 – Фази етапу виконання операції MPYID

Фаза	Опис
E1	Читання операндів та початок виконання операції
E2	Читання операндів та продовження виконання операції
...	...
E10	Завершення виконання та запис верхніх результатів до регістру

Згідно з табл. 1.1, в першій фазі починається виконання операції, а в другій – продовжується. Отже, етап виконання операції дійсно поділений на фази. Тобто, етап виконання конвеєризований.

1.2.4 Ядро Intel Sunny Cove

На рис. 1.5 виділено червоним два обчислювальні блоки, які виконують операції завантаження. Тому як вони під'єднані до різних портів планувальника (англ. sheulder), то можна стверджувати, що ядро Sunny Cove містить, принаймні, два апаратно різних (хоча і однакових за функціоналом) обчислювальні блоки.

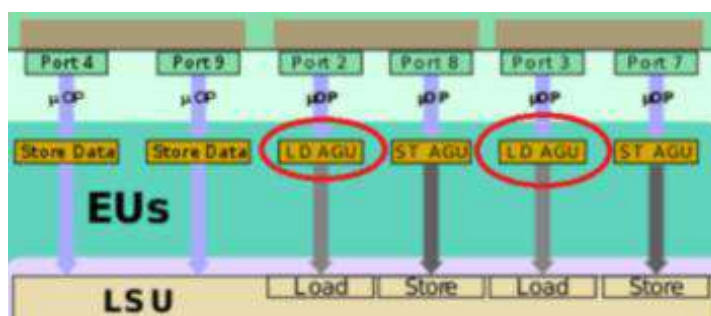


Рисунок 1.5 – Блоки завантаження в ядрі Sunny Cove (зверху порти планувальника) [8]

Інформації щодо конвеєризації на етапі виконання операцій не знайдено. Хоча, з огляду на рис. 1.6, можна припустити наявність фаз на етапі виконання операцій, але більш детальний огляд дозволяє з'ясувати, що це обчислювальні блоки. Кожний порт планувальника під'єднаний до сукупності обчислювальних блоків. Серед них можна зустріти як цілочисельні, так і векторні блоки.

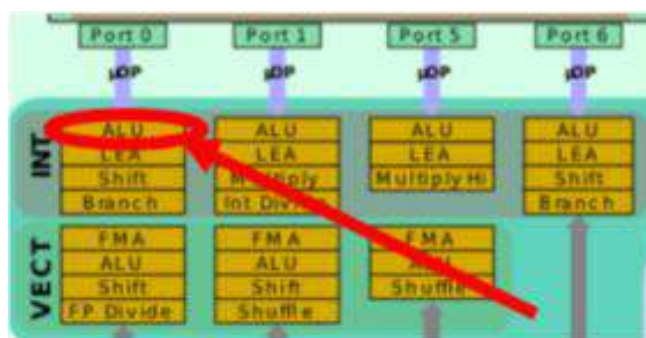


Рисунок 1.6 – Виділено обчислювальний блок АЛП в ядрі Sunny Cove [8]

Хоча в деяких джерелах [9] щодо подібних структур можна зустріти фразу вигляду «ALU ріре» (пер. АЛП конвеєр). Проте, в даному випадку, під словом «ріре» мається на увазі обчислювальний блок.

1.2.5 Ядро ARM Cortex A8

Згідно з рис. 1.7 можна переконатись, що процесорне ядро містить, принаймні, один обчислювальний блок – «floating point».

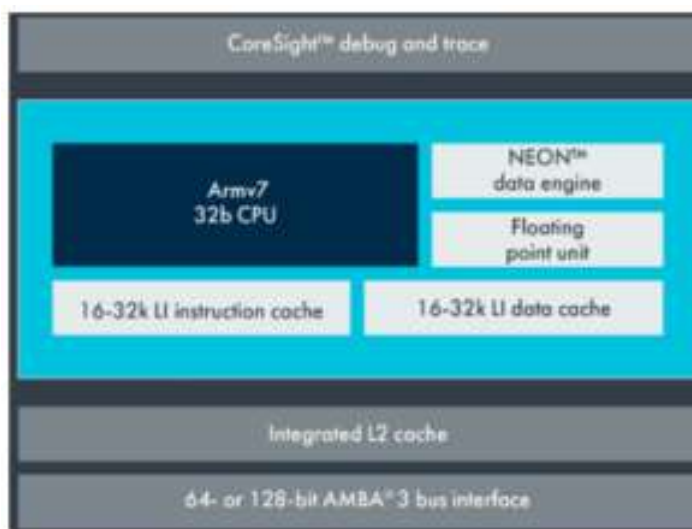


Рисунок 1.7 –Компоненти ядра ARM Cortex A8 [10]

Для підтвердження гіпотези того, що блок «NEON» є обчислювальним, розглянуто внутрішню будову даного блока (рис. 1.8).

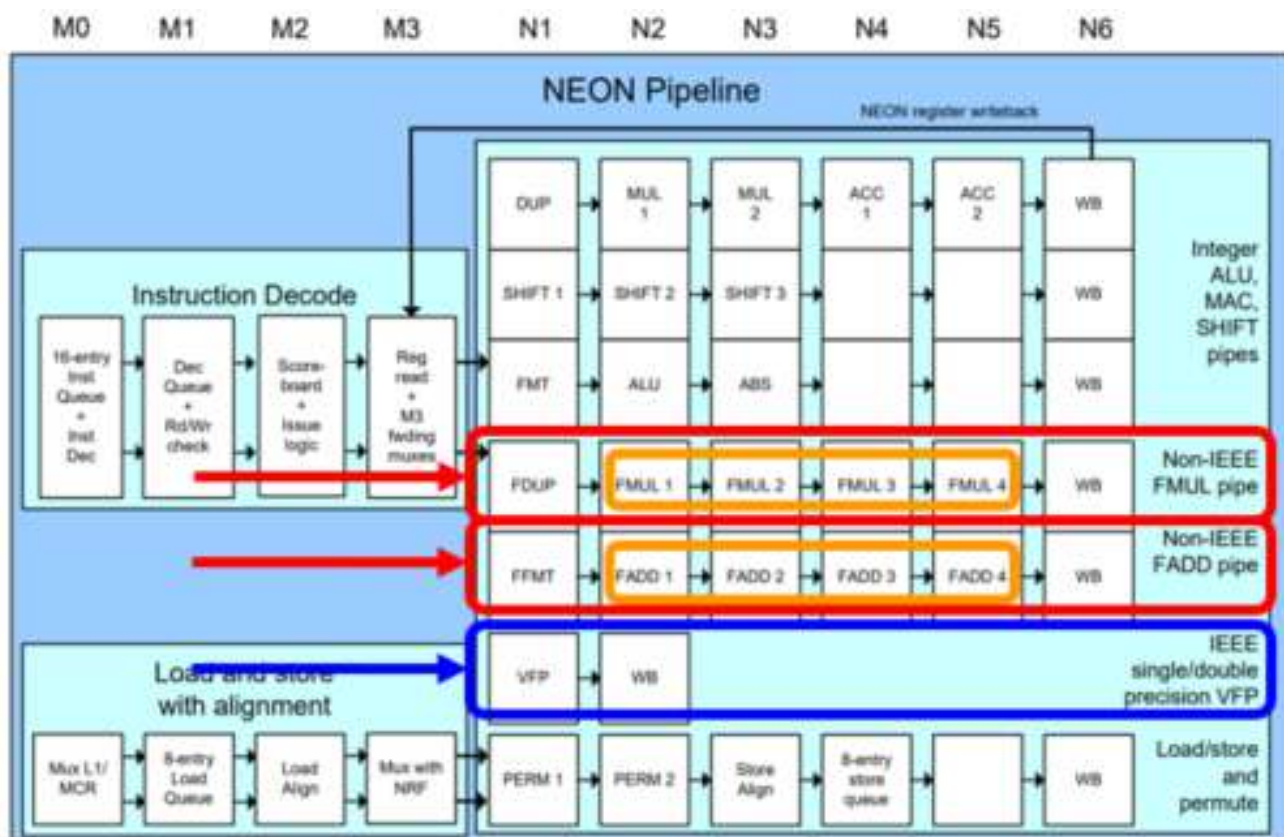


Рисунок 1.8 – Конвертовані (виділені червоним) та не конвертовані (виділені синім) операції в блоці «NEON» [11, с. 7]

Згідно з рис. 1.8, даний блок виконує, зокрема, наступні операції ОА: FMUL та FADD. Тож, можна стверджувати, що блок «NEON» є обчислювальним. Також, можна стверджувати про наявність, принаймні, двох конвеєрів: FMUL та FADD.

Серед етапів конвеєра FMUL можна виділити такі, що не відносяться безпосередньо до процесу виконання операції: FDUP та WB. А також, такі, що відносяться: FMUL1-4. Це перелік наявних фаз етапу виконання операції FMUL.

1.2.6 Ядро ARM Cortex M0+

Ядро має виконавчий конвеєр з двома етапами (рис. 1.9). Відповідно до рисунка можна виділити один обчислювальний блок (АЛП) та один потенційний обчислювальний блок (блок «Shift»).

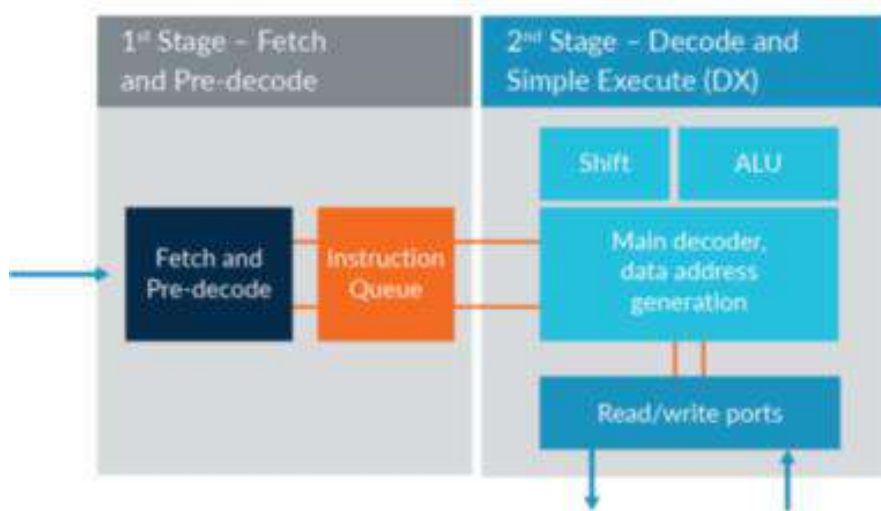


Рисунок 1.9 – Виконавчий конвеєр з двома етапами [12, с. 5]

Цитата: «Whilst the exact architecture of the Cortex-M processor is not publicly accessible, most public sources show a diagram as Figure 1 [Fur00]» [13, с. 7] (пер. «хоча точна архітектура ядер серії Cortex-M не є загальнодоступною, більшість публічних джерел наводять діаграму як на Рисунку 1 [Fur00]»). Мається на увазі рис. 1.10.

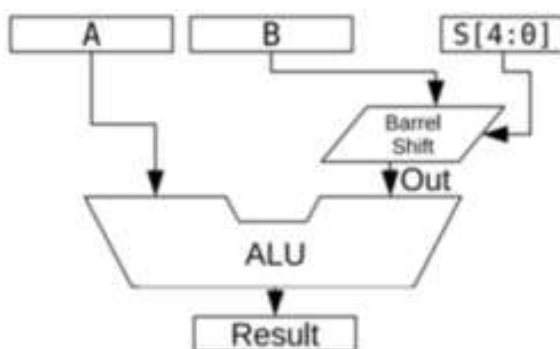


Рисунок 1.10 – Схематичне зображення частини «execution engine» в ядрах серії Cortex-M [13, с. 7]

Згідно з рис. 1.10, блоки АЛП та «Shift» під'єднуються послідовно. Тож, дані блоки працюють взаємозалежно та утворюють єдиний обчислювальний блок. Також, можна спостерігати, що АЛП має один вихід.

На рис. 1.11 наведено загальний випадок виконання інструкцій у виконавчому конвеєрі.

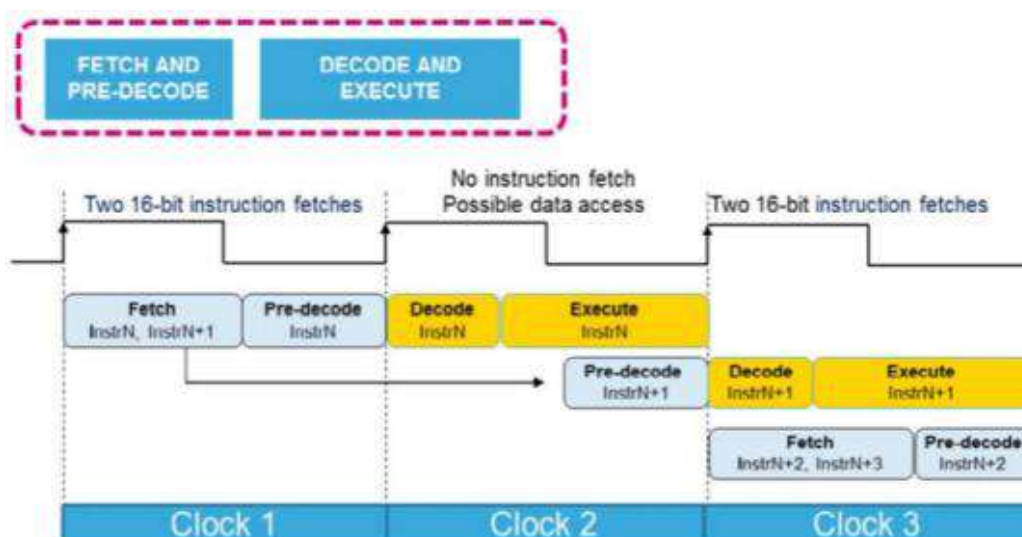


Рисунок 1.11 – Приклад роботи виконавчого конвеєра [14, с. 5]

Згідно з рис. 1.11, пересвідчимося, що етап виконання не поділений на фази. На другому такті на етапі виконання виконується інструкція «InstrN». На третьому – «InstrN+1». Тобто, наступна інструкція. Тож, в кожний момент часу в конвеєрі оброблюється декілька інструкцій на етапі виконання. Якби етап виконання був поділений на фази, тоді в першому та другому машинних циклах виконувалась би однакова інструкція.

1.2.7 Інші ядра [4, с. 555]

Розглянуто ОА в ядрах AMD Zen 1, 2, 4 та Intel Skylake. На основі блок-діаграм відповідних ядер [15, 16, 17, 18] визначено, що вони мають більше одного обчислювального блоку. Проте, інформації щодо поділу етапу виконання операцій на фази – не знайдено. Звідси, можна припустити, що конвеєризація на етапі виконання операцій відсутня.

1.3 Огляд елементної бази та інструментів проєктування

Елементною базою апаратної реалізації спецобчислювача є FPGA. Цитата: «FPGA (англ. Field-Programmable Gate Array) – це інтегральна схема, що, головним

чином, складається з логічних блоків та реконфігуруємих з'єднань, які можна програмувати за допомогою мов опису апаратури (англ. HDL)» [19, с. 40; 20, с. 53].

Цитата: «Мови опису апаратури – це високорівневі мови, що дозволяють описувати структуру та поведінку електронних схем. Найбільш поширеними мовами HDL на теперішній час є Verilog, VHDL та SystemVerilog. Всі три мови є індустріальними стандартами IEEE» [19, с. 41; 21, с. 2].

Цитата: «Перевагами використання Verilog є:

- можливість використовувати примітиви, які дозволяють виконувати опис схем на рівні вентилів, який є найнижчим для FPGA;
- слаботипізованість, яка дозволяє швидше та лаконічніше виконувати опис схеми, але вимагає більшої уваги від розробника;
- синтаксична схожість з мовою програмування Сі» [19, с. 41; 22].

Цитата: «Описи на мовах HDL можуть бути верифіковані за допомогою часових діаграм, створених з використанням симуляторів, або за допомогою статичної часової верифікації, що є більш зручним у міру ускладнення схеми, що проектується. На основі таких описів також можна синтезувати список з'єднань (англ. netlist), який, зокрема, можна перетворити у графічну схему та переглянути синтезовану логічну схему» [19, с. 41]. Також, на основі синтезованого списку з'єднань виконується етап імплементації. Імплементація – це один із етапів перетворення HDL в бітовий потік для конфігурації FPGA, на якому визначається перелік необхідних ресурсів FPGA (для апаратної реалізації HDL опису) та їх розташування на конкретній віртуальній моделі FPGA.

Результатом імплементації є звіт з імплементації, в якому міститься інформація про характеристики апаратної реалізації на базі FPGA. Серед таких характеристик є кількість використаних ресурсів FPGA, часові характеристики (наприклад, затримки встановлення/утримання) та показники живлення апаратної реалізації. В даній роботі верифікація, синтез та імплементація виконуються в середовищі Vivado (v2023.2) [23], бо в якості платформи для апаратної реалізації використовується FPGA фірми Xilinx/AMD [24].

1.4 Постановка задачі

Згідно з технічним завданням, необхідно розробити та апаратно реалізувати спецобчислювач, що обчислює задану формулу:

$$S = \begin{cases} (9(A \oplus B) + (4A + \frac{B}{8})) \oplus \overline{(5B + A)}, & A \leq 25 \\ \frac{A+B}{8} + \overline{(4A \wedge 2B)} \vee (A \oplus B) \end{cases} \quad (1.1)$$

Розрядність регістрів ОА в спецобчислювачі повинна дорівнювати восьми. ОА спецобчислювача повинен бути типу ІМр, згідно з базовою класифікацією. Результатом розробки ОА повинна бути його функціональна схема. Апаратна реалізація спецобчислювача може бути виконана на базі віртуальної моделі FPGA.

2 ПОРЯДОК КЛАСИФІКАЦІЇ СТРУКТУР ОПЕРАЦІЙНИХ АВТОМАТІВ

2.1 Пропозиція класифікації операційних автоматів

В результаті аналізу предметної області розглянуто структури ОА в деяких процесорних ядрах та з'ясовано, що базова класифікація є не актуальною через неможливість класифікувати за її допомогою ОА в сучасних процесорних ядрах. Тому, запропоновано власну класифікацію ОА. Нова класифікація утворена за ступенем паралелізму виконання операцій. Опис трьох типів ОА [4, с. 553-554]:

- S-автомат (sequential micro-operations; див. рис. 1.1, а): може виконувати операції лише послідовно. Характеристика автомата: найменші ресурсомісткість та енергоспоживання за рахунок відсутності паралелізму на апаратному рівні; найменша швидкодія через послідовне виконання операцій;

- Р-автомат (parallel micro-operations; див. рис. 1.1, б): виконує операції паралельно. Характеристика автомата відносно S-автомата: більші ресурсомісткість та енергоспоживання через використання більшої кількості обчислювальних блоків; більша швидкодія за рахунок паралельного виконання операцій;

- РС-автомат (parallel-conveyor micro-operations; рис. 2.1): виконує операції в паралельних конвеєрах. Характеристика автомата відносно Р-автомата: більша ресурсомісткість через виділення окремого апаратного блоку для кожного етапу конвеєра; більше енергоспоживання через додаткові порти входу/виходу, які належать апаратним блокам конвеєра; більша швидкодія за рахунок застосування конвеєризації в обчислювальних блоках.

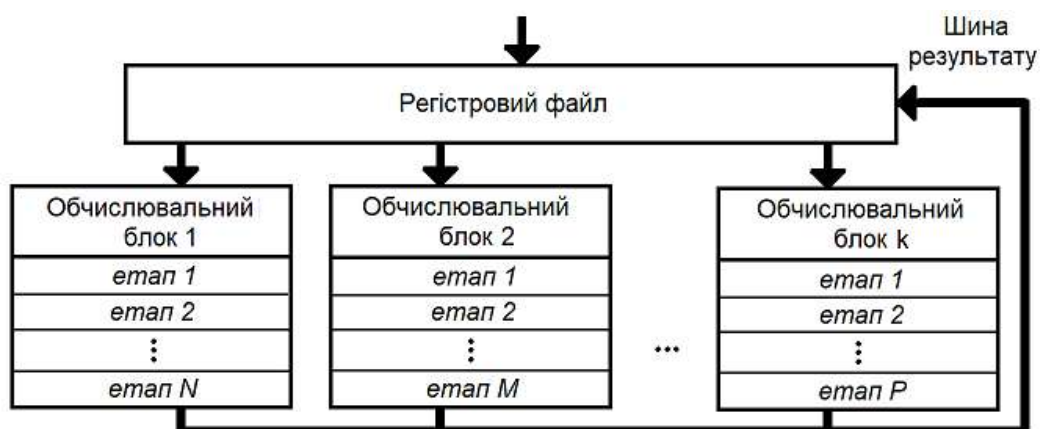


Рисунок 2.1 – Схематичне зображення ОА типу РС

2.2 Критерії класифікації

Для визначення типу ОА використовуються критерії класифікації ОА згідно з власною класифікацією [4, с. 554]:

- критерій перший: наявність зовнішнього паралелізму на рівні обчислювальних блоків; умова виконання: кількість обчислювальних блоків, що можуть працювати паралельно, більше одного;

- критерій другий: наявність внутрішнього паралелізму на рівні обчислювального блоку; умова виконання: кількість виходів, на яких формується результат операцій, обчислювального блоку, що можуть працювати паралельно, більше одного;

- критерій третій: наявність конвеєризації на етапі виконання операції; умова виконання: етап виконання операції, хоча б в одному обчислювальному блоці, поділений на фази.

