МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

до курсового проекту з курсу "Архітектура комп'ютерів" для студентів спеціальностей "Комп'ютерні системи і мережі" і "Системне програмування" денної і заочної форм навчання

Донецьк ДНТУ 2006

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

до курсового проекту з курсу "Архітектура комп'ютерів"
для студентів спеціальностей "Комп'ютерні системи і мережі" і
"Системне програмування" денної і заочної форм навчання

Затверджено
на засіданні кафедри
Електронних
обчислювальних машин.
Протокол № 9 від
28. 03. 2005 р.

Затверджено
на засіданні учбово-видавничої
ради ДНТУ.
Протокол № 3 від 17. 06. 2005 р.

Донецьк ДНТУ 2006

УДК 681.3

Методичні вказівки до курсового проекту з курсу "Архітектура комп'ютерів" для підготовки фахівців і магістрів за напрямком "Комп'ютерна інженерія" / Укл. В. В. Лапко, Ю.В. Губарь. – Донецьк: Видавництво ДНТУ, 2006. - 84 с.

Посібник містить методичні вказівки і індивідуальні завдання до курсового проекту для спеціальностей за напрямком "Комп'ютерна інженерія". Методичні вказівки охоплюють питання проектування різноманітних архітектур, вузлів і модулів комп'ютерних систем на базі мікропроцесорного комплекта з розрядно – модульною організацією на основі мікросхем серії К1804 (аналог мікросхем серії Am2900 фірми Advanced Micro Devices, США). Розглянуті питання зчитування із оперативній пам'яті та виконання набору команд різноманітної довжини (RR, RX та RS), виконання арифметичних операцій для операндів з рухомою та фіксованою комою, а також методи перевірки виконання арифметичних операцій з використанням контролю за модулем три (або за модулем сім). Наведено приклад VHDL –проекту мікропроцесорної системи на функціональному рівні з використанням інструментального пакету Active – HDL фірми ALDEC, США (з дозволу фірми).

Укладачі доц. Лапко В.В.,
доц. Губарь Ю.В.

Відповідальний
за випуск проф. Святний В.А.

Рецензент проф. Аверін Г. В.

Автори висловлюють подяку студентам Мирошкіну А. Н. (гр.СП-02б), Альохіну О. Г. (гр. СП-01н) та Гоголенко С. Ю. (гр.СП-01н) за сприяння у виданні методичних вказівок до курсового проекту.

ПЕРЕЛІК ОСНОВНИХ СКОРОЧЕНЬ

ЕОМ - електронна обчислювальна машина.

МОП - мікропроцесорний обчислювальний пристрій.

ЦОП - цифровий обчислювальний пристрій.

ОП - оперативна пам'ять.

РЗП - регістровий запам'ятовуючий пристрій.

КОП - код операції.

РК - регістр команд.

МПП - мікропрограмна пам'ять.

ПК - пульт керування.

ФАМ - формувач адреси мікрокоманд.

РАП - регістр адреси основної пам'яті.

ЛАК - лічильник адреси команд.

АЛП - арифметико - логічний пристрій.

БК - блок контролю.

СМЗ - схема формування залишків за модулем три або сім.

БМК - блок мікропрограмного керування.

ДК - додатковий код.

ВСТУП

Посібник містить методичні вказівки, індивідуальні завдання і додатки до курсового проекту з курсу "Архітектура комп'ютерів".

Метою курсового проектування є поглиблення і узагальнення знань спеціалістів за напрямком "Комп'ютерна інженерія". Теми індивідуальних завдань охоплюють питання проектування різноманітних архітектур, вузлів і модулів комп'ютерних систем на основі мікропроцесорного комплекту серії К1804 (процесора, оперативної пам'яті та схем обрамлення). Основна увага у посібнику приділяється системному рішенню питань проектування апаратних засобів та мікропрограмної підтримки одно-, дво- і трьохрівневих конвеєрних архітектур комп'ютерних систем. Курсовий проект містить також завдання науково – дослідного характеру, наукову і довідкову літературу.

Для поглиблення студентами практичних навичок проектування машин різноманітних архітектур у посібнику розглядається приклад розробки мікропроцесорного обчислювального пристрою (МОП) дворівневої конвеєрної архітектури з використанням конкретних ВІС - мікропроцесорних схем серії К1804 (аналог Am2900) для команд змінної довжини (RR, RX та RS) і операндів з рухомою та фіксованою комою. Розглянуті також питання перевірки роботи арифметичного пристрою (окремо для мантис і порядків) з використанням контролю за модулем три.

Важливе місце відведено питанням апаратної та мікропрограмної реалізації арифметичних операцій (підсумовування, множення чисел у додатковому коді та чисел з негативним нулем), а також операції перевірки арифметичних операцій з використанням контролю за модулем три або за модулем сім.

У посібнику приділяється також увага питанням початкового запуску і зупинки роботи МОП, зчитування команд з оперативній пам'яті та їхнього декодування, розробці функціональних та принципових схем блоків МОП, таблиць кодування оперативної та мікропрограмної пам'яті, розрахунку параметрів системи синхронізації.

Посібник містить також перелік літературних джерел та стандартів. Додатки вміщують приклад оформлення технічного завдання, а також лістинги програм симуляції роботи МОП.

Матеріали посібника містять також приклади використання мови апаратного опису VHDL та інструментального засобу Active – HDL для проектування апаратних засобів комп'ютерної техніки, які, як відомо [17 - 24], дають оперативну змогу виконати симуляцію (з наступним тестуванням) на функціональному та структурному рівні., швидко виявити та усувати хиби проектування. Ця частина посібника дає змогу студенту за відведений навчальним планом час практично опанувати методи розробки VHDL – проектів комп'ютерної системи та симуляцію її роботи.

1. ЗАВДАННЯ НА ПРОЕКТУВАННЯ

1.1. Мета курсового проектування

Метою курсового проектування за курсом "Архітектура комп'ютерів" є поглиблення та узагальнення знань в області проектування засобів обчислювальної техніки, і застосування цих знань для системного проектування і симуляції мікропроцесорних пристроїв. Курсове проектування спрямоване також на придбання навичок виконання наукових досліджень та роботи з науковою і довідковою літературою з комп'ютерної інженерії.

Завдання на курсове проектування передбачає розробку і виконання симуляції апаратної частини та програмного забезпечення мікропроцесорного обчислювального пристрою (МОП) для реалізації спеціальної функції. Елементну базу МОП складають ВІС мікропроцесорного комплекту серії К1804, які знаходять широке застосування в галузі обчислювальної техніки [3 – 7].

1.2. Початкові дані на проектування

Курсове проектування виконується згідно з індивідуального завдання. Кожен студент одержує від викладача номер варіанта завдання (N), згідно з яким вибирає з відповідних таблиць свої початкові дані.

Розрахункова формула, яку варто реалізувати на МОП, наведена в табл. 1. При обчисленні арифметичних операцій слід прийняти, що аргументи функцій цієї таблиці зображені у форматі з рухомою комою [1, 3, 8]. Роботу МОП варто організувати в такий спосіб. У головній програмі в залежності від результату порівняння змінних A і B здійснюється виклик першої підпрограми (верхня формула) або другої підпрограми (нижчя формула). Підрахунок значення суми ($\sum x_i$) або добутку ($\prod x_i$) масиву значень x_i варто виконувати з урахуванням можливостей реалізації індексних операцій заданої команди [1, 8, 30].

Формат операндів A, B, x_i та Y наведено на рис. 1. Код зображення порядку P та кількість розрядів порядку n наведені у табл. 2 і табл. 3 відповідно. Код зображення мантиси M та кількість розрядів мантиси m наведені в табл. 4 і табл. 5 відповідно. Підстава характеристики S наведена в табл. 6.

Вибірка алгоритмів множення нормалізованих мантис та метод прискорення операції множення виконується згідно з табл. 7 і табл. 8 відповідно [8, 26, 30]. Вибірка алгоритмів ділення нормалізованих мантис виконуються згідно з табл. 9 [8, 26, 30] . Контроль арифметичних операцій здійснюється згідно з табл. 10 [1, 2, 8, 9].

Вибірка формата базової команди МОП (одноадресної або двоадресної машини) здійснюється згідно з табл. 11 і рис. 2, де наведені наступні позначки: КОП - код команди (наприклад, додавання, множення, безумовний перехід, останов і інші операції); ПЗ - ознака запису (его призначення відбивають табл. 12 і табл. 13); ПА, ПА1 і ПА2 - поля для завдання способу адресації операнда (значення цього поля відбито в табл. 14); ПІ, ПІ1 і ПІ2 - поля для визначення ознак індексації (значення цих полів відбивають табл. 15 і табл. 16) [8, 30].

Тип архітектури МОП слід формувати згіно з даними, які наведено на рис. 3 та в табл. 17 [3, 5]. Варіанти процесорних елементів проекта наведено в табл. 18 [3, 4, 6, 7]. Організація блоку основної (оперативній) пам'яті МОП та тип мікросхем для її побудови наведені в табл. 19 і табл. 20 відповідно [13 – 16]. Спосіб синхронізації регістра ознак (коли він є у складі МОП) відображає табл. 21. Варіанти мікросхем для побудови МОП наведено в табл. 22 [12, 27 – 29] . Способи синхронізації блоків МОП слід обирати з табл. 23 [5].

Розрахункова формула. Таблиця 1.

(N)mod4	Формула Y
0	$Y_0 = \begin{cases} A \cdot \sum_{i=1}^{N+5} x_i - B; & \text{коли } A \geq B \\ A + B, & \text{коли } A < B \end{cases}$
1	$Y_1 = \begin{cases} A \,/\, \sum_{i=1}^{N+6} x_i - B; & \text{коли } A < B \\ A - B, & \text{коли } A \geq B \end{cases}$
2	$Y_2 = \begin{cases} A + \prod_{i=1}^{N+5} x_i - B, & \text{коли } A < B \\ A + B, & \text{коли } A \geq B \end{cases}$
3	$Y_3 = \begin{cases} A \,/\, \prod_{i=1}^{N+7} x_i - B, & \text{коли } A \geq B \\ A - B, & \text{коли } A < B \end{cases}$

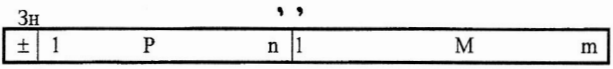


Рисунок 1. Формат операндів із рухомою комою.
(P - порядок числа; M - мантиса числа; Зн - знак числа).

Зображення порядку. Таблиця 2.

(N)mod3	Код порядку
0	ДК (додатковий код)
1	НН (з негативним нулем)
2	ПН (з позитивним нулем)

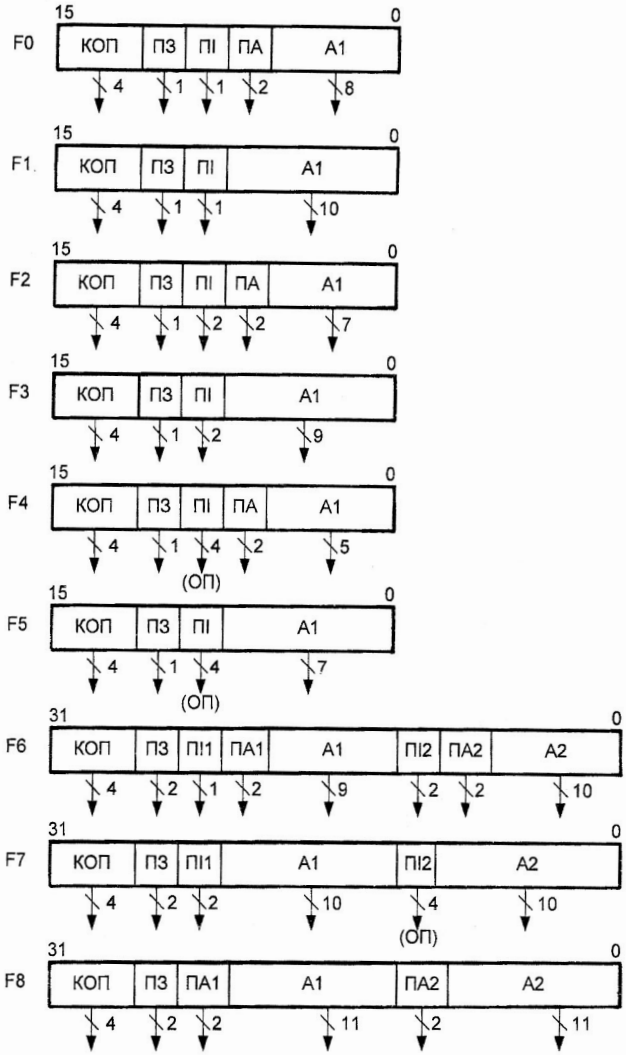


Рисунок 2. Варіанти форматів команд МОП

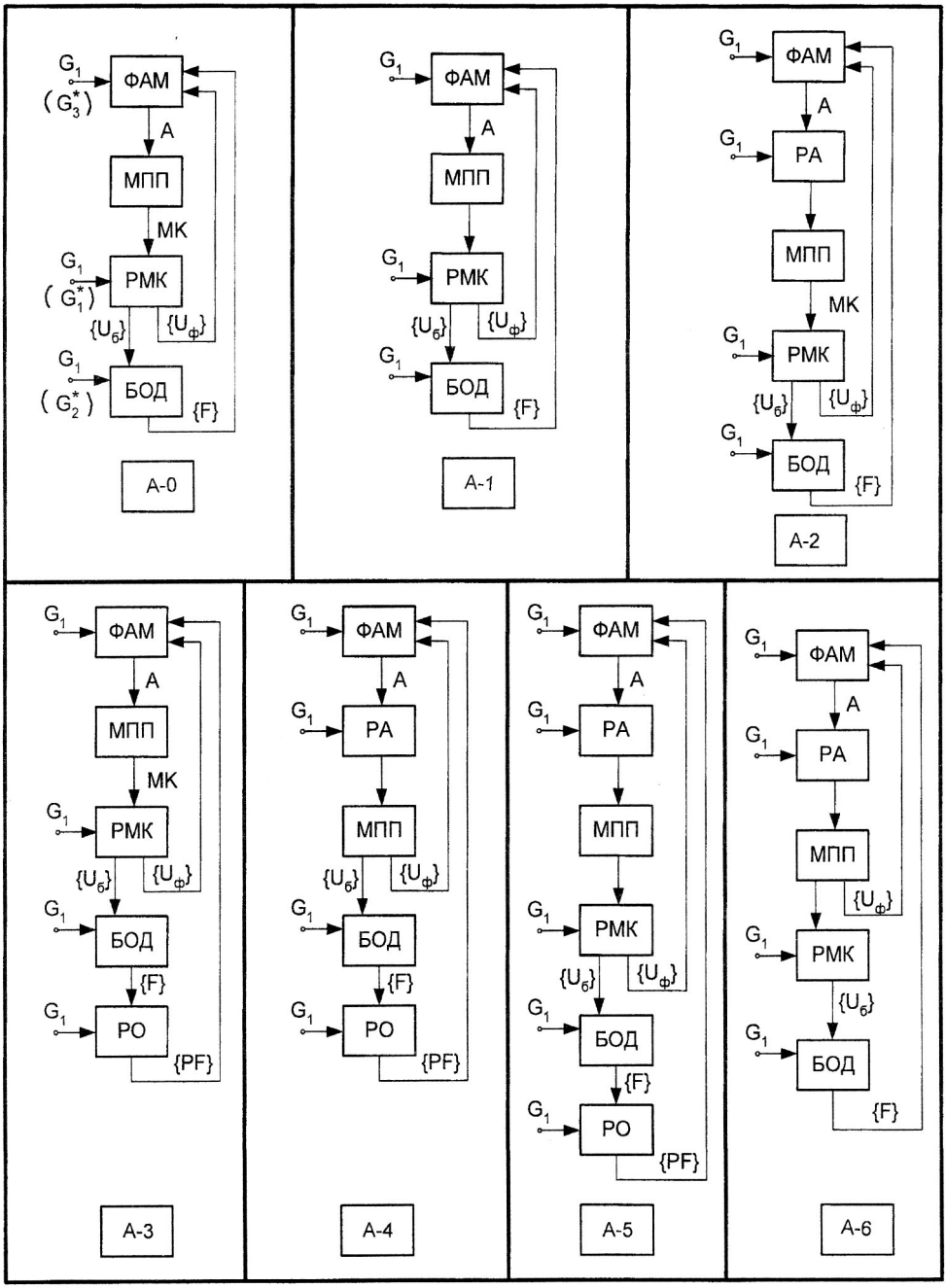


Рисунок 3. Варіанти задаваних архітектур МОП (А0 ÷ А6)

Кількість розрядів порядку. Таблиця 3.

(N)mod4	n
0	5
1	6
2	7
3	8

Зображення мантиси М. Таблиця 4.

(N)mod2	Код мантиси
0	ПК (прямий код)
1	ДК (додатковий код)

Кількість розрядів мантиси. Таблиця 5.

(N)mod5	M
0	10
1	14
2	16
3	20
4	24

Підстава характеристики S. Таблиця 6.

(N)mod3	S
0	2
1	4
2	8

Алгоритм множення мантис. Таблиця 7.

(N)mod4	Алгоритм множення
0	"А" (з молодших розрядів Мк, зсув Мк та Σ праворуч)
1	"Б" (з молодших розрядів Мк, зсув Мк – праворуч , Мн – ліворуч)
2	"В" (з старших розрядів Мк, зсув Мк та Σ ліворуч)
3	"Г" (з старших розрядів Мк, зсув Мк – ліворуч, Мн - праворуч)

Примітки. Мк – множник; Мн - множене; Σ - часткові добутки.

Метод прискорення множення мантис. Таблиця 8.

(N)mod5	Метод множення
0	Лемана
1	Бута
2	Карцева
3	Мак – Сорлі
4	Груповий метод (довжина групи - два розряди)

Алгоритм ділення мантис. Таблиця 9.

(N)mod2	Алгоритм ділення
0	"а" (зсув залишків ліворуч)
1	"б" (зсув дільника праворуч)

Модуль контроля арифметичних операцій. Таблиця 10.

(N)mod2	Модуль контроля
0	Mod 3
1	Mod 7

Формат команд МОП. Таблиця 11.

(N)mod9	Формат команди
0	F0
1	F1
2	F2
3	F3
4	F4
5	F5
6	F6
7	F7
8	F8

Кодування признака адресації (ПА) команди. Таблиця 12.

ПА	Спосіб адресації операнда
0 0	**Пряма** (адреса операнда задається в полі A1 (або A2) команди)
0 1	**Безпосередьня** (в полі A1 (або A2) команди знаходиться операнд)
1 0	**Побічна** (в полі A1 (або A2) команди знаходиться адрес адреса операнда)
1 1	Не використовується (*)

Кодування признака запису результату (ПЗ) в одноадресних командах. Таблиця 13.

ПЗ	Виконуєма дія
0	ОП (A1) → РР (АЛП) ; [РР – регістр результату]
1	РР (АЛП) → ОП (A1)

Кодування признака запису результату (ПЗ) в двоадресних командах. Таблиця 14.

ПЗ	Виконуєма дія
0 0	(A1) * (A2) → (A2)
0 1	(A1) * (A2) → PP
1 0	PP * (A1) → (A2)
1 1	PP * (A1) → PP

Примітка. * - арифметична або логічна операція АЛП.

Кодування признака індексації (ПІ) в форматах команд F0, F1 та F6. Таблиця 15.

ПІ	Формування виконавчої адреси Авик
1	Авик = A1 + IP
0	Авик = A1

Кодування признака індексації (ПІ) в форматах команд F2, F3, F6 та F7. Таблиця 16.

ПІ	Формування виконавчої адреси Авик
0 0	Авик = A1
0 1	Авик = A1 + IP1
1 0	Авик = A1 + IP2
1 1	Авик = A1 + IP3

Примітка. IP1, IP2, IP3 - індексні регістри МОП. В командах форматів F4, F5, F7 індексні регістри розташовані в перших 15 комірках оперативній пам'яті (ОП). Поле ПІ команди в цьому разі задає адрес комірки ОП, яка відбиває індексний регістр.

Варіанти архітектур МОП. Таблиця 17.

(N)mod7	Номер варіанта архітектури МОП
0	A – 0
1	A – 1
2	A – 2
3	A – 3
4	A – 4
5	A – 5
6	A – 6

ВІС МОП. Таблиця 18.

(N)mod4	Процесорні елементі проекта
0	К1804ВС1, К1804ВУ1
1	К1804ВС1, К1804ВУ4
2	К1804ВС2, К1804ВУ1
3	К1804ВС2, К1804ВУ4

Параметри оперативної пам'яті МОП. Таблиця 19.

(N)mod4	Ємність (в кілобайтах) та довжина (в бітах) комірки ОП
0	4 К х 8
1	16 К х 16
2	32 К х 32
3	64 К х 64

Базові елементі оперативній пам'яті. Таблиця 20.

(N)mod7	Мікросхема	Організація МС	Тип МС	Література
0	К565РУ3	16 К х 1	DRAM	13
1	К565РУ5	64 К х 1	DRAM	13
2	К565РУ7	256 х 1	DRAM	14
3	КМ185РУ7	256 х 4	SRAM	14
4	К589РУ01	16 х 4	SRAM	14
5	К537РУ4А	4 К х 1	SRAM	14
6	К565РУ1	4 К х 1	DRAM	15

Способи синхронізації регістра ознак (РО). Таблиця 21.

(N)mod2	Синхронізація РО
0	Керована
1	Не керована

Схеми обрамовування ВІС МОП. Таблиця 22.

(N)mod4	Типи мікросхем	Література
0	К155ТМ2, К155ИМ1, К155ИЕ6, К155ЛА3	12, 27, 29
1	К155ТМ5, К155ИМ2, К155ИЕ7, К155ЛА2	12, 27, 29
2	К155ТВ1, К155ИМ3, К155ИЕ5, К155ЛР1	12, 27, 29
3	К155ИР1, К155ИМ2, К155ИЕ2, К155ЛР3	12, 27, 29

Способи синхронізації регістрових схем МОП. Таблиця 23.

(N)mod2	Тип синхронізації
0	Однофазна
1	Багатофазна

Технічне завдання на курсове проектування (додаток 1) під керівництвом викладача складається на першій консультації (занятії), підписується студентом і керівником проекту, а також затверджується завідуючим кафедрою ЕОМ.

У курсовому проекті підлягають розробці: алгоритми обчислень для заданої функції в зазначеному форматі команд, граф – схеми мікропрограм вибірки і виконання команд арифметичних операцій, функціональні і принципові електричні схеми операційного та керуючого блоків цифрового обчислювального пристрою (ЦОП). Крім

того, повинні бути виконані розрахунки параметрів системи синхронізації та основних технічних характеристик мікрообчислювача, а також побудовані часові діаграми його роботи у типових режимах.

У курсовому проекті з метою верифікації прийнятих рішень бажано розробити VHDL - модель мікрообчислювального пристрою з використанням пакета Active – HDL [17 – 24]. Провести симуляцію на функціональному рівні блоків ЦОП (конкретні варіанти модифікації узгоджуються з керівником). При проведенні модельних експериментів може бути застосована при бажані також програма Electronics Workbench (електронна лабораторія) версії 6.02 (MultiSim) [25].

1.3. Загальні вимоги до курсового проекту

Курсовий проект мусить містить пояснювальну записку і графічний матеріал. Пояснювальна записка повинна включати наступні обов'язкові елементи:

- Титульну сторінку.
- Підписане та затверджене технічне завдання.
- Реферат (з великим штампом).
- Зміст.
- Вступ.
- Основну частину, до складу якої повинні входити наступні розділи:
    1. Програмна реалізація задаваної формули в форматі задаваної команди і приклади обчислень.
    2. Розробка функціональної схеми мікрообчислювача.
    3. Методика запуску МОП у роботу та його зупинка.
    4. Розробка алгоритмів виконання задаваних арифметичних операцій.
    5. Розробка таблиць кодування.
    6. Розробка функціональних і принципових схем мікрообчислювача.
    7. Розрахунок параметрів блоку синхронізації та основних технічних характеристик МОП.
    8. Часові діаграми роботи МОП у типових режимах.
- Висновки.
- Список використаних літературних джерел.
- Додатки (програми симуляції арифметичних операцій та результати симуляції алгоритмів).

Графічна частина курсового проекту повинна включати наступні обов'язкові креслення:

- Схема електрична функціональна мікрообчислювача (формат А1).
- Схеми алгоритмів обчислення задаваної функції на рівні регістрів процесора (формат А1).
- Схема електрична принципова вузлів МОП (формат А1).

Усі креслення і пояснювальна записка вважаються конструкторською документацією і повинні мати позначки відповідно до діючіх стандартів ЄСКД і ЄСПД [32, 33].

До записки необхідно додати архів файлів проекту arj, короткий опис яких повинен містити додаток.

Проекти, які не відповідають чинним вимогам, до розгляду не приймаються.

2. ОСНОВНІ ЕТАПИ ПРОЕКТУВАННЯ

2.1. Структура складу операцій процесора. Адресація операндів у командах

Для виконання довільної програми система команд ЕОМ повинна складатися по-перше, із арифметичних і булевських операцій з використанням як чисел з фіксованою комою, так і арифметичних операцій для чисел із рухомою комою. Для завдання необхідної послідовності як команд, так і програми в цілому, множина команд процесора мусить мати також групу команд передачі управління (для виконання операцій управління).

В кожній команді операція задається двійковим кодом КОП (КОП – код операції). Адресна частина команди вказує безпосередньо адресу операндів, які приймають участь в операції, або алгоритм, як сформувати адрес. У командах управління коди адреси вказують, куди (в яку комірку ОП) треба передати управління.

Припустимо, що код операції та ознака формату (Ф) в кожній команді вмістить вісім розрядів - один байт (рис. 4), а адресна частина має різну довжину: байт (RR) або три байта (24 біта). Цей спосіб адресації дозволяє адресувати ієрархічну структуру пам'яті, яка складається з великого об'єму основної оперативної пам'яті (ОП) і регістрового швидкодіючого запам'ятовуючого пристрою (РЗП) відносно малого об'єму.

Припустимо, що РЗП складається з 16 безпосередньо адресуємих регістрів, яким присвоєні номера від 0000 (0) до 1111 (15). З метою економії часу регістри будемо використовувати при роботі програми для зберігання початкових операндів і результату арифметичних та булевських операцій, а також для зберігання базової адреси і індекса (константи модифікації адреси), які використовуються для формування виконавчого адреса ОП. Номер регістра РЗП задається в командах 4 - х розрядними двійковими числами в полі РОН. При виконанні арифметичних операцій з рухомою комою два суміжних регістра використовуються спільно. В цьому вивадку у команді вказується адреса регістра, де зберігається мантиса. При цьому порядок цієї мантиси обов'язково зберігається в слідуючому регістрі РЗП. При цьому конкретний тип даних РЗП залежить від поточної команди процесора.

Адреса основної оперативної пам'яті (ОП) в загальному випадку обчислюється значно складніше. Звертання до неї може виконуватися з використанням відносної адресації. За такої адресації ОП умовно розподіляється на секції, кожна з яких складається, наприклад, з 2048 слів. Початок кожної такої секції визначається адресом її першого слова. Цю адресу будемо називати **базовою**. Секції можуть бути в довільному місці ОП, починаючи з довільного слова. В зв'язку з цим для завдання всіх можливих базових адресів для ОП, наприклад, об'єму 1М слів необхідно 20 біт. В межах секції адреса кожного слова відносно базової визначається 12 – розрядним двійковим числом (у секції об'ємом 2048 слів), яке називається зсувом (зміщенням) адреси поточного слова відносно адреси початку секції (базового адреса) і позначається буквою D.

З використанням бази B та зміщення D виконавча адреса $A_{ВИК}$, по якій виконується звертання до ОП, обраховується як сума по mod 2^{20} кода базової адреси і кода зміщення. Підсумовуються ці коди як цілі числа без знаку (формат int, але без знаку).

В залежності від кількості команд і обсягу (об'єму) даних програма може бути розташована в межах однієї секції або в межах декількох секцій. Якщо програма не виходить за межи однієї секції, то для формування виконавчої адреси основної пам'яті для всіх команд достатньо буде однієї базової адреси. Така адреса розміщується в одному із регістрів РЗП, який в цьому випадку називається регістром бази. Базова адреса займає у такому регістру з 19 – ої до 0 – ої позиції регістра бази. Для вказівки адреси ОП в команді RX (рис.4) в адресній частині розміщується 12 – розрядний двійковий код зміщення D і 4 – розрядний код адреси B (який визначає номер регістра РЗП, де знаходиться базова адреса.

