

Зміст	
Вступ.....	2
1 Огляд існуючих рішень.....	3
1.1 Мікропрограмне управління: вступ	3
1.2 Класифікація блоків управління.....	3
1.3 Опис роботи БМУ	5
2 Розробка БМУ та операційного пристрою.....	8
2.1 Опис математичної складової арифметико-логічного пристрою	8
2.2 Операційна схема	8
2.3 Змістовний мікроалгоритм.....	9
2.4 Функціональна схема арифметико-логічного пристрою	10
2.5 Закодований мікроалгоритм	10
2.6 Розрахунок параметрів БМУ.....	11
3 Набір, відлагодження та симуляція роботи розроблюваного пристрою.....	16
3.1 Загальні відомості.....	16
3.2 Створення проекту	16
3.3 Набір та відлагодження схем БМУ та арифметико-логічного пристрою	22
3.4 Набір та відлагодження розробленого пристрою в цілому	37
3.5 Опис роботи пристрою	40
3.6 Помилки та їх виправлення.....	48
4 Тестування розробленого пристрою на апаратному відлагоджувальному комплексі.....	49
4.1 Розмітка пінів	49
4.2 Підключення відлагоджувального комплексу	54
4.3 Програмування ПЛІС	55
Висновки.....	57
Список скорочень.....	58
Список літератури:.....	59

ВСТУП

У лабораторній роботі виконується розробка пристрою з мікропрограмним управлінням для обчислення заданої функції.

У розділі 1 приведено огляд існуючих рішень БМУ. Приводиться класифікація та опис різних видів БМУ. Розділ 2 присвячений розробці функціональних схем БМУ та арифметико-логічного пристрою для обчислення функції. Приводяться розрахунки параметрів БМУ, кодування операцій та керуючих сигналів, функціональний алгоритм обчислення, розміщення команд у ПМК та карта програмування БМУ. У розділі 3 описано набір та відлагодження схем БМУ та арифметико-логічного пристрою у програмному моделюючому комплексі Quartus 2. Розглядається створення проекту, створення функціональних блоків (реєстри, лічильник суматори) та загальні принципи створення схеми БМУ. Розділ 4 присвячений тестуванню розробленого пристрою на апаратному комплексі DE2 фірми Altera.

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Мікропрограмне управління: вступ

У більшості сучасних ЕОМ використовується принцип мікропрограмного управління, що отримав розповсюдження в обчислювальній техніці починаючи із 60-х років. Мікропрограмне управління дозволило значно розширити систему команд ЕОМ, так як це досягалося, в основному, за рахунок збільшення об'єму порівняно дешевої пам'яті блоку мікропрограм. У систему команд більшості машин були додані різноманітні варіанти інструкцій обробки даних (арифметичних і логічних команд), керування і т.д. Розширення набору команд пояснюється тим, що велика кількість команд машини дозволяв, при його ефективному використанні, скоротити число команд у програмі і цим самим підвищити швидкодію програми. Надалі у багатьох машинах почали розширювати набір команд все складнішими й складнішими командами, намагаючись наблизити машинну мову до рівня мов програмування.

Формат команд у більшості сучасних машин - двухадресний, з різними варіантами адресації (регістр-регістр, регістр-пам'ять, пам'ять-пам'ять) та способами обчислення виконавчої адреси. Це призвело до використання змінної довжини команд (від 2 байтів до 6 і більше).

Вперше усі ці особливості системи команд були реалізовані у архітектурі сімейства IBM360, потім вони перейшли у міні-ЕОМ та мікропроцесори. Але з розвитком мікроелектронних технологій та ускладнення мікропроцесорів, співвідношення між вартістю мікропрограм і обладнання стало змінюватися: вартість блоку пам'яті мікропрограм і блоку схем управління, а також площа, що займається ними на кристалі, мало відрізняються один від одного, а необхідність розміщення пам'яті мікропрограм та іншого обладнання на одному й тому ж кристалі ускладнювали реалізацію мікропрограм великого розміру.

1.2 Класифікація блоків управління

У мікропроцесорах використовуються два методи видачі сукупності управляючих сигналів: програмний та мікропрограмний.

Виконання операцій в машині зводиться до елементарних претворень інформації (передача інформації між вузлами у блоках, зсув інформації у вузлах, логічні порозрядні операції, перевірка умов тощо) у логічних елементах, вузлах та блоках під дією функціональних керуючих сигналів блоків (пристроїв) керування.

Елементарні перетворення, що не можуть бути розкладені на більш прості, виконуються на протязі одного такту сигналів синхронізації і називаються мікроопераціями.

У апаратних (схемних) пристроях управління кожній операції відповідає свій набір логічних схем, що виробляють певні функціональні сигнали для виконання мікрооперацій у певні моменти часу. При такому способі побудови пристрою управління реалізація мікрооперацій досягається за рахунок жорстко з'єднаних між собою логічних схем, тому ЕОМ з апаратним пристроєм управління називається ЕОМ з жорсткою логікою управління. Це поняття

відноситься до фіксації системи команд у структурі зв'язків у ЕОМ і визначає практичну неможливість будь-яких змін у системі команд ЕОМ після її виготовлення.

При мікропрограмній реалізації пристрою управління у склад останнього вводяться ЗП, кожний розряд вихідного коду якого визначає появу певного функціонального сигналу управління. Тому кожній мікрооперації ставиться у відповідність свій інформаційний код - мікрокоманда. Набір мікрокоманд і послідовність їх реалізації забезпечують

виконання будь-якої складної операції. Набір мікрооперацій називають мікропрограмами. Спосіб управління операціями шляхом послідовного зчитування та інтерпретації мікрокоманд з ЗП (найчастіше роль мікропрограминого ЗП виконують швидкодіючі програмовані логічні матриці), а також використання кодів мікрокоманд для генерації функціональних управляючих сигналів називають мікропрограмним, а мікроЕОМ з таким способом управління - мікропрограмними або з логікою управління, що зберігається (гнучкою логікою).

До мікропрограм ставиться ряд вимог щодо функціональної повноти та мінімальності. Перша вимога потрібна для забезпечення можливості розробки мікропрограм будь-яких машинних операцій, а друге пов'язане з бажанням зменшити об'єм потрібного обладнання. Врахування фактора швидкодії приводить до розширення мікропрограм, так як ускладнення останніх дозволяє скоротити час виконання команд програми.

Перетворення інформації виконується в універсальному арифметико-логічному блоці мікропроцесора. Він зазвичай будується на основі комбінаційних логічних схем.

Для прискорення виконання певних операцій вводяться додаткові спеціальні операційні вузли (наприклад, циклічні зсувачі). Крім цього у склад мікропроцесорної системи (МПС) БІС вводяться спеціалізовані оперативні блоки арифметичних розширювачів.

Операційні можливості мікропроцесора можна розширити за рахунок збільшення числа регістрів. Якщо у регістровому буфері закріплення функцій регістрів відсутнє, то їх можна використовувати як для зберігання даних, так і для зберігання адрес.

Подібні регістри мікропроцесора називаються регістрами загального призначення (РЗП). З розвитком технологій реально реалізується виготовлення у мікропроцесорі 16, 32 і більше регістрів.

