Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки

До захисту допущено
Кафедрою комп'ютерних систем та робототехніки
протокол № __ від __ червня 2026 р.

завідувач кафедри _____ Максим ХРУСЛОВ
(підпис)
«___»_____ 2026 р.

Кваліфікаційна робота

здобувача першого (бакалаврського) рівня вищої освіти

«НАВЧАЛЬНА МОДЕЛЬ ЦИФРОВОГО ПРОЦЕСОРА НА ОСНОВІ ПРОГРАМОВАНОЇ ЛОГІЧНОЇ МАТРИЦІ»

Спеціальність ***123 – Комп'ютерна інженерія.***

Освітня програма ***Комп'ютерна інженерія***

Виконавець _____ **Катерина КОВИЛКІНА**
(підпис)

Науковий керівник _____ **Сергій РЕВА**
(підпис)

Харків – 2026

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел і чотирьох додатків. Загальний обсяг роботи складає 66 сторінки, із яких 47 сторінок основної частини з 12 рисунками, 13 таблицями, 21 найменувань у списку використаних джерел та 4 додатків.

Мета кваліфікаційної роботи полягає у вдосконалені навчальної моделі 8-бітного процесора DPM08 шляхом розширення функціональних можливостей та використання технології програмованих логічних матриць.

Об'єкт дослідження — архітектурні підходи та технологічні методи створення цифрових процесорів.

Предмет дослідження — навчальна модель цифрового процесора з архітектурою Фон-Неймана.

Методи дослідження об'єднують структурне проектування цифрових автоматів, архітектурного синтез обчислювальних пристроїв та функціональну симуляцію на рівні RTL у середовищі MAX+PLUS II.

В межах дослідження реалізовано модель процесора з регістровим файлом із шістьма регістрами загального призначення, автоматом керування з жорсткою логікою та восьмитактним машинним циклом, комбінаційним АЛП з вісьмома операціями та флагами ZF і CF, блоком портів введення-виведення та налагоджувальним інтерфейсом. Розроблено систему команд у чотирьох функціональних блоках; тестові програми підготовлено засобами асемблера TASM. Функціональна симуляція підтвердила коректність реалізації.

Результати рекомендовано використовувати у навчальному процесі для вивчення архітектурних принципів процесорів та програмування мовою асемблера, а також як базову платформу для подальшого розширення процесорного ядра.

Ключові слова: комп'ютерна інженерія, цифровий процесор, програмована логічна матриця, архітектура фон неймана, система команд, автомат керування, арифметико-логічний пристрій, мікроконтролер, Maxplus II.

ABSTRACT

The Bachelor's thesis explanatory note consists of an introduction, three chapters, conclusions, a list of references, and four appendices. The total volume of the thesis is 66 pages, including 47 pages of the main text featuring 12 figures, 13 tables, 21 references, and 4 appendices.

The aim of the research is to improve the educational model of the 8-bit DPM08 processor by expanding its functionality and utilizing programmable logic array technology.

The object of the research is architectural approaches and technological methods for creating digital processors.

The subject of the research is an educational model of a digital processor based on the Von Neumann architecture.

The research methods combine structural design of digital automata, architectural synthesis of computing devices, and RTL functional simulation within the MAX+PLUS II environment.

Within the scope of the study, a processor model has been implemented, featuring a register file with six general-purpose registers, a hardwired control unit with an eight-clock machine cycle, a combinational ALU supporting eight operations along with ZF and CF flags, an I/O port block, and a debugging interface. An instruction set architecture (ISA) has been developed across four functional blocks; test programs were prepared using the TASM assembler. Functional simulation has verified the correctness of the implementation.

The results are recommended for use in the educational process to study the architectural principles of processors and assembly language programming, as well as a base platform for further expansion of the processor core.

Keywords: computer engineering, digital processor, programmable logic array, Von Neumann architecture, instruction set, control unit, arithmetic logic unit, microcontroller, Maxplus II.

ЗМІСТ

ЗМІСТ 4
ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ 6
ВСТУП 8
РОЗДІЛ 1 АНАЛІЗ ІСНУЮЧИХ НАВЧАЛЬНИХ МОДЕЛЕЙ ЦИФРОВИХ ПРОЦЕСОРІВ ТА ОБҐРУНТУВАННЯ ВИБОРУ ПЛМ 10
1.1. Постановка задачі. Об'єкт, предмет та мета дослідження 10
1.2. Аналіз архітектур існуючих навчальних процесорів 11
1.3. Обґрунтування вибору ПЛМ та середовища MaxPlus II 13
Висновки за розділом 1 14
РОЗДІЛ 2 СИНТЕЗ ТА АРХІТЕКТУРНА РЕАЛІЗАЦІЯ НАВЧАЛЬНОЇ МОДЕЛІ ЦИФРОВОГО ПРОЦЕСОРА DPM081 НА БАЗІ ПРОГРАМОВАНОЇ ЛОГІЧНОЇ МАТРИЦІ 15
2.1. Обґрунтування вибору технологічної бази ПЛМ для розробки обчислювальних структур 15
2.2. Архітектура та внутрішня регістрова модель процесора DPM081 . 16
2.3. Структура командного слова та формат кодування інструкцій 20
2.4. Реалізація принципів архітектури фон Неймана та циклу керування процесора 23
2.4.1. Реалізація принципів фон Неймана в архітектурі процесора ... 23
2.4.2. Автомат керування з жорсткою логікою та часова структура машинного циклу 24
2.4.3. Аналіз системних сигналів скидання та синхронізації 25
2.4.4. Специфікація інтерфейсних сигналів пристрою керування 26
2.5. Функціональні блоки: Арифметико-логічний пристрій та флаги ... 27
Арифметико-логічний пристрій (АЛП) 27
2.6. Специфікація та реалізація команд введення-виведення даних 28
2.7. Аналіз реалізації та функціонування налагоджувального блоку процесора 29

2.8. Інтерфейсна взаємодія з мікроконтролером STM32 30
Висновки за розділом 2 31
РОЗДІЛ 3 ТЕСТУВАННЯ ТА ОЦІНКА РЕЗУЛЬТАТІВ РОЗРОБЛЕННЯ НАВЧАЛЬНОЇ МОДЕЛІ ЦИФРОВОГО ПРОЦЕСОРА 32
3.1 Середовище симуляції MAX+PLUS II як інструмент верифікації .. 32
3.2 Верифікація ізольованих компонентів процесора 33
3.2.1 Арифметико-логічний пристрій (ALU) 33
3.2.2. Двонаправлений регістр-лічильник (UDC8B) 33
3.2.3. Модуль оперативної пам'яті (RAM8B) 35
3.2.4. Дешифратор інструкцій — блок MOV 35
3.2.5. Дешифратор інструкцій — блок LD/CMP 36
3.2.6. Пристрій керування (CPUTG) 37
3.3. Системна верифікація процесора за функціональними групами команд 39
3.3.1. Команди пересилання даних (MOV) 39
3.3.2. Команди інкременту та декременту (INC/DEC) 40
3.3.3. Команди завантаження константи та порівняння (LD/CMP) ... 41
3.3.4. Арифметико-логічні операції у складі процесора (CHECKALU) 42
3.3.5. Команди введення-виведення (IN/OUT) 44
3.4. Верифікація налагоджувального інтерфейсу 45
3.5. Зведена оцінка результатів та відповідність меті роботи 47
Висновки за розділом 3 48
ВИСНОВКИ 49
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 51
ДОДАТКИ 53

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

A[7..0] – Address bus — зовнішня/внутрішня шина адреси

АЛП – Арифметико-логічний пристрій

ALU – Arithmetic/Logic Unit — міжнародне позначення АЛП

БГКС – Блок генератора керуючих сигналів — автомат керування

CF – Carry Flag — флаг перенесення/переповнення

CLK – Clock — вхідний сигнал синхронізації (тактова частота)

D[7..0] – Data bus — основна шина даних

DC[3..0] – Debug Control — вхідна 4-бітна шина керування налагодженням

DEBRM – Debug Read Memory — сигнал примусового читання SRAM у режимі відладки

DEBWIC – Debug Write Instruction Counter — сигнал завантаження адреси у лічильник команд

DEBWM – Debug Write Memory — сигнал прямого запису байта у SRAM

DP[7..0] – Двоспрямована шина секції налагодження, ізольована від системних шин

DPM081 – Digital Processor Model 081 — навчальна модель цифрового процесора

EIC – Enable Instruction Counter — сигнал дозволу зміни стану лічильника команд

GPR – General Purpose Registers — регістри загального призначення (РЗП)

IC – Instruction Counter — лічильник команд

IR – Instruction Register — регістр команд

NT[2..0] – Вихідна 3-бітна шина станів поточного такту машинного циклу

PI[7..0] – Port Input — вхідна 8-бітна паралельна лінія введення

ПЛМ – Програмована логічна матриця

PLA – Programmable Logic Array — програмована логічна матриця

PO[7..0] – Port Output — вихідна 8-бітна паралельна лінія виведення

RC – Read Command — строб фази вибірки командного слова

RD – Read Data — сигнал фази читання операнда

RM – Read Memory — сигнал читання зовнішньої SRAM

RST – Reset — стабілізований сигнал скидання процесора

RSTI – Reset Input — вхідний сигнал асинхронного скидання

РЗП – Регістри загального призначення (A, B, C, D, E, S)

RTL – Register Transfer Level — рівень регістрових передач

SRAM – Static Random-Access Memory — статична оперативна пам'ять

TASM – Telemark Assembler — асемблер для підготовки тестових програм

USB-CDC – Universal Serial Bus — Communication Device Class — протокол зв'язку налагоджувального інтерфейсу

WC – Write Command — сигнал запису коду операції до регістра команд IR

WD – Write Data — сигнал фази запису результату обчислень

WM – Write Memory — сигнал запису у зовнішню SRAM

ZF – Zero Flag — флаг нульового результату АЛП

ВСТУП

Підготовка фахівців у галузі комп'ютерної інженерії сьогодні передбачає не лише теоретичне опанування принципів роботи цифрових обчислювальних пристроїв, але й практичне розуміння того, як техніка виконує програмний код на рівні логічних сигналів і тактів. Такий рівень розуміння неможливо сформувати тільки через програмну емуляцію або вивчення документації до промислових та комерційних мікропроцесорів — він вимагає роботи з реальною апаратною моделлю, де кожен сигнал і кожна стадія машинного циклу є наглядними і зрозумілими.

На кафедрі ХНУ імені В.Н. Каразіна для цих цілей була розроблена апаратна навчальна модель процесора DPM08 — 8-бітний процесор архітектури фон Неймана, реалізований на дискретній TTL-логіці серії К555/74LS у вигляді модульного стенду. Модель відрізняється простотою і наочністю: студент бачить кожен сигнал на фізичних провідниках, а виконання кожної інструкції відображається на світлодіодних індикаторах. Проте апаратна реалізація на дискретній логіці має принципове обмеження — будь-яке розширення архітектури вимагає фізичного перемонтажу плат, а пам'ять програм виконана виключно на ROM без можливості запису під час виконання.

Вирішення цієї проблеми полягає у перенесенні архітектури DPM08 на програмовану логічну матрицю (ПЛМ). Синтезована у кристалі схема зберігає апаратний паралелізм і фізичну достовірність оригіналу, але стає повністю на рівні логічного опису. Це дозволяє розширити систему команд, додати оперативну пам'ять із записом, апаратні порти введення-виведення та налагоджувальний інтерфейс — без жодного фізичного втручання у схему.

Актуальність роботи обумовлена потребою у сучасній навчальній платформі, яка поєднує апаратну автентичність з гнучкістю програмного редагування архітектури та підтримкою сучасних інструментів налагодження.

Об'єктом дослідження є процес реалізації архітектури цифрового процесора засобами логіки програмованої логічної матриці.

Предметом дослідження є методи архітектури синтезу обчислювального ядра, системи команд та налагоджувального інтерфейсу у середовищі MaxPlus II.

Метою роботи є вдосконаленя навчальної моделі 8-бітного процесора DPM08 шляхом розширення функціональних можливостей та використання технології програмованих логічних матриць.

Для досягнення мети вирішено такі завдання: виконано аналіз архітектур існуючих навчальних процесорів та попередньої моделі DPM08; розроблено вдосконалену систему команд та повну архітектуру процесора; реалізовано цифрову схему усіх функціональних блоків у середовищі MaxPlus II; проведено верифікацію моделі на рівні окремих компонентів та на рівні повного ядра шляхом функціональної симуляції типових тестових програм.

Практичне значення роботи полягає у тому, що розроблена модель може бути безпосередньо використана у навчальному процесі для вивчення архітектурних принципів побудови ЕОМ, засвоєння роботи автомата керування та практичного програмування мовою асемблера із застосуванням налагоджувального інтерфейсу на основі мікроконтролера STM32.

Робота складається зі вступу, трьох розділів, висновків, переліку використаних джерел та додатків. У першому розділі виконано аналіз аналогів і обґрунтовано вибір ПЛМ. У другому розділі описано розробку повної архітектуру процесора DPM081. У третьому розділі наведено результати верифікації усіх компонентів і функціональних груп команд.

РОЗДІЛ 1
АНАЛІЗ ІСНУЮЧИХ НАВЧАЛЬНИХ МОДЕЛЕЙ ЦИФРОВИХ ПРОЦЕСОРІВ ТА ОБҐРУНТУВАННЯ ВИБОРУ ПЛМ

1.1. Постановка задачі. Об'єкт, предмет та мета дослідження

Вивчення архітектурних принципів побудови цифрових обчислювальних пристроїв є невід'ємною складовою підготовки фахівців у галузі комп'ютерних наук та електронної інженерії. Водночас більшість навчальних моделей процесорів або надмірно складні для реалізації та розуміння, або настільки спрощені, що не придатні для практичного програмування. Наявна апаратна модель DPM08, розроблена на кафедрі ХНУ імені В.Н. Каразіна, поєднує просту архітектуру з апаратною наочністю, однак має принципове обмеження: реалізована на дискретній TTL-логіці, вона не допускає модифікації без фізичного перемонтажу плат [1].

Предметом дослідження є методи архітектури реалізації обчислювального ядра, системи команд та налагоджувального інтерфейсу засобами програмованої логіки.

Об'єктом дослідження є процес реалізації архітектури цифрового процесора засобами логіки програмованої логічної матриці.

Метою роботи є вдосконаленя навчальної моделі 8-бітного процесора’ DPM08 шляхом розширення функціональних можливостей та використання технології програмованих логічних матриць.

Для досягнення поставленої мети визначено такі завдання:

1. виконати аналіз архітектур існуючих навчальних процесорів та виявити їх недоліки;
2. виконати аналіз архітектури та принципів функціонування попередньої моделі DPM08 і визначити можливості її вдосконалення;
3. розробити вдосконалену систему команд та архітектуру нової моделі процесора;
4. реалізувати цифрову схему процесора в середовищі MaxPlus II;

5. провести тестування моделі на типових наборах команд (арифметичні, логічні, робота з пам'яттю).

1.2. Аналіз архітектур існуючих навчальних процесорів

Для обґрунтування доцільності розробки нової моделі розглянуто три процесорні архітектури, що широко вивчаються у навчальних курсах з цифрової електроніки та вбудованих систем: Intel 8080, MOS 6502 та Intel 8051, а також попередню апаратну модель DPM08.

Intel 8080. Процесор Intel 8080, випущений у 1974 році, є класичним 8-бітним мікропроцесором із сімома регістрами загального призначення (A, B, C, D, E, H, L), 16-бітною адресною шиною та набором з 76 базових інструкцій [5]. Для повноцінного функціонування вимагає допоміжних мікросхем — тактового генератора 8224 та системного контролера 8228. Адресна і шина даних є мультиплексованою, що суттєво ускладнює схемотехніку при реалізації з нуля. З навчальної точки зору архітектура Intel 8080 надмірно складна: велика кількість інструкцій і необхідність допоміжної периферії виходять за межі цілей вступних курсів.

MOS 6502. Процесор MOS 6502, представлений у 1975 році, відзначається мінімальною кількістю регістрів: акумулятор A та два індексних регістри X і Y [7]. Система команд налічує 151 опкод із розвиненою системою режимів адресації — безпосередня, нульова сторінка, абсолютна, індексована та непряма. Саме складна система адресації, де нульова сторінка пам'яті фактично виконує роль розширеного регістрового файлу, робить архітектуру неочевидною для первинного засвоєння принципів роботи процесора.

Intel 8051. Мікроконтролер Intel 8051, представлений у 1981 році, побудований за Гарвардською архітектурою з роздільними просторами програмної та даних пам'яті [9]. Містить акумулятор A, регістр B та чотири банки по вісім регістрів загального призначення, доступних через спеціальні функціональні регістри (SFR). До складу ядра інтегровано два 16-бітних таймери, послідовний порт UART та чотири 8-бітних порти введення-виведення. Система команд налічує 255 інструкцій. Попри орієнтованість на керуючі

застосування, надлишкова периферія, Гарвардська шина та система SFR роблять реалізацію з нуля надмірно складною для навчального проекту.

Разом з тим архітектура 8051 є найближчою до задачі даної роботи серед розглянутих аналогів: обидва пристрої орієнтовані на керуючі застосування, мають 8-бітну розрядність, апаратні порти введення-виведення та підтримують умовні переходи. Саме тому при проектуванні нової моделі DPM081 враховано досвід 8051 у частині організації портів введення-виведення та структури умовних переходів, проте без запозичення його надлишкової складності.

Попередня модель є DPM08. Навчальний процесор DPM08, розроблений на кафедрі ХНУ імені В.Н. Каразіна, являє собою 8-бітний процесор архітектури фон Неймана на дискретній TTL-логіці (К555/74LS) [1]. Містить чотири 8-бітних регістри загального призначення (A, B, C, D), комбінаційний АЛП з вісьмома функціями, 8-тактний машинний цикл та систему з 54 інструкцій. Архітектура є простою і наочною, однак має два принципових недоліки: реалізація на фізичних платах унеможливлює модифікацію архітектури без перемонтажу схеми, а пам'ять програм є виключно ROM — запис даних у пам'ять під час виконання програми не підтримується.

Машинний цикл DPM08 складається з восьми тактів і реалізований апаратом керування з жорсткою логікою. На першому такті генерується сигнал зчитування з пам'яті програм (RM), на другому — запис інструкції до регістра команд (WC). На тактах 3–4 дешифратор аналізує код команди і, у разі двобайтової інструкції, формує інкремент лічильника команд для зчитування другого байта — операнда. На тактах 5–6 виконується власне пересилання: сигнал ED виставляє дані на шину, SC фіксує їх у регістрі-приймачі. Такти 7–8 завершують цикл фінальним інкрементом лічильника команд [1]. Структура командного слова є 8-бітною і розподіляється на три частини: біти CM7–CM6 визначають тип команди (0–3), біти CM5–CM3 — адресу джерела даних або номер функції АЛП, біти CM2–CM1 — адресу регістра-приймача. Така організація командного слова є безпосередньою основою для системи команд нової моделі DPM081.

У таблиці 1.1 табличне порівняння описаних вище процесорів.

