

Центральноукраїнський національний технічний університет
Факультет будівництва, транспорту та енергетики
Кафедра «Автоматизації виробничих процесів»

«Допущено до захисту»
Зав. кафедрою АВП
к.т.н., доцент
Олександр Дідик
«___» _____ 20__ р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему:
«МІКРОПРОГРАМНИЙ ПРИСТРІЙ КЕРУВАННЯ НА ОСНОВІ
ІНТЕГРАЛЬНИХ СХЕМ»

Виконав здобувач вищої освіти
4 курсу, групи АК-19
ОПП «Автоматизація та
комп'ютерно-інтегровані технології»
спеціальності 151 «Автоматизація та
комп'ютерно-інтегровані технології»
_____ Копил О.Ю.
«___» _____ 20__ р.

Керівник роботи
доцент, к.т.н.
_____ Олена Голик
«___» _____ 20__ р.

Рецензент
доцент, к.т.н. кафедри КБПЗ
_____ Олег Кислун
«___» _____ 20__ р.

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет будівництва, транспорту та енергетики

Кафедра автоматизації виробничих процесів

Рівень вищої освіти перший (бакалаврський) рівень

Галузь знань 15 «Автоматизація та приладобудування»

Спеціальність 151 «Автоматизація та комп'ютерно-інтегровані технології»

Освітньо-професійна програма «Автоматизація та комп'ютерно-інтегровані технології»

ЗАТВЕРДЖУЮ

Завідувач кафедри Дідик О.К.

“___” _____ 20__ року

**ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТА
ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ
ЗДОБУВАЧА ВИЩОЇ ОСВІТИ**

Копила Олексія Юрійовича

(прізвище, ім'я, по батькові)

1. Тема роботи «Мікропрограмний пристрій керування на основі інтегральних схем»

2. Керівник роботи

Голик Олена Петрівна, к.т.н., доцент

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

3. Строк подання роботи до захисту 25.06.2023 року

4. Мета та завдання випускної кваліфікаційної роботи

Зменшення апаратних витрат для реалізації схем пристроїв мікропрограмного керування побудованих на ПЛІС з архітектурою FPGA. Завдання: аналіз сучасних ПЛІС типу "система на кристалі" за технологією FPGA; Розробити та побудова структури пристроїв мікропрограмного керування з ідентифікацією виходів та оптимальною адресацією мікрокоманд.

5. Перелік графічного матеріалу структури мікропрограмних автоматів; структури пристроїв мікропрограмного керування; граф-схеми алгоритмів та функціональні схеми пристроїв мікропрограмного керування.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Огляд програмованих логічних інтегральних схем. Класифікація. Переваги та недоліки.	12.04.2023	
2	Аналіз існуючих пристроїв мікропрограмного керування.	28.04.2023	
3	Побудова граф-схем алгоритмів керування.	12.05.2023	
4	Синтез та побудова пристроїв мікропрограмного керування.	26.05.2023	
5	Розробка структурних та функціональних схем пристроїв мікропрограмного керування.	10.06.2023	
6	Формування пояснювальної записки бакалаврської роботи	16.06.2023	
7	Оформлення графічної частини бакалаврської роботи	21.06.2023	

Дата видачі завдання

«___» _____ 20__р.

Підпис керівника

(прізвище та ініціали)

Завдання прийнято до виконання

«___» _____ 20__р.

Підпис здобувача

(прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська робота присвячена зниженню апаратних витрат у схемах пристроїв мікропрограмного керування на основі програмованих логічних інтегральних схем з архітектурою FPGA.

Виконано огляд програмованих логічних інтегральних схем та їх класифікацію. Визначено переваги та недоліки існуючих мікросхем.

Розглянуто підходи до організації та реалізації пристроїв мікропрограмного керування, методи оптимізації логічних схем для пристроїв мікропрограмного керування. Окреслено основні завдання.

Проведено аналіз та побудову пристроїв мікропрограмного керування за допомогою різних методів. Побудовано граф-схеми алгоритмів, структурні та функціональні схеми.

Сформовано таблиці кодування, таблиці переходів та таблиці вмісту керуючої пам'яті.

В результаті вдалось досягти поставленої мети та зменшити апаратні витрати для реалізації схем пристроїв мікропрограмного керування побудованих на основі програмованих логічних інтегральних схем.

Також в бакалаврській роботі проведено теоретичний аналіз доцільності проведеної оптимізації.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ.....	3
ВСТУП.....	4
РОЗДІЛ 1. ЗАГАЛЬНІ ПРИНЦИПИ МІКРОПРОГРАМНОГО КЕРУВАННЯ В АВТОМАТИЗАЦІЇ.....	6
1.1 Огляд програмованих логічних інтегральних схем.....	6
1.2 Класифікація програмованих логічних інтегральних схем.....	13
1.3 Переваги недоліки існуючих мікросхем.....	15
1.4 Висновки.....	17
РОЗДІЛ 2 АНАЛІЗ ІСНУЮЧИХ ПРИСТРОЇВ МІКРОПРОГРАМНОГО КЕРУВАННЯ.....	18
2.1 Підходи до організації та реалізації пристроїв мікропрограмного керування.....	18
2.2 Застосування в цифровій техніці мікросхем типу «система на кристалі».....	25
2.3 Методи оптимізації логічних схем пристроїв мікропрограмних керування.....	28
2.4 Основні завдання.....	29
2.5 Висновки.....	30
РОЗДІЛ 3 АНАЛІЗ ТА ПОБУДОВА ПРИСТРОЇВ МІКРОПРОГРАМНОГО КЕРУВАННЯ.....	31
3.1 Аналіз та побудова пристрою мікропрограмного керування з використанням методу ідентифікації виходів.....	31
3.2 Аналіз та побудова пристрою мікропрограмного керування з адресацією мікрокоманд.....	45
3.3 Висновки.....	54

РОЗДІЛ 4 РОЗРОБКА СТРУКТУРНИХ ТА ФУНКЦІОНАЛЬНИХ СХЕМ ПРИСТРОЇВ МІКРОПРОГРАМНОГО КЕРУВАННЯ.....	55
4.1 Структурна та функціональна схема пристрою мікропрограмного керування з ідентифікацією виходів.....	55
4.1.1. Формування таблиць вмісту керуючої пам'яті та переходів. Структурна схема.....	55
4.1.2. Етапи синтезу пристрою мікропрограмного керування. Функціональна схема.....	58
4.2 Структурна та функціональна схема пристрою мікропрограмного керування з адресацією мікрокоманд.....	63
4.2.1. Структура пристрою мікропрограмного керування з поділом кодів.....	63
4.2.2. Етапи синтезу пристрою мікропрограмного керування. Функціональна схема.....	65
4.3 Теоретичний аналіз доцільності проведеної оптимізації.....	71
4.3.1 Загальна методика розрахунку.....	71
4.3.2 Пристрій мікропрограмного керування з ідентифікацією виходів.....	73
4.3.3 Пристрій мікропрограмного керування з адресацією мікрокоманд.....	78
4.4 Висновки.....	81
ВИСНОВКИ.....	82
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	83

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

<i>АЧ</i>	адресна частина мікрокоманди
<i>ВБП</i>	вбудований блок пам'яті
<i>ЛЕ</i>	логічний елемент табличного типу
<i>ПЛБ</i>	програмований логічний блок
<i>SoC</i>	мікросхема типу «система на кристалі» system on a chip
<i>ГСА</i>	граф-схема алгоритму
<i>КП</i>	керуюча пам'ять
<i>ЛТ</i>	лічильник адреси мікрокоманд
<i>МК</i>	мікрокоманда
<i>МО</i>	мікрооперація
<i>ПМК</i>	пристрій мікропрограмного керування
<i>ОА</i>	операційний автомат
<i>ОЛЛ</i>	операторний лінійний ланцюг
<i>ОЧ</i>	операційна частина мікрокоманди
<i>ПАМ</i>	схема перетворювача адреси мікрокоманд
<i>ПК</i>	пристрій керування
<i>ПКВМ</i>	програмована користувачем вентильна матриця
<i>ПЛІС</i>	програмована логічна інтегральна схема
<i>ПСТ</i>	пряма структурна таблиця
<i>РАМК</i>	регістр адреси мікрокоманд
<i>РК</i>	регістр коду ОЛЛ
<i>РС</i>	регістр стану
<i>САМК</i>	схема адресації мікрокоманд
<i>СКЛУ</i>	схема кодування логічних умов
<i>СФМО</i>	схема формування мікрооперацій
<i>ТЧ</i>	тригер читання

ВСТУП

Актуальність теми. До складу будь якої цифрової системи входить пристрій керування (ПК), який може бути реалізований як пристрій мікропрограмного керування (ПМК) до складу якого входять автомати з «програмованою» та «жорсткою» логікою. Це дає можливість досягти мінімально можливої ємності керуючої пам'яті (КП) та забезпечити максимальну швидкодію ПК.

ПМК дає змогу використовувати вбудовані блоки пам'яті (ВБП) для реалізації системи мікрооперацій. Необхідно враховувати, що при цьому схему адресації мікрокоманд виконують на логічних елементах, а саме LUT-елементах (look up table) з яких побудована мікросхема FPGA. Як правило, така мікросхема має кількість входів, які не перевищують 6 входів.

Обмежена кількість числа входів логічних елементів призводить до необхідності декомпозиції системи (розклад великої загальної системи на маленькі підсистеми) булевих функцій за допомогою яких задають закон функціонування ПК. А це в свою чергу призводить до зменшення швидкодії ПМК, оскільки зростає кількість рівнів схеми. Таким чином, виникає необхідність у зменшенні кількості числа логічних LUT-елементів та їх рівнів у схемах ПМК.

Мета і завдання дослідження. Метою бакалаврської роботи є зменшення апаратних витрат для реалізації схем ПМК побудованих на ПЛІС з архітектурою FPGA.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

1. Проаналізувати сучасні ПЛІС типу "система на кристалі" за технологією FPGA.
2. Розробити та побудувати структуру ПМК з ідентифікацією виходів та оптимальною адресацією мікрокоманд.

Об'єктом дослідження є пристрої мікропрограмного керування.

Предметом дослідження є методи та способи синтезу і побудови для оптимізації апаратних витрат у схемах пристроїв мікропрограмного керування на програмованих логічних інтегральних схемах з архітектурою FPGA.

Методи досліджень: теорія кінцевих автоматів, теорія множин, булева алгебра та логіка, карти Карно, теорія графів, теорія імовірності, комп'ютерні програми для моделювання.

Практичне значення одержаних результатів полягає в розробці удосконалених структур ПМК на ПЛІС, які дають можливість зменшити апаратні витрати на реалізацію схем ПМК.

Апробація бакалаврської роботи. «День Науки-2023» (20 квітня 2023 року, ЦНТУ, м. Кропивницький). Доповідь на тему: «Мікропрограмний пристрій керування з ідентифікацією виходів».

Структура та обсяг бакалаврської роботи. Вступу, чотири розділа, загальні висновки, список використаних джерел містить 32 джерела. Загальний обсяг пояснювальної записки становить 86 сторінок.

РОЗДІЛ 1. ЗАГАЛЬНІ ПРИНЦИПИ МІКРОПРОГРАМНОГО КЕРУВАННЯ В АВТОМАТИЗАЦІЇ

1.1. Огляд програмованих логічних інтегральних схем

Мікросхеми FPGA (field-programmable gate array) відіграють важливу роль у розвитку різних технологічних досягнень і є одними з найважливіших продуктів. Існують різні типи чіпів FPGA, і деякі з них є кращими перед іншими у певних програмах та сценаріях [32].

FPGA можна представити, як апаратну мікросхема, яка використовується для виконання логічних операцій. Вони складаються з інтегрованої мережі або наборів логічних блоків, розміщених на чипі, де схеми є програмованими логічними елементами [5, 9, 13-15].

ПЛІС складаються з окремих конфігурованих логічних блоків, які з'єднані через програмовані з'єднання. Переваги FPGA відомі завдяки їх здатності програмувати в умовах реального часу на відміну від інших типів напівпровідникових чіпів (наприклад, ASIC), які є переважно жорсткими за своєю конструкцією та виконанням [5, 7, 9, 13-15, 21].

Ці пристрої мають кілька принципових подібностей. Насправді вони складаються з однакових основних компонентів. FPGA та мікроконтролер є інтегральними схемами, які працюють за допомогою комп'ютерної логіки та двійкового коду [13-15, 21, 26, 32].

Інтегральна схема — це набір електронних схем на одному плоскому кристалі, який складається з напівпровідникового матеріалу — найчастіше кремнію [5, 9, 13-15]. Він може виконувати обчислення та зберігати дані. Інтегральні схеми використовують або цифрову, або аналогову технологію. Цифрова технологія використовує двійкову логіку, тоді як аналогова технологія визначає лінійну функцію входу для обчислення виходу [5, 7, 9].

Архітектура FPGA.

Кожен постачальник FPGA має власну архітектуру, однак базова загальна структура показана на рис. 1.1.

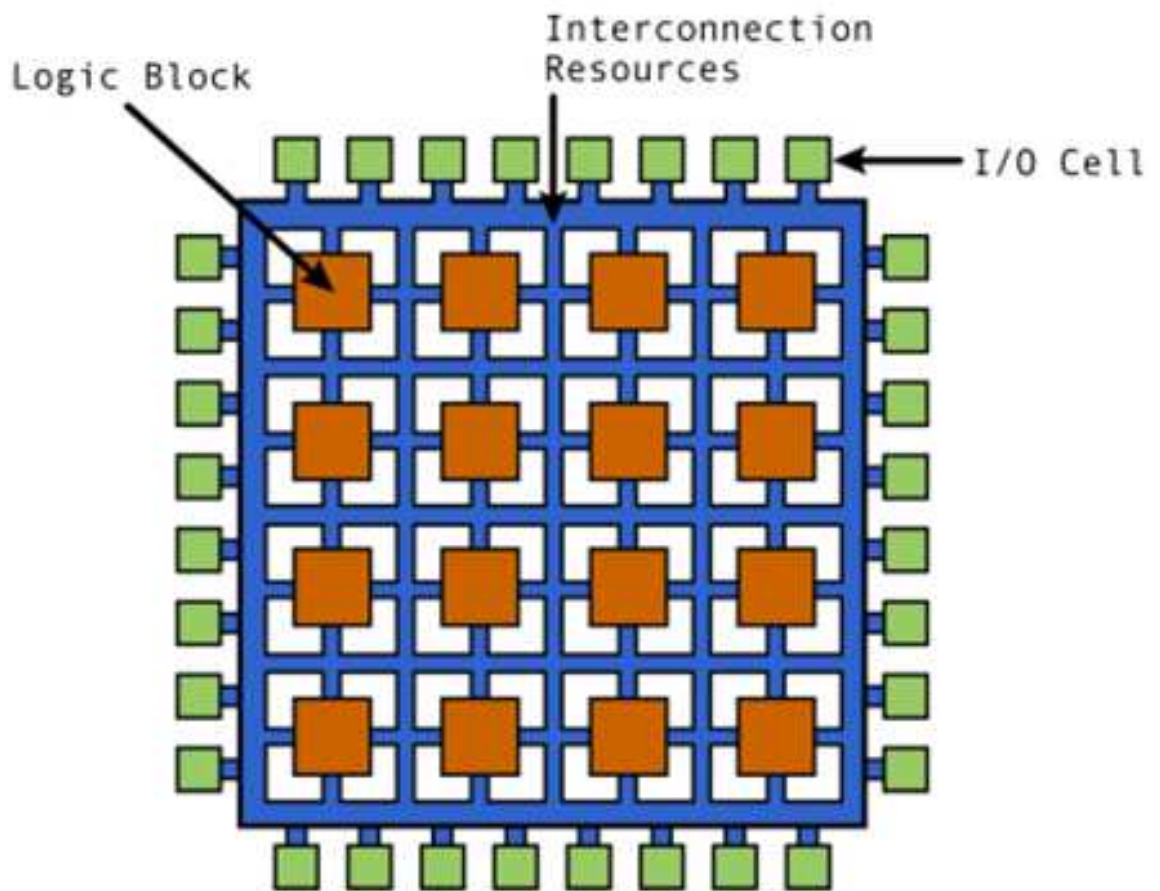


Рисунок 1.1. - Загальна структура FPGA

Архітектура складається з конфігурованих логічних блоків, конфігурованих блоків вводу/виводу та програмованого з'єднання. Крім того, до складу входить схема синхронізації для передачі тактових сигналів до кожного логічного блоку. Також можуть бути доступні додаткові логічні ресурси, такі як ALU, пам'ять і декодери [13-15, 21, 26, 32].

Налагоджуючі логічні блоки (CLB)

Вони будують логіку для FPGA. CLB містять достатньо логіки для створення невеликого кінцевого автомата, як показано на рис. 1.2.

Блок містить RAM для створення довільних комбінаторних логічних функцій, також відомих як таблиці пошуку (LUT). Він також містить тригери

для тактових елементів зберігання разом із мультиплексорами для маршрутизації логіки всередині блоку та до зовнішніх ресурсів [21, 26, 32].

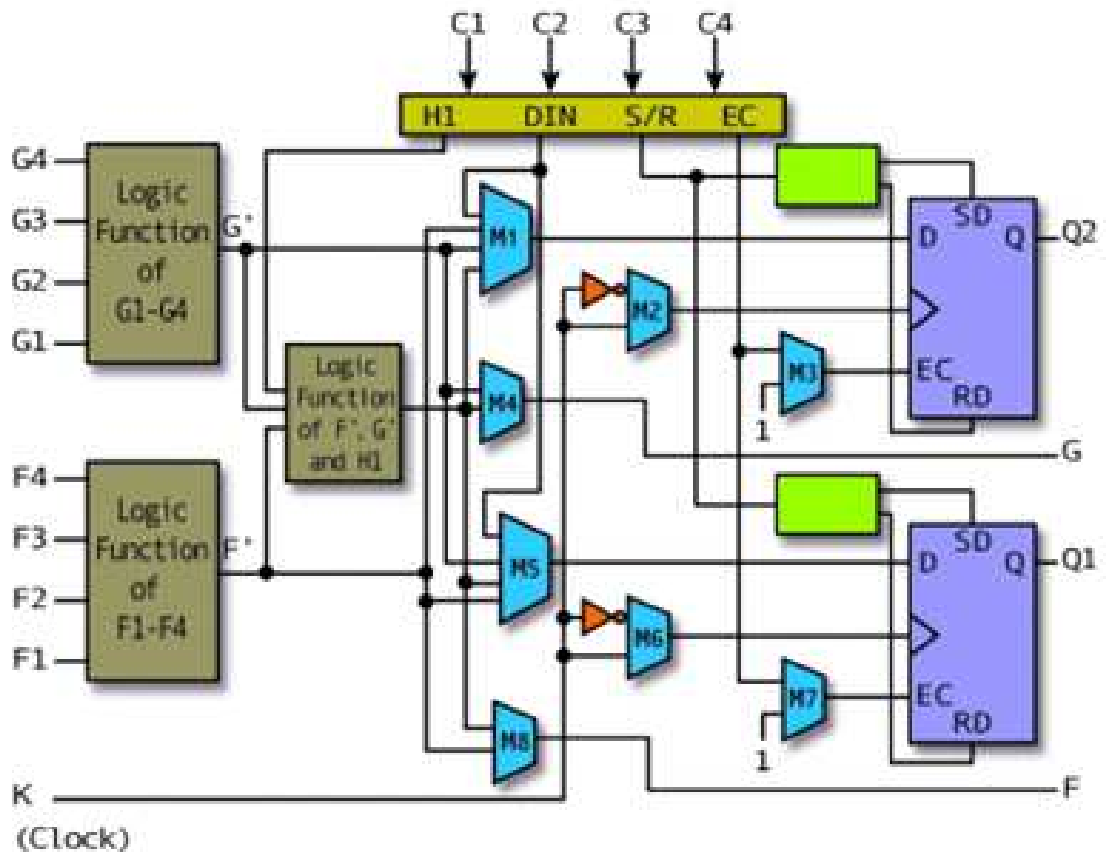


Рисунок 1.2. - Налаштовуваний логічний блок FPGA (CLB)

Конфігуровані блоки введення/виведення

Конфігурований блок вводу/виводу (I/O) показано на рис. 1.3.

Використовується для передачі сигналів на мікросхему та їх повторного відправлення. Складається з вхідного та вихідного буфера з трьома станами та керуванням виходом з відкритим колектором [26, 30, 32].

Як правило, на виходах є підтягуючі резистори, які можна використовувати для завершення сигналів і шин, не вимагаючи дискретних резисторів [13-15, 21, 26, 32]. Зазвичай на виходах є тригери, щоб тактовані сигнали могли виводитися безпосередньо на контакти без значної затримки, легше задовольняючи вимоги щодо часу налаштування для зовнішніх

пристроїв. Подібним чином тригери на входах зменшують затримку сигналу перед досягненням тригера, таким чином зменшуючи вимоги до часу утримання ПЛІС.

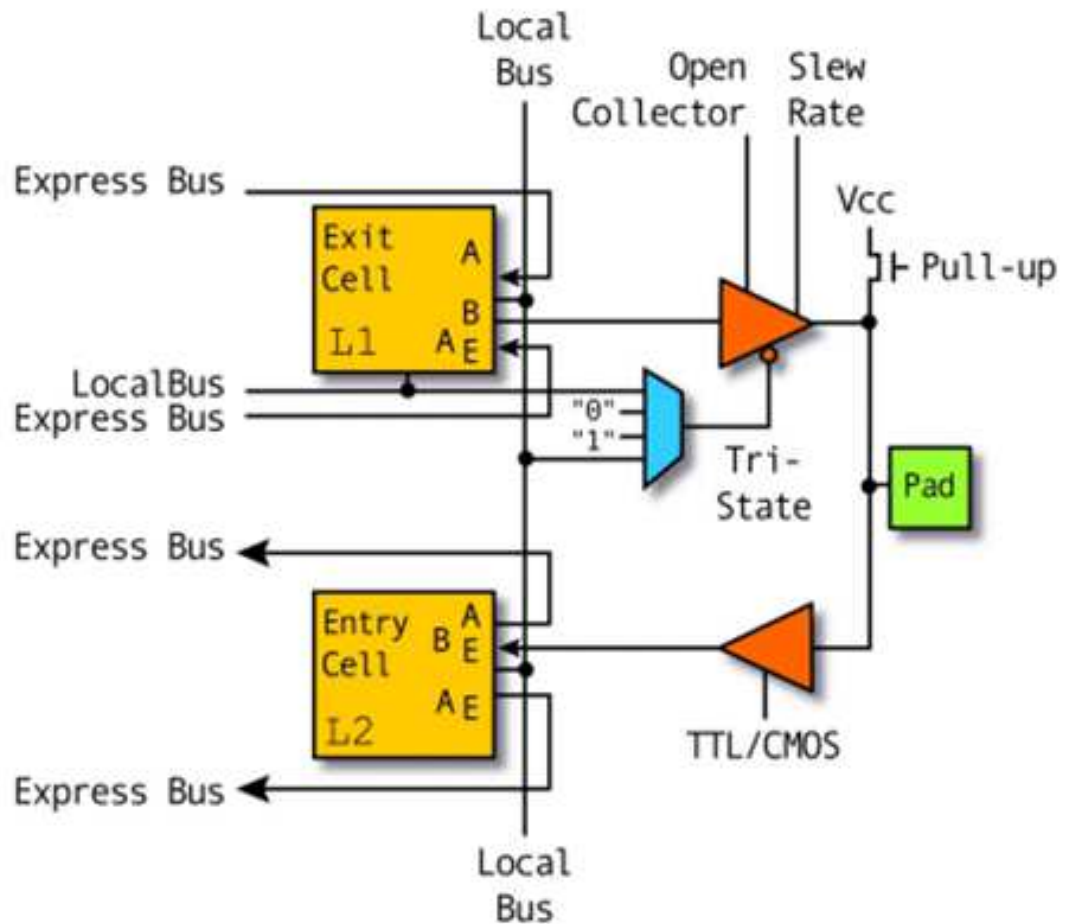


Рисунок 1.3. - Налагоджувачий блок вводу-виводу FPGA

Ядра

Існує два основних типи ядер. Програмне ядро, відоме як ІР-ядро, є функцією, яка описується логічною функцією, а не будь-якою фізичною реалізацією. Програмні ядра зазвичай складаються з коду мови опису обладнання (HDL) [13-15].