В цілому ж принцип мікропрограмного управління (ПМУ) включає в себе наступні пункти:

- 1) будь-яка операція, що реалізується пристроєм, є послідовністю елементарних дій - мікрооперацій;
- 2) для керування порядком, за яким будуть виконуватися мікрооперації, використовують логічні умови;
- 3) процес виконання операцій у пристрої описується у формі алгоритму, що представляється у термінах мікрооперацій і логічних умов, і який називається мікропрограмою;

4) мікропрограма використовується як форма представлення функції пристрою, на основі якої визначається структура та порядок функціонування пристрою у часі.

ПМУ надає гнучкості мікропроцесорній системі і дозволяє забезпечити проблемну орієнтацію мікро- та мініЕОМ.

Крім класифікації блоків управління за функціональними ознаками (програмні та мікропрограмні) існують інші:

- централізовані та децентралізовані. У централізованих БУ мікропрограми формуються в одному пристрої для всіх пристроїв у системі. Такі БУ забезпечують виконання усіх мікрооперацій послідовно у часі, що призводить до падіння швидкості системи. У децентралізованих БУ кожен пристрій має свій БУ, роботу яких синхронізує централізований пристрій управління. У різних пристроях можливе виконання мікрооперацій одночасно, що призводить збільшення швидкодії, але збільшує апаратні витрати. У сучасних ЕОМ більш поширені децентралізовані БУ;

- синхронні та асинхронні. У синхронних БУ для виконання кожної мікрооперації виділяються однакові проміжки часу, що дорівнюють максимальній тривалості МО. У асинхронних БУ на кожен МО виділяють стільки часу, скільки вона потребує для виконання. Асинхронні БУ є більш швидкодіючими порівняно із синхронними, але потребують збільшення апаратної складності. На практиці застосовують комбіновані БУ, у яких МО групуються за часом виконання.

1.3 Опис роботи БМУ

Можна виділити наступні етапи виконання команди в обчислювальній системі:

1. Вибірка команди. З ОП зчитується команда в регістр команд процесора, для чого виконується відповідна МП, що записана у пам'ять БМУ.

2. Розпакування команди. Команда розшифровується (аналізуються поля слова команди, визначаються операнди), що забезпечується виконанням відповідної МП.

3. Виконання операції. Виконується МП виконання заданої операції над визначеними операндами.

4. Формування адреси наступної команди. Відповідна МП формує адресу наступної команди у лічильнику команд.

Спрощена структурна схема БМУ наведена на рис. 1.1.

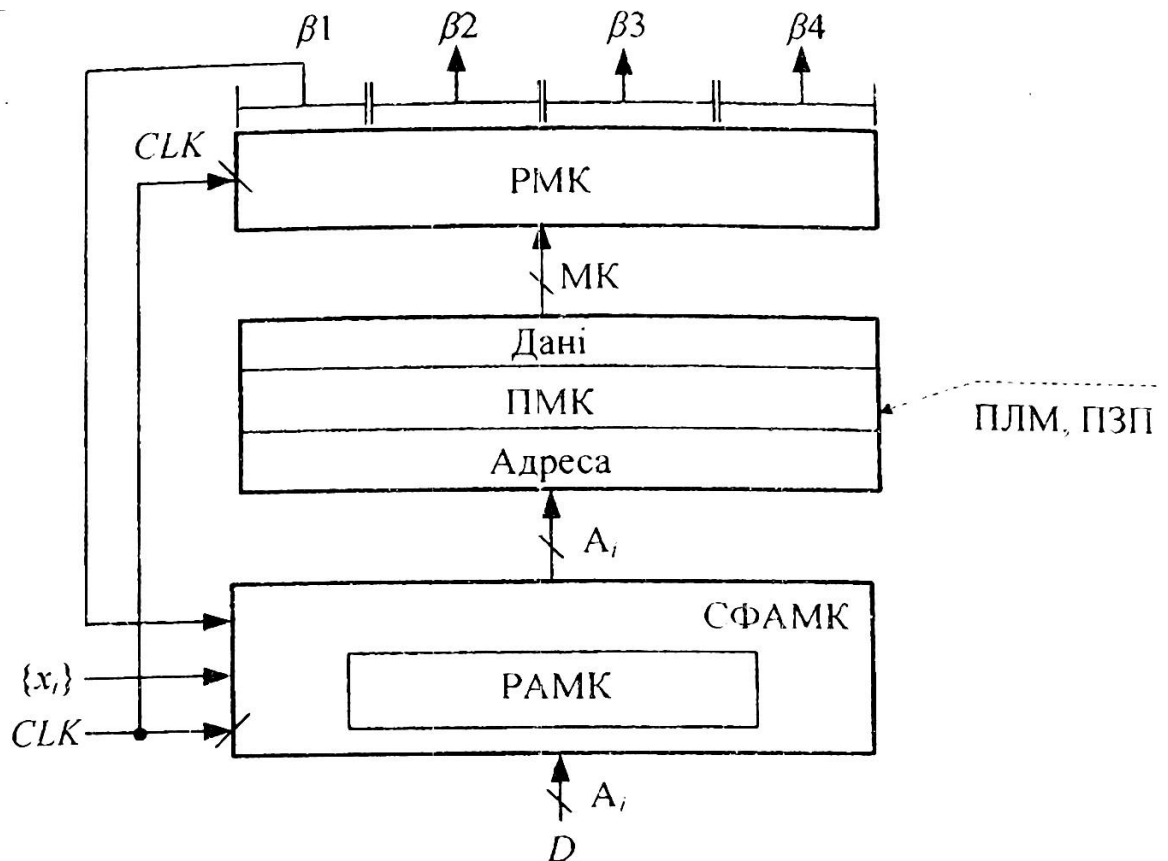


Рисунок 1.1 - Структурна схема БМУ

Основні функціональні частини БМУ:

- РАМК - регістр адреси МК;
- СФАМК - схема формування адреси МК;
- ПМК - пам'ять МК;
- РМК - регістр МК;
- A_i - адреса МК;
- CLK - синхросигнал;
- $\{x_i\}$ - логічні умови;
- D - вхід завдання початкової адреси мікропрограми.

МК розміщується у пам'яті мікропрограми. На рис.1.2 наведений формат мікрокоманди.

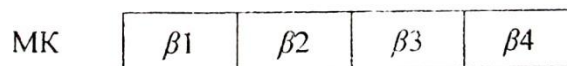


Рисунок 1.2 - Формат мікрокоманди

Основним призначенням СФАМК є реалізація управляючих структур, що зустрічаються у мікропрограмах: лінійна послідовність, структури виду «якщо А, то Б, інакше В» та структури виду «поки А, роби Б». При цьому схема виконує наступні функції:

- проводить дешифрацію коду операції команди (КОП) для звернення до першої мікрокоманди мікропрограми, що інтерпретує дону команду;
- формує адреси наступних мікрокоманд;

- зберігає ознаки переходів, до виходять з операційного блоку то створюються при виконанні мікрокоманд умовного переходу;

- виконує управління перериваннями на мікропрограмному рівні.

Пам'ять мікропрограм призначена для зберігання мікрокоманд, її ємність та розрядність однозначно визначаються набором мікропрограм, що реалізуються. Шляхом зміни набору мікропрограм можна змінювати систему команд мікропроцесора і тим самим орієнтувати його функціональну спрямованість.

У кожному такті за синхросигналом CLK адреса мікрокоманди поновлюється у РАМК і надходить на адресний вхід ПМК. За адресою, що надійшла у ПМК, обирається відповідна мікрокоманда і видається на вихід даних ПМК. Слово мікрокоманди записується у РМК за зворотнім перепадом синхросигналу CLK.