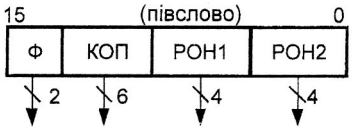
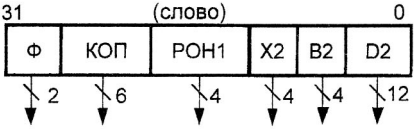
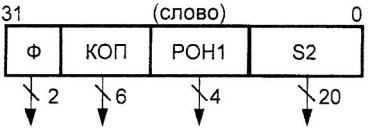
Умовне зображення формату	Формат і структура команди	Двійковий код ознаки формату команди (Ф)
RR		01
RX		10
RS		11

Рисунок 4. Формати команд, їх умовне зображення та структура команд

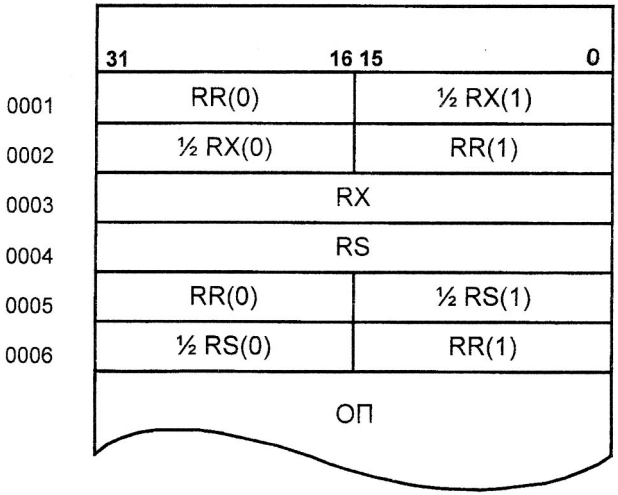


Рисунок 5. Схема компактного розміщення команд змінної довжини, які кратні півслову, в 32-розрядних комірках ОП:
(0), (1) - номер відповідно старшого та молодшего півслова комірки ОП;
0001, 0002, ... - номер (адреса) комірки пам'яті;
0001(0), 0001(1), ... - адреса півслова ОП.

Якщо програма розміщується в декількох секціях пам'яті, то в РЗП зберігаються адреса декількох секцій. При цьому в конкретній команді RX вказується той регістр РЗП, який зберігає поточну базу програми. Як базові можуть бути використані будь-які регістри РЗП, крім нульового. Якщо в команді RX вказується B = 0, то припускається, що базова адреса дорівнює нулю незалежно від вмісту регістра з номером 0000 (0).

Припустимо, що будь-які операнди (з фіксованою або рухомою комою) в ОП розміщуються в одній комірці і мають довжину, яка дорівнює одному слову. При цьому адреса операндів явно вказується в адресній частині будь якої із трьох типів команд (рис.4). При цьому в двомістних операціях припускається, що результат розміщується на місці першого операнда. В залежності від місця і способу адресації початкових операндів, формати команд позначаються як RX, RS та RR. Форматний код RR позначає операції регістр – регістр; формати RX та RS - операції, в яких один із операндів знаходиться в ОП. При цьому для визначення операнда із РЗП в будь-якій команді (RR, RX, RS) використовується пряма адресація – в командах вказуються 4 – розрядна адреса регістра РЗП.

Другий операнд в команді RX (рис. 4) знаходиться в ОП за адресою

$$A2_{ВИК} = <X2> + <B2> + D2, \qquad (1)$$

де $<X2>$ - вміст регістра РЗП з номером X2;

$<B2>$ - вміст регістра РЗП з номером B2.

Модифікація адреси шляхом підсумування індекса X2 використовується для організації циклічних обчислювальних процесів. При цьому індекс, як і базова адреса, до початку виконання програми повинен бути в будь – якому регістрі РЗП, який в цьому разі отримує назву регістра індекса.

Команда формату RS відноситься до типу регістр – пам'ять. При цьому модифікація адреси ОП тут не передбачається – в полі S команди вказується прямий адрес комірки ОП. При цьому необхідно враховувати слідуюче. Так як виконавча адреса завжди формується до початку виконання операції, в принципі вміст одного і того же регістра РЗП в одній команді може бути використаний як для формування виконавчої адреси, так і операнда.

Повну інформацію про тип операції, яка повина бути виконана, і тип даних вміщує код операції (КОП) команди. Команди RX та RS мають кожна довжину у два півслова, а RR - потребує менше розрядів - тількі одне півслово. Команди для більш економної трати об'єму ОП розміщуються, як правило, в ОП компактно (рис.5). В цьому випадку якщо довжина комірки дорівнює 32 біта, адреса команди складається з двох частин: адреси комірки ОП та адреси півслова в цій комірці. При цьому довга команда може починатися із старшої половини комірки, тобто з нульового півслова і займати всю комірку пам'яті і, навпаки, якщо "голова" команди починається з молодшого півслова комірки ОП, тобто команда починається з першого півслова комірки ОП, для зчитування "хвоста" команди необхідно звертатися до слідуючої по порядку комірки і використовувати її нульове півслово (старше півслово в такому випадку). Наприклад, початок команди RX може мати адресу 0001 (1), а кінець - 0002 (0). Очевидно, що для формування адреси молодшої частини команди RX 0002 (0) необхідно до адреси старшої половини RX 0001 (1) додати (+1), тобто виконати операцію: 0001(1) + 1 = 0002(0).

З використанням розглянутих форматів команд побудуємо, наприклад, програму для обчислення добутку:

$$\mathbf{Y} = \prod_{i=1}^{5} \alpha_i, \qquad (2)$$

де α_i - масив змінних, які мають формат з рухомою комою (рис.1).

При програмуванні для обчислення Y використуємо схему:

$$\mathbf{Y = \alpha_5;}$$
$$\mathbf{Y = Y * \alpha_i \ (i = 4 \div 1).} \qquad (3)$$

Для конкретності припустимо, що до початку виконання програми розподіл ОП та призначення регістрів РЗП і комірок ОП має вигляд, який наведено в табл. 24. З урахуванням цього розподілу пам'яті та уведенної системи команд один із варіантів програми обчислення Y наведено у табл. 25. При побудові програми передбачалося, що РЗП вміщує 16 регістрів, а об'єм ОП вміщує 2^{20} слів (1Мс) і відповідно 2^{21} півслів (2Мпс). При побудові програми також передбачено, що команда пересилки даних ОП → РЗП з рухомою комою (F) записує мантису числа комірки ОП та порядок відповідно у регістри R0 та R1. Припустимо також, що мнемоничне зображення кода операції цієї команди відображається як MOV_FMR. Таким чином, число з рухомою комою розміщується цією командою у двох суміжних регістрах РЗП. З другого боку, команда пересилки даних із рухомою комою (F) РЗП → ОП (команда запису MOV_FRM) формує одне слово ОП на основі вмісту відповідних двох суміжних регістрів РЗП. Наприклад, на основі вмісту регістрів R0 та R1 (R0.R1) команда MOV_FRM може формувати число із рухомою комою (F) у 35 комірці ОП (рис. 6).

Призначення комірок ОП і регістрів РЗП та початкове значення змінних. Таблиця 24

Адреса комірки ОП	**31 Вміст комірки ОП 0**	**Примітки**
000 00	0	Const (повний нуль)
000 31	α_1	Елемент масива (float)
000 32	α_2	Елемент масива (float)
000 33	α_3	Елемент масива (float)
000 34	α_4	Елемент масива (float)
000 35	α_5	Елемент масива (float)
000 36	000000 30	Базова адреса (B) масиву даних та програми (int)
000 38	4	Початкове значення індексного регістра (X2) РЗП (int)
000 39	*	Результат обчислення Y (float)
Адреса регістра РЗП	**Призначення регістра РЗП**	**Примітки**
0000 (R0)	Мантиса першого співмножника та в подальшому мантиса поточного Y (*)	У додатковому коді (m = 27)
0001 (R1)	Порядок першого співмножника та в подальшому порядок поточного Y (*)	З негативним нулем (n = 5)
0011 (R3)	Індексний регістр та лічильник циклу (*)	Модуль коду (індекс)
0100 (R4)	Регістр бази (*)	Модуль коду (бази)
1110 (R14)	Мантиса другого співмножника (при виконанні програми) (*)	У додатковому коді (m = 27)
1111 (R15)	Порядок другого співмножника в команді (при виконанні програми) (*)	З негативним нулем (n = 5)

Примітка. * - стан регістру невизначений до початку програми.

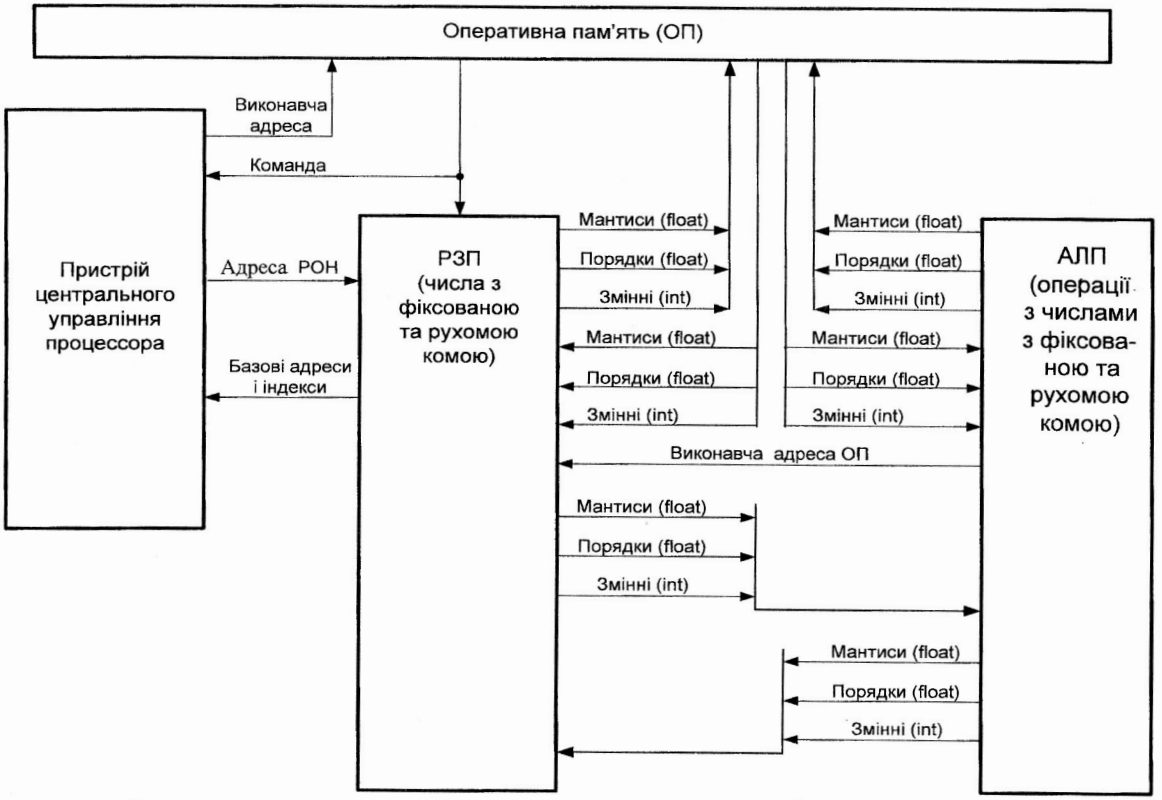


Рисунок 6. Узагальнена структура інформаційних зв'язків пристроїв МОП

Програма обчислення функції Y для гіпотетичної ЕОМ. Таблиця 25.

Адреса команд в ОП (адреса півслова)	**Ф**	**КОП**	**А1**	**А2**	**Примітки**
000 40 (0)	RS	MOV_FMR	0000 (R0)	S = 000 35	M (α_5) → R0 П (α_5) → R1
000 41 (0)	RS	MOV_IMR	0011 (R3)	S = 000 38	4 (X2) → R3
000 42 (0)	RS	MOV_IMR	0100 (R4)	S = 000 36	30 (B) → R4
000 43 (0)	RX	MOV_FMR	1110 (R14)	X2 = 0011 (R3) B2 = 0100 (R4) D2 = 000_{16}	M (<X2 + B2 + 0>) → R14 П (<X2 + B2 + 0>) → R15
000 44 (0)	RR	MUL_F	0000 (R0)	1110 (R14)	R0.R1 * R14.R15 → R0.R1
000 44 (1)	RR	DEC_I	0011 (R3)	*	R3 – 1 → R3
000 45 (0)	RS	JMP_Z	*	S = 000 43	УП = { 00046, Z = 1; 00043, Z = 0
000 46 (0)	RS	MOV_FRM	0000 (R0)	S = 000 39	R0.R1 → 000 39
000 47 (0)	RR	STOP	*	*	Зупинка програми

Примітки. DEC_I - декремент коду з фіксованою комою (int);
S – пряма адреса ОП команди RS;
RX (B2) = R4 (B2 = 30); RX (X2) = R3 [X2 = (4 ÷ 1)]; RX (D2) = 000_{16}.

2.2. Аналіз задаваної формули та приклади її обчислення

Припустимо, що для обчислення добутку (2) потрібно побудувати цифровий обчислювальний пристрій (ЦОП) на базі процесорних секцій ВС1 та ВУ4. При цьому припустимо, що порядок P відображається у форматі з негативним нулем (n = 5), мантиса M - у додатковому коді (m = 27), а підстава характеристики S дорівнює двом.

По-першому, розглянемо приклад виконання операції множення з рухомою комою.

Нехай співмножники відповідно рівні: A = (+ 5, 0)$_{10}$; B = (- 3, 0)$_{10}$. Як відомо, у нормалізованій формі ці числа в форматі з рухомою комою мають вигляд:

$$A = +0,101 \cdot 2^{+3}; \quad B = -0,110 \cdot 2^{+2}. \tag{4}$$

У двійковому коді порядки зображаються таким чином: P_A = + 0011'; P_B = + 0010'. У зміщенному коді ці порядки з негативним нулем (в приведеному форматі зміщення E = 15) мають вигляд:

$$P_A^{15} = 1.0010'; \quad P_B^{15} = 1.0001'.$$

Таким чином, у канонічному форматі операнди A і B будуть мати наступний вигляд:

A : 0.10010''1010 0000 0000 0000 0000 0000 00;
B : 1.10001''0100 0000 0000 0000 0000 0000 00.

Для обчислення порядку добутку визначемо суму зміщених порядків:

$$
\begin{array}{rl}
P_A^{15} = & 1.0010\text{‘} = (+3)^{15} \quad (n=5) \\
+ & \\
P_B^{15} = & 1.0001\text{‘} = (+2)^{15} \\
\text{ВхП} = (+1) = & 1\text{‘} \\
\hline
S = (P_A + P_B - 1)_{\text{ДК}} = (1) & 0.0100\text{‘} = (+4)_{\text{ДК}} \\
\text{ВивП\_S} & N_S
\end{array}
$$

Як відомо, після підсумовування порядків необхідно перевірити коректність результату операції - відсутність переповнення. В даному випадку знаковий розряд суми порядків з негативним нулем не збігається з вивідним переносом, тому переповнення суми порядків вітсутне. Отже, на виході суматора має місце коректне значення суми порядків (сума порядків мінус одиниця у додатковому коді). Для одержання зміщеного порядку добутку, як відомо, необхідно виконати інвертування старшого знакового розряду цієї суми порядків. Якщо виконати інвертування, одержимо:

$$
\begin{array}{rl}
+ \quad S = & 0.0100 \\
NE = & 1.0000 \\
\hline
(Pc)^{15} = (0) & 1.0100 \\
\text{Вив\_NE} & N_{PC}
\end{array}
$$

Розглянемо далі множення додаткових кодів мантис. Припустимо, що мантиси операндів мають вигляд:

$$
\begin{array}{ll}
MA_{\text{ПК}} = 0.\text{‘}\,101\ (+5/8); & MB_{\text{ПК}} = 1.\text{‘}\ 110\ (-6/8); \\
MA_{\text{ДК}} = 0.\text{‘}\,101; & MB_{\text{ДК}} = 1.\text{‘}\,010;
\end{array}
$$

Як відомо, в симетричній двійковій системі ДК негативної мантиси можливо представити слідуючим чином: $MB_{\text{ДК}} = \bar{1}.\text{‘}\,010 = -1 + 2/8 = -6/8$. Згідно з цим множення на знаковий розряд ДК потребує операції: (П + Адк + 1). Таким чином, множення на цифрові розряди виконуються як множення на беззнакові числа, а множення на знаковий розряд здійснюється, як корекція цього результату (рис. 7).

Знаковий розряд добутку мантис ($MC_{\text{ДК}}$) в розглянутому випадку збігається з суміжним розрядом, що вказує на порушення нормалізації результата праворуч на один розряд (рис. 7). Таким чином, для закінчення операції у цьому випадку необхідно виконати також операцію нормалізації шляхом зсуву мантиси добутку ліворуч на один розряд та зменшення порядку на одиницю. Таким чином, остаточно маємо:

$$
MC_{\text{ДК}} = 1.\text{‘}\,100010 \quad \overset{\leftarrow 1\text{р.}}{(MC_{\text{ДК}} = 1.\text{‘}\,000100)}.
$$

$$
\begin{array}{rll}
(P_A + P_B)^{15} = & 1.0100 & = (+5)^{15} \\
+ \quad (-1)^{15} = & 0.1110 & \\
\text{ВхП} = (1) = & 1 & \\
\hline
(P_A + P_B - 1 - 1)_{\text{ДК}} = (1) & 0.0011 & = (+3)_{\text{ДК}} = (P_C - 1)_{\text{ДК}}
\end{array}
$$

Знак декрементування суми порядків в даному випадку не збігається з вивідним переносом ($N_S = 0$, ВивП = 1), тому переповнення декрементора вітсутне. Після інвертування старшого (знакового) розряду суми остаточно одержимо порядок результату, як зміщеного коду:

$$(P_C)^{15} = 1.0011 = (+4)^{15}.$$

З урахуванням усіх перетворень одержимо наступний результат:

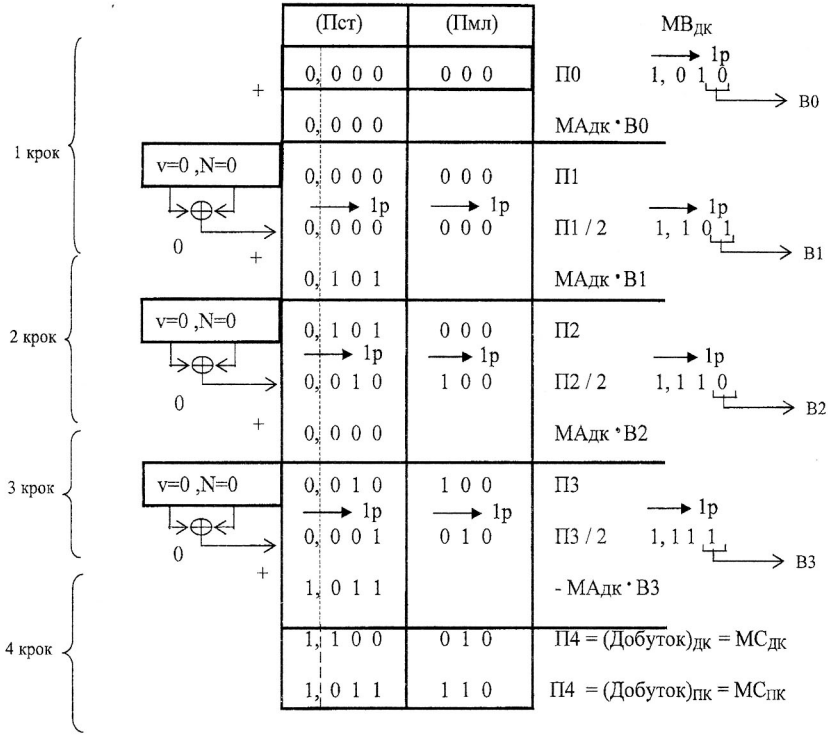


Рисунок 7. Діаграма множення додаткових кодів за алгоритмом "А"
(V – сигнал переповнення суматору, N - знак суми суматору)

Як відомо, множення мантис потребує на кожному кроці операції підсумовування часткових добутків Пст і множеного та зсуву часткових добутків Пст та Пмл (рис. 7).

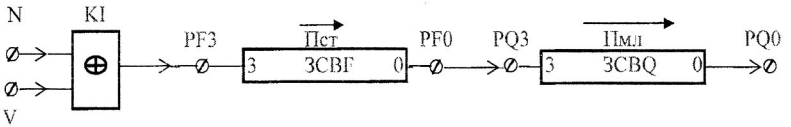


Рисунок 8. Організація зсуву часткових добутків при виконанні операції множення мантис за алгоритмом "А"

Керований інвертор КІ у схемі зсуву (рис. 8) виконує операцію над інформаційними сигналами переповнення V суматору та знака N згідно формули:

$$PF3 = V \oplus N = \overline{V} \cdot N \vee V \cdot \overline{N}, \qquad (5)$$

при цьому на вході PF3 зсувача ЗСВГ формується правильний знак часткового добутку (знак суми N після зсуву може бути зворотним початкову знаку суми, якщо сума обчислена з переповненням V = 1).

$C = -0,1111 \cdot 2^{+4}$, тобто $C = -1111'0 = (-15)_{10}$.

Далі розглянемо контроль коректності операцій над мантисами та порядками чисел із рухомою комою за модулем три.

Для формування контрольних рівнянь операції множення мантис у додатковому коді введемо наступні позначення окремих розрядів змінних:

$MA_{ДК} = (N_A\,'a)_{ДК}$, $MB_{ДК} = (N_B\,'b)_{ДК}$, $MC_{ДК} = (N_C\,'c)_{ДК}$,

де N_A, N_b, N_C - знак мантиси Мн (МА), Мк (МВ) та добутку Мс відповідно ($N_A = N_b = N_C = 0$);

$'a$, $'b$, $'c$ - модулі мантиси Мн, Мк та Мс відповідно.

У розглядаємому вище прикладі: $MA_{ДК} = \pm\,.'101$; $MB_{ДК} = \pm\,.'110$; $MC_{ДК} = \pm\,.'011110$. Кількість розрядів співмножників разом із знаковим розрядом дорівнює $n = 4$, кількість розрядів добутку - $(2n - 1) = 7$.

У залежності від знакових розрядів N_A і N_b можливо записати наступні контрольні рівняння.

1). $MA > 0$; $MB > 0$:

$MA_{ДК} = 0.'a$; $MB_{ДК} = 0.'b$; $MC_{ДК} = MA_{ДК} * MB_{ДК} > 0$.

$$\mathbf{R_K\,(MC_{ДК}) = R\,(MA_{ДК}) * R\,(MB_{ДК}) = R\,('a) * R\,('b),} \qquad (6)$$

де $R_K\,(MC_{ДК})$ - згортка за модулем три добутку $MC_{ДК}$;

$R\,(MA_{ДК})$, $R\,(MB_{ДК})$ - згортка Мн та Мк за модулем три.

2). $MA > 0$; $MB < 0$:

$MA_{ДК} = 0.'a$; $MB_{ДК} = (2_{n+1} - b)$, тобто $b = 2_{n+1} - MB_{ДК}$;

$MC < 0$, тому $MC_{ДК} = 2_{2n} - a * b = 2_{2n} - a * (2_{n+1} - MB_{ДК}) =$

$= 2_{2n} - 2_{n+1} * a + a * MB_{ДК} = 2_{2n} - 2_{n+1} * MA_{ДК} + MA_{ДК} * MB_{ДК}$.

Таким чином, в цьому випадку маємо:

$$\mathbf{R_K\,(MC_{ДК}) = R\,(\sigma_{2n}) - R(\sigma_{n+1}) * R(MA_{ДК}) + R(MA_{ДК}) * R(MB_{ДК}),} \qquad (7)$$

де σ_{2n} - вага $2n$-го розряду за модулем три (для $n = 4$ $\sigma_{2n} = 10$);

σ_{n+1} - вага $(n + 1)$-го розряду за модулем три (для $n = 4$ $\sigma_{n+1} = 01$).

Для розглянутого вище прикладу одержимо:

$R\,(\sigma_{2n}) = R(2_{2n}) = 10$; $R(\sigma_{n+1}) = R(2_{n+1}) = 01$.

3). $MA < 0$; $MB > 0$:

$MA_{ДК} = (2_{n+1} - a)$; $MB_{ДК} = 0.'b$;

Так як $MC < 0$ отримаємо:

$$\mathbf{R_K\,(MC_{ДК}) = R\,(\sigma_{2n}) - R(\sigma_{n+1}) * R(MB_{ДК}) + R(MB_{ДК}) * R(MA_{ДК}),} \qquad (8)$$

4). $MA < 0$; $MB < 0$:

$MA_{ДК} = (2_{n+1} - a)$, тобто $a = 2_{n+1} - MA_{ДК}$; $MB_{ДК} = (2_{n+1} - b)$, тобто $b = 2_{n+1} - MB_{ДК}$;

$MC > 0$, тобто $MC_{ДК} = a * b = (2_{n+1} - MA_{ДК}) * (2_{n+1} - MB_{ДК}) =$

$= (2_{n+1})^2 - 2_{n+1} * MA_{ДК} - 2_{n+1} * MB_{ДК} + MA_{ДК} * MB_{ДК}$.

Отож,

$$\mathbf{R_K(MC_{ДК}) = [R\,(\sigma_{n+1})]^2 - R\,(\sigma_{n+1}) * R(MA_{ДК}) - R\,(\sigma_{n+1}) * R(MB_{ДК}) + R(MB_{ДК}) * R(MA_{ДК}),} \quad (9)$$

Поєднуючи усі розглянуті вище викладки, отримаємо наступне контрольне рівняння для згортки добутку:

$$\mathbf{R_K\,(MC_{ДК}) = [R\,(\sigma_{n+1})]^2 * N_A * N_b + R\,(\sigma_{2n}) * (N_A \oplus N_b) - R\,(\sigma_{n+1}) * N_b * R(MA_{ДК}) - R\,(\sigma_{n+1}) * N_A * R(MB_{ДК}) + R(MB_{ДК}) * R(MA_{ДК}).} \quad (10)$$

Розглянемо використання формули (10) за умови, коли MA > 0 та MB < 0. Нехай $MA_{ДК}$ = 0.'101 , (N_A = 0); $MB_{ДК}$ = 1.'010 , (N_b = 1); R ($MA_{ДК}$) = R(0.'101) = 10 ; R($MB_{ДК}$) = R(1.'010) = 01 ; R (σ_{n+1}) = 01 ; (n = 4); $[R\,(\sigma_{n+1})]^2$ = 01 ; R (σ_{2n}) = 10 ; (2n = 8).

$R_K\,(MC_{ДК}) = (01)^2 * 0 * 1 + 10 * (0 \oplus 1) - (01) * 1 * (10) - (01) * 0 * (01) + (10) * (01) = 0 + 10 - 10 - 0 + 10 = 10.$

Для згортки добутку маємо: $MC_{ДК}$ = 1.'100010 $\rightarrow$ $R^{3Г}\,(MC_{ДК})$ = R(1.'100010) = 10;

Таким чином, $R_K\,(MC_{ДК}) = R^{3Г}\,(MC_{ДК}) = 10$.

Отож, результат операції множення мантис коректний.

Для виконання контролю операції обчислення порядку добутку у коді з негативним нулем (при кількості розрядів m = 5) можливо записати наступні рівняння:

$$\mathbf{R_K\,(S) = R\,[(P_A)^{15}] + R\,[(P_B)^{15}] + R\,(ВхП) - ВивП\_S * R\,(\sigma_{m+1})\,;} \quad (11)$$

$$\mathbf{R_K\,[(P_C)^{15}] = R_K\,(S) + R\,(NE) - ВивП\_NE * R\,(\sigma_{m+1})\,,} \quad (12)$$

де σ_{m+1} - вага (m + 1) – го розряду за модулем три ($\sigma_{m+1} = 2 * NE$; NE = 1.0000) ;

ВивП_S і ВивП_NE - вивідні переноси при формуванні суми S та порядку $(P_C)^{15}$ відповідно.

Недоліком формул (11) та (12) є те, що вивідні переноси ВивП_S і ВивП_NE після декількох тактів роботи МОП губляться. Для їхнього зберігання у склад МОП можливо увести тригери з керованою синхронізацією, але це приводить до ускладнення схеми. Тому формули (11) і (12) доцільно переписати наступним чином. Як відомо, якщо сигнал переповнення операції підсумовування порядків $\overline{OVR_{НН}} = 0$, ВивП_S $\neq N_S$, де N_S - знаковий розряд коду $S = (P_A + P_B - 1)_{ДК}$. Тому ВивП_S = N_S . З другого боку, $N_S = N_{PC}$, так як для одержання знаку N_{PC} знаковий розряд N_S треба інвертувати. З урахуванням цього можливо записати, що ВивП_S = N_{PC}, а ВивП_NE = N_{PC}.

Таким чином, остаточно формулу для виконання операції контролю над порядками у коді з негативним нулем можливо сформувати наступним чином:

$$\mathbf{R_K\,[(P_C)^{15}] = R\,[(P_A)^{15}] + R\,[(P_B)^{15}] + R\,(ВхП) - N_{PC} * R\,(\sigma_{n+1}) + R\,(NE) - \overline{N_{PC}} * R\,(\sigma_{n+1})\,.} \quad (13)$$

Розглянемо використання формули (13) у тому разі, коли $(P_A)^{15}$ = 1. 0010 ; $(P_A)^{15}$ = 1. 0001 ; m = 5 ; mod 3 ; R $[(P_A)^{15}]$ = R (1. 0010) = 11 ; R $[(P_B)^{15}$ = R (1. 0001) = 10. R (ВхП) = 01 ; R (NE) = R (1. 0000) = 01 ; R (σ_{m+1}) = 10 ; N_{PC} = 1.