Жорсткі ядра, з іншого боку, складаються з фізичних реалізацій функції. Що стосується ПЛІС, ці жорсткі ядра відомі як вбудовані ядра, оскільки вони фізично вбудовані в кристал чіпа й оточені програмованою логікою [21, 32].

Щоб розмістити ці складні функції в програмованому пристрої, є три варіанти [13-15, 21, 26, 32]:

- перший полягає в тому, щоб розробити функцію самостійно та розмістити її в програмованій логіці; придбати код HDL для функції та включити його у ваш код HDL; змусити постачальника включити функцію як комірку, вбудовану в програмований пристрій.

- другий варіант — IP-ядро або програмне ядро.

- третій варіант — вбудоване ядро або жорстке ядро.

Ядра IP

IP-ядра скорочують потреби в часі та робочій силі для розробника FPGA. Крім того, IP-ядра часто можна змінювати, тобто ви можете додавати або віднімати функціональні можливості відповідно до своїх потреб. Електричні характеристики, такі як синхронізація або енергоспоживання для ядер IP, можна оптимізувати в обмеженому ступені, але фактичні характеристики залежать від його використання в конкретному пристрої, а також залежать від логіки, до якої він підключений [13, 26, 32].

Вбудовані ядра

Деякі вбудовані ядра є аналоговими пристроями, які не можуть бути розроблені у звичайній FPGA. Інтегрувавши ці функції в пристрій, можна уникнути складного процесу проектування аналогових пристроїв, а також зберегти чіпи та компоненти, які в іншому випадку були б потрібні поза програмованим пристроєм [15, 21, 32].

Ядра процесора

Ядра процесора є одним із типів ядер, які зазвичай доступні як IP-ядра або вбудовані ядра. Ці процесори, як правило, призначені для вбудованих систем, оскільки, майже за визначенням, програмовані пристрої є вбудованими системами [13-15].

Для будь-якого типу ядра будуть доступні інструменти для розробки програмного забезпечення. Готові крос-компілятори та симулятори можна

використовувати для налагодження коду до того, як буде завершено проектування та доступний програмований пристрій [26, 32].

Ядра DSP

Цифрові сигнальні процесори (DSP) є ще одним поширеним типом ядра, яке пропонується як IP-ядро або вбудоване ядро. По суті, це спеціалізовані процесори, які використовуються для обробки аналогових сигналів. Вони зазвичай використовуються для фільтрації та стиснення відео чи аудіосигналів [13-15, 21, 26, 32].

Однак вимоги до оцифровки, стиснення та фільтрації відео та аудіо зросли в останні роки, оскільки мільйони користувачів підключаються до Інтернету та регулярно завантажують і завантажують усі види інформації через з'єднання з відносно обмеженою пропускною здатністю. Поки що попит на DSP для використання в мережевих і графічних пристроях зростає, а не зменшується [32].

Спеціальні драйвери введення/виведення

Спеціальні драйвери вводу-виводу також вбудовуються в програмовані пристрої. Новіші шини в персональних комп'ютерах повинні мати дуже жорстко контрольований час і керуватися спеціальними ланцюгами з високим опором. Буфери введення/виведення повинні мати входи з дуже конкретними пороговими значеннями напруги. Зараз багато постачальників пропонують програмовані пристрої з введенням/виведенням, які відповідають цим спеціальним вимогам [13-15, 21, 26, 32].

Нові архітектури

Розробляються нові базові архітектури для логічних блоків, які складають FPGA [32]. Одна нова архітектура має логічний блок, заснований на DSP, як показано на рис. 1.4.

Цей тип FPGA буде кращим для використання в мікросхемах, які потребують значної обробки сигналу. Але є певні сумніви щодо цього майбутнього шляху [21, 26, 32].

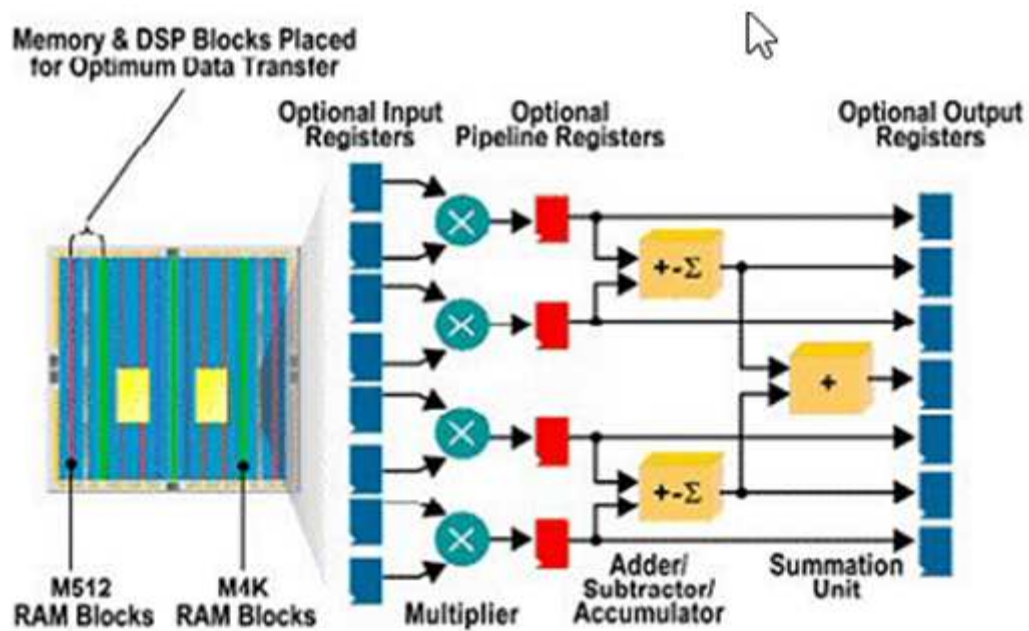


Рисунок 1.4. - Ядро DSP у FPGA

По-перше, більшість програмованих пристроїв не виконують DSP, тому ця архітектура націлена на відносно невеликий ринок.

По-друге, будуть потрібні спеціальні інструменти для перетворення алгоритмів цифрової сигналізації для використання в такій спеціалізованій FPGA. Ці інструменти повинні дуже добре оптимізувати алгоритм, щоб продуктивність у цій спеціалізованій FPGA справді була кращою, ніж стандартний DSP або загальний процесор, що виконує код, оптимізований за допомогою інструментів і компіляторів, які були доступні роками.

Найважливіша сфера майбутнього, полягає у створенні нових засобів розробки для FPGA. Оскільки програмовані пристрої стають все більшими, складнішими та включають один або більше процесорів, виникає величезна потреба в інструментах, щоб скористатися перевагами цих функцій і оптимізувати конструкції [21, 26, 32].

Оскільки ПЛІС включають процесори, засоби розробки потрібні для програмного забезпечення так само, як і для апаратного забезпечення. Інструменти апаратного синтезу дозволяють апаратним інженерам працювати

на вищих рівнях абстракції, не потребуючи розуміння деталей базової апаратної архітектури. Подібним чином необхідні інструменти синтезу програмного забезпечення, щоб дозволити інженерам програмного забезпечення працювати на вищому рівні абстракції без необхідності розуміння деталей базової архітектури програмного забезпечення [21, 26, 32].

Згодом платформи FPGA із вбудованими процесорами стануть домінуючою платформою для проектування вбудованих систем і, нарешті, дозволять реалізувати перспективи та примусити до подальшого розвитку проектування апаратного та програмного забезпечення [21, 26, 32].

1.2. Класифікація програмованих логічних інтегральних схем

Типи FPGA на основі програм

Програмовані вентильні матриці класифікуються на три типи на основі таких застосувань, як FPGA низького класу, FPGA середнього класу та FPGA високого класу [21, 26, 32].

Low End FPGA

Ці типи FPGA розроблені для низького енергоспоживання, низької логічної щільності та низької складності на мікросхемі. Прикладами низьких ПЛІС є сімейство Cyclone від Altera, сімейство Spartan від Xilinx, сімейство Fusion від Microsemi та Mach XO/ICE40 від Lattice semiconductor [21, 26, 32].

FPGA середнього класу

Ці типи FPGA є оптимальним рішенням між бюджетними та високоякісними FPGA, і вони розроблені як баланс між продуктивністю та ціною. Прикладами FPGA середнього класу є Arria від Altera, серії Artix-7/Kintex-7 від Xilinx, IGL002 від Microsemi та серії ECP3 і ECP5 від Lattice semiconductor [21, 26, 32].

High End FPGA

Ці типи FPGA розроблені для логічної щільності та високої продуктивності. Прикладами високоякісних FPGA є сімейство Stratix від Altera,

сімейство Virtex від Xilinx, сімейство Speedster 22i від Achronix і сімейство ProASIC3 від Microsemi [21, 26, 32].

Застосування FPGA [21, 26, 32]:

Багато програм покладаються на паралельне виконання ідентичних операцій; можливість конфігурувати CLB FPGA в сотні чи тисячі ідентичних блоків обробки має застосування в обробці зображень, штучному інтелекті (ШІ), апаратних прискорювачах центрів обробки даних, корпоративних мережах і передових автомобільних системах допомоги водієві (ADAS).

Багато з цих областей застосування змінюються дуже швидко, оскільки вимоги розвиваються та приймаються нові протоколи та стандарти. ПЛІС дозволяють виробникам впроваджувати системи, які можна оновлювати за потреби [21, 26, 32].

Гарним прикладом використання FPGA є високошвидкісний пошук: Microsoft використовує FPGA у своїх центрах обробки даних для запуску алгоритмів пошуку Bing. FPGA може змінюватися для підтримки нових алгоритмів у міру їх створення. Якщо потреби зміняться, конструкцію можна перепрофілювати для виконання процедур симуляції або моделювання в програмі HPC. Такої гнучкості важко або неможливо досягти за допомогою ASIC [21, 26, 32].

За останнє десятиліття FPGA набули швидкого зростання, оскільки вони корисні для широкого спектру застосувань. Конкретне застосування FPGA включає цифрову обробку сигналів, біоінформатику, контролери пристроїв, програмно визначене радіо, випадкову логіку, створення прототипів ASIC, медичну візуалізацію, емуляцію комп'ютерного обладнання, інтеграцію кількох SPLD, розпізнавання голосу, криптографію, фільтрацію та кодування зв'язку і багато іншого [21, 26, 32].

Зазвичай FPGA зберігаються для конкретних вертикальних застосувань, де обсяг виробництва невеликий. Для цих невеликих додатків провідні компанії сплачують вартість обладнання за одиницю. Сьогодні нова динаміка

продуктивності та вартість розширили діапазон життєздатних застосувань [21, 26, 32].

Деякі більш поширені програми FPGA: аерокосмічна промисловість і оборона, медична електроніка, прототипування ASIC, аудіо, автомобільна промисловість, телемовлення, споживча електроніка, розподілені грошові системи, центр обробки даних, високопродуктивне обчислення, промисловість, медицина, наукові прилади, системи безпеки, відео та зображення. Обробка, дротовий зв'язок, бездротовий зв'язок [21, 26, 32].

1.3. Переваги недоліки існуючих мікросхем

Три типи FPGA [21, 26, 32]:

- статична RAM (SRAM),
- антизапобіжники;
- флеш-пам'ять EPROM.

Програмування SRAM включає невеликий статичний біт RAM для кожного елемента програмування. Запис біта з нулем вимикає перемикач, тоді як запис з одиницею вмикає перемикач [21, 26, 32].

Інший метод включає антизапобіжник, який складається з мікроскопічної структури, яка, на відміну від звичайного запобіжника, зазвичай не створює зв'язку. Велика величина струму під час програмування пристрою викликає з'єднання двох сторін протизапобіжника [21, 26, 32].

Третій і відносно новий метод використовує флеш-біти EPROM для кожного елемента програмування [21, 26, 32].

Переваги ПЛІС на основі SRAM [21, 26, 32] – найпоширенішої технології програмування на сьогоднішній день – полягають у тому, що вони використовують стандартний процес виготовлення, який підприємства з виробництва мікросхем завжди оптимізують для кращої продуктивності. Оскільки SRAM можна перепрограмувати, FPGA можна перепрограмувати будь-яку кількість разів, навіть коли вони знаходяться в системі, як і запис у

звичайну SRAM. Пристрої SRAM можуть легко використовувати внутрішні SRAM як невеликі пам'яті в дизайні.

Недоліки FPGA на основі SRAM полягають у тому, що вони нестабільні, що означає, що збій живлення може потенційно пошкодити вміст пристрою. Теоретично пристрої SRAM мають великі затримки маршрутизації та працюють повільніше, ніж інші технології, але постійне вдосконалення технології SRAM ефективно усунуло цей недолік [21, 26, 32].

SRAMFPGA може споживати більше енергії та є менш безпечним, ніж інші технології, оскільки їх потрібно перепрограмувати після ввімкнення живлення, і можна спостерігати, як бітовий потік програмування надходить у пристрій. Спеціальні SRAM FPGA з вбудованими ключами, які розшифровують вхідні програмні потоки бітів, можна придбати у постачальників, але це зменшує низьку вартість і швидкий час виконання переваг FPGA. Бітові помилки також більш імовірні для SRAM FPGA, ніж для інших пристроїв [21, 26, 32].

Переваги антизапобіжних FPGA полягають у тому, що вони енергонезалежні, а затримки через маршрутизацію дуже малі, тому вони, як правило, швидше. Антизапобіжні FPGA, як правило, вимагають меншої потужності, і вони кращі для того, щоб зберегти інформацію про проект від рук конкурентів, оскільки їм не потрібен зовнішній пристрій для програмування їх після ввімкнення, як це роблять пристрої SRAM [26, 32].

Недоліки полягають у тому, що вони вимагають складного процесу виготовлення, вони вимагають зовнішнього програміста для їх програмування, і після того, як вони запрограмовані, їх неможливо змінити [21, 26, 32].

Складний, нестандартний процес виготовлення виявився ключовим недоліком, оскільки антизапобіжні FPGA мають нижчу ефективність, а технологія вдосконалюється повільніше, ніж SRAM FPGA.

FlashFPGA, здається, поєднує в собі найкраще з обох інших методів. Вони енергонезалежні, як антизапобіжні FPGA, але перепрограмовані, як SRAM FPGA. Вони використовують стандартний процес виготовлення, як SRAM

FPGA, і вони менш потужні та безпечні, як антизапобіжні FPGA. Вони також відносно швидкі. Наразі один постачальник підтримує флеш-ПЛІС, а інший постачальник має гібридну флеш-пам'ять/SRAM ПЛІС. Вони не набирають популярності так швидко, хоча це може змінитися в майбутньому [21, 26, 32].

1.4. Висновки

В даному розділі виконано огляд існуючих ПЛІС з архітектурою FPGA. Розглянуто класифікацію, будову та призначення елементів мікросхеми. Наведено переваги та недоліки різних типів мікросхем. В подальшому в бакалаврській роботі буде розглянуто ПМК на основі ПЛІС в архітектурі FPGA.

РОЗДІЛ 2 АНАЛІЗ ІСНУЮЧИХ ПРИСТРОЇВ МІКРОПРОГРАМНОГО КЕРУВАННЯ

2.1. Підходи до організації та реалізації пристроїв мікропрограмного керування

Одним із найважливіших складових цифрової системи є пристрій керування (ПК). Спосіб побудови ПК суттєво впливає на його функціональні можливості, продуктивність та рівень апаратних витрат операційного пристрою [1, 4, 5, 10, 13, 18, 20, 25]. Для організації, побудови та реалізації ПК відомі два основних загальних підходи [4, 5, 10, 13, 18, 20]:

- мікропрограмні автомати з «жорсткою» логікою;
- мікропрограмні автомати з «програмованою» логікою.

Мікропрограмні автомати з «жорсткою» логікою поділяють на два класи [1, 10, 13, 18]:

- автомати Мілі (автомат Мілі складається з тригерів, залежно від їх стану в певний момент часу визначають внутрішній стан автомата. Послідовне перемикання тригерів забезпечує формування сигналів вхідних сигналів в дискретні моменти часу. Таким чином здійснюється послідовна зміна станів автомата);
- автомати Мура.

Вихідний сигнал автомата Мура залежить лише від внутрішнього стану, а в автоматі Мілі вихідний сигнал залежить від вхідного сигналу та внутрішнього стану. Це пояснює головну різницю між ними.

Пристрій керування, побудований у вигляді автомата з «жорсткою» логікою наведений на рис. 2.1. Він складається з комбінаційної схеми (КС), яка формує функції F в залежності від логічних умов X та коду внутрішнього стану автомата. Внутрішній стан автомата представлений змінними T , які зберігаються в регістрі станів (РС), мікроопераціям Y відповідає поточній мікрокоманді алгоритму функціонування цифрової системи [1, 6, 10, 13, 18].

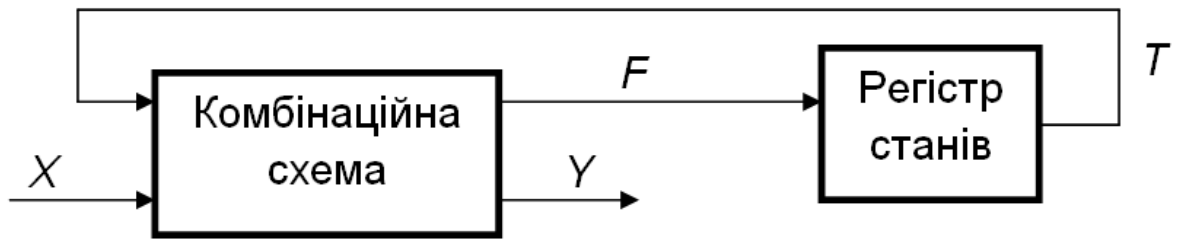


Рисунок 2.1. - Структура мікропрограмного автомата з «жорсткою» логікою

Автомати з «жорсткою» логікою дають змогу забезпечити максимальну швидкість з максимальними апаратними витратами. Відмінність у них в нерегулярності структури, яка суттєво ускладнює процес їх синтезу [10, 13].

На рис. 2.2 показаний пристрій керування, що побудований та реалізований у вигляді мікропрограмного автомата з «програмованою» логікою.

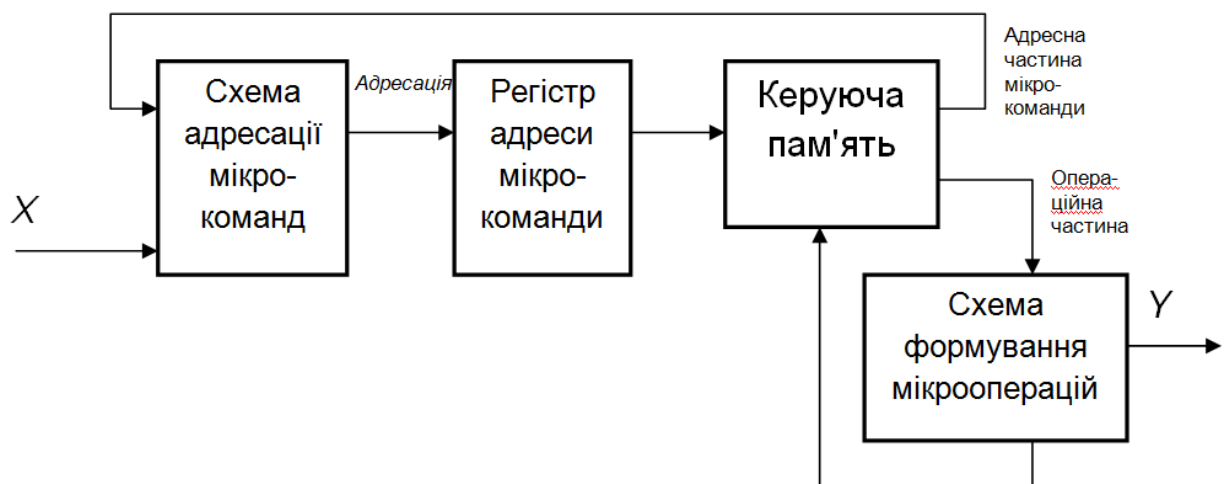


Рисунок 2.2. - Структура мікропрограмного автомата з «програмованою» логікою

Він складається із схеми адресації мікрокоманд (САМК), яка формує адресу наступної мікрокоманди в реєстрі адреси мікрокоманд (РАМК) (в залежності від логічних умов X та адресної частини поточної мікрокоманди ($АЧ$)). Для зберігання мікрокоманд застосовують керуючу пам'ять (КП). Операційна частина (ОЧ) мікрокоманд за допомогою схеми формування мікрооперацій (СФМО) перетворюється на набір мікрооперацій Y , які керують цифровою системою в цілому [1, 25].

Алгоритм об'єднання такого ПК залежить від адресації мікрокоманд. Розрізняють три основні способи адресації мікрокоманд [4, 13, 20, 25, 27]:

- мікрокоманди володіють максимальною довжиною, але при цьому можна забезпечити максимальну швидкодію - примусова адресація мікрокоманд;

- мікрокоманди володіють мінімальною довжиною, але при цьому довжина мікропрограми збільшується, і як наслідок, зменшується швидкодія ПК - природна адресація мікрокоманд;

- комбінована адресація мікрокоманд (поєднання примусової та природної адресації мікрокоманд).

Серед основних недоліків мікропрограмних автоматів з «програмованою» логікою можна виділити наступні [4, 13, 20, 27]:

- зростання часу виконання мікропрограм за рахунок багатонаправлених переходів одразу за кілька тактів;

- збільшення розрядності КП, оскільки мікрокоманда містить адресу переходу.

Наведених вище недоліків майже частково немає в дворівневій структурі ПК, тобто пристрої мікропрограмного керування (ПМК), запропонованому в роботах [22, 23], який представляє собою поєднання мікропрограмних автоматів з "жорсткою" та "програмованою" логікою.

Використання такого підходу має наступні переваги [3, 22-24]:

1. Оскільки мікрокоманда складається лише з ОЧ, то можна забезпечити мінімально можливу розрядність КП.

2. Можливість застосування природної адресації мікрокоманд при цьому не збільшуючи загальну довжину мікропрограми, що в свою чергу зменшує ємність КП.

3. Можливість забезпечення мінімально можливого часу виконання алгоритму (у порівнянні з іншими методами побудови ПМК) за рахунок багатонаправлених переходів у мікропрограмі, які виконуються за один такт.

Як правило, алгоритм керування цифрової системи можуть задавати у вигляді граф-схеми алгоритму (ГСА) Γ [1, 3-6], який містить початкову b_0 , кінцеву b_E , операторні та умовні вершини.

Операторні вершини утворюють множину B_1 , яка має M елементів. У вершинах $b_q \in B_1$ записуються мікрокоманди [3, 24]:

$$Y_q \subseteq Y,$$

де $Y = \{y_1, \dots, y_N\}$ множина мікрооперацій.

В умовних вершинах, які утворюють множину B_2 , записують елементи множини логічних умов $X = \{x_1, \dots, x_L\}$.

Вершини ГСА Γ утворюють множину $B = B_1 \cup B_2 \cup \{b_0, b_E\}$, елементи якого зв'язані дугами з множини E .

В цьому випадку для інтерпретації ГСА Γ може бути використане ПМК із загальною пам'яттю [3]. На рис. 2.3 показано структуру такого ПМК (1).

Схема адресації мікрокоманд САМК і лічильник ЛТ утворюють мікропрограмний автомат адресації мікрокоманд S_1 , а лічильник ЛТ, керуюча пам'ять КП та тригер читання ТЧ утворюють ПМК S_2 з природною адресацією мікрокоманд [3, 24]. ПМК (1) із загальною пам'яттю функціонує наступним чином.