Таблиця 1.1

Порівняльний аналіз навчальних процесорних архітектур

Критерій	Intel 8080	MOS 6502	Intel 8051	DPM08
Рік / тип	1974, МП	1975, МП	1981, МК	2011, навч. макет
Розрядність	8 біт	8 біт	8 біт	8 біт
Регістри	7-РЗП (A,B,C,D,E,H,L)	3 (A, X, Y)	A, B; 4 банки × 8 рег. через SFR	4 РЗП (A, B, C, D)
Кількість інструкцій	76	151	255	54
Адресний простір	64 Кбайт	64 Кбайт	64К код + 64К дані (Harvard)	256 байт ROM
Запис у пам'ять	Так	Так	Так (RAM)	Ні (тільки ROM)
Реконфігурованість	Ні	Ні	Ні	Ні
Придатність для реалізації з нуля	Ні — складна шина і допоміжні ІС	Ні — неочевидна адресація	Ні — SFR, Harvard, периферія	Так — проста архітектура
Налагоджувальний інтерфейс	Відсутній	Відсутній	Обмежений (UART)	Відсутній
Основний недолік для навчання	Складна мультиплексована шина	Неочевидна система адресації	Надлишкова периферія та Harvard	Нереконфігурований макет; відсутній запис у пам'ять

1.3. Обґрунтування вибору ПЛМ та середовища MaxPlus II

Виконаний аналіз показав, що Intel 8080, MOS 6502 та Intel 8051 є надмірно складними для реалізації з нуля в межах навчального проекту. Модель DPM-8, попри просту і зрозумілу архітектуру, має два принципових обмеження: нереконфігурованість апаратного макету та відсутність підтримки запису у пам'ять. Саме ці недоліки визначають вибір програмованої логічної матриці як технологічної бази для нової моделі.

Реалізація на ПЛМ вирішує обидві проблеми одночасно. По-перше, синтезована у матриці схема є справжнім апаратним автоматом — дешифратор, АЛП та регістри функціонують паралельно, що зберігає апаратну достовірність оригінальної DPM08. По-друге, будь-яке розширення архітектури — додавання регістрів, нових команд або блоку RAM з можливістю запису — виконується шляхом зміни логічного опису без фізичного втручання у схему [4].

Середовище MaxPlus II (Multiple Array Matrix and Programmable Logic User System) компанії Altera обрано як інструментальну базу. Воно забезпечує повний цикл від введення логічних рівнянь до функціональної симуляції у вбудованому Waveform Editor, що дозволяє перевірити роботу кожного вузла процесора без фізичного кристала [4].

Вибір MaxPlus II і ПЛМ середньої щільності (сімейство MAX V) замість сучасних FPGA-інструментів (Quartus Prime, Vivado) обумовлений навчальною спрямованістю роботи. По-перше, MaxPlus II дозволяє вводити схему безпосередньо у вигляді логічних рівнянь або принципової схеми — без необхідності знання мов HDL (VHDL, Verilog), що є важливим для вступного курсу. По-друге, синтезована схема повністю відповідає введеному опису без автоматичних оптимізацій, що робить зв'язок між логічним описом і апаратною реалізацією прозорим[4].

Висновки за розділом 1

У першому розділі сформульовано мету та завдання роботи, визначено об'єкт і предмет дослідження. Виконано порівняльний аналіз трьох процесорних архітектур (Intel 8080, MOS 6502, Intel 8051) та апаратної навчальної моделі DPM08. Встановлено, що Intel 8080 потребує допоміжних мікросхем і має складну мультиплексовану шину; MOS 6502 має неочевидну систему адресації; Intel 8051 містить надлишкову периферію та Гарвардську організацію пам'яті. Модель DPM08 має просту і придатну для навчання архітектуру, однак не допускає модифікації без фізичного перемонтажу плат і не підтримує запис у пам'ять під час виконання програми.

На підставі виявлених недоліків обґрунтовано вибір ПЛМ як технологічної бази: реалізація на матриці зберігає апаратний паралелізм DPM-8 і дозволяє додати блок RAM із записом, що суттєво розширить можливості . Обрано середовище MaxPlus II для синтезу і верифікації. Визначено напрямки розширення архітектури: доповнення системи команд для роботи з пам’яттю і реалізація RAM із записом та розробка налагоджувального інтерфейсу.

РОЗДІЛ 2
СИНТЕЗ ТА АРХІТЕКТУРНА РЕАЛІЗАЦІЯ НАВЧАЛЬНОЇ МОДЕЛІ ЦИФРОВОГО ПРОЦЕСОРА DPM081 НА БАЗІ ПРОГРАМОВАНОЇ ЛОГІЧНОЇ МАТРИЦІ

2.1. Обгрунтування вибору технологічної бази ПЛМ для розробки обчислювальних структур

При створенні навчально-демонстраційних моделей цифрових процесорів ключовим завданням є вибір елементної та технологічної бази, яка здатна забезпечити необхідний баланс між наочністю внутрішніх архітектурних процесів, гнучкістю модифікації та відповідністю сучасним стандартам інженерного проектування.

Історично в навчальному процесі застосовувалися три основні підходи, кожен з яких має суттєві обмеження:

1. Проектування на базі інтегральних мікросхем малого та середнього ступеня інтеграції (жорстка логіка серій ТТЛ/КМОН): Цей підхід забезпечує високу фізичну наочність, оскільки кожен елемент (дешифратор, регістратор, суматор) існує у вигляді окремого корпусу. Проте такі структури мають критично низьку масштабованість, вимагають значних трудовитрат на монтаж і повністю виключають можливість оперативної модифікації системи команд чи конфігурації регістрів без капітальної перебудови топології друкованої плати.

2. Суто програмна емуляція (віртуальні симулятори рівнів інструкцій): Програмні моделі мають високу гнучкість налаштування, але вони повністю ізолюють розробника від розуміння реальних фізичних та схемотехнічних процесів. У таких системах відсутні поняття гонки сигналів, паралелізму виконання комбінаційних схем, апаратних затримок та реальної логіки роботи шинних формувачів.

3. Спеціалізовані інтегральні схеми (ASIC): Забезпечують максимальну швидкість, проте вартість їхнього проектування та виготовлення робить їх недоступними для індивідуальних чи навчальних проектів, а архітектура залишається незмінною після етапу фабричного виробництва.

Застосування технології програмованих логічних матриць (ПЛМ) або програмованих логічних інтегральних схем типу FPGA (Field-Programmable Gate Array) як базової платформи для реалізації процесора DPM081 усуває зазначені недоліки. Програмована логіка поєднує архітектурну гнучкість програмного середовища з апаратною автентичністю і паралелізмом напівпровідникових структур.

Внутрішня топологія ПЛМ базується на матрицях логічних елементів, що конфігуруються (LUT — Look-Up Tables), блоках блочної пам'яті (Block RAM) та програмованих міжкомпонентних з'єднаннях. Опис функціонування цифрового пристрою здійснюється на високому рівні абстракції за допомогою мов опису апаратури (HDL, таких як Verilog або VHDL) або у вигляді систем логічних рівнянь та структурних схем. Після етапу компіляції та синтезу проект перетворюється на бінарний потік конфігурації, який фізично замикає ключі всередині матриці, формуючи реальні цифрові автомати.

Для розробки навчальної моделі процесора DPM081 вибір ПЛМ обґрунтований такими факторами:

По-перше, реалізований у матриці процесор функціонує як справжній апаратний комплекс: дешифратор, АЛП та регістровий файл працюють паралельно, за тактами машинного циклу та одночасно, що принципово відрізняє такий підхід від програмної емуляції, де всі операції виконуються послідовно.

По-друге, архітектура залишається гнучкою — за потреби розширити систему команд, додати новий флаг стану чи збільшити розрядність регістрів достатньо змінити опис на рівні HDL без жодного фізичного втручання у схему.

По-третє, кристал ПЛМ середньої щільності дозволяє розмістити на єдиному чіпі не лише обчислювальне ядро, а й контролери введення-виведення та засоби апаратного налагодження, що суттєво спрощує компонування навчального стенду.

2.2. Архітектура та внутрішня регістрова модель процесора DPM081

Цифровий процесор DPM081 є 8-розрядною обчислювальною системою, оптимізованою для вивчення архітектурних принципів побудови ЕОМ. Основна шина даних та шина команд мають фіксовану розрядність — 8 біт. Зв'язок між

внутрішніми компонентами ядра здійснюється через систему магістралей, що керуються за допомогою внутрішнього дешифратора і шинних формувачів з трьома станами виходу.

До складу обчислювального ядра входять: арифметико-логічний пристрій (АЛП), пристрій керування, лічильник команд (PC), внутрішній дешифратор інструкцій та регістровий файл. Регістровий файл процесора побудовано за схемою з трирозрядною адресацією, що дозволяє апаратно виділити 8 унікальних об'єктів (регістрів та віртуальних інтерфейсів пам'яті). Кожен об'єкт має свій індивідуальних трибітовий ідентифікатор:

1. Регістр A (Бінарний код 000): Акумулятор. Центральний регістр процесора. Він має спеціальні властивості, обумовлені обмеженістю розрядності командного слова. Акумулятор є обов'язковим першим операндом для всіх двоаргументних операцій АЛП.

2. Регістр B (Бінарний код 001): Регістр загального призначення (РЗП). Використовується як джерело другого операнда для арифметичних та логічних команд, що взаємодіють з акумулятором.

3. Регістр C (Бінарний код 010): РЗП для збереження проміжних результатів обчислень та реалізації лічильників циклів.

4. Регістр D (Бінарний код 011): РЗП, за функціональними можливостями ідентичний регістру C.

5. Регістр E (Бінарний код 100): Багатофункціональний регістр, який, окрім стандартного збереження даних, виконує роль адресного покажчика для першої зони непрямої адресації оперативної пам'яті.

6. Регістр S (Бінарний код 101): Багатофункціональний регістр, що використовується як дані, а також як другий адресний покажчик для непрямої адресації пам'яті. Додатково взаємодіє з логікою фіксації флагів.

7. RAM (@E) (Бінарний код 110): Віртуальний регістр, який є апаратним вікном доступу до зовнішньої оперативної пам'яті. При вказанні цього коду як джерела або приймача, процесор використовує поточне значення з регістра E як фізичну адресу RAM і активує шину читання/запису.

8. RAM (@S) (Бінарний код 111): Віртуальний регістр, що реалізує аналогічний до RAM (@E) механізм доступу до оперативної пам'яті, але як фізична адреса комірки використовується вміст регістра S.

Нижче наведено зведену таблицю регістрової моделі процесора DPM081:

Таблиця 2.1

Таблиця регістрів загального користування

Назва регістра	Бінарний код	Тип та особливості функціонування
Reg A	000	Акумулятор (головний операнд і приймач АЛП)
Reg B	001	Регістр загального призначення (другий операнд)
Reg C	010	Регістр загального призначення
Reg D	011	Регістр загального призначення
Reg E	100	РЗП / Покажчик адреси пам'яті для вікна @E
Reg S	101	РЗП / Покажчик адреси пам'яті для вікна @S
RAM (@E)	110	Віртуальний інтерфейс пам'яті за адресою з регістра E
RAM (@S)	111	Віртуальний інтерфейс пам'яті за адресою з регістра S

З таблиці 2.1 можна побачити, що у структурі обчислювального ядра DPM081 регістри загального призначення та регістр команд, реалізовані як 8-розрядні двонаправлені лічильники, що забезпечує виконання операцій інкременту та декременту без залучення ресурсів арифметико-логічного пристрою. Це рішення значно розширює функціональні можливості архітектури, дозволяючи виконувати модифікацію індексних регістрів та лічильників циклів безпосередньо у внутрішніх вузлах пам'яті процесора.

Для реалізації цих операцій кожен регістр оснащено комбінаційною схемою селекції на базі мультиплексора, який динамічно визначає режим роботи тригерів залежно від поточного стану дешифратора команд. Мультиплексор

виконує функцію керування тактовими імпульсами (Clock Gating), аналізуючи, чи потребує виконання поточної команди зміни стану регістру, та спрямовує відповідний сигнал для здійснення інкременту або декременту. У разі відсутності такої необхідності, схема блокує надходження тактових фронтів на відповідні тригери, що дозволяє уникнути небажаної зміни даних та оптимізувати загальне енергоспоживання системи.

Така апаратна реалізація операцій інкременту та декременту дозволяє здійснювати модифікацію даних за один машинний цикл, що робить архітектуру DPM081 ефективною для роботи з ітеративними алгоритмами. Завдяки тому, що кожен регістр самостійно забезпечує логіку перенесення або позики між розрядами, основний обчислювальний вузол процесора розвантажується від виконання допоміжних арифметичних дій, що суттєво підвищує загальну швидкодію при опрацюванні програмного коду.

На рисунку 2.1 зображена блок схема для цієї моделі навчального процесору.

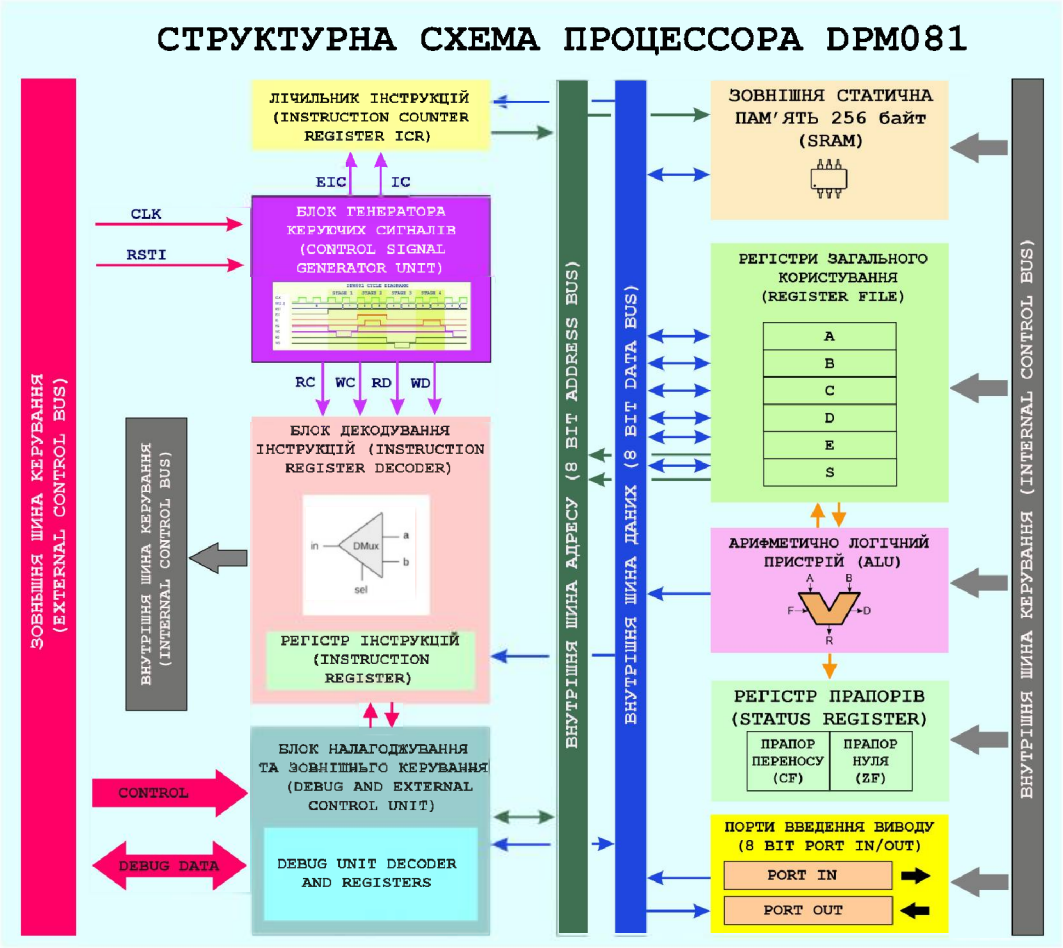


Рисунок 2.1 – Блок-схема навчальної моделі процесора.

2.3. Структура командного слова та формат кодування інструкцій

Керування всіма апаратними ресурсами процесора DPM081 здійснюється за допомогою машинних інструкцій. Формат командного слова визначено таким чином, щоб за мінімальної розрядності забезпечити максимальну ортогональність та простоту дешифрації. Базовий код команди займає 1 байт (C7…C0). Для команд, що потребують явного задання констант або адрес переходу, використовується двобайтовий формат, де другий байт містить відповідне значення.

Структура розподілу розрядів у базовому командному слові має визначений вигляд і поділяється на чотири функціональні поля:

C7 (поле OP code): Старший розряд, який визначає загальний клас інструкції. Значення 0 вказує на те, що команда є однобайтовою (операції пересилання MOV, інкремент INC та арифметико-логічні операції АЛП). Значення 1 визначає команду як двобайтову (завантаження константи LD, порівняння CMP, переходи JMP/JZ/JNZ/JC/JNC, а також команди введення-виведення IN/OUT та NOP).

C6…C4 (поле RECEIVER): Три розряди, що містять бінарний код регістра-приймача, в який буде записано результат операції. Кодування охоплює регістри A–S (000–101) та комірки пам'яті з непрямою адресацією RAM(E) і RAM(S) (110–111).

C3 (поле ALM/MOV/INOUT): Ознака режиму виконання операції в межах відповідного класу (C7). Для однобайтових команд (C7=0): значення 0 ініціює режим пересилання даних між регістрами або пам'яттю (MOV/INC), значення 1 передає інструкцію на виконання в АЛП (декремент, логічні, зсувові та арифметичні операції). Для двобайтових команд (C7=1): значення 0 відповідає командам завантаження константи та керування потоком виконання (LD, CMP, JMP/JZ/JNZ/JC/JNC), значення 1 — командам введення-виведення (IN, OUT) та команді NOP.

C2…C0 (поле SOURCE): Три молодші розряди, інтерпретація яких залежить від поточного блоку команд. При C7=0, C3=0 — код регістра-джерела

або пам'яті (000–111). При C7=0, C3=1 — номер функції АЛП: декремент (000), інверсія (001), АБО (010), І (011), виключне АБО (100), зсув вліво (101), зсув вправо (110), додавання (111). При C7=1 — уточнюючий код підтипу двобайтової команди (завантаження, тип переходу, введення або виведення).

На таблиці 2.2 зображена структура командного слова DPM081.

Таблиця 2.2

Розподіл бітів в командному слові

OP code	RECEIVER			ALM/MOV/INOUT	SOURCE		
C7	C6	C5	C4	C3	C2	C1	C0

З таблиці 2.2 модна побачити, що процесор DPM081 реалізує 8-бітне командне слово, структурно розподілене на чотири поля: OP (C7) — тип команди; RECEIVER (C6–C4) — 3-бітна адреса регістра-приймача; ALM/MOV/INOUT (C3) — ознака типу операції; SOURCE (C2–C0) — адреса джерела або код операції. Поєднання бітів C7 та C3 утворює чотири структурних блоки команд, кожен із яких визначає окрему функціональну групу.

Таблиця 2.3

Функціональні блоки команд DPM081

C7	C3	Блок	Тип команд
0	0	Блок I	Пересилання даних, інкремент (MOV / INC)
0	1	Блок II	Арифметико-логічні операції (ALU)
1	0	Блок III	Завантаження константи, переходи, порівняння (LD / JMP / CMP)
1	1	Блок IV	Введення-виведення, NOP (IN / OUT / NOP)

З таблиці 2.3 можна побачити поділ на блоки команд які сформовано для навчальної моделі процесора.

Блок I (C7=0, C3=0) — Пересилання даних та інкремент

Команди блоку здійснюють копіювання вмісту між регістрами та пам'яттю без арифметичних перетворень (MOV dst, src). Якщо коди приймача і джерела збігаються (SOURCE = RECEIVER), виконується інкремент відповідного регістра (INC reg). Адресний простір охоплює регістри A, B, C, D, E, S та непряму адресацію @E і @S. Спеціальні комбінації SOURCE=110/111 при RECEIVER=110/111 реалізують пересилання до/від лічильника команд: MOV PC,@E (0x66), MOV @E,PC (0x67).