Сигнали зони β_2 управляють вузлами МПС, зони β_3 - визначають тривалість цих сигналів, сигнали зони β_1 разом із логічними умовами $\{x_i\}$ поступають на вхід СФАМК і формують адресу наступної МК. За черговим сигналом CLK адреса наступної МК буде сформована у РАМК. Зона β_4 використовується для виконання допоміжних функцій, наприклад, контролю апаратури.

2 Розробка БМУ та операційного пристрою

2.1 Опис математичної складової арифметико-логічного пристрою

Виходячи з ТЗ, ділення виконується першим способом: із зсувом залишку. Під час ділення здійснюється зсув вліво залишку при нерухомому дільнику. Зсув залишку вліво рівноцінний його множенню. В залежності від його старшого розряду вибирається наступний розряд результату (проінвертований). Множення на 4 здійснюється за допомогою двох послідовних зсувів регістру із значенням Y . Спочатку виконується ділення $Z = \frac{1}{X}$, а потім – підсумовування результату з $4Y$. Приймається наступний формат операндів: дільник (1) представлений в вигляді цілого числа (00000001), ділене також має вигляд цілого числа; результат ділення – дробове число; операнд Y – також дробове число, отже результат ділення та $4Y$ можна підсумовувати без нормалізацій та приведення до одного виду. Результат виконання функції пристроєм також представлений дробовим числом.

2.2 Операційна схема

Операційна схема арифметико-логічного пристрою для обчислення функції зображена на рисунку 2.1. Видно, що схема базується на операційній схемі ділення двох чисел, але відрізняється можливістю подальшого використання результату ділення. Призначення регістрів: RZ – результат обчислення $Z = \frac{1}{X}$, R0 – ділене (1), значення Y та кінцевий результат обчислення функції, RX – дільник.

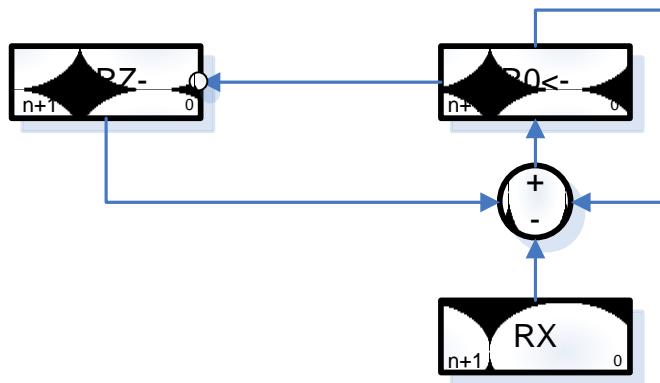


Рисунок 2.1–Операційна схема пристрою для обчислення функції

2.3 Змістовний мікроалгоритм

Змістовний алгоритм складається з алгоритму ділення двох чисел (верхня частина) та операція завантаження, подвійного зсуву та додавання до результату ділення операнда Y. Змістовний мікроалгоритм зображено на рисунку 2.2.

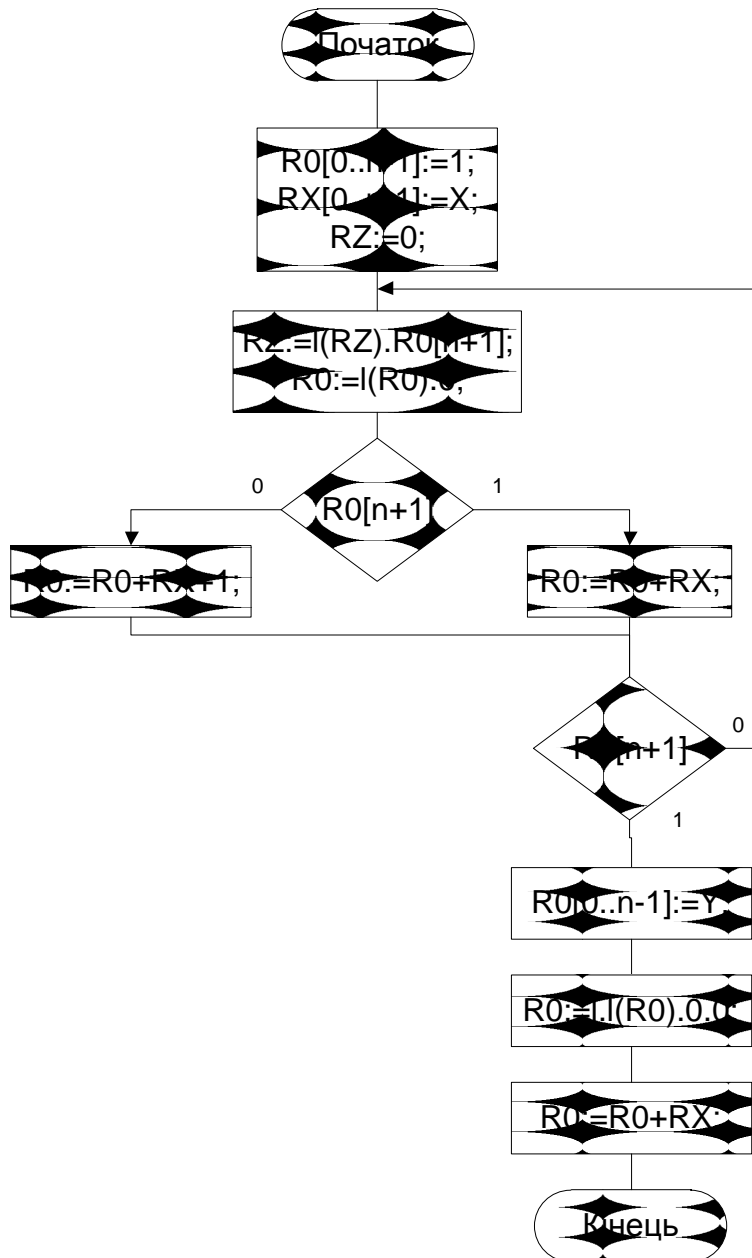


Рисунок 2.2—Змістовний мікроалгоритм арифметико-логічного пристрою для обчислення функції

2.4 Функціональна схема арифметико-логічного пристрою

В порівнянні із пристроєм ділення відбулися наступні зміни: додано мультиплексор на виході регістра RX , що дозволяє використовувати результат ділення для подальших обчислень та розширення кількості входів мультиплексора на виході суматора, що необхідно для занесення у регістр $R0$ значення операнда Y . Функціональна схема зображена на рисунку 2.3.