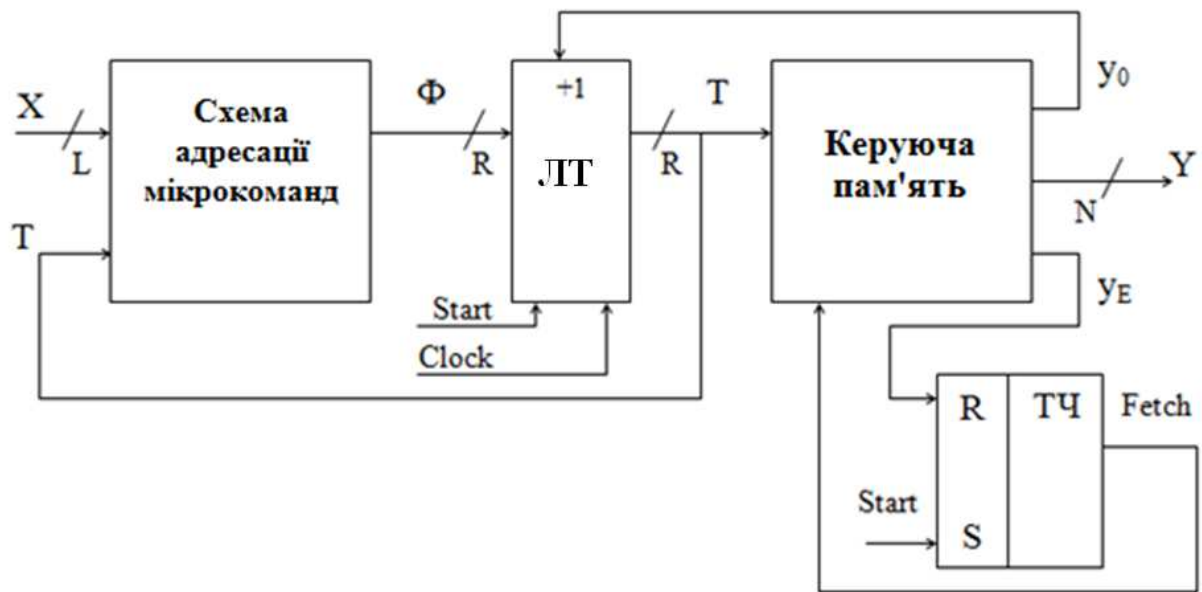


Рисунок 2.3. - Структура ПМК (1)

Після сигналу Start в лічильник ЛТ заноситься адреса першої мікрокоманди мікропрограми, яка відповідає ГСА L , після чого тригер читання ТЧ встановлюється в одиничний стан.

Якщо Fetch=1, то відбувається зчитування мікрокоманди з КП. У разі якщо вихід O_g поточного ОЛЛ $\alpha_g \in C$ не відповідає досягнутому, то одночасно з мікроопераціями $y_n \in Y$ формується сигнал $y_0=1$. В цьому випадку вміст ЛТ зростає на одиницю і потім здійснюється перехід у межах поточного ОЛЛ $\alpha_g \in C$.

Коли вихід ОЛЛ $\alpha_g \in C$ досягнуто, то адреса переходу формується функціями збудження лічильника $\phi_r \in \Phi$, де $\Phi = \{\phi_1, \dots, \phi_R\}$. В обох випадках вміст лічильника змінюється за сигналом Clock.

Коли виході КП формується сигнал y_E , то тригер ТЧ обнуляється і вибірка мікрокоманд з КП припиняється. Цикл функціонування ПМК (1) повторюється під час подачі чергового сигналу Start=1.

Побудова такого ПМК (1) складається з наступних етапів [3]:

1. Формування множини ОЛЛ, вмісту керуючої пам'яті та адресація мікрокоманд.
2. Формування таблиці переходів ПМК.
3. Формування системи функцій збудження лічильника Φ .
4. Організація та побудова логічної схеми ПМК.

Тут недоліком ПМК (1) може бути необхідність використання сигналів зворотного зв'язку R для формування функцій Φ , що зазвичай перевищує кількість змінних, необхідних для кодування станів автомата Мура [3, 5, 12]. Також, кількість рядків таблиці переходів ПМК (1) є максимально можливою і в подальшому її збільшити немає можливості. Тому для усунення цього недоліку запропоновано ПМК (2) з перетворювачем адреси мікрокоманд (ПАМ), який наведено на рис. 2.4 [3, 9, 24].

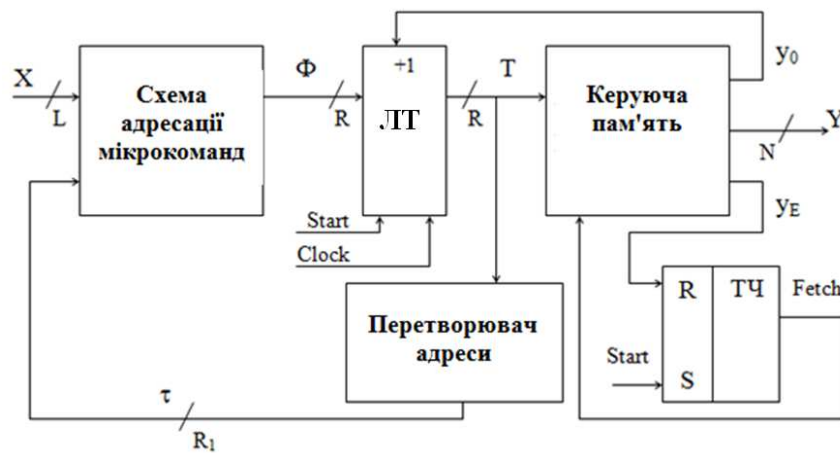


Рисунок 2.4. - Структура ПМК (2)

У ПМК (2) комбінаційна САМК, лічильник ЛТ та перетворювач адреси ПАМ утворюють автомат адресації мікрокоманд S_I .

ПМК (2) функціонує наступним чином [3, 24]. Після сигналу Start вміст ЛТ обнуляється, що відповідає вихідному стану ПМК, після чого відбувається зчитування мікрокоманди з КП.

При переходах всередині ОЛЛ $\alpha_g \in C$ сигнал $y_0 = 1$ та стан автомата S_l не змінюється. У випадку коли вихід поточного ОЛЛ досягнуто, то $y_0 = 0$ і автомат S_l формує за адресою виходу ОЛЛ $\alpha_g \in C$ код $K(B_i)$ ОЛЛ $B_i \in P_C$ та адресу входу чергового ОЛЛ $\alpha_g \in C_u$ ЛТ. Коли мікрокоманда досягла $Y(b_q) \subseteq Y$, що $\langle b_q, b_E \rangle \in E$, то формується сигнал $y_E = 1$, після чого функціонування ПМК U_2 припиняється.

Побудова такого ПМК (2) складається з наступних етапів [24]:

1. Формування множини ОЛЛ, вмісту КП та адресація мікрокоманд.
2. Розбиття множини ОЛЛ на класи ОЛЛ P_C (псевдоеквівалентні).
3. Кодування класів псевдоеквівалентних ОЛЛ.
4. Формування таблиці переходів ПМК.
5. Формування таблиці перетворювача адреси ПАМ.
6. Формування систем функцій збудження лічильника Φ та перетворення адреси τ .
7. Організація та побудова логічної схеми ПМК.

Таким чином використання такого ПМК (2) дасть можливість зменшити кількість аргументів у системі адресації мікрокоманд. І як наслідок, при виконанні умови $R_1 < R$ це призводить до зменшення кількості входів комбінаційної САМК.

Також перевагою є те, що кількість рядків таблиці переходів є мінімально можливою. Але необхідно звернути увагу, що досягнути цього можна за рахунок наявності схеми ПАМ, яка споживає деякі апаратні ресурси.

2.2. Застосування в цифровій техніці мікросхем типу «система на кристалі»

Сучасні існуючі SoC мають у своєму складі засоби для реалізації довільної логіки (FPGA (Field-Programmable Gate Array - програмована користувачем вентильна матриця (ПКВМ)) або CPLD (Complex programmable logic device -

програмовані логічні інтегральні схеми (ПЛІС))) [7, 11, 27, 29, 30, 32], засоби для реалізації пам'яті у вигляді вбудованих блоків ROM і RAM, вбудовані мікропроцесори та засоби реконфігурації [5, 13].

Як правило, FPGA складається з достатньо великої кількості логічних макрокомірок - програмованих логічних блоків (ПЛБ) PLB, які поєднані між собою програмованими схемами комутації. В залежності від виробника і структура ПЛБ відрізняється.

Основним елементом ПЛБ є логічна комірка, яка складається з S-входового функціонального генератора, який також називають логічним елементом табличного типу (ЛЕ) (LUT, look up table), логіки прискореного перенесення та елемента, призначеного для запам'ятовування. Структура такого ПЛБ наведена на рис. 2.5 [13].

Вихід кожного функціонального генератора кожної логічної комірки підключено до виходу ПЛБ та D-входу тригера.

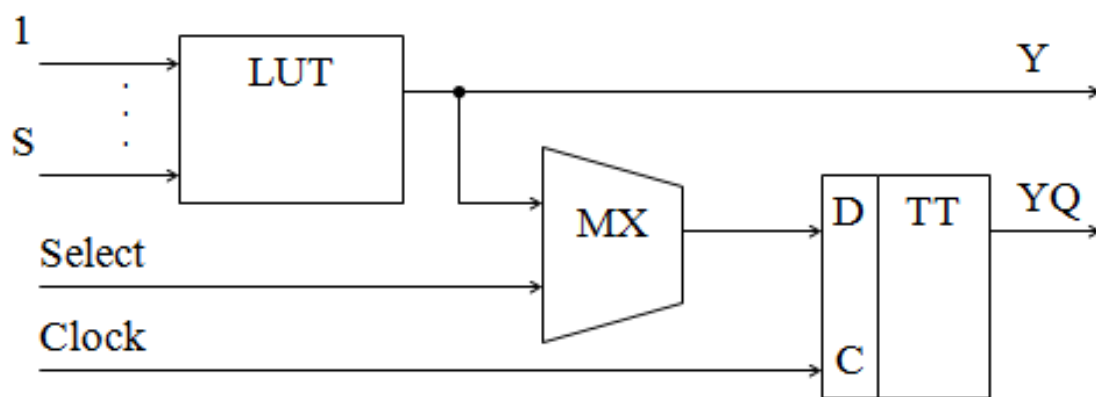


Рисунок 2.5. - Структура ПЛБ

Кожний ПЛБ вміщують кілька логічних комірок, які об'єднані у секції. Сучасні мікросхеми ПКВМ включають дуже багато (близько сотні тисяч) ПЛБ. На рис. 2.6 показана архітектура ПКВМ FPGA. Архітектура FPGA також включає вбудовані блоки пам'яті (ВБП) (DMB, dedicated memory block), блоки введення-виведення (IOB) модулі автопідстроювання затримок (DLL), призначені для компенсації затримок тактових сигналів, а також поділу,

множення та зсуву фази тактових частот, а також ряд додаткових елементів [16, 17, 19, 21, 25, 27, 29].

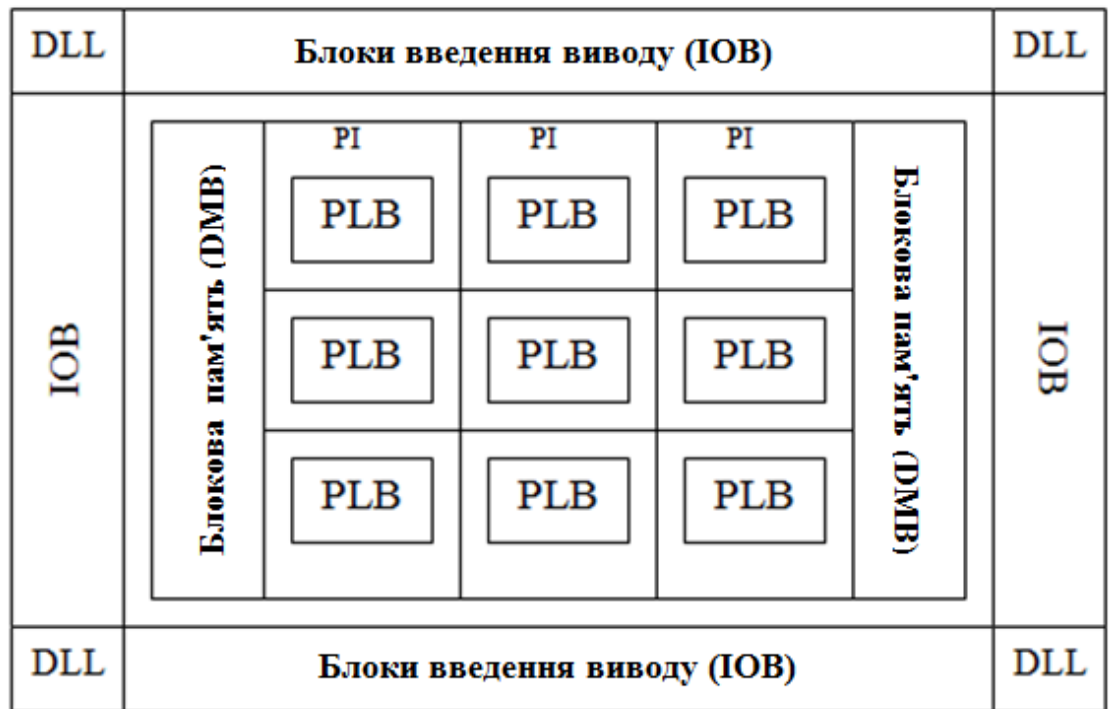


Рисунок 2.6. - Структура архітектури ПКВМ

Наразі основними виробниками ПЛІС з архітектурою FPGA є компанії Xilinx, Actel, Altera, Atmel.

Основні характеристики сучасних ПЛІС з архітектурою FPGA наведено в табл. 2.1 [3, 16, 17, 19, 21, 25, 27, 29]:

$N_{\text{ПЛБ}}$ - кількість програмованих логічних блоків;

S_{LUT} - кількість входів LUT елемента;

V_{RAM} - ємність блоків вбудованої пам'яті;

$N_{\text{ІО}}$ – кількість програмованих вводів/виводів користувача;

$F_{\text{МАХ}}$ – максимальна тактова частота.

**Таблиця 2.1. - Основні технічні характеристики
ПЛІС з архітектурою FPGA**

№ з/п	Фірма- виробник	Сімейство	N_{PLB}	S_{LUT}	V_{RAM}, КБіт	N_{IO}	F_{max}, МГц
1	XILINX	Virtex-5LX XC5VLX330	207306	6	3420	1200	550
2	XILINX	Virtex-4LX XC4LX200	200448	5	1392	960	450
3	XILINX	Spartan-3E XC3S1600E	33192	4	648	376	333
4	XILINX	Spartan-3 XC3S5000	74880	4	1872	784	326
5	Altera	Cyclone III EP3C55	55856	4	2340	535	600
6	Altera	StratixIII E EP3SE50	66500	6	5328	288	600
7	Actel	Axcelerator AX2000	32256	3	338	684	500
8	Actel	ProASIC ^{PLUS} APA1000	16128	3	198	712	350

Аналіз табл. 2.1 показує, що реалізація комбінаційних схем на ПЛІС з архітектурою FPGA можлива в основному на 4-6-входових LUT-елементах. Наявність на кристалі мікросхеми вбудованої пам'яті дозволяє реалізувати пристрої з ОЗУ/ПЗУ ємністю близько сотень кілобайт. Таким чином, структура ПМК відповідає можливостям мікросхем типу «система на кристалі» SoC.

2.3. Методи оптимізації логічних схем пристроїв мікропрограмних керування

Основні методи оптимізації логічної схеми автомата керування, що використовуються в даний час, можна поділити на три групи [3, 24]:

- методи структурної редукції, які в свою чергу призводять до багаторівневої логічної схеми мікропрограмного автомата;
- методи гетерогенної реалізації, полягають у застосуванні різного елементного базису для реалізації схем різних рівнів;
- алгоритмічні методи, полягають в оптимальному кодуванні станів системи та введенні додаткових змінних у систему.

На рис. 2.7 наведена структура ПМК (3) з поділом кодів. Це необхідно при використанні роздільної адресації ОЛЛ та їх компонент.

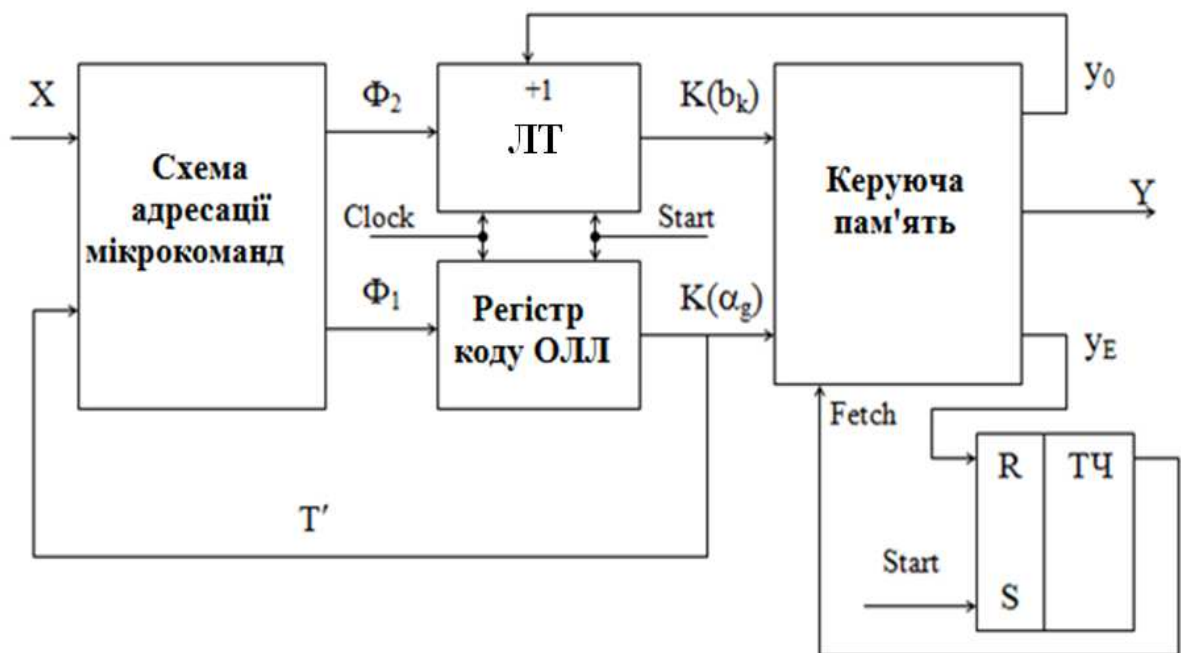


Рисунок 2.7. - Структурна схема ПМК (3)

За допомогою методу поділу кодів можна зменшити число сигналів зворотного зв'язку, які надходять на комбінаційну САМК до величини R_{11} , а також дозволяє використовувати всі відомі методи оптимізації автомата Мура [3] з метою зменшення апаратних витрат у САМК.

2.4. Основні завдання

ПМК завдяки мінімальним витратам у КП є ефективним засобом реалізації пристроїв керування на ПЛІС з архітектурою FPGA. Але в той же час, методи оптимізації мікропрограмних автоматів Мілі та Мура недостатньо повно застосовуються для мінімізації апаратних витрат ПМК.

В бакалаврській роботі пропонується використати методи структурної редукції (пониження порядку вихідних моделей системи) та алгоритмічні методи (надання алгоритму (порядку) дій у певній формі щодо мети, цільової функції, завдання та засобів і способів виконання поставленої задачі) зниження апаратних витрат в мікропрограмному автоматі адресації ПМК.

Щоб зменшити кількість ЛЕ (LUT-елементів) у схемі адресації ПМК треба щоб старші розряди адреси виходів всіх ОЛЛ $\alpha_g \in C$ мали різні значення, і тоді їх можна застосовувати для однозначної ідентифікації ОЛЛ. А в подальшому оптимізацію структури можна виконувати за допомогою використання методу кодування логічних умов.

Побудова та організація ПМК з оптимальною адресацією мікрокоманд, при цьому обов'язково має бути ГСА псевдоеквівалентних ОЛЛ, забезпечить зменшення кількості ЛЕ LUT елементів у схемі ПМК.

В бакалаврській роботі ставляться такі основні задачі:

1. Здійснити побудову ПМК з ідентифікацією виходів, з метою зменшення кількості входів схеми адресації ПМК із загальною пам'яттю.
2. Застосувати метод кодування логічних умов для ПМК з ідентифікацією виходів, які можна реалізувати на ПЛІС з архітектурою FPGA.
3. Здійснити побудову ПМК з оптимальною адресацією мікрокоманд з поділом кодів, з метою зниження апаратних витрат ПЛІС у схемі ПМК.

Для побудови ПМК пропонується застосовувати цифрові автомати, які запропоновано в [28], де для опису кінцевих автоматів використовується формат KISS2, який використовують в промислових системах автоматизованого проектування. Такий автомат представляють за допомогою

таблиці переходів [3]. Принцип роботи автомата та його опис задається у вигляді текстового файлу, в якому кожний рядок відповідає певному рядку таблиці переходів [9, 11-13, 28].

2.5. Висновки

1. Пристрої мікропрограмного керування дозволяють ефективно реалізовувати алгоритми керування цифровими системами. Головними перевагами ПМК є мінімально можлива ємність КП та можливість виконання за один такт багатонаправлених переходів.

2. Головними перевагами проектування ПМК типу "система на кристалі" FPGA є використання мінімальних можливих ресурсів ВБП DMV з метою реалізації системи мікрооперацій та зменшення кількості функцій з великим числом аргументів.

3. За рахунок поєднання різних методів оптимізації мікропрограмних автоматів з "жорсткою" логікою, наприклад, оптимальна адресація мікрокоманд, оптимальне кодування ОЛЛ та ін. можна зменшити апаратні витрати в автоматі адресації ПМК. Застосування даного підходу дасть змогу поліпшити та вдосконалити існуючі структури ПМК, які спрямовані оптимізацію використання ресурсів FPGA.

РОЗДІЛ 3 АНАЛІЗ ТА ПОБУДОВА ПРИСТРОЇВ МІКРОПРОГРАМНОГО КЕРУВАННЯ

3.1. Аналіз та побудова пристрою мікропрограмного керування з використанням методу ідентифікації виходів

У роботі [23] для розробки та оптимізації ПМК із загальною пам'яттю запропоновано метод ідентифікації виходів. Метод полягає в тому, що у випадку ідентифікації виходів ОЛЛ $\alpha_g \in C'$, де $C' \subseteq C$ – множина ОЛЛ, мінімальна потужність якої визначається як:

$$R_{\min} = \lceil \log_2 G_1 \rceil, \quad (3.1)$$

де $G_1 = |C'|$.

Як правило, цю властивість застосовують з метою мінімізації кількості числа входів схеми САМК ПМК із загальною пам'яттю. Структура ПМК (4) з ідентифікацією виходів [23, 24] показана на рис. 3.1.