За цих умов:

$R_K\,[(P_C)^{15}] = 11 + 10 + 01 - 1 * (10) + 01 - 0 * (10) = 10.$

$(P_C)^{15}$ = 1. 0100, таким чином, $R^{3Г}\,(P_C)^{15} = 10 \rightarrow R_K\,[(P_C)^{15}] = R^{3Г}\,(P_C)^{15}$,

що дає підставу рахувати коректним результат операції обчислення порядків.

2.3. Розробка функціональної схеми МОП

Функціональну схему (ФС) МОП доцільно будувати починаючи з окремих її блоків, які потім на завершальному етапі поєднуються в загальну схему. Очевидно, що ФС МОП доцільно поділити на блоки процесорних елементів (на основі ВІС ВС1, ВУ4), оперативної пам'яті (ОП), мікропрограмної пам'яті (МПП) і їх обрамлення (лічильників, регістрів та інших елементів середнього ступеня інтеграції).

Припустимо, що МОП треба реалізувати на базі архітектури типу А – 0 (рис. 3) з використанням двофазної системи синхронізації та ВІС ВС1 і ВУ4. Для побудови схем обрамлення (регістрів та лічильників) призначемо мікросхеми К155ТМ2 і К155ИЕ7 [12, 27].

На рис. 9 наведено фрагмент такої ФС з наступними позначками:

- **РК** - регістр команд (використовується для збереження коду поточної команди впродовж всього циклу виконання програми);
- **ППА** - перетворювач початкової адреси (здійснює перетворення 8 – розрядного коду операції обраної команди (КОП) у 12 - розрядний початковий адрес (ПОЧ_А) першої мікрокоманди мікропрограми УА);
- **ФАМ** - формувач адреси мікрокоманд (ФАМ) на основі мікросхеми ВУ4;
- **МПП** - мікропрограмна пам'ять (реалізується на елементах пам'яті типу ROM і зберігає мікропрограми команд МОП у вигляді послідовностей окремих зв'язаних між собою мікрокоманд);
- **РМК** - регістр мікрокоманд (використовується для збереження поточної мікрокоманди впродовж всього циклу виконання мікрокоманд);
- **ПК** - пульт керування (задавач початкової адреси поточної програми і формувач сигналів "Пуск" і "Зупинка" МОП);
- **ТП** - тригер пуску - зупинки (вмикає або вимикає тактові сигнали $\overline{G1}$ та $\overline{G2}$ для синхронізації блоків МОП);
- **БС** - блок синхронізації (формувач тактових сигналів $\overline{G1}$ і $\overline{G2}$);
- **БК** - блок кон'юнкторів (формувач початкової (нульової) адреси першої мікрокоманди);
- **DC** - формувач (дешифратор) керуючих сигналів U_1, U_2, . . . , U_{27}.

2.3.1. Початковий запуск і зупинка МОП

Припустимо, що до початку роботи МОП програма обчислення функції Y (табл. 25) та початкові дані (табл. 24) знаходяться в основній (оперативній) пам'яті, а в МПП записані мікропрограми команд МОП: MOV, MUL, . . . , STOP (їх ще треба розробити). Крім того, необхідно брати до уваги, що після вмикання живлення за допомогою RC – ланцюга тригер пуску – зупинки (ТП) скидається у нульовий стан. Припустимо також, що за допомогою тумблерного регістра з пульта керування в ЛАК автоматично завантажується початкова адреса програми ($A_{поч}$_ПК = 0).

Запуск системи відбувається після натискання кнопки "Пуск". При цьому формується короткочасний імпульс низького рівня (логічного нуля), за яким тригер ТП перекидається у стан одиниці і вмикає блок синхронізації (БС). Останній формує тактові сигнали $\overline{G1}$ і $\overline{G2}$, після чого на виході блоку кон'юнкторів (БК) підчас дії сигналу "Пуск" формується код інструкції ВУ4 JZ (MI = 0000) - тобто виконується безумовний перехід за нульовою адресою МПП.

Далі ФАМ (ВУ4) після надходження чергового тактового сигналу $\overline{G2}$ виконує інструкцію JZ. При цьому на його виході Y формується нульова адреса

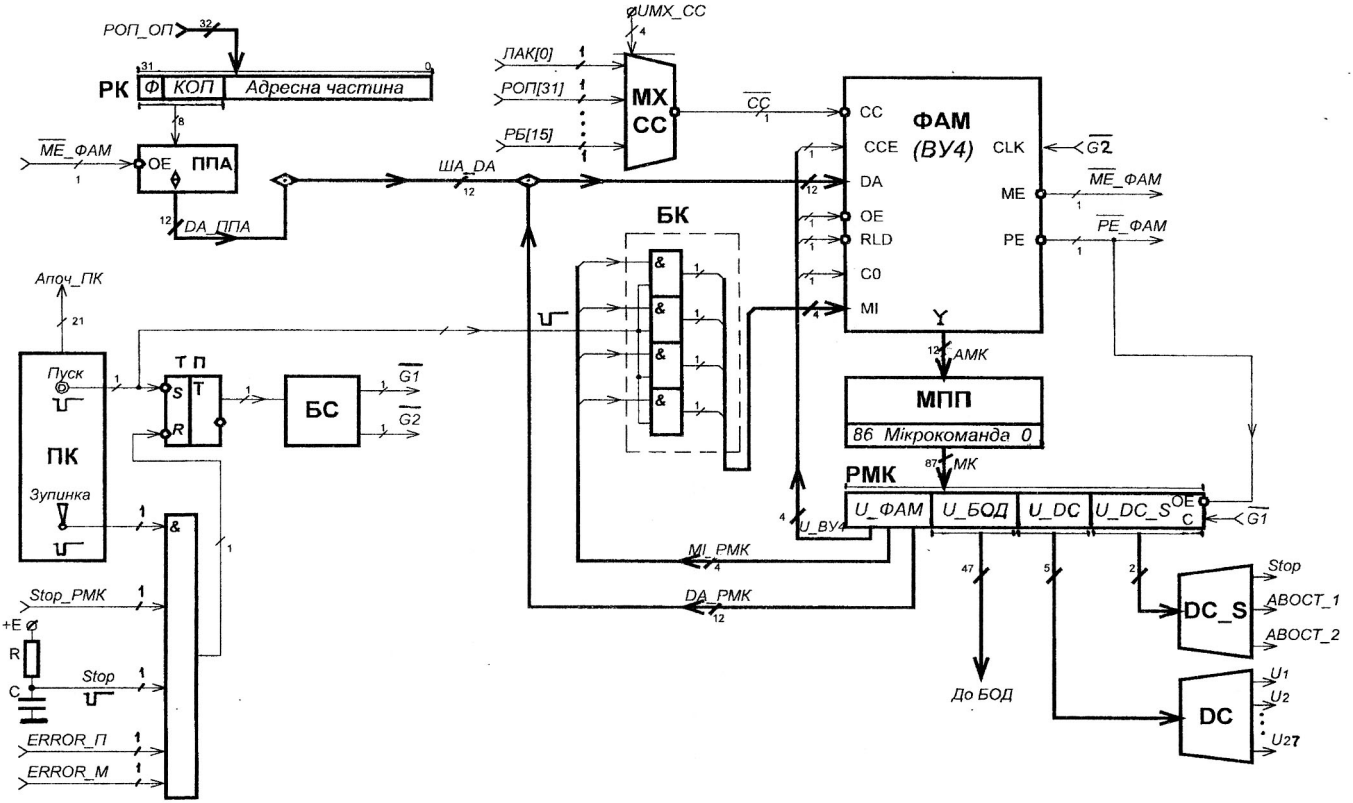


Рисунок 9. Функціональна схема МОП (фрагмент 1)

мікрокоманди (АМК = 0). Одночасно з цим відбувається скидання покажчика стека (SP = 0).

За адресою АМК із МПП зчитується нульова мікрокоманда (МК) мікропрограми і після надходження чергового тактового сигналу G1 МК записується у РМК. Вихід РМК утворює множину керуючих сигналів, які активізують необхідні блоки МОП.

Зупинка роботи МОП може бути здійснена трьома шляхами:

- натисканням кнопки "Зупинка" на пульті керування;
- за допомогою сигналу Stop_РМК = Stop ∨ АВОСТ_1 ∨ АВОСТ_2, де сигнали Stop, АВОСТ_1 і АВОСТ_2 формуються мікропрограмою і дешифруються DC_S (табл. 26);
- за допомогою апаратних сигналів ERROR_П та ERROR_М, які формуються відповідно схемами контролю порядків та мантис.

У всіх цих випадках відбувається скидання тригера ТП по входу R і вимикання блока формування тактових сигналів синхронізації БС, що припиняє роботу МОП (рис. 9).

Формування дешифратором DC_S сигналів зупинки МОП. Таблиця 26

Код управління U_DC_S	Позначка сигналу	Призначення сигналу
0 0	Stop	Нормальне завершення мікропрограми
0 1	АВОСТ_1	Переповнення суматору порядків
1 0	АВОСТ_2	Порушення адресації
1 1	-	Сигнал зупинки не формується

2.3.2. Зчитування з ОП команди та декодування команди

Як випливає з аналізу складу команд МОП (рис. 4), команди можливо розділити на дві групи: команди довгого формату RS, RX (32 розряду) і команди короткого формату RR (16 розрядів). За умовою одна комірка ОП містить 32 розряду. Отже, в одній комірці оперативної пам'яті може бути розташована одна команда формату RX (RS), або дві команди формату RR, або одна команда типу RR і півкоманди типу RX (RS) (рис. 5). На рис. 10 наведені ці приклади і використані наступні позначки блоків:

- **ОП** - оперативна пам'ять обсягом в 1М осередків із 32 розрядів кожна (призначена для збереження програм (на рівні команд) і даних);
- **РАП** - регістр адреси пам'яті (містить адресу комірки пам'яті, до якої відбувається звернення в поточному такті роботи);
- **РОП** - регістр оперативної пам'яті (є проміжною ланкою (буфером) при виконанні операцій запису або зчитування даних);
- **ЛАК** - лічильник адреси команд (містить адресу поточної команди комп'ютера);
- **РБ** - регістр буферний (зберігає одну команду формата RR або півкоманди формату RX або RS).
- **Т_JMP** - тригер ознаки переходу (використовується при виконанні умовного (або безумовного) переходу до поточної команди; після вмикання живлення за допомогою RC – ланцюга тригер установлюється в одиничний стан).

Припустимо, що формат команди задається старшими розрядами команди - полем Ф: Ф = 01 → команда формату RR; Ф = 10 → команда формату RX і

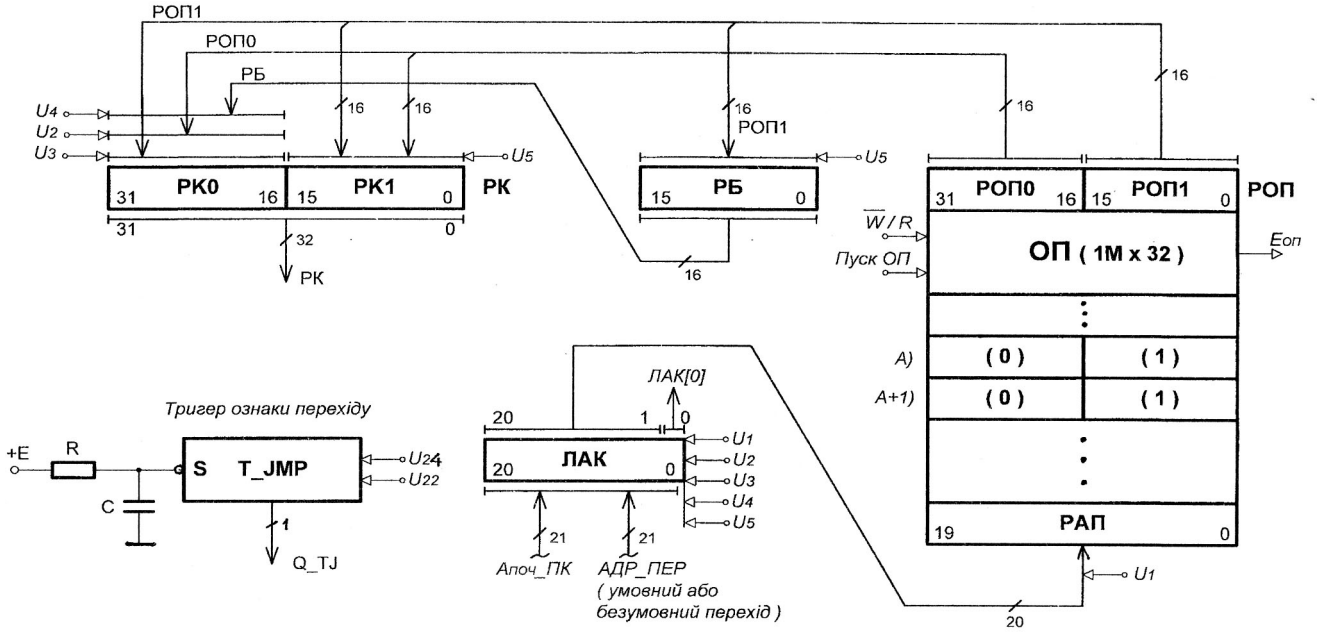


Рисунок 10. Функціональна схема формування поточної команди МОП (фрагмент 2)

$\Phi = 11 \rightarrow$ команда формату RS. Для зчитування команд використовується сигнал $\overline{w} / r$, який задає режим роботи пам'яті. Якщо $\overline{w} / r = 1$, в ОП відбувається зчитування вмісту комірки ОП в РОП за адресою РАП. Після закінчення зчитування формується сигнал $E_{ОП} = 1$ (сигнал завершення операції зчитування).

Нульовий розряд ЛАК вказує, з якого півслова РОП починається команда.

Граф – схема алгоритму вибору із ОП команд зазначених вище форматів наведена на рис.11. Буферний регістр (РБ) на 16 двійкових розрядів дозволяє організувати тимчасове збереження однієї команди формату RR або півкоманди формату RX (RS). За рахунок цього вдається уникнути повторного звернення до ОП за командою, якщо раніше уже було звернення до цієї комірки ОП і, таким чином, підвищити продуктивність МОП.

Інакше буде відбуватися вибірка команди із ОП, якщо попередня команда є командою умовного або безумовного переходу. Для вказівки на цей случай тригер ознаки переходу T_JMP установлюється в одиничний стан (Q_TJ =1). При цьому із ОП відбувається зчитування слова, а надалі у залежності від ознак ЛАК [0], РОП [31] та РОП [15] в РК пересилається команда формату півслова (RR), або слова (RX / RS). Скидання тригера T_JMP у нульовий стан відбувається після завершення етапу вибірки команди з використанням керуючого сигналу U_{24}.

Після етапу зчитування команда зберігається у регістрі команд (РК). Для декодування команди використовується перетворювач початкової адреси (ППА). Останній працює, якщо на його вході управління OE з'являється активний сигнал низького рівня ($\overline{OE} = 0$). Це можливо організувати з допомогою ФАМ (ВУ4), який у разі виконання інструкції JMAP (MI = 0010) формує на виході ME сигнал низького рівня ($\overline{ME} = 0$). В зв'язку з цим на рис. 9 вивід ME мікросхеми ВУ4 з'єднується із входом $\overline{OE}$ ППА і, таким чином, вмикається схема декодування команди. ППА формує на виході (з'єднується через шину ША_DA із входом DA ВУ4) 12 - розрядний двійковий код - адресу першої мікрокоманди підпрограми МПП, яка організує виконання у МОП поточної команди (RR, RX або RS), яка знаходиться у регістрі команд РК.

2.3.3. Команда пересилки даних (MOV)

Якщо обчислення функції Y за формулою (2) виконується з використанням операндів із рухомою комою, операції доцільно виконувати окремо над порядками і мантисами. При цьому, очевидно, розрядність БОД (кратна розрядності однієї МПС ВС1, тобто чотирьом) варто вибирати відповідно до розрядності мантиси у додатковому коді (27 розрядів, старший розряд - знаковий). В зв'язку з цим БОД побудуємо із 7-и секцій ВС1. При цьому молодший (нульовий) розряд коду мантиси треба заповнити нулем.

Внутрішню пам'ять (РЗП) ВС1 із 16 регістрів будемо використовувати таким чином, як це наведено у табл. 24.

Пересилка значення початкового співмножника (мантиси і порядку), бази B2, індексу X2 із оперативної пам'яті (ОП) у РЗП БОД здійснюється у МОП з використанням відповідних команд пересилання MOV.

На рис.12 наведено фрагмент 3 функціональної схеми взаємодії РК, ОП та БОД при виконанні команд пересилання даних. Формувачі коду ФК1 – ФК4 із трьома станами на виході виконують необхідні перетворення кодів (форматів float або integer) згідно із сигналами управління, які надходять із дешифратора DC (табл. 27).

ГСА команд формату RS для пересилання даних із ОП в РЗП БОД наведена на рис.13. Команда **MOV_IMR** пересилає дані типу integer - дані з фіксованою комою (рис. 13, а). При її виконанні вміст осередку ОП (за адресою S2 регістра команд) зчитується в РОП. Далі через формувач коду ФК1 дані РОП подаються на

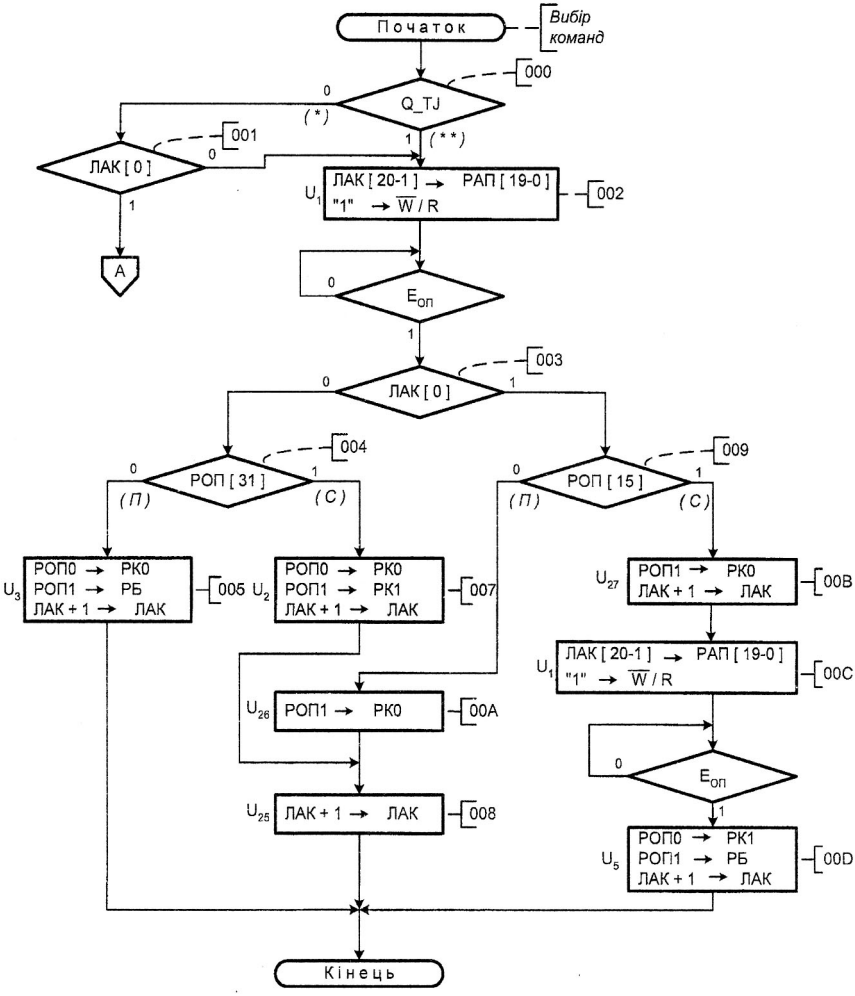


Рисунок 11 (початок). ГСА вибору із ОП команди формату RR (півслово – п) та RX / RS (слово – с): * — за природного переходу до початкової команди ; ** — за умовного (або безумовного) переходу до поточної команди

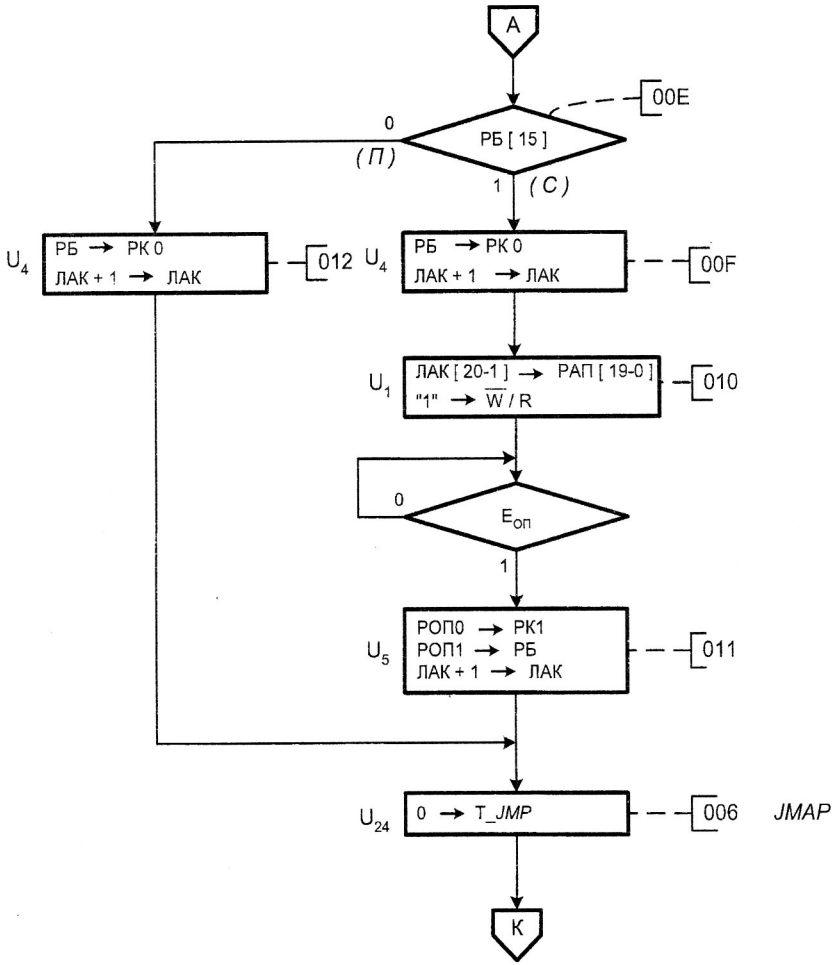


Рисунок 11 (кінець)

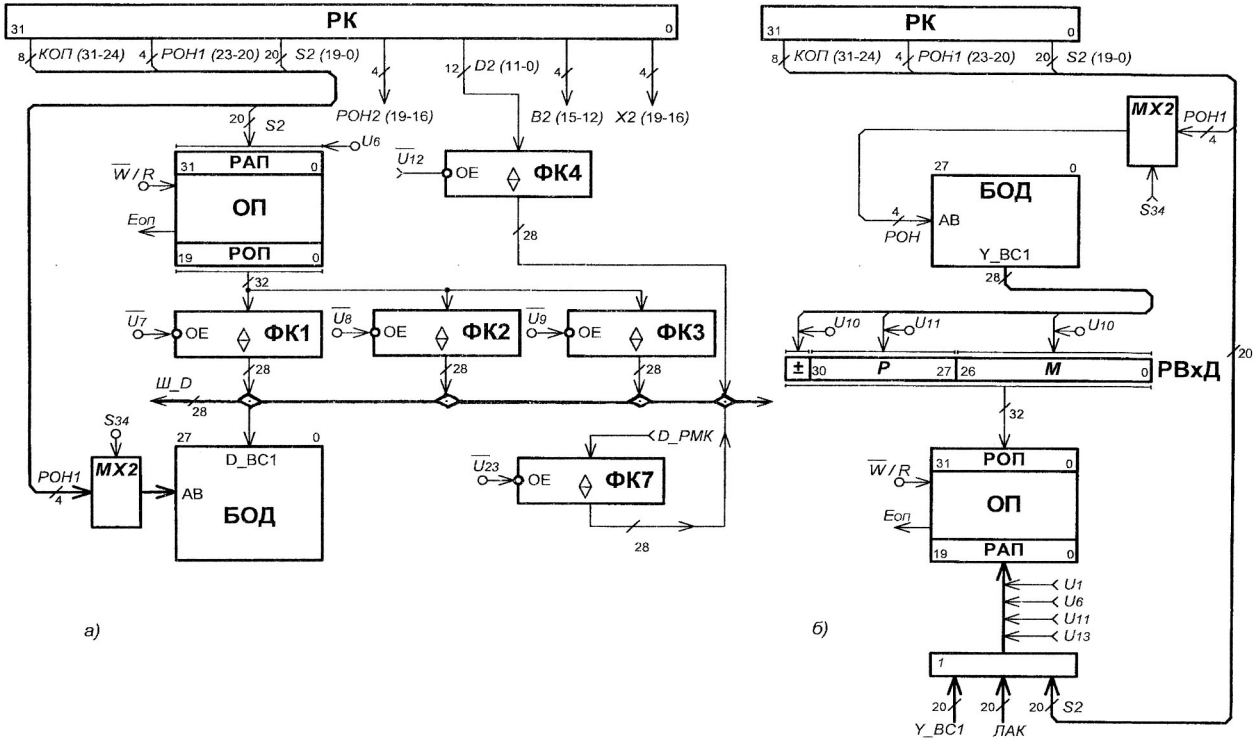


Рисунок 12. Функціональна схема МОП (фрагмент 3) :
а) – пересилання даних із ОП в БОД ;
б) – пересилання даних із БОД в ОП

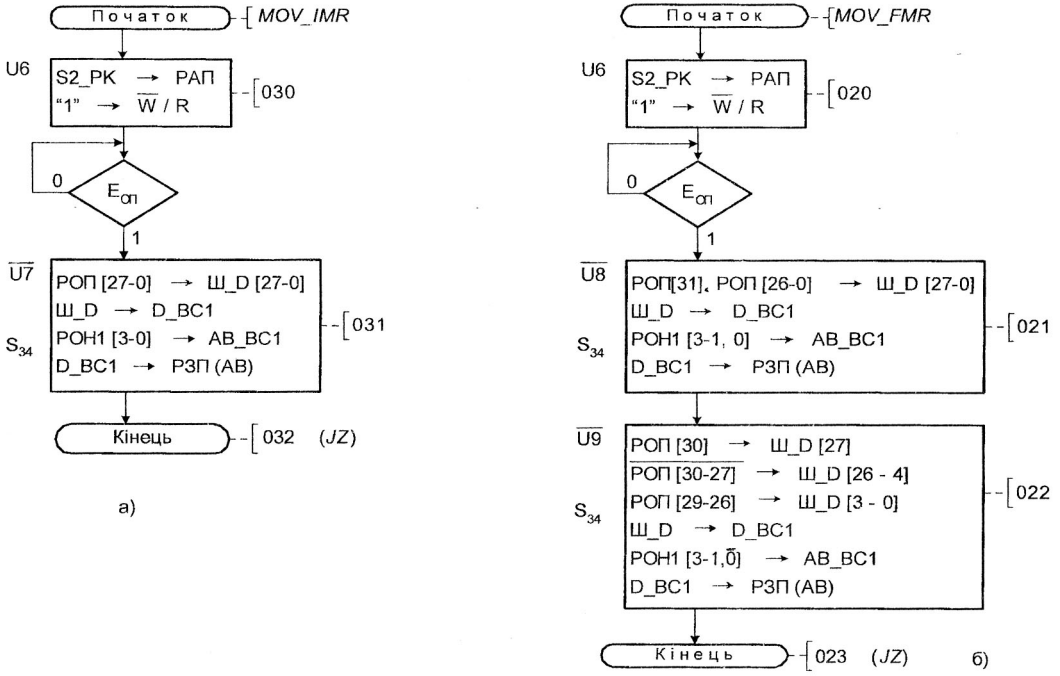


Рисунок 13. ГСА команд формату RS пересилки даних із ОП у РЗП БОД

Формування сигналів управління на виходу дешифратора DC. Таблиця 27

Код управління U_DC (4–0)	Сигнал управлін.	Виконувані мікрооперації
0 0 0 0 0	U0	Порожня операція
0 0 0 0 1	U1	ЛАК [20 – 1] → РАП [19 – 0]
0 0 0 1 0	U2	РОП0 → РК0 ; РОП1 → РК1 ; ЛАК + 1 → ЛАК
0 0 0 1 1	U3	РОП0 → РК0 ; РОП1 → РБ ; ЛАК + 1 → ЛАК
0 0 1 0 0	U4	РБ → РК0 ; ЛАК + 1 → ЛАК
0 0 1 0 1	U5	РОП0 → РК1 ; РОП1 → РБ ; ЛАК + 1 → ЛАК
0 0 1 1 0	U6	S2_РК → РАП
0 0 1 1 1	U7#	РОП [27 – 0] → Ш_D [27 – 0] ; ФК1
0 1 0 0 0	U8#	РОП[31].РОП[26–0] → Ш_D[27– 0] → Р1_ММК ФК2
0 1 0 0 1	U9#	РОП [30] → Ш_D [27] ; РОП [30]# → Ш_D [26 – 4]; ФК3 РОП [29 – 26] → Ш_D [3 – 0]
0 1 0 1 0	U10	Y_ВС1 [27] → РВхД [31] ; Y_ВС1 [26 – 0] → РВхД [26 – 0]
0 1 0 1 1	U11	Y_ВС1 [4 – 0] → РВхД [30 –26] ; S2_РК → РАП
0 1 1 0 0	U12	0 [27 – 12]. D2_РК→ Ш_D [27 – 0] ФК4
0 1 1 0 1	U13	Y_ВС1 → РАП
0 1 1 1 0	U14	РВхД → РОП
0 1 1 1 1	U15	Ш_D [27 – 0] → Р2_ММК
1 0 0 0 0	U16#	РОП [30] → Ш_К [4] ; РОП [30 – 27] → Ш_К [3 – 0] ФК5
1 0 0 0 1	U17	Ш_К [4 – 0] → Р1_ПМН БК
1 0 0 1 0	U18	Ш_К [4 – 0] → Р2_ПМК БК
1 0 0 1 1	U19#	Y_ВС1 [4 – 0] → Ш_ФК6 [4 – 0] БК , ФК6
1 0 1 0 0	U20	Ш_ФК6 [4 – 0] → Р3_ПС БК
1 0 1 0 1	U21	Z → TZ , D_РМК → D_ВС1
1 0 1 1 0	U22	S2_РК → ЛАК ; 1 → T_JMP
1 0 1 1 1	U23	D_РМК → D_ВС1
1 1 0 0 0	U24	0 → T_JMP
1 1 0 0 1	U25	ЛАК + 1 → ЛАК
1 1 0 1 0	U26	РОП1 → РК0
1 1 0 1 1	U27	РОП1 → РК0 ; ЛАК + 1 → ЛАК

Примітка. # - позначка низького рівня активності сигналу.