Сукупність аналізу всіх критеріїв на предмет їх виконання дозволяє визначати тип ОА (табл. 2.1).

Таблиця 2.1 – Критерії класифікації ОА [4, с. 554]

№ набору	Тип ОА	Виконання критеріїв		
		1	2	3
1	S-автомат	Ні	Ні	Ні
2	Не визначено	Ні	Ні	Так
3	P-автомат	Ні	Так	Ні
4	РС-автомат	Ні	Так	Так
5	P-автомат	Так	–	Ні
6	РС-автомат	Так	–	Так

Згідно з табл. 2.1, розглянуто всі можливі комбінації критеріїв. Серед розглянутих процесорних ядер не зустрічались такі, що відповідали б наборам критеріїв 2, 3 та 4.

Якщо виконується перший критерій, то другий не розглядається. Тому що, перші два критерії покликані визначити наявність паралелізму в ОА загалом. Тож, не важливо, чи є він наявний на рівні обчислювальних блоків (використовується декілька обчислювальних блоків), на рівні обчислювального блоку

(обчислювальний блок має декілька виходів, на яких формується результат операцій, що можуть працювати паралельно) або на обох рівнях одночасно.

Якщо не виконується жоден з критеріїв, то автомат належить до типу S. При виконанні лише першого або другого критерію, автомат належить до типу P. При виконанні третього критерію та першого або другого, автомат належить до типу PC.

Отже, до S-автоматів належать ОА, що складаються з одного обчислювального блоку. Причому, цей блок має лише один вихід або, у разі наявності декількох виходів, вони не можуть працювати паралельно.

До P-автоматів належать ОА, що складаються з декількох обчислювальних блоків, в незалежності від кількості їх виходів. Також, ОА може складатися з одного обчислювального блоку, що має декілька виходів, які можуть працювати паралельно.

До PC-автоматів належать ОА, що складаються з декількох обчислювальних блоків, із яких хоча б в одному наявний конвеєр на етапі виконання. В іншому випадку, ОА може складатися з одного обчислювального блоку, що має декілька виходів, які можуть працювати паралельно. Причому, хоча б одна з операцій повинна виконуватись в конвеєрі на етапі виконання.

2.3 Класифікація операційних автоматів розглянутих процесорних ядер

На основі розглянутих процесорних ядрах в попередніх розділах виконано класифікацію їх ОА згідно з власною класифікацією (табл. 2.2-2.8).

Таблиця 2.2 – Перевірка критеріїв для ОА Intel 8008 [4, с. 554]

№ Крит.	Виконання	Обґрунтування
1	Ні	Один обчислювальний блок (АЛП)

Продовження таблиці 2.2

№ Крит.	Виконання	Обґрунтування
2	Ні	АЛП має один вихід
3	Ні	За машинний цикл АЛП виконує лише одну операцію

Таблиця 2.3 – Перевірка критеріїв для ОА Elbrus 2000 [4, с. 554]

№ Крит.	Виконання	Обґрунтування
1	Так	Три обчислювальних блоки (відповідно до наявних конвеєрів)
2	—	—
3	Так	В блоці «floating point» етап виконання поділений на n фаз

Таблиця 2.4 – Перевірка критеріїв для ОА TMS320C6x [4, с. 555]

№ Крит.	Виконання	Обґрунтування
1	Так	Два обчислювальних блоки: «floating point» та «integer»
2	—	—
3	Так	В блоці «floating point» етап виконання поділений на 10 фаз

Таблиця 2.5 – Перевірка критеріїв для ОА Intel Sunny Cove

№ Крит.	Виконання	Обґрунтування
1	Так	Принаймні, два обчислювальних блоки: апаратно різні блоки завантаження
2	—	—

Продовження таблиці 2.5

№ Крит.	Виконання	Обґрунтування
3	Ні	Інформації щодо конвеєризації не знайдено

Таблиця 2.6 – Перевірка критеріїв для ОА ARM Cortex A8

№ Крит.	Виконання	Обґрунтування
1	Так	Два обчислювальні блоки: «NEON» та «integer»
2	—	—
3	Так	Етап виконання поділений на фази FMUL1-4

Таблиця 2.7 – Перевірка критеріїв для ОА ARM Cortex M0+

№ Крит.	Виконання	Обґрунтування
1	Ні	Один обчислювальний блок АЛП, що працює у зв'язці з блоком «Shift»
2	Ні	АЛП має один вихід
3	Ні	В кожний момент часу в конвеєрі оброблюються різні інструкції на етапі виконання

Таблиця 2.8 – Перевірка критеріїв для ОА AMD Zen 1, 2, 4 та Intel Skylake [4, с. 555]

№ Крит.	Виконання	Обґрунтування
1	Так	Кількість обчислювальних блоків більше одного
2	—	—
3	Ні	Інформації щодо конвеєризації не знайдено

У табл. 2.9 підведено підсумки класифікації та огляду ОА в розглянутих процесорних ядрах. Таблицю відсортовано за роком випуску ядер.

Таблиця 2.9 – Підсумки класифікація та огляду ОА в розглянутих процесорних ядрах

Ядро	Рік випуску	Одноядерний пристрій ¹	Виконання критеріїв			Тип ОА
			1	2	3	
Intel 8008	1972 [5]	Intel 8008	Ні	Ні	Ні	S-автомат
TMS320C6x	1999 [6, с. 1]	–	Так	–	Так	PC-автомат
ARM Cortex A8	2005 [25]	AM335x [28, с. 4]	Так	–	Так	PC-автомат
Elbrus 2000	2007 [26]	Elbrus 2000	Так	–	Так	PC-автомат
ARM Cortex M0+	2012 [27, с. 1]	LPC81xM [29, с. 1]	Ні	Ні	Ні	S-автомат
Intel Skylake	2015 [18]	–	Так	–	Ні	P-автомат
AMD Zen 1	2017 [15]	–	Так	–	Ні	P-автомат
Intel Sunny Cove	2019 [8]	–	Так	–	Ні	P-автомат
AMD Zen 2	2019 [16]	–	Так	–	Ні	P-автомат
AMD Zen 4	2022 [17]	–	Так	–	Ні	P-автомат

¹Приклад мікроконтролера або одноядерного пристрою, в якому використовується відповідне ядро. Лише пристрій LPC81xM є мікроконтролером.

Згідно з табл. 2.9, в теперішній час найбільш розповсюдженим в процесорних ядрах є P-автомат, що є досягненням консенсусу між швидкістю та енергоефективністю [4, с. 555]. PC-автомат використовується лише у високопродуктивних паралельних обчисленнях, зокрема, в ядрах сигнальних

процесорів (наприклад, TMS320C6x). Хоча S-автомат зазвичай зустрічається лише в ранніх моделях процесорів (наприклад, Intel 8008) [4, с. 555], але він і наразі використовується в ультраенергоефективних процесорних ядрах (наприклад, ARM Cortex M0+).

Зазвичай, сучасні процесори є багатоядерними [4, с. 555]. У сфері ультраенергоефективних обчислень процесори витіснені мікроконтролерами (наприклад, LPC81xM з ядром ARM Cortex M0+). Задача паралельних обчислень може ефективно виконуватися як на багатоядерних (наприклад, TMS320C6x), так і на одноядерних процесорах (наприклад, AM335x з ядром ARM Cortex A8 та E2K).

В ядрах, які застосовуються в одноядерних пристроях, використовуються ОА як з послідовною (S-автомат), так і з паралельною (Р- або РС-автомат) структурами. В ядрах, які не застосовуються в одноядерних пристроях, використовуються ОА лише з паралельною структурою.

3 ТЕОРЕТИЧНЕ ПРОЄКТУВАННЯ ОПЕРАЦІЙНОГО АВТОМАТА З КОМБІНОВАНОЮ СТРУКТУРОЮ

Розробляється ОА типу ІМр, згідно з базовою класифікацією, що обумовлюється наявністю методології розробки відповідних типів ОА. Методологія розробки ІМр-автомата складається з етапів:

- формування пам'яті;
- розбиття множини операцій на двомісні та одномісні;
- розподіл регістрів по шинах;
- формування операторів;
- кодування функціональної граф-схеми алгоритму (ГСА);
- створення функціональної схеми операційного автомата.

3.1 Формування пам'яті

Для визначення необхідної кількості регістрів ОА побудовано функціональну ГСА (рис. 3.1) на основі формули 1.1 та обмежень і вимог до ОА. Обмеження накладене на ОА: паралельно може виконувати по одній операції різного типу. Виділяється два типи операцій: одномісні та двомісні. Вимоги до ОА: досягнення максимальної можливої кількості операцій виконуваних паралельно та мінімальної ресурсомісткості.

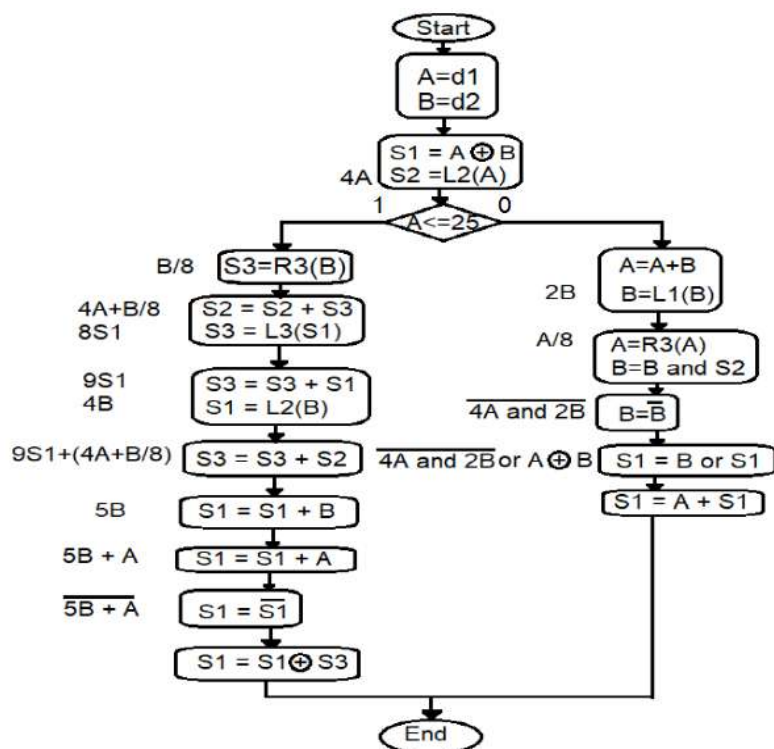


Рисунок 3.1 – Функціональна ГСА, яку реалізує спецобчислювач

Згідно з рис. 3.1, всього п'ять внутрішніх слів: A, B, S1, S2, S3. Внутрішнє слово – регістр-приймач, що використовуються для збереження проміжного результату. Тому як, всі змінні ГСА в апаратній реалізації представлені відповідними їм регістрами, то регістрам-приймачам відповідають змінні-приймачі. Отже, кількість регістрів ОА також дорівнює п'яти. Їх розрядність дорівнює восьми, згідно з постановкою задачі.

3.2 Розбиття множини операцій на двомісні та одномісні

На основі функціональної ГСА (див. рис. 3.1) отримано множину унікальних операцій: $Y = \{y1\#A=d1, y2\#B=d2, y3\#S1=A \oplus B, y4\#S2=4A, y5\#S3=R3(B), y6\#S2=S2+S3, y7\#S3=L3(S1), y8\#S3=S3+S1, y9\#S1=L2(B), y10\#S3=S3+S2, y11\#S1=S1+B, y12\#S1=A+S1, y13\#S1=!S1, y14\#S1=S1 \oplus S3, y15\#A=A+B, y16\#B=L1(B), y17\#A=R3(A), y18\#B=B \text{ and } S2, y19\#B=!B, y20\#S1=B \text{ or } S1\}$.

Одномісними називаються операції з одним операндом, двомісними – з двома. До множин одномісних $Y1$ та двомісних $Y2$ операцій відносяться відповідні n -місні операції, а також операції ініціалізації, що розподілені порівну між цими множинами. Розподіл операцій ініціалізації необхідний для можливості виконання двох операцій ініціалізації паралельно. Отже, маємо множини: $Y1 = \{y1, y4, y5, y7, y9, y13, y16, y17, y19\}$, $Y2 = \{y2, y3, y6, y8, y10, y11, y12, y14, y15, y18, y20\}$. Потужність множини $Y1$ дорівнює дев'яти, а потужність множини $Y2$ – 11.

3.3 Розподіл регістрів по шинах

Всього існує три шини $A1$ - $A3$, із яких $A1$ та $A2$ використовується для передачі першого та другого операндів, відповідно, для виконання двомісних операцій. Шина $A3$ призначена для передачі операнда для виконання одномісних операцій. До кожної із трьох шин $A1$ - $A3$ під'єднана відповідна множина регістрів $A1$ - $A3$, із яких завантажуються операнди. Потужність перетину цих множин повинна бути мінімальною для мінімізації кількості зв'язків регістрів з шинами.

Розподіл регістрів по шинах (множинам регістрів $A1$ - $A3$) відбувається на основі множини операцій Y . Таким чином, до множини $A3$ включені всі регістри, вміст яких використовується в якості операндів в одномісних операціях. Вміст множини $A3 = \{A, B, S1\}$.

До множин $A1$ та $A2$ повинні належати регістри, вміст яких використовується в якості операндів двомісних операцій. Створено табл. 3.1, в якій кожній двомісній операції відповідає її перший та другий операнди.

Таблиця 3.1 – Таблиця двомісних операцій

Двомісна операція	Перший операнд	Другий операнд
y3#S1=A $\oplus$ B	A	B
y6#S2=S2+S3	S2	S3
y8#S3=S3+S1	S3	S1
y10#S3=S3+S2	S3	S2
y11#S1=S1+B	S1	B
y12#S1=A+S1	A	S1
y14#S1=S1 $\oplus$ S3	S1	S3
y15#A=A+B	A	B
y18#B=B and S2	B	S2
y20#S1=B or S1	B	S1

Згідно з табл. 3.1, створено перший варіант вмісту множин $A1$ та $A2$: $A1 = \{A, S2, S3, S1, A, B\}$, $A2 = \{B, S3, S1, S2\}$. Якщо змінити місцями операнди двомісних операцій у6, у11, у14, то можна зменшити потужність перетину множин $A1$ та $A2$. Другий варіант вмісту множин $A1$ та $A2$: $A1 = \{A, S3, B\}$, $A2 = \{B, S2, S1\}$.

3.4 Формування операторів

Заповнення таблиць 3.2 та 3.3 необхідне для позначення зв'язків між регістрами, комбінаційними схемами (КСх), що виконують операції ОА, та шинами. Табл. 3.2 містить оператори для виконання одномісних операцій.

Таблиця 3.2 – Таблиця операторів для виконання одномісних операцій

yn	A3	Z1	RS1
y1	–	d1 ψ 1	RA e1
y4	RA c1	L2(A3) ψ 2	RS2 e2
y5	RB c2	R3(A3) ψ 3	RS3 e3
y7	RS1 c3	L3(A3) ψ 4	RS3 e3
y9	RB c2	L2(A3) ψ 2	RS1 e4
y13	RS1 c3	!A3 ψ 5	RS1 e4
y16	RB c2	L1(A3) ψ 6	RB e5
y17	RA c1	R3(A3) ψ 3	RA e1
y19	RB c2	!A3 ψ 5	RB e5

Згідно з табл. 3.2, колонка «yn» містить позначення одномісних операцій. Колонка «A3» містить регістри, що під'єднуються до шини A3 та зв'язок «с», що сполучає дані регістри з шиною A3. Кожному регістру відповідає певний зв'язок «с». Колонка «Z1» містить позначення операторів та зв'язок « ψ », яким під'єднується КСх, що реалізує даний оператор, до шини результату одномісних операцій Z1. Кожному оператору відповідає певний зв'язок « ψ ». Колонка «RS1» містить регістри-приймачі певної одномісної операції та зв'язок «е», яким під'єднується даний регістр до шини результату одномісних операцій Z1. Кожному регістру відповідає певний зв'язок «е».

Табл. 3.3 містить оператори для виконання двомісних операцій.

Таблиця 3.3 – Таблиця операторів для виконання двомісних операцій

yn	A1	A2	Z2	RS2
y2	–	–	d2 ϕ 1	RB d1
y3	RA a1	RB b1	$A1 \oplus A2$ ϕ 2	RS1 d2
y6	RS2 a2	RS3 b2	$A1 + A2$ ϕ 3	RS2 d3

Продовження таблиці 3.3

yn	A1	A2	Z2	RS2
y8	RS1 a3	RS3 b2	$A1 + A2 \varphi 3$	RS3 d4
y10	RS2 a2	RS3 b2	$A1 + A2 \varphi 3$	RS3 d4
y11	RS1 a3	RB b1	$A1 + A2 \varphi 3$	RS1 d2
y12	RA a3	RS1 b3	$A1 + A2 \varphi 3$	RS1 d2
y14	RS1 a3	RS3 b2	$A1 \oplus A2 \varphi 2$	RS1 d2
y15	RA a1	RB b1	$A1 + A2 \varphi 3$	RA d5
y18	RS2 a2	RB b1	$A1 \text{ and } A2 \varphi 4$	RB d1
y20	RS1 a3	RB b1	$A1 \text{ or } A2 \varphi 5$	RS1 d2

Згідно з табл. 3.3, колонка «yn» містить позначення двомісних операцій. Колонка «A1» містить регістри, що під'єднуються до шини A1 та зв'язок «a», що сполучає дані регістри з шиною A1. Колонка «A2» містить регістри, що під'єднуються до шини A2 та зв'язок «b», що сполучає дані регістри з шиною A2. Колонка «Z2» містить позначення операторів та зв'язок «φ», яким під'єднується КСх, що реалізує даний оператор, до шини результату двомісних операцій Z2. Колонка «RS2» містить регістри-приймачі певної двомісної операції та зв'язок «d», яким під'єднується даний регістр до шини результату двомісних операцій Z2.

Згідно з функціональною ГСА (див. рис. 3.1), наявна лише одна умова $A \leq 25$, для перевірки якої виконується операція $x1\#A \leq 25$: вміст регістру A завантажується до шини A3 по ланцюгу «c1»; стан шини A3 завантажується до шини Z1 по ланцюгу «ψ7». Тому, виконано заміну та отримано оператор умови $x1\#Z1 \leq 25$.

3.6 Декомпозиція операцій граф-схеми алгоритму

На рис. 3.2 наведено атомарний граф, отриманий в результаті декомпозиції операцій ГСА (див. рис. 3.1). Згідно з рис. 3.2, цифро-символьне позначення «bn» означає певну вершину атомарного графа. У вершинах графа замість операцій ОА вказано зв'язки, обрання яких відповідає виконанню відповідної операції ОА. Наприклад, позначення « $\psi_1 e_1$ » у вершині означає обрання зв'язків « ψ_1 » та « e_1 ». Обрання ланцюга « ψ_1 » означає завантаження вхідного значення d1 до шини Z1. Обрання ланцюга « e_1 » означає завантаження стану шини Z1 до регістру A. Таким чином, обрання обидвох зв'язків означає виконання операції ініціалізації $y1\#A=d1$.