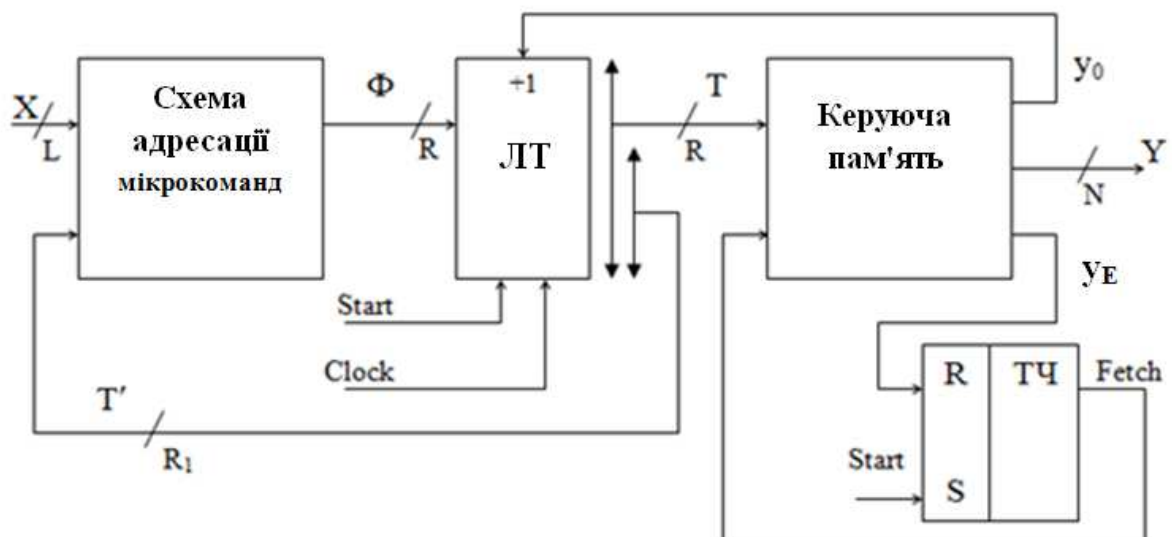


Рисунок 3.1. - Структурна схема ПМК (4)

САМК разом з лічильником ЛТ утворюють автомат адресації мікрокоманд S1. Лічильник ЛТ, тригер ТЧ та КП утворюють ПМК S2 з природною адресацією мікрокоманд [3, 24].

Лічильник ЛТ зберігає адреси $A(b_k)$ мікрокоманд, які надають змінні $T_r \in T$
 $= \{T_1, \dots, T_R\}$, де $R = \lceil \log_2 M \rceil$.

Мікрокоманди $Y(b_k)$ зберігаються в пам'яті. Один з додаткових розрядів використовується для зберігання сигналу y_0 , за допомогою якого виконується режим адресації, а y_E використовується для зберігання сигналу, після чого виконується завершення функціонування пристрою.

Після сигналу Start в лічильник ЛТ заноситься адреса першої мікрокоманди відповідного алгоритму. ТЧ встановлюється в одиничний стан (Fetch=1), що дозволяє здійснювати вибірку мікрокоманд з КП.

Припустимо, що під час t ($t=0, 1, \dots, n$) в ЛТ є адреса $A(b_q)$. Якщо вершина $b_q \in B_1$ не є виходом ОЛЛ $\alpha_g \in C$, то одночасно з мікроопераціями $A(b_q) \subseteq Y$, які записані у вершині b_q , повинен формуватися сигнал y_0 . Тоді до вмісту ЛТ додається 1, це відповідає режиму природної адресації, після чого в ЛТ заноситься адреса $A(b_t)$ наступної мікрокоманди, де $\langle b_q, b_t \rangle \in E$.

Якщо $b_q = O_g$, тоді сигнал y_0 не виробляється і адреса переходу формується вихідними функціями схеми САМК:

$$\Phi = \Phi(T', X), \quad (3.2)$$

де $T' \subseteq T$, $|T'| = R_1$, $T' = \{T_1, \dots, T_{R_1}\}$ – множина адресних розрядів, яку вистачає для однозначної ідентифікації виходів ОЛЛ $\alpha_g \in C'$. Таким чином відбувається перехід між виходом O_g та входом ОЛЛ $\alpha_i \in C$.

Перемикання лічильника ЛТ ініціюється сигналом синхронізації Clock. У випадку коли ЛТ містить адресу $A(b_t)$ та $\langle b_t, b_E \rangle \in E$, то одночасно з мікроопераціями $y_n \in Y(b_t)$ формується сигнал y_E .

У випадку $y_E=1$, Fetch=0 вибірка мікрокоманд з КП припиняється.

Цикл функціонування ПМК (4) знову повторюється під час подачі

чергового сигналу $Start=1$.

Основною перевагою ПМК (4) є можливість використання вбудованих у SoC блоків пам'яті DBM за допомогою яких можна реалізувати наступне:

$$Y = Y(T), \quad (3.3)$$

Таким чином можна зменшити загальну кількість ЛЕ LUT-елементів у схемі пристрою керування у порівнянні з її побудовою у вигляді мікропрограмного автомата [23, 25].

Даний метод побудови ПМК (4) з ідентифікацією виходів складається з наступних етапів:

1. Формування множини ОЛЛ С;
2. Адресація мікрокоманд;
3. Формування вмісту КП;
4. Формування множини T' , яка призначена для ідентифікації виходів ОЛЛ;
5. Формування таблиці переходів ПМК;
6. Формування системи функцій збудження ЛТ;
7. Побудова (синтез) логічної схеми ПМК.

Основну ідею методу можна показати з прикладу синтезу з прикладу синтезу пристрою керування алгоритму “dk14”, зазначеного в роботі [25]. Опис роботи мікропрограмного автомата наведений у форматі KISS2 у вигляді текстового файлу:

```
.i 3
.o 5
.p 16
.s 12
--- state_1 state_2 11000
--- state_2 state_3 00110
--- state_3 state_4 01000
```

```

1-- state_4 state_5 00100
01- state_4 state_7 00110
00- state_4 state_10 01000
--- state_5 state_6 01001
-1- state_6 state_2 00000
-0- state_6 state_8 00010
--- state_7 state_8 11000
--- state_8 state_9 00001
--1 state_9 state_11 10100
--0 state_9 state_12 01010
--- state_10 state_2 00110
--- state_11 state_1 00001
--- state_12 state_1 00100

```

.e

Тут:

- .i 3 - кількість входів автомата;
- .o 5 – кількість виходів автомата;
- .p 16 - кількість переходів автомата;
- .s 12 – кількість внутрішніх станів автомата.

Таким чином за даним описом було побудовано ГСА Γ_1 , яка наведена на рис. 3.2. На основі цієї ГСА розроблено алгоритм керування цифровою системою.

З рис. 3.2 є можливість побудувати множину V_1 операторних вершин ГСА Γ_1 , де $V_1 = \{b_1, \dots, b_{15}\}$, $M = 15$. Таким чином для адресації мікрокоманд можна використати $R = 4$ змінних $T_r \in T$. В подальшому для побудови ГСА Γ_1 було використано результати роботи [3].

Тоді отримаємо множину:

$$C = \{\alpha_1, \alpha_2, \dots, \alpha_7\}$$

де $\alpha_1 = \langle b_1, b_2, b_3 \rangle$, $\alpha_2 = \langle b_4, b_5 \rangle$, $\alpha_3 = \langle b_6, b_7, b_8 \rangle$, $\alpha_4 = \langle b_9, b_{10} \rangle$, $\alpha_5 = \langle b_{11} \rangle$, $\alpha_6 = \langle b_{12}, b_{13} \rangle$, $\alpha_7 = \langle b_{14}, b_{15} \rangle$.

Як видно з табл. 3.1 старші $R_1 = 3$ розряду адрес виходів ОЛЛ якщо $\alpha_g \in C$, то для всіх ОЛЛ вони мають різні значення. Отже старші R_1 розряду адреси виходу ОЛЛ можна застосовувати для її ідентифікації.

В прикладі множина $C' = \{\alpha_1, \alpha_2, \dots, \alpha_5\}$ при цьому $G_I = 5$ та $R_{Imin} = R_I$, тоді в ПМК (4) ГСА Γ_1 для ідентифікації виходів ОЛЛ можливо застосування елементів множини $T' = \{T_1, T_2, T_3\}$.

Таблиця переходів ПМК (4) містить наступні стовпці: O_g ; $K(O_g)$; I_q^j ; $A(I_q^j)$; X_h ; Φ_h ; h .

де O_g – вихід ОЛЛ $\alpha_g \in C'$;

$K(O_g)$ – старші R_1 розрядів адреси мікрокоманди, яка відповідає виходу ОЛЛ $\alpha_g \in C'$;

I_q^j – j -й вхід ОЛЛ $\alpha_q \in C$ ($j = \overline{1, F_g - 1}$);

$A(I_q^j)$ – адреса входу I_q^j ;

X_h – вхідний сигнал, за допомогою якого визначають перемикання лічильника ЛТ з адреси виходу ОЛЛ $\alpha_g A(O_g)$ на адресу входу $A(I_q^j)$;

$\Phi_h \subseteq \Phi$ – набір функцій збудження лічильника ЛТ, для перемикання ЛТ з $A(O_g)$ в $A(I_q^j)$ вони приймають одиничне значення;

$h = \overline{1, N}$ – номер переходу.

Для ПМК (4) ГСА Γ_1 таблиця переходів (табл. 3.2) містить $N = 9$ рядків.

Слід зауважити, що переходи з виходів ОЛЛ $\alpha_g \in C$, які пов'язані з кінцевою вершиною, розглянуто не було. Це пояснюється з тим, що при досягненні таких виходів функціонування ПМК (4) зупиняється.

Система функцій збудження лічильника (3.2) формується на основі таблиці переходів ПМК (4) наступним чином:

$$\varphi_r = \bigvee_{h=1}^N C_{rh} E_g^h X_h \quad (r = \overline{1, R}) , \quad (3.4)$$

де C_{rh} – булева змінна, яка дорівнює 1, тоді і тільки тоді, коли в h -му рядку

таблиці переходів записана функція $\varphi_r = 1$ ($h = \overline{1, R_1}$);

Таблиця 3.2. - Таблиця переходів ПМКУ₇ (Г₁)

O_g	$K(O_g)$	I_{q^j}	$A(I_{q^j})$	X_h	Φ_h	h
O_1	001	I_2^1	0011	x_1	$D_3 D_4$	1
		I_3^1	0101	$\overline{x_1 x_2}$	$D_2 D_4$	2
		I_4^1	1000	$\overline{x_1 x_2}$	D_1	3
O_2	010	I_1^2	0001	x_2	D_4	4
		I_5^1	1010	$\overline{x_2}$	$D_1 D_3$	5
O_3	011	I_6^1	1011	x_3	$D_1 D_3 D_4$	6
		I_7^1	1101	$\overline{x_3}$	$D_1 D_2 D_4$	7
O_4	100	I_1^2	0001	$-$	D_4	8
O_5	101	I_3^2	0111	$-$	$D_2 D_3 D_4$	9

E_g^h – кон'юнкція (множення) змінних $T_r \in T'$, що відповідає коду $K(O_g)$ адреси виходу O_g з рядка таблиці:

$$E_g^h = \bigwedge_{r=1}^{R_1} T_r^{lgr}, \quad (3.5)$$

де $lgr \in \{0, 1\}$ – значення r -го розряду коду $K(O_g)$ $T_r^0 = \overline{T_r}$, $T_r^1 = T_r$ ($r = \overline{1, R_1}$).

Після підстановки виразів (3.4) та (3.5) до системи (3.2) отримаємо:

$$D_1 = \overline{T_1} \overline{T_2} T_3 \overline{x_1 x_2} \vee \overline{T_1} T_2 \overline{T_3 x_2} \vee \overline{T_1} T_2 T_3,$$

$$D_2 = \overline{T_1} \overline{T_2} T_3 \overline{x_1 x_2} \vee \overline{T_1} T_2 T_3 \overline{x_3} \vee T_1 \overline{T_2} T_3,$$

$$D_3 = \overline{T_1} \overline{T_2} T_3 x_1 \vee \overline{T_1} T_2 \overline{T_3 x_2} \vee \overline{T_1} T_2 T_3 x_3 \vee T_1 \overline{T_2} T_3,$$

$$D_4 = \overline{T_1} \overline{T_2} T_3 x_2 \vee \overline{T_1} T_2 \overline{T_3 x_2} \vee \overline{T_1} T_2 T_3 \vee T_1 \overline{T_2}.$$

Побудова логічної схеми ПМК тепер зводиться до реалізації виразів системи (3.2) на ЛЕ LUT-елементах та реалізації КП на DBM.

В якості прикладу було проведено синтез схеми САМК, при цьому було використано ПЛБ (PLB) з 4-входними ЛЕ LUT-елементами. Спочатку необхідно перетворити отриману систему (3.2) в наступний вигляд:

$$D_1 = \overline{T_1} \left(\overline{T_2 T_3 x_1 x_2} \vee T_2 \overline{T_3 x_2} \right) \vee \overline{T_1} T_2 T_3,$$

$$D_2 = \overline{T_1} T_3 \left(\overline{T_2 x_1 x_2} \vee T_2 \overline{x_3} \right) \vee T_1 \overline{T_2} T_3,$$

$$D_3 = \overline{T_1} \left(\overline{T_2 T_3 x_1} \vee T_2 \overline{T_3 x_2} \right) \vee T_3 \left(\overline{T_1 T_2 x_3} \vee T_1 \overline{T_2} \right),$$

$$D_4 = \overline{T_1} x_2 (T_2 \oplus T_3) \vee \overline{T_1} T_2 T_3 \vee T_1 \overline{T_2},$$

Таким чином: для реалізації функції D_1 потрібно 2 ЛЕ LUT-елементи; функції D_2 – 2 ЛЕ LUT-елементи; функції D_3 – 3 ЛЕ LUT елементи; функції D_4 – 2 ЛЕ LUT-елементи.

На рис. 3.3 наведена схема реалізації функції D_1 .

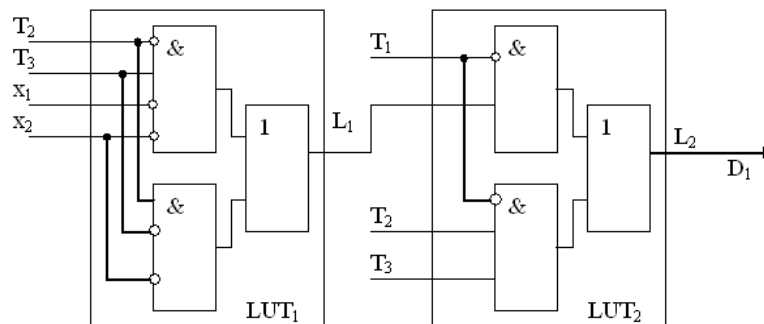


Рисунок 3.3. - Реалізація функції D_1 на логічних елементах

Виконаємо подальше спрощення. Введемо позначення: L_i вихід i -го ЛЕ LUT-елемента, тоді отримаємо наступну систему функцій (3.2):

$$L_1 = \overline{T_2} \overline{T_3} \overline{x_1} \overline{x_2} \vee T_2 \overline{T_3} \overline{x_2}, L_2 = D_1 = \overline{T_1} L_2 \vee \overline{T_1} T_2 T_3,$$

$$L_3 = \overline{T_2} \overline{x_1} x_2 \vee T_2 \overline{x_3}, L_4 = D_2 = \overline{T_1} T_3 L_3 \vee T_1 \overline{T_2} T_3,$$

$$L_5 = \overline{T_2} T_3 x_1 \vee T_2 \overline{T_3} \overline{x_2}, L_6 = \overline{T_1} T_2 x_3 \vee T_1 \overline{T_2}, L_7 = D_3 = \overline{T_1} L_5 \vee T_3 L_6,$$

$$L_8 = \overline{T_1} x_2 (T_2 \oplus T_3), L_9 = D_4 = L_8 \vee \overline{T_1} T_2 T_3 \vee T_1 \overline{T_2}.$$

На рис. 3.4 показана схема ПМК (4), яка інтерпретує ГСА Г₁. Тут використані 4-вхідні LUT-елементи та ВБП (DMB), які мають s=4-адресних входи та t=8-виходів. І в результаті, для побудови САМК необхідно використати 9 ЛЕ LUT-елементів.

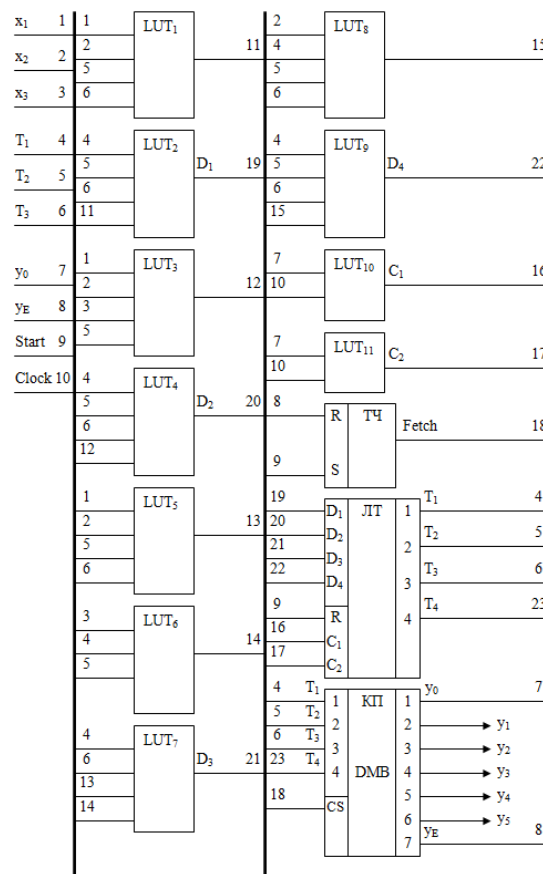


Рисунок 3.4. - Функціональна схема ПМК (4) ГСА (Г₁)

Таким чином можна зменшити кількість логічних LUT-елементів у схемі адресації ПМК шляхом зменшення числа сигналів зворотнього зв'язку.

Використання даного методу полягає у збільшенні годин такту ПМК, тому що було введено в схему нові структурні елементи.

Крім того також, зменшення числа кількості аргументів системи функцій збудження дає можливість зменшити години такту, у випадку якщо зменшиться число кількості рівнів комбінаційної схеми.

Розглянемо випадок здійснення звичайної процедури адресації, запропонований в [25]. В такому разі адреси виходів мають випадковий характер, тоді для ідентифікації виходів ОЛЛ $\alpha_g \in C'$ може знадобитися така кількість розрядів R_1 , що буде:

$$R_1 \geq R_{1min}, \quad (3.6)$$

Наприклад, для ГСА Γ_2 в [25], яка задає алгоритм керування “dk17” [25], маємо:

$$C = \{\alpha_1, \alpha_2, \dots, \alpha_4\},$$

де $\alpha_1 = \langle b_1, b_2 \rangle$; $\alpha_2 = \langle b_3 \rangle$; $\alpha_3 = \langle b_4 \rangle$; $\alpha_4 = \langle b_5, b_6, \dots, b_{10} \rangle$; $M = 10$.

Тоді для адресації мікрокоманд ПМК (4) ГСА Γ_2 потрібно $R=4$ розрядів.

Для ГСА Γ_2 на рис. 3.5 показана вихідна таблиця адресації.

Як бачимо з рис. 3.5:

$$A(O_1)=0001, A(O_2)=0010, A(O_3)=0011, A(O_4)=1001.$$

При цьому $G_I=3$ та $R_{1min}=2$. Однак для ідентифікації виходів ОЛЛ $\alpha_g \in C'$ ГСА Γ_2 необхідно щоб був код розрядністю $R_I=4$, отже, $K(O_g) = A(O_g)$ для $g=1, 2, 3$.

Таким чином, $R_I=R > R_{1min}$ і тоді в даному випадку число кількості сигналів зворотнього зв'язку не зменшується. Що призводить до неможливості реалізації ПМК (4) з ідентифікацією виходів.

T ₁ T ₂ \ T ₃ T ₄		T ₁ T ₂			
		00	01	10	11
T ₃ T ₄	00	b ₁	b ₅	b ₉	*
	01	b ₂	b ₆	b ₁₀	*
	10	b ₃	b ₇	*	*
	11	b ₄	b ₈	*	*

Рисунок 3.5. - Адресація мікрокоманд ПМК (4) ГСА Г₂

Тому в бакалаврській роботі запропоновано процедуру спеціальної адресації мікрокоманд, яка у разі виконання умови [22-24]:

$$M < 2^R, \quad (3.7)$$

дозволить ідентифікувати виходи ОЛЛ $\alpha_g \in C'$.

При цьому таблицю адресації мікрокоманд слід розглядати як регістр зсуву праворуч, який має 2^R розрядів.

Алгоритм процедури спеціальної адресації мікрокоманд [3] виконується в наступній послідовності:

1. Початок.
2. Зробити адресацію мікрокоманд.
3. Задати, що прийнято $R_I = R_{Imin}$.
4. Побудувати таблицю адресації мікрокоманд (яка має 2^{R_I} стовпчиків, позначених R_I старшими змінними адреси, та 2^{R-R_I} рядків, позначених $R - R_I$ молодшими змінними адресами).

5. Якщо виходи ОЛЛ $\alpha_i, \alpha_j \in C$ знаходяться в одному стовпці таблиці, то треба здійснювати зсув інформації праворуч (починаючи з першої вершини ОЛЛ $\alpha_j (j > i)$). Клітинки таблиці, які звільняються, заповнювати символами *. Після чого продовжувати зсув до тих пір, поки виходи O_i і O_j не будуть в різних стовпчиках таблиці.

6. У разі якщо виходи всіх ОЛЛ $\alpha_g \in C$ ідентифікуються за допомогою R_1 розрядів, то переходимо до п.9.

7. Якщо в процесі зсуву стався вихід якоїсь вершини за межі таблиці, то $R_1 := R_1 + 1$.

8. У випадку $R_1 < R$, переходимо до п. 4.

9. Завершення.

Використання спеціальної процедури адресації у даному прикладі здійснюють в наступний спосіб.

Виходи O_1 , O_2 та O_3 знаходяться в першому стовпчику таблиці, тому на першому етапі інформація зсувається на 2 розряди праворуч, починаючи з вершини b_3 .

Таким чином, у першому стовпчику залишається тільки вихід O_1 , а виходи O_2 та O_3 переходять до другого стовбчика, як показано на рис. 3.6а. Потім на другому етапі проводиться зсув на 3 розряди праворуч, починаючи з вершини b_4 .

В результаті у другому стовпчику таблиці залишиться тільки вихід O_2 , а вихід O_3 перейде до третього стовпця таблиці, як наведено на рис. 3.6 б.

Тепер $K(O_1)=00$, $K(O_2)=01$, $K(O_3)=10$, $K(O_4)=11$, $T' = \{T_1, T_2\}$. При цьому кількість числа сигналів зворотнього зв'язку зменшилось з $R_1=4$ на $R_1=2$. Як результат, побудова (синтез) ПМК (4) ГСА Γ_2 з ідентифікацією виходів доцільна.

Слід зауважити, що досягти мінімального значення параметра R_1 можливе не в будь-якому випадку. Це залежить від того яким чином будуть поєднані параметри ГСА, а саме: кількість ОЛЛ, кількість компонентів ОЛЛ, число операторних вершин.

a)

T ₁ T ₂		T ₃ T ₄			
		00	01	10	11
T ₃ T ₄	00	b ₁	b₃	b ₇	*
	01	b₂	b₄	b ₈	*
	10	*	b ₅	b ₉	*
	11	*	b ₆	b₁₀	*

б)

T ₁ T ₂		T ₃ T ₄			
		00	01	10	11
T ₃ T ₄	00	b ₁	b₃	b₄	b ₈
	01	b₂	*	b ₅	b ₉
	10	*	*	b ₆	b₁₀
	11	*	*	b ₇	*

Рисунок 3.6. - Приклад використання алгоритму (процедури) спеціальної адресації мікрокоманд для ПМК (4) ГСА Г₂

Ця спеціальна процедура адресації мікрокоманд ПМК дає можливість зменшити сигнали зворотнього зв'язку САМК. Що в свою чергу призводить до зменшення вимог до входів логічних LUT-елементів, які реалізують САМК.

Таким чином в подальшому можливо зменшити апаратні витрати часу такту ПМК [3].

Даний метод побудови ПМК (4) з ідентифікацією виходів складається з наступних етапів:

1. Зформувати множину ОЛЛ С.
2. Виконати адресацію мікрокоманд.
3. Виконати спеціальну процедуру адресації мікрокоманд.
4. Зформування вміст КП.
5. Зформування множини Т', яка ідентифікує виходи ОЛЛ.
6. Зформування таблицю переходів ПМК.
7. Зформування систему функцій збудження ЛТ.
8. Побудова (синтез) логічної схеми ПМК.