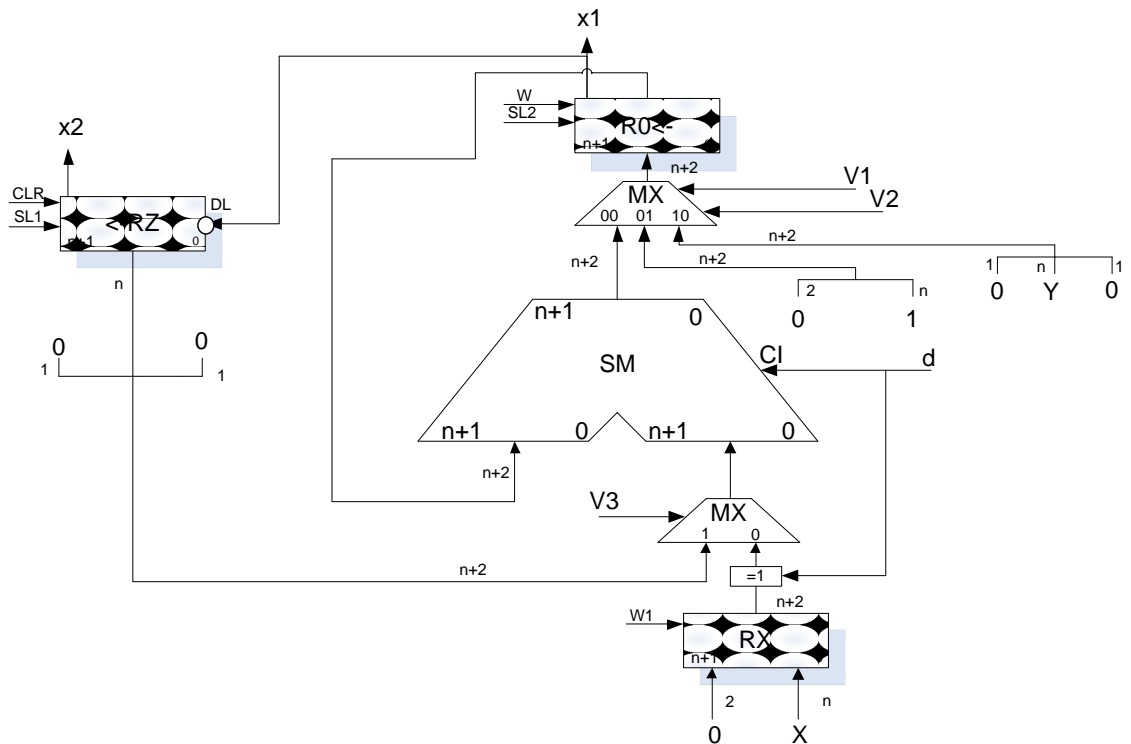


Рисунок 2.3—Функціональна схема арифметико-логічного пристрою для обчислення функції

2.5 Закодований мікроалгоритм

Закодовані мікрооперації та сигнали приведені, відповідно у таблицях 2.1 та 2.2.

Таблиця 2.1–Таблиця кодування мікрооперацій

Мікрооперації	Управляючі сигнали
CLR, W1, V2	y1
SL1	y2
SL2	y3
d	y4
W	y5
V1	y6
V3	y7

Таблиця 2.2–Таблиця кодування сигналів

Логічні умови	Позначення
R0[n+1]	x1
RZ[n+1]	x2

Закодований мікроалгоритм приведений на рисунку 2.4. Додаткові вершини α та β додані для того, щоб не виникало поєднань послідовно слідуючих сигналів у один.

2.6 Розрахунок параметрів БМУ

Складемо таблицю кодування розрядів поля управління мультиплексором (таблиця 2.3).

Таблиця 2.3–Таблиця кодування розрядів поля управління мультиплексором

$m_2 m_1$	УС
00	0
01	x1
10	not x2
11	1

Для визначення максимального приросту адреси розмістимо мікрокоманди у ПМК (рис. 2.5).

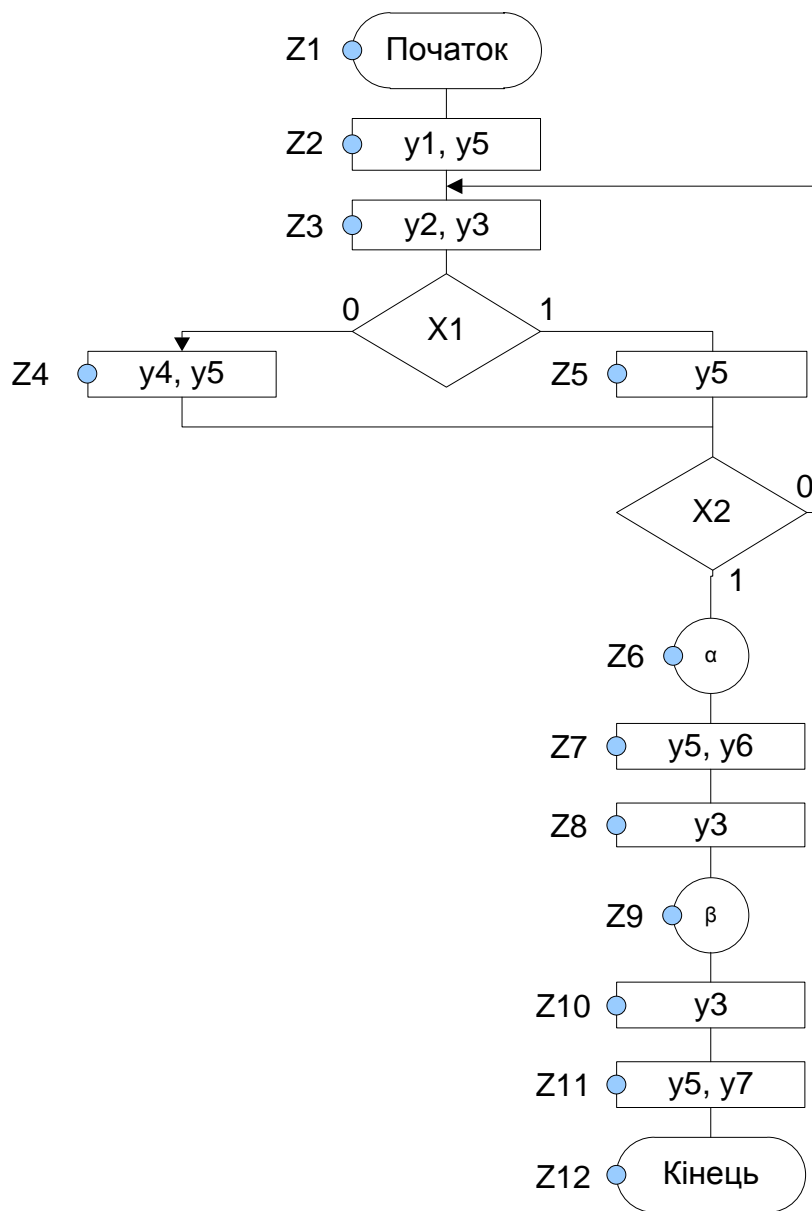


Рисунок 2.4– Закодований мікроалгоритм пристрою для обчислення функції

З рисунку 2.5 видно, що максимальний приріст адреси становить $N = 4$. Визначимо параметри зони β :

$$\begin{aligned}
 n_M &= \lceil \log_2 4 \rceil = 2; \\
 n_S &= \lceil \log_2 N \rceil + 1 = \lceil \log_2 4 \rceil + 1 = 3; \\
 n_{\beta 1} &= 5;
 \end{aligned}$$

Визначимо розрядність адреси ПМК:

$$n_a = \lceil \log_2 64 \rceil = 6;$$

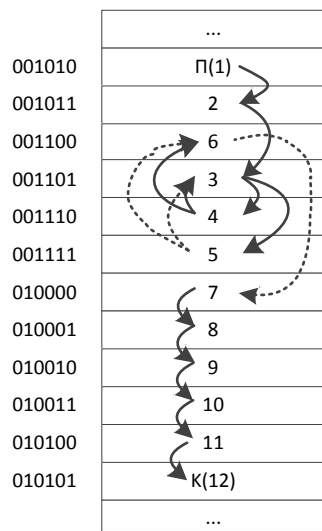


Рисунок 2.5–Розміщення команд у ПМК

Для реалізації такого розміщення команд за допомогою БМУ з відносною адресацією потрібно при $m_2m_1=01$ (таблиця 2.3) подавати не x_2 , а $\text{not } x_2$ (при $x_2=1$ приріст адреси буде меншим, чим при $x_2=0$, отже, до адреси потрібно додавати якусь константу та проінвертоване значення x_2). Можна поміняти місцями команди 3 та 6, тоді не потрібно буде інвертувати x_2 на вході мультиплексора.