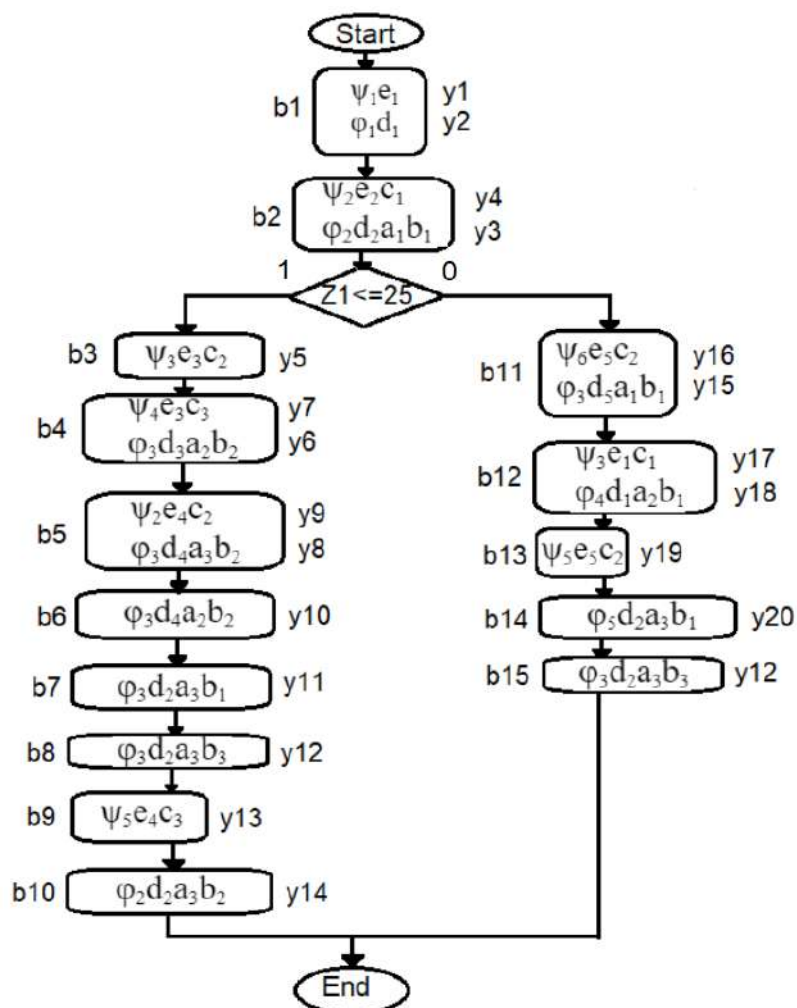


Рисунок 3.2 – Атомарний граф на основі функціональної ГСА

3.7 Створення функціональної схеми операційного автомата

Результат синтезу функціональної схеми наведено на рис. 3.3. Порядок побудови функціональної схеми наступний. Розміщено регістри ОА та КСх, що реалізують операції ОА. Між виходами регістрів та входами КСх прокладено шини A1-A3. Між виходами КСх та входами регістрів прокладено шини Z1 та Z2. Під'єднано регістри та КСх до шин за допомогою зв'язків згідно з табл. 3.2 та 3.3.

Виходи регістрів до шин A1-A3 під'єднуються за допомогою зв'язків «а», «b», «с», відповідно. Входи КСх, що реалізують одномісні операції, під'єднуються до шини A3. Входи КСх, що реалізують двомісні операції, під'єднуються до шин A2 та A3. Вхід КСх, що виконує перевірку умови, під'єднується до шини Z1.

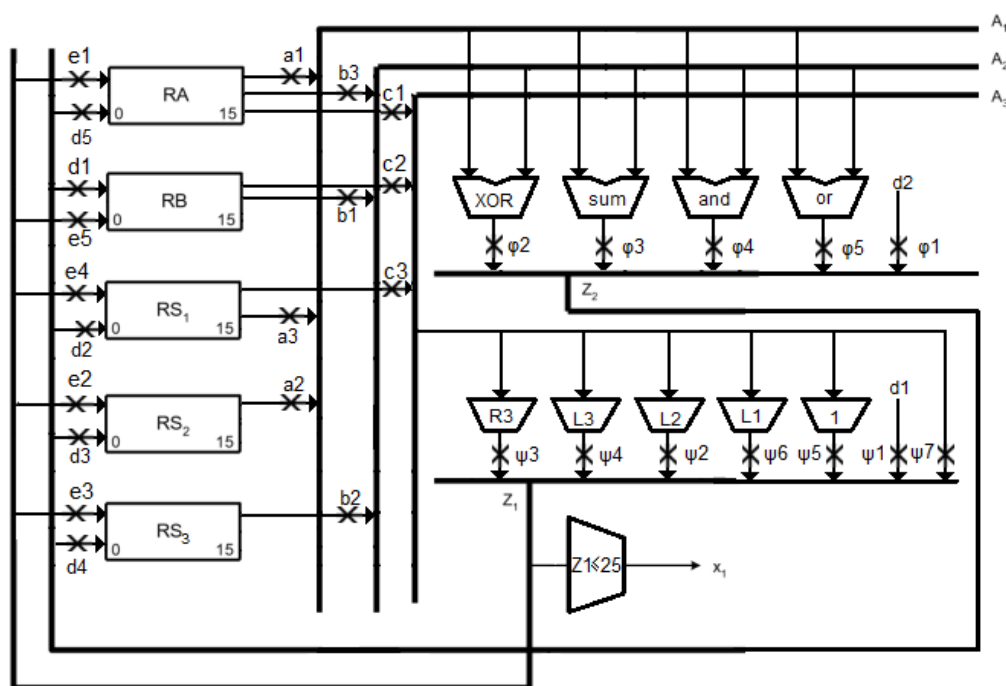


Рисунок 3.3 – функціональна схема ОА з комбінованою структурою

Виходи КСх, що реалізують одномісні операції, під'єднуються до шини Z1 за допомогою зв'язка «ψ». Виходи КСх, що реалізують двомісні операції, під'єднуються до шини Z2 за допомогою зв'язка «φ». На виході КСх, що реалізує операцію перевірки умови, формується зовнішній вихідний сигнал ОА «х».

Входи регістрів під'єднуються до шин Z1 та Z2 за допомогою зв'язків «е» та «d», відповідно. До шин Z1 та Z2 під'єднано зв'язки «ф», на яких формуються зовнішні входні сигнали ОА «d1» та «d2», відповідно. Також, до шини Z1 під'єднано зв'язок «ψ», призначений для передачі операнда до КСх, що реалізує операцію перевірки умови.

4 РОЗРОБКА КЕРУЮЧОГО АВТОМАТА МУРА

4.1 Теоретичне проєктування

Порядок проєктування [1, с. 226]:

- кодування ГСА;
- позначення ГСА;
- створення графа алгоритму;
- створення таблиці функцій виходів;
- створення таблиці функцій переходів та збудження тригерів;
- складання рівнянь виходів та збудження тригерів;
- мінімізація рівнянь;
- створення переліку унікальних мінтермів;
- виконання заміни в рівняннях виходів та збудження тригерів.

Спочатку виконується кодування функціональної ГСА (див. рис. 3.1). Закодована ГСА наведена на рис. 4.1. В процесі кодування виконується заміна операцій ОА на їх позначення.

Далі, виконується позначення ГСА (рис. 4.1). На цьому етапі вершини ГСА відмічаються позначеннями станів керуючого автомата «Sn». В залежності того, в якому стані перебуває керуючий автомат, генерується певне значення на виході Y. Кожній одиниці у вихідному значенні відповідає операція ОА, яка повинна бути виконана операційним автоматом у відповідь на керуючі сигнали КА.

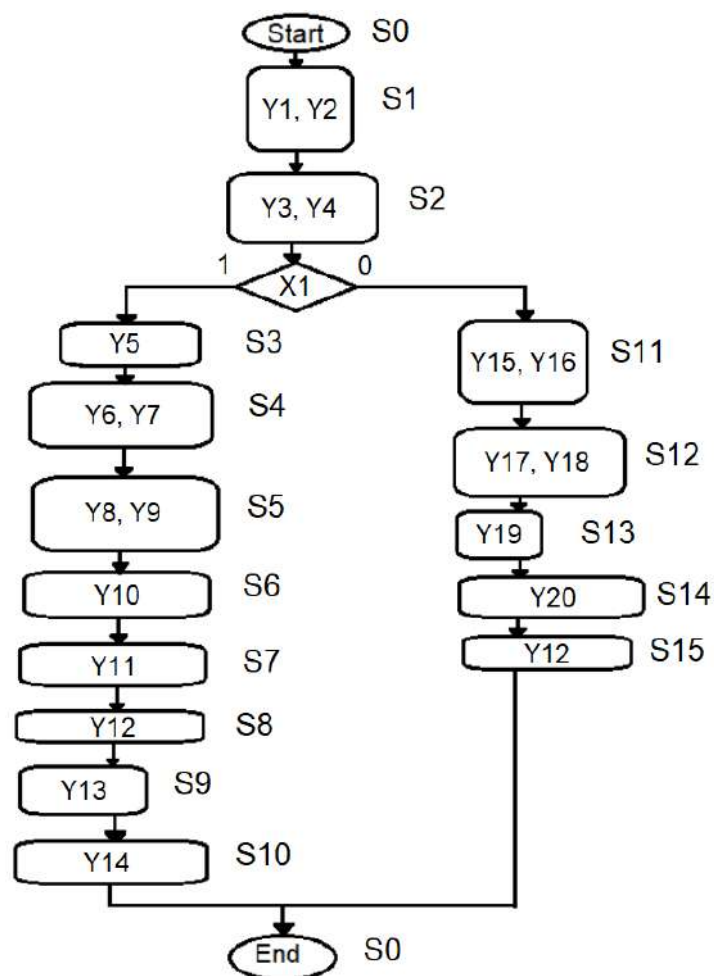


Рисунок 4.1 – Закодована та відмічена ГСА

На основі відміченої ГСА створено граф алгоритму (рис. 4.5).

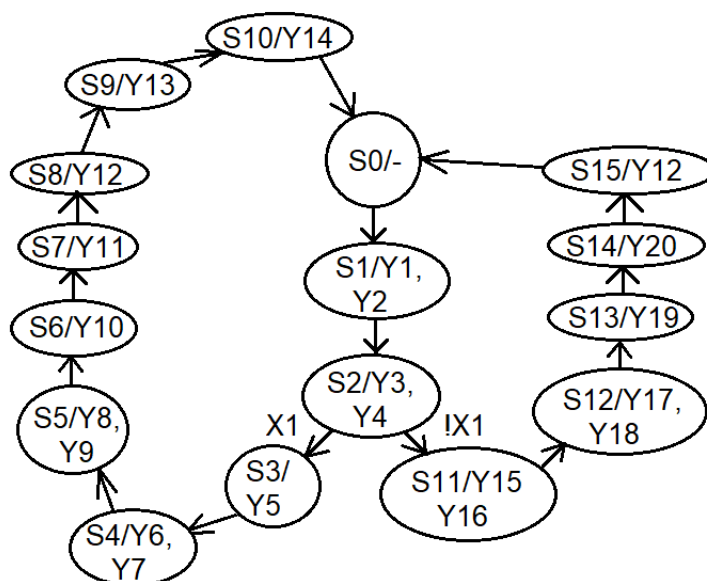


Рисунок 4.5 – Граф алгоритму

Тому, як передбачається використання D-тригерів в якості пам'яті КА, то для подальшого складання рівнянь збудження тригерів необхідно взяти таблицю збудження D-тригера.

Таблиця 4.1 – збудження D-тригера [30, с. 2]

$q(t) \rightarrow q(t+1)$	d
0 $\rightarrow$ 0	0
0 $\rightarrow$ 1	1
1 $\rightarrow$ 0	0
1 $\rightarrow$ 1	1

На основі графа алгоритму створено таблицю функцій виходів, що поділена на табл. 4.2 та 4.3. Дані таблиці включають коди станів КА та керуючі сигнали Y, що генеруються у відповідь на перехід КА у відповідний стан. Для кодування станів КА обрано найпростіше двійкове кодування.

Таблиця 4.2 – Керуючі сигнали (Y1-10) та відповідні їм коди стану Q

Стан	Код стану				Керуючі сигнали									
	Q3	Q2	Q1	Q0	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9	Y10
S0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S1	0	0	0	1	1	1	0	0	0	0	0	0	0	0
S2	0	0	1	0	0	0	1	1	0	0	0	0	0	0
S3	0	0	1	1	0	0	0	0	1	0	0	0	0	0
S4	0	1	0	0	0	0	0	0	0	1	1	0	0	0
S5	0	1	0	1	0	0	0	0	0	0	0	1	1	0
S6	0	1	1	0	0	0	0	0	0	0	0	0	0	1

Таблиця 4.3 – Керуючі сигнали (Y11-21) та відповідні їм коди стану Q

Стан	Код стану				Керуючі сигнали									
	Q3	Q2	Q1	Q0	Y11	Y12	Y13	Y14	Y15	Y16	Y17	Y18	Y19	Y20
S7	0	1	1	1	1	0	0	0	0	0	0	0	0	0
S8	1	0	0	0	0	1	0	0	0	0	0	0	0	0
S9	1	0	0	1	0	0	1	0	0	0	0	0	0	0
S10	1	0	1	0	0	0	0	1	0	0	0	0	0	0
S11	1	0	1	1	0	0	0	0	1	1	0	0	0	0
S12	1	1	0	0	0	0	0	0	0	0	1	1	0	0
S13	1	1	0	1	0	0	0	0	0	0	0	0	1	0
S14	1	1	1	0	0	0	0	0	0	0	0	0	0	1
S15	1	1	1	1	0	1	0	0	0	0	0	0	0	0

На основі графа алгоритму (див. рис. 4.5) та таблиці збудження D-тригера (див. табл. 4.1) створено таблицю функцій переходів та збудження тригерів (табл. 4.4).

Таблиця 4.4 – Таблиця функцій переходів та збудження тригерів

Початковий стан					Наступний стан					Умова переходу	Функції збудження			
Стан	Код стану				Стан	Код стану					D3	D2	D1	D0
	Q3	Q2	Q1	Q0		Q3	Q2	Q1	Q0					
S0	0	0	0	0	S1	0	0	0	1	1	0	0	0	1
S1	0	0	0	1	S2	0	0	1	0	1	0	0	1	0
S2	0	0	1	0	S3	0	0	1	1	X1	0	0	1	1
S3	0	0	1	1	S4	0	1	0	0	1	0	1	0	0
S4	0	1	0	0	S5	0	1	0	1	1	0	1	0	1
S5	0	1	0	1	S6	0	1	1	0	1	0	1	1	0
S6	0	1	1	0	S7	0	1	1	1	1	0	1	1	1
S7	0	1	1	1	S8	1	0	0	0	1	1	0	0	0
S8	1	0	0	0	S9	1	0	0	1	1	1	0	0	1
S9	1	0	0	1	S10	1	0	1	0	1	1	0	1	0
S10	1	0	1	0	S0	0	0	0	0	1	0	0	0	0
S2	0	0	1	0	S11	1	0	1	1	!X1	1	0	1	1
S11	1	0	1	1	S12	1	1	0	0	1	1	1	0	0
S12	1	1	0	0	S13	1	1	0	1	1	1	1	0	1
S13	1	1	0	1	S14	1	1	1	0	1	1	1	1	0
S14	1	1	1	0	S15	1	1	1	1	1	1	1	1	1
S15	1	1	1	1	S0	0	0	0	0	1	0	0	0	0

На основі таблиці 4.4 складено рівняння виходів (Y1-Y20) та збудження тригерів (D0-D3):

- « $D0 = !Q3!Q2!Q1!Q0 \vee !Q3!Q2Q1!Q0X1 \vee !Q3Q2!Q1!Q0 \vee !Q3Q2Q1!Q0 \vee Q3!Q2!Q1!Q0 \vee !Q3!Q2Q1!Q0!X1 \vee Q3Q2!Q1!Q0 \vee Q3Q2Q1!Q0$ »;

- « $D1 = !Q3!Q2!Q1Q0 \vee !Q3!Q2Q1!Q0X1 \vee !Q3Q2!Q1Q0 \vee !Q3Q2Q1!Q0 \vee Q3!Q2!Q1Q0 \vee !Q3!Q2Q1!Q0!X1 \vee Q3Q2!Q1Q0 \vee Q3Q2Q1!Q0$ »;

- « $D2 = !Q3!Q2Q1Q0 \vee !Q3Q2!Q1!Q0 \vee !Q3Q2!Q1Q0 \vee !Q3Q2Q1!Q0 \vee Q3!Q2Q1Q0 \vee Q3Q2!Q1!Q0 \vee Q3Q2!Q1Q0 \vee Q3Q2Q1!Q0$ »;

- « $D3 = !Q3Q2Q1Q0 \vee Q3!Q2!Q1!Q0 \vee Q3!Q2!Q1Q0 \vee !Q3!Q2Q1!Q0!X1 \vee Q3!Q2Q1Q0 \vee Q3Q2!Q1!Q0 \vee Q3Q2!Q1Q0 \vee Q3Q2Q1!Q0$ »;

- « $Y1 = Y2 = !Q3!Q2!Q1Q0$ »;
- « $Y3 = Y4 = !Q3!Q2Q1!Q0$ »;
- « $Y5 = !Q3!Q2Q1Q0$ »;
- « $Y6 = Y7 = !Q3Q2!Q1!Q0$ »;
- « $Y8 = Y9 = !Q3Q2!Q1Q0$ »;
- « $Y10 = !Q3Q2Q1!Q0$ »;
- « $Y11 = !Q3Q2Q1Q0$ »;
- « $Y12 = Q3!Q2!Q1!Q0 \vee Q3Q2Q1Q0$ »;
- « $Y13 = Q3!Q2!Q1Q0$ »;
- « $Y14 = Q3!Q2Q1!Q0$ »;
- « $Y15 = Y16 = Q3!Q2Q1Q0$ »;
- « $Y17 = Y18 = Q3Q2!Q1!Q0$ »;
- « $Y19 = Q3Q2!Q1Q0$ »;
- « $Y20 = Q3Q2Q1!Q0$ ».

Мінімізовано складені рівняння за допомогою веб-застосунку:

- « $D0 = !Q0!Q1 \vee !Q0!Q3 \vee !Q0Q2$ »;
- « $D1 = Q0!Q1 \vee !Q0Q1!Q3 \vee !Q0Q1Q2$ »;
- « $D2 = !Q0Q2 \vee !Q1Q2 \vee Q0Q1!Q2$ »;
- « $D3 = Q0!Q2Q3 \vee !Q0Q2Q3 \vee !Q1Q3 \vee !Q3Q0Q1Q2 \vee !Q0Q1!Q2!Q3!X1$ ».

Для уникнення апаратного дублювання при реалізації мінтермів, що повторюються серед складених рівнянь, складено перелік унікальних мінтермів:

- « $M1 = !Q3!Q2!Q1Q0$ »;
- « $M2 = !Q3!Q2Q1!Q0$ »;
- « $M3 = !Q3!Q2Q1Q0$ »;
- « $M4 = !Q3Q2!Q1!Q0$ »;
- « $M5 = !Q3Q2!Q1Q0$ »;
- « $M6 = !Q3Q2Q1!Q0$ »;
- « $M7 = !Q3Q2Q1Q0$ »;
- « $M8 = Q3!Q2!Q1!Q0$ »;
- « $M9 = Q3Q2Q1Q0$ »;

- « $M_{10} = Q_3!Q_2!Q_1Q_0$ »;
- « $M_{11} = Q_3!Q_2Q_1!Q_0$ »;
- « $M_{12} = Q_3!Q_2Q_1Q_0$ »;
- « $M_{13} = Q_3Q_2!Q_1!Q_0$ »;
- « $M_{14} = Q_3Q_2!Q_1Q_0$ »;
- « $M_{15} = Q_3Q_2Q_1!Q_0$ »;
- « $M_{16} = !Q_0!Q_1$ »;
- « $M_{17} = !Q_0!Q_3$ »;
- « $M_{18} = !Q_0Q_2$ »;
- « $M_{19} = Q_0!Q_1$ »;
- « $M_{20} = !Q_0Q_1!Q_3$ »;
- « $M_{21} = !Q_0Q_1Q_2$ »;
- « $M_{22} = !Q_1Q_2$ »;
- « $M_{23} = Q_0Q_1!Q_2$ »;
- « $M_{24} = Q_0!Q_2Q_3$ »;
- « $M_{25} = !Q_0Q_2Q_3$ »;
- « $M_{26} = !Q_0Q_1!Q_2!Q_3!X_1$ »;
- « $M_{27} = !Q_1Q_3$ ».

В складених рівняннях виконано заміну мінтермів на позначення унікальних мінтермів:

- « $Y_1 = Y_2 = M_1$ »;
- « $Y_3 = Y_4 = M_2$ »;
- « $Y_5 = M_3$ »;
- « $Y_6 = Y_7 = M_4$ »;
- « $Y_8 = Y_9 = M_5$ »;
- « $Y_{10} = M_6$ »;
- « $Y_{11} = M_7$ »;
- « $Y_{12} = M_8 \vee M_9$ »;
- « $Y_{13} = M_{10}$ »;
- « $Y_{14} = M_{11}$ »;

- «Y15 = Y16 = M12»;
- «Y17 = Y18 = M13»;
- «Y19 = M14»;
- «Y20 = M15»;
- «D0 = M16 v M17 v M18»;
- «D1 = M19 v M20 v M21»;
- «D2 = M18 v M22 v M23»;
- «D3 = M24 v M25 v M27 v M7 v M26».

Складені рівняння необхідні при створенні опису КА на мові Verilog.

4.2 Проєктування

На основі складених рівнянь створено опис керуючого автомата Мура на мові Verilog (лістинг 4.1). Основна логіка роботи КА описується за допомогою конструкцій послідовного присвоювання «assign» [31, с. 98]. Рівняння виступають в якості функцій в чутливій частині (права частина від символу присвоювання «=») конструкції «assign». В чутливій частині змінна рівня будь-якого сигналу-складової рівняння призводить до повторного обчислення функції та запису її результату до ланцюга в лівій частині від символу присвоювання блоку «assign». Автомат має додатковий вихід state призначений для відлагодження внутрішнього стану КА та він не використовується в апаратній реалізації.