Мультиплексор МХ2 у залежності від коду S34 (1 – 0) (табл. 28) виконує функції адресного комутатора на вході порта АВ секцій ВС1 БОД.

Алгоритм роботи мультиплексора МХ2. Таблиця 28

S34 (1 –0) – значення адреси на вході МХ2	Коди сигналів на виходу МХ2
0 0	РОН1_РК (3 – 0)
0 1	РОН1_РК (3 – 1, 0)
1 0	РОН1_РК (3 – 1, 0#)
1 1	АВ_РМК (3 – 0)

вивід D_BC1 БОД. Потім за адресою РОН1 РК здійснюється запис цього коду до відповідного регістру РЗП.

Команда **MOV_FMR** пересилає дані типу float – дані з рухомою комою (рис. 13, б). При її виконанні, аналогічно попередній команді, вміст осередку ОП (за адресою S2) зчитується в РОП. Надалі з використанням формувачів ФК2 і ФК3 відбувається формування кодів мантиси та порядку на вході БОД згідно з вимогами інтерфейсу БОД. Наприкінці за адресою РОН1 РК здійснюється запис коду у два суміжні регістри РЗП.

Команда **MOV_FRM** виконується за допомогою команди RS і дозволяє виконати завантаження результату (у форматі float) із РЗП в основну пам'ять МОП. ГСА виконання цієї команди наведена на рис. 14. Припустимо, що результат обчислення функції (2) міститься у регістрі мантиси R0 (в додатковому коді) і регістрі порядку R1 (у коді з негативним нулем). Тому до початку операції запису в ОП R0.R1 за допомогою зовнішнього регістра РВхД згортується у канонічний формат з рухомою комою (знак мантиси - порядок - мантиса) Так як мантиса і порядок результату розташовані у суміжних регістрах РЗП (R0.R1), адреса порядку в алгоритмі згортання виконується інвертуванням молодшого розряду поля РОН регістра команд. Умовно цю операцію інвертування будемо записувати у наступному вигляді: РОН_РК [3 – 1, 0].

На рис. 15 наведена ГСА команди **MOV_FMR** пересилання даних з рухомою комою із ОП в РЗП з використанням команди RX. Цю команду доцільно організувати у вигляді підпрограм обчислення виконавчої адреси $A_{ВИК2}$ та зчитування операнда з ОП за цією адресою.

ГСА підпрограми обчислення виконавчої адреси $A_{ВИК2}$

При обчисленні $A_{ВИК2}$ згідно з формулою (1) як акумулятор (АКК) використуємо регістр R2 РЗП. В зв'язку з цим до початку обчислень (рис.15) у ГСА виконується скидання R2 (блок 1). Потім до R2 послідовно додається значення індексу X2 (блок 2), бази B2 (блок 4) і зміщення D2 (блок 5). При цьому, якщо код B2 в РК дорівнює нулю, база безумовно рахується рівною нулю і у цьому випадку блок 4 ГСА не виконується.

В загальному випадку, очевидно, на кожному кроці проміжних обчислень $A_{ВИК2}$ необхідно реалізувати перевірку порушення адресації основної пам'яті. За попередньою умовою обсяг ОП складає 1М осередків з 32 розрядами кожний. Так як $1M = 2^{20}$, тому максимальна кількість розрядів РАП дорівнює 20. В зв'язку з цим, якщо у процесі обчислення $A_{ВИК2}$ в одному з розрядів регістра R2 з 21 до 31 з'являється хоча б одна одиниця, варто формувати сигнал аварійного завершення операції (АВОСТ_2) через порушення адресації.

ГСА підпрограма зчитування операнда за адресою $A_{ВИК2}$

Після обчислення виконавчої адреси $A_{ВИК2}$ виконується підпрограма зчитування з ОП за цією адресою операнду. Для цього виконується цикл читання з основної пам'яті операнда у форматі з рухомою комою. Далі виконується спеціальне перетворення формату операнда з використанням формувачів ФК2 та ФК3 і занесення його в РЗП у суміжні регістри (блоки 10 та 11 ГСА), наприклад, запис мантиси в R14 і порядку числа в R15.

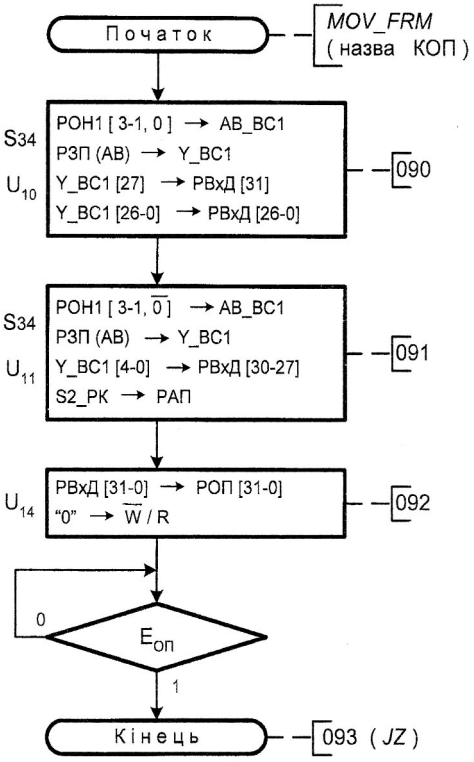


Рисунок 14. ГСА команди формату RS для пересилки даних із РЗП БОД в основну оперативну пам'ять МОП

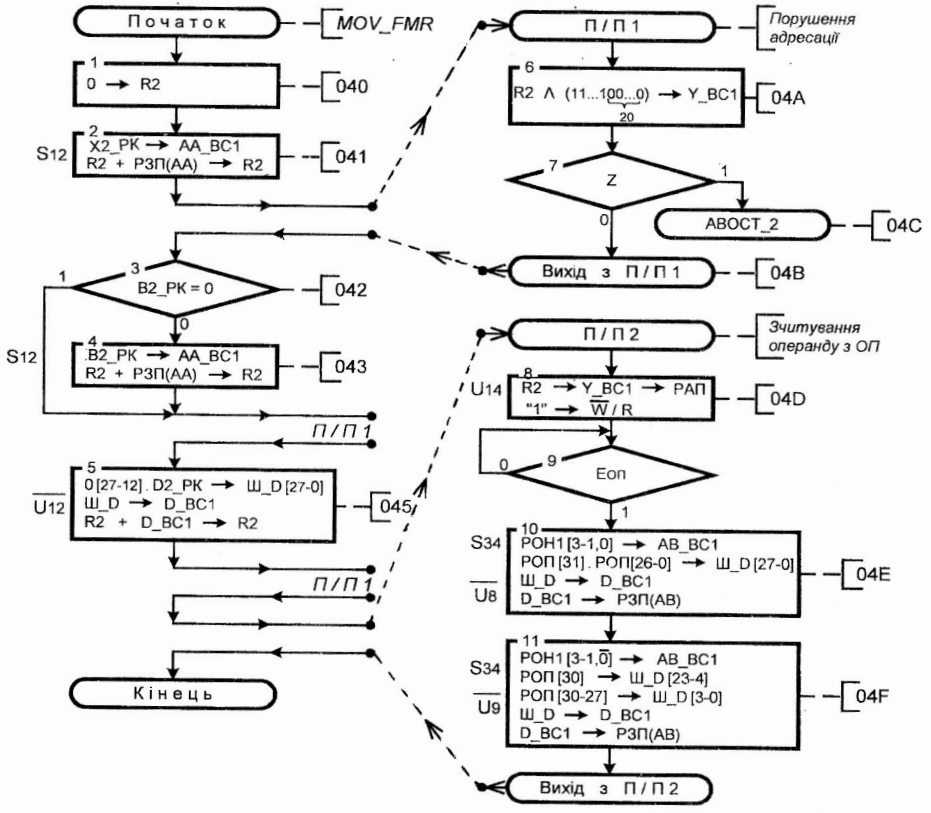


Рисунок 15. ГСА команди RX для пересилки даних із ОП в РЗП БОД

2.3.4. Команди обробки даних (DEC_I, JMP_Z, MUL_F)

Припустимо, що команда декрементування **DEC_I** має формат типу RR і при її виконанні відбувається зменшення на одиницю вмісту регістру РЗП, номер якого задається в полі РОН1 регістра команд (рис. 16. а). При цьому одночасно результат операції передається на шину F ВС1 і формується ознака нуля Z, яка запам'ятовується у тригері TZ за керуючим сигналом U_{21}.

Приймемо також, що команда умовного переходу **JMP_Z** формату RS працює звичайно після команди DEC_I (рис. 16, б). При цьому відбувається перевірка вмісту тригера TZ. Якщо вміст TZ = 1, адреса S2 регістра команд передається у лічильник адреси команд (ЛАК) (сигнал U_{22}) і, таким чином, змінюється послідовний хід виконання програми. Одночасно відбувається установлення в одиничний стан тригера T_JMP. У противному разі (якщо TZ = 0) команда завершує свою роботу звичайним чином, тобто виконується інструкція JZ ФАМ ВУ4.

Припустимо, що команда **MUL** відноситься до команд формату RR. Отож, для її виконання операнди (множене та множник) зазделегідь необхідно розташувати в РЗП БОД. Тому попередньо до виконання команди MUL перший співмножник необхідно завантажити, наприклад, в регістри R0 (мантиса) і R1 (порядок), а другий - в регістри R14 (мантиса) та R15 (порядок) за допомогою відповідних команд пересилання даних MOV.

На рис.17 наведено фрагмент 4 функціональної схеми БОД та схем обрамлення, які необхідні для виконання програми. Буферні підсилювачі (БП) з трьома станами підключені до клем зсувачів БОД PF3, PF0, PQ3 та PQ0 і дозволяють організувати комутацію інформаційних розрядів в операціях зсуву. Для цього команда зсуву праворуч старших розрядів часткових добутків ВС1 (у цьому випадку розряд I7 інструкції ВС1 дорівнює нулю) подає на клему PF3 вихід керованого інвертора (КІ) відповідно з (5). При виконанні операції відновлення нормалізованого результату мантиси добутку необхідно виконати її зсув ліворуч на один розряд (у цьому випадку розряд I7 = 1). Для цього на клеми PF0 і PQ0 БОД в цьому випадку через БП подається код логічного нуля.

Нехай старша частина добутків $П_{ст}$ знаходиться в регістрі R10 РЗП, а молодша $П_{мл}$ - в регістрі RQ. Тоді до початку циклу множення (звернення до підпрограми MULT) доцільно переслати мантису множника із регістру R14 в регістр RQ, що дозволить у подальшому виконувати зсув подвійної довжини (рис. 8).

Робота формувачів кодів ФК1 – ФК4 та мультиплексора МХ2 ФС була розглянута раніше. Мультиплексор коду МХ1 у залежності від коду S12 (2 – 0) формує код (табл. 29), який потім подається до адресного порту АА БОД.

Алгоритм роботи мультиплексора МХ1. Таблиця 29

S12 (2-0) – значення адреси на вході МХ1	Коди сигналів на виході МХ1
0 0 0	X2_РК
0 0 1	B2_РК
0 1 0	AA_РМК
0 1 1	AB_РМК
1 0 0	РОН1_РК

Сигнал $OVR_{НН}$ ФС дозволяє виявити виникнення переповнення операції підсумовування порядків у коді з негативним нулем і формується згідно з формулою

$$OVR_{НН} = F3 \oplus C4. \qquad (14)$$

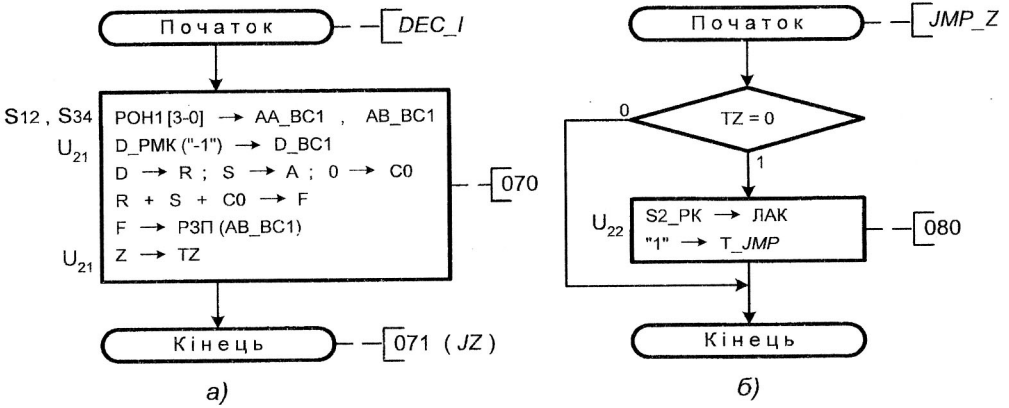


Рисунок 16. ГСА виконання команд DEC_I та JMP_Z

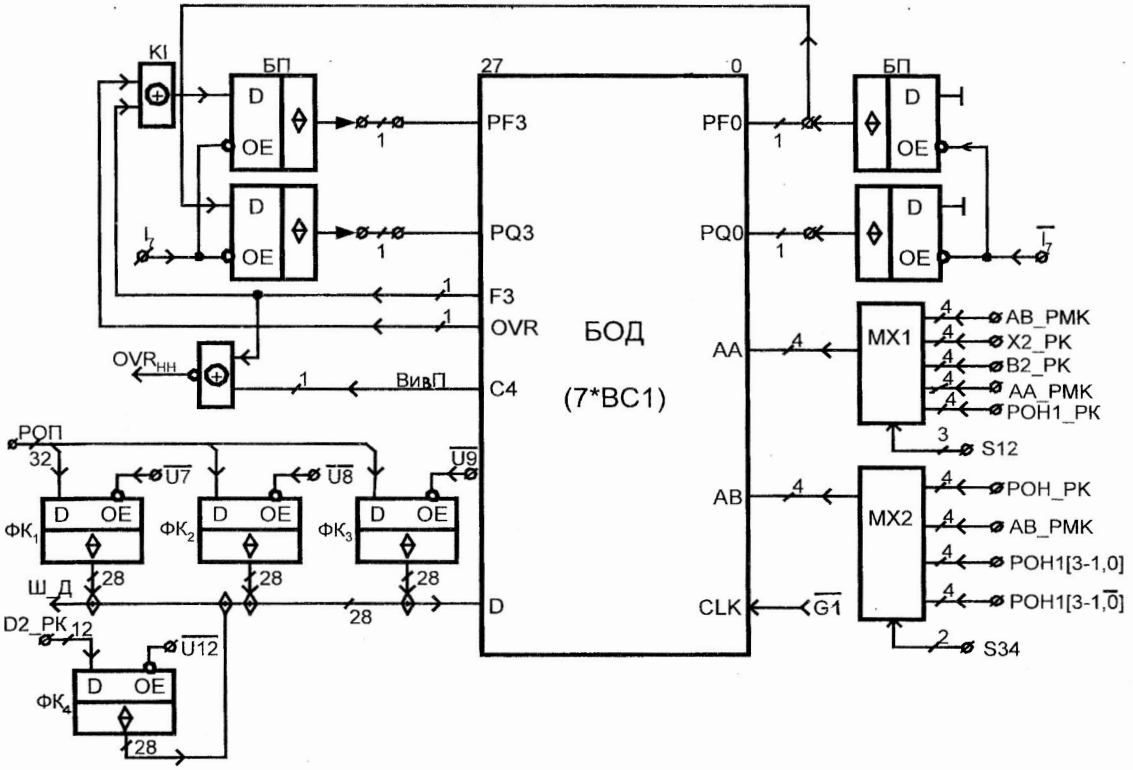


Рисунок 17. Функціональна схема МОП (фрагмент 4)

ГСА підпрограми множення чисел з рухомою комою

Припустимо, що до звертання до підпрограми множення мантиси (у додатковому коді) множемого $ММН_{ДК}$ та множника $ММК_{ДК}$ розташовані відповідно в регістрах R0 і R14 РЗП ВС1 у наступному форматі:

±.' XXX XXXX XXXX XXXX XXXX XXXX XXX0. (15)

Запис нуля праворуч від молодшого розряду мантиси обумовлено узгодженням розрядності мантиси в ОП (27 розрядів) із розрядністю семи секцій БОД (28 розрядів).

Припустимо також, що до початку операції порядок (у коді з негативним нулем) множемого $ПМН_{НН}$ і множника $ПМК_{НН}$ знаходяться відповідно в регістрах R1 і R15 РЗП у наступному форматі:

± +++ ++++ ++++ ++++ ++++ ++++ XXXX'. (16)

Призначемо також регістр R10 для збереження старших розрядів добутку і RQ - молодших розрядів добутку, а спочатку - мантиси множника (ММК).

При цих умовах граф – схема алгоритму множення операндів наведена на рис.18. До початку циклу множення мантис (за алгоритмом "А") відбувається скидання регістра старших розрядів часткових добутків (R10) і в лічильник (РА / СТ) ВУ4 записується код 26 (на одиницю менше кількості повторень циклу). Ці дії відбиті у блоці 1 ГСА.

Цикл множення мантис (у додатковому коді) виконується у такий спосіб. З використанням операції маскування виконується аналіз (блоки 2, 3) по ознаці нульового результату (Z) молодшого розряду множника RQ [0]. Якщо RQ[0] містить одиницю, то до старших розрядів часткових добутків R10 додається множене (блок 4). Далі відбувається зсув праворуч подвійної довжини отриманого результату підсумовування в R10 і множника в RQ, яка наведена за схемою рис.8. Якщо RQ [0] = 0, здійснюється тільки операція подвійного зсуву (блок 5) R10 і RQ (без додавання множемого).

Один цикл множення на поточний розряд множника завершується зменшенням вмісту лічильника РА / СТ на одиницю і перевіркою його стану на нуль. Ця операція здійснюється ФАМ ВУ4 (блок 6). БОД у цьому такті виконує порожню операцію (NOP).

Якщо РА / СТ ≠ 0, цикл множення продовжується шляхом передачі керування до блоку 2. У протилежному випадку відбувається вихід з циклу. Далі виконується крок корекції - множення на знаковий розряд множника (блок 7). Якщо знаковий розряд негативний (дорівнює одиниці), від отриманого часткового добутку віднімається множене (блок 8) шляхом підсумовування перетвореного множемого.

Етап округлення старшої частини мантиси результату виконується у блоках 9 і 10. Для цього виконується аналіз старшого розряду (F3) регістра RQ. Зазначемо, що після завершення операції множення мантис, у регістрі RQ зберігаються молодші розряди часткових добутків. Якщо F3 = 1, то для округлення результата до старших розрядів часткових добутків R10 додається одиниця (блок 10).

У блоках 11, 12 та 14 ГСА здійснюється перевірка порушення нормалізації результату ($МС_{ДК}$) праворуч на один розряд. Якщо знаковий і суміжний розряди $МС_{ДК}$ не збігаються (1.0 або 0.1), то отриманий результат не вважається нормалізованим. Для одержання нормалізованого результату $МС_{ДК}$ зсувається

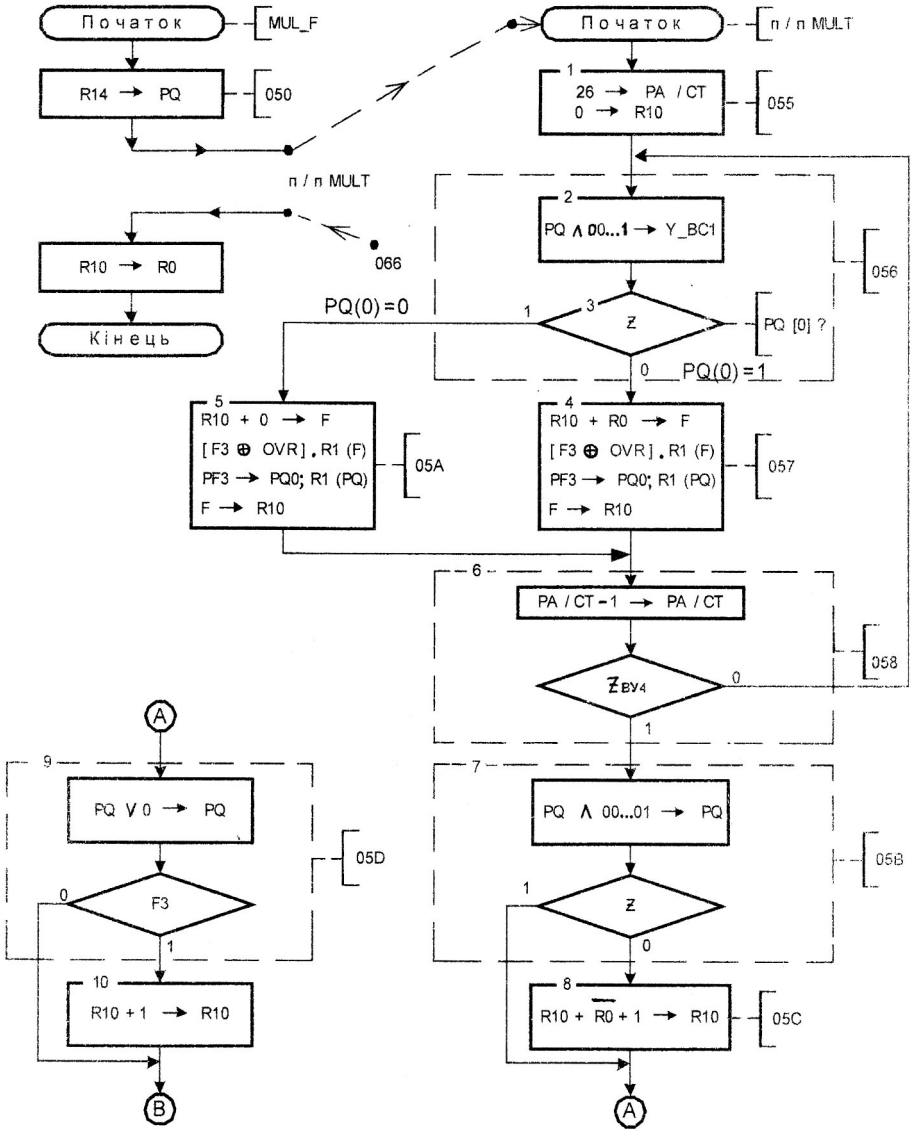


Рисунок 18. ГСА виконання команди множення з рухомою комою (початок)

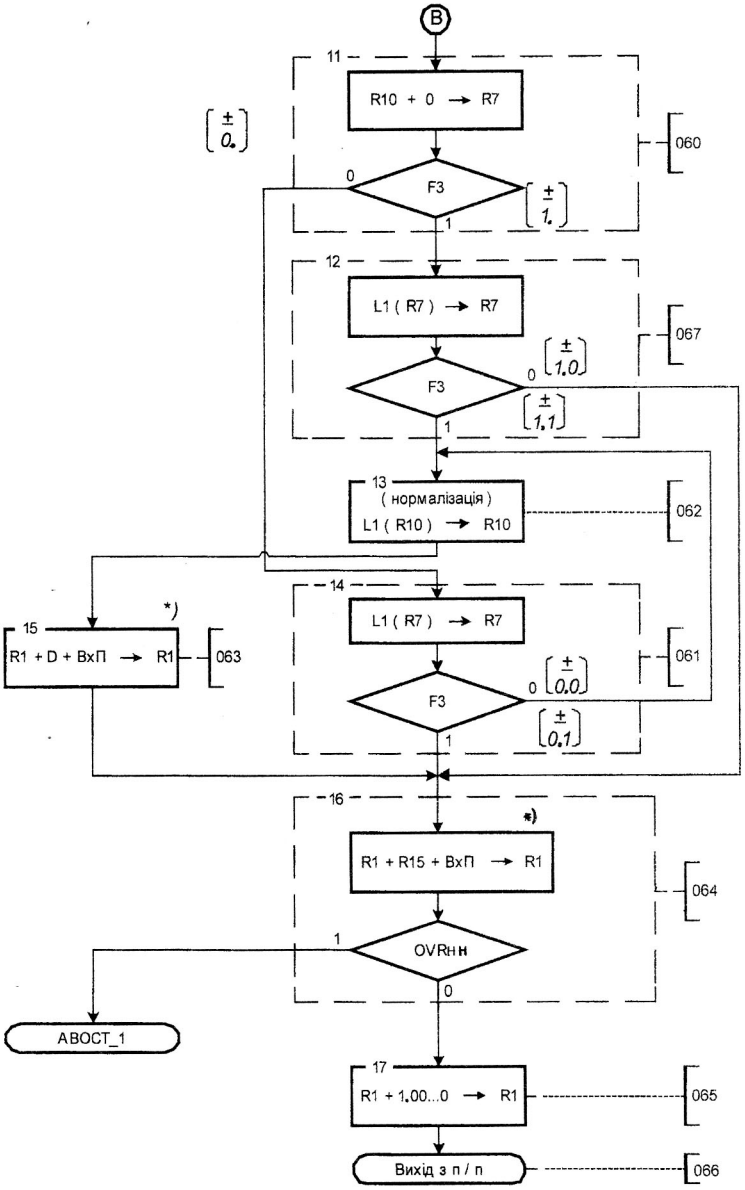


Рисунок 18 . Закінчення .
Примітка: *) $(-1)^{15}$ = 0.1110 → D ;
ВхП = 1

ліворуч на один розряд (блок 13) і до регістру R1 (у ньому розташуємо порядок результату) додається (-1) у коді з негативним нулем (блок 15).

Далі відбувається формування у регістрі R1 остаточного порядку добутку шляхом додавання у коді з негативним нулем порядку множника, тобто R15 (блок 16). Якщо виникнє переповнення ($OVR_{НН} = 1$), формується сигнал аварійної зупинки (ABOCT_1). У протилежному випадку шляхом інвертування старшого (знакового) розряду (блок 17) остаточно формується значення порядку добутку у коді з негативним нулем.

Після закінчення підпрограми множення для розташування результату в належних регістрах виконується пересилання старшої частини добутку мантиси із регістру R10 до регістру R0.

2.3.5. Розробка блоку контролю (перевірки) операції множення з рухомою комою

Контроль операції множення з рухомою комою за модулем три будемо будувати окремо для порядків та мантис.

Функціональна схема блоку контролю (БК) за модулем три операції множення мантис (у додатковому коді), яка побудована згідно з формулою (10) наведена на рис.19, де використані наступні позначки блоків:

- **Р1_ММК** і **Р2_ММН** - вхідні регістри для зображення у БК додаткових кодів мантис відповідно Мн та Мк;
- **Р3_МС** - регістр для збереження добутку мантис, який формується на виходу Y_BC1 БОД;
- **СФЗ** - схема формування залишків за модулем три;
- **БМЗ** - блок множення залишків за модулем три;
- **SM_{M3}** - суматор за модулем три;
- **СП** - схема порівняння.

Алгоритм синтезу та аналізу роботи схем БПЗ та СФЗ можливо знайти у [1, 2].