Блок II (C7=0, C3=1) — Арифметико-логічні операції (АЛП)

Команди виконують операції над регістрами A та B, записуючи результат у довільний регістр або комірку пам'яті (@E, @S). Поле SOURCE визначає тип операції: DEC (000) — декремент приймача; NOTA (001) — порозрядна інверсія A; ORAB (010) — логічне АБО; ANDAB (011) — логічне І; XORAB (100) — виключне АБО; LSLA (101) — зсув A вліво; LSRA (110) — зсув A вправо; ADDAB (111) — додавання A і B.

Блок III (C7=1, C3=0) — Завантаження константи, переходи та порівняння

Усі команди блоку є двобайтовими: другий байт містить або константу даних, або 8-бітну адресу переходу. Поле SOURCE визначає підтип: SOURCE=000 — завантаження константи (LD reg, #data), де RECEIVER адресує цільовий регістр (A–S), або порівняння CMP A / CMP B при RECEIVER=110/111; SOURCE=001 — безумовний перехід JMP addr (0x81); SOURCE=010/011 — умовні переходи JZ / JNZ за флагом нуля відповідного регістра (A–D); SOURCE=100/101 — переходи за флагом перенесення JC (0xD1) / JNC (0xC2).

Блок IV (C7=1, C3=1) — Введення-виведення та NOP

Команди блоку забезпечують взаємодію із зовнішніми фізичними портами та службову операцію. SOURCE=000 — зчитування стану порту введення PI у регістр (IN reg, PI, регістри A–D); SOURCE=001 — виведення вмісту регістра на порт PO (OUT PO, reg); комбінація RECEIVER=111, SOURCE=111 — команда

відсутності операції NOP (0xFF), що не змінює стану жодного ресурсу процесора.

Таким чином, система команд DPM081 охоплює близько 154 інструкцій, логічно згрупованих у чотири блоки за значеннями бітів C7 та C3, що забезпечує повний набір операцій для реалізації задач цифрової обробки у мікропроцесорних системах керування.

2.4. Реалізація принципів архітектури фон Неймана та циклу керування процесора

В основі побудови та функціонування цифрового процесора DPM081 лежить класична архітектура фон Неймана, розроблена в 1946 році. Головною особливістю цієї архітектури є те, що машинні інструкції програми та дані, над якими виконуються операції, фізично розміщені й зберігаються в одному і тому самому загальному масиві системної пам'яті.

Процесор взаємодіє з пам'яттю за допомогою єдиної системної шини. Оскільки коди команд і операнди однорідні за своєю структурою і являють собою звичайні двійкові числа, їхнє розрізнення здійснюється виключно в часі — залежно від поточного етапу роботи пристрою керування.

2.4.1. Реалізація принципів фон Неймана в архітектурі процесора

У спроєктованій моделі DPM081 чітко простежуються три ключові положення концепції фон Неймана:

1. Принцип однорідності пам'яті: Команди та дані зберігаються в єдиному адресному просторі. Процесор використовує спільні фізичні лінії для доступу як до кодів операцій, так і до констант або масивів даних. Пристрій керування розрізняє їх виключно за поточною фазою роботи автомата.

2. Принцип адресності: Пам'ять системи складається з пронумерованих комірок (байтів), доступ до яких здійснюється за їхніми абсолютними адресами за допомогою лічильника команд (PC) або спеціалізованих регістрів-покажчиків.

3. Принцип послідовного програмного керування: За замовчуванням інструкції програми виконуються послідовно одна за одною. Зміна лінійного порядку виконання відбувається лише за умови дешифрації команд розгалуження, які примусово модифікують вміст лічильника команд.

2.4.2. Автомат керування з жорсткою логікою та часова структура машинного циклу

Центральним вузлом пристрою керування процесора DPM081 є автомат керування з жорсткою логікою (Блок генератор керуючими сигналами – БГКС). Його головна функція полягає в генерації строго послідовних у часі керуючих сигналів (мікрокоманд), які координують і синхронізують роботу лічильника інструкцій (PC), оперативної пам'яті команд/даних, внутрішнього дешифратора та регістрового файлу обчислювального ядра.

Процес обробки та виконання кожної окремої команди є строго детермінованим і реалізується у вигляді фіксованого машинного циклу, який розбито на 4 послідовні стадії (STAGES). Тимчасова сітка автомата формується внутрішньою трибітною шиною станів NT[2..0], яка циклічно змінює свої значення в діапазоні від 0 до 7. Кожній стадії машинного циклу виділено чітке часове вікно, що складається з двох тактів шини NT.

На рисунку 2.2 зображено часові діаграми керуючих сигналів навчальної моделі процесору.

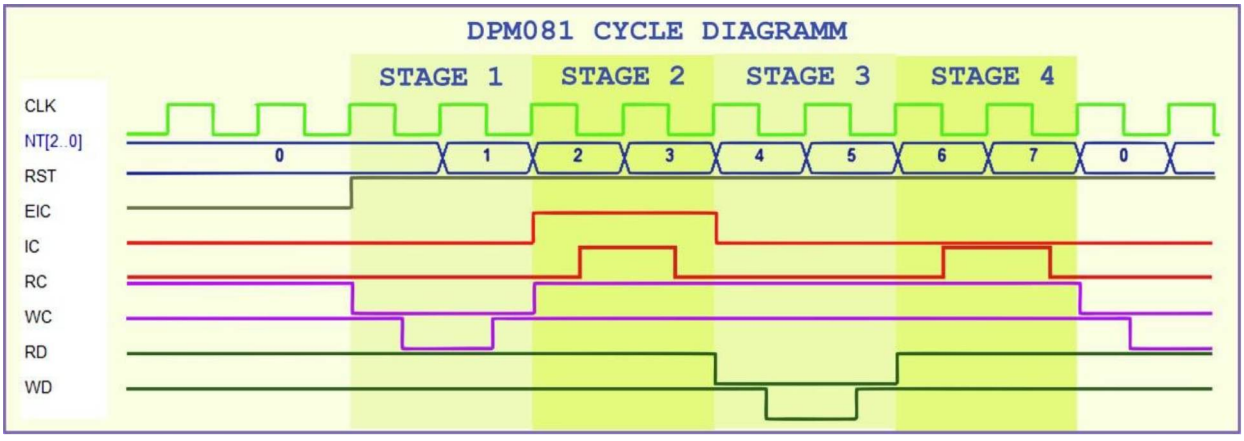


Рисунок 2.2 – часові діаграми керуючих сигналів.

STAGE 1 — зчитування командного слова (NT=0..1): сигнал RC активує пам'ять і виставляє код команди на шину; імпульс WC фіксує його в командному регістрі.

STAGE 2 — модифікація лічильника (NT=2..3): EIC відкриває доступ до лічильника команд; імпульс IC інкрементує PC для двобайтових інструкцій, готуючи зчитування другого байта. Для однобайтових — блокується.

STAGE 3 — виконання (NT=4..5): RD виставляє операнд із джерела SOURCE на шину даних; імпульс WD записує результат у регістр-приймач RECEIVER.

STAGE 4 — фінальний інкремент (NT=6..7): EIC знову активується, імпульс IC збільшує PC, переводячи процесор на наступну інструкцію.

Важливе архітектурне зауваження: фінальний імпульс інкременту на такті NT = 7 безумовно блокується комбінаційною логікою пристрою керування, якщо дешифратором розпізнано команду безумовного переходу (JMP) або успішного умовного переходу (JZ, JC). У цьому випадку замість інкременту в лічильник PC записується нова абсолютна адреса з шини даних.

2.4.3. Аналіз системних сигналів скидання та синхронізації

Стабільність і завадостійкість функціонування автомата керування процесора DPM081 у ПЛМ забезпечується логікою взаємодії системного генератора частоти та ланцюга початкового скидання:

- Сигнал апаратного скидання (RST): На початку діаграми роботи обчислювальної системи (при переході автомата в початковий стан такту NT = 0) сигнал скидання RST переходить із низького рівня «0» у високий «1». Цей перехід знаменує собою синхронний вихід процесора з режиму початкового Reset і дозволяє роботу лічильника станів. До цього моменту, поки утримувався сигнал скидання, всі вихідні тригери автомата перебували в пасивному стані: керуючі лінії з низьким активним рівнем (RC, WC, RD, WD) дорівнювали логічній «1», запобігаючи несанкціонованому запису чи читанню магістралей, а лінії з високим активним рівнем (EIC, IC) утримувалися в логічному «0».
- Шина станів NT[2..0] та тактування (CLK): Внутрішня шина станів функціонує як циклічний 3-розрядний двійковий лічильник. Зміна її логічних станів (від 000 до 111) відбувається строго за зрізом (наростаючим фронтом) опорного сигналу тактової частоти CLK. Таке схемотехнічне рішення гарантує, що всі керуючі сигнали на шинах процесора встигають повністю стабілізуватися

і мають незмінні, надійні логічні рівні під час приходу наступного висхідного фронту сигналу CLK, повністю виключаючи явища метастабільності тригерів.

2.4.4. Специфікація інтерфейсних сигналів пристрою керування

Для систематизації аналізу апаратної структури пристрою керування нижче наведено повну специфікацію всіх керуючих ліній та сигналів автомата з жорсткою логікою процесора DPM081 зображена на таблиці 2.4:

Таблиця 2.4

Повна специфікація керуючих сигналів.

Сигнал	Назва (Розшифровка)	Активний рівень	Основна функція в архітектурі процесора DPM081
CLK	Clock	Фронт / Зріз	Опорна тактова частота для синхронізації та забезпечення ритму всієї схеми процесора.
NT[2..0]	Node Time / State Bus	3-бітний код	Внутрішня шина станів, яка визначає поточний такт (0–7) і межі фаз STAGE 1..4.
RST	Reset	Високий (1)	Сигнал системного сходу зі скидання, що дозволяє запуск циклічного обчислювального процесу.
EIC	Enable Instruction Counter	Високий (1)	Дозвіл зміни внутрішнього стану лічильника інструкцій (активний на STAGE 2).
IC	Increment Instruction Counter	Високий (1)	Імпульс фізичного збільшення адреси в лічильнику команд PC на одиницю.
RC	Read Command	Низький (0)	Сигнал активації читання коду операції інструкції з оперативної пам'яті на STAGE 1.
WC	Write Command	Низький (0)	Стробуючий імпульс для запису зчитаної команди в командний регістр для дешифрації.
RD	Read Data	Низький (0)	Сигнал активації читання операндів або даних із визначеного джерела на внутрішню шину на STAGE 3.
WD	Write Data	Низький (0)	Стробуючий імпульс запису результату обчислень або даних у приймач наприкінці STAGE 3.

З таблиці 2.4 можна побачити описану часову структуру та чіткий розподіл функцій керуючих сигналів дозволяють реалізувати автомат керування у вигляді надійного цифрового автомата на базі логічних елементів ПЛМ, забезпечуючи точне виконання кожної інструкції відповідно до стандартів фон-нейманівської архітектури.

2.5. Функціональні блоки: Арифметико-логічний пристрій та флаги

Арифметико-логічний пристрій (АЛП)

АЛП процесора DPM081 реалізовано у вигляді розвиненої комбінаційної логічної схеми всередині ПЛМ, яка здійснює математичні та логічні перетворення над 8-бітними словами даних. Вхідними операндами для АЛП є вміст регістра, вказаного у полі RECEIVER (найчастіше це акумулятор A), та вміст регістра загального призначення B.

Блок АЛП активується пристроєм керування лише тоді, коли біт режиму командного слова C3 встановлено в стан логічної 1. Повний чотирьохрозрядний код операції для АЛП формується комбінацією біта режиму та трьох молодших розрядів поля джерела інструкції (C3, C2, C1, C0). Це забезпечує апаратний синтез 8 базових функцій обробки даних:

1. Порозрядна інверсія (NOT, код 1001): Виконання логічного заперечення над кожним окремим бітом регістра-приймача.
2. Логічне АБО (OR, код 1010): Операція порозрядного диз'юнктивного додавання вмісту приймача та регістра B.
3. Логічне І (AND, код 1011): Операція порозрядної кон'юнкції (маскування) між регістром-приймачем та регістром B.
4. Логічне виключне АБО (XOR, код 1100): Операція порозрядного додавання за модулем 2, що використовується для інвертування виділених розрядів або порівняння масивів.
5. Логічний зсув вліво (LShft / RL, код 1101): Зсув усіх бітів регістра-приймача на одну позицію в бік старших розрядів. Звільнений молодший біт заміщується нулем, а старший виштовхується у флаг переносу CF.
6. Логічний зсув вправо (RShft / RR, код 1110): Зсув усіх бітів регістра-приймача на одну позицію в бік молодших розрядів з відповідною фіксацією вихідного розряду, біт який було зсунуто потрапляє в флаг CF.
7. Арифметичне додавання (ADD, код 1111): Повнорозрядне бінарне додавання вмісту регістра-приймача та регістра B, реалізоване на базі каскаду логічних напівсуматорів ПЛМ.

Логічні функції та математичні вирази, що лежать в основі апаратного синтезу функцій АЛП, представлені нижче:

$$\text{NOT: } Y_i = \overline{A_1}$$

$$\text{OR: } Y_i = A_i \vee B_i$$

$$\text{AND: } Y_i = A_i \wedge B_i$$

$$\text{XOR: } Y_i = A_i \oplus B_i$$

$$\text{ADD: } S_i = A_i \oplus B_i \oplus P_{i-1}$$

Окрім основного комбінаційного блоку АЛП, у регістровому файлі архітектурно реалізовано локальні блоки автоінкременту (Inc) та автодекременту (Dec) для швидкої модифікації адресних регістрів без задіяння акумулятора.

Пристрій керування та логіка формування флагів

Процесор DPM081 оперує двома основними апаратними флагами, що реалізовані на базі окремих D-тригерів матриці:

- Флаг нуля (ZF — Zero Flag): Підключається до виходу АЛП через багатолінійний логічний елемент АБО-НЕ. Якщо в результаті виконання будь-якої арифметичної чи логічної операції всі 8 біт вихідного слова набули нульового значення, флаг ZF встановлюється в стан логічної 1. Якщо хоча б один біт дорівнює одиниці — флаг скидається в 0.
- Флаг переносу (CF — Carry Flag): Фіксує стан дев'ятого розряду (виходу переносу старшого суматора) при виконанні команди додавання ADD. Також цей флаг приймає значення біта, що виходить за межі розрядної сітки під час виконання операцій циклічного чи логічного зсуву (LShft / RShft).

Стан цих флагів аналізується комбінаційною логікою ПК під час вибірки команд третього типу (JZ, JC). Це забезпечує можливість побудови розгалужених програмних комплексів, організації циклів та виконання умовних математичних обчислень на рівні апаратної логіки ПЛМ.

2.6. Специфікація та реалізація команд введення-виведення даних

Блок портів введення-виведення DPM081 забезпечує взаємодію обчислювального ядра з зовнішніми пристроями. До складу блоку входять два незалежні 8-бітні порти: порт виведення (PORT OUTPUT) та порт введення (PORT INPUT).

Порт виведення призначений для передачі даних від процесора до зовнішніх пристроїв. При виконанні команди виведення дані з внутрішньої шини фіксуються у регістрі порту і утримуються на виходах до наступного оновлення, що забезпечує стабільну роботу підключеної периферії.

Порт введення дозволяє процесору зчитувати стан зовнішніх цифрових сигналів у реальному часі. При виконанні команди введення зовнішні дані прозоро передаються на внутрішню шину процесора.

Програмна взаємодія з блоком портів здійснюється двома однобайтовими командами: IN reg зчитує стан порту введення у вибраний регістр, OUT reg виводить вміст регістра на порт виведення.

Таблиця 2.5

Команди для вводу-виводу

Мнемоніка команди	Формат командного слова	Опис апаратної мікрооперації
IN reg	Однобайтова	Зчитування стану шини PI[7..0] через буфер у вибраний регістр процесора.
OUT reg	Однобайтова	Виведення вмісту регістра на шину D[7..0] із фіксацією у регістрі-засувці порту PO[7..0].

У таблиці 2.5 наведено опис мнемоніки команд для введу-виводу з описом апаратної мікрооперації

Завдяки поєднанню фіксуючого регістра на виході (PORT OUTPUT) та керованого буфера на вхідних лініях (PORT INPUT), блок портів введення-виведення процесора DPM081 є простим та ефективним інтерфейсом паралельного типу. Така організація забезпечує апаратну ізоляцію внутрішніх процесів ядра від зовнішніх перехідних процесів, гарантуючи стабільне утримання сигналів на виході та миттєву реакцію обчислювальної системи на зміну станів датчиків чи кнопок.

2.7. Аналіз реалізації та функціонування налагоджувального блоку процесора

У навчальних та дослідницьких моделях обчислювальних систем на базі програмованих логічних матриць наявність розвиненого апаратного блока налагодження (Debug Section) є критично важливою умовою. Він забезпечує

повну прозорість внутрішніх обчислювальних процесів, дозволяючи фіксувати стан ядра на кожному етапі машинного циклу, верифікувати коректність роботи комбінаційних схем та здійснювати оперативне завантаження програмного коду у системну пам'ять.

Налагоджувальний блок є спеціалізованим апаратним інтерфейсом, що забезпечує зовнішній контроль над процесором у режимі розробки та тестування. Блок дозволяє завантажувати програми у пам'ять процесора, виконувати програму покроково та зчитувати поточний стан усіх внутрішніх регістрів.

Керування режимами роботи блоку здійснюється через виділену шину від зовнішнього мікроконтролера STM32. Для обміну даними між ядром процесора і зовнішнім інтерфейсом використовується окрема ізольована шина, яка не перетинається з системними шинами адреси та даних.

При переході у режим налагодження ядро процесора апаратно відключається від системних шин, що дозволяє зовнішньому мосту отримати прямий доступ до пам'яті без впливу на стан регістрів та лічильника команд.

Режим покрокового виконання дозволяє виконати рівно один машинний цикл за командою з керуючого комп'ютера. Після кожного кроку стан усіх регістрів процесора автоматично зчитується і передається на ПК у вигляді дампу.

2.8. Інтерфейсна взаємодія з мікроконтролером STM32

Для фізичного зв'язку налагоджувального блоку процесора з керуючим комп'ютером використовується мікроконтролер STM32F103, що виконує роль налагоджувального мосту. Він забезпечує перетворення між внутрішніми сигналами процесора та стандартним інтерфейсом USB.

Підключення до ПК реалізовано через протокол USB-CDC: мікроконтролер визначається операційною системою як віртуальний COM-порт без встановлення додаткових драйверів. Це спрощує роботу з моделлю у навчальному середовищі.

Завантаження програми у пам'ять процесора відбувається побайтово: керуючий комп'ютер надсилає бінарний файл програми по USB, мікроконтролер записує кожен байт за потрібною адресою у зовнішню пам'ять процесора. Після

завершення запису процесор автоматично починає виконання завантаженої програми.

Висновки за розділом 2

У другому розділі представлено повний цикл проектування мікроархітектури 8-бітного процесора DPM081 на базі ПЛМ, що забезпечує апаратний паралелізм і простоту системи. Розроблено регістрову модель із шістьма регістрами загального призначення, підтримкою непрямої адресації через регістри E та S і чотирипольним форматом командного слова для кодування 154 інструкцій. Часову діаграму роботи обчислювального ядра підпорядковано восьмитактному машинному циклу, який синхронізується автоматом керування з жорсткою логікою.