За технічним завданням, тривалість мікрооперації підсумовування становить 4 од. Тривалість усіх інших мікрооперацій вважатимемо рівною 1. Отже, довжина зони β_3 становить:

$$n_{\beta_3} = \lceil \log_2 4 \rceil + 1 = 3;$$

При мінімальному кодуванні управляючих сигналів та горизонтальному програмуванню довжина зони β_2 дорівнює кількості управляючих сигналів:

$$n_{\beta_2} = 7;$$

Враховуючи попередні обчислення отримаємо довжину команди:

$$n_{MK} = 5 + 7 + 3 + 1 = 16;$$

Карта програмування БМУ зображена у таблиці 2.4, структурна схема БМУ зображена на рисунку 2.6.

Таблиця 2.4–Карта програмування БМУ

[illegible]

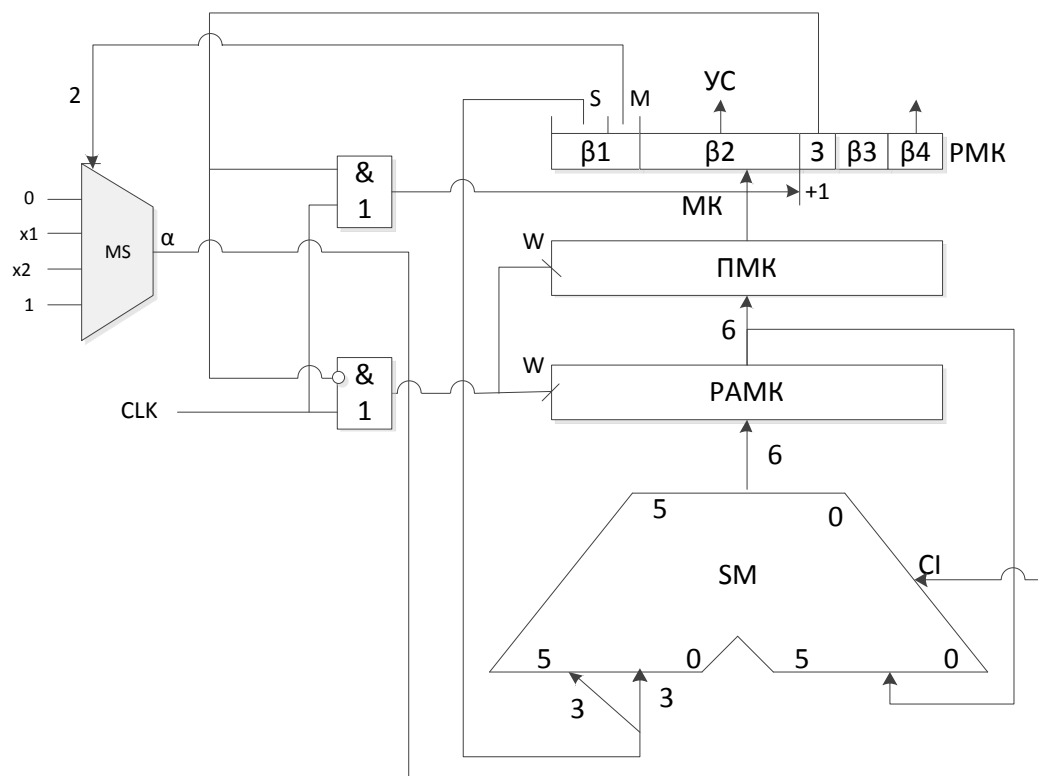


Рисунок 2.5–Структурна схема БМУ

3 Набір, відлагодження та симуляція роботи розроблюваного пристрою

3.1 Загальні відомості

Для набору, відлагодження та симуляції роботи створюваного пристрою використовувалася САПР Quartus || Version 9.1 Build 222 10/21/2009 SJ Full Version.

3.2 Створення проекту

Для створення нового проекту потрібно запустити програму Quartus|| та перейти File->New Project Wizard (рисунок 3.1)

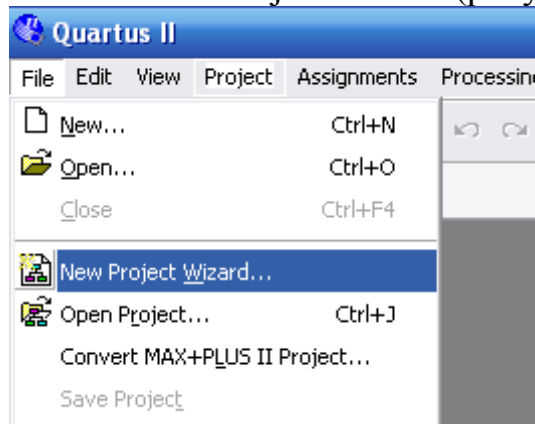
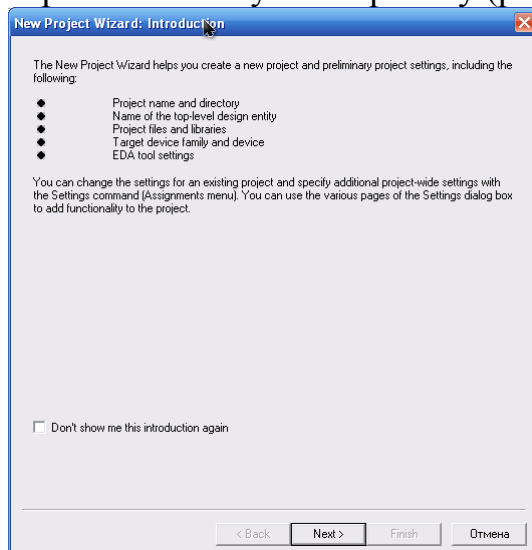
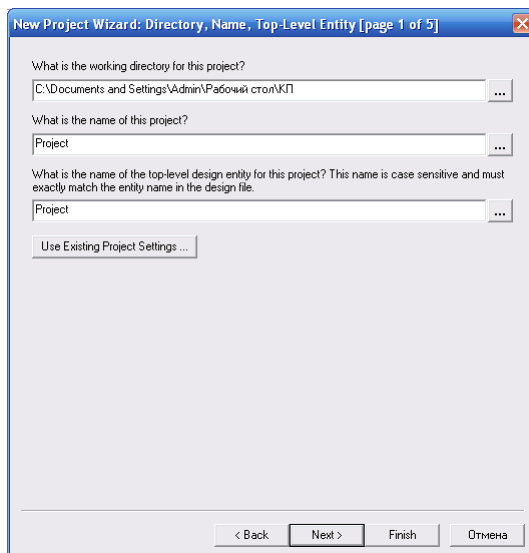


Рисунок 3.1-Запуск програми створення нового проекту New Project Wizard складається з 6 пунктів:

- перелік налаштувань проекту (рисунок 3.2);



- Рисунок 3.2-Перелік налаштувань проекту
- форма введення директорії розташування проекту, ім'я проекту та ім'я за замовчуванням для файлу симуляції (рисунок 3.3);



- Рисунок 3.3-Форма введення імен директорії та проекту
- форма для додавання до проекту проектних файлів (рисунок 3.4), залишаємо порожньою;