Докладно алгоритм контролю операції множення мантис можливо звести до наступних послідовних кроків:

1. Мантиси Мн і Мк, які формуються ФК2, запам'ятовуються у регістрах БК Р1_ММК та Р2_ММН.
2. З використанням СФЗ знаходяться контрольні залишки мантис за модулем три відповідно R ($МВ_{ДК}$) та R ($МА_{ДК}$).
3. З використанням блоку множення залишків (БМЗ) знаходиться добуток контрольних залишків операндів.
4. З використанням суматорів за модулем три формується контрольний код $R_К(МС_{ДК})$ згідно з (10).
5. Добуток початкових значень мантис із БОД записується в регістр БК Р3_МС і далі з використанням СФЗ формується згортка цього коду за модулем три ($R^{3Г}(МС_{ДК})$).
6. Далі з використанням схеми порівняння (СП) виконується аналіз кодів $R_К(МС_{ДК})$ та $R^{3Г}(МС_{ДК})$. При розбіжності кодів формується сигнал помилки ERROR_M = 1 і відбувається зупинка роботи МОП.

Функціональна схема блоку контролю за модулем три операції підсумовування порядків співмножників з негативним нулем, яка побудована згідно з формулою (13), наведена на рис. 20, де використані наступні позначки блоків:

- **Р1_ПМН** і **Р2_ПМК** - вхідні регістри для збереження в БК порядків Мн та Мк у коді з негативним нулем;
- **Р3_ПС** - регістр збереження результату операції підсумовування порядків, який формується на виходу Y_BC1 БОД;

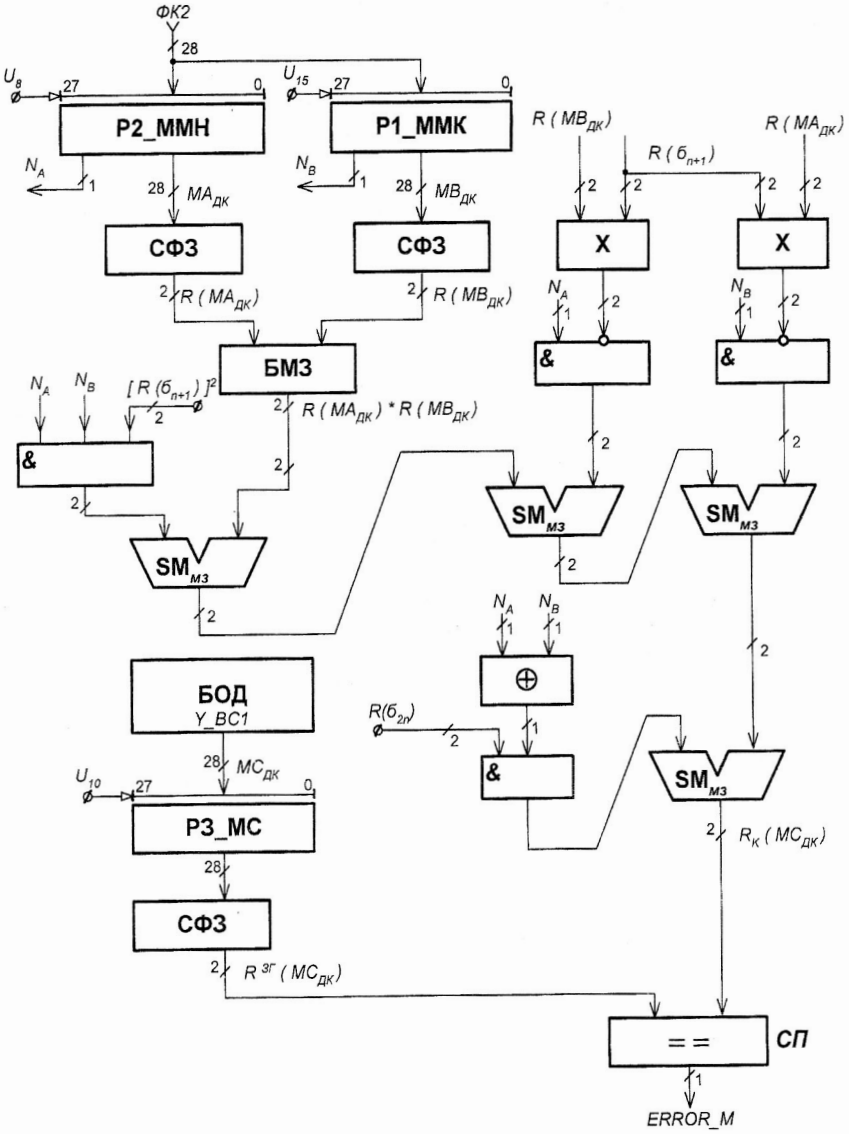


Рисунок 19. Функціональна схема блоку для контролю (перевірки) множення мантис за модулем три (фрагмент 5)

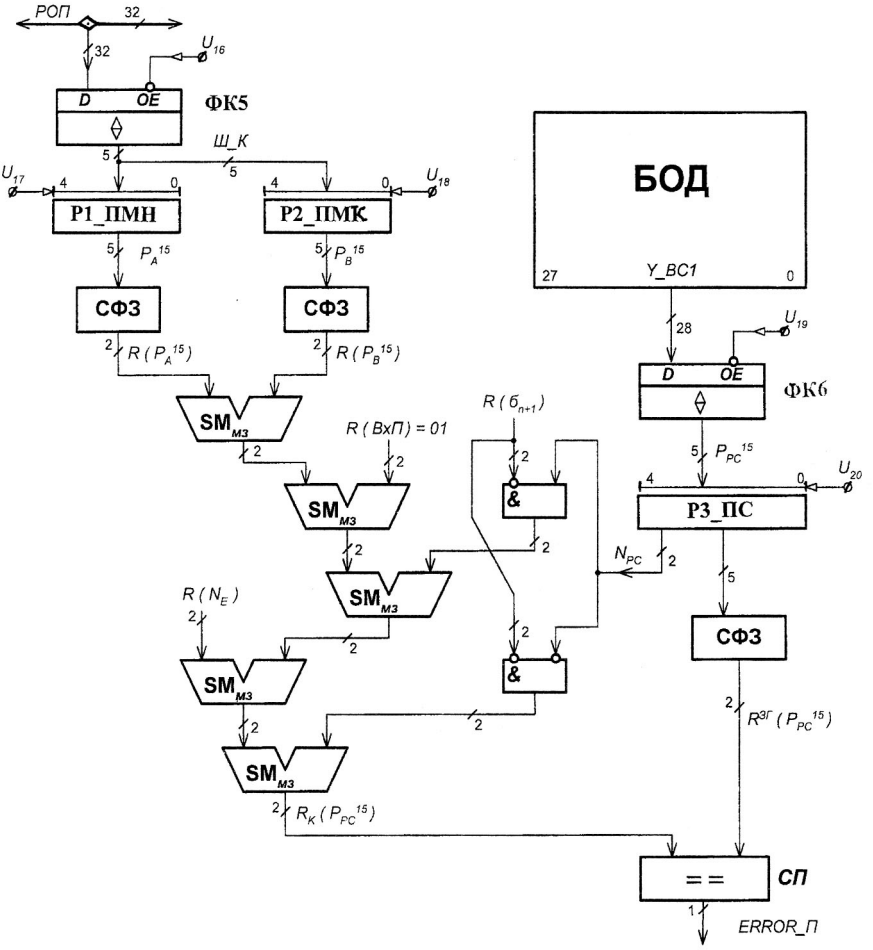


Рисунок 20. Функціональна схема блоку для контролю (перевірки) підсумовування порядків за модулем три (фрагмент 7).

- **ФК5** і **ФК6** - формувачі 5 – и розрядних порядків у коді з негативним нулем.

Обчислювальний алгоритм контролю перевірки операції підсумовування порядків формально складається з наступних послідовних кроків:

1. Порядки ПМн та ПМк, які знаходяться на виходу ФК5, запам'ятовуються в регістрах Р1_ПМН та Р2_ПМК блоку контролю.
2. З використанням СФЗ знаходяться згортка порядків, тобто коди $R[(P_A)^{15}]$ і $R[(P_B)^{15}]$.
3. З використанням суматорів за модулем три (SM_{m3}) формується контролювальний код $R_K[(P_{PC})^{15}]$ згідно з (13).
4. Результат операції підсумовування початкових порядків з виходу БОД Y_ВС1 та формувача ФК6 запам'ятовується у регістру Р3_ПС БК.
5. Знаходиться код згортки результату операції підсумовування порядків$R^{3Г}[(P_{PC})^{15}]$.
6. З використанням схеми порівняння (СП) виконується аналіз кодів $R_K[(P_{PC})^{15}]$ та $R^{3Г}[(P_{PC})^{15}]$. При розбіжності кодів формується сигнал помилки ERROR_П і відбувається апаратна зупинка роботи МОП.

2.4. Розробка функціональних та принципових схем блоків МОП

Після побудови функціональних схем та алгоритмів роботи МОП може бути виконана розробка окремих блоків: оперативної та мікропрограмної пам'яті, БОД, блока синхронізації, блока контроля та схем обрамлення (регістрів, формувачів кодів, лічильників).

Блоки МІІП та ППА, як правило, будуються на мікросхемах ПЗП (ROM) або ПЛМ. Як ROM можуть бути використані ТТЛ - схеми з діодами Шотки: К556РТ4 (за організацією 256 комірок по 4 розряда кожна) або К556РТ5 (512 х 8). При цьому затримка цих мікросхем дорівнює (або менше) 70 нс [11]. На виходах цих мікросхем знаходяться схеми з віткритим колектором, тому для формування сигналів на виходу необхідно підключити зовнішній резістор, значення якого залежить від схем навантаження [34].

Регістри мікрокоманд (РМК), команд (РК) та буферний регістр (РБ) можуть бути виконані на базі мікросхем типу К555ИР23, який має вісім біт та виконує запис даних по передньому фронту сигналу синхронізації [12].

Як мультиплексор коду ознак (МХ СС) може бути використана мікросхема комутатора 8 х 1 типу К155КП5 [12, 27]. При реалізації мультиплексорів входів АА та АВ БОД (МХ1 і МХ2), як правило, використовуються мікросхеми типу К555КП12 або К531КП11 [29].

Блок оперативній пам'яті (ОП) побудуємо на мікросхемах статичного типу (SRAM) К541РУ1А за організацією 4К х 1 [13 – 15]. Другі приклади проектування блоків статичної та динамичної пам'яті можливо знайти, наприклад, в [16, 35].

Блок обробки даних (БОД) довжиною 28 двійкових розрядів у МОП складається з 7 процесорних секцій К1804ВС1. Приклади проектування варіантів цього блоку можливо знайти, наприклад, в [1].

Формувачі кодів із трьома станами (ФК1 – ФК3) можуть бути реалізовані на мікросхемах типу К155ЛП8 або КР1533ЛП8 [27, 28].

В пояснювальной записки до курсового проекту треба виконати синтез кожного блоку МОП (привести відповідні таблиці роботи і формули), навести функціональні (або принципові) схеми, часові діаграми роботи блоку та разрахунок його параметрів (час спрацьовування, кількість і типи мікросхем та потужність живлення схеми).

2. 5. Розробка таблиць кодування

Після початкового запуску МОП (пункт 2.3.1) управління передається на мікрокоманду з нульовою адресою (АМК = 0). Ця мікрокоманда виконує зчитування команди із основної пам'яті МОП за адресою $А_{поч\_пк}$ = 000 40, яка попередньо була занесена в ЛАК з пульта керування.

Далі обрана команда із РОП пересилається в РК, а в ЛАК додається одиниця, тобто формується адреса наступної команди. Після цього виконується інструкція ФАМ ВУ4 JMAP (MI = 0010), яка формує активний сигнал ME = 0. Цей сигнал вмикає перетворювач початкової адреси (ППА). Останній на основі полів Ф та КОП регістра команд формує на своєму виході початкову адресу мікропрограми (знаходиться в МПП) обраної команди (табл. 30).

Алгоритм роботи ППА. Таблиця 30

РК (2 с. ч.)		Вихід ППА (16 с. ч.) (початкова адреса команди)	Мнемоничне позначення КОП команди та її формат
Ф	К О П		
0 1	0 0 0 0 0 0	1 0 0	STOP (RR)
1 1	0 0 0 0 0 1	0 2 0	MOV_FMR (RS)
1 1	0 0 0 0 1 0	0 3 0	MOV_IMR (RS)
1 0	0 0 0 0 1 1	0 4 0	MOV_FMR (RX)
0 1	0 0 0 1 0 0	0 5 0	MUL_F (RR)
0 1	0 0 0 1 0 1	0 7 0	DEC_I (RR)
1 1	0 0 0 1 1 0	0 8 0	JMP_Z (RS)
1 1	0 0 0 1 1 1	0 9 0	MOV_FRM (RS)

Далі відбувається виконання мікропрограми поточної команди, яка активізована у РК. Після її завершення управління передається на нульову адресу МПП, в якій виконується зчитування із ОП наступної команди. Якщо обрана команда STOP (КОП = 0 0 0 0 0 0), формується сигнал зупинки МОП (Stop_PMK = 1).

При виконанні мікрокоманд умовного переходу треба вибирати ознаку, яка буде на вході СС (ФАМ ВУ4). Необхідно формувати ту ознаку, по якій виконується перехід в поточній команді. Для цієї мети в схемі передбачено мультиплексор МХСС, роботу і алгоритм якого наведено у табл. 31.

Алгоритм роботи мультиплексора МХСС. Таблиця 31

UMX_CC (3-0)	Код виходу МХСС	Пояснення ознаки
0 0 0 0	РК [30]	Команда типу RX (0) або RS (1) (*)
0 0 0 1	Z	Нульовий стан АЛП БОД
0 0 1 0	F3	Старший розряд АЛП БОД
0 0 1 1	$OVR_{НН}$	Переповнення з негативним нулем
0 1 0 0	B2_PK_Z	Нульовий стан бази B2
0 1 0 1	ЛАК [0]	Нульовий розряд ЛАК
0 1 1 0	РОП [31]	Команда RR (0) або RS (RX) (1) (*)
0 1 1 1	РБ [15]	Команда RR (0) або RS (RX) (1) (*)
1 0 0 0	TZ	Стан тригера TZ
1 0 0 1	T_JMP	Стан тригера T_JMP
1 0 1 0	РОП [15]	Команда RR (0) або RS (RX) (1) (*)

Примітка. (*) : 0 - команда RR, 1 - команда RX або RS.

Формат слова мікрокоманди (МК) наведено в табл. 32. Одна МК в цілому містить 87 двійкових розрядів.

Формат мікрокоманди (86 – 0). Таблиця 32.

Розряди МК	Позначення сигналу	Призначення
0 – 2	I (8 – 6)	Код інструкції ВС1 для вказівки приймача результату
3 – 5	I (5 – 3)	Код інструкції ВС1 для вказівки операції АЛП
6 – 8	I (2 – 0)	Код інструкції ВС1 для вказівки джерел операндів
9 – 12	AA (3 – 0)	Адреса каналу А РЗП ВС1
13 – 16	AB (3 – 0)	Адреса каналу В РЗП ВС1
17	C0_BC1	Вхід переносу АЛП ВС1 БОД
18	OE#	Вхід для керування станом виходу Y (з трьома станами)
19 – 47	D (27 – 0)	Зовнішня шина даних ВС1
48 – 51	MI (3 – 0)	Шина коду інструкції мікрокоманди ВУ4
52	CCE	Сигнал дозволу перевірки сигналу умови на вході CC#
53	OEY#	Сигнал дозволу зчитування адреси на вихідну шину Y ВУ4
54	RLD#	Сигнал дозволу запису інформації в регістр РА / СТ ВУ4
55	C0_ВУ4	Сигнал інкрементації лічильника мікрокоманд ВУ4
56 – 68	DA (11 – 0)	Зовнішня шина адреси ВУ4
69 – 70	U_DC_S	Входи дешифратору сигналів зупинки роботи МОП
71 – 75	U_DC	Входи дешифратору керуючих сигналів (DC)
76 – 77	W# / R	Сигнал завдання режиму роботи ОП (0 – запис, 1 – зчитування, 2 – зберігання інформації)
78 – 81	UMX_CC	Сигнал для вмикання дешифратора МХСС
82 – 84	S12	Управляючий вхід мультиплексора МХ1
85 – 86	S34	Управляючий вхід мультиплексора МХ2

В табл. 33 для розглянутого випадку наведено вміст основної пам'яті ОП (у 16 –ій системі числення), тобто програма обчислення функції Y (табл. 25), масив початкових значень зміних α_i (табл. 24), базова адреса (B2), початкове значення індексу (X2). Комірка за адресою 000 39 призначена для розміщення (після закінчення програми) результату Y = +120 (у 10 – ій системі).

Кодування вмісту ОП для розглянутої програми (табл. 25). Таблиця 33

Адреса (16 с. ч.)	Вміст комірки ОП (16 с.ч.)	Пояснення
00000	0000 0000	Код повного нуля
00031	4200 0000	$\alpha_1 = +1$ (float)
00032	C600 0000	$\alpha_2 = -2$ (float)
00033	4700 0000	$\alpha_3 = +3$ (float)
00034	CA00 0000	$\alpha_4 = -4$ (float)
00035	4A80 0000	$\alpha_5 = +5$ (float)
00036	0000 0030	Базова адреса B2 (int)
00038	0000 0004	Початкове значення індексу X2 (int)
00039	5BC0 0000	Результат обчислення Y = 120 (float)
00040 (0)	C100 0035	MOV_FMR (RS)
00041 (0)	C230 0038	MOV_IMR (RS)
00042 (0)	C240 0036	MOV_IMR (RS)
00043 (0)	83E3 4000	MOV_FMR (RX)
00044 (0), (1)	440E 4530	MUL_F (RR), DEC_I (RR)
00045 (0)	C600 0043	JMP_Z (RS)
00046 (0)	C700 0039	MOV_FRM (RS)
00047 (0)	4000 0000	STOP (RR)

На основі побудованих вище ГСА створимо далі таблиці кодування (мікропрограми) мікропрограмної пам'яті (МПП). Робота БОД (на базі ВС1), мультиплексорів МХ1, МХ2 і дешифратора керуючих сигналів DC відображається табл. 34. Робота ФАМ (на базі ВУ4), оперативній пам'яті, мультиплексора ознак (МХСС) відображається табл. 35. В цих таблицях вказана як адреса мікрокоманд, так і їхній вміст. Для скорочення розмірів таблиць значення усіх полів наведено в 16 – й системі числення. Позначка "*" у таблицях відображає байдужий стан окремих розрядів двійкового коду (0 або 1).

Архітектура МОП (А – 0), відноситься до класу дворівневих конвеєтних структур [3, 5]. При послідовній вибірці мікрокоманд одночасно зчитуються дві мікрокоманди (МК): БОД обробляє зчитану із МПП в РМК мікрокоманду, а ФАМ у цьому такті підготовляє адресу наступної МК. З урахуванням того, що у МОП вітсутній регістр стану прапорів, мікрокоманди умовного переходу виконуються послідовно: спочатку у БОД виконується обрана МК і формуються ознаки (Z, N, OVR, ВивП), а потім ФАМ ВУ4 перевіряє необхідний прапор (Z або N або OVR або ВивП) і надалі формує адресу наступної мікрокоманди.

Адреса окремих мікрокоманд у таблиці кодування МПП визначається ГСА команд МОП, на яких праворуч з кожною мікрооперацією вказана адреса (у 16 – й системі числення) відповідної мікрокоманди МПП.

Таблиця кодування БОД (ВС1), DC, МХ1 та МХ2. Таблиця 34

Адреса МК	Б О Д (В С 1)								U_DC(4-0)	S12(2-0)	S34(1-0)	Примітки
	I(8-6)	I(5-3)	I(2-0)	AA(3-0)	AB(3-0)	C0_BC1	OE#	D(27-0)				
Вибірка команди із ОП												
000	1	4	2	*	*	*	*	*	00	*	*	NOP, U0
001	1	4	2	*	*	*	*	*	00	*	*	NOP, U0
002	1	4	2	*	*	*	*	*	01	*	*	NOP, U1
003	1	4	2	*	*	*	*	*	00	*	*	NOP, U0
004	1	4	2	*	*	*	*	*	00	*	*	NOP, U0
005	1	4	2	*	*	*	*	*	03	*	*	NOP, U3
006	1	4	2	*	*	*	*	*	18	*	*	NOP, U24
007	1	4	2	*	*	*	*	*	02	*	*	NOP, U2
008	1	4	2	*	*	*	*	*	19	*	*	NOP, U25
009	1	4	2	*	*	*	*	*	00	*	*	NOP, U0
00A	1	4	2	*	*	*	*	*	1A	*	*	NOP, U26
00B	1	4	2	*	*	*	*	*	1B	*	*	NOP, U27
00C	1	4	2	*	*	*	*	*	01	*	*	NOP, U1
00D	1	4	2	*	*	*	*	*	05	*	*	NOP, U5
00E	1	4	2	*	*	*	*	*	00	*	*	NOP, U0
00F	1	4	2	*	*	*	*	*	04	*	*	NOP, U4
010	1	4	2	*	*	*	*	*	01	*	*	NOP, U1
011	1	4	2	*	*	*	*	*	05	*	*	NOP, U5
012	1	4	2	*	*	*	*	*	04	*	*	NOP, U4
Команда STOP (RR)												
100	1	4	2	*	*	*	*	*	00	*	*	NOP, U0
Команда MOV_FMR (RS)												
020	1	4	2	*	*	*	*	*	06	*	*	NOP, U6
021	2	0	7	*	*	*	*	*	08	*	1	D_BC1→РЗП(AB),U8
022	2	0	7	*	*	*	*	*	09	*	2	D_BC1→РЗП(AB),U9
023	1	4	2	*	*	*	*	*	00	*	*	NOP, U0
Команда MOV_IMR (RS)												
030	1	4	2	*	*	*	*	*	06	*	*	NOP, U6

031	2	0	7	*	*	*	*	*	07	*	1	D_BC1→РЗП(АВ),U7
032	1	4	2	*	*	*	*	*	00	*	*	NOP
Команда MOV_FMR (RX)												
040	3	4	3	*	2	*	0	*	00	*	3	0 → R2, U0
041	3	0	1	*	2	0	0	*	00	1	3	R2 + РЗП (АА) → R2
042	1	4	2	*	*	*	*	*	00	*	*	NOP, U0
043	3	0	1	*	2	0	0	*	00	0	3	R2 + РЗП (АА) → R2
044	1	4	2	*	*	*	*	*	00	*	*	NOP, U0
045	2	0	5	2	2	0	0	*	0C	2	3	R2 + D_BC1 → R2
046	1	4	2	*	*	*	*	*	00	*	*	NOP, U0
047	1	4	2	*	*	*	*	*	00	*	*	NOP, U0
048	1	4	2	*	*	*	*	*	00	*	*	NOP, U0
Підпрограма п / п №1 (порушення адресації)												
04A	1	4	5	2	2	*	0	FF00000	00	2	3	R2 ∧ Маска →Y_BC1
04B	1	4	2	*	*	*	*	*	00	*	*	NOP, U0
04C	1	4	2	*	*	*	*	*	00	*	*	NOP, U0
Підпрограма п/ п №2 (зчитування з ОП операнду)												
04D	1	3	3	2	2	*	0	*	0D	2	3	R2 → Y-BC1 → РАП
04E	2	0	7	*	*	0	0	*	08	*	1	D_BC1→ РЗП(РОН1)
04F	2	0	7	*	*	0	0	*	09	*	2	D_BC1→ РЗП(РОН1)
Команда DEC_I (RR)												
070	2	0	5	*	*	0	0	FFFFFFF	15	4	1	DEC [(РЗП(РОН1)], Z → TZ (U21)
071	1	4	2	*	*	*	*	*	00	*	*	NOP, U0
Команда MULT_F (RR)												
050	0	0	3	*	E	0	0	*	00	*	3	R14 → PQ
051	1	4	2	*	*	*	*	*	00	*	*	NOP
052	2	0	4	A	0	0	0	*	00	2	3	R10 → R0
053	1	4	2	*	*	*	*	*	00	*	*	NOP
Підпрограма MULT												
055	2	4	4	A	A	*	0	*	00	2	3	0 → R10 (C_{CT})
056	1	4	6	*	*	*	0	0000001	00	*	*	RQ ∧ 00…1 → Y
057	4	0	1	0	A	0	0	*	00	2	3	R1 [(R10 + R0), PQ]
058	1	4	2	*	*	*	*	*	00	*	*	NOP
059	1	4	2	*	*	*	*	*	00	*	*	NOP
05A	4	0	3	*	A	0	0	*	00	*	3	R1 [(R10 + 0), PQ]
05B	1	4	6	*	*	*	0	0000001	00	*	*	PQ ∧ 00…1 → Y
05C	2	4	1	0	A	1	0	*	00	2	3	R10 – R0 → R10
05D	0	0	2	*	*	0	0	*	00	*	*	PQ ∨ 0 → PQ

Продовження таблиці 34.

05E	1	4	2	*	*	*	*	*	00	*	*	NOP
05F	1	4	2	*	*	*	*	*	00	*	*	CONT, NOP
060	2	0	4	A	7	0	0	*	00	2	3	R10 + 0 → R7
061	7	0	3	*	7	0	0	*	00	*	3	L1 (R7) → R7
062	7	0	3	*	A	0	0	*	00	*	3	L1 (R10) → R10
063	2	0	5	1	1	0	0	FFFFFFF	00	2	3	R1 + $(-1)^{15}$ → R1
064	2	0	1	F	1	0	1	*	00	2	3	R1 + R15 → R1
065	2	0	5	1	1	0	0	8000000	00	2	3	Інвертув. знаку R1
066	1	4	2	*	*	*	*	*	00	*	*	NOP
067	7	0	3	*	7	0	0	*	00	*	3	L1 (R7) → R7
068	1	4	2	*	*	*	*	*	00	*	*	NOP
069	1	4	2	*	*	*	*	*	00	*	*	NOP
Команда MOV_FRM (RS)												
090	1	0	3	*	*	0	0	*	0A	*	1	М → РВхД
091	1	0	3	*	*	0	0	*	0B	*	2	П → РВхД, S2_РК → РАП
092	1	4	2	*	*	*	*	*	0E	*	*	NOP, РВхД → РОП
093	1	4	2	*	*	*	*	*	00	*	*	NOP, U0
Команда JMP_F (RS)												
080	1	4	2	*	*	*	*	*	00	*	*	NOP
081	1	4	2	*	*	*	*	*	00	*	*	NOP
082	1	4	2	*	*	*	*	*	16	*	*	NOP, S2_РК → ЛАК

Таблиця кодування ФАМ (ВУ4), ОП, МХСС. Таблиця 35

Адреса МК	Ф А М (В У 4)						STOP	W# / R	UMX_CC	Примітки
	MI(3–0)	CCE	OEY#	RLD#	C0_BY4	DA (11 – 0)				
Вибірка команди із ОП										
000	3	1	0	1	1	002	3	2	9	CJP, Q_TJ = 1? Да → 2
001	3	1	0	1	1	00E	3	2	5	CJP, ЛАК [0] = 1? Да → E
002	E	*	0	1	1	*	3	1	*	CONT, читання ОП
003	3	1	0	1	1	009	3	2	5	CJP, ЛАК [0] = 1? Да → 9
004	3	1	0	1	1	007	3	2	6	CJP, ЛАК [31] = 1? Да → 7
005	E	*	0	1	1	*	3	2	*	CONT
006	2	*	0	1	0	*	3	2	*	JMAP, декодування КОП
007	E	*	0	1	0	*	3	2	*	CONT
008	3	0	0	1	0	006	3	2	*	CJP, БП на 6
009	3	1	0	1	1	00B	3	2	A	CJP, РОП [15] = 1? Да → B

Продовження таблиці 35.