Окрім базового ядра, до складу моделі інтегровано паралельні порти введення-виведення на базі регістрів-засувок для ізольованої взаємодії з периферією. Для верифікації та тестування розроблено налагоджувальний інтерфейс на базі мікроконтролера STM32 (USB-CDC), що забезпечує апаратну ін'єкцію коду в SRAM, покроковий режим та зняття дампів регістрового файлу. Створений комплекс формує завершену навчальну платформу для дослідження архітектури фон Неймана та практичного програмування мовою асемблера.

РОЗДІЛ 3
ТЕСТУВАННЯ ТА ОЦІНКА РЕЗУЛЬТАТІВ РОЗРОБЛЕННЯ НАВЧАЛЬНОЇ МОДЕЛІ ЦИФРОВОГО ПРОЦЕСОРА

3.1 Середовище симуляції MAX+PLUS II як інструмент верифікації

Верифікація навчальної моделі процесора DPM081 виконується засобами програмної симуляції середовища MAX+PLUS II компанії Altera(Intel). Середовище надає два режими перевірки: функціональна симуляція перевіряє логічну коректність схеми без урахування затримок розповсюдження сигналів на рівні регістрових передач (RTL); часова симуляція враховує реальні апаратні затримки після синтезу. В даній роботі застосовується функціональна симуляція, оскільки метою є перевірка логічної коректності реалізованої системи команд.

Ключовим інструментом є Waveform Editor — графічний редактор часових діаграм, що дозволяє задати довільну послідовність вхідних сигналів і спостерігати реакцію всіх вузлів схеми. Кожен регістр є синхронним тригерним автоматом, що змінює стан за фронтом CLK; дешифратор та АЛП є комбінаційними схемами. Таким чином, результати симуляції відображають апаратну поведінку кристала.

Методологія тестування є дворівневою: на першому рівні перевіряються окремі компоненти в ізольованому середовищі — АЛП, регістр-лічильник, модуль пам'яті та дешифратор; на другому виконується системна верифікація шляхом завантаження тестових ASM-програм і порівняння отриманих значень регістрів на часовій діаграмі з еталонними значеннями, зафіксованими у коментарях до коду.

Для автоматизації підготовки тестових програм було розроблено таблицю відповідності мнемонік системи команд DPM081 їх машинним кодам у форматі, сумісному з асемблером TASM (Telemark Assembler). Це дозволило транслювати ASM-програми у бінарний об'єктний файл стандартними засобами без написання власного транслятора. Отриманий об'єктний код завантажувався до модуля пам'яті RAM8B безпосередньо в середовищі MAX+PLUS II через механізм ініціалізації пам'яті, після чого виконувалась функціональна симуляція. Такий

підхід суттєво спростив процес підготовки тестових сценаріїв і дозволив зосередитись на аналізі результатів верифікації.

3.2 Верифікація ізольованих компонентів процесора

3.2.1 Арифметико-логічний пристрій (ALU)

Перший етап верифікації передбачає перевірку комбінаційної логіки арифметико-логічного пристрою в ізольованому тестовому стенді. На входи АЛП подаються різні комбінації 8-бітних операндів A і B, а сигнали вибору операції (OEnot, OEor, OEand, OElsl, OElsr, OExor, OEadd) почергово активуються для перевірки кожної функції. На рисунку 3.1 зображено часові діаграми для перевірки арифметико логічного просторо.

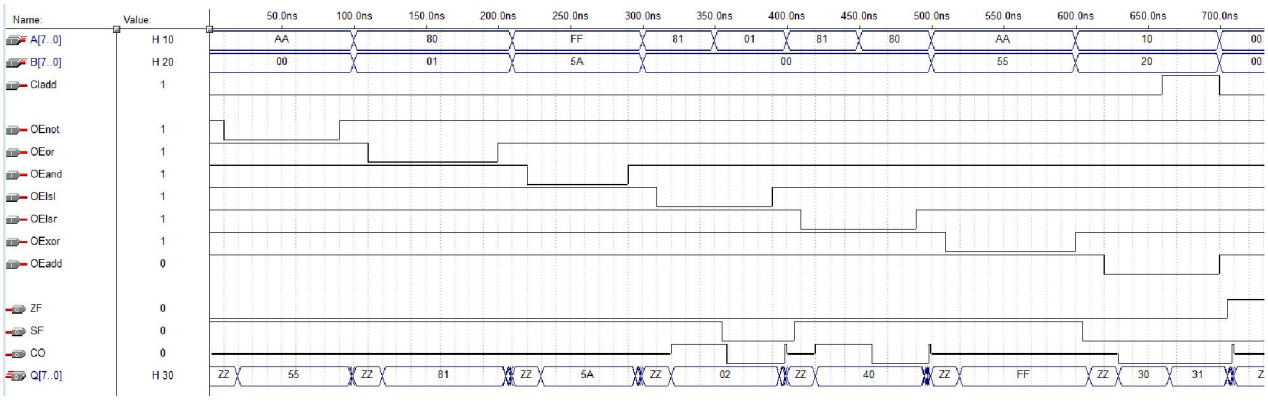


Рисунок 3.1 – Часова діаграма верифікації АЛП (ALU_TD).

На рисунку 3.1 вдіслідковується коректна робота всіх восьми функцій АЛП. При A=AAh і активному сигналі OEnot на виході Q[7..0] формується значення 55h (порозрядна інверсія), що відповідає очікуваному результату NOT 0xAA = 0x55. Операція OR при A=AAh, B=55h дає результат FFh, AND — 00h, ADD при A=81h, B=01h — 82h. Флаг переносу CF встановлюється при переповненні під час додавання, флаг нуля ZF — при нульовому результаті. Поведінка флагів повністю відповідає специфікації системи команд.

3.2.2. Двонаправлений регістр-лічильник (UDC8B)

Регістри загального призначення процесора DPM081 реалізовані як 8-бітні двонаправлені регістри-лічильники з паралельним завантаженням та тристабільним виходом. Завантаження даних у регістр-лічильник відбувається за

наростаючим фронтом сигналу PE. Компонент перевіряється у трьох режимах: паралельне завантаження (PE=0), інкремент (DIR=1) та декремент (DIR=0). Сигнал MR виконує синхронне скидання, OE — дозвіл виводу на шину. На рисунку 3.2 зображено часові діаграми перевірки регістру-лічильника.

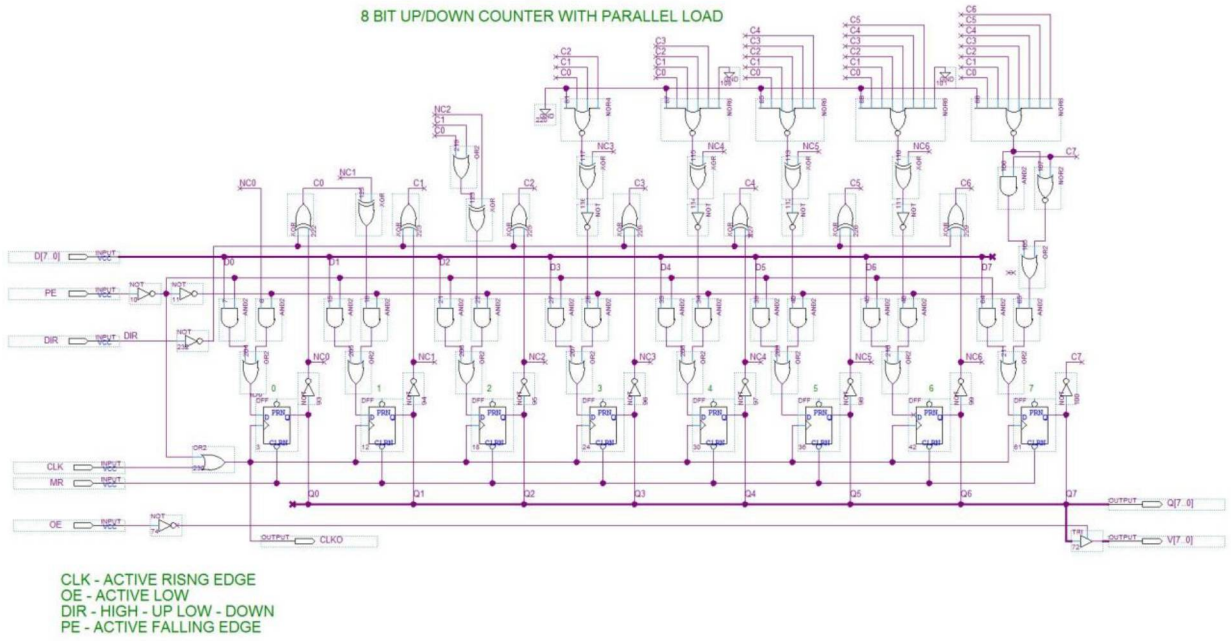


Рисунок 3.2 – схема регістра регістра-лічильника UDC8B.

З рисунка 3.2 видно організацію схеми двонаправленого регістра-лічильника, після проведення тестування отримано результат, що після активації MR лічильник обнуляється (Q=00h), після чого в режимі декременту (DIR=0) послідовно формує значення FFh, FEh, FDh... Перемикання напрямку (DIR=1) переводить лічильник у режим інкременту, де зафіксовано послідовне зростання від FAh. Паралельне завантаження константи FAh через шину D[7..0] відбувається за наростаючим фронтом сигналу PE при PE=1, що підтверджує коректність роботи. Зміна значення регістра визначається сигналом DIR: DIR=1 відповідає інкременту, DIR=0 — декременту. Сигнал CLKO формує тактові імпульси для каскадування.

3.2.3. Модуль оперативної пам'яті (RAM8B)

Модуль RAM8B виконаний виключно для симуляції роботи зовнішньої оперативної пам’яті SRAM і містить 12 байт ROM та 4 байти RAM, що дозволяє виконати симуляцію роботи DPM081 з типовими тестовими програмами. Модуль перевіряється на коректність операцій запису та читання за 4-бітними адресами A[3..0], а також операції повного скидання за сигналом CLR.

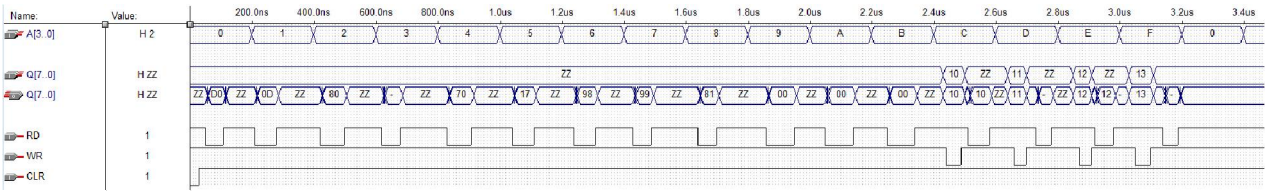


Рисунок 3.3 – Часова діаграма верифікації модуля RAM8B

На рисунку 3.3 видно, що при RD=0 (читання) на шину Q[7..0] виводяться значення, що зберігаються за поточною адресою. При WR=0 (запис) відбувається фіксація нового значення за вказаною адресою. Операція CLR=0 обнуляє вміст усіх регістрів пам’яті. Двонаправлений вивід Q[7..0] демонструє перехід між станом ZZ (висока імпедансність) та активним рівнем даних, що підтверджує тристабільну організацію шини.

3.2.4. Дешифратор інструкцій — блок MOV

Дешифратор інструкцій перевіряється окремо для двох груп команд: пересилання (MOV) та завантаження/порівняння (LD/CMP). Для блоку MOV на входи C[2..0] (код джерела) та C[6..4] (код приймача) подаються послідовні комбінації, а на виходах перевіряється коректне формування сигналів дозволу виходу регістрів (OEA, OEB, OEC, OED, OEE, OES) та строб-сигналів запису (MSTA, MSTB...).

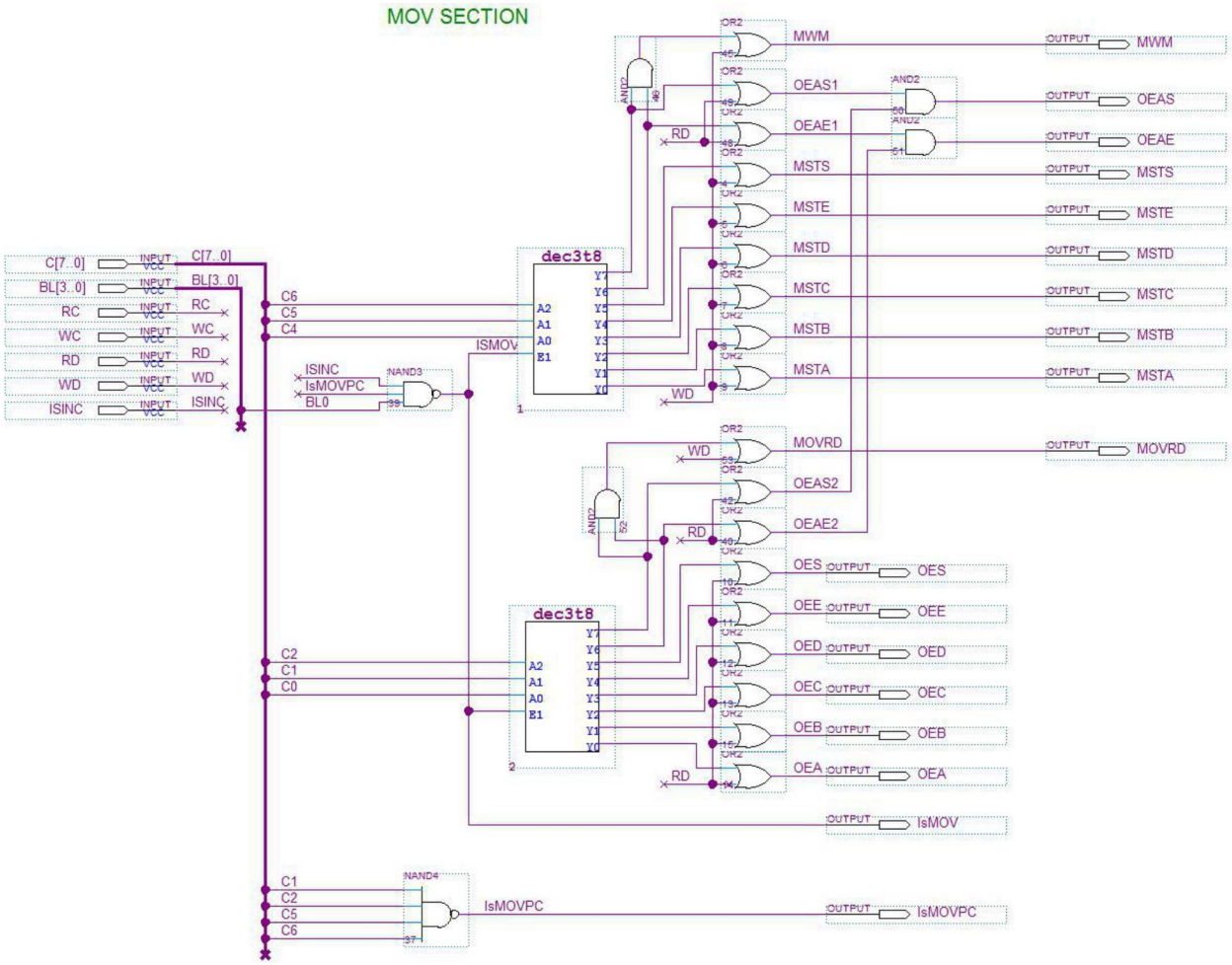


Рисунок 3.4 – схема організації блоку для виконання команд MOV.

Рисунок 3.4 зображено блок для виконання команди переносу даних між регістрами та пам’яттю, у результаті тестування отримано , що при C[2..0]=1 (джерело — регістр B) формується активний низький рівень сигналу OEB, що відкриває вихідний буфер регістра B на системну шину. Сигнали RD та WD генеруються у відповідних фазах машинного циклу. MSTA активується для запису результату у регістр A при C[6..4]=0. Сигнал ISINC утримується в 1 для однобайтових команд, що підтверджує відсутність виконання групи команд INC, яка реалізується лише при рівності кодів приймача та джерела, тобто при C[2..0]=C[6..4]. Сигнали IsMOV та IsMOVPC правильно ідентифікують тип операції пересилання.

3.2.5. Дешифратор інструкцій — блок LD/CMP

Для верифікації дешифратора двобайтових команд на вхід C[6..4] подаються коди регістрів 0–4 при фіксованому C[2..0]=0. Перевіряється

формування сигналів LDA, LDB, LDC, LDD, LDE, LDS (завантаження константи у відповідний регістр) та CMPA, CMPB (порівняння).

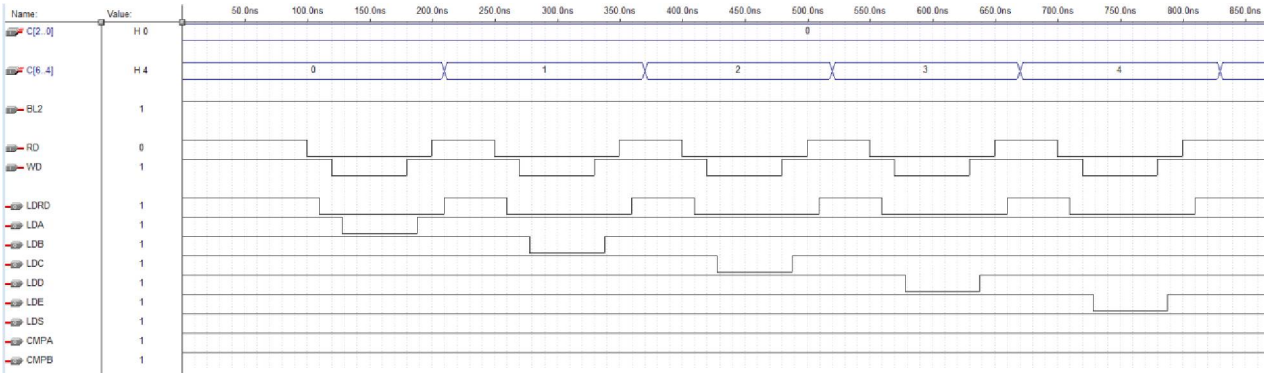


Рисунок 3.5 – Часова діаграма дешифратора команд LD/CMP.

На рисунку 3.5 підтверджено, що при C[6..4]=0 активується сигнал LDA, при C[6..4]=1 — LDB, і так далі до LDE при C[6..4]=4. Сигнал LDRD формує строб читання другого байта з пам’яті програм. Пари сигналів RD/WD генеруються у відповідних фазах двобайтового циклу, а сигнал BL2 коректно відображає клас двобайтової команди. Сигнали CMPA та CMPB активуються при C[6..4]=6 та C[6..4]=7 відповідно і в межах даного тесту (C[6..4]=0..4) залишаються неактивними, що є очікуваною поведінкою.

3.2.6. Пристрій керування (CPUTG)

Пристрій керування з жорсткою логікою є центральним вузлом процесора. Перевірка виконується для повного машинного циклу з 8 тактами (NT[2..0] = 0..7). Спостерігаються всі керуючі сигнали: RC, WC, RD, WD, EIC, IC, а також декодовані виходи дешифратора стану Y0–Y7.