3.2. Аналіз та побудова пристрою мікропрограмного керування з адресацією мікрокоманд

Запропонуємо при синтезі ПМК (1) із загальною пам'яттю застосовувати

оптимальну адресацію мікрокоманд, тоді структура ПМК (5) [22-24] буде мати вигляд, наведений на рис. 3.7.

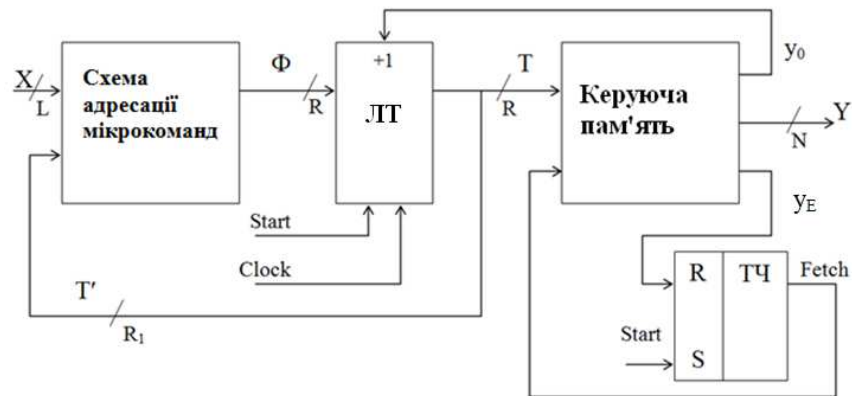


Рисунок 3.7. - Структурна схема ПМК (5)

Структури ПМК (1) і ПМК (5) однакові, але САМК ПМК (5), завдяки оптимальній адресації мікрокоманд, має $R_1 \leq R$ сигналів зворотнього зв'язку. Вона реалізує наступну систему функцій, які формують у лічильнику ЛТ адресу $A(I_g^j)$ j -го входу ОЛЛ $\alpha_g \in C (j \leq Fg)$:

$$\Phi = \Phi(T', X), \quad (3.8)$$

Принцип роботи функціонування ПМК (5) з оптимальною адресацією мікрокоманд здійснюється наступним чином.

Після сигналу $Start$ в ЛТ заноситься адреса першої мікрокоманди мікропрограми, яка відповідає граф-схемі ГСА Γ , тоді ТЧ встановлюється в 1 стан. Коли $Fetch = 1$ відбувається зчитування мікрокоманди із КП. Якщо вихід O_g поточного ОЛЛ $\alpha_g \in C$ не досягнутий, то одночасно з мікроопераціями $y_n \in Y$ формується сигнал $y_0=1$.

У цьому випадку вміст ЛТ збільшується на одиницю та відбувається перехід у межах поточного ОЛЛ $\alpha_g \in C$. Якщо вихід ОЛЛ $\alpha_g \in 3$ досягнуто, то адреса переходу формується функціями збудження лічильника.

В обох випадках стан лічильника змінюється за сигналом Clock. Якщо виході КП формується сигнал y_E , то ТЧ обнуляється і функціонування ПМК (5) зупиняється.

Основну ідею методу синтезу ПМК (5) з оптимальною адресацією мікрокоманд можна розглянути з прикладу ГСА Γ_2 (рис. 3.8), відповідного автомату “tav” [24, 25].

Як видно із рис. 3.8 для ГСА Γ_2 множина ОЛЛ:

$$C = \{\alpha_1, \dots, \alpha_5\},$$

де $\alpha_1 = \langle b_1, b_2 \rangle, I_1^1 = b_1, I_1^2 = b_2, O_1 = b_2$;

$\alpha_2 = \langle b_3, b_4 \rangle, I_2^1 = b_3, O_2 = b_4$;

$\alpha_3 = \langle b_5, b_6 \rangle, I_3^1 = b_5, O_3 = b_6$;

$\alpha_4 = \langle b_7, b_8, b_9 \rangle, I_4^1 = b_7, O_4 = b_9$;

$\alpha_5 = \langle b_{10}, b_{11} \rangle, I_5^1 = b_{10}, O_5 = b_{11}$;

множина виходів ГСА Γ_2 $O(\Gamma_2) = \{b_2, b_4, b_6, b_9, b_{11}\}$.

Для адресації мікрокоманд будується вектор $\alpha = \alpha_1 * \alpha_2 * \dots * \alpha_G$, де $*$ знак конкатенації, і i -й компонент кортежу α присвоюється код $K_i = (i-1)_2$ розрядності $R = \lceil \log_2 M \rceil$. Якщо $b_q = pr_i \alpha$, то $A(b_q) = K_i$ ($i = 1, \dots, M$). В даному прикладі $M=11$, $R=4$ та адреси мікрокоманд наведені у табл. 3.3.

Таблиця 3.3. - Адресація мікрокоманд ПМК (5) (Γ_2)

$T_1 T_2 T_3 T_4$	b_q	$T_1 T_2 T_3 T_4$	b_q	$T_1 T_2 T_3 T_4$	b_q	$T_1 T_2 T_3 T_4$	b_q
0000	b_1	0011	b_4	0110	b_7	1001	b_{10}
0001	b_2	0100	b_5	0111	b_8	1010	b_{11}
0010	b_3	0101	b_6	1000	b_9	1011	*

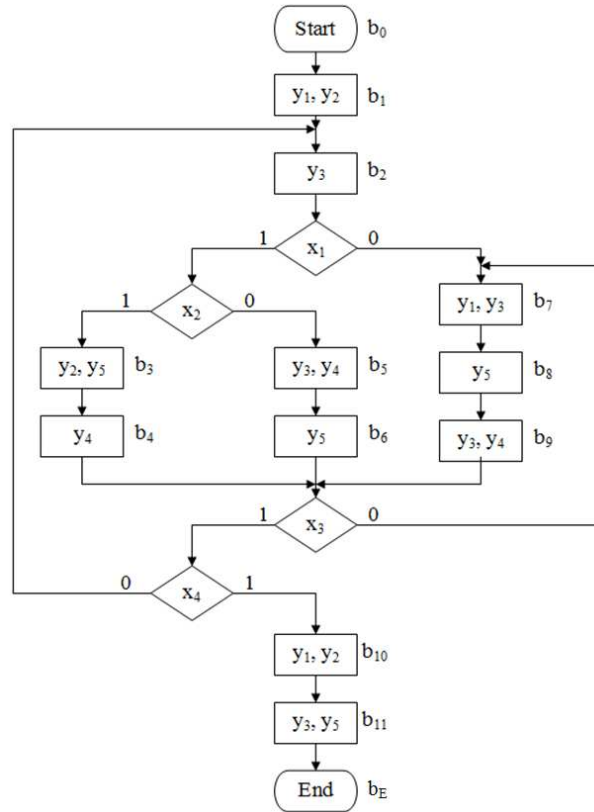


Рисунок 3.8. - Граф-схема алгоритму Γ_2

Система переходу ПМК (5) формується для виходів ОЛЛ $\alpha_g \in C'$, де $\alpha_g \notin C'$, якщо вихід цього ОЛЛ пов'язаний з кінцевою вершиною ГСА. Для даного прикладу ця система має наступний вигляд:

$$\begin{aligned}
 O_1 &\rightarrow x_1 x_2 I_2^1 \vee x_1 \overline{x_2} I_3^1 \vee \overline{x_1} I_4^1; \\
 O_2 &\rightarrow x_3 x_4 I_5^1 \vee x_3 \overline{x_4} I_1^2 \vee \overline{x_3} I_4^1; \\
 O_3 &\rightarrow x_3 x_4 I_5^1 \vee x_3 \overline{x_4} I_1^2 \vee \overline{x_3} I_4^1; \\
 O_4 &\rightarrow x_3 x_4 I_5^1 \vee x_3 \overline{x_4} I_1^2 \vee \overline{x_3} I_4^1.
 \end{aligned}
 \tag{3.9}$$

Зазначимо, що для ПМК (5) (Γ_2) $C' = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$.

При формуванні таблиці переходів ПМК кожен її рядок буде відповідати одному терму формули переходу тоді кількість рядків таблиці:

$$H_0 = \sum_{g=1}^{G_0} m_g, \quad (3.10)$$

де m_g – число переходів із виходу O_g ($g = \overline{1, G_0}$), $G_0 = |C'|$.

Необхідно оптимізувати схему САМК таким чином, щоб зменшити кількість числа літералів у термах та зменшити кількість термів. В результаті аналізу системи (3.9) було визначено, що формули переходу для виходів O_2 , O_3 та O_4 збігаються.

Припустимо, що $T(O_g)$ – підтаблиця таблиці переходів ПМК (5), де перераховані переходи з виходу ОЛЛ $\alpha_g \in C'$. Підтаблиця $T(O_g)$ містить m_g рядків ($g = \overline{1, G_0}$). Підтаблиці $T(O_2)$, $T(O_3)$ та $T(O_4)$ для даного прикладу будуть співпадати у стовпцях $I_q^j - \Phi_h$.

Якщо ОЛЛ $\alpha_2, \alpha_3, \alpha_4 \in C'$ однозначно подати одним об'єктом B_2 , то формули переходів для O_2 , O_3 та O_4 можна записати наступним чином:

$$B_2 \rightarrow x_3 x_4 I_5^1 \vee x_3 \overline{x_4} I_1^2 \vee \overline{x_3} I_4^1, \quad (3.11)$$

Тоді таблиця переходів ПМК (5) буде мати на 6 рядків менше, ніж таблиця, побудована за системою (3.9). Знайдемо розбиття ПСТ = $\{B_1, \dots, B_I\}$ множини ОЛЛ $C' \subseteq C$ на класи псевдоеквівалентних ОЛЛ.

Нехай клас $B_i \in P_C$ включає n_i елементи. Якщо у вихідній таблиці переходів усі ОЛЛ $\alpha_g \in C'$ замінемо класами $B_i \in P_C$, то система функцій збудження лічильника буде наступний вигляд:

$$H = \sum_{i=1}^I n_i \quad (3.12)$$

термів замість H_0 термів, які формуються за вихідною таблицею переходів. Зменшення ΔH числа термів визначається

$$\Delta H = \sum_{i=1}^I (n_i - 1) + m_i, \quad (3.13)$$

де $m_i = m_g$, якщо $\alpha_g \in V_i$.

Однозначну ідентифікацію класів $V_i \in \Pi_C$ можна здійснити шляхом оптимальної адресації мікрокоманд. Скористаємось картами Карно [5].

Карта Карно, яка містить адреси мікрокоманд дає клітин для адресації мікрокоманд ОЛЛ $\alpha_g \in C'$:

$$\Delta = 2^R - M_1, \quad (3.14)$$

тут M_1 - число компонент в ОЛЛ $\alpha_g \notin C'$. При виконанні умови:

$$\Delta \geq \sum_{i=1}^I \Delta_i, \quad (3.15)$$

адреси компонентів будь-якого ОЛЛ $\alpha_g \in C'$ можуть бути розташовані в сусідніх клітинах карти Карно, причому компоненти всіх ОЛЛ $\alpha_g \in V_i$ будуть розташовані в одному кубі розмірності Δ_i .

Оптимальна адресація мікрокоманд виконується за допомогою модифікованої карти Карно.

Модифікація карти полягає в тому, що з вертикалі записуються двійкові набори в довільному порядку. На рис. 3.9 показано Варіант оптимальної адресації.

T ₁ T ₂					
T ₃ T ₄		00	01	11	10
	00	b ₁	b ₃	b ₇	b ₁₀
	01	b ₂	b ₄	b ₈	b ₁₁
	10	*	b ₅	b ₉	*
	11	*	b ₆	*	*

Рисунок 3.9. - Модифікована карта Карно для результата адресації мікрокоманд ПМК (5) (Γ_2)

Для даного прикладу множина $C' = \{\alpha_1, \alpha_2, \dots, \alpha_4\}$,

$\Pi_C = \{B_1, B_2\}$, $I=2$, $B_1 = \{\alpha_1\}$, $B_2 = \{\alpha_2, \alpha_3, \alpha_4\}$.

Куби, що відповідають класам $B_1, B_2 \in \Pi_C$ та ОЛЛ α_5 , дають такі коди, що однозначно ідентифікують класи $B_i \in \Pi_C$: $K(B_1)=00^{**}$, $K(B_2)=*1^{**}$.

З цих кодів видно, що оптимальна адресація мікрокоманд дає можливість зменшити кількість змінних зворотнього зв'язку до R_1 .

Даний метод побудови ПМК (5) з оптимальною адресацією мікрокоманд включає наступні етапи:

1. Формуємо множину ОЛЛ. Для ГСА Γ_2 цей етап вже виконаний, і множина ОЛЛ $C = \{1, \dots, 5\}$.
2. Формуємо розбиття Π_C . $\Pi_C = \{B_1, B_2\}$.
3. Виконуємо оптимальну адресацію мікрокоманд.
4. Формуємо вміст КП. Для ПМК (5) вміст КП наведено у табл. 3.4.

Таблиця 3.4. - Вміст керуючої пам'яті ПМК (5) (Γ_2)

$A(b_q)$	$Y(b_q)$	b_q	$A(b_q)$	$Y(b_q)$	b_q	$A(b_q)$	$Y(b_q)$	b_q
0000	$y_0y_1y_2$	b_1	0110	$y_0y_3y_4$	b_5	1100	$y_0y_1y_3$	b_7
0001	y_3	b_2	0111	y_5	b_6	1101	y_0y_5	b_8
0100	$y_0y_2y_5$	b_3	1000	$y_0y_1y_2$	b_{10}	1110	y_3y_4	b_9
0101	y_4	b_4	1001	$y_3y_5y_E$	b_{11}	*	*	*

Тут сигнал y_0 введений у всі вершини $b_q \neq O_g$ ($g = \overline{1, G}$), сигнал y_E введений у пов'язану з кінцевою вершиною вершину b_{11} . Адреси мікрокоманд взято з карти Карно на рис. 3.9.

5. Формуємо системи формул переходу ПМК. Ця система будується шляхом заміни виходів ОЛЛ у системі формул переходу (3.9) на відповідні класи $B_i \in \Pi_C$.

6. Формуємо таблиці переходів ПМК. Таблиця будується за системою формул переходу та має стовпці: B_i , $K(B_i)$, I_q^j , $A(I_q^j)$, X_h , Φ_h , h , де $h=\overline{1, N}$ – номер рядка. У даному випадку таблиця переходів ПМК (5) має $N = 6$ рядків (табл. 3.5).

Таблиця 3.5. - Таблиця переходів ПМК (5) (Γ_2)

B_i	$K(B_i)$	I_q^j	$A(I_q^j)$	X_h	Φ_h	h
B_1	00^{**}	I_2^1	0100	$x_1 x_2$	D_2	1
		I_3^1	0110	$x_1 \overline{x_2}$	$D_2 D_3$	2
		I_4^1	1100	$\overline{x_1}$	$D_1 D_2$	3
B_2	$*1^{**}$	I_5^1	1000	$x_3 x_4$	D_1	4
		I_1^2	0001	$x_3 \overline{x_4}$	D_4	5
		I_4^1	1100	$\overline{x_3}$	$D_1 D_2$	6

Адреси входів $A(I_q^j)$ взяті з карти Карно на рис. 3.9.

7. Формуємо систему функцій збудження лічильника.

$$\varphi_r = \bigvee_{h=1}^N C_{rh} E_i^h X_h \quad (r = \overline{1, R_1}) \quad (3.16)$$

де C_{rh} – булева змінна, яка дорівнює 1, лише тоді, коли в h -му рядку таблиці переходів записана функція $\varphi_r = 1$; E_i^h – кон'юнкція змінних $T_r \in T'$, що відповідає коду $K(B_i)$ з h -го рядка таблиці переходів $h = \overline{1, N}$;

$$E_i^h = \bigwedge_{r=1}^{R_1} T_r^{l_{ir}} \quad (i = \overline{1, I}) \quad (3.17)$$

де $l_{ir} \in \{*, 0, 1\}$ – значення r -го розряду коду $K(B_i)$,

$$T_r^* = 1, \quad T_r^0 = \overline{T_r}, \quad T_r^1 = T_r \quad (r = \overline{1, R_1}).$$

Для ПМК (5), застосувавши систему (3.16) будемо мати:

$$D_1 = \overline{T_1 T_2 x_1} \vee T_2 x_3 x_4 \vee T_2 \overline{x_3}.$$

8. Побудова логічної схеми ПМК. Побудова зводиться до реалізації схеми САМК на LUT-елементах і КП на ВБД (DMB) (рис. 3.10).

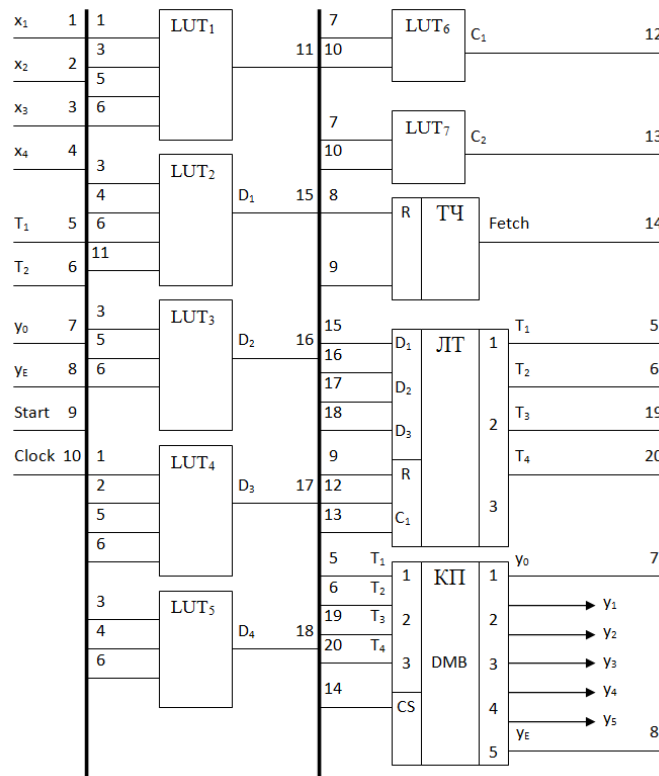


Рисунок 3.10. - Функціональна схема ПМК (5) (Γ_2)

При цьому число використаних у схемі САМК LUT-елементів знижується від 6 (ПМК (1)) до 5 для ПМК (5) (Γ_2).

Даний метод оптимальної адресації мікрокоманд ПМК дає змогу при виконанні такої умови:

$$R_1 < R, \quad (3.18)$$

зменшити кількість аргументів у системі функцій збудження ЛТ адреси

мікрокоманд.

А при виконанні умови

$$N < N_0, \quad (3.19)$$

зменшити кількість числа термів у цій системі порівняно з ПМК із загальною пам'яттю.

Застосування методу не потребує використання додаткових ресурсів кристала, а ПМК (1) та ПМК (5) мають однаковий час такту. Оптимальна адресація мікрокоманд виконується з використанням методики кодування псевдоеквівалентних станів автомата Мура [3, 22-24].

3.3. Висновки

1. Побудова ПМК з ідентифікацією виходів дає можливість зменшити кількість числа сигналів зворотнього зв'язку у схемі адресації мікрокоманд мікропрограми. В свою чергу це призводить до зменшення кількості числа LUT-елементів у схемі САМК. Таким чином можна оптимізувати площу кристала, яку він займає, та скоротити час такту ПМК загалом.

2. Побудова ПМК з оптимальною адресацією мікрокоманд мікропрограми дозволяє спростити схему адресації ПМК, яка реалізована за допомогою архітектури FPGA у порівнянні з ПМК із загальною пам'яттю. Застосування даного методу не потребує застосування додаткових ресурсів кристалу і забезпечує зменшення кількості числа необхідних LUT-елементів. Час такту, у порівнянні ПМК із загальної пам'яттю, залишається незмінним.

РОЗДІЛ 4 РОЗРОБКА СТРУКТУРНИХ ТА ФУНКЦІОНАЛЬНИХ СХЕМ ПРИСТРОЇВ МІКРОПРОГРАМНОГО КЕРУВАННЯ

4.1. Структурна та функціональна схема пристрою мікропрограмного керування з ідентифікацією виходів

4.1.1. Формування таблиць вмісту керуючої пам'яті та переходів.

Структурна схема.

Для ПМК (4) на основі ГСА, що описує алгоритм "opus" [25], було побудовано таблицю вмісту КП (табл. 4.1) і таблицю переходів (табл. 4.2). За допомогою цих таблиць можна сформуванати системи функцій збудження Ф лічильника ЛТ та побудувати схеми САМК.

Таблиця 4.1. - Вміст керуючої пам'яті ПМК (4)

$A(b_k)$	$Y(b_k)$	Коментарі	$A(b_k)$	$Y(b_k)$	Коментарі
0000	$y_0 y_1 y_2$	$b_1 I_1^1$	0110	$y_0 y_1 y_2$	$b_5 I_3^1$
0001	$y_0 y_3 y_4$	$b_2 I_1^2$	0111	$y_0 y_5$	$b_7 I_3^2$
<u>0010</u>	y_2	$b_3 O_1$	<u>1000</u>	$y_2 y_4$	$b_9 O_3$
0011	$y_0 y_1 y_4$	$b_4 I_2^1$	1001	$y_0 y_2 y_3$	$b_{10} I_4^1$
0100	$y_0 y_5$	$b_6 I_2^2$	<u>1010</u>	$y_5 \underline{y_K}$	$b_{12} O_4$
<u>0101</u>	$y_3 y_4$	$b_8 O_2$	1011	$y_0 y_1 y_4$	$b_{11} I_5^1$
			<u>1100</u>	$y_1 y_5 \underline{y_K}$	$b_{13} O_5$

З табл. 4.1 видно, що старші R_1 розрядів адреси виходу ОЛЛ $\alpha_g \in C$ однозначно її ідентифікують. У даному випадку $R_1=3$ та $T'=\{T_1, T_2, T_3\}$.

Таблиця переходів містить 9 рядків і складається із стовпців:

$O_g, K(O_g), I_q^j, A(I_q^j), X_h, \Phi_h, h;$

де O_g – вихід ОЛЛ $\alpha_g \in C$;

$K(O_g)$ – старші R_1 розрядів адреси мікрокоманди, що відповідає виходу

ОЛЛ $\alpha_g \in$, які використовуються для однозначної ідентифікації ОЛЛ $\alpha_g \in C$;

I_q^j - j й вхід ОЛЛ $\alpha_q \in C$ ($j = \overline{1, F_g - 1}$);

$A(I_q^j)$ – адреса входу I_q^j ;

X_h – вхідний сигнал, який визначає перемикання лічильника ЛТ з адреси виходу ОЛЛ $\alpha_g A(O_g)$ на адресу входу $A(I_q^j)$;

$\Phi_h \subseteq \Phi$ – набір функцій збудження лічильника ЛТ, що приймають 1 значення для перемикання ЛТ з $A(O_g)$ в $A(I_q^j)$;

$h = \overline{1, N}$ номер переходу.

Таблиця 4.2. - Таблиця переходів ПМК (4)

O_g	$K(O_g)$	I_q^j	$A(I_q^j)$	X_h	Φ_h	h
O_1	001	I_1^2	0001	x_1	D_4	1
		I_2^1	0011	$\overline{x_1} x_2$	$D_3 D_4$	2
O_1	001	I_3^1	0110	$\overline{x_1} x_2$	$D_2 D_3$	3
O_2	010	I_4^1	1001	$x_3 x_4$	$D_1 D_4$	4
		I_2^2	0100	$x_3 \overline{x_4}$	D_2	5
		I_5^1	1011	$\overline{x_3}$	$D_1 D_3 D_4$	6
O_3	100	I_5^1	1011	$x_3 x_5$	$D_1 D_3 D_4$	7
		I_3^2	0111	$x_3 \overline{x_5}$	$D_2 D_3 D_4$	8
		I_3^1	0110	$\overline{x_3}$	$D_2 D_3$	9

В даному випадку не розглядаються переходи з виходів ОЛЛ $\alpha_g \notin C'$, де $C' \subseteq C$ – множина ОЛЛ, виходи яких не пов'язані з входом вершини b_E .