00A	3	0	0	1	1	008	3	2	*	CJP, БП на 8
00B	E	*	0	1	0	*	3	2	*	CONT
00C	E	*	0	1	0	*	3	1	*	CONT, читання ОП
00D	3	0	0	1	0	006	3	2	*	CJP, БП на 6
00E	3	1	0	1	0	012	3	2	7	CJP , РБ [15] = 1 ? Да → 12
00F	E	*	0	1	1	*	3	2	*	CONT
010	E	*	0	1	1	*	3	2	*	CONT
011	3	0	0	1	0	006	3	2	*	CJP, БП на 6
012	3	0	0	1	0	006	3	2	*	CJP, БП на 6
Команда STOP (RR)										
100	0	*	0	1	1	*	0	2	*	JZ, БП на адресу 0
Команда MOV_FMR (RS)										
020	E	*	0	1	1	*	3	1	*	CONT, читання із ОП
021	E	*	0	1	1	*	3	2	*	CONT
022	E	*	0	1	1	*	3	2	*	CONT
023	0	*	0	1	0	*	3	2	*	JZ, БП на адресу 0
Команда MOV_IMR (RS)										
030	E	*	0	1	1	*	3	1	*	CONT, читання із ОП
031	E	*	0	1	1	*	3	2	*	CONT,
032	0	*	0	1	0	*	3	2	*	JZ, БП на адресу 0
Команда MOV_FMR (RX)										
040	E	*	0	1	1	*	3	2	*	CONT
041	1	0	0	1	0	04A	3	2	*	CJS (виклик п / п № 1)
042	3	1	0	1	1	044	3	2	4	CJP, B2_PK_Z = 1?
043	E	*	0	1	1	*	3	2	*	CONT
044	1	0	0	1	0	04A	3	2	*	CJS (виклик п / п № 1)
045	E	*	0	1	1	*	3	2	*	CONT
046	1	0	0	1	0	04A	3	2	*	CJS (виклик п / п № 1)
047	1	0	0	1	0	04D	3	2	*	CJS (виклик п / п № 2)
048	0	*	0	1	0	*	3	2	*	JZ, БП на адресу 0
Підпрограма п / п №1 (порушення адресації)										
04A	3	1	0	1	1	04C	3	2	1	CJP, Z = 1 ?
04C	A	0	0	1	0	*	3	2	*	CRTN (вихід з п / п № 1)
04B	E	*	0	1	1	*	2	2	*	ABOCT_2 (зупинка МОП)
Підпрограма п/ п №2 (зчитування з ОП операнду)										
04D	E	*	0	1	1	*	3	1	*	CONT, читання із ОП
04E	E	*	0	1	1	*	3	2	*	CONT
04F	A	0	0	1	0	*	3	2	*	CRTN, (вихід з п / п № 2)
Команда DEC_I (RR)										
070	E	*	0	1	1	*	3	2	*	CONT
071	0	*	0	1	0	*	3	2	*	JZ, БП на адресу 0

Команда MOV_FRM (RS)										
090	E	*	0	1	1	*	3	2	*	CONT
091	E	*	0	1	1	*	3	2	*	CONT
092	E	*	0	1	1	*	3	0	*	CONT, запис в ОП
093	0	*	0	1	0	*	3	2	*	JZ, БП на адресу 0
Підпрограма MULT										
055	4	*	0	0	1	01A	3	2	*	PUSH, 26 → PA / CT
056	3	1	0	1	1	05A	3	2	1	CJP, Z = 1?
057	E	*	0	1	1	*	3	2	*	CONT
058	8	*	0	1	1	*	3	2	*	RFCT
059	3	0	0	1	0	05B	3	2	*	CJP, БП на адресу 05B
05A	3	0	0	1	0	058	3	2	*	CJP, БП на адресу 05B
05B	3	1	0	1	1	05D	3	2	1	CJP, Z = 1? (знак Мк)
05C	E	*	0	1	1	*	3	2	*	CONT
05D	3	1	0	1	1	060	3	2	2	CJP, F3 = 1?
05E	3	0	0	1	1	060	3	2	*	БП на адресу 060
05F	E	1	0	1	1	*	3	2	0	CONT
060	3	1	0	1	1	067	3	2	2	CJP, F3 = 1?
061	3	1	0	1	1	064	3	2	2	CJP, F3 = 1?
062	E	*	0	1	0	*	3	2	*	CONT
063	E	*	0	1	1	*	3	2	*	CONT
064	3	1	0	1	1	069	3	2	3	CJP, OVR_{HH} = 1?
065	E	*	0	1	1	*	3	2	*	CONT
066	A	0	0	1	0	*	3	2	*	CRTN (вихід з п / п)
067	3	1	0	1	1	062	3	2	2	CJP, F3 = 1?
068	3	0	0	1	0	064	3	2	*	БП на адресу 064
069	E	*	0	1	1	*	1	2	*	ABOCT_1 (зупинка МОП)
Команда MUL_F (RR)										
050	E	*	0	1	1	*	3	2	*	CONT
051	1	0	0	1	0	055	3	2	*	CJS, виклик п / п MULT
052	E	*	0	1	1	*	3	2	*	CONT
053	0	*	0	1	0	*	3	2	*	JZ, БП на адресу 0
Команда JMP_F (RS)										
080	3	1	0	1	1	082	3	2	8	CJP, TZ = 1? Да → 082
081	0	*	0	1	0	*	3	2	*	JZ, БП на адресу 0
082	3	0	0	1	1	081	3	2	*	БП на адресу 081

2.6. Розрахунок параметрів системи синхронізації МОП

Розглянемо алгоритм розрахунку окремих часових інтервалів тактових сигналів для конкретної архітектури МОП, яка наведена на рис. 21. Припустимо, що МОП синхронізується з використанням сигналів $\overline{G1}$ і $\overline{G2}$. При цьому припустимо, що регістри МОП виконані на DC – тригерах зі спрацьовуванням по позитивному фронту (н → в) тактового сигналу (мікросхема К155ТМ2 [27]), а ЛАК - на основі мікросхеми К155ИЕ7 [12]. Синхросигнал G1 будемо подавати на С – входи РМК та в БОД (РЗП, РQ, РА і РВ), сигнал G2 - на С- входи регістрів РК, РБ, РАП, РОП, РВхД, ЛАК, а також ФАМ ВУ4 (РС). Припустимо також, що оперативна пам'ять являє собою масив запам'ятовуючих комірок (МЗК).

Звичайний цикл роботи МОП (якщо не ураховувати роботи оперативної пам'яті) починається з надходженням (з виходу генератора G) сигналів синхронізації $\overline{G1}$ і $\overline{G2}$, після чого інформація передається від одних компонентів пристрою до інших. Припустимо, що зміна сигналів синхронізації відбувається згідно із рис. 22. Тоді для періоду синхросигналів такту роботи системи можуть бути записані співвідношення:

$$T_1 \geq t_{12} + t_{23} + t_{34} + t_{45}; \tag{17}$$

$$T_2 \geq t_{34} + t_{45} + t_{56} + t_{67}. \tag{18}$$

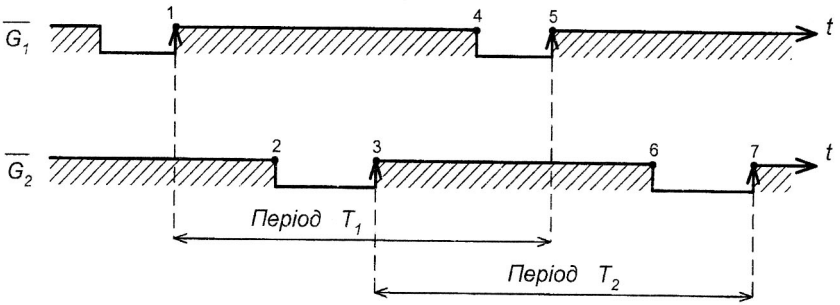


Рисунок 22. Часові діаграми тактових сигналів ($T_1 = T_2$)

Спочатку розглянемо розрахунок окремих інтервалів в такту з урахуванням інформаційного тракту МОП (рис. 23). В схемі тактовий сигнал $\overline{G1}$ подається на РМК та БОД (із 7 – и секцій ВС1) і розповсюджується шляхами, які вказані пунктирними лініями. Згідно з цим після спрацьовування РМК (по позитивному фронту сигналу $\overline{G1}$) на виході дешифратора керуючих сигналів (DC) із затримкою $t_{ЗАТ}^{DC} = 20$ нс з'являються керуючі сигнали U_i, які потім подаються у відповідні блоки МОП. Після спрацьовування мультиплексорів МХ1 і МХ2 (із затримкою $t_{ЗАТ}^{MX} = 20$ нс) на входах адресних портів БОД з'являються коди АА і АВ поточної мікрокоманди, які разом із інструкцією І та переносом на вході С0_ВС1 (від РМК) надходять далі у блок обробки даних. Після цього БОД виконує задану арифметичну або логічну операцію. У "гіршому" випадку

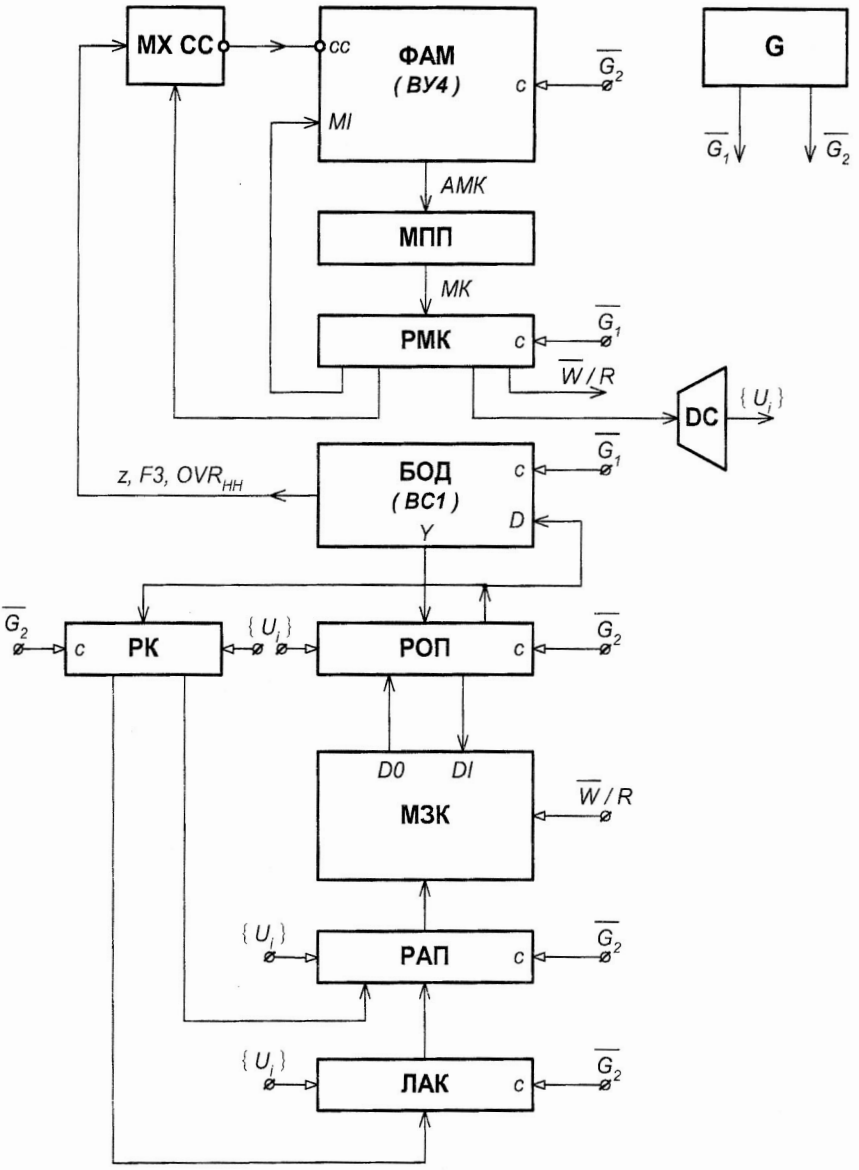


Рисунок 21. Архітектура МОП

(при виконанні арифметичних операцій) на виводі переносу (ВивП) із затримкой $t_{ЗАТ}^{БОД}$ з'являється сигнал переносу. Далі цей сигнал переносу подається на мультиплексор ознак (МХСС) і після його спрацьовування (із затримкою $t_{ЗАТ}^{МХСС} = 20$ нс) з'являється на виводі (СС) мультиплексора, який потім подається на одноіменний вхід ФАМ (ВУ4).

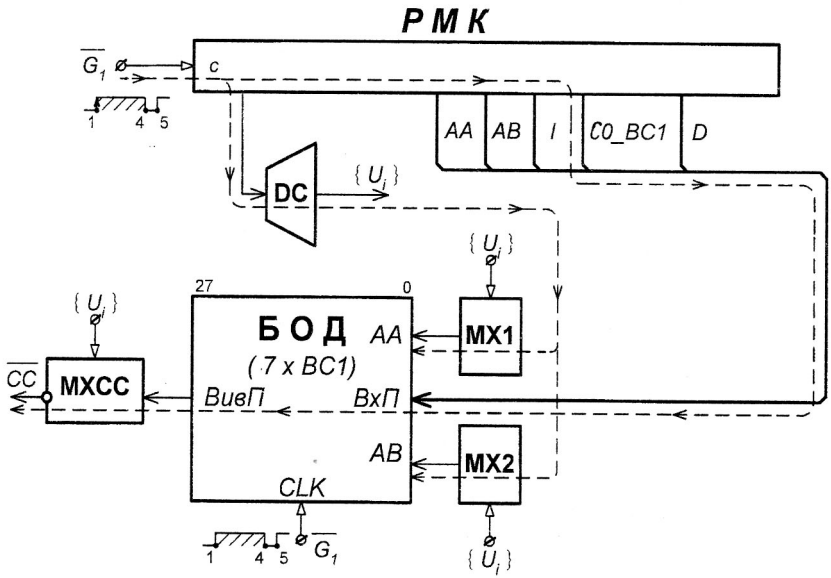


Рисунок 23. Схема розповсюдження керуючих сигналів в інформаційному тракті МОП

Таким чином, для надійного спрацьовування інформаційного тракту МОП інтервал t_{12} необхідно розраховувати слідуючим чином:

$$t_{12}^{*} \geq t_{ЗАТ}^{РМК} + t_{ЗАТ}^{DC} + t_{ЗАТ}^{МХ} + t_{ЗАТ}^{БОД} + t_{ЗАТ}^{МХСС}, \qquad (19)$$

де $t_{ЗАТ}^{РМК}$ - час затримки РМК (25 нс).

Надалі розглянемо алгоритм розрахунку параметрів сигналу синхронізації однієї МПС БОД, який розповсюджується згідно з схемою, яка наведена на рис. 24. Згідно з цією схемою у "гіршому" випадку для надійного спрацьовування регістрів – приймачів ВС1, тривалість активного рівня (⎍) тактового сигналу (G_1) та керуючих кодів I, AA і AB мусить бути не менше:

$$t_{BC1} \geq \max\left[\left(t_{КЛС}^{BC1} + t_{ЗАТ}^{АЛП} + t_{ЗАТ}^{ЗСВР} \right), \left(t_{КЛС}^{BC1} + t_{ЗАТ}^{АЛП} + t_{ЗАТ}^{МХ} + t_{ЗАТ}^{PQ} \right) \right]; \qquad (20)$$

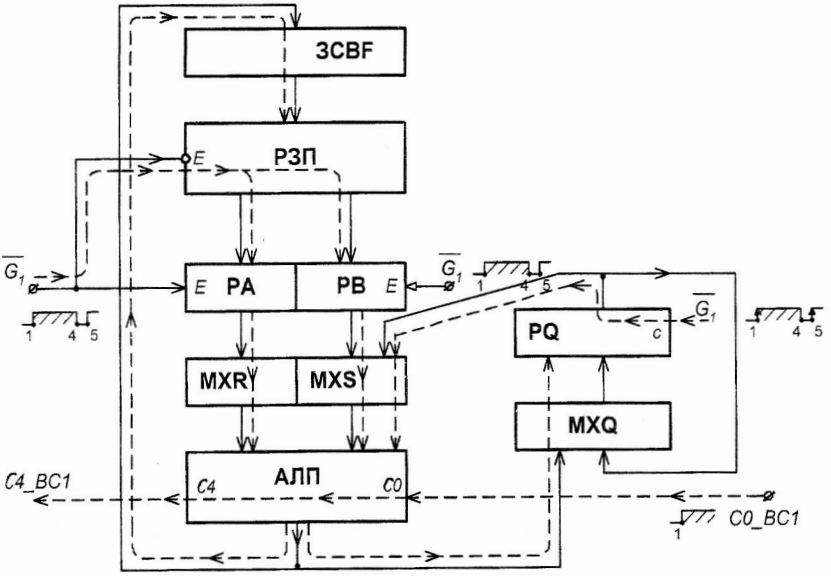


Рисунок 24. Схема для розрахунку системи синхронізації у БОД, який використовує одну секцію ВС1

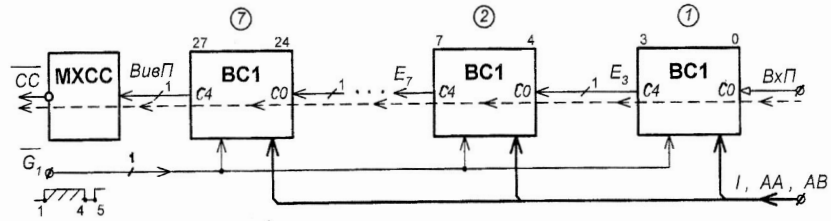


Рисунок 25. Схема поширення переносу в БОД із семи секцій

$$t_{КЛС}^{ВС1} \geq t_{ЗЧ}^{РЗП} + t_{ЗАП}^{РА} + t_{ЗАТ}^{MXR},$$

де $t_{ЗАТ}^{PQ}$ - затримка регістра PQ (25 нс), який спрацьовує по фронту (н $\rightarrow$ в) $\overline{G}_1$;

$t_{ЗЧ}^{РЗП}$ - час вибірки інформації з регістрової пам'яті ВС1 (20 нс);

$t_{ЗАТ}^{РА}$ - затримка прозорих регістрів РА і РВ ВС1 (10 нс);

$t_{ЗАТ}^{MX} = t_{ЗАТ}^{MXQ}(t_{ЗАТ}^{MXR})$ - затримка мультиплексорів MXQ і MXR (MXS) відповідно (10 нс);

$t_{ЗАТ}^{ЗСВF}$ - затримка КЛС зсувача ЗСВF (10 нс);

$t_{ЗАТ}^{АЛП}$ - затримка КЛС АЛП ВС1 (40 нс).

При розрахунку синхронізації у БОД із декількох секцій ВС1 (рис. 25) необхідно ураховувати час поширення переносу при виконанні арифметичних операцій. Розглянемо розрахунок часу поширення переносу, якщо в БОД не використовується схема прискореного переносу К1804ВР1. У молодшій МПС перенос E_3 на виході схеми затримується відносно керуючих кодів I, AA і AB та переносу на вході ВхП на час

$$t_{E3} = t_{КЛС}^{ВС1} + t_{ЗАТ}^{АЛП} = 40 + 40 = 80 \text{ (нс)}.$$

В інших секціях БОД (середніх і старшої) затримка переносу на виході відносно переносу на вході, очевидно, значно менше і включає затримку тільки в схемі АЛП ВС1, так як КЛС ВС1 всіх МПС БОД спрацьовують паралельно. В зв'язку з цим, наприклад, для другої секції затримка переносу E_7 відносно переносу E_3 складає $t_{E7} = t_{ЗАТ}^{АЛП2} = 40$ нс.

У зв'язку з цим, у БОД, наприклад, із семи секцій ВС1 затримка суми і вивідного переносу ВивП відносно початку активного рівня керуючих сигналів (I, AA, AB і ВхП) буде дорівнювати:

$$t_{ЗАТ}^{БОД7} = t_{E3} + 6 * t_{ЗАТ}^{АЛП} = 80 + 6 * 40 = 320 \text{ нс}. \qquad (21)$$

З урахуванням (19) і (21) одержуємо $t_{12} \geq 405$ нс.

Для надійного запису інформації у регістри ВС1 часовий інтервал t_{45} необхідно вибирати за умови:

$$t_{45} \geq \max[t_{ПІДГ}^{РМК}, t_{ПІДГ}^{PQ}, t_{ЗАП}^{РЗП}], \qquad (22)$$

де $t_{ПІДГ}^{РМК}$, $t_{ПІДГ}^{PQ}$ - час підготовки (тривалість неактивного рівня синхросигналу $\overline{G}_1$ ⊔) DC – тригерів РМК і PQ відповідно (20 нс);

$t_{ЗАП}^{РЗП}$ - час запису (тривалість неактивного рівня синхросигналу $\overline{G}_1$ ⊔) інформації у прозорі D – тригери регістрової пам'яті ВС1 (20 нс).

Остаточно одержимо $t_{45} \geq 20$ нс.

Розглянемо далі розрахунок параметрів синхросигналу для керуючого тракту МОП (рис. 26), тобто формувача адреси мікрокоманд (ФАМ). При цьому припустимо, що на

ФАМ, який побудовано на базі ВУ4, надходить тактовий сигнал $\overline{G2}$. Як відомо, лічильник адреси мікрокоманд ВУ4 складається із схеми інкрементора (INC) та регістра (РС) на DC – тригерах із спрацьовуванням по позитивному фронту тактового сигналу (в нашому випадку $\overline{G_2}$). Тому для надійного спрацьовування DC – тригерів РС часові інтервали t_{23} і t_{23} необхідно обирати за умови:

$$t_{12} \geq t_{MX}^{BY4} + t_{INC}^{BY4}; \tag{23}$$

$$t_{23} \geq t_{ПІДГ}^{PC}, \tag{24}$$

де t_{MX}^{BY4}, t_{INC}^{BY4} - вітповідно затримки МХ та INC ВУ4 (25 нс і 25 нс);

$t_{ПІДГ}^{PC}$ - час підготовки до спрацьовування DC – тригерів РС (20 нс).

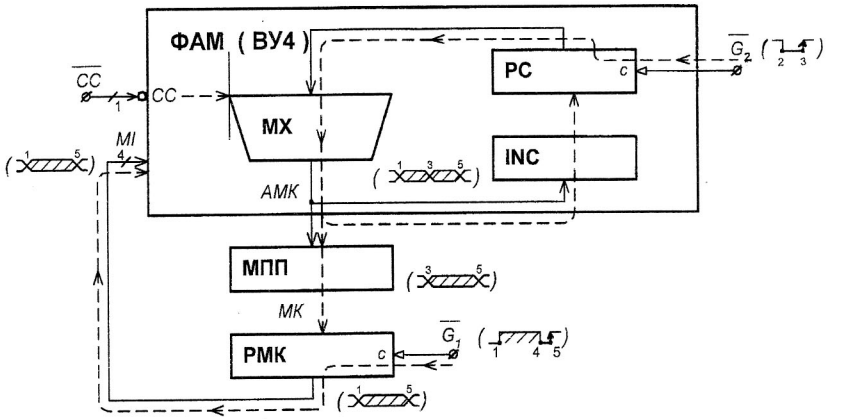


Рисунок 26. Схема поширення сигналів у КЛС керуючого тракту МОП (ФАМ)

До початку підготовки РМК (t_4) на його інформаційних входах необхідно забезпечити сталі значення сигналів, тому інтервал t_{34} необхідно вибирати згідно з нерівності:

$$t_{34} \geq t_{ЗАТ}^{PC} + t_{MX}^{BY4} + t_{ЗАТ}^{МПП}, \tag{25}$$

де $t_{ЗАТ}^{PC}$ - затримка регістру РС ВУ4, який спрацьовує після t_3 (25 нс);

t_{MX}^{BY4} - затримка МХ ВУ4 (25 нс);

$t_{ЗАТ}^{МПП}$ - затримка мікропрограмної пам'яті (70 нс).

Остаточно одержимо $t_{34} \geq 120$ нс.

Аналогічно розраховуються інтервали t_{56} та t_{67}, тому $t_{56} = t_{12}$ і $t_{67} = t_{23}$.

Деякі мікрокоманди формують керуючи сигнали зчитування або запису інформації в основну (оперативну) пам'ять МОП. У разі роботи ОП при розрахунку часових параметрів тактових сигналів G_1 і G_2 треба враховувати затримки часу ОП, РАП, РОП та ЛАК. Припустимо, що звернення до оперативній пам'яті у режимі запису даних виконується в МОП за два такта роботи (дві мікрокоманди). Це обумовлено тим, що пересилка даних та адреси (вітповідно в РОП та РАП) відбувається під дією різних керуючих сигналів (наприклад, U_{14} і U_{11}), які формує дешифратор керуючих сигналів (DC) у різних тактах. Тому під дією однієї мікрокоманди адреса із зовнішнього джерела даних (ЛАК, РК або БОД) записується в РАП, а під дією другої мікрокоманди дані пересилаються в РОП та формується керуючий сигнал дозволу запису в пам'ять (W / R = 0), після чого масив запам'ятовуючих комірок (МЗК) виконує операцію запису (після спрацьовування РОП) в інтервалі t_{34}.

Операція зчитування даних із пам'яті відбувається після пересилки адреси в РАП та подачи сигналу зчитування даних (W / R = 1). До t_4 (до початку режиму підготовки РМК) на виході DO пам'яті мусять сформуватися стабільні дані, які можуть бути далі записуватися в БОД, РК, РБ або РВхД.

Слід також відзначити наступне. Якщо час спрацьовування основної пам'яті ($t_{ЗАТ}^{ОП} = 500$ нс) відомо на етапі розробки функціональних та принципових схем МОП, його можливо враховувати при розрахунку довжини тактових інтервалів. При такому підході можливо вилучити із ГСА МОП умовні вершини перевірки сигналу завершення роботи ОП ($E_{ОП}$) і значно скоротити розміри таблиці кодування МПП. Крім того, запис команд в РК та обчислення виконавчої адреси операнду в цьому разі виконуються в МОП двома різними мікрокомандами і тільки потім (під дією керуючих сигналів) ця адреса пересилається в РАП.

При цих умовах (з позиції обрамлення ОП) тривалість окремих інтервалів сигналів синхронізації необхідно вибирати згідно з нерівностями:

$$t_{12} \geq \max [t_{12}^{*}, (t_{ЗАТ}^{РМК} + t_{ЗАТ}^{DC} + t_{ЗАТ}^{КЛС\_ЛАК}), (t_{ЗАТ}^{РМК} + t_{ЗАТ}^{DC} + t_{ЗАТ}^{КЛС\_РАП})]; \quad (26)$$

$$t_{23} = t_{67} \geq \max [t_{ПІДГ}^{РОП}, t_{ПІДГ}^{РАП}, t_{ПІДГ}^{РОП}, t_{ПІДГ}^{ЛАК}, t_{ПІДГ}^{РК}]; \quad (27)$$

$$t_{34} = t_{78} \geq \max [((t_{ЗАТ}^{РАП}, t_{ЗАТ}^{РОП}) + t_{ЗАТ}^{ОП}), (t_{ЗАТ}^{РС} + t_{МХ}^{ВУ4} + t_{ЗАТ}^{МПП})], \quad (28)$$

де $t_{ПІДГ}^{ЛАК}$, $t_{ПІДГ}^{РОП}$, $t_{ПІДГ}^{РАП}$, $t_{ПІДГ}^{ЛАК}$, $t_{ПІДГ}^{РК}$ - час підготовки спрацьовування відповідно ЛАК, РОП, РАП, ЛАК і РК (20 нс);

$t_{ЗАТ}^{ЛАК}$, $t_{ЗАТ}^{РОП}$, $t_{ЗАТ}^{РАП}$ - затримки відповідно ЛАК, РОП і РАП (25 нс);

$t_{ЗАТ}^{КЛС\_РАП}$ - затримка КЛС на вході РАП (20 нс);

$t_{ЗАТ}^{КЛС\_ЛАК}$ - затримка КЛС на вході ЛАК (20 нс);

$t_{ЗАТ}^{ОП}$ - затримка роботи запам'ятувального масиву оперативної пам'яті (500 нс);

Остаточно одержимо $t_{12} \geq 65$ нс; $t_{23} \geq 65$ нс; $t_{34} \geq 525$ нс; $t_{45} \geq 20$ нс.

Базовим елементом системи синхронізації МОП є генератор тактових сигналів (ГТС), в якості якого можливо використати мікросхему К1804ГГ1 [6,10] або схему, структура якої наведена на рис. 27. До її складу входять задавальний генератор (ЗГ) з частотою f = 10 МГц (період імпульсів T_G = 100нс), розподільник тактових сигналів (РТС) та посилювач – формувач сигналів (ПФС). Як задавальний генератор можливо використати функціональну схему, яка наведена на рис. 28. Для підвищення стабільності частоти схема побудована з використанням кварцевого резонатора BQ з резонансною частотою 10 МГц [37, 43].

ПФС повинен мати необхідний коефіцієнт розгалуження та стрімкі фронти тактових сигналів. В якості ПФС можуть бути рекомендовані мікросхеми серій К531 [29] або КР1533 [28].

На етапі синтезу РТС послідовність тактових сигналів $\overline{G_1}$ і $\overline{G_2}$ доцільно представити у вигляді елементарних імпульсів та пауз, тривалість яких визначається періодом T_G (рис. 29). Далі в схемі РТС для формування тактових сигналів можливо використати автомат з пам'ятью (наприклад, Мілі [30]).

2.7. VHDL – проект симуляції МОП

Мета створення VHDL – проекту МОП полягає в опануванні методів проектування вузлів та блоків комп'ютерної системи з використанням мови апаратного опису VHDL і виконанні симуляції МОП з наступним тестуванням його роботи на функціональному рівні. VHDL – модель МОП сумісно з інструментальним засобом Active – HDL фірми Aldec забезпечує можливість швидкої модифікації архітектури МОП і дозволяє оперативно виявити та усунути хиби проектування [17 – 21].