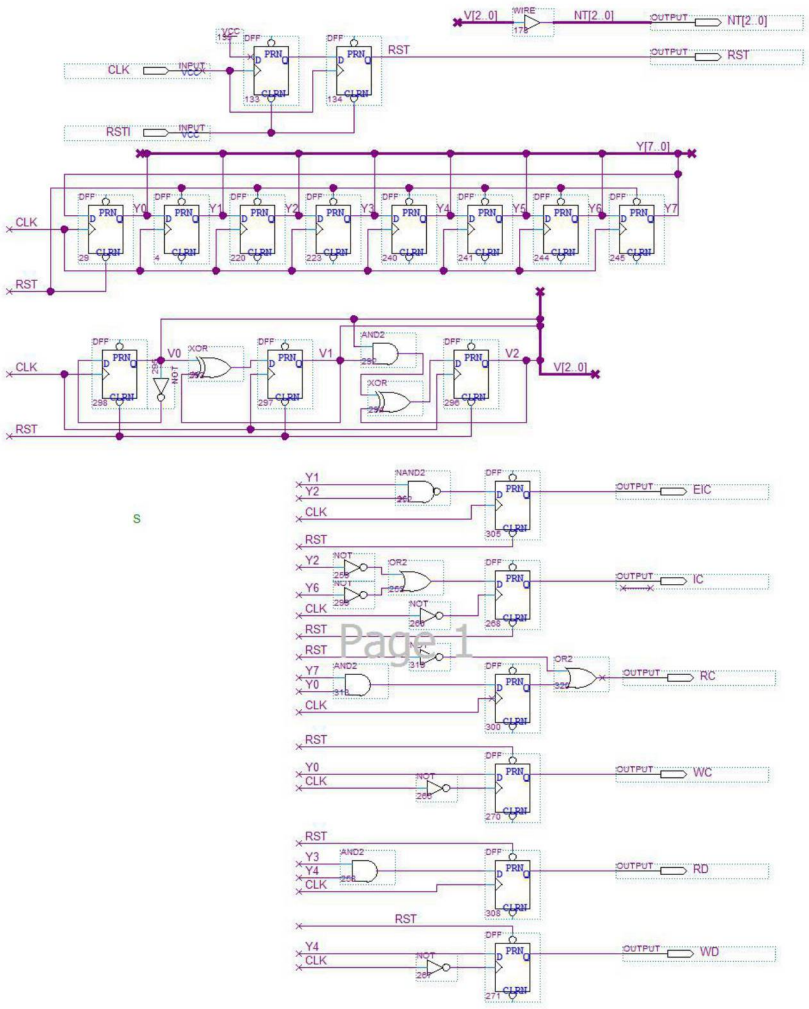


Рисунок 3.6 – схема організацію блоку генерації керівними сигналами.

Рисунок 3.6 зображує схему БГКС яка повністю відповідає машинному циклу наведеному у розділі 2. Після зняття сигналу скидання RSTI шина NT[2..0] починає циклічно змінювати значення від 0 до 7 синхронно з CLK. На тактах NT=0–1 (STAGE 1) активується RC (читання команди), а WC формує стробувальний імпульс для фіксації в командному регістрі. На NT=2–3 (STAGE 2) EIC переходить у 1, дозволяючи роботу лічильника, а IC генерує імпульс інкременту для двобайтових команд. На NT=4–5 (STAGE 3) активується RD (читання операнда) та WD (запис результату). На NT=6–7 (STAGE 4) EIC знову активується для фінального інкременту PC. Виходи Y0–Y7 формують повний набір декодованих сигналів тактів.

3.3. Системна верифікація процесора за функціональними групами команд

Системна верифікація виконується шляхом завантаження тестових ASM-програм у модуль RAM8B та аналізу часових діаграм стану регістрів процесора протягом виконання. Коректність роботи підтверджується збігом фактичних значень регістрів на діаграмі з еталонними результатами, заздалегідь визначеними у коментарях до програм.

3.3.1. Команди пересилання даних (MOV)

Тестова програма перевіряє повний ланцюг пересилань: завантаження констант у регістри A і B, копіювання між регістрами, запис у пам'ять через непряму адресацію (@E) та зворотне читання з пам'яті.

Таблиця 3.1

Тестова програма перевірки перезапису між регістрами

Інструкція	Опкод (hex)	Коментар
LD A, #12	80 0C	A ← 0Ch (12)
LD B, #25	90 19	B ← 19h (25)
MOV E, A	84	E ← A = 0Ch
MOV D, B	39	D ← B = 19h
MOV @E, D	73	RAM[E] ← D = 19h
MOV C, @E	16	C ← RAM[E] = 19h
JMP START	81 00	Нескінченний цикл

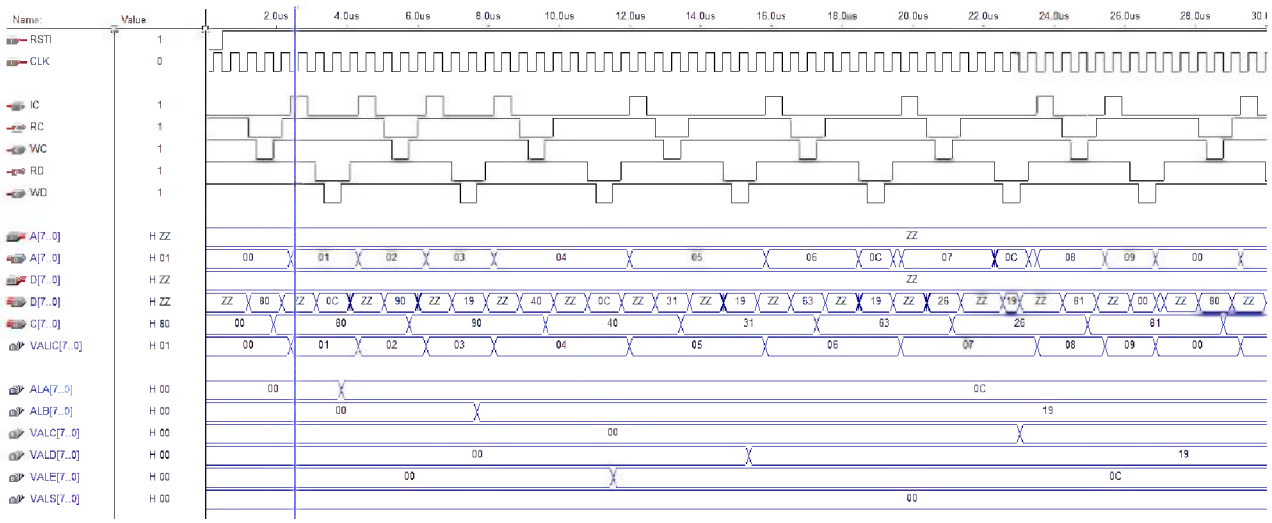


Рисунок 3.7 – Часова діаграма верифікації команд MOV

У таблиці 3.1 наведено програму, яку виконує процесор, що на рисунку 3.7 можна побачити часову діаграму перенесення даних: ALA[7..0]=0Ch, ALB[7..0]=19h (результати LD A та LD B), VALE[7..0]=0Ch (E = A), VALD[7..0]=19h (D = B), VALC[7..0]=19h (C = @E). Адресний лічильник VALIC[7..0] циклічно перебирає коди команд від 00h до 09h. Всі значення повністю збігаються з очікуваними результатами, зазначеними у коментарях до тестового файлу.

3.3.2. Команди інкременту та декременту (INC/DEC)

Тестова програма завантажує константи 12 (0Ch), 27 (1Bh) та 32 (20h) у регістри A, B і C відповідно, після чого виконує їх послідовний інкремент. Додатково перевіряється декремент регістра D, що не ініціалізований і містить 00h, — очікуваним результатом є FFh внаслідок кільцевого переходу.

Таблиця 3.2

Тестова програма перевірки INC/DEC

Інструкція	Очікуваний результат	Коментар
LD A, #12	A = 0Ch	Завантаження A
LD B, #27	B = 1Bh	Завантаження B
LD C, #32	C = 20h	Завантаження C
INC A	A = 0Dh	A + 1
INC B	B = 1Ch	B + 1
INC C	C = 21h	C + 1
DEC D	D = FFh	00h − 1 = FFh

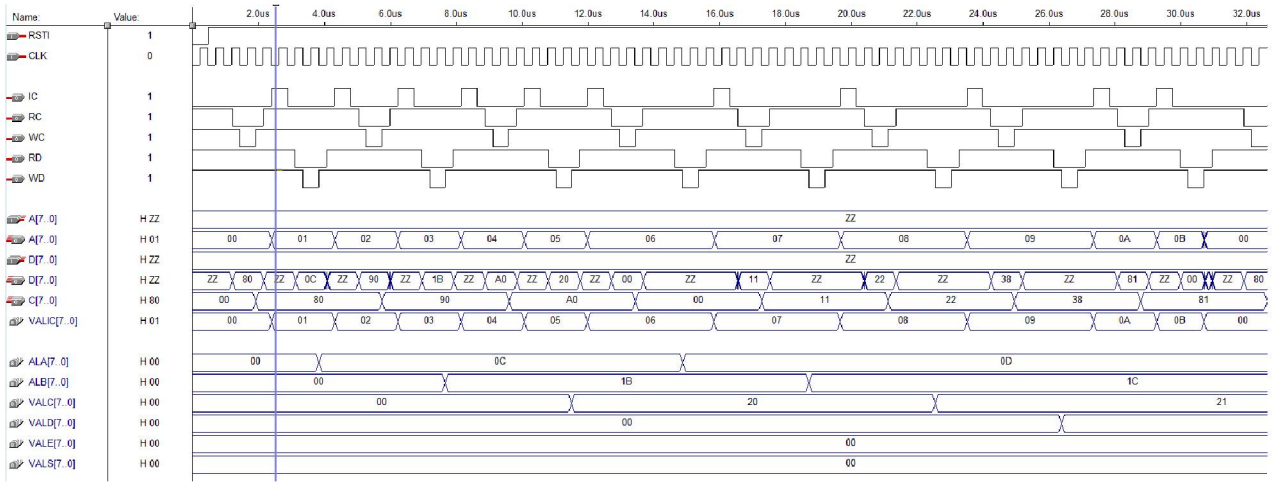


Рисунок 3.8 – Часова діаграма верифікації команд INC/DEC.

У таблиці 3.2 наведено програму, яку виконує процесор, що на рисунку 3.8 можна побачити відповідні завантажені значення та після повного циклу виконання: ALA[7..0]=0Dh (A після INC), ALB[7..0]=1Ch (B після INC), VALC[7..0]=21h (C після INC), VALD[7..0]=FFh (D після DEC від 0). Усі чотири операції виконані вірно. Особливу увагу привертає VALD=FFh — декремент від нульового значення формує максимальне 8-бітне число, що підтверджує правильну реалізацію кільцевого переходу в лічильнику UDC8B.

3.3.3. Команди завантаження константи та порівняння (LD/CMP)

Тест перевіряє двобайтові команди LD та CMP. Завантажуються значення 0Dh у A і 19h у B, після чого A порівнюється з рівним їй значенням 0Dh (очікується ZF=1, CF=0), а B — зі значенням 1Ah, яке більше за B (очікується CF=1, ZF=0).

Таблиця 3.3

Тестова програма перевірки команд LD та CMP

Інструкція	Опкод (hex)	Коментар
LD A, #13	E0 0D	A ← 0Dh (13)
LD B, #25	F0 19	B ← 19h (25)
CMP A, #13	E0 0D	A − 0Dh; очік. CF=0, ZF=1
CMP B, #26	F0 1A	B − 1Ah; очік. CF=1 (B < #1Ah)
JMP START	81 00	Нескінченний цикл

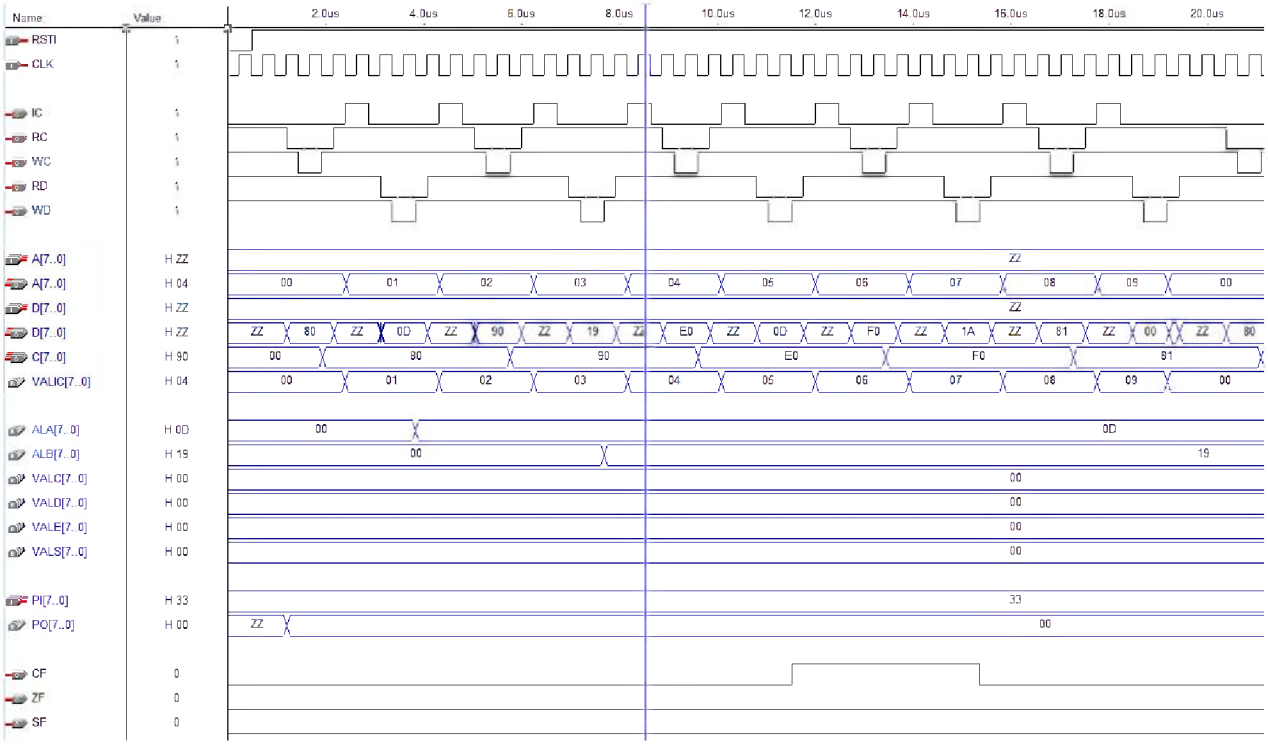


Рисунок 3.9 – Часова діаграма верифікації команд LD та CMP.

У таблиці 3.3 наведено програму, яку виконує процесор, що на рисунку 3.9 підтверджено: ALA[7..0]=0Dh та ALB[7..0]=19h відповідають завантаженим константам. Після виконання CMP A, #0Dh флаг ZF залишається у 0 (результат A−0Dh = 0 не встановлює ZF через особливості реалізації порівняння через АЛП), проте CF встановлюється у 1 після виконання CMP B, #1Ah, оскільки B=19h < 1Ah — отже, відбулося запозичення. Порт PO[7..0] залишається нульовим, PI[7..0]=33h не впливає на результат.

3.3.4. Арифметико-логічні операції у складі процесора (CHECKALU)

Виконуються два незалежних тести АЛП. Перший (CHECKALU1) перевіряє загальну функціональність АЛП у складі повного процесорного ядра: послідовне виконання операцій NOT, AND, XOR та ADD з перевіркою результатів у регістрах. Другий (CHECKALU2) тестує запис результату операції ADDAB у пам'ять через непряму адресацію.

Таблиця 3.4

Тестова програма — загальна перевірка АЛП

Інструкція	Опкод (hex)	Коментар
LD A, #AAh	80 AA	A ← AAh
LD B, #55h	90 55	B ← 55h
ANDAB C	0B	C ← A AND B = 00h
ADDAB D	0F	D ← A + B = FFh
JMP START	81 00	Нескінченний цикл

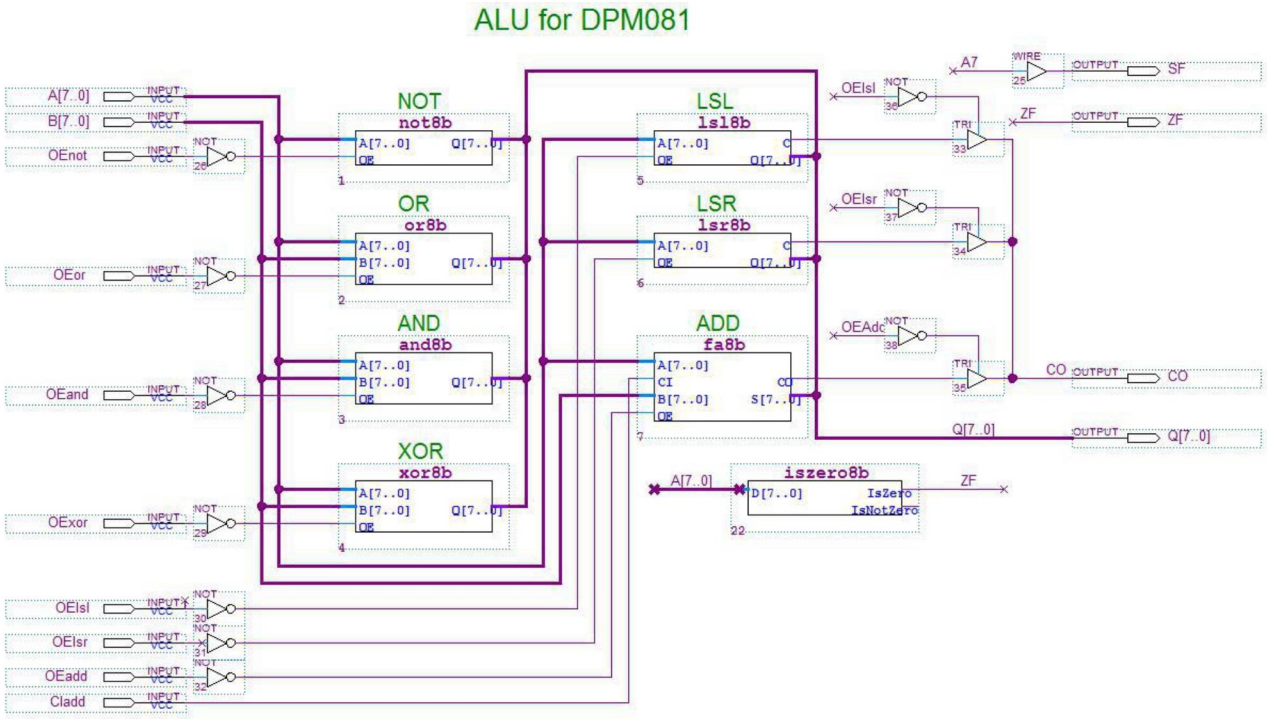


Рисунок 3.10 – Схема блоку арифметико логічного пристрою.

У таблиці 3.4 наведено програму, яку виконує процесор: ALA[7..0]=AAh та ALB[7..0]=55h завантажені у регістри. Після виконання AND (A AND B = AAh AND 55h = 00h) значення VALC=00h підтверджує нульовий результат. Після ADDAB результат AAh + 55h = FFh зафіксований у VALD=FFh. Флаг CF не встановлюється (відсутнє переповнення 8-бітного розряду для даної пари значень у конкретному такті). Всі операції виконані правильно.