Лістинг 4.1 – Опис керуючий автомата Мура мовою Verilog.

```
module automatMura(X, reset, clk, Y, state);
//зовнішні інтерфейси КА
input X, reset, clk;
output [19:0] Y;
output reg [3:0] state;
```

```

//внутрішні з'єднання
wire [3:0] D;
reg [3:0] Q;
//запис внутрішнього стану на вихід КА для його відлагодження
always @(Q) state = Q;
//формування вхідних сигналів тригерів на основі функцій
//збудження тригерів
assign D[0] = ~Q[0]&~Q[1] | ~Q[0]&~Q[3] |
~Q[0]&Q[2];
assign D[1] = Q[0]&~Q[1] | ~Q[0]&Q[1]&~Q[3] | ~Q[0]&Q[1]&Q[2];
assign D[2] = ~Q[0]&Q[2] | ~Q[1]&Q[2] | Q[0]&Q[1]&~Q[2];
assign D[3] = Q[0]&~Q[2]&Q[3] | ~Q[0]&Q[2]&Q[3] | !Q[1]&Q[3] |
~Q[3]&Q[0]&Q[1]&Q[2] | ~Q[0]&Q[1]&~Q[2]&~Q[3]&~X;
//формування керуючих сигналів на основі функ. виходів
assign Y[0] = ~Q[3]&~Q[2]&~Q[1]&Q[0];
assign Y[1] = ~Q[3]&~Q[2]&~Q[1]&Q[0];
assign Y[2] = ~Q[3]&~Q[2]&Q[1]&~Q[0];
assign Y[3] = ~Q[3]&~Q[2]&Q[1]&~Q[0];
assign Y[4] = ~Q[3]&~Q[2]&Q[1]&Q[0];
assign Y[5] = ~Q[3]&Q[2]&~Q[1]&~Q[0];
assign Y[6] = ~Q[3]&Q[2]&~Q[1]&~Q[0];
assign Y[7] = ~Q[3]&Q[2]&~Q[1]&Q[0];
assign Y[8] = ~Q[3]&Q[2]&~Q[1]&Q[0];
assign Y[9] = ~Q[3]&Q[2]&Q[1]&~Q[0];
assign Y[10] = ~Q[3]&Q[2]&Q[1]&Q[0];
assign Y[11] = Q[3]&~Q[2]&~Q[1]&~Q[0] | Q[3]&Q[2]&Q[1]&Q[0];
assign Y[12] = Q[3]&~Q[2]&~Q[1]&Q[0];
assign Y[13] = Q[3]&~Q[2]&Q[1]&~Q[0];
assign Y[14] = Q[3]&~Q[2]&Q[1]&Q[0];
assign Y[15] = Q[3]&~Q[2]&Q[1]&Q[0];
assign Y[16] = Q[3]&Q[2]&~Q[1]&~Q[0];
assign Y[17] = Q[3]&Q[2]&~Q[1]&~Q[0];
assign Y[18] = Q[3]&Q[2]&~Q[1]&Q[0];
assign Y[19] = Q[3]&Q[2]&Q[1]&~Q[0];

```

```
//формування внутрішнього стану КА на основі вхідних сигналів
//тригерів
always@(posedge clk, reset)
if(!reset) Q = D;
else if(reset) Q = 0;
endmodule
```

Для тестування опису керуючого автомата створено тестовий модуль на мові Verilog (лістинг 4.2). Тестовий модуль призначений лише для тестування іншого HDL модуля та не бере участь при синтезі списку з'єднань.

Лістинг 4.2 – Модуль на мові Verilog для тестування опису керуючого автомата.

```
module tb_automatMura;
reg X, reset, clk;
wire [19:0] Y;
wire [3:0] state;
//реалізація модуля керуючого автомата
automatMura automatMural(X,reset,clk,Y,state);
//генерація вхідних сигналів
initial begin
clk = 0; X = 0; reset = 1;
#15 reset = 0;
#200 X = 1;
#200 X = 0;
#200 reset = 1;
#40 $finish;
end
always #10 clk = ~clk;
endmodule
```

Згідно з лістингом 4.2, тестовий модуль генерує вхідні значення для КА таким чином, щоб перевірити роботу КА в кожному із його станів.

На рис. 4.7 наведено результати тестування КА для правої гілки алгоритму (див. рис. 3.1).



Рисунок 4.7 – Результати тестування керуючого автомата Мура для правої гілки алгоритму

Згідно з рис. 4.7, поки $x = 0$ генеруються керуючі сигнали для виконання операцій в правій гілці алгоритму. В стані один генеруються активні сигнали на виходах Y1 та Y2, згідно з алгоритмом (див. рис. 3.1). В стані два – на виходах Y3, Y4. В стані три – Y15, Y16 і т.д.

5 РОЗРОБКА СПЕЦОБЧИСЛЮВАЧА

5.1 Прокєтування

На лістингу 5.1 наведено опис спецобчислювача на мові Verilog, який призначений для детального його тестування. Тому даний опис має додаткові виходи, на яких формуються значення вмісту внутрішніх регістрів ОА, а саме А, В, S2, S3. Ці виходи не використовуються в апаратній реалізації.

Модуль спецобчислювача складається з двох модулів: верхнього модуля `specialized_digital_device` та екземпляра модуля `automatMura`. Верхній модуль містить зовнішні інтерфейси спецобчислювача, опис ОА та реалізує модуль `automatMura`.

Лістинг 5.1 – Verilog-опис спецобчислювача, призначений для його детального тестування

```
`timescale 1ns / 1ps
module specialized_digital_device(d1, d2, reset, clk, x, oA,
oB, oS1, oS2, oS3, state);
//зовнішні інтерфейси
input [0:7] d1, d2;
input reset, clk;
output x;
output [0:7] oA, oB, oS1, oS2, oS3;
output [3:0] state;
//внутрішні зв'язки
reg [0:7] A, B, S1, S2, S3;
//запис вмісту внутрішніх регістрів на вихід КА для
відлагодження
assign x = A<=25;
assign oA = A;
assign oB = B;
assign oS1 = S1;
```

```

assign oS2 = S2;
assign oS3 = S3;
//обчислення умови алгоритму та запис результату на вихід КА
assign x = A<=25;
wire [19:0] Y;
wire [3:0] state;
//реалізація модуля КА
automatMura automatMural(x,reset,clk,Y,state);
//опис операційного автомата
always @ (Y, reset)
if (!reset) begin
//в залежності від активних керуючих сигналів на вході Y
виконуються певні операції OA
case (Y)
3: begin
    A <= d1; B <= d2;
    end
12: begin
    S2 <= A >> 2; S1 <= A^B;
    end
16: S3 = B << 3;
96: begin
    S3 <= S1 >> 3; S2 <= S2 + S3;
    end
384: begin
    S1 <= B >> 2; S3 <= S3 + S1;
    end
512: S3 = S3 + S2;
1024: S1 = S1 + B;
2048: S1 = A + S1;
4096: S1 = ~S1;
8192: S1 = S1^S3;
49152: begin
    B <= B>>1; A <= A+B;
    end

```

```

196608: begin
    A <= A << 3; B <= B & S2;
end
262144: B=~B;
default: S1 = B | S1;
endcase
end
else if (reset)
begin
    A <=0; B <= 0; S1 <= 0; S2 <= 0;
    S3 <= 0;
end
endmodule

```

На лістингу 5.2 наведено модуль на мові Verilog для тестування опису спецобчислювача.

Лістинг 5.2 – Модуль на мові Verilog для тестування опису спецобчислювача

```

`timescale 1ns/1ps
module tb_spec_dig_dev;
reg [0:7] d1, d2;
reg reset, clk;
wire x;
wire [0:7] oA, oB, oS1, oS2, oS3;
wire [3:0] state;
//реалізація модуля спецобчислювача
specialized_digital_device spec_dig_dev(d1, d2, reset, clk, x,
oA, oB, oS1, oS2, oS3, state);
//генерація вхідних сигналів
initial begin
clk = 0;
//вхідні сигнали даних
d1 = 6; d2 = 8; //для тестування лівої гілки алгоритму
//d1 = 26; d2 = 22; для тестування правої гілки алгоритму

```

```

reset = 1;
#15 reset = 0;
#300 reset = 1;
#40 $finish;
end
always #10 clk = ~clk;
endmodule

```

Згідно з лістингом 5.2, тестовий модуль містить два набори вхідних значень, призначених для тестування певної гілки алгоритму. Для використання певного набору необхідно розкоментувати відповідний рядок тестового модуля та повторити поведінкову симуляцію.

Спроектований спецобчислювач не враховує переповнень. Тому, він коректно обчислює задану формулу лише для обмеженого діапазону комбінацій вхідних значень. Тому, для його тестування підібрано вхідні значення таким чином, щоб уникнути переповнення та переконатися, що спецобчислювач здатен виконувати задачу на основі деяких вхідних наборів. Використовується два вхідні набори d1 та d2 для тестування обох гілок алгоритму.

5.2 Тестування

5.2.1 Перевірка гілки алгоритму, де умова виконується

Перевіряється ліва гілка алгоритму. На рис. 5.1 наведено результат поведінкової симуляції спецобчислювача в середовищі Vivado, що відображає результат виконання операцій ОА на основі керуючих сигналів КА, які генеруються в стані S1.

Згідно з рис. 5.1, в стані S1 отримано результати операцій ініціалізації $A = d1$ та $B = d2$, де d1 та d2 – вхідні дані спецобчислювача. Значення операндів d1, d2

визначені в попередньому стані $S0$ та дорівнюють 6_{16} , 8_{16} , відповідно. Перевірено правильність виконання операцій: $A = d1 = 6_{16}$, $B = d2 = 8_{16}$.

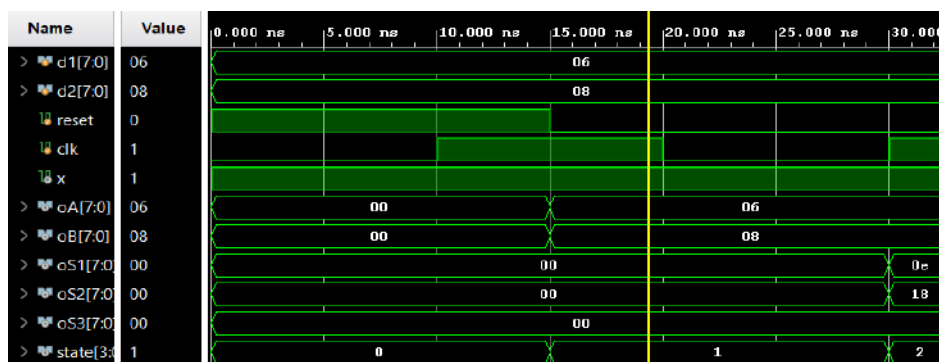
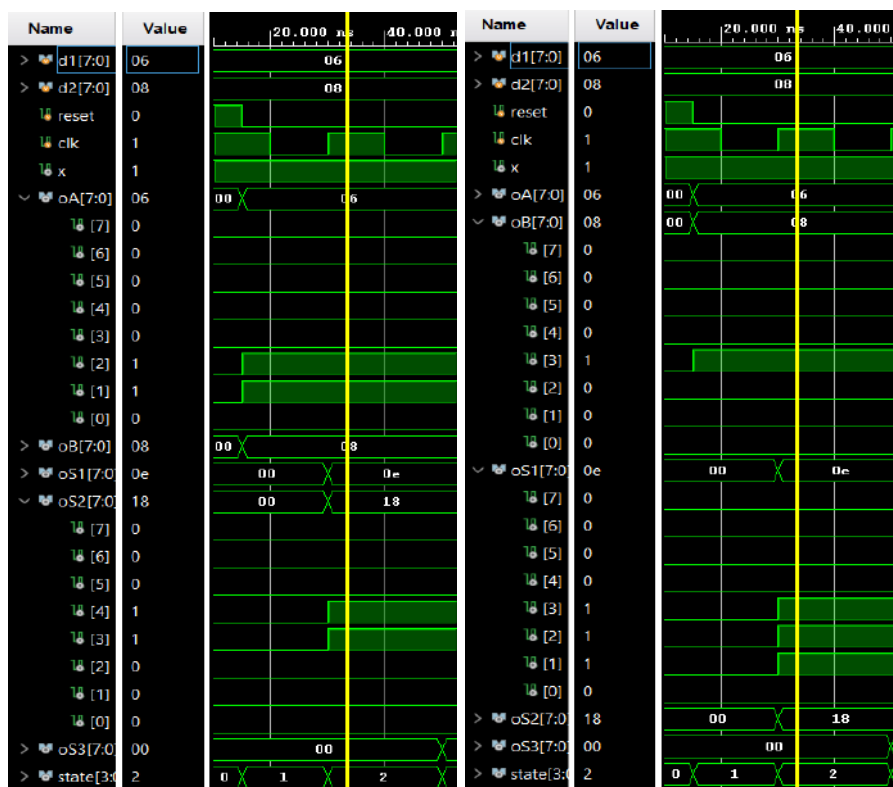


Рисунок 5.1 – Стан $S1$: виконання операцій ініціалізації $A = d1$ та $B = d2$

На рис. 5.2 наведено результати виконання операцій у стані $S2$ керуючого автомата.



а)

б)

Рисунок 5.2 – Стан $S2$: виконання одномісної $S2 = L2(A)$ та двомісної $S1 = A \oplus B$ операцій, де а) – вміст регістрів A , $S2$; б) – вміст регістрів B , $S1$

Згідно з рис. 5.2, в стані S2 отримано результати одномісної операції $S2 = L2(A)$, яка еквівалентна операції $4_{10} * A$, та двомісної $S1 = A \oplus B$. Значення операндів A, B визначені в попередньому стані S1 та дорівнюють 6_{10} (00000110_2), 8_{16} (00001000_2), відповідно.

Перевірено правильність виконання одномісної операції: $S2 = 4_{10} * A = 4_{10} * 6_{10} = 24_{10}$ (18_{16}). Перевірено правильність виконання двомісної операції: $S1 = 00000110_2 \oplus 00001000_2 = 00001110_2$ (E_{16}).

Тому, як вміст регістру A дорівнює 6_{10} , то виконується умова $A \leq 25_{10}$. Отже, наступним станом повинен бути S3 (виконується ліва гілка алгоритму).

На рис. 5.3 наведено результати виконання операцій у стані S3 керуючого автомата.

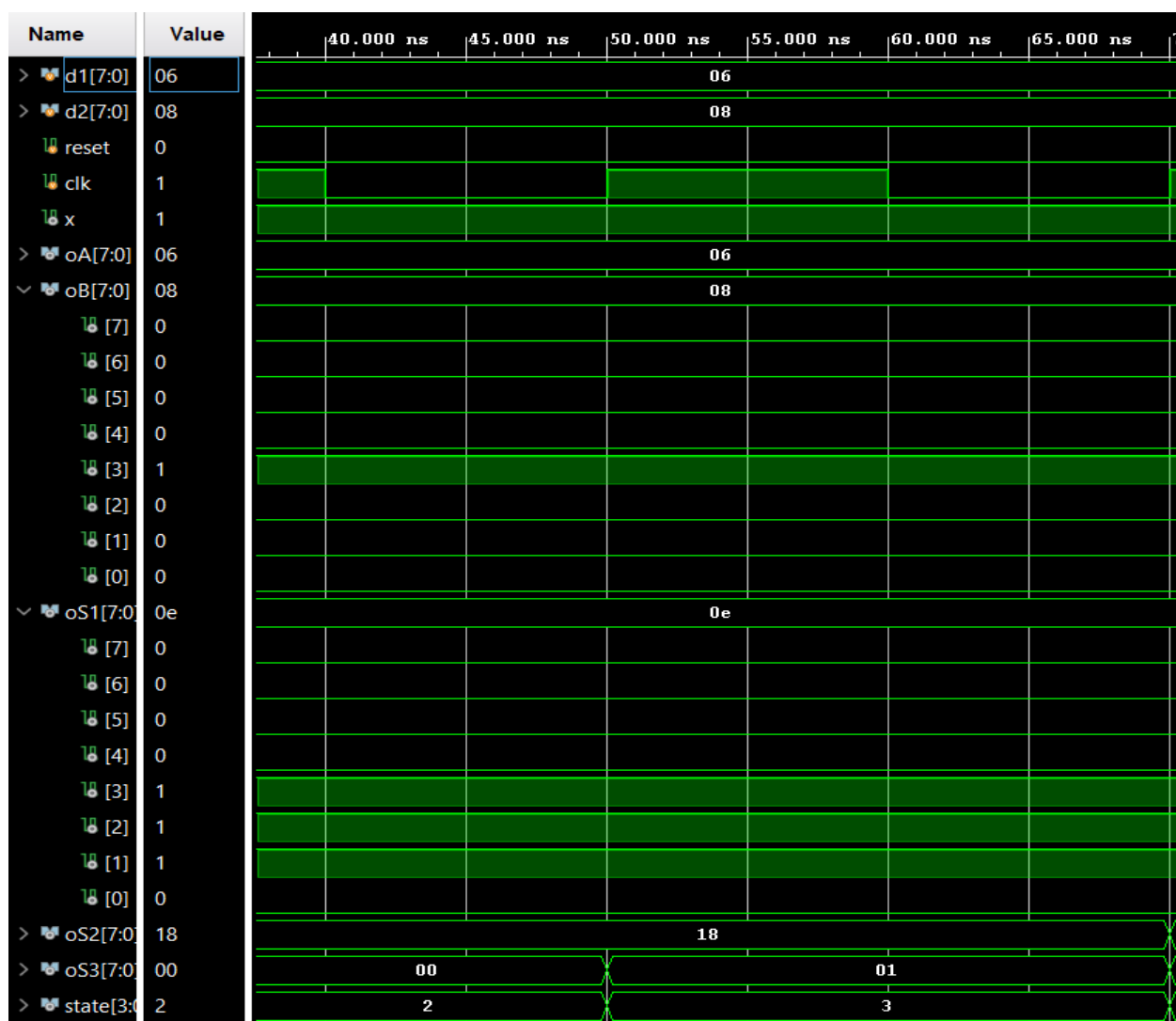


Рисунок 5.3 – Стан S3: виконання одномісної операції $S3 = R3(B)$

Згідно з рис. 5.3, в стані S3 отримано результат одномісної операції $S3 = R3(B)$, яка еквівалентна операції $B / 8_{10}$. Значення операнда B визначене в попередньому стані S2 та дорівнює 8_{10} . Перевірено правильність виконання одномісної операції: $S3 = B / 8_{10} = 8_{10} / 8_{10} = 1_{16}$.

На рис. 5.4 наведено результати виконання операцій у стані S4 керуючого автомата.

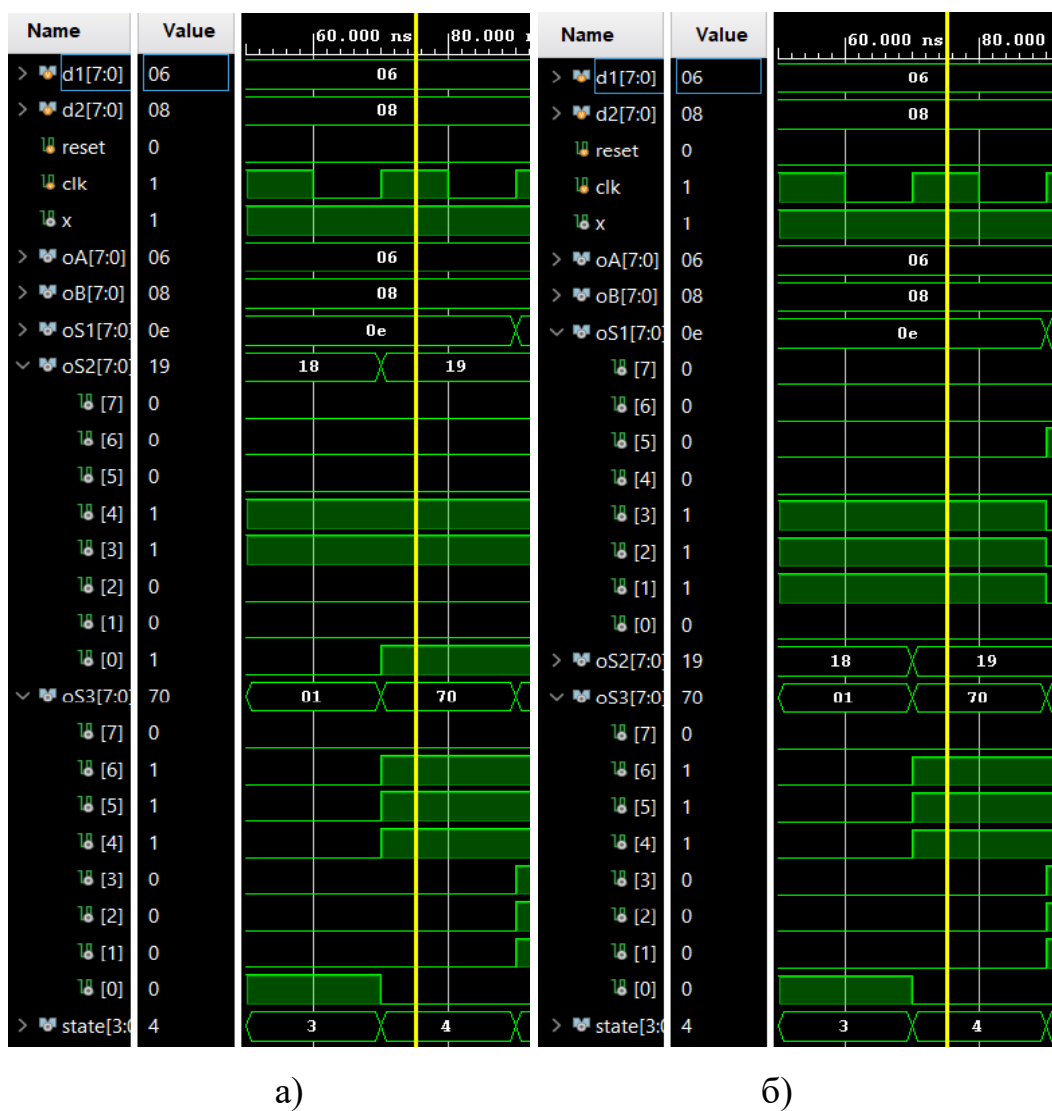


Рисунок 5.4 – Стан S4: виконання одномісної $S3 = L3(S1)$ та двомісної $S2 = S2 + S3$ операцій, де а) – вміст регістрів S2, S3; б) – вміст регістрів S1, S3

Згідно з рис. 5.4, в стані S4 отримано результати одномісної операції $S3 = L3(S1)$, яка еквівалентна операції $8_{10} * S1$, та двомісної операції $S2 = S2 + S3$.