Модель мікропрограмної пам'яті (МРР) та регістра мікрокоманд (РМК) будемо описувати як пакет (package) та розміщувати його в окремому файлі MPP_FILE (додаток Д4). В подальшому такий підхід дозволить використати цей пакет при опису інших блоків МОП [20]. До пакету MPPPack підключено опис файлу MPP_FILE, з використанням якого відбувається відкриття текстового файлу MPP_content. txt. Останній містить склад МПП (таблиці кодування табл. 34 і табл. 35) у слідуючому вигляді (приведено фрагмент для перших трьох мікрокоманд):

/- Адр. I86 I53 I20 AA AB C0 OE D MI CCE OEY RLD C0 DA Stop UDC W/R UCC S12 S34

/-- Вибірка команди із ОП

000 1 4 2 0 0 0 0 0000000 3 1 0 1 1 002 3 00 2 9 0 0
001 1 4 2 0 0 0 0 0000000 3 1 0 1 1 00E 3 00 2 5 0 0
002 1 4 2 0 0 0 0 0000000 E 1 0 1 1 000 3 01 1 0 0 0

.....

Для роботи з текстовими файлами доцільно користуватися пакетом TEXTIO, якій знаходиться у стандартній бібліотеці IEEE [18]. Цій пакет має опис типів та процедур, які потрібні при виконанні операцій зчитування або операцій запису у текстові файли об'єктів різноманітних типів.

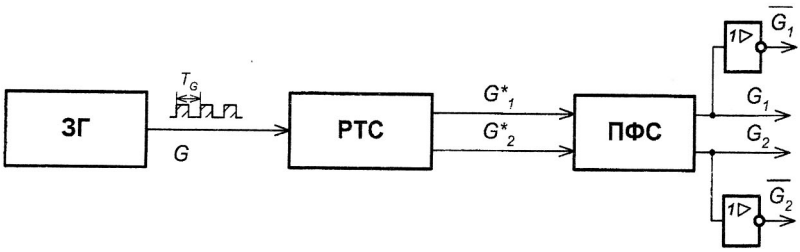


Рисунок 27. Структура генератора тактових сигналів

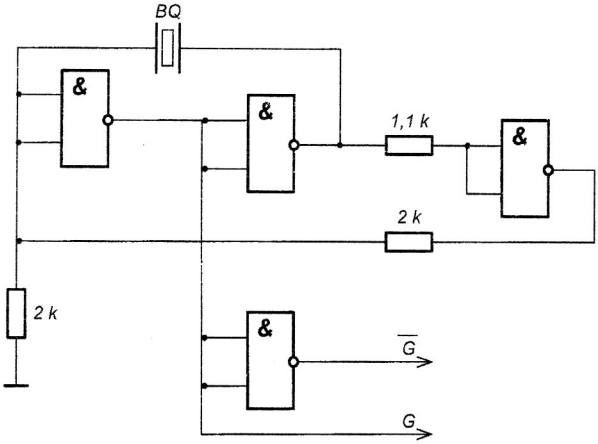


Рисунок 28. Функціональна схема задавального генератора

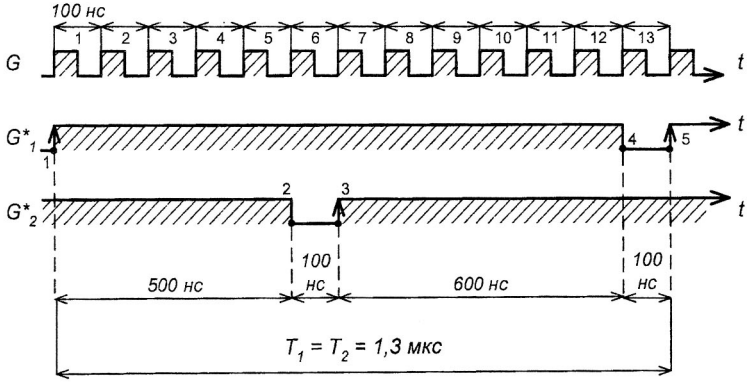


Рисунок 29. Діаграма роботи розподільника тактових сигналів

VHDL - модель МПП наведена в додатку Д4. Спочатку симуляції (змінна now = 0 ps) відбувається ініціалізація МПП шляхом зчитування полів кожної мікрокоманди із окремого рядка файлу MPP_FILE в вітповідний рядок МПП. При виконанні операції зчитування використовується функція read_mode пакету TEXTIO [18]. Для перетворення строкового представлення (у 16 – й системі числення) у двійковий тип даних використовується процедура - функція HstrToBV. Функція SkipSpaces цього пакету використовується на етапі зчитування текстового файлу для відопромлення пропусків, які розділяють окремі поля мікрокоманди. В подальшому у процесі симуляції МОП при кожній зміні адреси (змінна ADDR) в МПП кожного разу обирається відповідна комірка, яка містить актуальну мікрокоманду.

VHDL – модель МПП має наступні порти входу та виходу:

- ADDR - беззнакове ціле число, яке задає номер (адресу) мікрокоманди;
- MPP_CELL – мікрокоманда вітповідного формату (табл. 32).

Беззнакова ціла константа MPP_size задає розрядність адреси МПП.

Підкреслемо, що аналогічном чином відбувається ініціалізація основної оперативної пам'яті МОП, у яку із зовнішнього текстового файлу OP_content.txt пересилається код поточної програми пристрою.

Після зчитування МК із комірки МПП мікрокоманда позитивним фронтом тактового сигналу G_1 (CLK_PMK) записується у регістр мікрокоманд (РМК). В додатку Д4 наведено інтерфейс портів сигналів РМК та проведено опис його архітектури з використанням функції чутливості до вітповідного тактового сигналу. Інтерфейс РМК також має у своєму складі опис окремих полів мікрокоманди.

Після спрацьовування РМК вітповідні керуючи сигнали надходять в окремі блоки МОП і далі ініціалізують ті чи інши мікрооперації.

Додаток Д4 також вміщує фрагменти VHDL – моделей МПС ВС1 та БОД із 7 секцій ВС1. VHDL - модель секції ВС1 складається із окремих компонентів (АЛП, MXR, MXS, РЗП, регістра PQ та інших блоків), які взаємодіють з використанням вітповідних процесів з задавальним складом функцій чутливості сигналів.

VHDL - модель БОД із 7 секцій ВС1 побудована з використанням структурної технології VHDL, яка дозволяє проводити поєднання окремих підсистем [17]. При цьому використовується оператор мови VHDL port map ( ). Такий підхід дозволяє створювати копії інтерфейсів окремих МПС ВС1, які потім убудуються в тіло вітповідної архітектури. При цьому карта портів дає опис зв'язків інтерфейсу компонента з сигналами архітектури. Можливо також окремий порт не поєднувати з другими компонентами, тоді його треба описати ключовим словом open.

Аналогічним чином створені фрагменти моделі ФАМ на основі ВІС ВУ4 (Д4). У якості компонентів VHDL – моделі ФАМ використовані стек, регістр РС, інкрементор INC, мультиплексор адреси MX, регістр адреси /лічильник РА / СТ.

У додатку наведено також фрагмент VHDL – моделі схеми МОП в цілому, яка поєднує усі блоки та елементи пристрою. Створені також опис сигналів та окремих компонентів МОП, та вказана їх взаємодія.

Після завершення опису VHDL –моделі МОП виконується компіляція. Для цього в системі Active – HDL є вітповідна команда Compile All із меню Design. На етапі компіляції можливо усунути можливі синтаксичні помилки. Для проведення симуляції роботи МОП необхідно виконати наступні дії:

- вибрати модель поточної схеми МОП (MOP. vhd) в Design Browser;
- створити нову часову діаграму (позначка з ім'ям New WaveForm);
- вибрати MOP. vhd на вкладку Structural в Design Browser, додати у діаграму порти тактових сигналів G_1 і G_2 та задати для кожного з них генератор (Clock) (з параметрами часових інтервалів, які були розраховані раніше) в якості стимуляторів;

- створити новий склад сигналів (позначка New List);
- додати у цей лист усі порти VHDL-моделі МОП;
- провести симуляцію з використанням команд Initialize Simulation та Run For із меню Simulation;
- вибрати опцію Collapse Details з меню List та зберегти список сигналів симуляції у окремому файлі, наприклад, з ім'ям L1_MOP.lst.

Результати симуляції можливо спостерігати безпосередньо на екрані монітору як часову діаграму сигналів, або (після відкриття файлу L1_MOP.lst) як відповідну таблицю. Приклад початкового етапу симуляції МОП наведено у табл. 36. На рис. 30 наведено приклад часової діаграми МОП при обчисленні виконавчої адреси $A_{ВИК2}$ (мікрокоманди з адресами 45, 46, 4A, 4C та 47).

Початковий етап симуляції МОП Таблиця 36.

Time	Delta	G1b	G2b	UBY4/Y	UBY4/MI	UBY4/nCC	ULAK/ADR_LAK	ULAK/LAK0	UBOD/Y_BC	UBOD/AA	UBOD/AB
0.000	2	1	1	ZZZ	U	U	UUUUU	U	UUUUUUU	U	U
1.000 ns	1	1	1	ZZZ	U	U	UUUUU	U	0000000	U	U
2.000 ns	0	1	1	ZZZ	U	0	UUUUU	U	0000000	0	0
5.000 ns	0	1	1	ZZZ	U	0	00000	0	0000000	0	0
10.000 ns	1	1	1	UUU	U	0	00000	0	0000000	0	0
12.000 ns	0	1	1	UUU	0	0	00000	0	0000000	0	0
32.000 ns	1	1	1	000	0	0	00000	0	0000000	0	0
55.000 ns	0	1	1	000	0	0	00040	0	0000000	0	0
400.000 ns	2	0	1	000	0	0	00040	0	0000000	0	0
412.000 ns	0	0	1	000	3	1	00040	0	0000000	0	0
500.000 ns	2	1	1	000	3	1	00040	0	0000000	0	0
600.000 ns	2	1	0	000	3	1	00040	0	0000000	0	0
700.000 ns	2	1	1	000	3	1	00040	0	0000000	0	0
730.000 ns	1	1	1	001	3	1	00040	0	0000000	0	0
800.000 ns	2	0	1	001	3	1	00040	0	0000000	0	0
900.000 ns	2	1	1	001	3	1	00040	0	0000000	0	0
1.000 us	2	1	0	001	3	1	00040	0	0000000	0	0
1.100 us	2	1	1	001	3	1	00040	0	0000000	0	0
1.130 us	1	1	1	002	3	1	00040	0	0000000	0	0
1.200 us	2	0	1	002	3	1	00040	0	0000000	0	0
1.212 us	0	0	1	002	E	1	00040	0	0000000	0	0
1.300 us	2	1	1	002	E	1	00040	0	0000000	0	0
1.400 us	2	1	0	002	E	1	00040	0	0000000	0	0
1.500 us	2	1	1	002	E	1	00040	0	0000000	0	0
1.530 us	1	1	1	003	E	1	00040	0	0000000	0	0
1.600 us	2	0	1	003	E	1	00040	0	0000000	0	0
1.612 us	0	0	1	003	3	1	00040	0	0000000	0	0
1.700 us	2	1	1	003	3	1	00040	0	0000000	0	0

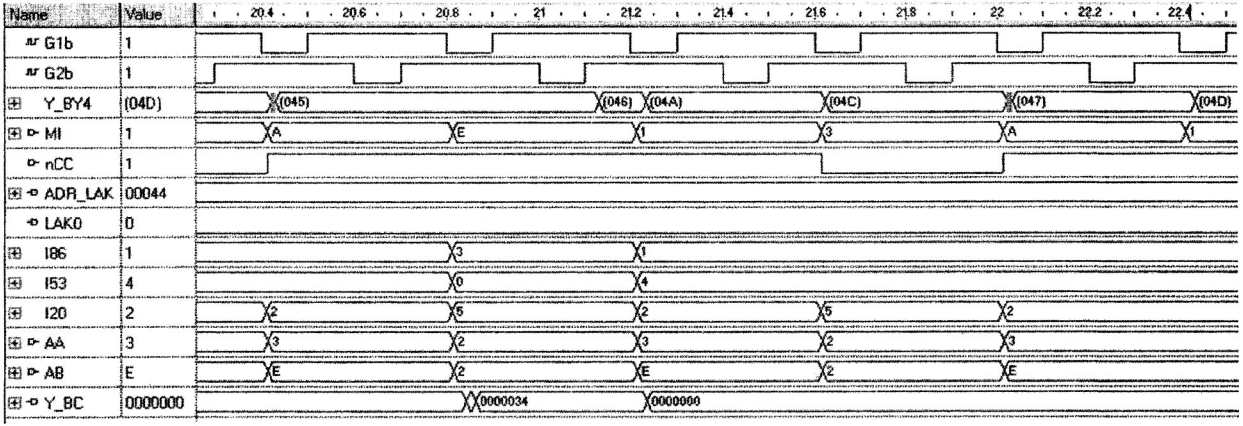


Рисунок 30. Часова діаграма симуляції МОП при обчисленні виконавчої адреси $A_{ВИК2}$

ДОДАТОК

Д1. ПРИКЛАД ТИТУЛЬНОГО ЛИСТА ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

ПОЯСНЮВАЛЬНА ЗАПИСКА

з курсового проекту за курсом "Архітектура комп'ютерів" на тему: "Мікропроцесорний обчислювальний пристрій для реалізації задаваної формули"

Розробив студент групи КС – 03а

_____ Іванов І. П.

Керівник проекту_____ Петров І. В.

Завідуючий кафедрою ЕОМ _____ Сідоров К. М.

Донецьк ДНТУ 2005

Д2. ПРИКЛАД ТЕХНІЧНОГО ЗАВДАННЯ

ЗАТВЕРДЖУЮ
Зав. кафедрою ЕОМ
_____ Святний В. А.
14 лютого 2005 р.

ТЕХНІЧНЕ ЗАВДАННЯ

Студентові групи КС – 03б Воскобойнику Сергію Вікторивочу
Термін здачи курсового проекту 20 травня 2005 року.

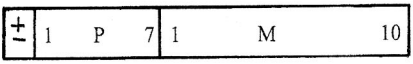
Початкові дані

Варіант N = 30 ;
Архітектура МОП: А – 5;

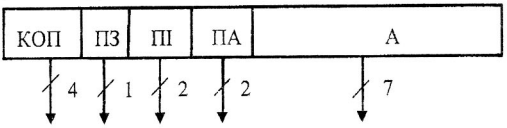
Розрахункова формула:

$$Y_0 = \begin{cases} A \cdot \sum_{i=1}^{N+5} x_i - B; & \text{коли } A \geq B \\ A + B, & \text{коли } A < B \end{cases}$$

Тип процесорних елементів: К1804ВС1, К1804ВУ1;
Алгоритм множення: "А";
Метод множення: Лемана;
Формат операндів з рухомою комою:



Р – зображення порядку у коді з негативним нулем (НН);
М - зображення мантиси у додатковому коді (ДК);
Формат команди F2:



Модуль контроля арифметичних операцій: mod 3;
Параметри оперативної пам'яті МОП: 32К комірок по 32 розряду кожна;
Тип мікросхем RAM: К589РУ1;
Типи мікросхем обрамлення: К155ТМ2, К155ИМ1, К155ИЕ6, К155ЛА3;
Спосіб синхронізації регістра ознак: керована;
Спосіб синхронізації регістрових схем МОП: однофазна;

Розділи пояснювальної записки, що підлягають розробленню:

- аналіз розрахункової формули та приклади її виконання у заданому форматі команди та RAM;
- розробка функцыональних та принципових схем МОП;
- розробка ГСА вибірки із RAM та виконання команд МОП;

- розробка мікропрограм та таблиць кодування;
- розробка VHDL – проекту МОП (цей пункт виконується факультативно);
- розробка часових діаграм роботи МОП та розрахунок системи сінхронізації;

Зміст графічної частини курсового проекту:

- схема електрична функціональна МОП (формат А1);
- схема електрична принципова основних вузлів МОП (формат А1);
- ГСА виконання команд МОП.

Графік виконання курсового проекту:

1 тиждень - аналіз задаваної функції і розробка її реалізації в указаному форматі команди;

2 тиждень - розробка функціональних схем фрагментів МОП;

3 тиждень - розробка ГСА реалізації мінімального набору команд МОП;

4 тиждень - розробка VHDL – проекту МОП (виконується факультативно);

5 – 6 тиждень - розробка мікропрограм виконання арифметичних операцій;

7 – 8 тиждень - розробка функціональних та принципових схем вузлів МОП;

9 тиждень - розробка таблиць кодування;

10 – 11 тиждень - розробка часових діаграм МОП та розрахунок системи синхронізації;

12 – 13 тиждень - оформлення пояснювальної записки;

14 – 15 тиждень - оформлення графічної частини проекту;

16 тиждень - захист проекту.

Дата видачи завдання: 14 лютого 2005 року.

Керівник проекту _____ Петров І. С.

Студент _____ Воскобойник С. В.

ДЗ. СКЛАД ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Титульний ліст
Технічне завдання (подписане та затверджене)
Реферат (з великім штампом)
Анотація (на англійської мові)
Зміст (з указівкою номерів сторінок)

ВСТУП (одна сторінка)

1. АНАЛІЗ РОЗРАХУНКОВОЇ ФОРМУЛИ ТА ПРИКЛАДИ ЇЇ ВИКОНАННЯ У ЗАДАНОМУ ФОРМАТІ КОМАНДИ

2. РОЗРОБКА ФРАГМЕНТІВ ФУНКЦІОНАЛЬНОЇ СХЕМИ МОП

2.1. Блок обробки даних (БОД)
2.2. Блок мікропрограмного керування (БМК)
2.3. Блок контроля за модулем три арифметичних операцій
2.4. Блок схем обладнання МОП (регістри, лічильники і інши вузли)

3. РОЗРОБКА ГСА РЕАЛІЗАЦІЇ КОМАНД МОП

3.1. Вибірка команди із ОП та її декодування
3.2. Вибірка операндів
3.3. Виконання арифметичних операцій з контролем за модулем три
3.3.1. Операція підсумовування з рухомою комою
3.3.2. Операція множення з рухомою комою

4.* РОЗРОБКА VHDL - ПРОЕКТУ МОП ТА ЙОГО СИМУЛЯЦІЯ

(цей пункт проекта виконується факультативно)

5. РОЗРОБКА ПРОГРАМНОГО ЗАБЕСПЕЧЕННЯ МОП

5.1. Формат мікрокоманди
5.2. Таблиці кодування МПП (повинна відповідать розробленої ГСА) окремо для БОД і БМК
5.3. Таблиці кодування для ОП та ППА

6. РОЗРОБКА ФУНКЦІОНАЛЬНИХ ТА ПРИНЦИПОВИХ СХЕМ МОП

6.1. Блок обробки даних
6.2. Блок оперативній пам'яті
6.3. Блок мікропрограмної пам'яті
6.4. Блок синхронізації
6.5. Блок електронного обладнання

7. ПОБУДОВА ЧАСОВИХ ДІАГРАМ РОБОТИ МОП ТА РОЗРАХУНОК ПАРАМЕТРІВ СИНХРОНІЗАЦІЇ

ЗАКЛЮЧЕННЯ

Список використаних джерел
ДОДАТКИ

P. S. До записки необхідно додати архів файлів проекту з результатами симуляції.

Проекти, що не відповідають чинним вимогам, до розгляду не приймаються.

Д4. Файл VHDL – проекту симуляції МОП (фрагмент)

-- Модель МПП (фрагмент)

```
library IEEE;
use IEEE. std_logic_1164. all;
use STD. TEXTIO. all;
use IEEE. STD_LOGIC_TEXTIO. all;
Package MPPPack is
                                        -- декларування файлу вмісту МПП

    file MPP_FILE: TEXT open read_mode is "MPP_content.txt"; -- завдання ім'я текстового файлу МПП
    type MPP_TYPE is record                                  -- завдання формату мікрокоманди МОП
                        ------------------------------------------------------------
            I86 :  bit_vector (2 downto 0);          -- коди полів інструкції ВС1
            I53 :  bit_vector (2 downto 0);
            I20 :  bit_vector (2 downto 0);
            AA :   bit_vector (3 downto 0);          -- коди адресних портів ВС1
            AB :    bit_vector (3 downto 0);
            C0_BC: bit;                              -- вхідной перенос ВС1
            nOE :  bit;                              -- вхід для керування станом виходу Y ВС1
            D      : std_logic_vector (27 downto 0); -- зовнішня шина даних ВС1
                        ------------------------------------------------------------
            MI :   bit_vector (3 downto 0);          -- код інструкції ВУ4
            CCE :  bit;                              -- сигнал дозволу перевірки умови на вході CC#
            nOEY : bit;                              -- сигнал дозволу зчитування адреси ВУ4
            nRLD : bit;                              -- сигнал дозволу запису інформації в РА / СТ
            C0_BY: bit;                              -- сигнал інкрементації лічильника мікрокоманд ВУ4
            DA :   std_logic_vector(11 downto 0);    -- зовнішня шина адреси ВУ4
                        ------------------------------------------------------------
            Stop_RMK : bit_vector (1 downto 0);      -- входи дешифратору сигналів зупинки МОП
            U_DC     : std_logic_vector(4 downto 0); -- входи дешифратору керуючих сигналів
            nW_R     : bit_vector (1 downto 0);      -- сигнал завдання режиму роботи ОП
            UMX_CC   : std_logic_vector(3 downto 0); -- сигнал для вмикання дешифратора МХСС
            S12      : std_logic_vector(2 downto 0); -- управляючий вхід мультиплексора МХ1
            S34      : std_logic_vector(1 downto 0); -- управляючий вхід мультиплексора МХ2
    end record;
    type MPParr is array (NATURAL range <>) of MPP_TYPE;
end MPPPack;
```

-- Опис мікропрограмної пам'яті

```
library IEEE;
use IEEE. STD_LOGIC_1164. all;
use IEEE. STD_LOGIC_UNSIGNED. all;
use IEEE. NUMERIC_STD. all;
use MPPPack. all;
use STD. TEXTIO. all;

entity MPP is                                    -- інтерфейс МПП
    generic (MPP_size: natural : = 12; delay: time := 70 ns);
    port (
            ADDR : in std_logic_vector;
            MPP_CELL : out MPP_TYPE
        );
end MPP;
-----------------------------------------------------------------------------------------------------------------------
architecture MPP of MPP is
begin
    process is
    variable MPP: MPParr (0 to 2**MPP_size-1);
    variable FLINE: LINE;
    variable STRDWORD: string (4 downto 1) ;
```

```
        variable STRBYTE:    string  (1 to 1);
        variable STRWORD:    string  (2 downto 1);
        variable STR3WORD:   string  (3 downto 1);
        variable SPACE:      character;
        variable index:      natural;

                                        -- процедура пропуску усіх пробілів у рядку

procedure SkipSpaces (variable ILINE: inout LINE) is
variable C : character;
begin
        loop
                C : = ILINE (ILINE. all' LEFT);
                if (C = ' ')  then  read (ILINE,C);
                else  exit;
                end  if;
        end  loop;
end  SkipSpaces;
-----------------------------------------------------------------------------------------------------------------------
function  bit_vec2int  (A : BIT_VECTOR)  return  integer  is
variable  RESULT:  integer : = 0;
variable  TMP :  integer : = 1;
begin
 if  A' length  =  0  then  return  RESULT;
 end  if ;
 for  i  in  A' reverse_range  loop
  if a ( i )  =  '1'
   then  RESULT : = RESULT + TMP;
  end  if;
  TMP : = TMP * 2;
 end  loop;
 return  RESULT;
end;
                                -- Функція перетворення 16 – е число у двійковий тип (фрагмент)

procedure  HStrToBV   (variable  Str  :  in   string ;
                       variable  Vec :  out   bit_vector)  is
begin
        for N  in  Str' range  loop
                case  Str ( N )  is
                        when ' 0 ' => Vec  (( N-1 ) * 4+3 downto  (N-1) * 4) : = "0000" ;
                        when ' 1 ' => Vec  (( N-1 ) * 4+3 downto  (N-1) * 4) : = "0001" ;
                        . . . . . .

                        when ' F ' => Vec (( N-1 ) * 4+3 downto  (N-1) * 4) : = "1111" ;
                        when  others => Vec (( N-1 ) * 4+3 downto  (N-1) * 4) : = "0000" ;
                        report  "error reading  from hex  string: "& Str severity  warning ;
                end  case;
        end  loop;
 end  HStrToBV;

variable buff2  : bit_vector  (7 downto 0);
variable buff3  : bit_vector  (11 downto 0);
variable buff   : bit_vector  (3 downto 0);
begin
                if (Now = 0 ps)  then  --  читання у момент часу  0  (до початку виконання програми)
                        index : = 0;   --  індекс комірки пам'яті
                        while  not (endfile (MPP_FILE)  or  index = (2**MPP_size-1))  loop
                                readline (MPP_FILE,  FLINE) ;      -- читання рядка
                        -- аналіз  рядка на нульову довжину та на коментарі
                                if  ( FLINE' length  / = 0 and  FLINE ( 1 ) / = ' / ') then
                                        SkipSpaces  (FLINE) ;
```

```
                                    Read (FLINE, STR3WORD) ;
                                    HStrToBV (STR3WORD, buff3) ;
                                    SkipSpaces (FLINE) ;
                                    Index : = bit_vec2int (buff3) ;
                                    read (FLINE, STRBYTE) ;          -- читання поля I86
                                    HStrToBV (STRBYTE, buff) ;
                                    MPP (index). I86 : = buff (2 downto 0);
                                    SkipSpaces (FLINE) ;             -- пропустити пробіли
                          . . . . . .

        end if;
              end loop;
            else
                  index : = bit_vec2int (to_bitvector ( ADDR ));
                  MPP_CELL < = MPP (index) after delay;          -- читання комірки пам'яті

            end if ;
            wait on ADDR ;
      end process ;
end MPP ;
```

-- Модель РМК (фрагмент)

```
library IEEE;
use IEEE. STD_LOGIC_1164. all;
use MPPpack. all;
use OutFilePack. all;
use STD. TEXTIO. all;
use IEEE. STD_LOGIC_TEXTIO. all;

entity RMK is                                    -- інтерфейс РМК
      generic (delay: time : = 30 ns);
      port(
            MK :          in MPP_Type;
            nOE :         in std_logic;
            CLK_RMK :     in std_logic;
            I86 :         out std_logic_vector ( 2 downto 0 );
            I53 :         out std_logic_vector ( 2 downto 0 );
            I20 :         out std_logic_vector ( 2 downto 0 );
            AA_RMK :      out std_logic_vector ( 3 downto 0 );
            . . . . . .
            S34 :         out std_logic_vector ( 1 downto 0 );
            );
end RMK;
```

```
architecture RMK of RMK is
begin
process (CLK_RMK) is
  variable FLINE: LINE;
      variable MKbuf: MPP_TYPE;
      begin
      if CLR_RMK = ' 1 ' and CLK_RMK' event then
      I86 < = to_stdlogicvector ( MKbuf. I86 ) after delay;
      I53 < = to_stdlogicvector ( MKbuf. I53) after delay;
      I20 < = to_stdlogicvector ( MKbuf. I20) after delay;
      AA_RMK < = to_stdlogicvector ( MKbuf. AA) after delay;
      . . . . . .

      S34 < = MKbuf. S34 after delay;
      end process;
```

```
end RMK;
```

-- Модель ВС1 (фрагмент)

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;
entity ALU is
        generic (delay: time: = 40 ns);                    -- завдання затримки
        port(
                C0 : in  std_logic ;                       -- вхідний перенос ВС1
                R :  in  bit_VECTOR (3 downto 0) ;         -- операнд 1 на вході АЛП
                S :  in  bit_VECTOR (3 downto 0) ;         -- операнд 2 на вході АЛП
                I :  in  std_logic_vector (2 downto 0) ;   -- інструкція мікрокоманди ВС1
                C4 : out  std_logic ;                      -- вивідной перенос ВС1
                OVR :  out bit ;                           -- познака переполнения АЛП
                Z :    out bit ;                           -- познака нульового результату
                F3 :   out bit ;                           -- познака знаку
                F :    out  bit_VECTOR (3 downto 0)        -- вихідна шина АЛП
        );
end ALU;
-----------------------------------------------------------------------------------------------------------
architecture ALU of ALU is                  -- опис архітектури АЛП
begin
        process  ( C0, R, S, I)
        variable C : std_logic : = '0' ;
        variable R1, S1 :  bit_vector (3 downto 0) ;
        variable buf1, Rbuf1, Sbuf1 :  std_logic_vector (3 downto 0) ;
        variable buf, Rbuf, Sbuf : std_logic_vector (4 downto 0) ; -- тимчасові буфери

        begin
                C : = C0 ;
                -- в молодші 4 разряди буферів запам'ятовуються значения R і S
                Rbuf : = '0' & to_stdlogicvector (R) ;  Sbuf : = '0' & to_stdlogicvector (S) ;
                R1 : = R ; R1(3) : = '0' ;   S1 : = S ; S1 (3) : = '0' ;
                -- в молодші 3 разряди буферів запам'ятовуються значения R1 і S1
                Rbuf1 : = to_stdlogicvector (R1) ;  Sbuf1 : = to_stdlogicvector (S1) ;
                buf : = "00000" ; buf1: = "0000" ;
                case I is                                          -- фаза виконання операції АЛП
                        when  "000"  =>  buf : = Rbuf + Sbuf + C ;  buf1 : = Rbuf1 + Sbuf1 + C ;
                        when  "001"  =>  Rbuf : = '1'  &  to_stdlogicvector (R) ;
                                buf : = Sbuf + not ( Rbuf ) + C ;  R1 (3) : = '1' ;
                                Rbuf1 : = to_stdlogicvector (R1) ;
                                buf1 : = Sbuf1 + not ( Rbuf1 ) + C ;
                        when  "010" => Sbuf : = '1' &  to_stdlogicvector (S) ;
                                buf : = Rbuf + not ( Sbuf ) + C ;  S1(3) : = '1' ;
                                Sbuf1 : = to_stdlogicvector (S1) ;
                                buf1 : = Rbuf1 + not ( Sbuf1 ) + C ;
                        when "011" => buf : = Rbuf  or  Sbuf ;  buf1 : = Rbuf1  or  Sbuf1;
                        . . . . . .
                        when  others  => null ;
                end  case;