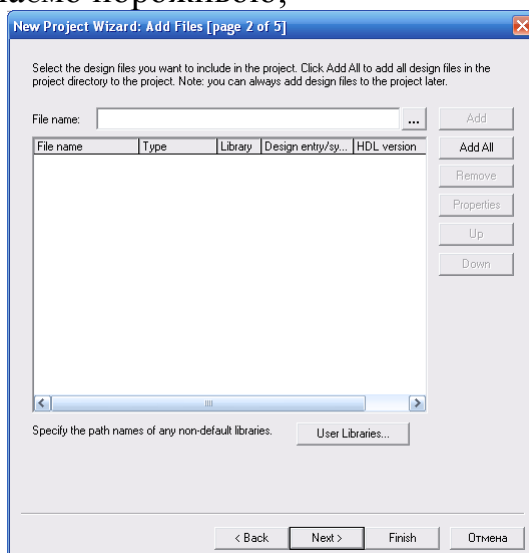


Рисунок 3.4-Форма додавання проектних файлів

- форма для вибору ПЛІС, у яку буде завантажуватися проект, що розробляється. Необхідно вибрати сімейство пристроїв CycloneII, а з доступних девайсів – EP2C35F672C6, який і є тією FPGA, що використовується на апаратному моделюючому пристрої фірми Altera (рисунок 3.5);
- форма для вибору програм синтезу, симуляції та аналізу проекту, залишаємо порожньою (рисунок 3.6);

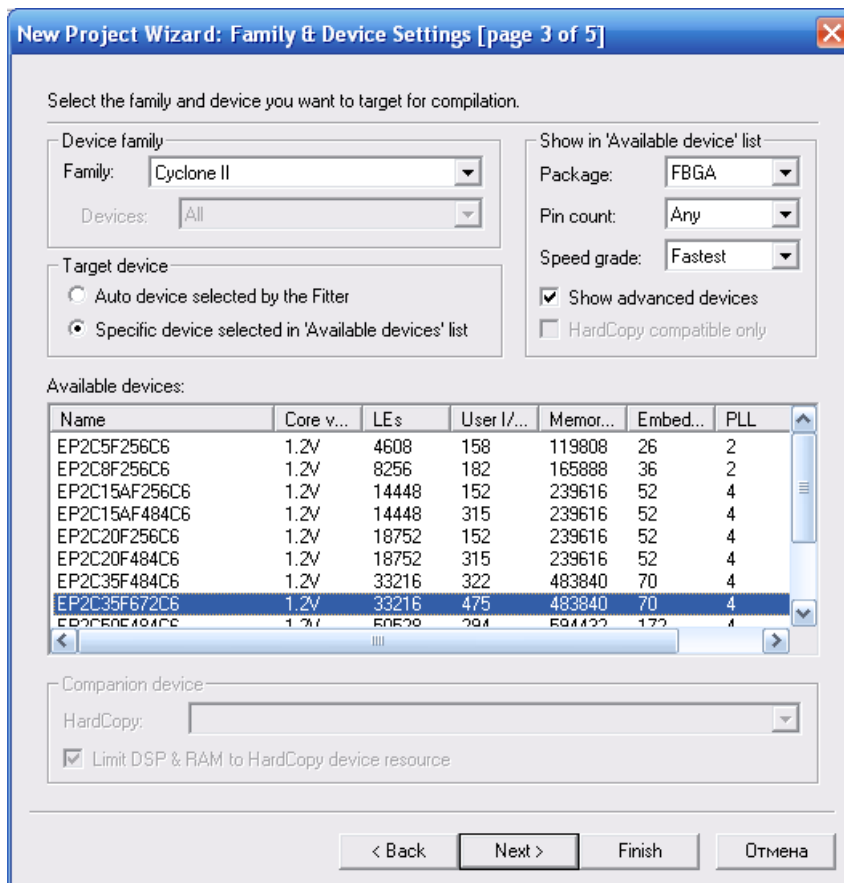


Рисунок 3.5-Форма вибору ПЛІС

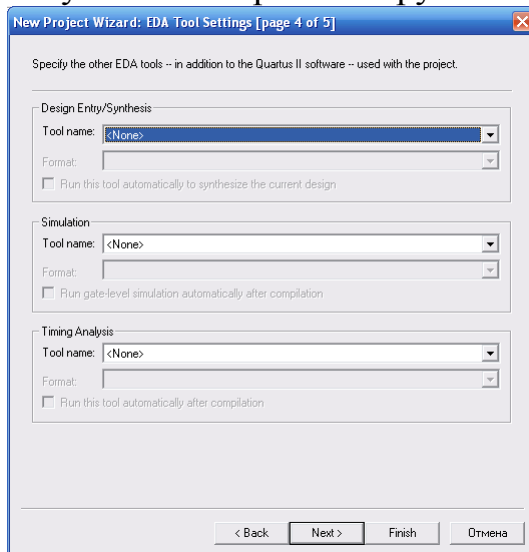


Рисунок 3.6-Форма вибору програм синтезу, симуляції та аналізу проекту

- підсумки: виводяться для перегляду усі налаштування, що були вибрані (рисунок 3.7).

Після натискання кнопки Finish проект буде створено і він з'явиться у панелі Project Navigator (рисунок 3.8).

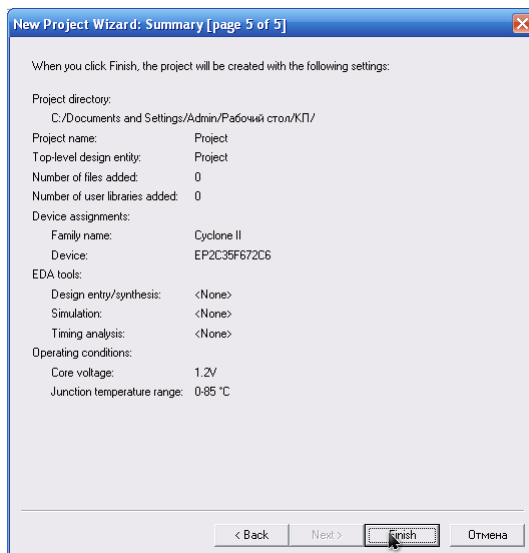


Рисунок 3.7-Форма підсумків

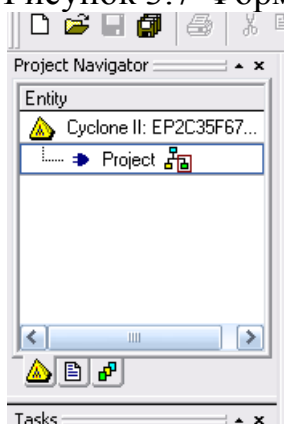


Рисунок 3.8-Створений проект

Тепер необхідно створити схемний файл, верхній у ієрархії. Для цього виконаємо File->New, у з'явившомуся вікні (рисунок 3.9) виберемо пункт Block Diagram/Schematic File та натиснемо ОК. Буде створено новий схемний файл з ім'ям Block1 та розширенням bdf (рисунок 3.10). Для зберігання схемного файлу у нього потрібно додати якийсь логічний елемент. Для цього потрібно двічі натиснути лівою клав'яшею миші на вільному місці наборного поля. Відкриється вікно вставки елементів (рисунок 3.11).