Значення операндів S1-S3 визначено в попередньому стані S3 та дорівнюють E_{16} (14_{10} , 00001110_2), 18_{16} (24_{10}), 1_{10} , відповідно.

Перевірено правильність виконання одномісної операції: $S3 = 8_{10} * S1 = 8_{10} * 14_{10} = 112_{10}$ (70_{16}). Перевірено правильність виконання двомісної операції: $S2 = 24_{10} + 1_{10} = 25_{10}$ (19_{16}).

На рис. 5.5 наведено результати виконання операцій у стані S5 керуючого автомата.

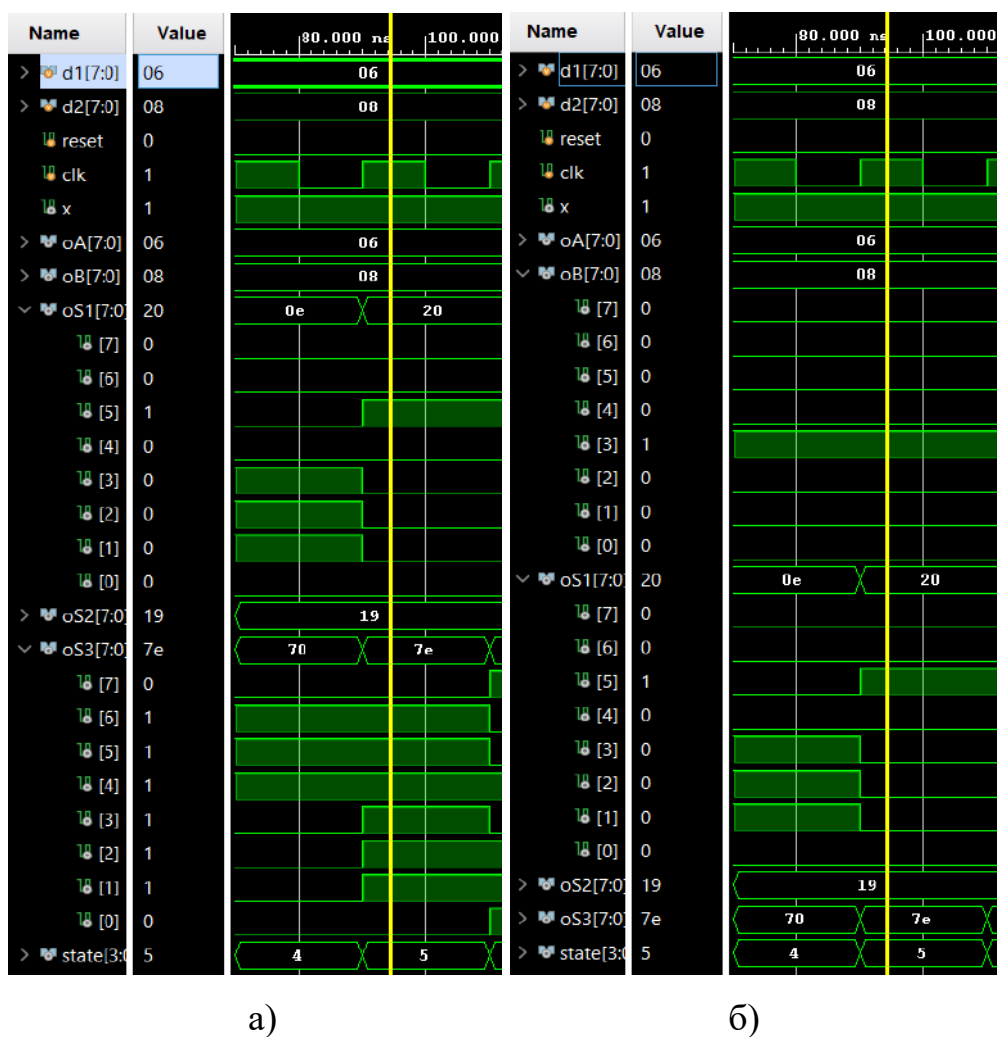


Рисунок 5.5 – Стан S5: виконання одномісної $S1 = L2(B)$ та двомісної $S3 = S3 + S1$ операцій, де а) – вміст регістрів S1, S3; б) – вміст регістрів B, S1

Згідно з рис. 5.6, в стані S5 отримано результати одномісної операції $S1 = L2(B)$, яка еквівалентна операції $4_{10} * B$, та двомісної операції $S3 = S3 + S1$.

Значення операндів S1, S3, B визначено в попередньому стані S4 та дорівнюють E_{16} (14_{10}), 70_{16} (112_{10}), 8_{10} , відповідно.

Перевірено правильність виконання одномісної операції: $S1 = L2(B) = 4_{10} * 8_{10} = 32_{10}$ (20_{16}). Перевірено правильність виконання двомісної операції: $S3 = 112_{10} + 14_{10} = 126_{10}$ ($7E_{16}$).

На рис. 5.6 наведено результати виконання операцій у стані S6 керуючого автомата.

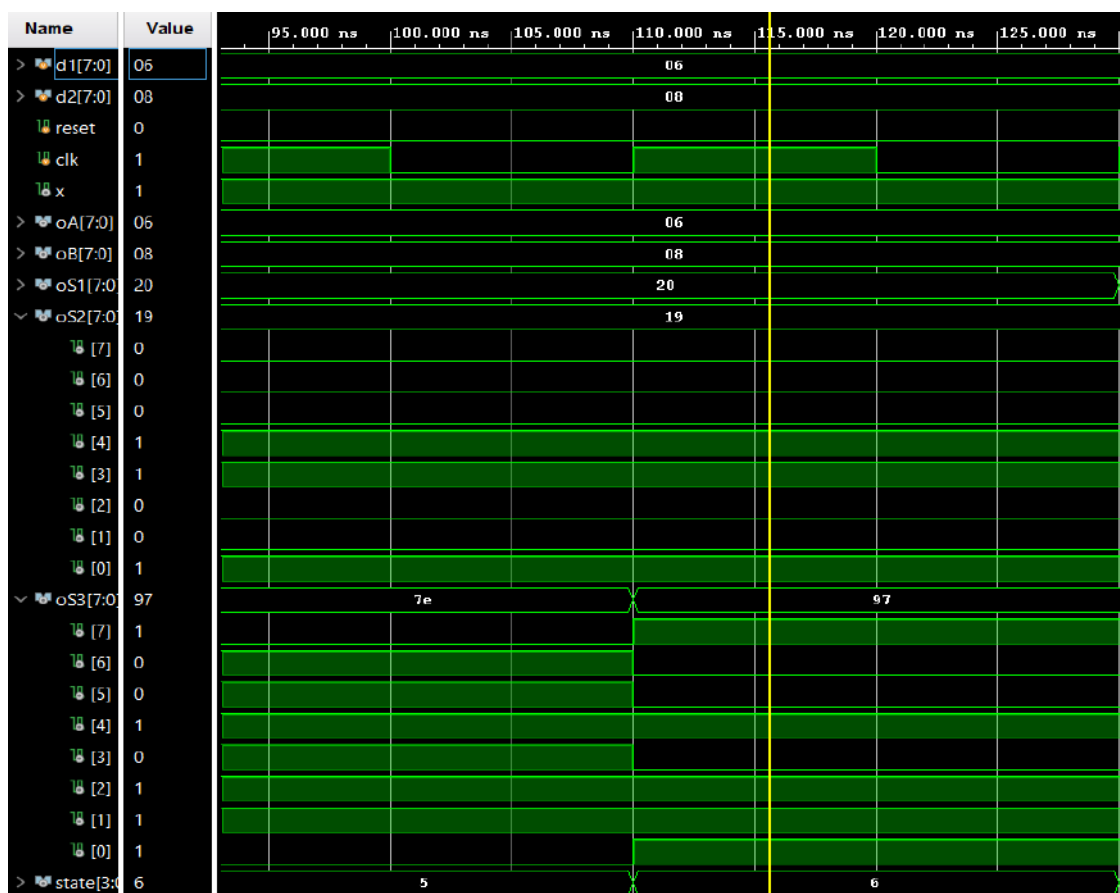


Рисунок 5.6 – стан S6: виконання двомісної операції $S3 = S3 + S2$

Згідно з рис. 5.6, в стані S6 отримано результат двомісної операції $S3 = S3 + S2$. Значення операндів S3, S2 визначено в попередньому стані S5 та дорівнюють $7E_{16}$ (126_{10}), 19_{16} (25_{10}), відповідно. Перевірено правильність виконання двомісної операції: $S3 = 126_{10} + 25_{10} = 151_{10}$ (97_{16}).

На рис. 5.7 наведено результати виконання операцій у стані S7 керуючого автомата.

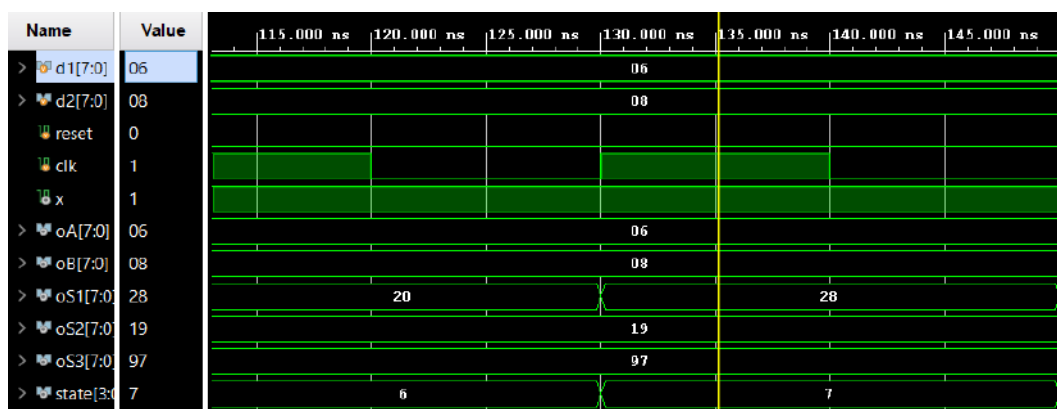


Рисунок 5.7 – Стан S7: виконання двомісної операції $S1 = B + S1$

Згідно з рис. 5.7, в стані S7 отримано результат двомісної операції $S1 = B + S1$. Значення операндів B, S1 визначено в попередньому стані S6 та дорівнюють 8_{10} , 20_{16} (32_{10}), відповідно. Перевірено правильність виконання двомісної операції: $S1 = 8_{10} + 20_{10} = 40_{10}$ (28_{16}).

На рис. 5.8 наведено результати виконання операцій у стані S8 керуючого автомата.

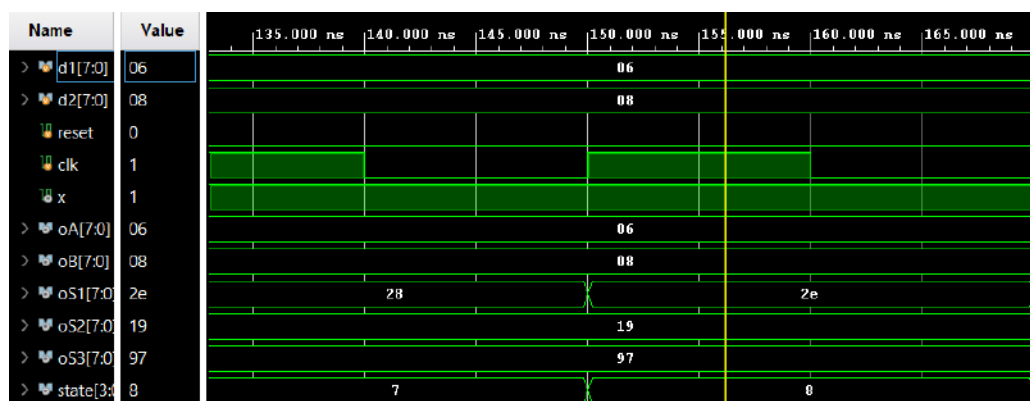


Рисунок 5.8 – Стан S8: виконання двомісної операції $S1 = A + S1$

Згідно з рис. 5.8, в стані S8 отримано результат двомісної операції $S1 = A + S1$. Значення операндів A, S1 визначено в попередньому стані S7 та дорівнюють 6_{10} , 28_{16} (40_{10}), відповідно. Перевірено правильність виконання двомісної операції: $S1 = 6_{10} + 40_{10} = 46_{10}$ ($2E_{16}$).

На рис. 5.10 наведено результати виконання операцій у стані S9 керуючого автомата.

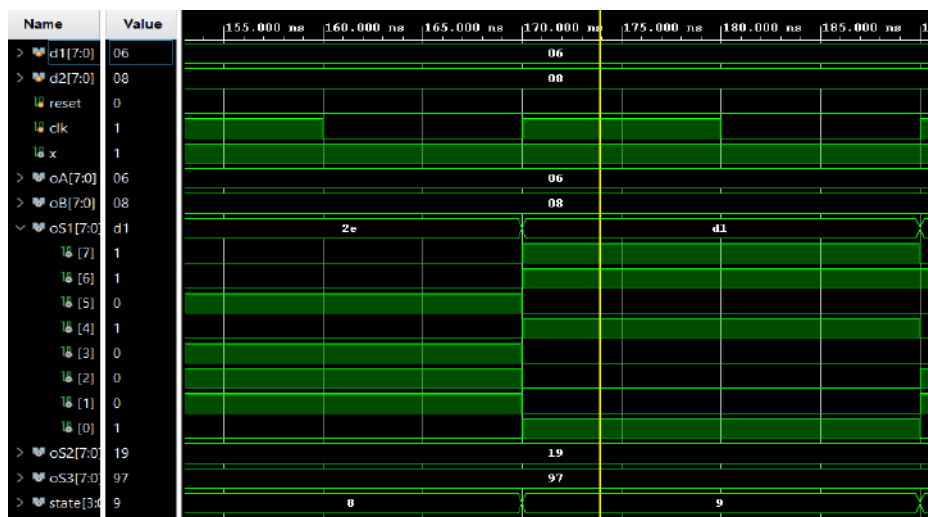


Рисунок 5.9 – Стан S9: виконання одномісної операції $S1 = !S1$

Згідно з рис. 5.9, в стані S9 отримано результат одномісної операції $S1 = !S1$. Значення операнда S1 визначене в попередньому стані S8 та дорівнює 00101110_2 . Перевірено правильність виконання одномісної операції: $S1 = !00101110_2 = 11010001_2$.

На рис. 5.10 наведено результати виконання операцій у стані S10 керуючого автомата.

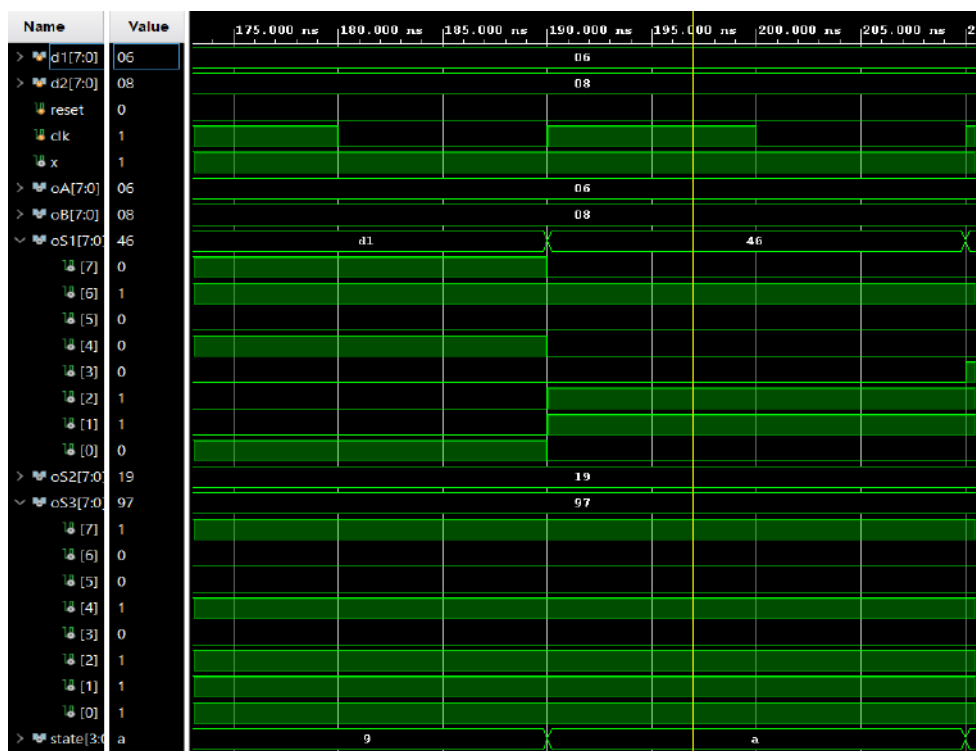


Рисунок 5.10 – Стан S10: двомісна операція $S1 = S1 \oplus S3$

Згідно з рис. 5.10, в стані S10 отримано результат двомісної операції $S1 = S1 \oplus S3$. Значення операндів S1, S3 визначено в попередньому стані S9 та дорівнюють 11010001_2 , 10010111_2 , відповідно. Перевірено правильність виконання двомісної операції, згідно з таблицею істинності операції XOR (див. рис. 5.3): $S1 = 11010001_2 \oplus 10010111_2 = 01000110_2 (46_{16})$.

5.2.2 Перевірка гілки алгоритму, де умова не виконується

Для перевірки правої гілки алгоритму необхідно розкоментувати відповідний рядок в тестовому модулі (див. лістинг 5.2).

На рис. 5.11 наведено результати виконання операцій у стані S1 керуючого автомата.

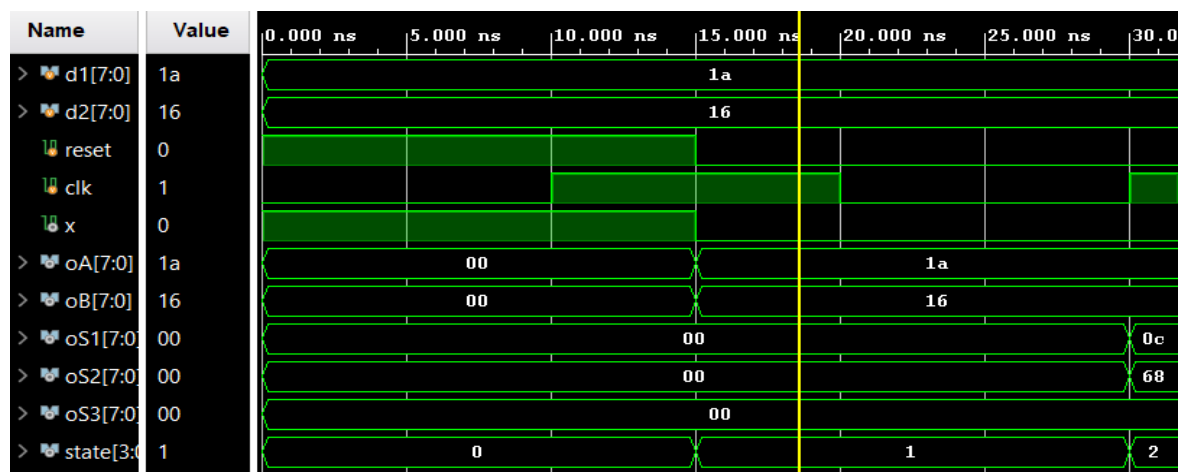


Рисунок 5.11 – Стан S1: виконання операцій ініціалізації $A = d1$ та $B = d2$

Згідно з рис. 5.11, в стані S1 отримано результати операцій ініціалізації $A = d1$ та $B = d2$. Значення операндів d1, d2 визначено в попередньому стані S0 та дорівнюють $1A_{16}$, 16_{16} , відповідно. Перевірено правильність виконання операцій ініціалізації: $A = d1 = 1A_{16}$, $B = d2 = 16_{16}$.