Це пов'язано з тим, що при досягненні таких виходів функціонування ПМК (4) припиняється.

Як видно з табл. 4.2, максимальна кількість логічних умов, що

визначають перемикання лічильника ЛТ з адреси виходу $A(O_g)$ ОЛЛ α_g на адресу входу $A(I_q^j)$ ОЛЛ α_q . У цьому випадку дорівнює 2, що менше загальної кількості $L=5$ логічних змінних з множини $X=\{x_1, \dots, x_L\}$. Таким чином можна здійснити заміну множини логічних умов X на множину $P=\{p_1, \dots, p_U\}$ [3, 23].

У бакалаврській роботі пропонується в ПМК (4) застосувати метод кодування логічних умов для зменшення кількості числа входів схеми САМК, тоді ПМК (6) матиме вигляд, показаний на рис. 4.1. Принципи роботи функціонування ПМК (6) та ПМК (4) аналогічні.

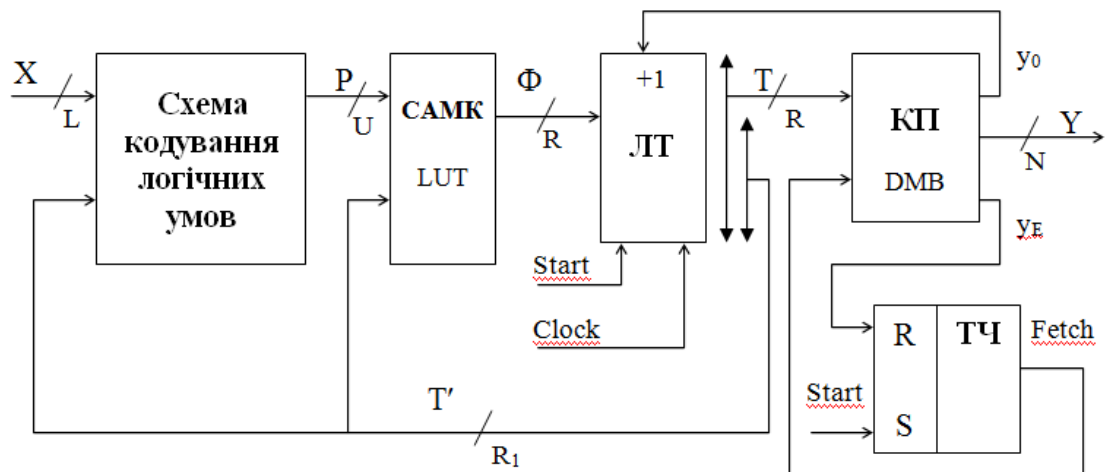


Рисунок 4.1. - Структурна схема ПМК (6)

Додаткова комбінаційна схема кодування логічних умов (СКЛУ) виконує заміну логічних умов $X=\{x_1, \dots, x_L\}$, где $L=|X|$, елементами множини $P=\{p_1, \dots, p_U\}$, де $U=|P|$, шляхом формування функцій кодування логічних умов

$$P = P(T', X), \quad (4.1)$$

При цьому комбінаційна схема САМК формує функції збудження лічильника ЛТ при переході між різними ОЛЛ $\alpha_g \in C$

$$\Phi = \Phi(T', P), \quad (4.2)$$

4.1.2. Етапи синтезу пристрою мікропрограмного керування.

Функціональна схема.

Застосуємо метод синтезу ПМК (6), що включає такі етапи:

1. Утворення множини ОЛЛ, за допомогою процедури спеціальної адресації мікрокоманд та формування вмісту КП.
2. Формування проміжної таблиці переходів ПМК.
3. Формування таблиці кодування логічних умов.
4. Організація системи функцій кодування логічних умов.
5. Формування перетвореної таблиці переходів ПМК.
6. Організація системи функцій збудження лічильника.
7. Реалізація логічної схеми ПМК за технологією FPGA.

Застосування методу синтезу ПМК (6) з прикладу ГСА Г₁.

1. Утворення безлічі ОЛЛ, застосування процедури спеціальної адресації мікрокоманд та формування вмісту керуючої пам'яті. Цей етап виконується за методикою [3, 23].

Для ПМК (6) таблиця вмісту КП повністю збігається з аналогічною таблицею для ПМК (4) (табл. 4.1). У виконанні процедури спеціальної адресації мікрокоманд потреби немає, оскільки $R_1 = R_{1\min} = 3$.

2. Формування проміжної таблиці переходів ПМК. Цей етап також виконується за методикою [3, 23] і проміжна таблиця повністю збігається з аналогічною таблицею переходів для ПМК (4) (табл. 4.2).

3. Формування таблиці кодування логічних умов. Цей етап виконується в наступній послідовності:

3.1. Формування множини $X(O_g)$ – множини логічних змінних, що визначають переходи з ОЛЛ $\alpha_g \in C$. При використанні проміжної таблиці переходів ПМК (6) (табл. 4.2) будемо мати:

$$X(O_1) = \{x_1, x_2\},$$

$$X(O_2) = \{x_3, x_4\},$$

$$X(O_3) = \{x_3, x_5\}.$$

3.2. Визначення числа U кодуючих змінних $p_i \in P$ за формулою

$$U = \max(U_1, \dots, U_G), \quad (4.3)$$

де $U_g = |X(O_g)|$, $g = \overline{1, G}$. У разі $U = 2$, тобто $P = \{p_1, p_2\}$.

3.3. Побудова таблиці кодування логічних умов, що складається з: O_g , $K(O_g)$, $p_1, \dots, p_U, g$,

де O_g – вихід ОЛЛ $\alpha_g \in C$;

$K(O_g)$ – код адреси мікрокоманди, що відповідає виходу ОЛЛ $\alpha_g \in C$, який використовується для її однозначної ідентифікації;

$p_1, \dots, p_U$ – кодують змінні $p_u \in P$;

$g = \overline{1, G}$ номер виходу.

На перетині рядка O_g і стовпця p_i записується змінна $x_i \in X$, що замінюється для виходу O_g змінної $p_u \in P$. При цьому кількість числа появи логічної умови $x_i \in X$ у різних стовпцях p_i таблиці має бути мінімальною. це можливо лише при виконанні наступної умови:

$$|X(p_i) \cap X(p_j)| \rightarrow \min, \quad (4.4)$$

де $p_i, p_j \in P$; $X(p_i), X(p_j)$ – множина логічних умов у стовпцях p_i, p_j відповідно.

Для ПМК (6) таблицю кодування логічних умов наведено в табл. 4.3.

Таблиця 4.3. - Таблиця кодування логічних умов ПМК (6)

O_g	$K(O_g)$	p_1	p_2	g
O_1	001	x_1	x_2	1
O_2	010	x_3	x_4	2
O_3	100	x_3	x_5	3

4. Організація системи функцій кодування логічних умов.

Система функцій (4.1) формується за таблицею кодування логічних умов ПМК (6) наступним чином:

$$p_u = \bigvee_{g=1}^G C_{ug} E_i^g x_1 \quad (u = \overline{1, U}), \quad (4.5)$$

де C_{ug} – булева змінна, що дорівнює 1, тільки в тому випадку якщо змінна $x_1 \in X$ замінюється для виходу O_g ОЛЛ $\alpha_g \in C$ змінною $p_u \in P$;

У даному випадку використання виразу (4.5) приводить до системи (4.1) в такому вигляді:

$$p_1 = \overline{T_1} \overline{T_2} T_3 x_1 \vee (T_1 \oplus T_2) \overline{T_3} x_3, p_2 = \overline{T_1} \overline{T_2} T_3 x_2 \vee \overline{T_1} T_2 \overline{T_3} x_4 \vee T_1 \overline{T_2} \overline{T_3} x_5.$$

5. Формування перетвореної таблиці переходів ПМК.

Перетворена таблиця переходів (табл. 4.4) є основою для формування системи функцій (4.2). Вона містить усі стовпці проміжної таблиці переходів ПМК, крім стовпця X_h , який замінюється стовпцем P_h .

У стовпці P_h змінні $x_1 \in X$ замінюються змінними $p_u \in P$. Інверсному значенню x_1 відповідає інверсне значення p_u .

6. Організація системи функцій збудження лічильника.

Систему (4.2) можна сформувати за перетвореною таблицею переходів ПМК (6):

$$\varphi_r = \bigvee_{h=1}^H C_{rh} E_g^h P_h \quad (r = \overline{1, R_1}), \quad (4.6)$$

де C_{rh} – булева змінна, що дорівнює 1, лише в тому випадку коли в h -й рядок перетвореної таблиці переходів записана функція $\varphi_r = 1$ ($h = \overline{1, H}$);

E_g^h – кон'юнкція змінних $T_r \in T'$, що відповідає коду $K(O_g)$ виходу O_g з рядка таблиці.

Таблиця 4.4. - Перетворена таблиця переходів ПМК (6)

O_g	$K(O_g)$	I_q^j	$A(I_q^j)$	P_h	Φ_h	h
O_1	001	I_1^2	0001	p_1	D_4	1
		I_2^1	0011	$\overline{p_1 p_2}$	$D_3 D_4$	2
		I_3^1	0110	$\overline{p_1 p_2}$	$D_2 D_3$	3
O_2	010	I_4^1	1001	$p_1 p_2$	$D_1 D_4$	4
		I_2^2	0100	$\overline{p_1 p_2}$	D_2	5
		I_5^1	1011	$\overline{p_1}$	$D_1 D_3 D_4$	6
O_3	100	I_5^1	1011	$p_1 p_2$	$D_1 D_3 D_4$	7
		I_3^2	0111	$\overline{p_1 p_2}$	$D_2 D_3 D_4$	8
		I_3^1	0110	$\overline{p_1}$	$D_2 D_3$	9

Використання виразу (4.6) для ПМК (6) надає змогу здійснити наступне:

$$D_1 = \overline{T_1} T_2 \overline{T_3} p_1 p_2 \vee \overline{T_1} T_2 \overline{T_3} \overline{p_1} \vee T_1 \overline{T_2} \overline{T_3} p_1 p_2,$$

$$D_2 = \overline{T_1} \overline{T_2} T_3 \overline{p_1} \overline{p_2} \vee \overline{T_1} T_2 \overline{T_3} \overline{p_1} \overline{p_2} \vee T_1 \overline{T_2} \overline{T_3} \overline{p_1} \overline{p_2} \vee T_1 \overline{T_2} \overline{T_3} \overline{p_1},$$

$$D_3 = \overline{T_1} \overline{T_2} T_3 \overline{p_1} \vee \overline{T_1} T_2 \overline{T_3} \overline{p_1} \vee T_1 \overline{T_2} \overline{T_3},$$

$$D_4 = \overline{T_1} \overline{T_2} T_3 (p_1 \vee \overline{p_1} p_2) \vee \overline{T_1} T_2 \overline{T_3} (\overline{p_1} \vee p_1 p_2) \vee T_1 \overline{T_2} T_3 p_1.$$

7. Реалізація логічного схеми ПМК.

Побудова (синтез) зводиться до реалізації систем (4.1) та (4.2) на логічних елементах LUT-елементах та реалізації КП на ВБП (DMB). Виконуємо синтез комбінаційних схем СКЛУ та САМК ПМК (6).

Для цього використовуємо 4-входові логічні LUT-елементи, а КП – на блоці пам'яті DMB, яка має $s = 4$ адресних входів та $t = 8$ виходів (показано на рис. 4.2).

Загальна кількість LUT-елементів, які необхідно застосувати для реалізації схем СКЛУ та САМК ПМК (6) дорівнює 13, а для схеми САМК ПМК (4) потрібно 15 логічних LUT-елементів.

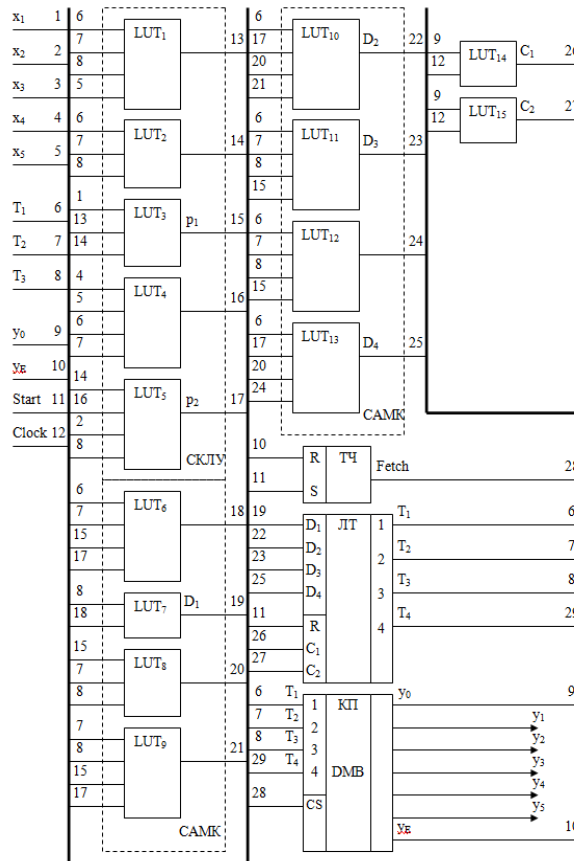


Рисунок 4.2. - Функціональна схема ПМК (6)

Використання методу кодування логічних умов зменшує кількість входів схеми САМК ПМК (6) порівняно з ПМК (4) на величину, що обернено пропорційна коефіцієнту η (визначається співвідношенням між величинами U та L) [3]:

$$\eta = \frac{U + R_1}{L + R_1}, \quad (4.7)$$

4.2. Структурна та функціональна схема пристрою мікропрограмного керування з адресацією мікрокоманд

4.2.1. Структура пристрою мікропрограмного керування з поділом кодів

У бакалаврській роботі пропонується для вдосконалення побудови ПМК з адресацією мікрокоманд використовувати процедуру елементаризації ОЛЛ із

поділом кодів. При цьому елементарним ОЛЛ будемо вважати ОЛЛ, що має тільки 1 вхід [3, 24]. Структура такого ПМК показана на рис. 4.3.

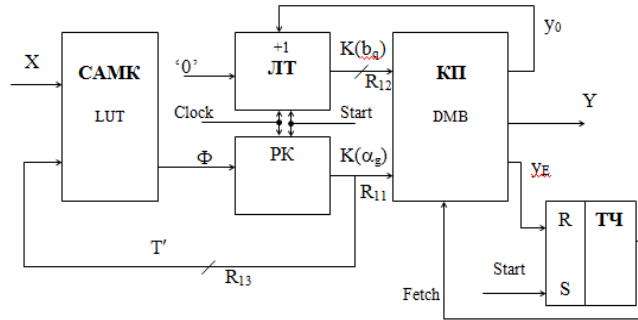


Рисунок 4.3. - Структурна схема ПМК (7)

Схема САМК, реалізована за допомогою архітектури FPGA, і в результаті формує наступну систему функцій

$$\Phi = \Phi(T', X), \quad (4.8)$$

яку задають у регістрі РК код $K(\alpha_g)$ поточного ОЛЛ $\alpha_g \in C$.

Адреса мікрокоманди представлена у наступному вигляді кодів [3, 24]:

$$A(Y_q) = K(\alpha_g) * K(b_q),$$

де $A(Y_q)$ – адреса мікрокоманди Y_q , що записана у вершині b_q , яка входить в ОЛЛ α_g і зберігається в КП;

$K(\alpha_g)$ – код ОЛЛ $\alpha_g \in C = \{\alpha_1, \dots, \alpha_G\}$ розрядності $R_{11} = \lceil \log_2 G \rceil$;

$K(b_q)$ – код вершини b_q кортежу $\alpha_g \in C$ розрядності

$$R_{12} = \lceil \log_2 F_{\max} \rceil, \quad (4.9)$$

де $F_{\max}$ - максимальна кількість компонент ОЛЛ у множині C .

Розрядність адреси мікрокоманди становить $R = R_{11}R_{12}$ і для її подання можна використати наступну множину внутрішніх змінних:

$$T = \{T_1, \dots, T_{R_{11}}, T_{R_{11}+1}, \dots, T_{R_{11}+R_{12}}\}$$

Використання елементарних ОЛЛ дозволяє подавати на вхід лічильника ЛТ нульовий код розрядності R_{12} , який відповідає коду першої компоненти поточного ОЛЛ $\alpha_g \in C$. При такому підході коди ОЛЛ та коди їх компонентів є незалежними один від одного. Тоді на вхід комбінаційної схеми САМК подаються тільки R_{11} змінних зворотнього зв'язку, що утворюють множину. Це дозволяє зменшити кількість числа логічних LUT-елементів, які використовують для реалізації схеми САМК, на відміну від схеми ПМК (5).

Керуюча пам'ять КП, реалізована на DMB, зберігає набори мікрооперацій Y_q , де $Y_q \subseteq Y = \{y_1, \dots, y_N\}$ – мікрооперації, записані у вершині $b_q \in B_I$ ГСА Г, та складається з $2^R \cdot (N+2)$ біт.

Перший додатковий розряд використовується для зберігання сигналу y_0 , що забезпечує природну адресацію компонент ОЛЛ $\alpha_g \in C$.

Другий – для організації режиму зупинки ПМК.

Функціонування ПМК (7) відбувається в такий спосіб.

Після сигналу Start вміст ЛТ і РК обнуляється. Тобто код першої компоненти першого ОЛЛ $\alpha_g \in C$, є адресою першої мікрокоманди відповідної ГСА Г. Тригер читання ТЧ встановлюється в 1 стан і відбувається зчитування мікрокоманди з КП. Якщо адреса $A(Y_q)$ компоненти не є адресою виходу поточного ОЛЛ $\alpha_g \in C$, то одночасно з мікроопераціями $Y_q \in Y$ формується сигнал $y_0=1$. Таким чином вміст лічильника збільшується на 1 і відбувається перехід до наступної компоненти Y_t поточного ОЛЛ $\alpha_g \in C$.

Якщо адреса $A(Y_q)A(Y_q)$ компоненти є адресою виходу O_g поточного ОЛЛ, сигнал $y_0 = 0$, лічильник обнуляється, і схема САМК формує код $K(\alpha_q)$ наступного ОЛЛ $\alpha_g \in C$.

Для представлення коду використовуються змінні $\varphi_r \in \Phi = \{\varphi_1, \dots, \varphi_{R_{11}}\}$.

При формуванні сигналу y_E тригер ТЧ скидається та функціонування ПМК (7) завершується.

4.2.2. Етапи синтезу пристрою мікропрограмного керування.

Функціональна схема.

Метод синтезу ПМК (7) включає в себе такі етапи:

1. Формування множини $C = \{\alpha_1, \dots, \alpha_G\}$ ОЛЛ вихідної ГСА Г.
 2. Розбиття ОЛЛ $\alpha_g \in C$ вихідної ГСА Г на елементарні ОЛЛ та формування множини елементарних ОЛЛ $C_E = \{\alpha_1, \dots, \alpha_{G_E}\}$.
 3. Формування розбиття $\Pi_{C_E} = \{V_1, \dots, V_l\}$ множини ОЛЛ C_E на класи псевдоеквівалентних ОЛЛ.
 4. Оптимальне кодування елементарних ОЛЛ $\alpha_g \in C_E$ двійковими кодами $K(\alpha_g)$ розрядності $R_{11} = \lceil \log_2 G_E \rceil$ та кодування компонентів b_q ОЛЛ $\alpha_g \in C_E$ двійковими кодами $K(b_q)$ розрядності R_{12} .
 5. Формування вмісту КП.
 6. Формування таблиці переходів ПМК зі стовпцями:
 $V_i, K(V_i), \alpha_q, K(\alpha_q), X_h, \Phi_h, h$,
де V_i – клас псевдоеквівалентних ОЛЛ $V_i \in \Pi_{C_E}$;
 $K(V_i)$ – код класу $V_i \in \Pi_{C_E}$;
 α_q – ОЛЛ $\alpha_q \in C_E$, в який є перехід з виходу ОЛЛ $\alpha_g \in V_i$;
 $K(\alpha_q)$ – код ОЛЛ $\alpha_q \in C_E$;
 X_h – сигнал умови переходу;
 Φ_h – набір змінних $\phi_r \in \Phi$, що приймають 1 значення для формування в РК коду $K(\alpha_q)$ наступної ОЛЛ $\alpha_q \in C_E$;
 h – номер переходу (номер рядка таблиці переходів) $h = \overline{1, N}$.
 7. Формування таблиці переходів ПМК системи функцій Φ (4.8).
 8. Синтез логічної схеми ПМК на архітектурі FPGA.
- Оптимальне кодування елементарних ОЛЛ $\alpha_g \in C'_E$ виконується аналогічно кодуванню псевдоеквівалентів станів автомата Мура.

При цьому кожному елементарному ОЛЛ $\alpha_g \in C'_E$ ставиться у відповідність R_{11} розрядний код та всі ОЛЛ $\alpha_g \in B_i$ повинні розташовуватися в одному кубі бульового простору.

Варіант оптимального кодування наведено на рис. 4.4 (оптимізація на основі карт Карно).

Як видно із рис. 4.4, отримані куби бульового простору дають коди, що однозначно ідентифікують класи $B_i \in \Pi_{C'_E}$: $K(B_1) = 00^*$, $K(B_2) = 01^*$,

$$K(B_3) = 10^*, K(B_4) = 11^*.$$

Аналіз результуючих кодів показує, що оптимальне кодування елементарних ОЛЛ призводить до зменшення кількості змінних зворотнього зв'язку $T_r \in T'$ від R_{11} до $R_{13} = \lceil \log_2 I \rceil = 2$.

	T_1	0	1
$T_2 T_3$	00	α_1	α_3
	01	α_6	α_4
	10	α_2	α_5
	11	*	*

Рисунок 4.4. - Оптимальне кодування ОЛЛ ПМК (7)

Результати кодування елементарних ОЛЛ наведено у табл. 4.5, коди ОЛЛ $\alpha_g \in C'_E$ взяті з карти Карно на рис. 4.4. Для кодування ОЛЛ $\alpha_7 \notin C'_E$ взято один із несуттєвих кодів.

Таблиця 4.5. - Кодування елементарних ОЛЛ

α_g	α_1	α_2	α_3	α_4	α_5	α_6	α_7
$K(\alpha_g)$	000	010	100	101	110	001	011

Відповідно (4.9) маємо $F_{\max}=4$, тобто для кодування компонентів ОЛЛ $\alpha_g \in C_E$ необхідно $R_{12}=2$ змінних $T_r \in T$. Для кодування компонентів ОЛЛ використовуються змінні T_4, T_5 (табл. 4.6).

Таким чином, для адресації мікрокоманд ПМК (7) необхідно використовувати $R = R_{11} + R_{12} = 3 + 2 = 5$ змінних, що не перевищує відповідного показника для ПМК (5) (Γ_2).

Однак, число змінних зворотного зв'язку $R_{13} = 2$ в ПМК (7), менше, ніж $R_1 = 5$ в ПМК (5) (Γ_2). Отже, застосування ПМК (7), що інтерпретує елементарні ОЛЛ з поділом кодів, є доцільним.

Таблиця 4.6. - Кодування компонент елементарних ОЛЛ

b_q	$K(b_q)$	b_q	$K(b_q)$	b_q	$K(b_q)$
b_1	00	b_8	10	b_{15}	01
b_2	01	b_9	00	b_{16}	10
b_3	00	b_{10}	01	b_{17}	11
b_4	01	b_{11}	00	b_{18}	00
b_5	10	b_{12}	01	b_{19}	01
b_6	00	b_{13}	10	-	-
b_7	01	b_{14}	00	-	-

Вміст пам'яті КП для ПМК (7) наведено в табл. 4.7, що містить 19 рядків. Тут сигнал y_0 введений у всі вершини, що не є виходами ОЛЛ $\alpha \alpha_g \in C_E$, сигнал y_E введений у вершину b_{19} , пов'язану з кінцевою вершиною b_E .