Усі стандартні елементи розташовані у бібліотеці c:/altera/91/quartus/libraries/ та поділяються на наступні класи:

- мегафункції – елементи, параметри яких (кількість входів-виходів, додаткова функціональність) можуть настраюватися;
- примітиви – елементарні логічні елементи – піни вводу та виводу, І, НЕ, АБО, суматори слів, мультиплексори, шифратори тощо, їхня функціональність жорстко задана та не може бути змінена;
- інші логічні елементи – до них входять реалізації пристроїв, що випускаються у світі.

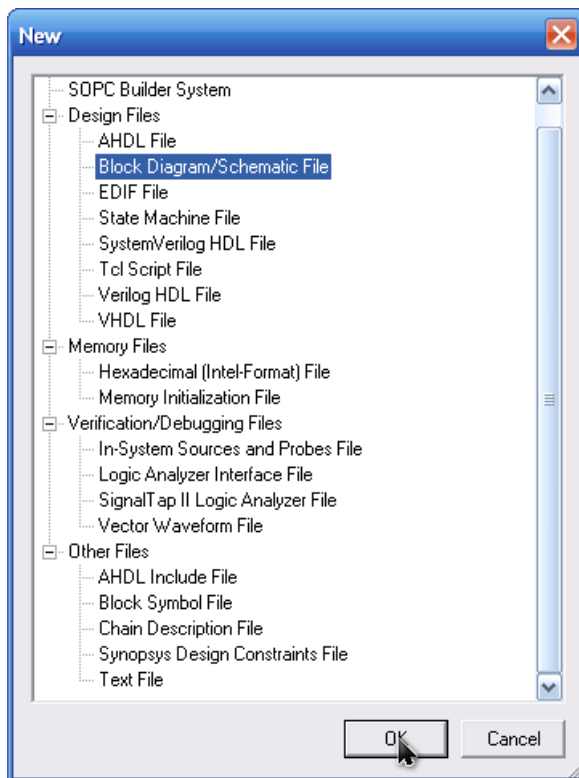


Рисунок 3.9-Вікно для вибору типу створюваного файлу

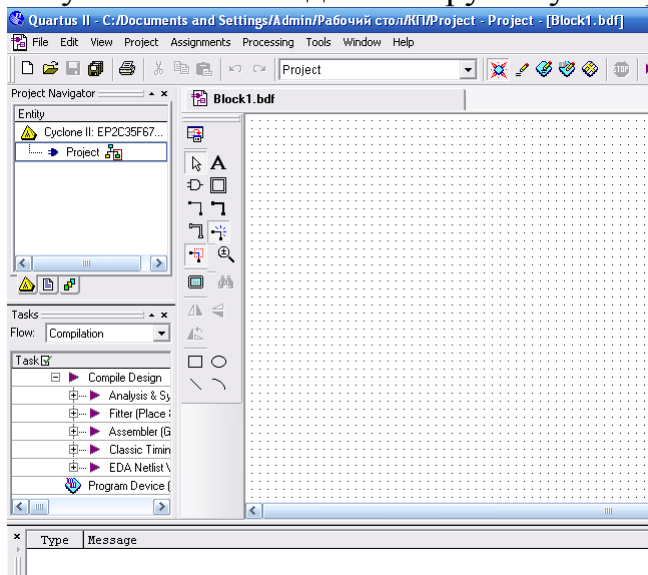


Рисунок 3.10-Вигляд нового схемного файлу

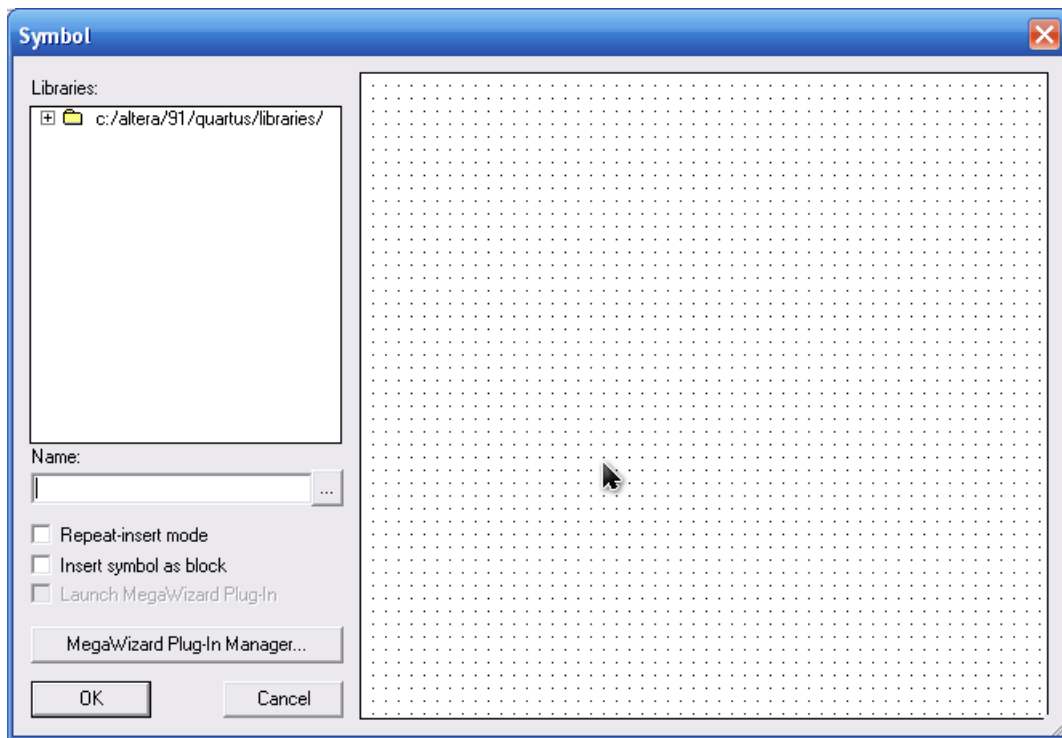


Рисунок 3.11-Вікно вставки елементів

Виберемо primitives->pin->input (це можна зробити, ввівши у полі Name “input”), при цьому елемент, що був вибраний, буде відображений на полі справа (рисунок 3.12).