На рис. 5.12 наведено результати виконання операцій у стані S2 керуючого автомата. Згідно з рис. 5.12, в стані S2 отримано результати одномісної операції $S2 = L2(A)$, яка еквівалентна операції $4_{10} * A$, та двомісної операції $S1 = A \oplus B$. Значення операндів A, B визначено в попередньому стані S1 та дорівнюють $1A_{16}$ (26_{10} , 00011010_2), 16_{16} (00010111_2), відповідно.

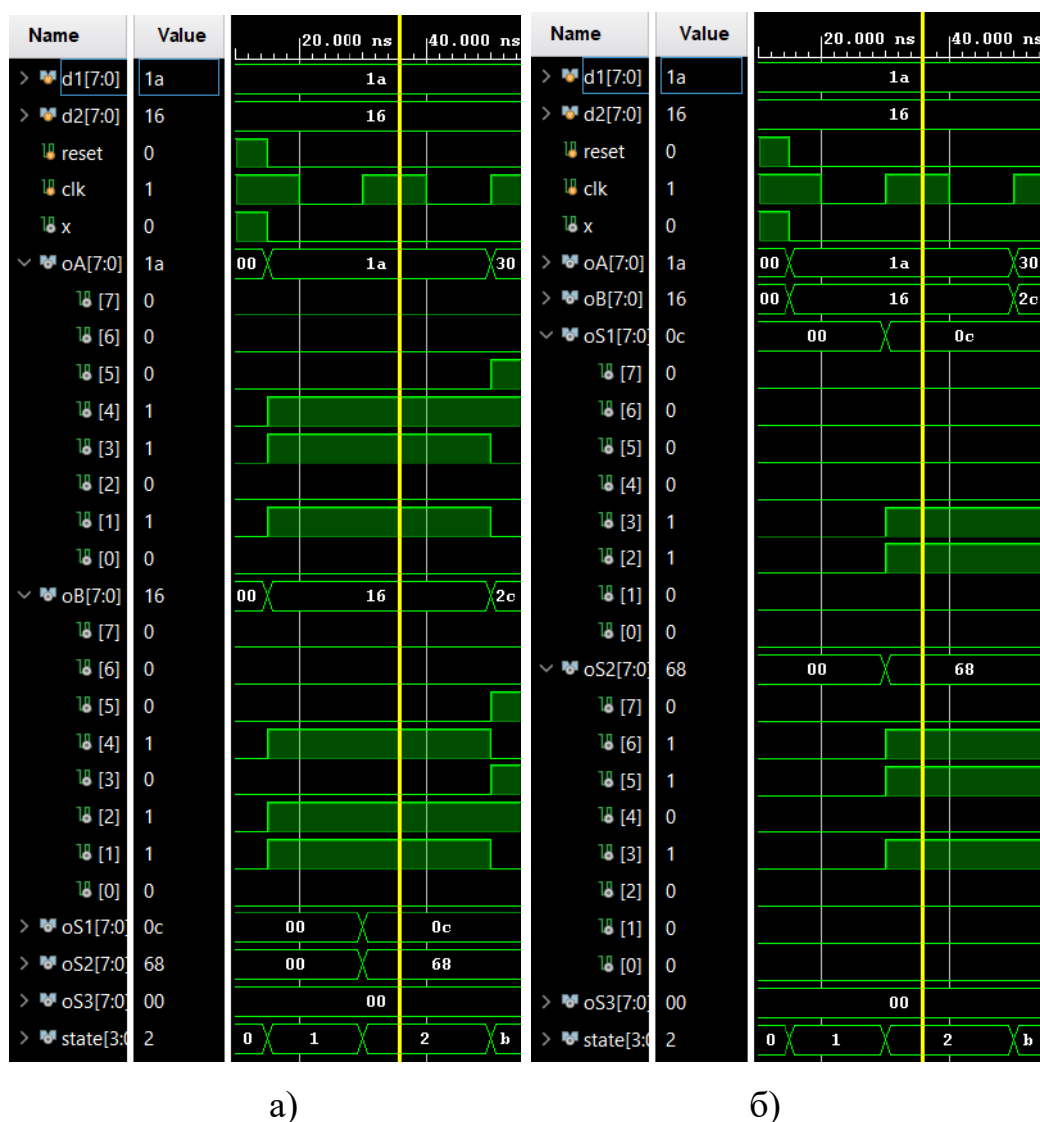


Рисунок 5.12 – Стан S2: виконання двомісної $S1 = A \oplus B$ та одномісної $S2 = L2(A)$ операцій, де а) – вміст регістрів A, B; б) – вміст регістрів S1, S2

Перевірено правильність виконання одномісної операції: $S2 = 4_{10} * A = 4_{10} * 26_{10} = 104_{10} (64_{16})$. Перевірено правильність виконання двомісної операції: $S1 = 00011010_2 \oplus 00010110_2 = 00001100_2 (C_{16})$.

Тому, як вміст регістру A дорівнює $1A_{16} (26_{10})$, то умова $A \leq 25_{10}$ не виконується. Отже, наступним станом повинен бути S11 (виконується права гілка алгоритму).

На рис. 5.13 наведено результати виконання операцій у стані S11 керуючого автомата.

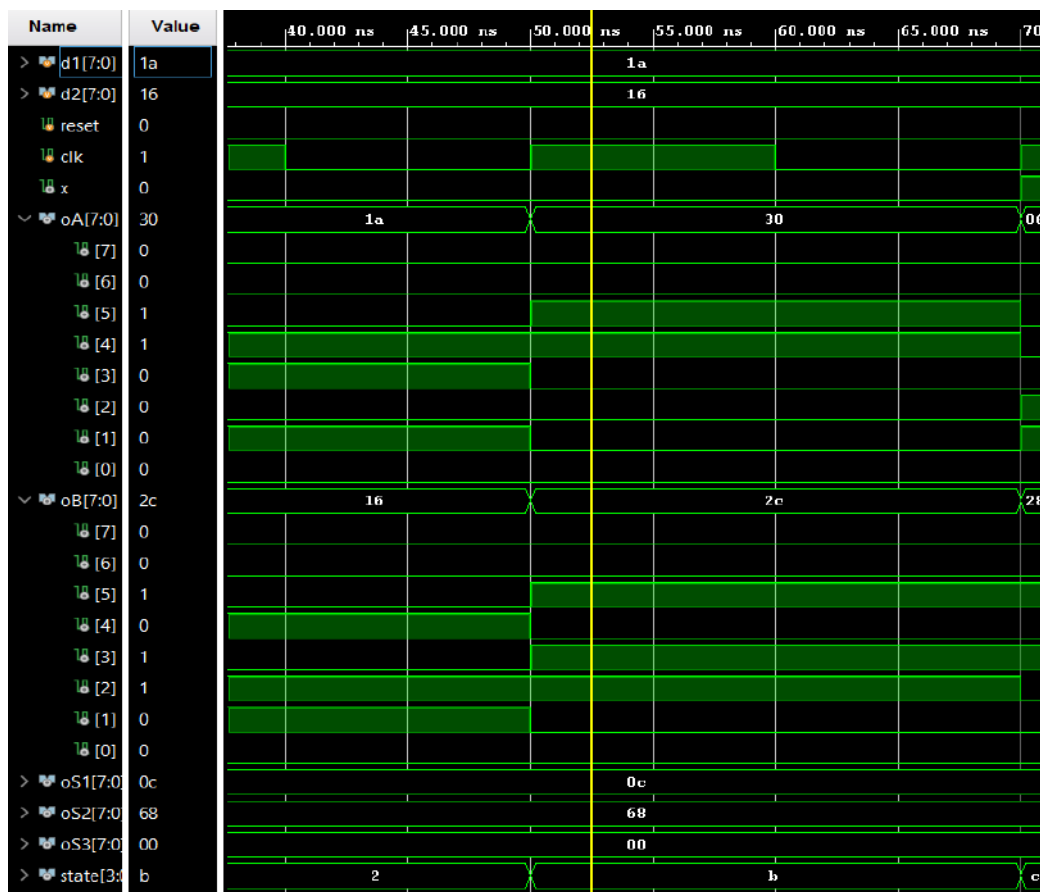


Рисунок 5.13 – Стан S11: виконання двомісної $A = A + B$
та одномісної $B = L1(B)$ операцій

Згідно з рис. 5.13, в стані S11 отримано результати одномісної операції $B = L1(B)$, яка еквівалентна операції $2_{10} * B$, та двомісної операції $A = A + B$. Значення операндів A, B визначено в попередньому стані S10 та дорівнюють $1A_{16}$ (26_{10}), 16_{16} (22_{10} , 00010110_2), відповідно.

Перевірено правильність виконання одномісної операції: $B = 2_{10} * B = 2_{10} * 22_{10} = 44_{10}$ ($2C_{16}$). Перевірено правильність виконання двомісної операції: $A = 26_{10} + 22_{10} = 48_{10}$ (30_{16}).

На рис. 5.14 наведено результати виконання операцій у стані S12 керуючого автомата. Згідно з рис. 5.14, в стані S12 отримано результати одномісної операції $A = R3(A)$, яка еквівалентна операції $A / 8_{10}$, та двомісної операції $B = B \text{ and } S2$. Значення операндів A, B, S2 визначено в попередній стан S11 та дорівнюють 30_{16} (48_{10}), $2C_{16}$ (00101100_2), 68_{16} (01101000_2), відповідно.



а)

б)

Рисунок 5.14 – Стан S12: виконання двомісної $B = B$ and S2 та одномісної $A = R3(A)$ операцій, де а) – вміст регістрів B, S2; б) – вміст регістрів A, S2

Перевірено правильність виконання одномісної операції: $A = A / 8_{10} = 48_{10} / 8_{10} = 6_{16}$. Перевірено правильність виконання двомісної операції: $B = 00101100_2$ and $01101000_2 = 00101000_2$ (28_{16}).

На рис. 5.15 наведено результати виконання операцій у стані S13 керуючого автомата. Згідно з рис. 5.15, в стані S13 отримано результат одномісної операції $B = !B$. Значення операнда B визначене в попередньому стані S12 та дорівнює 00101000_2 . Перевірено правильність виконання одномісної операції: $B = !0010.1000_2 = 1101.0111_2$.

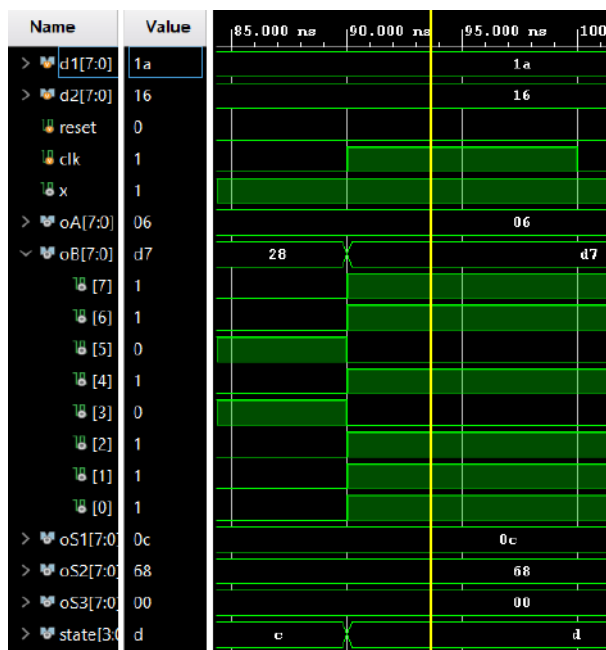


Рисунок 5.15 – Стан S13: виконання одномісної операції $B = !B$

На рис. 5.16 наведено результати виконання операцій у стані S14 керуючого автомата.

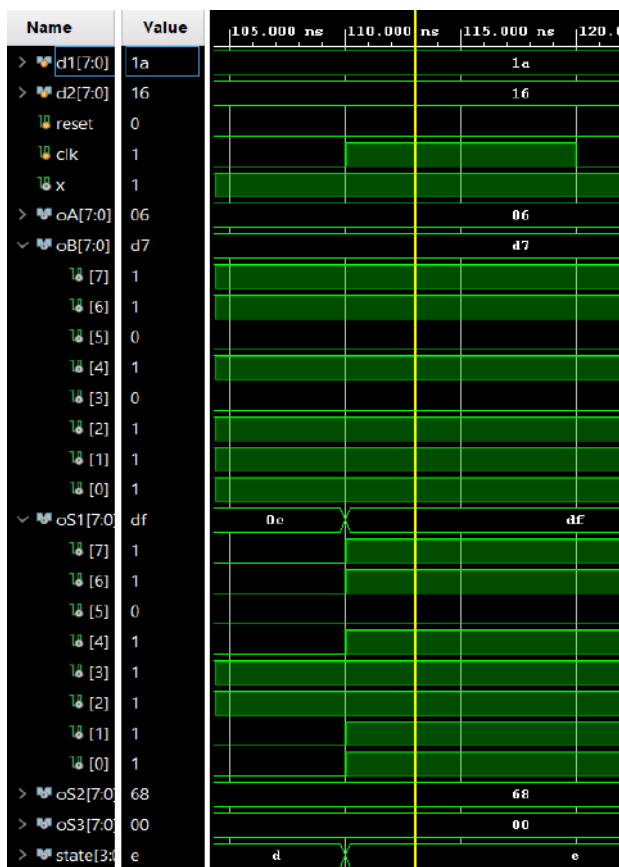


Рисунок 5.16 – Стан S14: виконання двомісної операції $S1 = B \text{ or } S1$

Згідно з рис. 5.16, в стані S14 отримано результат двомісної операції $S1 = B \text{ or } S1$. Значення операндів B, S1 визначено в попередньому стані S13 та дорівнюють $D7_{16}$ (11010111_2), C_{16} (00001100_2), відповідно. Перевірено правильність виконання двомісної операції: $S1 = 11010111_2 \text{ or } 00001100_2 = 11011111_2$.

На рис. 5.17 наведено результати виконання операцій у стані S15 керуючого автомата.

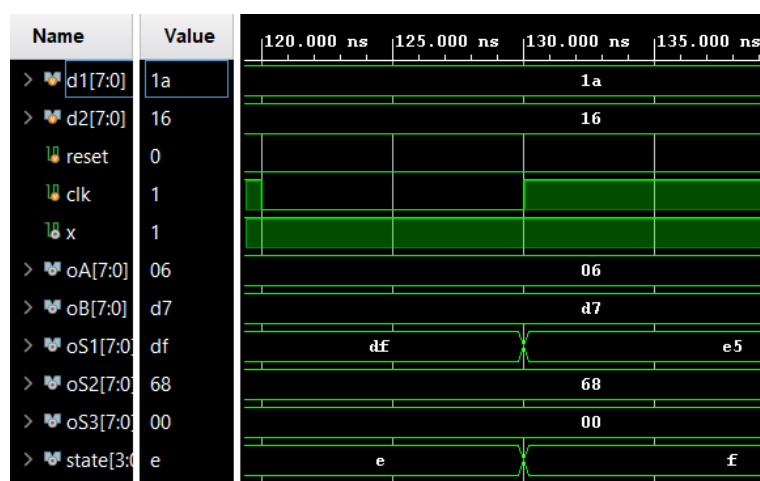


Рисунок 5.17 – Стан S15: двомісна операція $S1 = A + S1$

Згідно з рис. 5.17, в стані S15 отримано результат двомісної операції $S1 = A + S1$. Значення операндів A, S1 визначено в попередньому стані S14 та дорівнюють 6_{10} , DF_{16} (223_{10}), відповідно. Перевірено правильність виконання двомісної операції: $S1 = 6_{10} + 223_{10} = 229_{10}$ ($E5_{16}$).

5.3 Апаратна реалізація

На лістингу 5.3 наведено опис спецобчислювача, призначений для апаратної реалізації. Для зменшення ресурсомісткості апаратної реалізації даний опис не містить виходів, призначених для перегляду вмісту внутрішніх регістрів. Даний опис складається лише з одного модуля, що включає в собі описи ОА та КА.

Лістинг 5.3 – Verilog-опис спецобчислювача, призначений для апаратної реалізації

```

`timescale 1ns / 1ps
module specialized_digital_device2(d1, d2, reset, clk, oS1);
//зовнішні інтерфейси
input [0:7] d1, d2;
input reset, clk;
output [0:7] oS1;
//внутрішні зв'язки
reg [0:7] A, B, S1, S2, S3;
reg [3:0] Q;
wire [19:0] Y;
wire X;
wire [3:0] D;
//опис керуючого автомата
//перевірка умови та запис результату на вихідний інтерфейс X
assign X = A<=25;
//запис результату обчислення формули на вихідний інтерфейс oS1
assign oS1 = S1;
assign D[0] = ~Q[0]&~Q[1] | ~Q[0]&~Q[3] |
~Q[0]&Q[2];
assign D[1] = Q[0]&~Q[1] | ~Q[0]&Q[1]&~Q[3] | ~Q[0]&Q[1]&Q[2];
assign D[2] = ~Q[0]&Q[2] | ~Q[1]&Q[2] | Q[0]&Q[1]&~Q[2];
assign D[3] = Q[0]&~Q[2]&Q[3] | ~Q[0]&Q[2]&Q[3] | !Q[1]&Q[3] |
~Q[3]&Q[0]&Q[1]&Q[2] | ~Q[0]&Q[1]&~Q[2]&~Q[3]&X;
assign Y[0] = ~Q[3]&~Q[2]&~Q[1]&Q[0];
assign Y[1] = ~Q[3]&~Q[2]&~Q[1]&Q[0];
assign Y[2] = ~Q[3]&~Q[2]&Q[1]&~Q[0];
assign Y[3] = ~Q[3]&~Q[2]&Q[1]&~Q[0];
assign Y[4] = ~Q[3]&~Q[2]&Q[1]&Q[0];
assign Y[5] = ~Q[3]&Q[2]&~Q[1]&~Q[0];
assign Y[6] = ~Q[3]&Q[2]&~Q[1]&~Q[0];
assign Y[7] = ~Q[3]&Q[2]&~Q[1]&Q[0];
assign Y[8] = ~Q[3]&Q[2]&~Q[1]&Q[0];
assign Y[9] = ~Q[3]&Q[2]&Q[1]&~Q[0];

```

```

assign Y[10] = ~Q[3]&Q[2]&Q[1]&Q[0];
assign Y[11] = Q[3]&~Q[2]&~Q[1]&~Q[0] | Q[3]&Q[2]&Q[1]&Q[0];
assign Y[12] = Q[3]&~Q[2]&~Q[1]&Q[0];
assign Y[13] = Q[3]&~Q[2]&Q[1]&~Q[0];
assign Y[14] = Q[3]&~Q[2]&Q[1]&Q[0];
assign Y[15] = Q[3]&~Q[2]&Q[1]&Q[0];
assign Y[16] = Q[3]&Q[2]&~Q[1]&~Q[0];
assign Y[17] = Q[3]&Q[2]&~Q[1]&~Q[0];
assign Y[18] = Q[3]&Q[2]&~Q[1]&Q[0];
assign Y[19] = Q[3]&Q[2]&Q[1]&~Q[0];
always@(posedge clk)
if(!reset) Q = D;
else Q = 0;
//опис операційного автомата
always @ (Y, reset)
if (!reset)
    case (Y)
3: begin
    A <= d1;
    B <= d2;
    end
12: begin
    S2 <= A >> 2;
    S1 <= A^B;
    end
16:S3 = B << 3;
96: begin
S3 <= S1 >> 3;
    S2 <= S2 + S3;
    end
384: begin
    S1 <= B >> 2;
    S3 <= S3 + S1;
    end
512: S3 = S3 + S2;

```

```

1024: S1 = S1 + B;
2048: S1 = A + S1;
4096: S1 = ~S1;
8192: S1 = S1^S3;
49152: begin
    B <= B>>1;
    A <= A+B;
end
196608: begin
    A <= A << 3;
    B <= B & S2;
end
262144: B=~B;
default: S1 = B | S1;
endcase
else
begin
    A <=0; B <= 0; S1 <= 0;
    S2 <= 0; S3 <= 0;
end
endmodule

```

Апаратна реалізація виконується на базі віртуальної Xilinx FPGA під кодовою назвою 7k70t (серія Kintex 7) з використанням пакета fbv676 та швидкістю «1» (повна назва пристрою xc7k70tfbv676-1 [32]).

У результаті імплементації опису спецобчислювача, згідно з лістингом 5.3, виникає попередження (рис. 5.21).