                C4 <= buf (4) after delay ;                -- формування сигналів ознак АЛП ВС1
                F3 <=  to_bit (buf (3) )  after delay ;
                OVR <= to_bit (buf  (4)  xor  buf1  (3));
                F  < = transport  to_bitvector (buf (3 downto 0)) after  delay ;
                Z <= not  to_bit ( buf (0) or buf (1)  or  buf (2)  or  buf (3) ) after  delay ;
        end  process ;
end  ALU;
```

– Модель однієї секції ВС1

```
library IEEE;
use IEEE. STD_LOGIC_1164. all;

entity BC1 is
        port (
                PQ0 : inout std_logic ; PF0 : inout std_logic ; PQ3 : inout std_logic ; PF3 : inout std_logic;
                CLK : in std_logic ; OE : in std_logic ; C0 : in std_logic ; I : in std_logic_vector (8 downto 0) ;
                AB : in std_logic_vector (3 downto 0) ;  AA : in std_logic_vector (3 downto 0) ;
                D : in std_logic_vector (3 downto 0) ;  OVR : out bit ; F3 : out bit ;
                Z : out bit ; C4 : out std_logic ;  Y : out std_logic_VECTOR (3 downto 0)
                );
end BC1;

architecture BC1 of BC1 is
-------------------------------------------------------------------------------------------------
component ALU is                                    -- АЛП ВС1
        generic (delay : time : = 40 ns) ;
        port(
                C0 : in std_logic ; R : in bit_VECTOR (3 downto 0) ; S : in bit_VECTOR (3 downto 0) ;
                I : in std_logic_vector (2 downto 0) ; C4 : out std_logic ; OVR : out bit ;
                Z : out bit ; F3 : out bit ; F : out bit_VECTOR (3 downto 0)
          );
end component;
-------------------------------------------------------------------------------------------------
component MXR is                                    -- мультиплексор на вході R АЛП ВС1
        generic ( delay : time : = 10 ns) ;
        port(
                I : in std_logic_vector (2 downto 0) ;  PA : in bit_VECTOR (3 downto 0) ;
                D : in std_logic_vector (3 downto 0) ;  R : out bit_VECTOR (3 downto 0)
);
end component;
-------------------------------------------------------------------------------------------------

                                                    -- Опис сигналів ВС1

signal QD , FC, A, B, PA, PB, R, S, Q, F, CQ, Yout : bit_vector (3 downto 0) ;
begin
        -- регістри РЗП
        P1 : RON port map (CLK => CLK, AA => AA, AB => AB, FC => FC,
        I => I (8 downto 6), A => A, B = > B ) ;
        -- регістри РА та РВ
        P21 : RG_A port map (D = > A, CLK = > CLK, RG_A = > PA) ;
        P22 : RG_B port map (D = >B, CLK = > CLK, RG_B = > PB ) ;
        -- мультиплексори MXR та MXS
        P31 : MXR port map (I = > I (2 downto 0) , PA = > PA , D = > D, R =>R );
        P32 : MXS port map (I => I (2 downto 0 ) , PA => PA , PB => PB, PQ = > Q, S => S);
        -- АЛП
        P4 : ALU port map (C0 = > C0, R = > R, S => S, I => I (5 downto 3),
        C4 = > C4, OVR = > OVR, Z = > Z, F3 = > F3, P = > P, G = > G, F = > F );
        -- мультиплексор MXY
        P5 : MXY port map (I = > I ( 8 downto 6), PA => PA, F => F, Yout = > Yout );
        -- шиний формувач
        P61 : SF port map (OE => OE, Yout => Yout, Y => Y);
        -- зсувачі ЗСВF і ЗСВQ
        P62 : SDVF port map (F => F, I => I (8 downto 6), FC => FC, PF0 => PF0, PF3 => PF3);
        P7 : SDVQ port map (PQ => Q, I => I (8 downto 6), CQ => CQ, PQ0 => PQ0, PQ3 => PQ3);
        -- мультиплексор MXQ
        P8 : MXQ port map (I => I (8 downto 6), CQ => CQ, F => F, QD => QD);
```

```
        -- регістр  PQ
        P9 : PQ  port  map  (QD => QD,   I => I (8 downto 6),  CLK => CLK,  RQ => Q);
end BC1;
```

-- Модель БОД із 7 секцій BC1 (фрагмент)

```
library  IEEE ;
use  IEEE. std_logic_1164. all ;
entity  BOD  is                                          --  інтерфейс  БОД
        port (
                CLK :   in  std_logic;
                D_BC1 : in  std_logic_vector  ( 27 downto 0);
                I :       in  std_logic_vector (8  downto 0);
                AA :    in  std_logic_vector (3 downto 0);
                AB :    in  std_logic_vector (3 downto 0);
                C0 :    in  std_logic;
                OE :    in  std_logic;
                Y_BC : out std_logic_VECTOR  (27 downto 0);
                 OVRnn :  out  std_logic;
                 Z :      out  std_logic
        );
end BOD;
```

```
architecture BOD of BOD is
component BC1 is
        port(
                PQ0 : inout  std_logic;  PF0 :  inout std_logic;  PQ3  : inout std_logic;  PF3 :  inout std_logic;
                CLK : in  std_logic;  OE :  in std_logic;  C0 :  in std_logic;  I : in std_logic_vector (8 downto 0);
                AB :  in  std_logic_vector (3 downto 0);  AA : in std_logic_vector  (3 downto 0);
                D : in  std_logic_vector  (3 downto 0);  OVR :  out  bit;
                F3 : out bit;    Z : out  bit;  C4 :  out  std_logic;
                Y :  out  std_logic_VECTOR (3 downto 0)
                );
end  component ;                                  --   перелік  сигналів  БОД

signal  PF3_0,  PQ3_0,  PQ3_1,  PF3_1,  PF3_2,  PQ3_2,  PF3_3,  PQ3_3,  PQ3_4,  PF3_4,  PF3_5,  PQ3_5  : std_logic;
signal  C4_0,  C4_1,  C4_2,  C4_3,  C4_4,  C4_5,  C4_6,  C4_7,  b3 :  std_logic;
signal  Z_0,  Z_1,  Z_2,  Z_3,  Z_4,  Z_5,  Z_6,  b1,  b2  : bit ;                    -- вивідні  переноси  із  секцій  BC1
signal  PQ0,  PQ3,  PF0,  PF3 :  std_logic;
begin
        -- нульова  секція  BC1
        MPS0 :  BC1  port  map  ( PQ0  =>  PQ0,  PF0 => PF0,  PQ3 => PQ3_0,   PF3 => PF3_0,  CLK => CLK,
                OE =>  OE,  C0  = > C0,  I => I,  AB => AB,  AA => AA,   D => D_BC1 (3 downto 0),
                OVR = >  open,  F3 =>  open,  Z => Z_0,  C4 => C4_0,  Y => Y_BC  (3 downto 0));

        -- перша  секція  BC1
        MPS1:  BC1  port  map ( PQ0 => PQ3_0,  PF0 => PF3_0,  PQ3 => PQ3_1,   PF3 => PF3_1,   CLK = > CLK,
                OE = > OE,  C0 = > C4_0,  I => I,  AB => AB,  AA = > AA, D = > D_BC1(7 downto 4),
                OVR = > open,  F3 = > open,  Z => Z_1,  C4 = >  C4_1,  Y = >  Y_BC  (7 downto 4));
```

```
        -- шоста  секція  BC1
        MPS6 :  BC1  port  map  ( PQ0 = > PQ3_5,  PF0 => PF3_5,  PQ3 => PQ3,   PF3 = > PF3,   CLK = > CLK,
                OE = >  OE, C0  = >  C4_5,  I = > I,  AB = > AB, AA = > AA,  D = > D_BC1 (27 downto 24),
                OVR = > b1, F3 = > b2, Z = > Z_6,  C4 = >  b3,  Y = > Y_BC  (27 downto 24));
        Z < = to_stdulogic  (Z_0  and  Z_1  and  Z_2  and  Z_3  and  Z_4  and  Z_5  and Z_6);
        PQ0 < = '0'  when  I (7) = '0'  else  'Z';       PF0 < = '0'  when  I(7) = '0'  else  'Z';
        PQ3  < = PF0  when  I (7) = '1'  else 'Z';  PF3 < =  to_stdulogic  (b1  xor  b2)  when  I(7) =  '1'  else  'Z';
        OVRnn  < =  to_stdulogic (not  (b2  xor  to_bit (b3) ) );
        end  BOD;
```

-- Модель ФАМ на базі ВУ4 (фрагмент)

```
library IEEE;
use IEEE. STD_LOGIC_1164. all;
use ieee. std_logic_unsigned. all;
use IEEE. std_logic_arith. all;

                                        -- Модель інкрементора INC

library IEEE;
use IEEE. std_logic_1164. all;
use IEEE. std_logic_unsigned. all;
use IEEE. std_logic_arith. all;
entity Incrementor is
        port (
                X : in std_logic_vector ( 11 downto 0);          Y : out std_logic_vector ( 11 downto 0);
                C0 : in std_logic
                );
end Incrementor;
-----------------------------------------------------------------------------------------------------------------------
architecture Incrementor of Incrementor is
procedure Inc_stdlogvect (variable Vect : inout std_logic_vector) is
variable int : integer : = 0;
begin
        int : = conv_integer (Vect);  int : = int + 1;
        Vect : = CONV_STD_LOGIC_VECTOR (int, Vect ' LENGTH);
end;
begin
        process (C0, X)
        variable tempvect : std_logic_vector (11 downto 0);
        begin
                if (C0 = '1') then tempvect : = X; Inc_stdlogvect (tempvect); Y < = tempvect after 25 ns;
                else tempvect : = X; Y < = tempvect after 25 ns;
                end if;
        end process;
end     Incrementor;
-----------------------------------------------------------------------------------------------------------------------

                                        -- Модель регістра PC

library IEEE;
use IEEE. std_logic_1164. all;
use IEEE. std_logic_unsigned. all;
use IEEE. std_logic_arith. all;
entity RGCMK is
        generic (delay: time : = 25 ns);
        port (
                CLK : in std_logic; DY : in std_logic_vector (11 downto 0);
                PC : out std_logic_vector (11 downto 0)
          );
end RGCMK;

architecture RGCMK of RGCMK is
begin
        process (CLK) is
        variable temp : std_logic_vector (11 downto 0);
        begin
                if (Now = 0 ns) then PC < = "000000000000"; end if;
```

```
        , if (CLK' event and CLK = '1' and not (Now = 0 ns) ) then PC < = DY after delay; end if;
          end process;
end RGCMK;

-----------------------------------------------------------------------------------------------------------
                                        -- Модель мультиплексора MX

library IEEE;
use IEEE. STD_LOGIC_1164. all;
entity MX is
        generic (delay : time: = 25 ns);
        port(
                D : in std_logic_vector (11 downto 0); R : in std_logic_vector (11 downto 0);
                F : in std_logic_vector (11 downto 0); M : in std_logic_vector (11 downto 0);
                PLM : in bit_VECTOR (2 downto 0);              -- вхідна шина з ПЛМ
                DY : out std_logic_vector (11 downto 0)        -- вихідна шина
            );
end MX;
-----------------------------------------------------------------------------------------------------------
architecture MX of MX is
begin
        process (PLM, D, F, R, M)
        variable DY_buf : std_logic_vector (11 downto 0);
        begin
                if (PLM = "000") then DY_buf : = D; elsif (PLM = "001") then DY_buf : = R;
                elsif (PLM = "010") then DY_buf : = F; elsif (PLM = "011") then DY_buf : = M;
                else DY_buf : = "000000000000"; end if;
                DY < = DY_buf after delay;
        end process;
end MX;
-----------------------------------------------------------------------------------------------------------

                                        -- Модель ВІС ВУ4 (фрагмент)

ibrary IEEE;
use IEEE.STD_LOGIC_1164.all;

entity BY4 is                                   -- інтерфейс BY4
        port (
        D : in std_logic_vector (11 downto 0); CLK : in std_logic; RLD : in std_logic;
        C0 : in std_logic; nCC : in std_logic; CCE : in std_logic; OE : in std_logic;
        MI : in std_logic_vector (3 downto 0); nPE : out std_logic; nME : out std_logic;
        NVE : out bit; NFL : out bit; Y : out std_logic_vector (11 downto 0)
        );
end BY4;
-----------------------------------------------------------------------------------------------------------
architecture BY4 of BY4 is
component Incrementor is
        port (
                X : in std_logic_vector ( 11 downto 0); Y : out std_logic_vector ( 11 downto 0); C0 : in std_logic
                );
end component;
-----------------------------------------------------------------------------------------------------------
component RGCMK is                              -- регістр PC
        generic (delay : time : = 25 ns);
        port(
                CLK : in std_logic;       DY : in std_logic_vector (11 downto 0);
                PC : out std_logic_vector (11 downto 0)
            );
end component;
-----------------------------------------------------------------------------------------------------------
```

```
component MX is                          -- мультиплексор адреси мікрокоманди (MX)
        generic (delay :  time : = 25  ns);
        port(
                D : in std_logic_vector (11 downto 0);   R : in std_logic_vector (11 downto 0);
                F : in std_logic_vector (11 downto 0);   M : in std_logic_vector(11 downto 0);
                PLM : in  bit_VECTOR (2 downto 0);    DY : out std_logic_vector (11 downto 0)
            );
end component;
-----------------------------------------------------------------------------------------------------------
signal Z, U : bit;
signal CT, PC, DY, ST, Inc_ADR :  std_logic_vector ( 11 downto 0 );
signal S :  bit_vector ( 2 downto 0);

begin
        -- регістр / лічильник  PA / CT
        P1: RGA port map ( CLK = > CLK,  RLD => RLD, U => U,   MI => MI,  D => D,  CT => CT );
        -- детектор  нуля
        P2:  FPN  port map  ( CT = > CT,   Z => Z);
        -- стек
        P3:  STACKBY4  port map ( CLK => CLK,  U => U,  Z => Z, MI  => MI  , PC =>PC  , NFL => open,
             --NFL = > NFL,  ST = > ST );
        -- лічильник мікрокоманд
        P4:  RGCMK  port  map  ( CLK => CLK,   DY => Inc_ADR,  PC => PC );
        --  мультиплексор
        P5:  MX  port  map ( D =>D,  R => CT,  F => ST,  M => PC,  PLM => S,  DY => DY );
        -- ПЛМ
        P6: PLM  port  map ( NCC => NCC,  CCE => CCE,  Z => Z, MI => MI,
            NPE = > NPE,  NME = > NME,  NVE => open,  -- NVE  = > NVE,  PLM => S,  U => U);
        -- шиний  формувач
        P7:  BF  port  map  ( OE = > OE,  DY = > DY,  Y => Y);
        -- інкрементор
        P8:  Incrementor  port  map  (X => DY,  Y => Inc_ADR,  C0 => C0);
end BY4;
-----------------------------------------------------------------------------------------------------------
```

-- Модель усієї схеми МОП (фрагмент)

```
library  IEEE;
use  IEEE .STD_LOGIC_1164. all;
use  MPPPack. all;

entity  Shema  is                                   -- інтерфейс схеми  МОП
        port (
                Pusk_PK  :  in  std_logic;    Stop  :  in  std_logic;    Stop_PK  :  in std_logic
                );
end Shema;
-----------------------------------------------------------------------------------------------------------
architecture  Shema  of  Shema  is                  -- перелік компонентів схеми МОП
component  BOD  is      -- БОД  із  7  секцій  BC1
        port(
                CLK :   in  std_logic;   D_BC1 : in  std_logic_vector  (27 downto 0);
                I :     in  std_logic_vector ( 8 downto 0);
                AA :    in  std_logic_vector(3 downto 0);   AB : in  std_logic_vector  (3 downto 0);
                C0 :    in  std_logic;  OE :  in std_logic;  Y_BC  :  out  std_logic_VECTOR  (27 downto 0);
                        OVRnn :  out   std_logic;  Z :  out std_logic
                );
end  component;
-----------------------------------------------------------------------------------------------------------
component  BY4  is                                  -- мікросхема  BY4
        port(
                D :  in  std_logic_vector (11 downto 0);  CLK :  in  std_logic;  RLD :  in  std_logic;
```

```
            C0 : in  std_logic; nCC :  in std_logic; CCE : in  std_logic;  OE : in  std_logic;
            MI : in  std_logic_vector (3 downto 0); nPE :  out  std_logic; nME :  out  std_logic;
            NVE : out  bit;  NFL :  out  bit;  Y :  out  std_logic_vector (11 downto 0)
            );
end  component;
-----------------------------------------------------------------------------------------------------------
component  MPP  is                              --  мікропрограмна  пам'ять  (МПП)
      port (
            ADDR :  in  std_logic_vector;  MPP_CELL :  out  MPP_TYPE
        );
end  component;
-----------------------------------------------------------------------------------------------------------
component  RMK  is                              --  регістр мікрокоманд (РМК)
      port(
            MK :  in  MPP_Type;  nOE :  in  std_logic;  CLK_RMK :  in  std_logic;
            I86 :  out  std_logic_vector (2 downto 0);  I53 :  out  std_logic_vector (2 downto 0);
            I20 :  out  std_logic_vector (2 downto 0);  AA_RMK :  out  std_logic_vector (3 downto 0);
            AB_RMK :  out  std_logic_vector (3 downto 0);  C0_BC :  out  std_logic;  nOE_BC :  out  std_logic;
            D_BC :  out  std_logic_vector (27 downto 0);  CCE :  out  std_logic;  nOEY :  out  std_logic;
            nRLD :  out  std_logic;  C0_BY :  out  std_logic;  DA_RMK :  out  std_logic_vector (11 downto 0);
            MI_RMK :  out  std_logic_vector (3 downto 0);  STOP_RMK :  out  std_logic_vector (1 downto 0);
            U_DC :  out  std_logic_vector (4 downto 0);  nW_R :  out  std_logic_vector(1 downto 0);
            UMX_CC :  out  std_logic_vector (3 downto 0);  S12 :  out  std_logic_vector (2 downto 0);
            S34 :  out  std_logic_vector (1 downto 0)
            );
end component;
-----------------------------------------------------------------------------------------------------------
component  BS  is                               --  блок  синхронізації  (БС)
      port(
            Work : in  std_logic;  G1  ,G2 :  out  std_logic
            );
end  component;
-----------------------------------------------------------------------------------------------------------
signal  MPP_REC:  MPP_TYPE;  signal  KOP_RK :  std_logic_vector  (7 downto 0);  -- перелік сигналів
signal  RON1_RK :  std_logic_vector  (3 downto 0);  signal  S2_RK :  std_logic_vector  (19 downto 0);
. . . . . .
begin
      -- регістр  команд  (РК)
      URK:  RK  port map (CLK_RK = > G1b, RB => RB_RB, ROP0 => ROP0_OP ,ROP1 => ROP1_OP,
      ROP  => ROP_OP,  U2 => U2,  U3  => U3,  U4 => U4,  U5 => U5, U26 => U26,  U27 => U27,
      KOP  => KOP_RK,  RON1 =>RON1_RK,  S2 => S2_RK,  B2 => B2_RK,  D2 => D2_RK,  X2 => X2_RK,
      RON2 => RON2_RK,  RK30 => RK30_RK);
      -- буферний  регістр  (РБ)
      UBR:  RB  port  map (CLK_RB => G1b,  ROP0 => ROP0_OP,  U3 => U3,  U5 => U5,
      RB => RB_RB,  RB15 => RB15_RB);
      . . . . . .
      -- тригер ознаки переходу (T_JMP)
      UT_JMP:  T_JMP  port  map  (u22 = > u22,  u24 => u24,  Q_TJ => Q_TJ);

      Y_BC1 < = Y_BC (19  DOWNTO 0);  G1b < = not (G1);  --U23 < = U8  or  U9;
      G2b < =  not (G2);
end  Shema;
-----------------------------------------------------------------------------------------------------------
```

Список рекомендованої літератури

1. Методичні вказівки до лабораторного практикуму з курсу "Архітектура комп'ютерів" для студентів спеціальностей "Комп'ютерні системи і мережі" та "Системне програмування" / Укл. Лапко В. В., Губарь Ю. В. - Донецьк: Видавництво ДНТУ, 2005.– 120 с.

2. Методичні вказівки до лабораторного практикуму PC&EWB з курсу "Цифрові ЕОМ" для студентів спеціальностей "Комп'ютерні системи і мережі" та "Системне програмування" / Укл. Лапко В. В., Губарь Ю. В. - Донецьк: Видавництво ДНТУ, 2004.– 78 с.

3. Проектирование цифровых систем на комплектах микропрограммируемых БИС / С.С. Булгаков, В. М. Мещеряков, В.В. Новоселов, Л.А. Шумилов; Под ред. В. Г. Колесникова – М.: Радио и связь, 1984. – 240 с.

4. Жабин В. И. Однокристальные и микропрограммируемые ЭВМ. – К.: Диалектика, 1995. – 116 с.

5. Мик Дж., Брик Дж. Проектирование микропроцессорных устройств с разрядно – модульной организацией – М.: Мир, 1984. – т.1, т.2.

6. Хвощ С.Т., Варлинский Н.Н., Попов Е.А. Микропроцессоры и микроЭВМ в системах автоматического управления – Л.: Машиностроение, 1988. – 640 с.

7. Справочник по устройствам цифровой обработки информации / Н.А. Виноградов, В. Н. Яковлев, В.В. Воскресенский и др. – К.: Техніка, 1988. - 415 с.

8. Каган Б. М. Электронные вычислительные машины и системы. 3 – е изд – е. - М.: Энергоатомиздат, 1991.

9. Путинцев Н. Д. Аппаратный контроль управляющих цифровых вычислительных машин. – М.: Советское радио, 1966.

10. Микропроцессоры и микропроцессорные комплекты интегральных микросхем: Справочник: В 2 – х томах. / Под ред. В. А. Шахнова. – М.: Радио и связь, 1988.

11. Аналоговые и цифровые интегральные микросхемы / Под ред. С. В. Якубовского. - М.: Радио и связь, 1984. – 432 с.

12. Шило В. Л. Популярные цифровые микросхемы: Справочник. - М.: Радио и связь, 1988. – 352 с.

13. Полупроводниковые БИС запоминающих устройств. Справочник / В. В. Баранов, Н. В. Бекин, А. Ю. Гордонов и др.; Под ред. А. Ю. Гордонова и Ю. Н. Дьякова – М.: Радио и связь, 1987. – 360 с.

14. Большие интегральные схемы запоминающих устройств: Справочник / А. Ю . Гордонов, Н. В. Бекин, В. В. Цыркин и др.; Под ред. А. Ю. Гордонова и Ю. Н. Дьякова – М.: Радио и связь, 1990. – 288 с.

15. Полупроводниковые запоминающие устройства и их применение / В. П. Андреев, В.В. Баранов, Н. В. Бекин и др.; Под ред. А. Ю. Гордонова – М.: Радио и связь, 1981. – 344 с.

16. Микропроцессорный комплект К1810: Структура, программирование, применение: Справочная книга / Ю. М. Казаринов, В. Н. Номоканов, Г. С. Подклетнов, В. Ф. Филипов; Под ред. Ю. М. Казаринова – М.: Высшая школа, 1990. – 269 с.

17. Семенец В. В., Хаханова И. В., Хаханов В. И. Проектирование цифровых систем с использованием языка VHDL: Учебное пособие – Харьков: ХНУРЭ, 2003. – 492 с.

18. Суворова Е. А., Шейнин Ю. Е. Проектирование цифровых систем на VHDL – СПб.: БХВ – Петербург, 2003. – 576 с.

19. Сергиенко А. М. VHDL для проектирования вычислительных устройств –К.: ЧП "Корнейчук', ООО "ТИД ДС", 2003. – 208 с.

20. Поляков А. К. Языки VHDL и VERILOG в проектировании цифровой аппаратуры – М.: СОЛОН –Пресс, 2003. – 320 с.

21. Кондратенко Ю П., Сидоренко С. А., Підопригора Д. М. Поведінковий синтез цифрових пристроїв у середовищі Active – HDL: Навчальний посібник / За ред. Ю. П. Кондратенка - Миколаєв: Вид – во МФ НА УКМА, 2002. – 116 с.

22. Бибило П. Н. Синтез логических схем с использованием языка VHDL – М.: - СОЛОН – Р, 2002. – 384 с.

23. Армстронг Дж. Р. Моделирование цифровых систем на языке VHDL: Пер. с англ. – М.: Мир, 1992. – 175 с.

24. Яицков А. С. VHDL - язык описания аппаратных средств: Учебное пособие / Под ред. акад. В. С. Бурцева, акад. Б. С. Митина – М.: Изд – во МАТИ – РГТУ "ЛАТМЭС", 1998. – 119 с.

25. Карлащук В.И. Электронная лаборатория на IBM PC. Программа Electronics Workbench и ее применение. – 2-е изд. – М.: "Солон-Р", 2001. – 726 с.

26. Прикладная теория цифровых автоматов / К.Г. Самофалов, А.М. Романкевич, В.Н. Валуйский и др. – К.: Вища шк. Головное изд – во, 1987. – 375 с.

27. Микросхемы серии К155, КМ155. Краткие технические данные – Северодонецк: НПО "Импульс", 1988.

28. Логические ИС КР1533, КР1554. Справочник. В двух частях / И.И. Петровский, А.В. Прибыльский, А.А. Троян, В.С. Чувелев – М.: ТОО "Бином", 1993.

29. Микросхемы серии К531. Краткие технические данные – Северодонецк: НПО "Импульс", 1981.

30. Майоров С.А., Новиков Г.И. Принципы организации цифровых машин. – Л.: Машиностроение, 1974. – 432 с.

31. Применение интегральных микросхем в электронной вычислительной технике: Справочник / Р.В. Данилов, С.А. Ельцова, Ю.П. Иванов и др.; Под ред. Б.Н. Файзулаева, Б.В. Тарабрина. – М.: Радио и связь, 1987. - 384 с.

32. Усатенко С. Т., Каченюк Т. К., Терехова М. В. Выполнение электрических схем по ЕСКД: Справочник. – М.: Издательство стандартов, 1989. - 325 с.

33. Единая система программной документации. ГОСТ 19.001 – 77. ГОСТ 19.003 – 80. ГОСТ 19.004 – 80. ГОСТ 19.005 – 85. ГОСТ 19.101 – 77. ГОСТ 19.102 – 77. ГОСТ 19.103 – 77. ГОСТ 19.104 – 78. ГОСТ 19.402 - 78. ГОСТ 19.503 – 79. ГОСТ 19.504 – 79.

34. Башков Е. А., Губарь Ю. В. Микропроцессорные системы повышенного быстродействия с микропрограммным управлением. – Донецк: ДПИ, 1983. – 112 с.

35. Методичні вказівки і завдання до лабораторних робіт за курсом "Проектування запам'ятовуючих пристроїв" / Укл. Губарь Ю. В. – Донецьк: ДонДТУ, 2001. – 64 с.

36. Самофалов К. Г., Корнейчук В. И., Тарасенко В. П. Цифровые ЭВМ: Теория и проектирование / Под общ. ред. К. Г. Самофалова - 3 – е изд., перер. и доп. – К.: Выща школа. Головное изд – во, 1989. – 424 с.

37. Угрюмов Е. П. Цифровая схемотехника. – СПб.: БХВ – Петербург, 2004. – 528 с.

38. Карцев М. А., Брик В. А. Вычислительные системы и синхронная арифметика. – М.: Радио и связь, 1981. – 360 с.

39. Карцев М. А. Арифметика цифровых машин. М.: Наука, 1969.

40. Потемкин И. И. Функциональные узлы цифровой автоматики. – М.: Радио и связь, 1985. – 210 с.

41. Каламбеков Б. А. Микропроцессоры и их применение в системах передачи и обработки сигналов. – М.: Радио и связь, 1988. – 415 с.

42. Корнейчук В. И., Тарасенко В. П. Основы компьютерной арифметики. – К.: 2002. 175 с.

43. Уэйкерли Д. Проектирование цифровых устройств. В 2 – х томах. – М.: Постмаркер, 2002. – 544 с. (1 т.); 528 с. (2 т.).

ЗМІСТ