До побудови таблиці переходів ПМК (7) виконується формування системи формул переходу, у яких виходи ОЛЛ $\alpha_g \in B_i$ замінюються відповідними класами $B_i \in \Pi_{C'_E}$, а входи ОЛЛ замінюються ОЛЛ $\alpha \alpha_q \in C_E$, до яких відбувається перехід. У нашому випадку маємо:

$$\begin{aligned}
 B_1 &\rightarrow \alpha_2; B_2 \rightarrow x_1 \alpha_3 \vee \overline{x_1} x_2 \alpha_4 \vee \overline{x_1} \overline{x_2} \alpha_6; \\
 B_3 &\rightarrow \alpha_5; B_4 \rightarrow x_3 \alpha_7 \vee \overline{x_3} \alpha_4,
 \end{aligned}
 \tag{4.10}$$

При цьому не використовується формула для класу $V_5 \in \Pi_{C_E}$, до якого входить ОЛЛ $\alpha_7 \in C_E$, що містить вершину b_{19} , пов'язану з кінцевою вершиною b_E . Таблиця переходів ПМК (7) має $N=7$ рядків (табл. 4.8).

Коди $K(\alpha_q)$ ОЛЛ $\alpha_q \in C_E$ взято з табл. 4.8.

Для ПМК (7) маємо, наприклад, $D_2 = \overline{T_1} \overline{T_2} \vee T_1 \overline{T_2} \vee T_1 T_2 x_3$.

Синтез логічної схеми ПМК (7) виконується на логічних LUT-елементах (регістр РК реалізований на тригерах ТТ, що входять до складу PLB) і керуючої пам'яті КП на блоках DMB. Функціональну схему показано на рис. 4.5.

Таблиця 4.7. - Вміст керуючої пам'яті ПМК (7)

Адрес $A(b_q)$	Мікрокоманда $Y(b_q)$							Примітки	
	y_0	y_1	y_2	y_3	y_4	y_5	y_E	b_q	α_q
00000	1	1	0	0	0	0	0	b_1	α_1
00001	0	0	1	1	0	0	0	b_2	
01000	1	0	1	0	1	0	0	b_3	α_2
01001	1	1	1	0	0	0	0	b_4	
01010	0	0	0	1	1	0	0	b_5	
10000	1	1	1	0	0	0	0	b_6	α_3
10001	1	0	0	0	0	1	0	b_7	
10010	0	0	0	0	1	0	0	b_8	
10100	1	0	0	0	0	1	0	b_9	α_4
10101	0	1	0	0	1	0	0	b_{10}	
11000	1	0	1	0	0	1	0	b_{11}	α_5
11001	1	0	1	0	1	0	0	b_{12}	
11010	0	0	0	1	0	1	0	b_{13}	
00100	1	0	0	1	0	0	0	b_{14}	α_6
00101	1	1	0	0	0	1	0	b_{15}	
00110	1	0	1	0	1	0	0	b_{16}	
00111	0	1	0	1	0	0	0	b_{17}	
01100	1	0	1	0	0	0	0	b_{18}	α_7
01101	0	0	0	0	1	1	1	b_{19}	

Таблиця 4.8. - Таблиця переходів ПМК (7)

B_i	$K(B_i)$	α_q	$K(\alpha_q)$	X_h	Φ_h	h
B_1	00*	α_2	010	—	D_2	1
B_2	01*	α_3	100	x_1	D_1	2
		α_4	101	$\overline{x_1} x_2$	$D_1 D_3$	3
		α_6	001	$\overline{x_1} \overline{x_2}$	D_3	4
B_3	10*	α_5	110	—	$D_1 D_2$	5
B_4	11*	α_7	011	x_3	$D_2 D_3$	6
		α_4	101	$\overline{x_3}$	$D_1 D_3$	7

В результаті для реалізації автомата адресації ПМК (7) знадобилося 4 логічних елемента LUT-елементи, тоді як для ПМК (5) – 9 LUT-елементів.

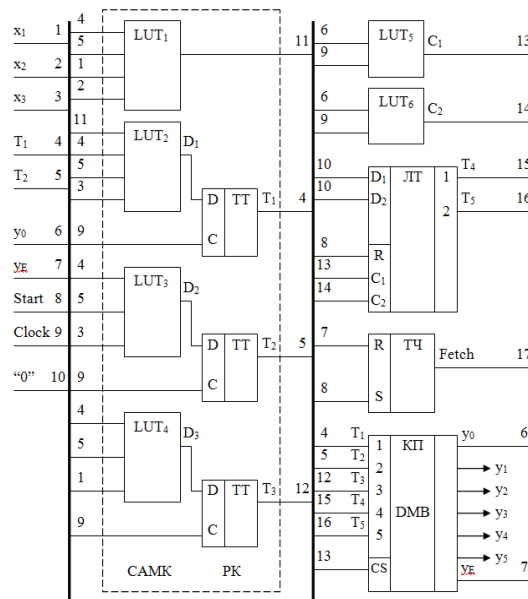


Рисунок 4.5. - Функціональна схема ПМК (7)

4.3. Теоретичний аналіз доцільності проведеної оптимізації

4.3.1. Загальна методика розрахунку

Зменшення апаратних витрат у логічній схемі ПМК є основним головним критерієм оптимальності. Оскільки метод побудови автомата адресації не впливає на ємність керуючої пам'яті, то мінімум апаратних витрат в ПМК можна визначити як мінімум витрат в схемі автомата адресації. Для розрахунку

використано методику та ймовірнісне подання характеристик граф-схеми алгоритму [3].

Припустимо, що K - загальна кількість вершин ГСА, K_1 - кількість операторних вершин ГСА, K_2 - кількість умовних вершин ГСА. Тоді:

$$p_1 = \frac{K_1}{K}; \quad p_2 = \frac{K_2}{K}, \quad (4.11)$$

де p_1 – доля операторних вершин ГСА; p_2 – доля умовних вершин ГСА. Слід зазначити, що $p_1 + p_2 = 1 - 2/K$, де параметр 2 враховує наявність початкової та кінцевої вершин ГСА. При зростанні параметра K величина $p_1 + p_2$ наближається до 1.

Використовуючи результати роботи [3], отримаємо параметри формул через загальну кількість вершин K і деякі ймовірнісні коефіцієнти:

1) Кількість рядків прямої структурованої таблиці автомата Мура:

$$H = 17.4 + 1.5p_1K. \quad (4.12)$$

2) Кількість логічних умов в ГСА

$$L = 0.7(1-p_1)K. \quad (4.13)$$

3) Розрядність адреси мікрокоманди:

$$R = \lceil \log_2 p_1 K \rceil. \quad (4.14)$$

ПМК (1) з загальною пам'яттю (взято за основу базової структури). Матрична модель ПМК (1) з загальною пам'яттю представлена на рис. 4.6.

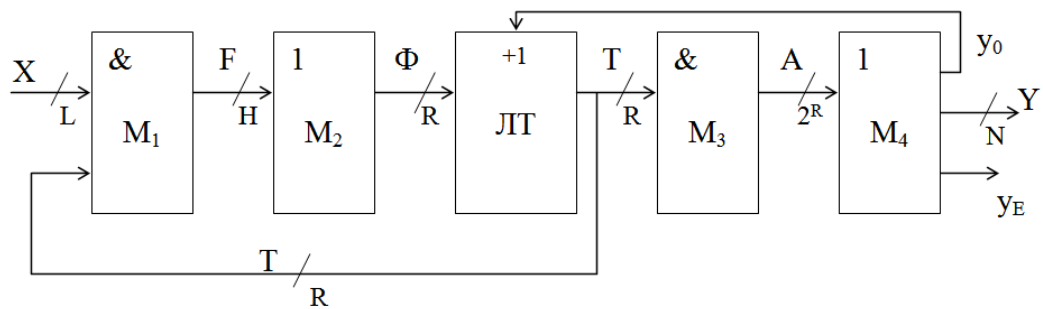


Рисунок 4.6. - Матрична модель ПМК (1)

Матриці M_1 і M_2 являють собою автомат адресації мікрокоманд S_1 , а матриці M_3 і M_4 - керуючу пам'ять ПК.

M_1 – кон'юнктивна матриця, яка реалізує терми $F_1, \dots, F_H$ автомата адресації та має площу $S(M_1)=2(L+R)*H$, де $R = \lceil \log_2 M \rceil$ - розрядність адреси мікрокоманди;

M_2 – диз'юнктивна матриця, яка реалізує функції збудження лічильника Φ і має площу $S(M_2)=R*H$;

M_3 – кон'юнктивна матриця, що реалізує адреси A мікрокоманд і має площу $S(M_3)=2R*2^R$;

M_4 – диз'юнктивна матриця, що реалізує набір мікрооперацій Y і має площу $S(M_4)=(N+2)*2^R$.

Таким чином, для реалізації автомата адресації ПМК (1) потрібна матриця площею [3]:

$$S_1(1) = (2L+3R) \cdot H, \quad (4.15)$$

для реалізації КП ПК потрібна матриця площею:

$$S(KП_1) = (2R + N + 2) \cdot 2^R. \quad (4.16)$$

З урахуванням виразів (4.12)-(4.13) формули (4.15) та (4.16) перетворюються до наступного вигляду:

$$S_1(1) = (1.4 \cdot (1 - p_1) \cdot K + 3 \cdot \lceil \log_2 p_1 K \rceil) \cdot (17.4 + 1.5 p_1 K), \quad (4.17)$$

$$S(K\Pi_1) = (2 \cdot \lceil \log_2 p_1 K \rceil + N + 2) \cdot 2^{\lceil \log_2 p_1 K \rceil}. \quad (4.18)$$

Таким чином апаратні витрати $S(1)$ треба рахувати як суму виразів (4.17) та (4.18).

Подальші розрахунки будуть проведені шляхом порівняння ПМК з ідентифікацією виходів та адресацією мікрокоманд з базовою схемою ПМК (1) з загальною пам'яттю.

4.3.2. Пристрій мікропрограмного керування з ідентифікацією виходів

Матрична модель ПМК з ідентифікацією виходів показана на рис. 4.7.

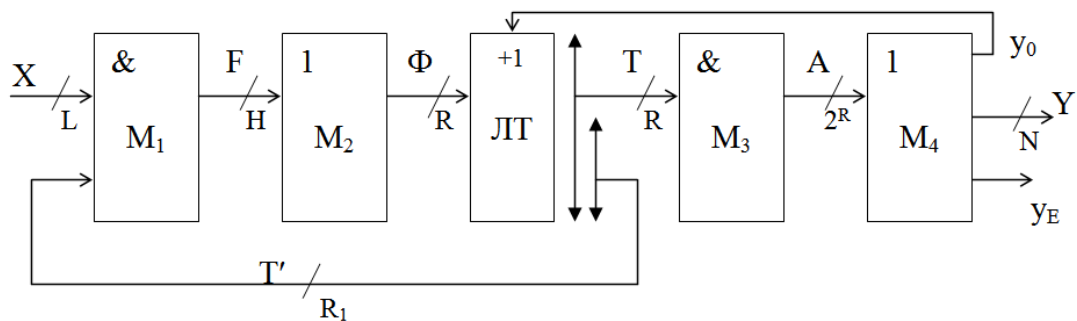


Рисунок 4.7. - Матрична модель ПМК з ідентифікацією виходів

Матрична модель ПМК схожа на матричну модель ПМК (1). Єдиний виняток це кон'юнктивна матриця M_1 , яка реалізує терми F прямої структурної таблиці автомата адресації.

На її вхід надходять змінні T' , які використовують для однозначної ідентифікації виходів ОЛЛ $\alpha_g \in C$, $|T'| = R_1 = \lceil \log_2 G \rceil$. Параметр $G = |C|$ можна

визначити як $G = p_1 p_3 K$, де p_3 – відношення кількості ОЛЛ до операторних вершин граф-схеми алгоритму.

Площа матриці M_1 становить $S(M_1) = 2(L + R_1) \cdot N$, і для реалізації автомата адресації ПМК необхідна матриця, яка має площу:

$$S_1(U_7) = (2L + 2R_1 + R) \cdot N. \quad (4.19)$$

З урахуванням виразів (4.12)-(4.13) тоді формулу (4.19) можна перетворити до наступного вигляду [3, 23]:

$$S_1(\text{ПМК}) = (1.4 \cdot (1 - p_1) \cdot K + 2 \cdot \log_2 p_1 p_3 K [+]) \log_2 p_1 K [-] \cdot (17.4 + 1.5 p_1 K), \quad (4.20)$$

Таким чином апаратні витрати $S(\text{ПМК})$ можна розрахувати як суму виразів (4.20) и (4.18).

Для визначення доцільності використання ПМК з ідентифікацією виходів у порівнянні з ПМК (1) з загальною пам'яттю необхідно побудувати залежність f_1 [3]:

$$f_1 = \frac{S_1(\text{ПМК})}{S_1(\text{ПМК}(1))}, \quad (4.21)$$

Також необхідно побудувати залежність f_2 , яка враховує загальні витрати включно з витратами на реалізацію КП:

$$f_2 = \frac{S_1(\text{ПМК}) + S(\text{КП})}{S_1(\text{ПМК}(1)) + S(\text{КП}(1))}, \quad (4.22)$$

Вирази (4.17) та (4.20) показують залежність функцій f_1 и f_2 від коефіцієнтів p_1 , p_3 , K – кількість вершин граф-схеми алгоритму; N – кількість

мікрооперацій.

Для дослідження залежностей (4.21) та (4.22) необхідно задати межі зміни цих параметрів. В роботі [3, 24] зазначено, що використання ПМК є доцільним при $p_1 \geq 0.75$. Тому прийmemo $p_1 \in \{0.75, 0.85, 0.95\}$. Параметр p_3 представимо як елемент множини $\{0.1, 0.3, 0.5\}$.

З [3, 24] відомо, що число мікрооперацій в реальних ПК може досягати $N=200$. Тому було прийнято N з множини $\{10, 50, 100, 200\}$. На рис. 4.8-4.9 наведені результати дослідження функції f_1 .

Зазвичай складність граф-схеми алгоритму цифрових систем знаходиться в діапазоні $100 \leq K \leq 1000$ [3, 24], тому дослідження функцій f_1 і f_2 необхідно виконати у всьому цьому діапазоні.

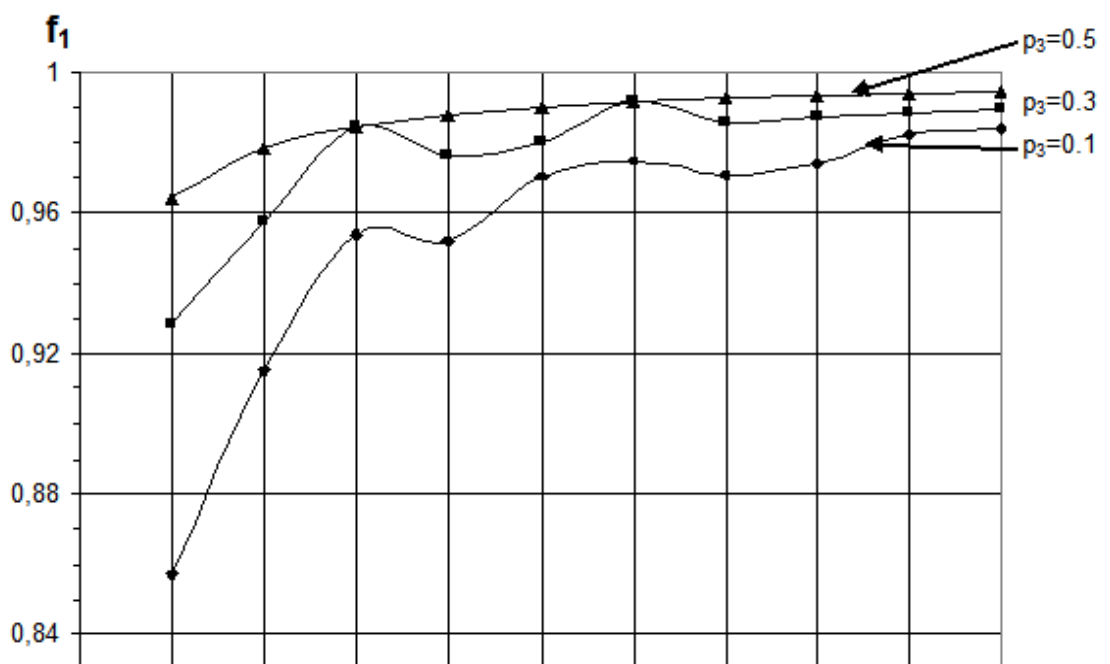


Рисунок 4.8. – Порівняння апаратних витрат в схемах автомата адресації ПМК та ПМК (1) при $p_1=0.75$

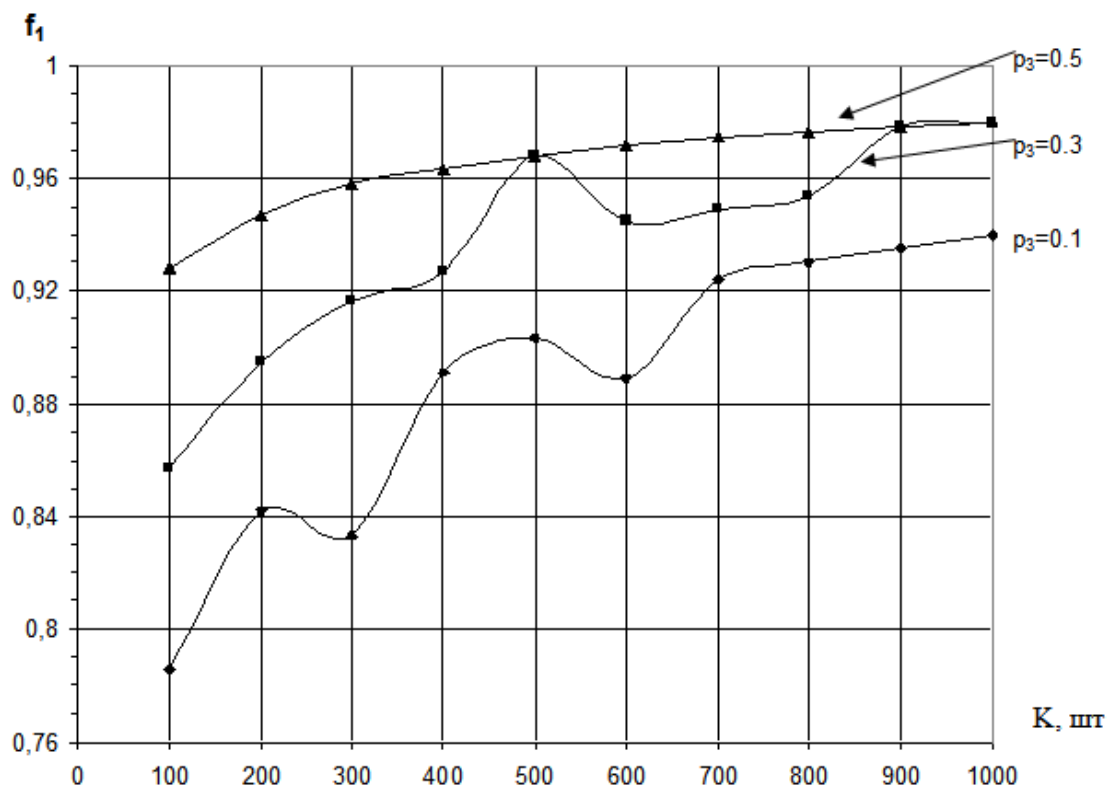


Рисунок 4.9. - Порівняння апаратних витрат в схемах автомата адресації ПМК та ПМК (1) при $p_1=0.95$

Як видно з графіків, показаних на рис. 4.8-4.9, впливає, що апаратні витрати автомата адресації ПМК з ідентифікацією виходів завжди нижчі за відповідний показник ПМК (1) з загальною пам'яттю, прийнятого за базову структуру.

Економія ресурсів зростає разом із зростанням параметра p_1 , тобто кількості операторних вершин ГСА. І в результаті досягає близько 22% (при $p_1=0.95$).

Результати досліджень функції f_2 представлені на рис. 4.10–4.11.

Як видно з рис. 4.10-4.11, загальна економія ресурсів збільшується зі зростанням кількості операторних вершин ГСА (параметр p_1) та зменшення числа мікрооперацій N . При цьому максимальне зниження апаратних витрат у схемі ПМК з ідентифікацією виходів досягає майже 16%, а мінімальне – близько 0.5%.

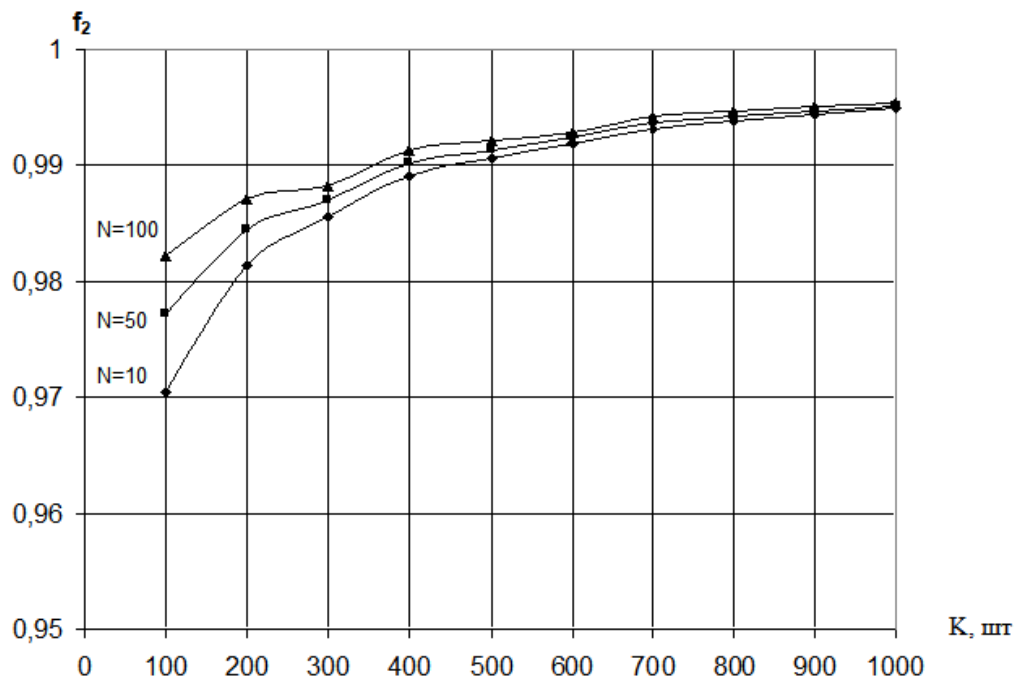


Рисунок 4.10. - Порівняння загальних апаратних витрат в схемах ПМК з ідентифікацією виходів та ПМК (1) з загальною пам'яттю при $p_1=0.75$; $p_3=0.5$

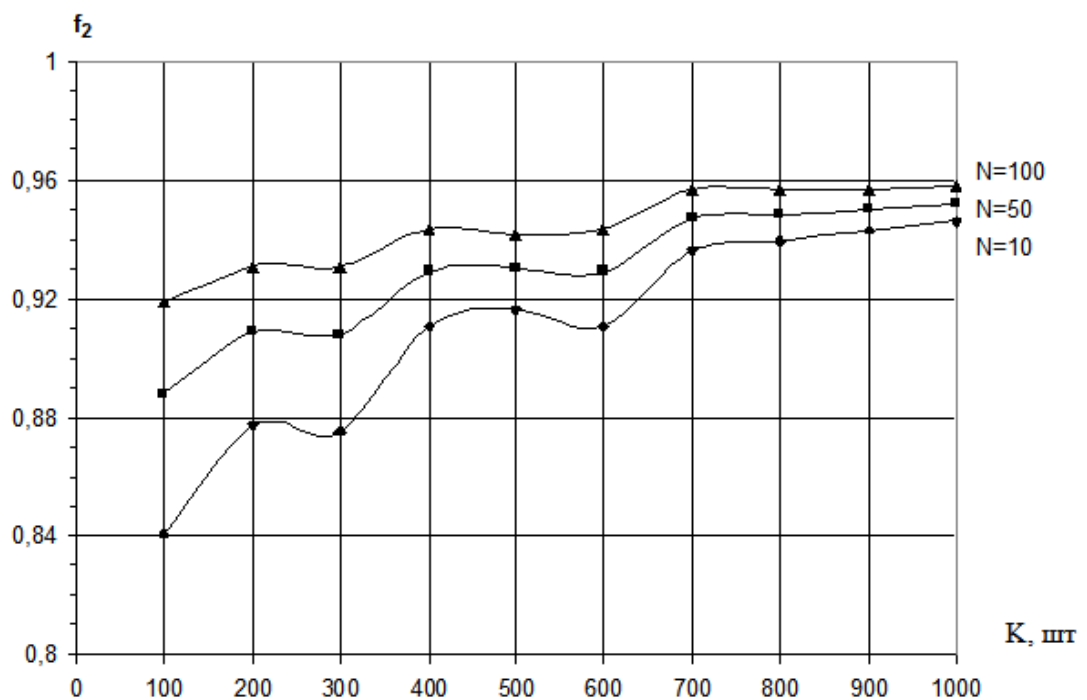


Рисунок 4.11. - Порівняння загальних апаратних витрат в схемах ПМК з ідентифікацією виходів та ПМК (1) з загальною пам'яттю при $p_1=0.95$; $p_3=0.1$

4.3.3. Пристрій мікропрограмного керування з адресацією мікрокоманд

Структура матричної моделі ПМК з адресацією мікрокоманд показана на рис. 4.12.