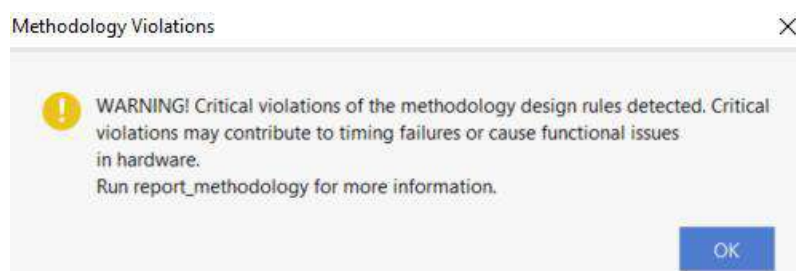
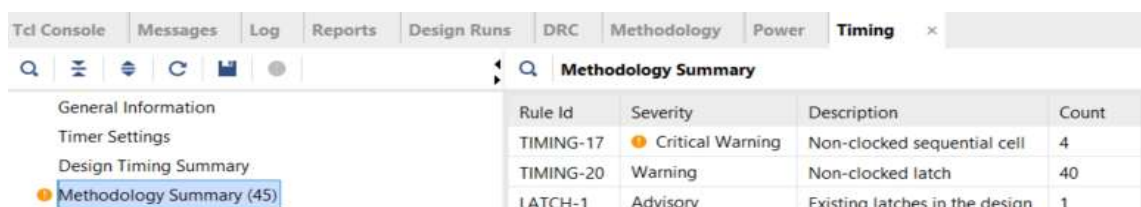


Рисунок 5.21 – Попередження після імплементації

Згідно рис. 5.21, наявні критичні попередження в звіті про відповідність методології UltraFast Design (звіт «Methodology») [33]. Дане попередження може свідчити про порушення, на функціональному рівні, апаратної реалізації спецобчислювача. Для підтвердження наявності функціональних порушень необхідно провести функціональну симуляцію результатів імплементації (англ. post-implementation functional simulation) [34].

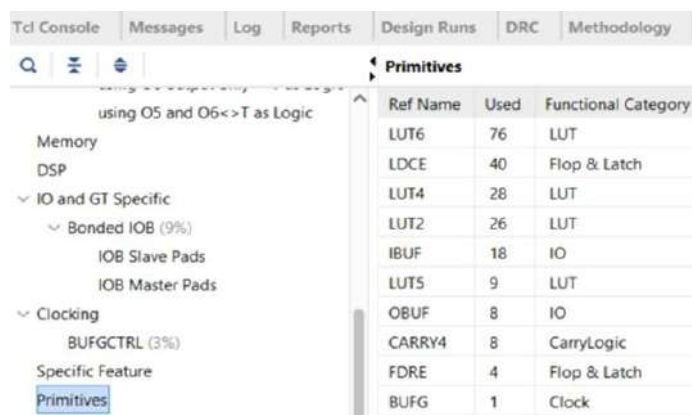
В результаті імплементації генерується HDL опис, що, в разі успіху, в точності реалізує логіку вхідного HDL, що імплементується, на RTL (register-transfer level) рівні [31, с. 319; 35, с. 7], використовуючи в якості елементної бази стандартні комірки виробника цільової FPGA [31, с. 318]. Саме такий згенерований HDL опис виступає в якості опису, що симулюється на функціональному (поведінковому) рівні на етапі «post-implementation functional simulation».

З огляду на звіти «Timing» (рис. 5.22, а) та «Utilization» (рис. 5.22, б) можна стверджувати, що всі тригери є асинхронними.



Rule Id	Severity	Description	Count
TIMING-17	Critical Warning	Non-clocked sequential cell	4
TIMING-20	Warning	Non-clocked latch	40
LATCH-1	Advisory	Existing latches in the design	1

а)

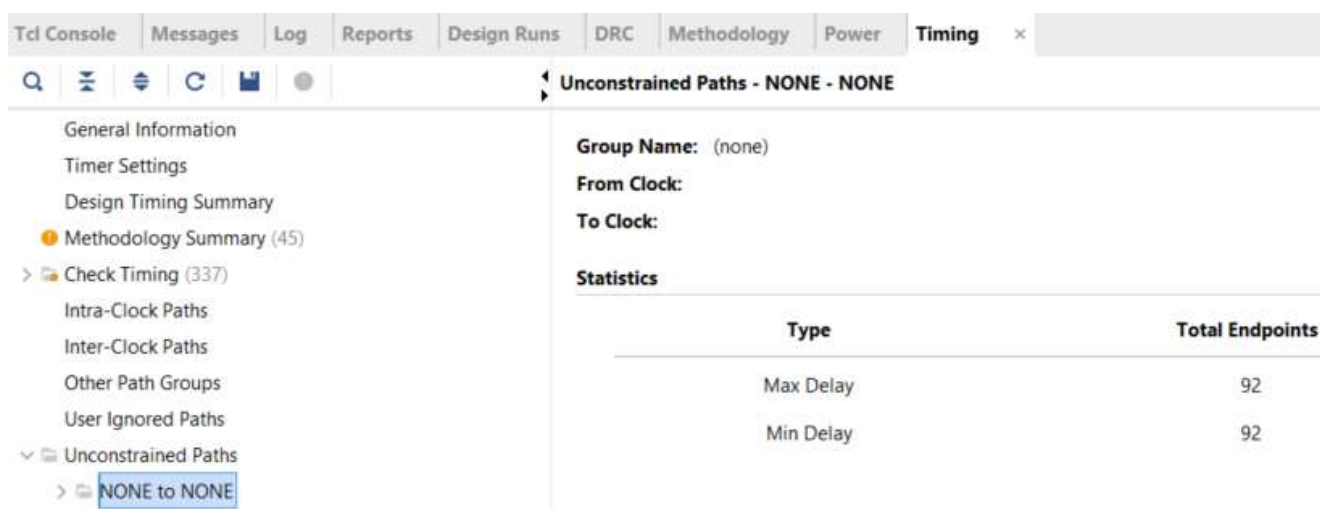


Ref Name	Used	Functional Category
LUT6	76	LUT
LDCE	40	Flop & Latch
LUT4	28	LUT
LUT2	26	LUT
IBUF	18	IO
LUT5	9	LUT
OBUF	8	IO
CARRY4	8	CarryLogic
FDRE	4	Flop & Latch
BUFG	1	Clock

б)

Рисунок 5.22 – Звіт: а) – «Timing» (результат підрахунку попереджень в звіті «Methodology»); б) – «Utilization»

Тому в апаратній реалізації не використовуються синхросигнали, то розділ «необмежені шляхи» (англ. unconstrained paths) в звіті «Timing» (рис. 5.23) дає уявлення про відносну швидкодію апаратної реалізації. Зокрема, показник «total endpoints» можна використовувати при порівнянні апаратних реалізацій на базі спільної FPGA. В даному випадку, даний показник вказує на мінімальну та максимальну кількості логічних блоків на найдовшому шляху між вхідними та вихідними портами апаратної реалізації [36]. Для апаратної реалізації спецобчислювача значення цього показника дорівнює 92.



Unconstrained Paths - NONE - NONE	
Group Name: (none)	
From Clock:	
To Clock:	
Statistics	
Type	Total Endpoints
Max Delay	92
Min Delay	92

Рисунок 5.23 – Звіт «Timing» (перегляд значення показника «total endpoints»)

Також, в звіті «Timing» вказано затримки встановлення T_{setup} (рис. 5.24, а) та утримання T_{hold} (рис. 5.24, б) на шляхах між вхідними та вихідними портами апаратної реалізації. T_{setup} та T_{hold} це проміжки часу перед та після, відповідно, фронтом синхросигналу, протягом яких на вході тригера повинен бути встановлений певний логічний рівень для перемикання тригера [37, с. 39].

Згідно з рис. 5.24, корисними показниками швидкодії можуть бути агреговані значення на основі всіх шляхів: середня, мінімальна та максимальна затримки, обчисленні на основі стовпця «Total Delay».

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	∞	2	7	S1_reg[5]/G	oS1[5]	4.779	2.937	1.842
Path 2	∞	2	7	S1_reg[7]/G	oS1[7]	4.617	2.924	1.693
Path 3	∞	2	8	S1_reg[11]/G	oS1[11]	4.548	2.920	1.627
Path 4	∞	2	7	S1_reg[6]/G	oS1[6]	4.542	2.963	1.579
Path 5	∞	2	8	S1_reg[2]/G	oS1[2]	4.479	2.938	1.541
Path 6	∞	2	8	S1_reg[3]/G	oS1[3]	4.361	2.935	1.426
Path 7	∞	2	8	S1_reg[4]/G	oS1[4]	4.352	2.935	1.417
Path 8	∞	5	74	Q_reg[2]/C	S1_reg[4]/D	4.329	1.057	3.272
Path 9	∞	2	7	S1_reg[0]/G	oS1[0]	4.297	2.924	1.372
Path 10	∞	6	74	Q_reg[2]/C	S1_reg[2]/D	4.247	1.218	3.029

а)

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement
Path 11	∞	2	73	Q_reg[3]/C	A_reg[7]/D	0.319	0.128	0.191	-∞
Path 12	∞	2	73	Q_reg[3]/C	Q_reg[3]/D	0.324	0.128	0.196	-∞
Path 13	∞	2	76	Q_reg[0]/C	Q_reg[0]/D	0.335	0.129	0.206	-∞
Path 14	∞	2	74	Q_reg[2]/C	Q_reg[2]/D	0.342	0.146	0.196	-∞
Path 15	∞	2	76	Q_reg[0]/C	S2_reg[5]/D	0.343	0.128	0.215	-∞
Path 16	∞	2	76	Q_reg[0]/C	S2_reg[4]/D	0.344	0.128	0.216	-∞
Path 17	∞	2	76	Q_reg[1]/C	Q_reg[1]/D	0.357	0.128	0.229	-∞
Path 18	∞	2	73	Q_reg[3]/C	S2_reg[0]/D	0.389	0.129	0.260	-∞
Path 19	∞	2	76	Q_reg[1]/C	S2_reg[7]/D	0.393	0.128	0.265	-∞
Path 20	∞	2	8	B_reg[1]/G	B_reg[1]/D	0.408	0.214	0.194	-∞

б)

Рисунок 5.24 – Звіт «Timing»: а) – затримки встановлення;

б) – затримки утримання

З огляду на звіт «Methodology» (рис. 5.25) підтверджуються застереження описані в попередженні, яке отримано одразу після імплементації (рис. 5.21). В даному звіті є критичні попередження, які повідомляють, що використання асинхронних тригерів є поганою практикою.

Name	Severity	Details
All Violations (45)		
Timing (44)		
Bad Practice (44)		
TIMING-17 (4)		
TIMING #1	Critical Warning	The clock pin Q_reg[0].C is not reached by a timing clock
TIMING #2	Critical Warning	The clock pin Q_reg[1].C is not reached by a timing clock
TIMING #3	Critical Warning	The clock pin Q_reg[2].C is not reached by a timing clock
TIMING #4	Critical Warning	The clock pin Q_reg[3].C is not reached by a timing clock

Рисунок 5.25 – Звіт «Methodology»

В звіті «Power» вказано показники живлення (рис. 5.26). Далі, P – споживана потужність у ватах, T – температура у цельсіях.

$T_{\text{перехід}}$ (англ. junction temperature) [38] – робоча температура при включенні та конфігуруванні (ініціалізації конфігураційних комірок) FPGA. $T_{\text{перехід}}$ вища за 125 °C вважається недопустимою в незалежності від встановленого порогового значення користувачем.

$P_{\text{запас}}$ (англ. thermal margin) [39] – запас потужності відносно максимального значення, що відповідає цільовій FPGA. $P_{\text{запас}}$ є від’ємним, якщо $T_{\text{перехід}}$ перевищує порогове значення. В цьому випадку необхідно усунути надлишкову потужність, яку споживає апаратна реалізація.

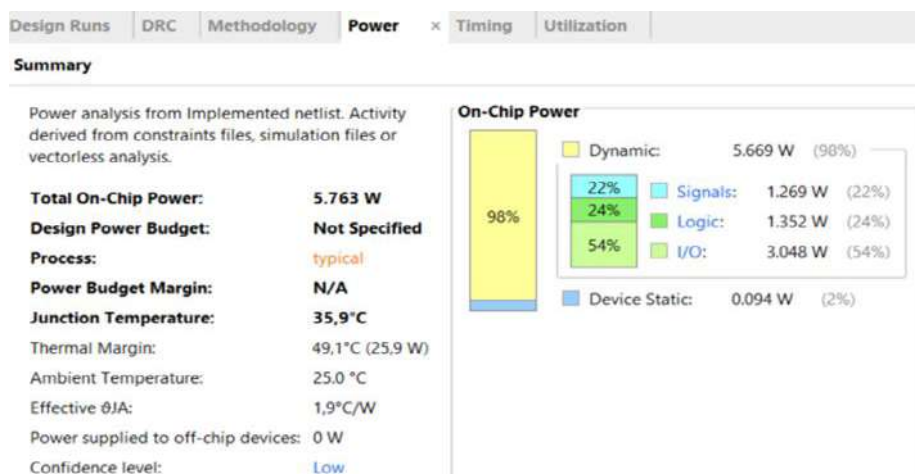


Рисунок 5.26 – Звіт «Power»

Згідно з рис. 5.26, жодний з показників живлення не виділено помаранчевим кольором. Тому, жодне з порогових значень не перевищено. Порогові значення для всіх показників залишено за замовчуванням. Проте, згідно з параметром «Confidence level», точність оцінки показників живлення є низькою [40]. На рис. 5.27 наведено детальний опис параметра «Confidence level».

Згідно з рис. 5.27, причиною низької точності оцінки є використання стандартних обмежень на розташування (англ. location constraints, LOC) для понад 75% I/O (input/output) портів та відсутність 75% синхросигналів.

Confidence Level Details	
Design State:	High Design is routed
Clock Activity:	Low User specified less than 75% of clocks
I/O Activity:	Low More than 75% of inputs are missing user specification
Internal Activity:	Medium User specified less than 25% of internal nodes
Characterization Data:	High Device models are Production

Рисунок 5.27 – Детальний опис параметра «Confidence level»

Це підтверджується наявністю відповідних попереджень в звіті «DRC» (рис. 5.28). Також, наявність такого попередження може призвести до пошкодження реальної FPGA або пристроїв, підключених до неї. Разом з цим, наявність таких попереджень блокує можливість генерації бітового потоку, що необхідно для конфігурування (прошивки) реальної FPGA.

Severity ^1 Details	
All Violations (8)	
Pin Planning (3)	
NSTD-1 (1)	
Critical Warning	26 out of 26 logical ports use I/O standard (IOSTANDARD) value 'DEFAULT', instead of a user assigned specific value. This may cause I/O contention or incompatibility with the board power or connectivity affecting performance, signal integrity or in extreme cases cause damage to the device or the components to which it is connected. To correct this violation, specify all I/O standards. This design will fail to generate a bitstream unless all logical ports have a user specified I/O standard value defined. To allow bitstream creation with unspecified I/O standard values (not recommended), use this command: <code>set_property SEVERITY {Warning} [get_drc_checks NSTD-1]</code> . NOTE: When using the Vivado Runs infrastructure (e.g. <code>launch_runs Tcl</code> command), add this command to a .tcl file and add that file as a pre-hook for write_bitstream step for the implementation run. Problem ports: d110:71 , d210:71 , o5110:71 , clk , reset .
UCIO-1 (1)	
Critical Warning	26 out of 26 logical ports have no user assigned specific location constraint (LOC). This may cause I/O contention or incompatibility with the board power or connectivity affecting performance, signal integrity or in extreme cases cause damage to the device or the components to which it is connected. To correct this violation, specify all pin locations. This design will fail to generate a bitstream unless all logical ports have a user specified site LOC constraint defined. To allow bitstream creation with unspecified pin locations (not recommended), use this command: <code>set_property SEVERITY {Warning} [get_drc_checks UCIO-1]</code> . NOTE: When using the Vivado Runs infrastructure (e.g. <code>launch_runs Tcl</code> command), add this command to a .tcl file and add that file as a pre-hook for write_bitstream step for the implementation run. Problem ports: d110:71 , d210:71 , o5110:71 , clk , reset .

Рисунок 5.28 – Звіт «DRC»

Проте, згідно з рис. 5.28, за допомогою відповідної консольної команди можна розблокувати можливість генерації бітового потоку, проігнорувавши попередження, але цього робити не рекомендується. Для уникнення такого попередження рекомендується явно вказати розташування всіх використаних I/O портів.

ВИСНОВКИ

Розглянуто структуру ОА в деяких процесорних ядрах. При спробі класифікації ОА, з'ясовано, що базова класифікація [3, с. 216] є не актуальною через неможливість класифікувати за її допомогою ОА в сучасних процесорних ядрах. Тому, запропоновано власну класифікацію ОА [4, с. 553-554], згідно з якою виконано класифікацію ОА розглянутих процесорних ядер.

В результаті класифікації (див. табл. 2.9), з'ясовано, що в сучасних процесорних ядрах найчастіше використовуються Р-автомати [4, с. 555]. S-автомат широко використовувався в ранніх моделях процесорних ядрах [4, с. 555]. Хоча, зараз даний автомат також використовується в ультраенергоефективних процесорних ядрах. РС-автомат використовується у високопродуктивних паралельних обчисленнях, зокрема, в ядрах сигнальних процесорів.

В ядрах, які застосовуються в одноядерних пристроях, використовуються ОА як з послідовною (S-автомат), так і з паралельною (Р- або РС-автомат) структурами. В ядрах, які не застосовуються в одноядерних пристроях, використовуються ОА лише з паралельною структурою.

Створено функціональну ГСА (див. рис. 3.1) на основі заданої формули 1.1 та вимог до операційного автомата з комбінованою структурою. На основі функціональної ГСА завершено теоретичне проєктування операційного автомата та керуючого автомата Мура. В результаті теоретичного проєктування операційного автомата побудовано його функціональну схему (див. рис. 3.3). В результаті теоретичного проєктування керуючого автомата отримано рівняння виходів та збудження тригерів.

Створено описи апаратури для обох автоматів на мові Verilog та тестові модулі для їх поведінкової симуляції. Об'єднано описи обох автоматів в єдиний модуль спецобчислювача (див. лістинг 5.3) та виконано його апаратну реалізацію на базі віртуальної FPGA (xc7k70tfbv676-1). Поведінкову симуляцію та апаратну реалізацію проведено за допомогою програмного середовища Vivado (v2023.2).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Лахно В.А. Комп'ютерна логіка [Текст] / В.А. Лахно, Б.С. Гусев, Д.Ю. Касаткін; КОМПІНТ, Київ, 2018. – 422 с. – Режим доступу: https://nubip.edu.ua/sites/default/files/u34/kompyuterna_logika_2018_lahno.pdf
2. Tanenbaum A. S. Structured computer organization [Text] / A. S. Tanenbaum, T. Austin; Pearson, 2021. – 6th ed. – 801 p. – Access mode: <https://csc-knu.github.io/sys-prog/books/Andrew%20S.%20Tanenbaum%20-%20Structured%20Computer%20Organization.pdf>
3. Null L. The Essentials of Computer Organization and Architecture [Text] / L. Null; Jones & Bartlett India Private Limited, 2015. – 884 p. – Access mode: <https://archive.org/details/essentialsofcomp0000lind/page/216/mode/2up>
4. Бутко В.О., Зеленьова І.Я., Грушко С. С. Огляд сучасних структур операційної частини процесорного ядра. XXI Міжнародній науково-практичній конференції «Theoretical methods of research of the latest problems», Прага, Чеська Республіка, 27-29 трав. 2024 р. С. 553-555. – Режим доступу: <https://eu-conf.com/wp-content/uploads/2024/04/9-789-40372-404-1.pdf>
5. Reverse-engineering the surprisingly advanced ALU of the 8008 microprocessor [Electronic resource]. – Access mode: <https://www.righto.com/2017/02/reverse-engineering-surprisingly.html>
6. Diefendorff K. The Russians Are Coming. Supercomputer Maker Elbrus Seeks to Join x86/IA-64 Melee [Text] / K. Diefendorff; Microprocessor report, 1999. – №2. – 7 p. – Access mode: <https://www.cecs.uci.edu/~papers/mpr/MPR/19990215/130201.pdf>
7. TMS320C6000 CPU and Instruction Set Reference Guide [Text] / Texas Instruments, 1999. – 391 p. – Access mode: <https://www.rose-hulman.edu/class/ee/yoder/ece581/ECE581%20documents/Old/spru189d.pdf>
8. Sunny Cove - Microarchitectures - Intel [Electronic resource]. – Access mode: https://en.wikichip.org/wiki/intel/microarchitectures/sunny_cove