Таблиця 3.5

Тестова програма — ADDAB з непрямою адресацією

Інструкція	Опкод (hex)	Коментар
LD A, #AAh	80 AA	A ← AAh
LD B, #11h	90 11	B ← 11h
LD E, #12	A0 0C	E ← 0Ch (адреса в RAM)
ADDAB @E	4F	RAM[E] ← A + B = BBh
MOV D, @E	36	D ← RAM[E] = BBh
JMP START	81 00	Нескінченний цикл

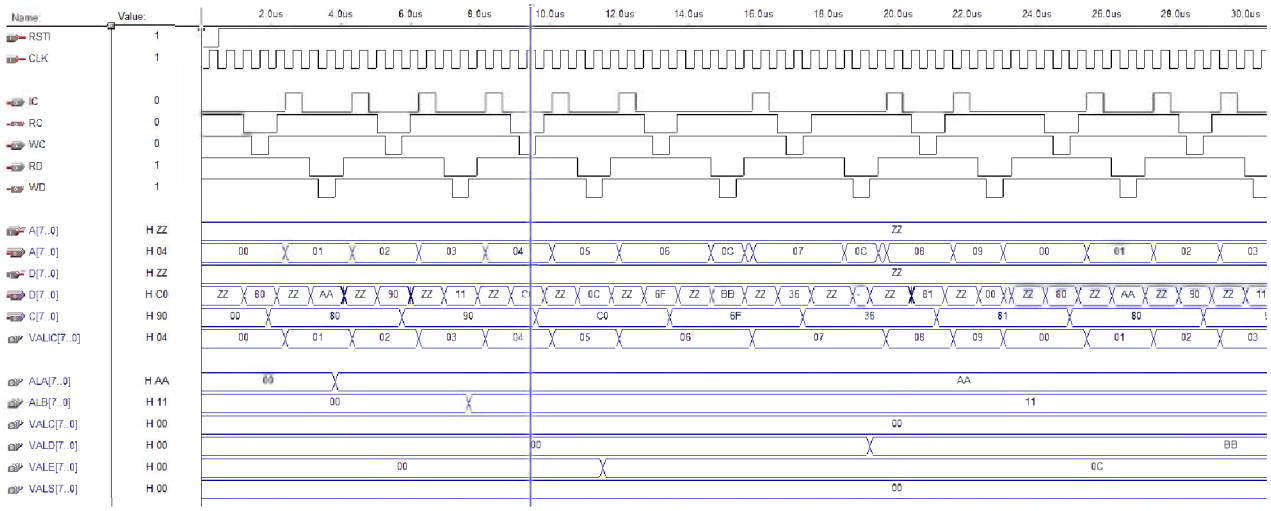


Рисунок 3.11 – Часова діаграма перевірки арифметико-логічного пристрою — ADDAB з непрямою адресацією.

У таблиці 3.5 наведено програму, яку виконує процесор, що на рисунку 3.11 перевіряється програма: *LD A, #AAh / LD B, #11h / LD E, #12 / ADDAB @E / MOV D, @E*. Після виконання: ALA[7..0]=AAh, ALB[7..0]=11h, VALE[7..0]=0Ch (адреса 12). Результат ADDAB записується у RAM[12h]=BBh, що підтверджується значенням VALD[7..0]=BBh після виконання MOV D, @E. Адреса E=0Ch відповідає десятковому значенню 12, збіг між значенням у пам'яті та вмістом D підтверджує коректність непрямої адресації.

3.3.5. Команди введення-виведення (IN/OUT)

Тестова програма реалізує безперервний цикл зчитування стану фізичного порту введення PI та трансляції прочитаного значення на порт виведення PO: IN

B, PI / OUT PO, B / JMP START. На вхід PI[7..0] подається тестова константа 33h для перевірки прозорості каналу введення-виведення.

Таблиця 3.6

Тестова програма перевірки команд IN/OUT

Інструкція	Опкод (hex)	Коментар
IN B, PI	98	B ← стан порту PI = 33h
OUT PO, B	99	PO ← B = 33h
JMP START	81 00	Нескінченний цикл

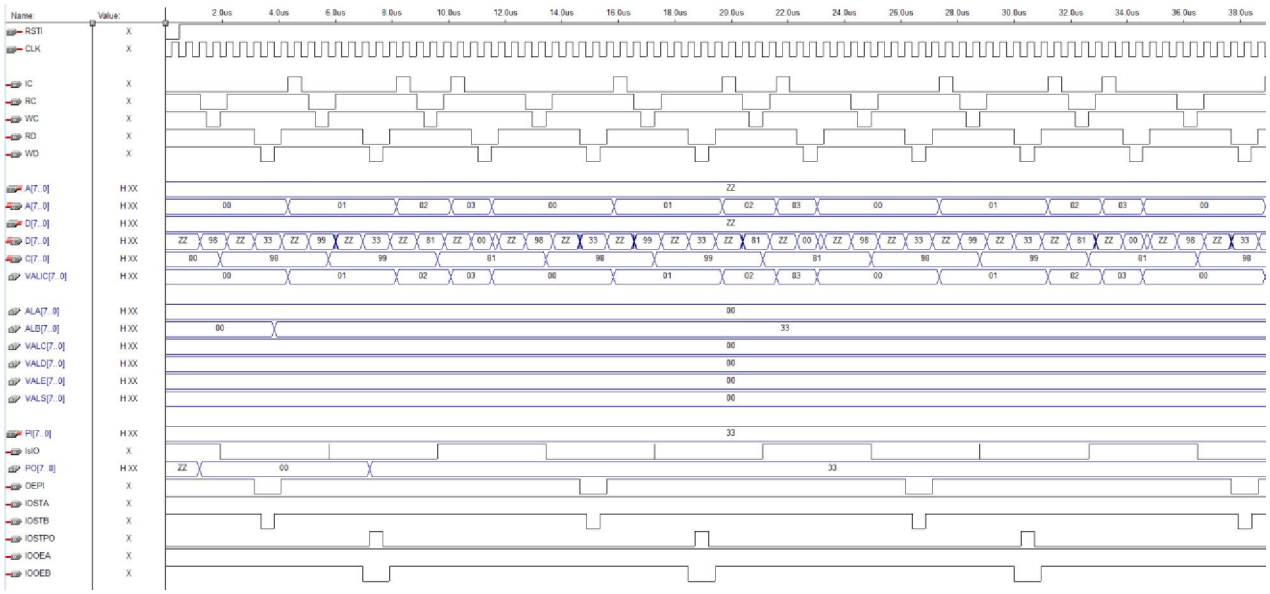


Рисунок 3.12 – Часова діаграма верифікації команд IN/OUT.

У таблиці 3.6 наведено програму, яку виконує процесор, що на рисунку 3.12 після виконання IN B,PI значення PI[7..0]=33h зафіксоване у ALB[7..0]=33h. Після виконання OUT PO,B сигнал IOSTPO формує строб запису, і PO[7..0] набуває значення 33h. Сигнали OEPI та IOOEA/IOOEB активуються у правильній послідовності, що підтверджує коректну роботу обох портів. Цикл виконується безперервно: кожне наступне зчитування та виведення відтворюють той самий результат, що свідчить про детермінізм роботи процесора.

3.4. Верифікація налагоджувального інтерфейсу

Налагоджувальний блок перевіряється в режимі зчитування внутрішніх регістрів без зупинки виконання програми. У пам'ять завантажується коротка програма: LD A, #AAh / LD B, #11h / JMP START. Після виконання обох команд завантаження налагоджувальний міст STM32 активує послідовний перегляд регістрів через шину DC[3..0].

Таблиця 3.7

Тестова програма верифікації налагоджувального інтерфейсу

Інструкція	Опкод (hex)	Коментар
LD A, #AAh	80 AA	A ← AAh
LD B, #11h	90 11	B ← 11h
JMP START	81 00	Нескінченний цикл; налагоджувач зчитує регістри

Примітка: після виконання JMP START лінійка DC[3..0] перемикається 0h→1h→2h; шина DP[7..0] виводить AAh, 11h, 00h (вміст A, B, C).

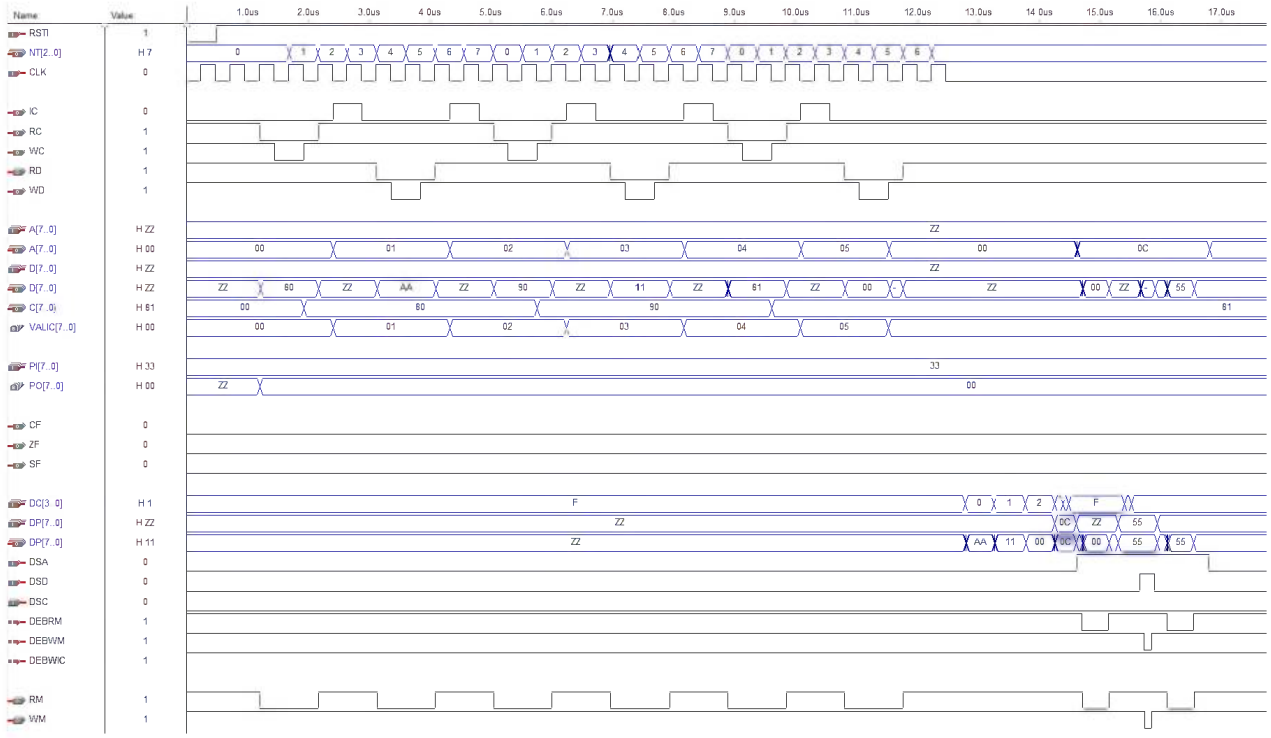


Рисунок 3.13 – Часова діаграма верифікації налагоджувального інтерфейсу.

У таблиці 3.7 наведено програму, яку виконує процесор, що на рисунку 3.13 спостерігається повний цикл роботи системи. Шина NT[2..0] циклічно

перебирає значення 0–7, формуючи правильну часову структуру машинних циклів. Після виконання команд LD: ALA[7..0]=AAh та ALB[7..0]=11h відповідають завантаженим константам. На часовому відрізку ≈10,5–13,5 мкс лінійка DC[3..0] перемикається із значення Fh (режим нормального виконання) до послідовності 0h, 1h, 2h, під час яких шина DP[7..0] почергово виводить AAh, 11h, 00h — вміст регістрів A, B та C відповідно. Сигнали DEBRM та DEBWM генеруються у правильній послідовності для організації читання/запису в налагоджувальному режимі. Результат збігається з очікуваним, що підтверджує коректність роботи налагоджувального інтерфейсу та мосту STM32.

3.5. Зведена оцінка результатів та відповідність меті роботи

На підставі проведеного тестування складено зведену таблицю результатів верифікації всіх компонентів і функціональних груп команд навчальної моделі процесора DPM081.

Таблиця 3.3

Зведені результати верифікації навчальної моделі DPM081

Об'єкт тестування	Тестовий сценарій	Очікуваний результат	Статус
АЛП (ізольований)	ALU_TD: NOT, OR, AND, XOR, ADD	Коректні значення Q[7..0], ZF, CF	Пройдено
Регістр UDC8B	INC, DEC, паралельне завант.	FF→FE→FD (DEC), FA→FB (INC)	Пройдено
RAM8B	Запис/читання (12 байт ROM + 4 байти RAM), скидання CLR	Коректні дані за адресами 0–F	Пройдено
Дешифратор MOV	Коди приймача/джерела 0–4	OEA..OES, MSTA..MSTS	Пройдено
Дешифратор LD/CMP	Коди 0–4 при C[2..0]=0	LDA..LDE, LDRD, CMPA/CMPB	Пройдено
Пристрій керування CPUTG	Повний цикл NT=0..7	RC,WC,RD,WD,EIC,IC у фазах 1–4	Пройдено
Команди MOV	CHECKMOVE: 6 пересилань	A=0Ch, B=19h, E=0Ch, D=19h, C=19h	Пройдено
Команди INC/DEC	CHECKINCDEC: 4 операції	A=0Dh, B=1Ch, C=21h, D=FFh	Пройдено
Команди LD/CMP	CHECKLDCMP: 2 LD + 2 CMP	A=0Dh, B=19h, CF=1 (B<#1Ah)	Пройдено

Продовження таблиці 3.3

Об'єкт тестування	Тестовий сценарій	Очікуваний результат	Статус
АЛП у складі ядра (2)	CHECKALU2: ADDAB + MOV @E	RAM[12h]=BBh, D=BBh	Пройдено
Команди IN/OUT	CHECKPIO: IN B,PI / OUT PO,B	PO=PI=33h, цикл детермінований	Пройдено
Налагоджувальний інтерфейс	CHECKDEBUG1: зчитування A, B, C	DP: AAh, 11h, 00h	Пройдено

З таблиці 3.3 можна побачити, що усі тринадцять тестових сценаріїв, що охоплюють як ізольовані компоненти, так і повне обчислювальне ядро, завершились успішно. Отримані на часових діаграмах значення у всіх випадках точно збігаються з очікуваними результатами, наперед визначеними у коментарях до тестових ASM-програм.

Висновки за розділом 3

Поставлена мета роботи — розроблення навчальної моделі цифрового процесора на основі програмованої логічної матриці — досягнута у повному обсязі. Реалізована модель охоплює всі чотири функціональні блоки системи команд (MOV/INC, ALU, LD/JMP/CMP, IN/OUT/NOP), містить повноцінний пристрій керування з жорсткою логікою, двонаправлені регістри-лічильники, комбінаційний АЛП з восьма операціями та налагоджувальний інтерфейс для взаємодії із зовнішнім мікроконтролером.

Розроблена модель може бути використана у навчальному процесі для демонстрації принципів архітектури процесора фон Неймана, вивчення системи команд та машинного циклу, а також як практична платформа для написання та виконання програм мовою асемблера. Перспективами подальшого розвитку є розширення розрядності командного слова до 12 біт, реалізація апаратної системи переривань та таймерів, а також інтеграція з більш сучасними середовищами САПР, такими як Quartus Prime.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи проведено порівняльний аналіз архітектур процесорів Intel 8080, MOS 6502 та Intel 8051. Встановлено, що Intel 8080 потребує допоміжних мікросхем і має надмірно складну мультиплексовану шину; MOS 6502 відрізняється неочевидною системою режимів адресації; Intel 8051, попри орієнтованість на керуючі застосування, містить надлишкову периферію та Гарвардську організацію пам'яті, що ускладнює реалізацію з нуля.

Виконано детальний аналіз архітектури та принципів функціонування попередньої моделі DPM-8: досліджено структуру восьмибітного командного слова, восьмитактний машинний цикл, склад регістрового файлу та систему з 54 інструкцій. Визначено ключові обмеження моделі — відсутність можливості запису в пам’ять та обмежений набір регістрів, що ускладнює організацію програм.

Розроблено вдосконалену систему команд та повну архітектуру процесора DPM081: розширено регістровий файл до шести регістрів загального призначення (A, B, C, D, E, S), сформовано чотирипольний формат командного слова та систему інструкцій у чотирьох функціональних блоках, створену можливість непрямої адресації у пам’ять та запису в неї, синтезовано автомат керування з жорсткою логікою, комбінаційний АЛП з вісьмома операціями та апаратними флагами ZF і CF, блок введення та блок вивдення розмаплений на пам’яті, та налагоджувальний інтерфейс для взаємодії із комп’ютером.

Реалізовано цифрову схему процесора в середовищі MaxPlus II. Для підготовки тестових програм розроблено таблицю відповідності мнемонік DPM081 і використано асемблер TASM. Проведено тестування моделі шляхом функціональної симуляції тринадцяти тестових сценаріїв на типових наборах команд — арифметичних, логічних, пересилання даних, роботи з пам'яттю та введення-виведення; усі сценарії підтвердили коректність реалізації.

Таким чином, усі п'ять завдань кваліфікаційної роботи виконано в повному обсязі, що свідчить про досягнення поставленої мети — розробку вдосконаленої навчальної моделі 8-бітного цифрового процесора DPM081 на базі

програмованої логічної матриці. У результаті виконання кваліфікаційної роботи отримано процесор, який є кращим за попередника завдяки підтримці запису в оперативну пам'ять, що дозволяє зберігати проміжні результати обчислень; наявності команд інкременту та декременту прямо в регістрах, що дозволяє організовувати лічильник циклу в одному з них; наявності двох регістрів-вказівників E та S з непрямою адресацією, що створює апаратну основу для організації стека та апаратного виклику процедур.

Практичне значення отриманих результатів полягає у тому, що розроблена модель є повноцінною навчальною платформою для вивчення архітектурних принципів фон Неймана, роботи автомата керування та машинного циклу, а також для практичного програмування мовою асемблера із застосуванням сучасного налагоджувального інструментарію. Перспективами подальшого розвитку є розширення розрядності командного слова, реалізація апаратної системи переривань та перенесення проекту на сучасні FPGA-платформи у середовищі Quartus Prime.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Журавель Ю. А., Рева С. М. Модель цифрового процесора. *Вісник Харківського національного університету імені В. Н. Каразіна. Серія: Математичне моделювання. Інформаційні технології. Автоматизовані системи управління*. 2011. № 987, вип. 18. С. 5–18. URL: http://nbuv.gov.ua/UJRN/VKhIMAM_2011_987_18_3 (дата звернення: 20.05.2026).

2. Білинський Й. Й., Ратушний П. М., Мельничук А. О. Цифрова схемотехніка : навчальний посібник. Вінниця : ВНТУ, 2017. 171 с.

3. Intel MCS-51 Microcontroller Family User's Manual. Intel Corporation, 1994. 329 с.

4. MAX+PLUS II Programmable Logic Development System & Software. Reference Manual. Altera Corporation, 1995. 504 с.

5. Intel 8080 Microcomputer Systems User's Manual. Intel Corporation, 1975. 234 с.

6. Tanenbaum A. S., Austin T. Structured Computer Organization. 6th ed. Upper Saddle River : Prentice Hall, 2012. 800 с.

7. MOS Technology MCS6500 Microcomputer Family Hardware Manual. MOS Technology Inc., 1975. 138 с.

8. Patterson D. A., Hennessy J. L. Computer Organization and Design. 5th ed. Waltham : Morgan Kaufmann, 2013. 800 с.

9. Intel 8051 Microcontroller Architecture. *RF Wireless World*. URL: https://www.rfwireless-world.com/tutorials/intel-8051-microcontroller-architecture (дата звернення: 20.05.2026).