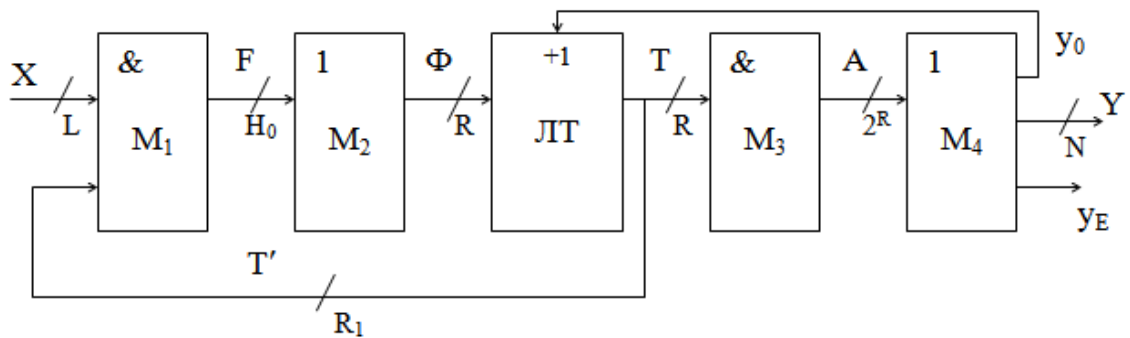


Рисунок 4.12. - Матрична модель ПМК з адресацією мікрокоманд

Матрична реалізація ПМК з адресацією мікрокоманд аналогічна до матричної реалізації ПМК з ідентифікацією виходів. Різниця полягає в тому, що кон'юнктивна матриця M_1 реалізує H_0 термів $F_1, \dots, F_{H_0}$ прямої структурованої таблиці автомата адресації (автомат Мілі).

Результати дослідження функції f_3 представлені на рис. 4.13-4.14.

ПМК з адресацією мікрокоманд завжди призводить до стабільного зниження апаратних витрат. У міру зростання кількості вершин ГСА економія ресурсів становить від 60 до 35%.

Дослідження функції f_4 при значеннях параметрів, що дають максимальне зменшення апаратних витрат, представлено на рис. 4.15.

Як видно із рис. 4.15 наявність КП істотно впливає (знижує в середньому на 15%) на загальну економію ресурсів від застосування оптимальної адресації мікрокоманд. Зменшення ефективності із зростанням кількості мікрооперацій, трохи залежить від загальної кількості вершин граф-схеми алгоритму.

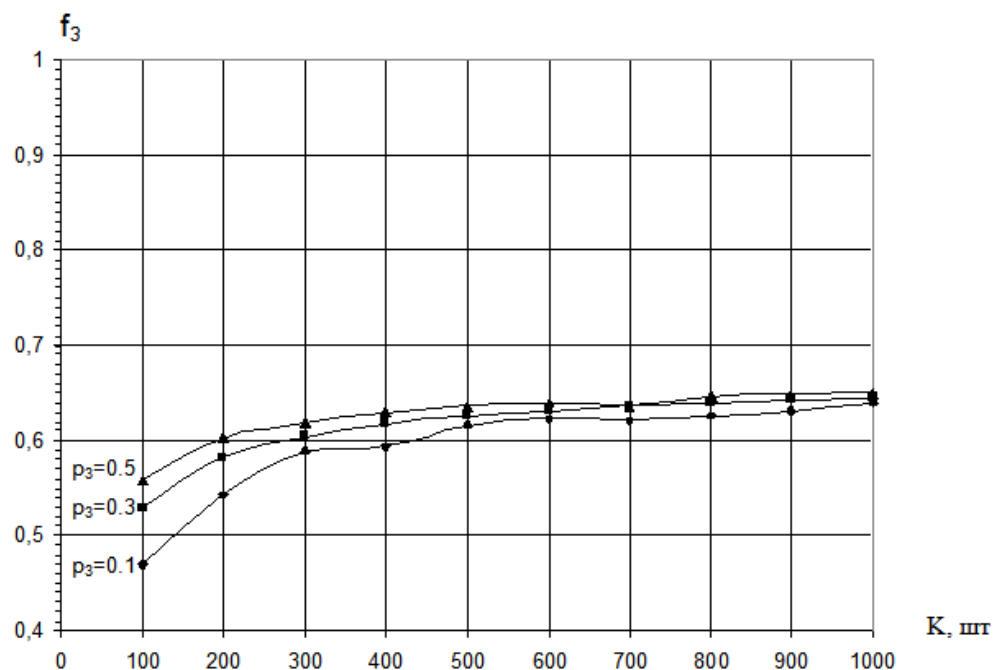


Рисунок 4.13. - Порівняння апаратних витрат в схемах автомата адресації ПМК з адресацією мікрокоманд та ПМК (1) з загальною пам'яттю при $p_1=0.85$; $p_5=0.4$

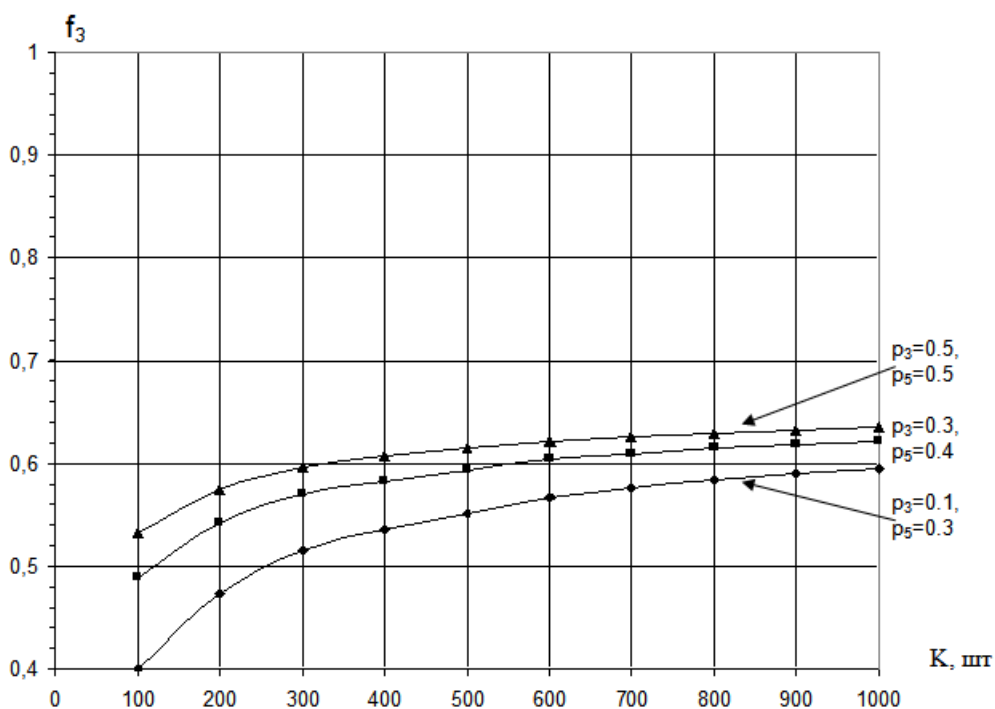


Рисунок 4.14. - Порівняння апаратних витрат в схемах автомата адресації ПМК з адресацією мікрокоманд та ПМК (1) з загальною пам'яттю при $p_1=0.95$

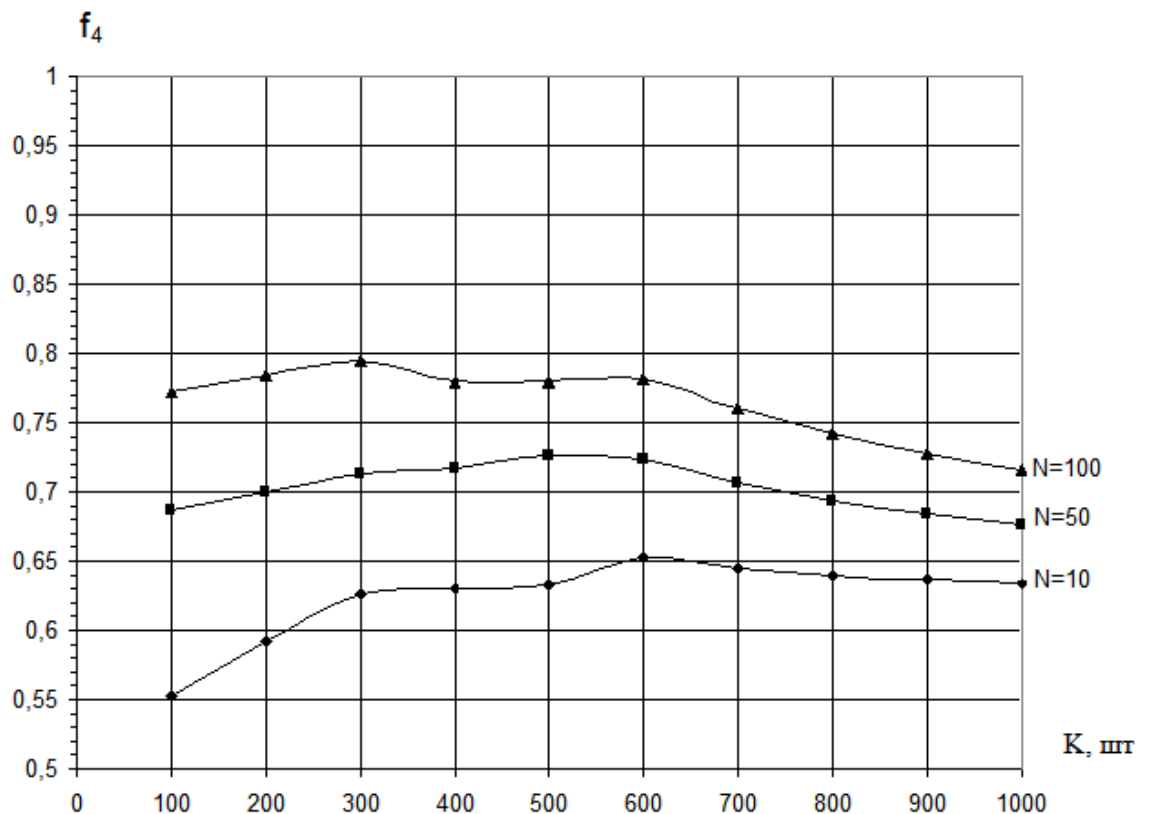


Рисунок 4.15. - Порівняння загальних апаратних витрат в схемах ПМК з адресацією мікрокоманд та ПМК (1) з загальною пам'яттю при $p_1=0.95$; $p_3=0.1$; $p_5=0.3$

4.4. Висновки

1. Даний метод оптимізації ПМК за допомогою ідентифікації виходів дає можливість зменшити кількість аргументів функцій, які реалізуються на логічних LUT-елементах, схеми формування адреси мікрокоманди, від $L+R_1$ до ПМК (4) до $U+R_1$ до ПМК (6).

При цьому час такту ПМК (4) та ПМК (6) не збігається, тобто оптимізація за апаратними витратами іноді може призводити до втрати продуктивності. Для зниження (зменшення) значення коефіцієнта η надалі можна використовувати перетворення кодів зворотного зв'язку на коди логічних умов.

2. Застосований у роботі метод елементаризації ОЛЛ з поділом кодів та оптимальним кодуванням ОЛЛ дозволяє при виконанні умови $R_{13} < R_{11}$

зменшити кількість змінних зворотнього зв'язку, які необхідні для реалізації системи функцій, що формують код наступного ОЛЛ ГСА.

Застосування методу не потребує застосування додаткового регістру, оскільки для його реалізації достатньо використовувати тригери, що є у складі PLB. У той же час відбувається зменшення розрядності ЛТ, який використовують тільки для адресації компонентів ОЛЛ. При цьому час такту ПМК (5) та ПМК (7) однаково збігаються.

3. Використання ПМК з ідентифікацією виходів дозволяє зменшити апаратні витрати приблизно на 20% у порівнянні з ПМК (1) з загальною пам'яттю.

4. Застосування оптимальної адресації мікрокоманд (в залежності від параметрів граф-схеми алгоритму) може дати можливість знизити апаратні витрати у схемі ПМК майже на 60%, ніж ПМК (1) з загальною пам'яттю.

ВИСНОВКИ

В ході виконання бакалаврської роботи було отримано такі результати:

1. Здійснено аналіз сучасних програмованих логічних інтегральних схем, побудованих за технологією FPGA, зокрема, виділено їх переваги та недоліки.
2. В результаті аналізу ПЛІС було визначено існуючі методи синтезу та оптимізації ПМК, які доцільно застосовувати для реалізації пристроїв керування. Використання відомих методів оптимізації ПМК мають певні обмеження, які впливають на ефективність їх застосування для ПЛІС з технологією FPGA.
3. Розроблено та побудовано структури ПМК з ідентифікацією виходів та спеціальною адресацією мікрокоманд, які дозволяють зменшити апаратні витрати автомата адресації та підвищити загальну продуктивність ПМК.
4. Розроблено та побудовано структури ПМК з оптимальною адресацією мікрокоманд, яким не потрібні додаткові ресурси кристалу ПЛІС, а також відбувається зменшення кількості логічних LUT-елементів, які використовують у схемі автомата адресації.
5. Теоретичні розрахунки показали, що використання ПМК з ідентифікацією виходів дозволяє знизити апаратні витрати до 20% у порівнянні з ПМК з загальною пам'яттю. Крім того, застосування оптимальної адресації мікрокоманд (в залежності від параметрів граф-схеми алгоритму) може дати можливість зменшити апаратні витрати у схемі ПМК до 60%, ніж ПМК з загальною пам'яттю.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Аналіз, синтез і проектування цифрових систем керування : навч. посібник / С. М. Єсаулов, О. Ф. Бабічева; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2018. – 150 с.
2. АРХІТЕКТУРА ТА МЕТОДИ СИНТЕЗУ ПЛІС ПРИСТРОЇВ. *eNTUKhPIIR* :: Репозитарій НТУ 'ХПІ' :: Головна. URL: <https://repository.kpi.kharkov.ua/server/api/core/bitstreams/7abb5092-572e-4bd3-8742-6f4e4356cd15/content>.
3. Баркалов, О., Тітаренко, Л., Головін, О., Матвієнко, О., & Сабурова, С. (2022). МЕТОД ЗМЕНШЕННЯ ЧИСЛА ЕЛЕМЕНТІВ LUT В СХЕМІ МІКРОПРОГРАМНОГО АВТОМАТУ МУРА. *Science-based technologies*, 53(1), 13–21. <https://doi.org/10.18372/2310-5461.53.16402>
4. Жабін В. І., Жуков І. А., Клименко І. А., Ткаченко В. В. Прикладна теорія цифрових автоматів: Навч. посібник. — К.: Книжкове вид-во НАУ, 2007. — 364 с.
5. Кочубей О.О. Прикладна теорія цифрових автоматів. Логічні основи : Навч. посіб./ О.О. Кочубей, О.В. Сопільник. - Д.: РВВ ДНУ; Вид-во ДНУ, 2009. - 264 с.
6. Кузьменко, І. (б. д.). *Теорія графів*. ELAKPI: Home. https://ela.kpi.ua/bitstream/123456789/35854/1/Teoriia_hrafiiv.pdf
7. Луценко А. В. Огляд основних виробників ПЛІС в світі / А. В. Луценко, О. В. Воргуль // Радиоелектроника и молодежь в XXI веке : материалы 25-го Междунар. молодеж. форума, 20–22 апр. 2021 г. – Харьков : ХНУРЭ, 2021. – Т. 3. – С. 197–198.
<https://openarchive.nure.ua/handle/document/15450>
8. *Мікроконтролери ATMEL - radiodetali.com.ua/ua*. (б. д.). Інтернет-крамниця радіотехніки та компонентів - Радіодеталі. <https://radiodetali.com.ua/ua/catalog/mikroshemy/mikrokontrollery-atmel>

9. Мікропроцесорні пристрої: навч. посібник для студентів зі спец-ті «Електроніка» / Т. О. Терещенко, В. А. Тодоренко, Л. М. Батрак, Ю. С. Ямненко. – К.: Кафедра, 2017. – 244 с. ISBN 978-617-7301-37-9
10. *Основи технічної електроніки: Т. 2. Схемотехніка/Бойко В.І., Гуржій А.М., Жуйков В.Я., Зорі А.А., В.М. Співак та ін. (2007). Вища школа.*
11. Принцип мікропрограмного керування та автоматизація проектування операційних пристроїв. I / А. М. Петрушенко // Control systems & computers. — 2020. — № 1. — С. 3-22.
12. Проектування спеціалізованих комп'ютерних систем на ПЛІС. Лабораторний практикум. [Електронний ресурс] : навч. посіб. для студ. спеціальності 123 Комп'ютерна інженерія / КПІ ім. Ігоря Сікорського ; Я.М. Клятченко, О.В. Тарасенко-Клятченко, О.К. Тесленко. – Електронні текстові дані (1 файл: 2,3 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2021. – 54 с.
13. Рябенський В. М., Жуйков В. Я., Гулий В. Д. Цифрова схемотехніка: Навч. посібник. — Львів: "Новий Світ-2000", 2009. — 736 с.
14. Ситнік Б. Т. Комп'ютерні системи керування : навч. посіб. Ч. 1. Моделювання систем. Харків : УкрДУЗТ, 2019. 182 с.
15. Ситнік Б. Т. Комп'ютерні системи керування : навч. посіб. Ч. 2. Цифрові комп'ютерні системи керування. Харків : УкрДУЗТ, 2019. 141 с.
16. *Сімейства MAX CPLD компанії Altera — Цифрова лабораторія FPGA / DSP.* (б. д.). Цифрова лабораторія FPGA / DSP — Інформаційно-навчальний проект для цифрової електроніки. <http://www.fpga.keoa.kpi.ua/fpga/pld-architecture/сімейства-max-cpld-компанії-altera.html>
17. *Сімейство мікроконтролерів STM32. Особливості та характеристики — Цифрова лабораторія FPGA / DSP.* (б. д.). Цифрова лабораторія FPGA / DSP — Інформаційно-навчальний проект для цифрової електроніки. <http://www.fpga.keoa.kpi.ua/other/сімейство-мікроконтролерів-stm32-особли.html>
18. Схемотехніка електронних систем: У 3кн. Кн.2. Цифрова схемотехніка: Підручник/ Бойко В.І., Гуржій А.М. Багрій В.В. та інші. – 2-ге вид., допов.

переробл.- К.: Вища школа, 2004. – 423с.

19. Технологія розробки систем на кристалі. *kanyevsky.kpi.ua*.
URL: <https://kanyevsky.kpi.ua/підручник-vhdl/технологія-розробки-систем-на-криста/>.
20. Толбатов, А.В. Методологія створення автоматизованих систем керування [Текст] / А.В. Толбатов, В.Д. Черв'яков, Т.Л. Щербак // Вісник Сумського державного університету. Серія Технічні науки. — 2005. — №9(81). — С. 124-130. <http://essuir.sumdu.edu.ua/handle/123456789/471>
21. Що таке ПЛІС: поняття, визначення, правила програмування та основи для початківців. *What – маємо відповідь*. URL: <https://what.com.ua/sho-take-plis-poniattia-vizn/>.
22. *Barkalov A., Titarenko L.* Logic synthesis for compositional microprogram control units. — Berlin: Springer, 2008. — 272 pp.
23. *Barkalov A., Titarenko L.* Synthesis of operational and control automata. — Donetsk: DonNTU, 2009. — 256 pp.
24. Barkalov, O. O. (2014). *Rozrobka i doslidzhennia metodiv syntezu mikroprohramnykh avtomativ z peretvorenniam kodiv ob"iektiv* [Neopubl. avtoref. dys. kand. tekhn. nauk]. Derzh. ekon.-tekhnol. un-t transp.
25. Boit, Christian & Schlangen, Rudolf & Kerst, Uwe & Lundquist, T.R. (2008). Physical Techniques for Chip-Backside IC Debug in Nanotechnologies. Design & Test of Computers, IEEE. 25. 250 - 257. 10.1109/MDT.2008.82. https://www.researchgate.net/publication/3251629_Physical_Techniques_for_Chip-Backside_IC_Debug_in_Nanotechnologies
26. FPGA Applications - HardwareBee. *HardwareBee*. URL: <https://hardwarebee.com/fpga-common-applications/#:~:text=Main%20FPGA%20applications%20are:%20Medical,both%20wired%20and%20wireless%20communications.>
27. HAVRILOV, D. V., VOLOVYK, A. Y., HAVRILOVA, N. M., KOFANOVA, N. V., & YAROVYI, D. V. (2019). UNIVERSAL REGISTER ON FPGA. *HERALD of Khmelnytskyi national university*, 271(2), 188–

191. <https://doi.org/10.31891/2307-5732-2019-271-2-188-191>
28. Logic Synthesis and Optimization Benchmarks User Guide Version 3.0 | Semantic Scholar. *Semantic Scholar | AI-Powered Research Tool*. URL: <https://www.semanticscholar.org/paper/Logic-Synthesis-and-Optimization-Benchmarks-User-Yang/4cd302de6b2e911c980c9d5fcc009e8215c2c560>.
29. Srinivasan, Suresh & Krishnan, Ramakrishnan & Mangalagiri, Prasanth & Xie, Yuan & Narayanan, Vijaykrishnan & Irwin, Mary & Sarpatwari, Karthik. (2008). Toward increasing FPGA lifetime. Dependable and Secure Computing, IEEE Transactions on. 5. 115 - 127. 10.1109/TDSC.2007.70235. https://www.researchgate.net/publication/3449471_Toward_increasing_FPGA_lifetime
30. UNESCO - *ENCYCLOPEDIA OF LIFE SUPPORT SYSTEMS (EOLSS)*. URL: <https://www.eolss.net/sample-chapters/c18/E6-43-04.pdf>.
31. VHDL як алгоритмічна мова для опису паралельних алгоритмів. *kanyevsky.kpi.ua*. URL: <https://kanyevsky.kpi.ua/статті/vhdl-як-алгоритмічна-мова-для-опису-парал/>.
32. What Is FPGA (Field-Programmable Gate Array)? Meaning, Working, and Uses - Spiceworks. *Spiceworks*. URL: <https://www.spiceworks.com/tech/networking/articles/what-is-fpga/>.