9. Popping the Hood on Golden Cove [Electronic resource]. – Access mode: <https://chipsandcheese.com/2021/12/02/popping-the-hood-on-golden-cove/>
10. Cortex-A8 [Electronic resource]. – Access mode: <https://developer.arm.com/Processors/Cortex-A8>
11. Architecture and Implementation of the ARM Cortex-A8 microprocessor [Text] / ARM, 2005. – 10 p. – Access mode: <https://people.freebsd.org/~chuckr/arm/TigerWhitepaperFinal.pdf>
12. Arm Cortex-M0+ Processor Datasheet [Text] / ARM, 2020. – 8 p. – Access mode: <https://documentation-service.arm.com/static/620545c494e7af28dd7c9cbc>
13. Gao S. Share-slicing: Friend or Foe? [Text] / S. Gao, B. Marshall , D. Page, E. Oswald; IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020. – №1. – 23 p. – Access mode: <https://tches.iacr.org/index.php/TCHES/article/view/8396/7780>
14. STM32G0 – ARM Core. ARM Cortex-M0+ Core. Revision 1.0 [Text] / STMicroelectronics. – 12 p. – Access mode: https://www.st.com/resource/en/product_training/STM32G0-System-ARM-Cortex-M0plus-Core.pdf
15. Zen - Microarchitectures - AMD [Electronic resource]. – Access mode: <https://en.wikichip.org/wiki/amd/microarchitectures/zen>
16. Zen 2 - Microarchitectures - AMD [Electronic resource]. – Access mode: https://en.wikichip.org/wiki/amd/microarchitectures/zen_2
17. AMD's Zen 4 Part 1: Frontend and Execution Engine [Electronic resource]. – Access mode: <https://chipsandcheese.com/2022/11/05/amds-zen-4-part-1-frontend-and-execution-engine/>
18. Skylake (client) - Microarchitectures - Intel [Electronic resource]. – Access mode: [https://en.wikichip.org/wiki/intel/microarchitectures/skylake_\(client\)](https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(client))
19. Бутко В.О., Грушко С.С. Оптимальний синтез простих комбінаційних схем на основі описів Verilog HDL для FPGA. Комп'ютерні інтелектуальні системи та мережі : матеріали XVI Всеукраїнської науково практичної WEB конференції аспірантів, студентів та молодих вчених, м. Кривий Ріг, 21-23 берез. 2023 р. Кривий

Ріг : Криворізький національний університет, 2023. С. 40–43. – Режим доступу: https://sites.google.com/view/kicm#h.p_2GJDU4EZFcVZ

20. Білинський Й. Й. Цифрова схемотехніка. Електронно-обчислювальні пристрої [Текст] / Й. Й. Білинський, Б. П. Книш; ВНТУ, Вінниця, 2021. – 66 с. – Режим доступу: http://pdf.lib.vntu.edu.ua/books/2021/Bilinskii_2021_66.pdf

21. Bailey S. Comparison of VHDL, Verilog and SystemVerilog [Text] / S. Bailey; Model Technology, 2003. – 7 p. – Access mode: http://xilinx.pe.kr/_hdl/2/_ref/verilog_vhdl_sys.pdf

22. Verilog vs VHDL: Explain by Examples [Electronic resource]. – Access mode: <https://www.fpga4student.com/2017/08/verilog-vs-vhdl-explain-by-example.html>

23. AMD Vivado Design Suite [Electronic resource]. – Access mode: <https://www.xilinx.com/products/design-tools/vivado.html>

24. Офіційний веб-сайт фірми Xilinx/AMD [Електронний ресурс]. – Режим доступу: <http://www.xilinx.com/>

25. Cortex-A8 - Microarchitectures – ARM [Electronic resource]. – Access mode: https://en.wikichip.org/wiki/arm_holdings/microarchitectures/cortex-a8

26. Elbrus 2000 [Electronic resource]. – Access mode: https://en.wikipedia.org/wiki/Elbrus_2000

27. Release Note - ARM Cortex-M0+ Processor r0p1-30rel0 [Text] / ARM, 2012. – 14 p. – Access mode: <https://picture.iczhiku.com/resource/eetop/sYkSlpUflYYqzNcv.pdf>

28. AM335x Sitara Processors [Text] / Texas Instruments, 2020. – 258 p. – Access mode: https://www.ti.com/lit/ds/symlink/am3354.pdf?ts=1717677879155&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FAM3354

29. LPC81xM [Text] / NXP, 2021. – 79 p. – Access mode: <https://www.nxp.com/docs/en/data-sheet/LPC81XM.pdf>

30. Елементарні цифрові автомати [Електронний ресурс] / Запорізький національний університет. – 10 с. – Режим доступу: https://moodle.znu.edu.ua/pluginfile.php/1134360/mod_resource/content/1/%D0%9B%D0%B5%D0%BA%D1%86%D1%96%D1%8F%207.pdf

31. Palnitkar S. Verilog HDL: A Guide to Digital Design and Synthesis [Text] / S. Palnitkar; Prentice Hall PTR, 2003. – 2nd ed. – 442 p. – Access mode: https://dl.amobbs.com/bbs_upload782111/files_33/ourdev_585395BQ8J9A.pdf

32. Офіційна веб-сторінка опису характеристик FPGA сімейства Kintex 7 [Електронний ресурс]. – Режим доступу: <https://www.amd.com/en/products/adaptive-socs-and-fpgas/fpga/kintex-7.html>

33. About the UltraFast Design Methodology [Electronic resource]. – Access mode: <https://docs.amd.com/r/en-US/ug949-vivado-design-methodology/About-the-UltraFast-Design-Methodology>

34. Running Post-Synthesis and Post-Implementation Simulations [Electronic resource]. – Access mode: <https://docs.amd.com/r/2021.2-English/ug900-vivado-logic-simulation/Running-Post-Synthesis-and-Post-Implementation-Simulations>

35. Vivado Design Suite User Guide: Synthesis (UG901) [Text] / Xilinx/AMD, 2020. – 295 p. – Access mode: <https://docs.amd.com/v/u/2019.2-English/ug901-vivado-synthesis>

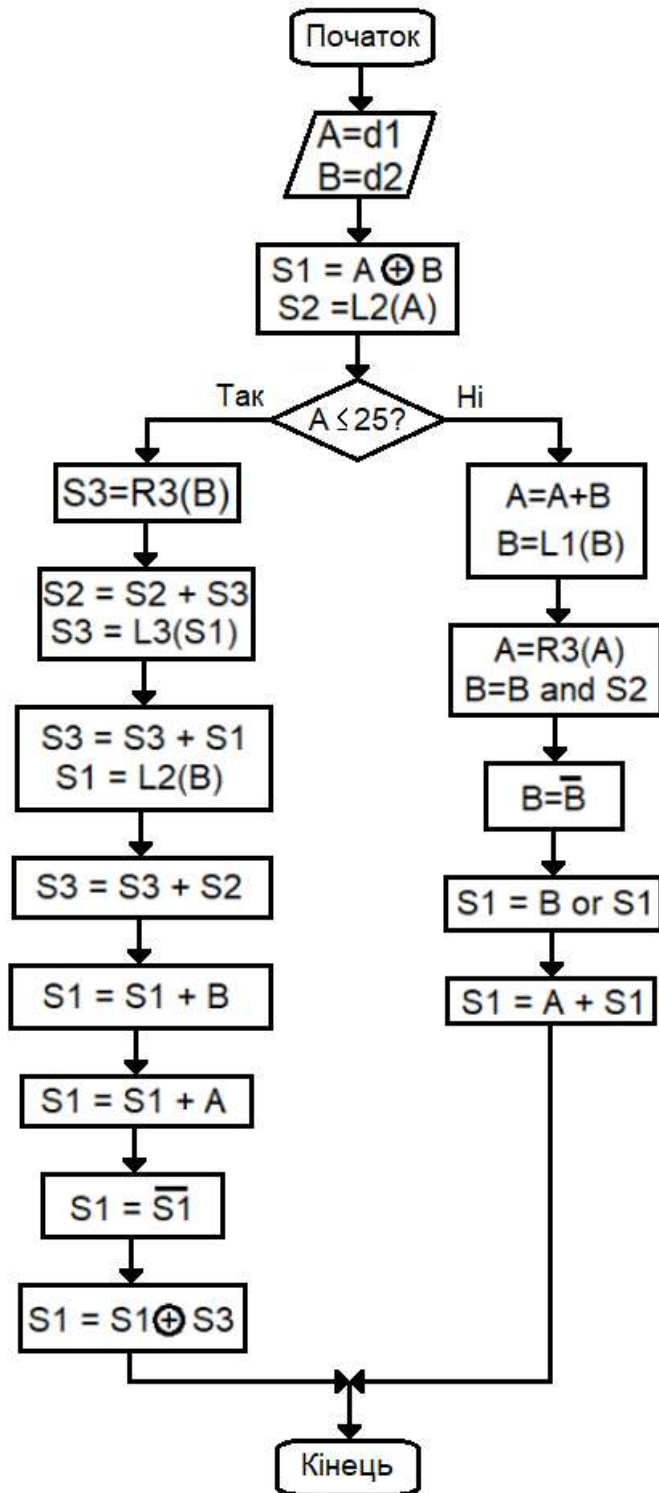
36. Unconstrained Paths Section [Electronic resource]. – Access mode: <https://docs.amd.com/r/en-US/ug906-vivado-design-analysis/Unconstrained-Paths-Section>

37. Gaschet C. Understanding Setup and Hold Times OneKey to Successful Designs [Text] / C. Gaschet; A Tale of Two Giants. – Pp. 39-42. – Access mode: <http://bitsavers.informatik.uni-stuttgart.de/components/xilinx/xcell/Xcell35.pdf>

38. Junction Temperature [Electronic resource]. – Access mode: <https://docs.amd.com/r/en-US/ug907-vivado-power-analysis-optimization/Junction-Temperature>

39. Thermal Simulation [Electronic resource]. – Access mode: <https://docs.amd.com/r/en-US/xapp1377-heatsinks-thermal/Thermal-Simulation>

40. Running Report Power with Vivado Simulator SAIF Data [Electronic resource]. – Access mode: <https://docs.amd.com/r/en-US/ug997-vivado-power-analysis-optimization-tutorial/Step-3-Running-Report-Power-with-Vivado-Simulator-SAIF-Data>



					13.02070849.00035 ПЛ1			
					Розробка спецобчислювача на FPGA з комбінованою операційною частиною			
					Функціональна граф-схема алгоритму спецобчислювача			
Зам.	Лист.	№ докум.	Підпис	Дата				
Розроб.		Бутко В.О.						
Перевір.		Зеленьова І.Я.						
Г.контр.								
Н.контр.		Польська О.В.						
Затвердж.		Кудерметов Р.К.						
					Лист. 1			
					Маса			
					Масшт.			
					Листів 1			
					НУ "Запорізька політехніка"			
					KHT-520			

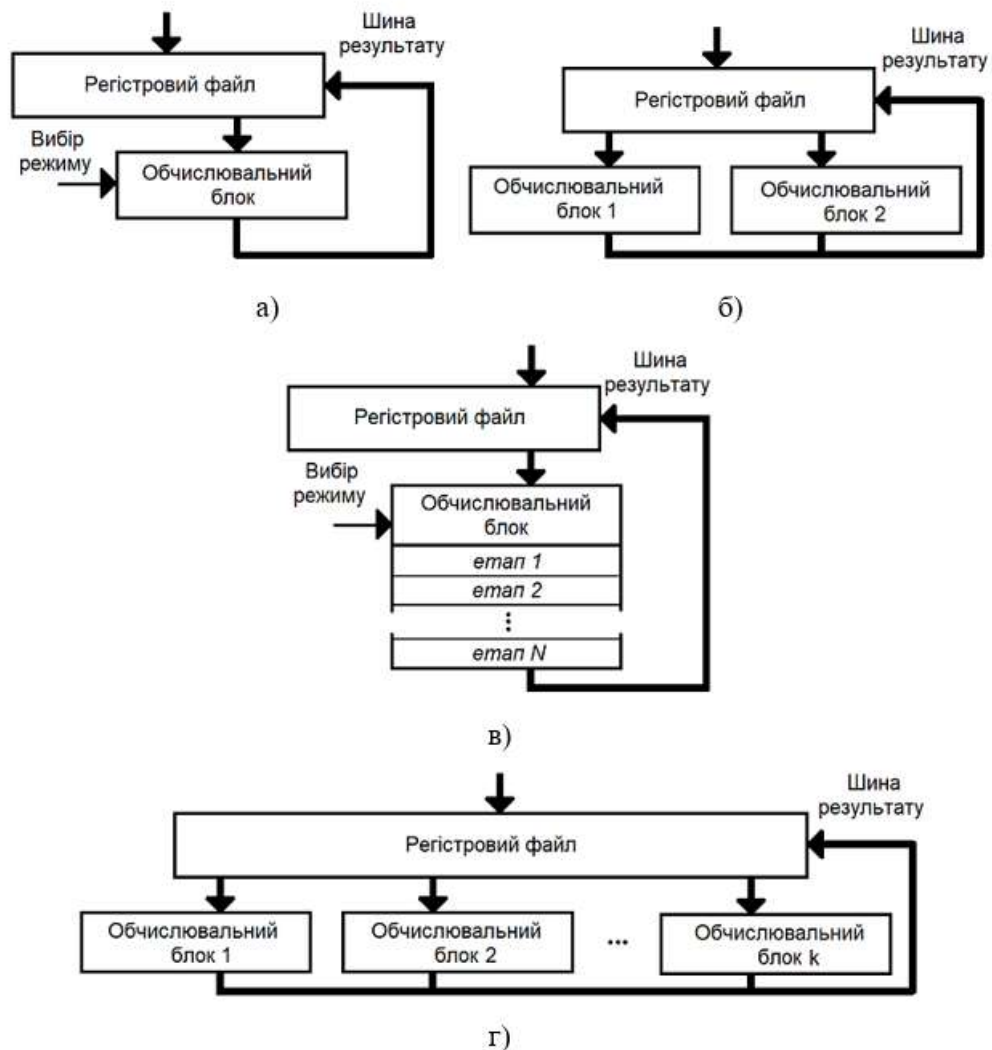
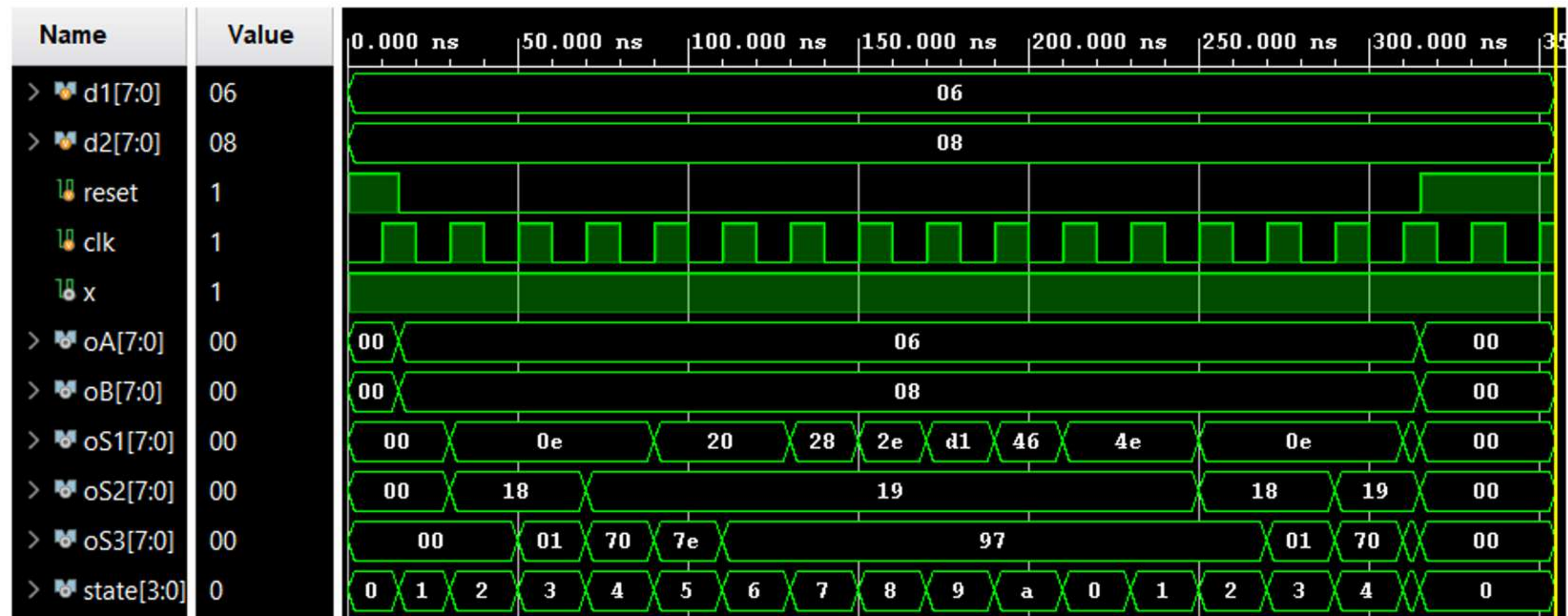
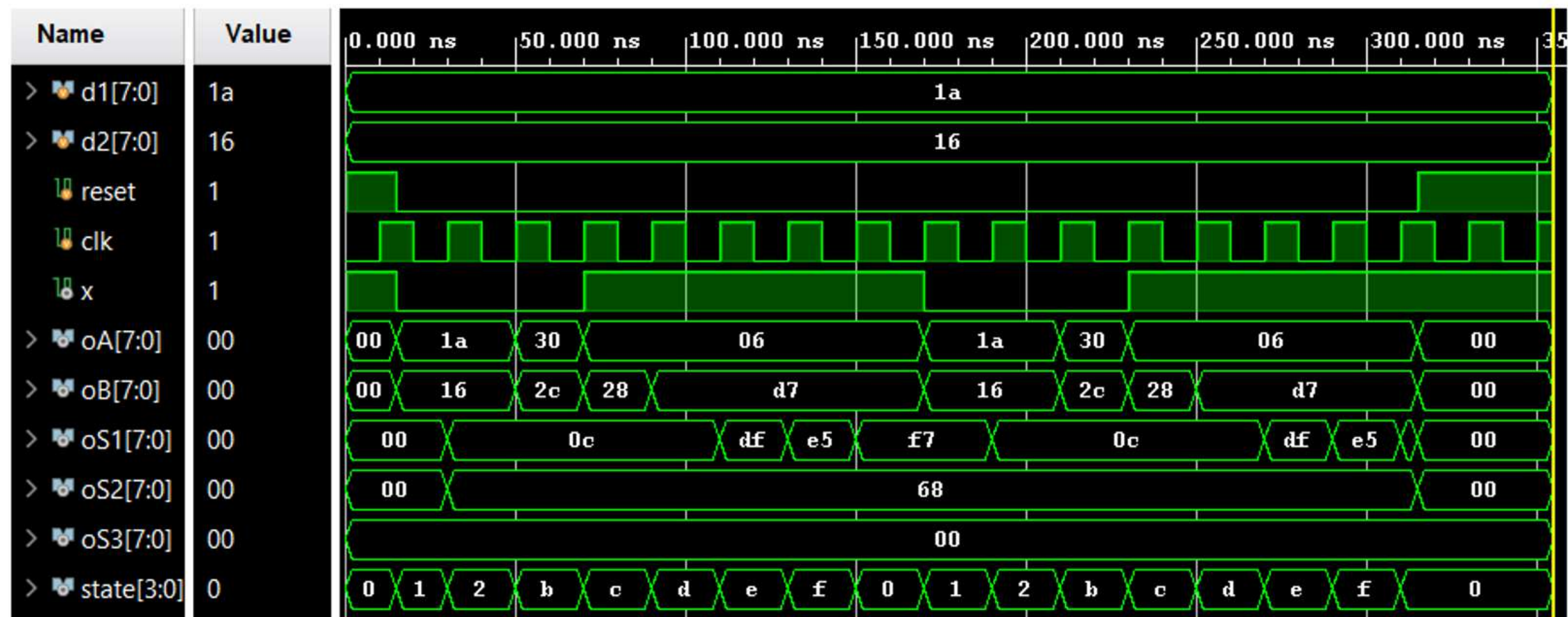


Рисунок 1 – Схематичне зображення типів ОА згідно з базовою класифікацією: а) М-автомат; б) ІМр-автомат; в) ІМs-автомат; г) І-автомат

					13.02070849.00035 ПЛЗ									
					Розробка спецобчислювача на FPGA з комбінованою операційною частиною									
					Схематичне зображення типів операційних автоматів згідно з базовою класифікацією									
Зам.	Лист.	№ докум.	Підпис	Дата	Лист.					Маса		Масшт.		
Розроб.		Бутко В.О.												
Перевір.		Зеленьова І.Я.												
Т.контр.					Лист 1					Листів 1				
Н.контр.		Польська О.В.			НУ "Запорізька політехніка" КНТ-520									
Затвердж.		Кудерметов Р.К.												



					13.02070849.00035 ПЛ4			
Зам.	Лист.	№ докум.	Підпис	Дата	Розробка спецобчислювача на FPGA з комбінованою операційною частиною Результат тестування спецобчислювача (ліва гілка алгоритму)	Лист.	Маса	Масшт.
Розроб.	Бутко В.О.							
Перевір.	Зеленьова І.Я.							
Т.контр.								
Н.контр.	Польська О.В.							
Затвердж.	Кудерметов Р.Р.							
						Лист 1		
						Листів 1		
						НУ "Запорізька політехніка"		
						КНТ-520		



					13.02070849.00035 ПЛ5		
Зам.	Лист.	№ докум.	Підпис	Дата	Розробка спецобчислювача на FPGA з комбінованою операційною частиною Результати тестування спецобчислювача (права гілка алгоритму)		
Розроб.	Бутко В.О.						
Перевір.	Зеленьова І.Я.						
Т.контр.							
Н.контр.	Польська О.В.				Лист 1 Листів 1 НУ "Запорізька політехніка" КНТ-520		
Затвердж.	Кудерметов Р.К.						