10. Masswerk. MOS 6502 Instruction Set. URL: https://www.masswerk.at/6502/6502_instruction_set.html (дата звернення: 20.05.2026).

11. Laplante P. A. Real-Time Systems Design and Analysis. 4th ed. Hoboken : IEEE Press/Wiley, 2011. 560 с.

12. Brown S., Vranesic Z. Fundamentals of Digital Logic with Verilog Design. 3rd ed. New York : McGraw-Hill, 2014. 624 c.

13. Weste N., Harris D. CMOS VLSI Design. 4th ed. Boston : Addison-Wesley, 2011. 864 c.

14. STMicroelectronics. STM32F103 Reference Manual RM0008. ST, 2021. 1136 c.

15. Horowitz P., Hill W. The Art of Electronics. 3rd ed. Cambridge : Cambridge University Press, 2015. 1220 c.

16. Stallings W. Computer Organization and Architecture: Designing for Performance. 11th ed. Upper Saddle River : Pearson, 2019. 864 p.

17. Patterson D. A., Hennessy J. L. Computer Organization and Design: The Hardware/Software Interface. 5th ed. Waltham : Morgan Kaufmann, 2014. 800 c.

18. Harris D., Harris S. Digital Design and Computer Architecture. 2nd ed. Waltham : Morgan Kaufmann, 2012. 712 c.

19. Mano M. M., Ciletti M. D. Digital Design: With an Introduction to the Verilog HDL. 5th ed. Hoboken : Pearson, 2013. 720 c.

20. Maxfield C. The Design Warrior's Guide to FPGAs. Burlington : Newnes, 2004. 576 c.

21. Altera Corporation. MAX V Device Handbook: MV51001-1.2. San Jose: Altera, 2011. URL: https://cdrdv2-public.intel.com/654699/mv51001.pdf (дата звернення: 20.05.2026).

ДОДАТКИ

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних
систем та робототехніки
Максим ХРУСЛОВ
_____
(підпис)
«13» жовтня 2025року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувача першого (бакалаврського) рівня вищої освіти
(першого (бакалаврського) / другого (магістерського))

КОВИЛКІНОЇ КАТЕРИНИ ОЛЕКСАНДРІВНИ
(прізвище, ім'я, по батькові студента)

Спеціальність: 123 «Комп'ютерна інженерія»
Освітня програма «Комп'ютерна інженерія»

1. Тема роботи **НАВЧАЛЬНА МОДЕЛЬ ЦИФРОВОГО ПРОЦЕСОРА НА ОСНОВІ ПРОГРАМОВАНОЇ ЛОГІЧНОЇ МАТРИЦІ**
керівник роботи **Рева Сергій Миколайовнч, к.т.н., доцент зво кафедри,**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від ***№ 4101-5/1245 від 28.05.2026 року.***

2. Строк подання студентом роботи ***30 травня 2026 року***

3. Вихідні дані до роботи:
- Архітектора навчальної моделі DPM-8
- Архітектури процесорів промислового виробництва

4. Перелік питань, які потрібно розробити:
1) Аналіз сучасних підходів до реалізації класичних та сучасних мікропроцесорів на базі програмованих логічних матриць (PLA, PAL).
2) Аналіз архітектури та принципів функціонування попередньої моделі навчального процесора та можливості її вдосконалення.
3) Розробка вдосконаленої системи команд та архітектури нової моделі процесора.
4) Реалізація цифрової схеми процесора в середовищі моделювання MaxPlus II.
5) Проведення тестування на рівні моделювання роботи процесора в середовищі MaxPlus II на типових наборах команд (арифметичні, логічні, робота з пам'яттю).

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Ознайомлення з темою, постановка цілей і завдань кваліфікаційної роботи	15.10.2025 – 10.11.2025
2	Аналіз архітектури простих цифрових процесорів та вибір базового набору команд	11.11.2025 – 30.11.2025
3	Розробка концептуальної та структурної схеми процесора на основі PLA	01.12.2025 – 20.12.2025
4	Реалізація процесора та пам'яті в середовищі розробки MaxPlus II	21.12.2025 – 31.01.2026
5	Розробка тестових програм та перевірка з їх допомогою роботи створеної моделі процесора	01.02.2026 – 30.04.2026
6	Оформлення пояснювальної записки, підготовка презентації та доповіді підготовка до захисту	01.05.2026 – 30.05.2026

5. Дата видачі завдання ***13 жовтня 2025 року.***

Здобувач вищої освіти _____ (підпис) К.О.Ковилкіна (ініціали, прізвище)

Керівник роботи _____ (підпис) С.М.Рева (ініціали, прізвище)

Додаток Б

Технічне завдання

на розробку програмного виробу «Навчальна модель цифрового процесора на основі програмованої логічної матриці»

1. Введення	1.1. Назва програмного виробу — Навчальна модель цифрового процесора на основі програмованої логічної матриці. 1.2. Галузь застосування — навчальний процес за спеціальністю «Комп'ютерна інженерія», вивчення архітектурних принципів побудови ЕОМ.
2. Підстава для розробки	2.1. Навчальний план за спеціальністю 123 «Комп'ютерна інженерія». 2.2. Завдання на кваліфікаційну роботу бакалавра ***№4101-5/255 від 28.05.2026 року***
3. Призначення розробки	3.1. Мета розробки: реалізація вдосконаленої навчальної моделі 8-бітного цифрового процесора DPM081 на базі програмованої логічної матриці у середовищі MaxPlus II шляхом перенесення та розширення архітектури попередньої апаратної моделі DPM-8. 3.2. Призначення: забезпечення можливості спостереження за виконанням інструкцій на рівні сигналів та вивчення принципів роботи архітектури фон Неймана.
4. Початкові дані	Середовище розробки: Altera MaxPlus II. Елементна база: програмовані логічні матриці (PLA/PAL) сімейства MAX. Попередня версія процесора DPM08, розроблена на кафедрі на дискретній TTL-логіці.
5. Технічні вимоги	5.1. Функціональні: реалізація восьмитактного машинного циклу, системи команд з чотирьох

<table>
<tr><td></td><td colspan="2">функціональних блоків, АЛП з вісьмома операціями, портів введення-виведення та налагоджувального інтерфейсу.
5.2. Вимоги до надійності: коректне виконання всіх тестових сценаріїв при функціональній симуляції.
5.3. Вимоги до сумісності: ОС Windows; середовище Altera MaxPlus II.</td></tr>
<tr><td>6. Техніко-економічні показники</td><td colspan="2">Розроблення навчальної моделі процесора DPM081 не потребує значних матеріальних витрат, оскільки реалізується виключно програмними засобами у середовищі MaxPlus II на наявній комп'ютерній базі. Результатом є відкрита навчальна платформа багаторазового використання.</td></tr>
<tr><td rowspan="8">7. Стадії та етапи розробки</td><td>Назва етапу</td><td>Термін виконання</td></tr>
<tr><td>Ознайомлення з темою, постановка цілей і завдань</td><td>15.10.2025 – 10.11.2025</td></tr>
<tr><td>Аналіз архітектур навчальних процесорів та попередньої моделі DPM-8</td><td>11.11.2025 – 30.11.2025</td></tr>
<tr><td>Розробка системи команд та мікроархітектури DPM081</td><td>01.12.2025 – 20.12.2025</td></tr>
<tr><td>Реалізація схеми процесора в середовищі MaxPlus II</td><td>21.12.2025 – 31.01.2026</td></tr>
<tr><td>Розробка тестових програм і верифікація моделі</td><td>01.02.2026 – 30.04.2026</td></tr>
<tr><td>Оформлення записки, підготовка презентації та доповіді</td><td>01.05.2026 – 30.05.2026</td></tr>
</table>

8. Порядок контролю і приймання	1) Перевірку ходу розробки керівник виконує раз на два тижні. 2) Випробування провести відповідно до програми та методики випробувань на базі комп'ютерного класу кафедри. 3) Захист провести на засіданні Атестаційної комісії. 4) Пояснювальну записку подати в електронному та паперовому вигляді.

Виконавець

студент групи К1-41

Ковилкіна К.О.

Замовник

канд. техн. наук, доц.

Рева С.М.

Додаток В

Програма і методика випробувань програмного виробу

«Навчальна модель цифрового процесора на основі програмованої логічної матриці»

1 Об'єкт випробувань

1. Назва програмного виробу: «Навчальна модель цифрового процесора DPM081, реалізована у середовищі Altera MaxPlus II на базі програмованої логічної матриці».

2. Галузь застосування: навчальний процес за спеціальністю «Комп'ютерна інженерія».

3. Перераховані відомості запозичуються з відповідних розділів Технічного завдання (Додаток Б).

2. Мета випробувань

Перевірка відповідності функціональності реалізованої моделі процесора заявленим вимогам технічного завдання — коректності виконання всіх груп команд, роботи АЛП, портів введення-виведення та налагоджувального інтерфейсу.

3. Загальні положення

3.1 Підстави для проведення випробувань

Підставою для проведення випробувань є наказ ***№4101-5/255 від 28.05.2026 року*** про призначення атестаційної комісії

3.2 Місце і тривалість випробувань

Приймальні випробування проводяться на базі комп'ютерного класу кафедри комп'ютерних систем та робототехніки у період роботи атестаційної комісії.

3.3 Обсяг випробувань

Приймальні випробування проводяться в обсязі, що відповідає цій програмі та методиці випробувань.

3.4 Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4. Вимоги до програмного виробу

Модель повинна задовольняти таким вимогам:

1. виконувати машинний цикл із восьми тактів з правильною послідовністю сигналів RC, WC, RD, WD, EIC, IC;
2. коректно виконувати всі чотири групи команд: пересилання (MOV/INC/DEC), АЛП, завантаження/переходи (LD/JMP/CMP), введення-виведення (IN/OUT);
3. забезпечувати читання та запис у зовнішню SRAM через непряму адресацію (@E, @S);
4. передавати стан регістрів через налагоджувальний інтерфейс STM32 по шині DP[7..0];
5. середовище розробки — Altera MaxPlus II під ОС Windows.

5. Вимоги до програмної документації

Програмною документацією до виробу «Навчальна модель цифрового процесора DPM081» вважати:

1. справжнє технічне завдання на розробку виробу (Додаток Б до пояснювальної записки);
2. програму і методику випробувань (Додаток В до пояснювальної записки);
3. опис архітектури та результати верифікації (Розділи 2 та 3 пояснювальної записки).

6. Засоби і порядок випробувань

6.1 Засоби випробувань

Для проведення випробувань необхідний персональний комп'ютер з ОС Windows та встановленим середовищем Altera MaxPlus II. Файли проекту у форматі .gdf і вектори тестів у форматі .scf.

6.2 Порядок проведення випробувань

Випробування проводяться у два етапи:

– ознайомчий (1-й етап) — перевірка комплектності документації та коректності компіляції проекту;

– випробування програмного виробу (2-й етап) — виконання тестових сценаріїв.

Перелік перевірок 1-го етапу:

1. перевірка комплектності програмної документації за критерієм наявності зазначеної в ТЗ;
2. перевірка успішної компіляції проекту у MaxPlus II без помилок;
3. перевірка якості документації на відповідність вимогам ДСТУ 8302:2015.

Перелік перевірок 2-го етапу:

1. відповідність часових діаграм машинного циклу вимогам ТЗ;
2. коректність виконання всіх функціональних груп команд;
3. правильність роботи налагоджувального інтерфейсу.

Порядок запуску симуляції:

1. відкрити файл проекту (.gdf) у середовищі MaxPlus II;
2. виконати компіляцію: Processing → Compile;
3. відкрити Simulator Tool і завантажити файл векторів тестів (.scf);
4. натиснути Start для запуску симуляції;
5. порівняти отримані значення сигналів з еталонними з коментарів до тестової ASM-програми.

Тест 1. Перевірка команд пересилання даних (MOV)

Завантажуються константи у регістри A і B, виконується ланцюг пересилань E←A, D←B, @E←D, C←@E.

Очікуваний результат: A=0Ch, B=19h, E=0Ch, D=19h, C=19h.

Висновок: тест вважається пройденим, якщо значення всіх регістрів на часовій діаграмі збігаються з очікуваними.

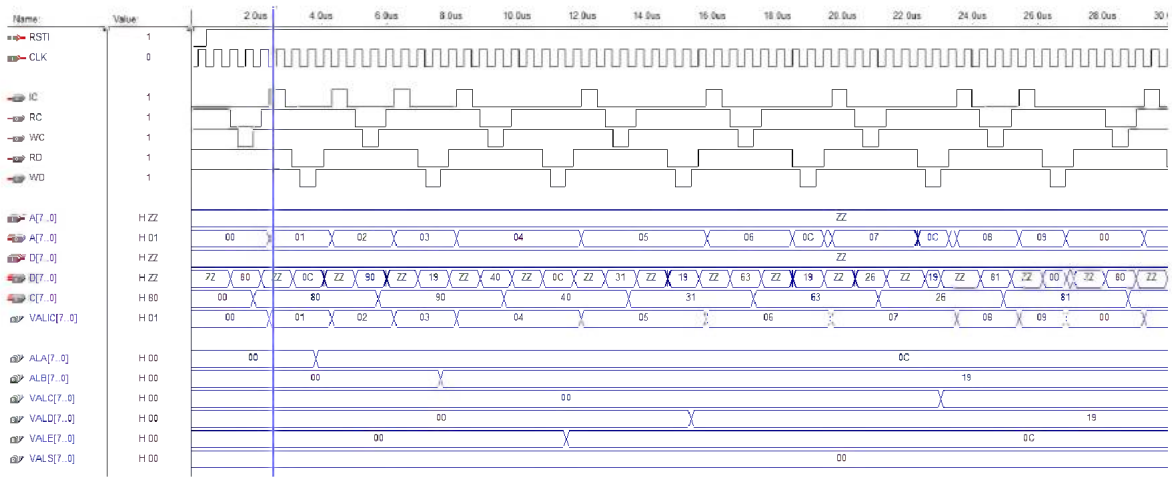


Рис. В.1 – Часова діаграма тесту 1 (MOV).

Тест 2. Перевірка команд інкременту та декременту (INC/DEC)

Завантажуються константи 0Ch, 1Bh, 20h у регістри A, B, C; виконуються INC для кожного та DEC D (від 00h).

Очікуваний результат: A=0Dh, B=1Ch, C=21h, D=FFh.

Висновок: тест вважається пройденим при збігу всіх значень.

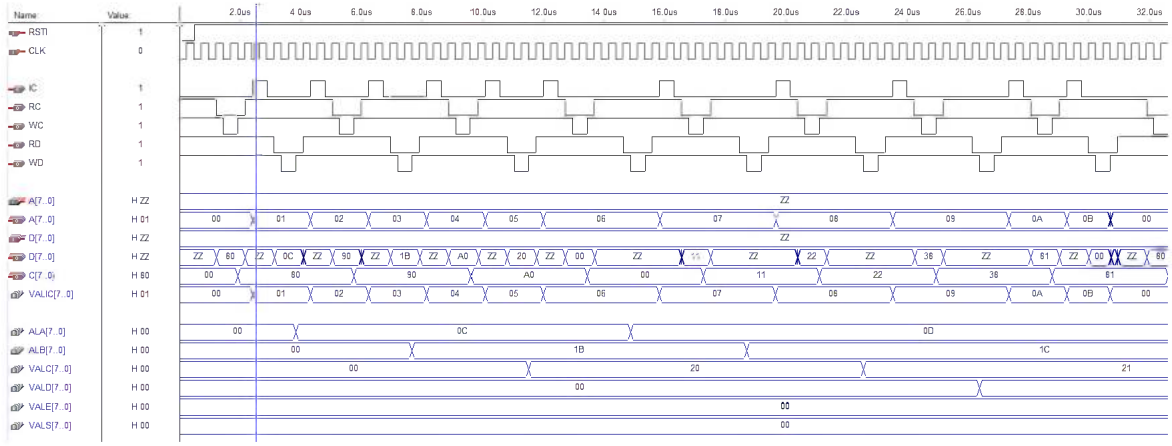


Рис. В.2 – Часова діаграма тесту 2 (INC/DEC).

Тест 3. Перевірка команд LD та CMP

Завантажуються A=0Dh, B=19h; виконується CMP A,#0Dh та CMP B,#1Ah.

Очікуваний результат: CF=1 (B < #1Ah), ZF=0.

Висновок: тест пройдено при CF=1 після другого порівняння.

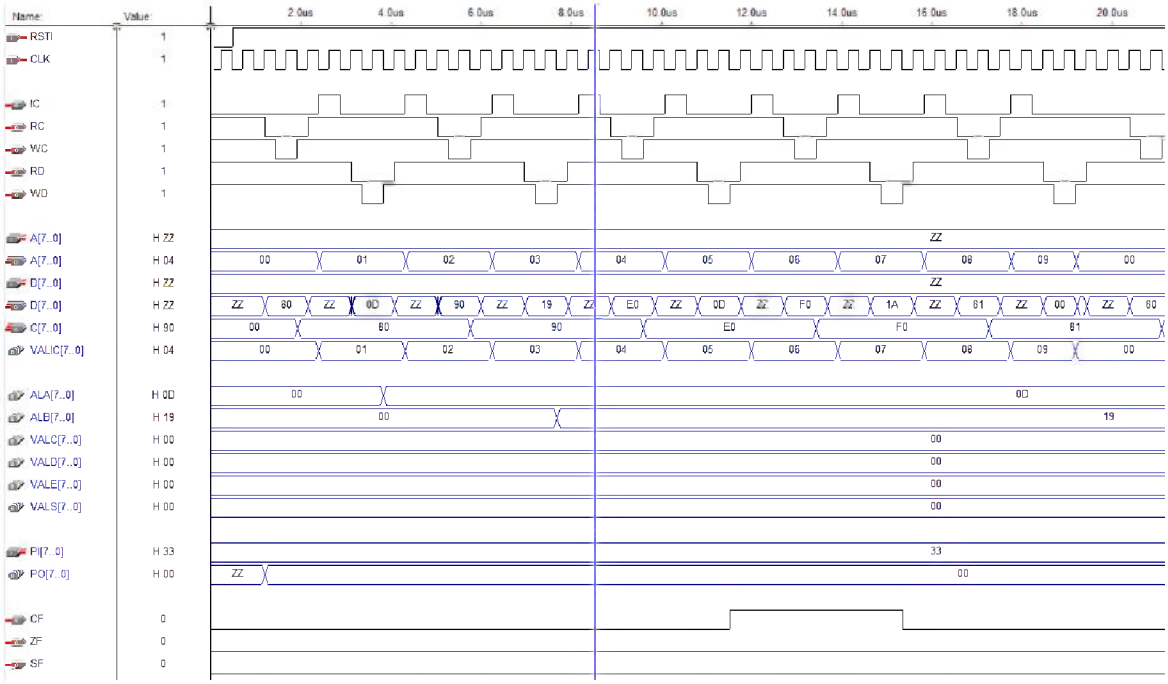


Рис. В.3 – Часова діаграма тесту 3 (LD/CMP).

Тест 4. Верифікація АЛП — CHECKALU1

Завантажуються A=AAh, B=55h; виконуються ANDAB C та ADDAB D.

Очікуваний результат: C=00h, D=FFh.

Висновок: тест пройдено при збігу значень C і D.

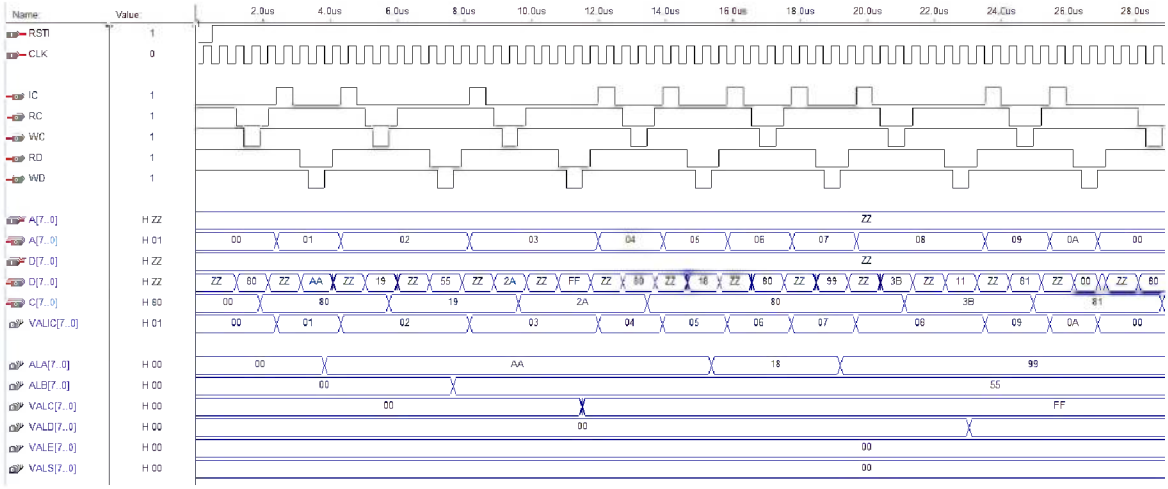


Рис. В.4 – Часова діаграма тесту 4 (CHECKALU1).

Тест 5. Верифікація АЛП — CHECKALU2

Завантажуються A=AAh, B=11h, E=0Ch; виконується ADDAB @E, потім MOV D,@E.

Очікуваний результат: RAM[E]=BBh, D=BBh.

Висновок: тест пройдено при D=BBh.

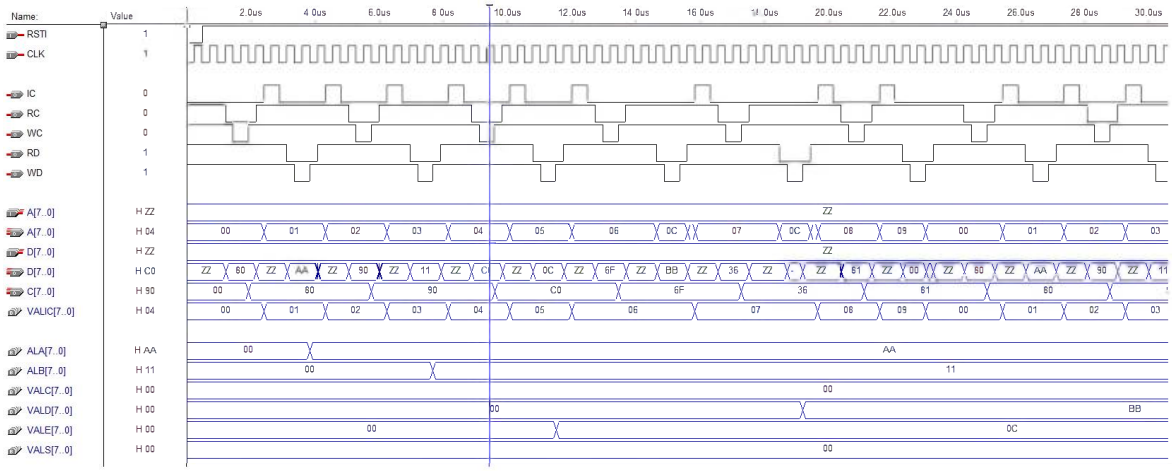


Рис. В.5 – Часова діаграма тесту 5 (CHECKALU2).

Тест 6. Перевірка команд введення-виведення (IN/OUT)

Виконується цикл IN B,PI / OUT PO,B при PI[7..0]=33h.

Очікуваний результат: PO[7..0]=33h.

Висновок: тест пройдено при PO=PI=33h.

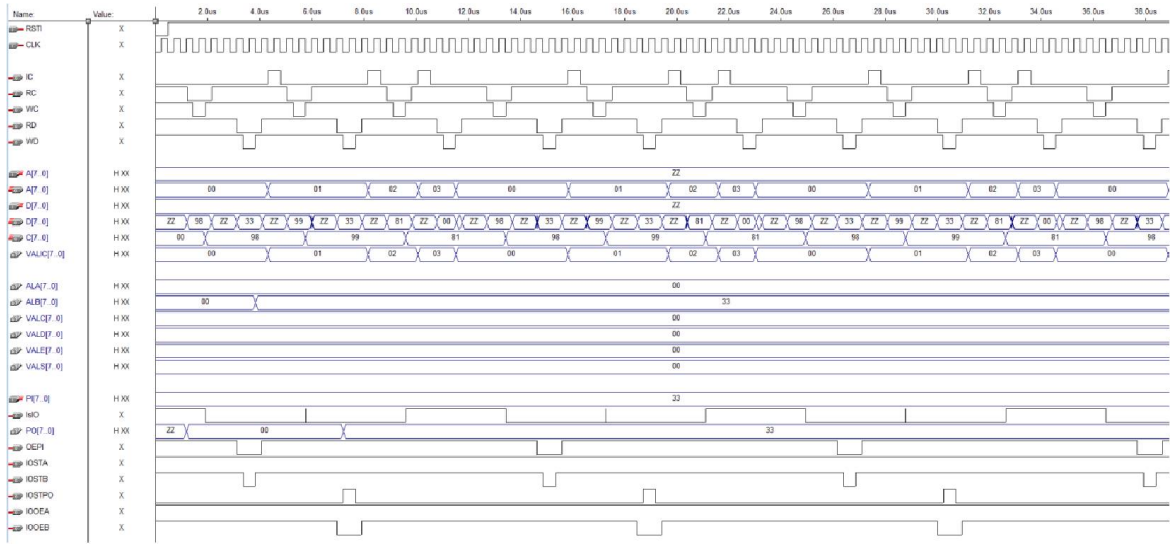


Рис. В.6 – Часова діаграма тесту 6 (IN/OUT).

Тест 7. Верифікація налагоджувального інтерфейсу

Завантажуються A=AAh, B=11h; налагоджувальний міст зчитує регістри через DC[3..0]=0h,1h,2h.

Очікуваний результат: DP[7..0] виводить послідовно AAh, 11h, 00h.

Висновок: тест пройдено при правильній послідовності на шині DP.

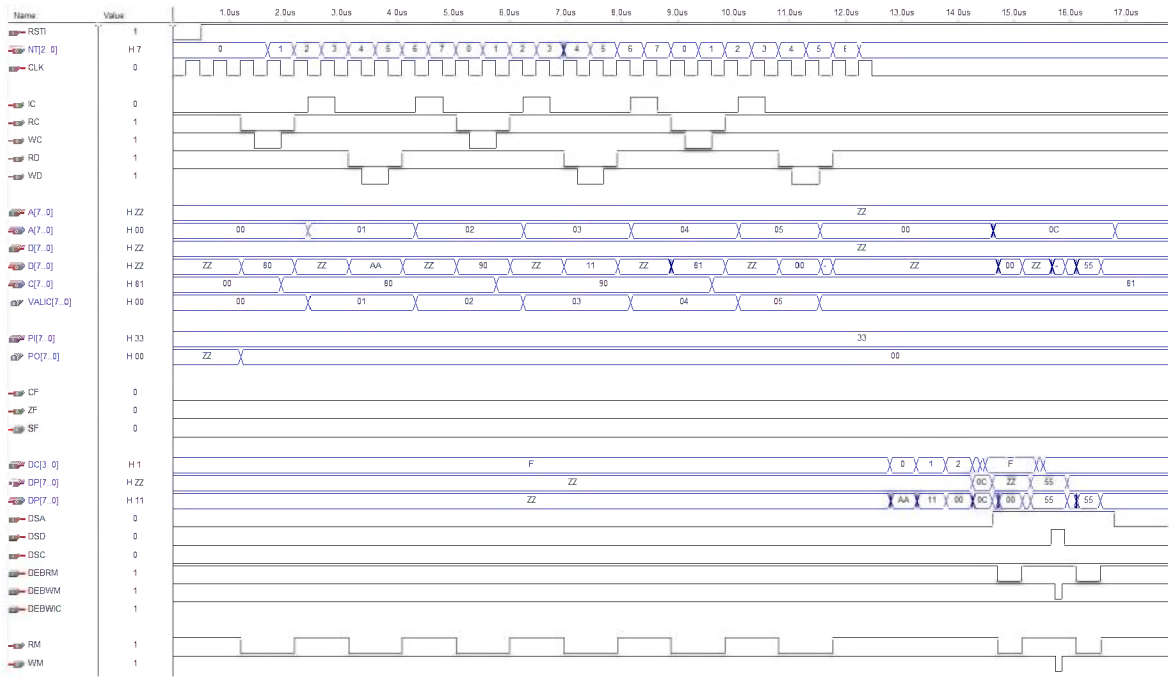


Рис. В.7 – Часова діаграма тесту 7 (Debug).

Загальний висновок: усі сім тестів успішно пройшли випробування. Модель процесора DPM081 відповідає вимогам технічного завдання.

Виконавець: студент групи КІ-41, Ковилкіна К.О. _____

ДОДАТОК Г

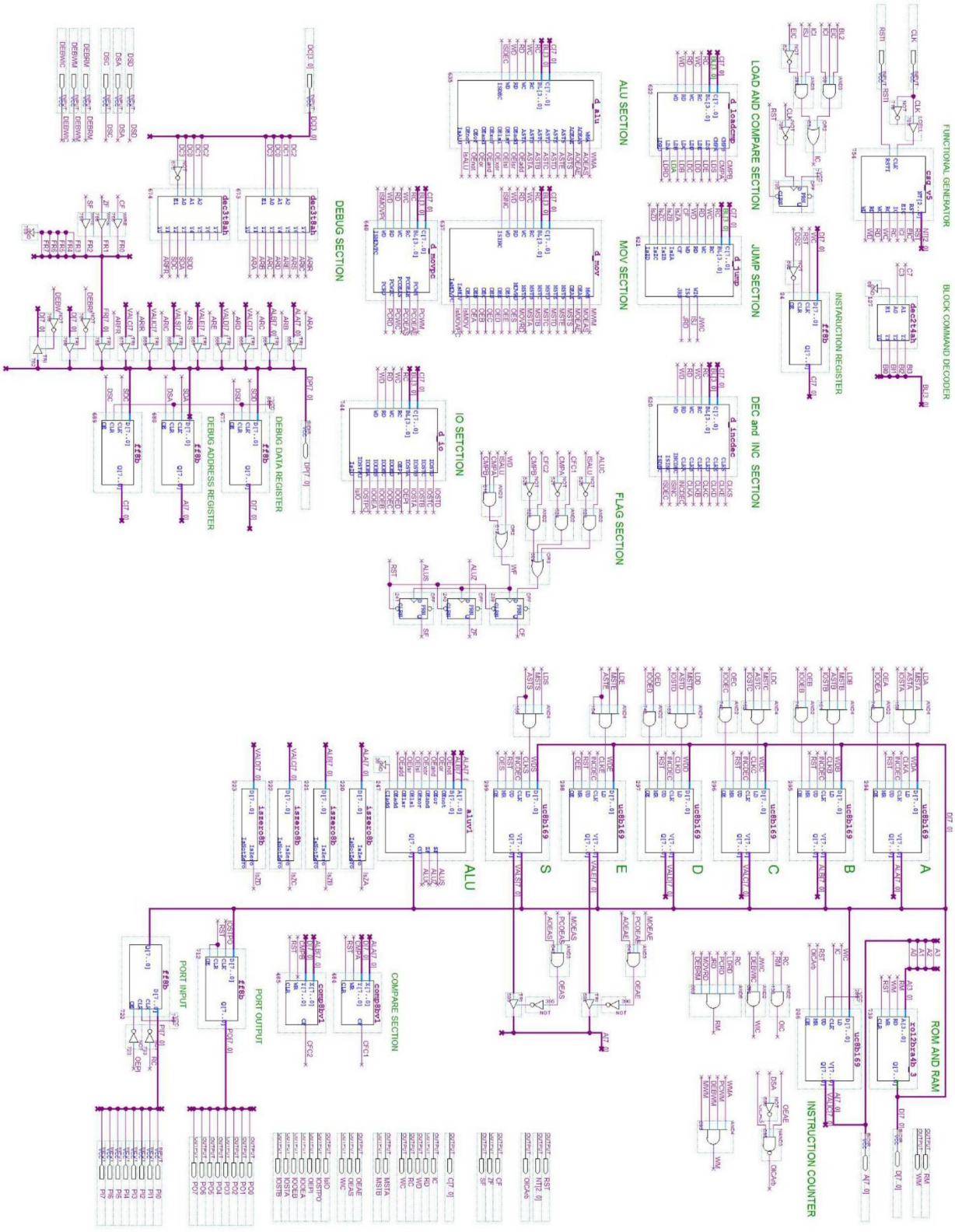


Рис. Г.1 – Схема навчального процесора DPM081.

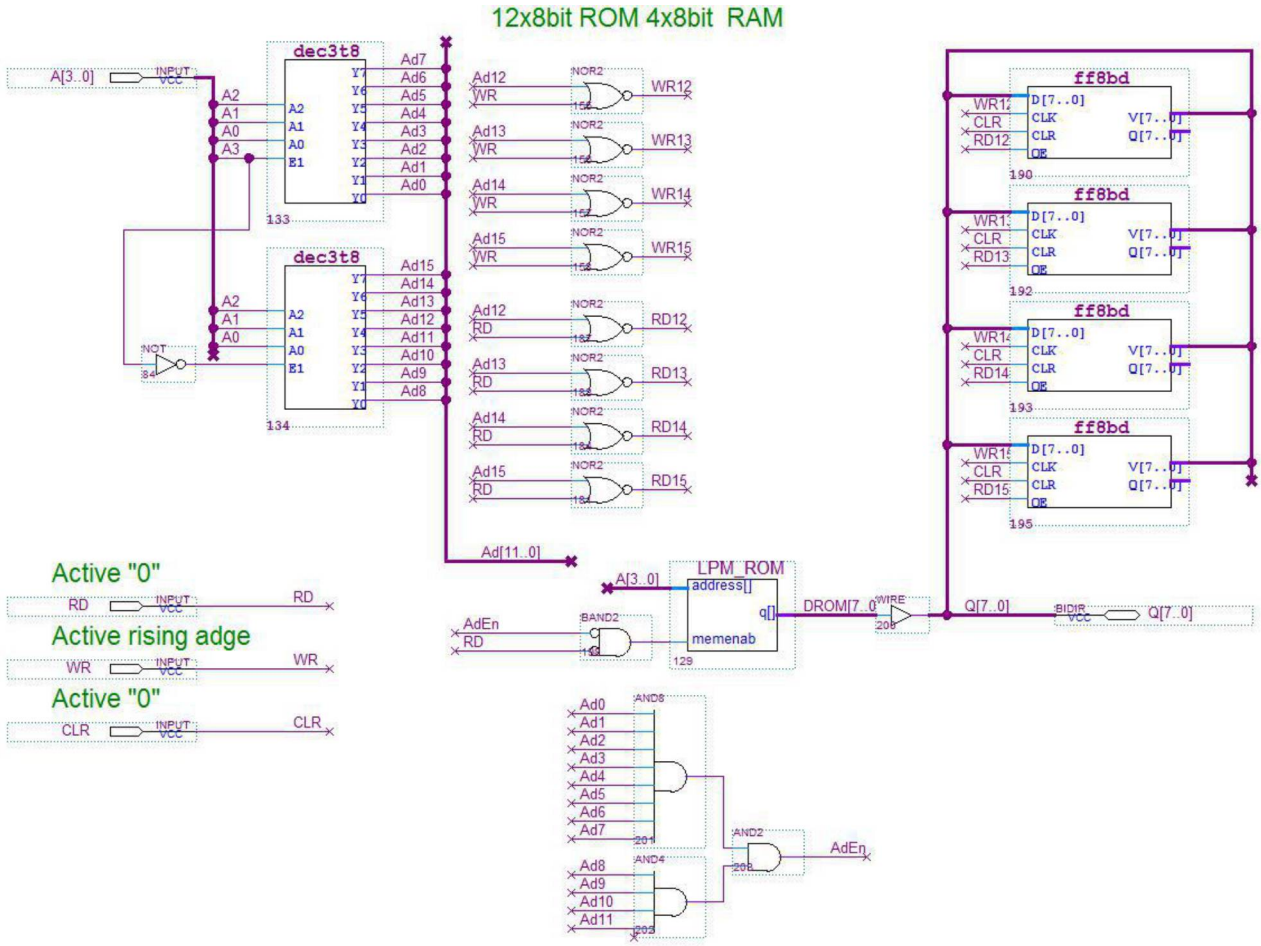


Рис. Г.2 – Схема RAM/ROM.

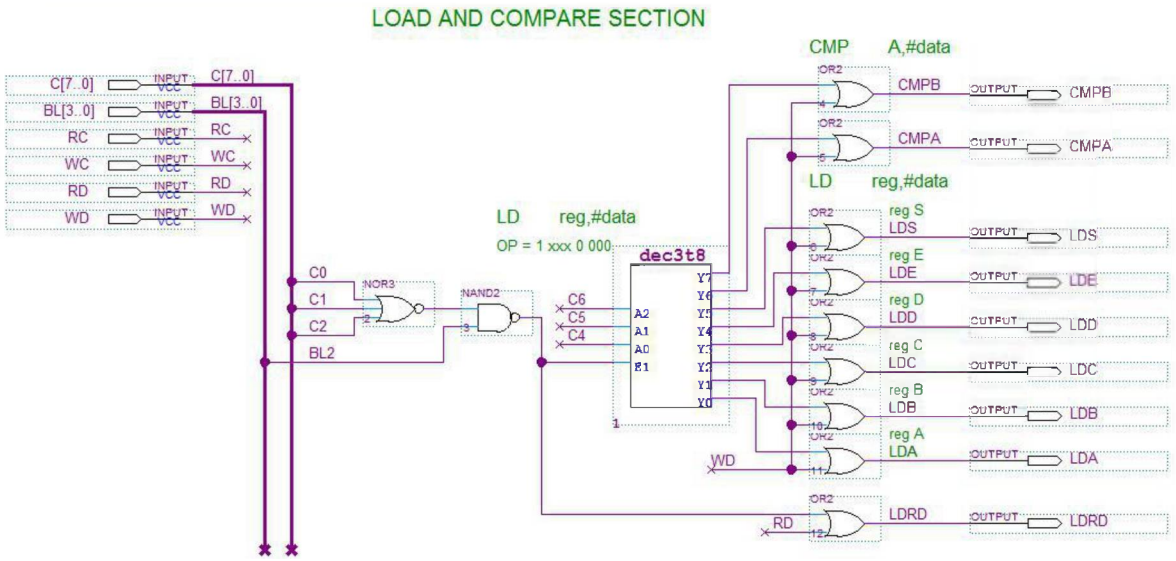


Рис. Г.3 – Схема завантаження та порівняння.