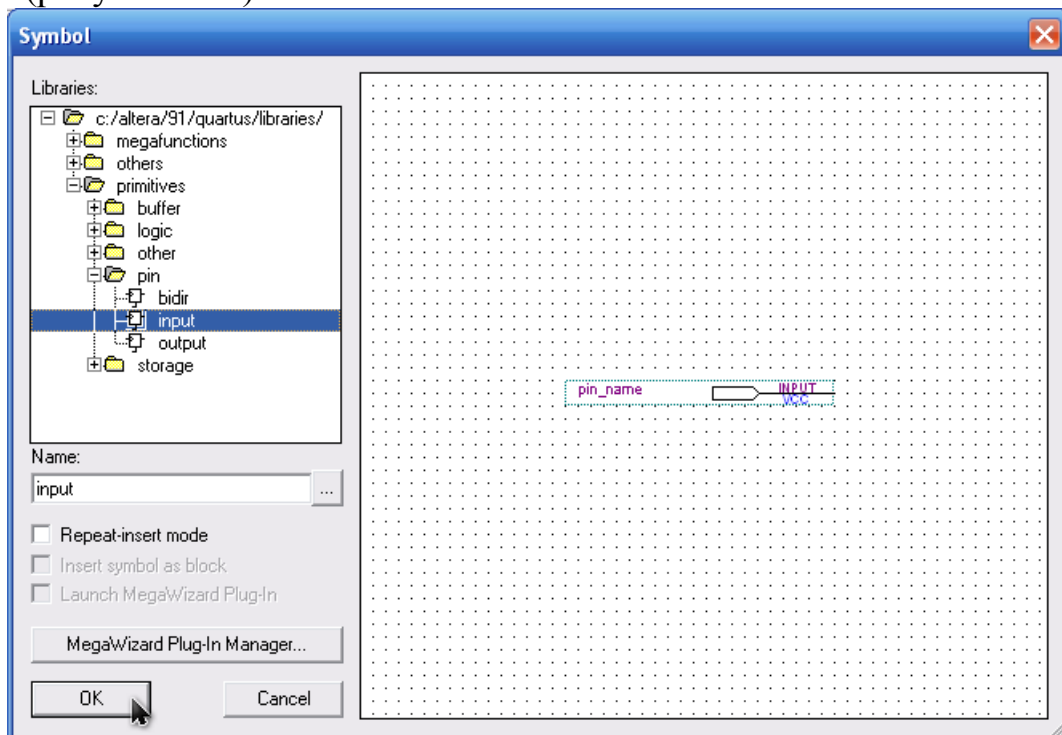


Рисунок 3.12-Вибір елементу

Після натискання ОК, вікно вибору елементу закриється, а біля вказівника миші з'явиться контур вибраного елементу (рисунок 3.13), закріпити який на наборному полі можна кліком лівої клавіші миші.

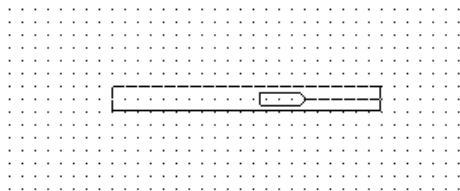


Рисунок 3.14-Вибір місцезнаходження елемента

Двічі натиснувши на встановлений елемент, можна отримати доступ до налаштувань елемента, у якому можна ввести (в даному випадку, для вхідного піна) назву шини або лінії, яку він обслуговує. Так само можна додавати вихідні та двонаправлені піни, тригери, прості логічні елементи.

Після цього можна зберегти схемний файл. За замовчуванням йому присвоїться ім'я "Project".

3.3 Набір та відлагодження схем БМУ та арифметико-логічного пристрою

Для набору схеми БМУ створимо новий схемний файл BMU.bdf. Відкриємо цей файл та виконаємо Project->Set as Top-Level Entity. Тепер файл БМУ буде розглядатися як головний і його можна компілювати та аналізувати окремо від файлу Project. Аналогічно створюється та компілюється схемний файл ALU.bdf для схеми арифметико-логічного пристрою.

Розглянемо створення необхідних елементів за допомогою мегафункцій.

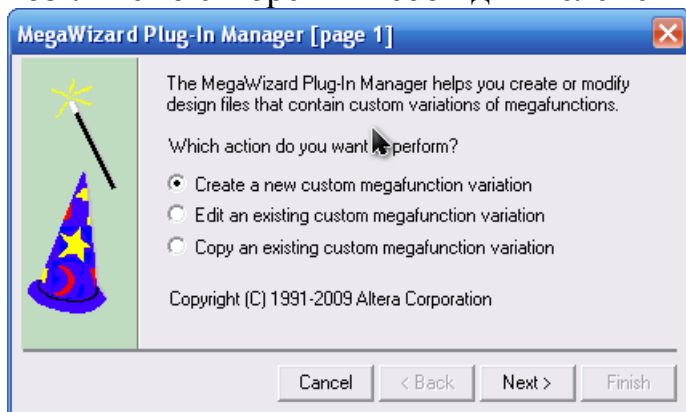


Рисунок 3.15-Вікно вибору джерела створення елемента

Для того, щоб створити елемент з мегафункції, необхідно викликати вікно вибору елемента та натиснути на кнопку MegaWizard Plug-In Manager. Запуститься програма конфігурування мегафункцій (рисунок 3.15). Виберемо створення нового елемента з мегафункції. У наступному вікні потрібно вибрати, яку мегафункцію потрібно використовувати, та як називатиметься створюваний елемент (рисунок 3.16).

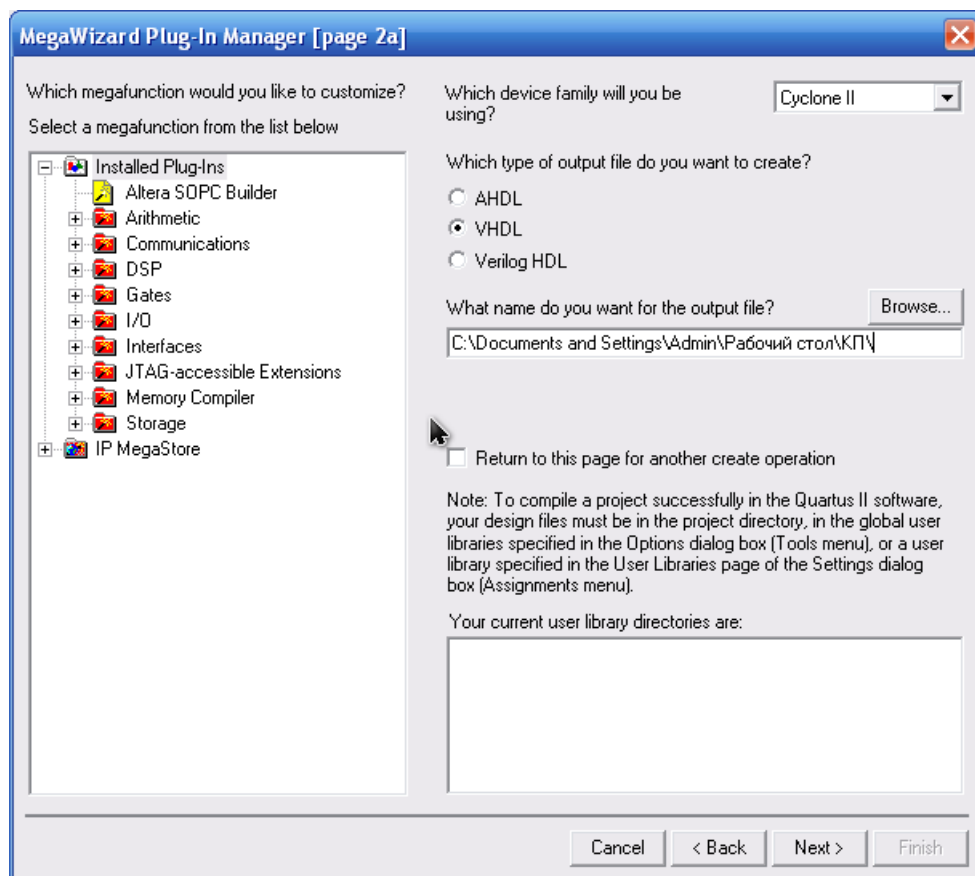


Рисунок 3.16-Вибір мегафункції-предка та імені створюваного елемента
Відкриється діалог конфігурування мегафункції (рисунок 3.17). Для кожної мегафункції індивідуальний діалог. Розглянемо створення необхідних ним елементів.

На попередніх двох рисунках зображено початок створення суматора для БМУ. Потрібно вибрати розрядність суматора (6 бітів). Можна створити суматор та віднімач одному блоці, для цього необхідно вибрати потрібний пункт (рисунок 3.18). При зміні параметрів майбутнього елемента, його вигляд буде змінюватися на полі з лівого боку, це дає уявлення про призначення та використання деяких функцій.

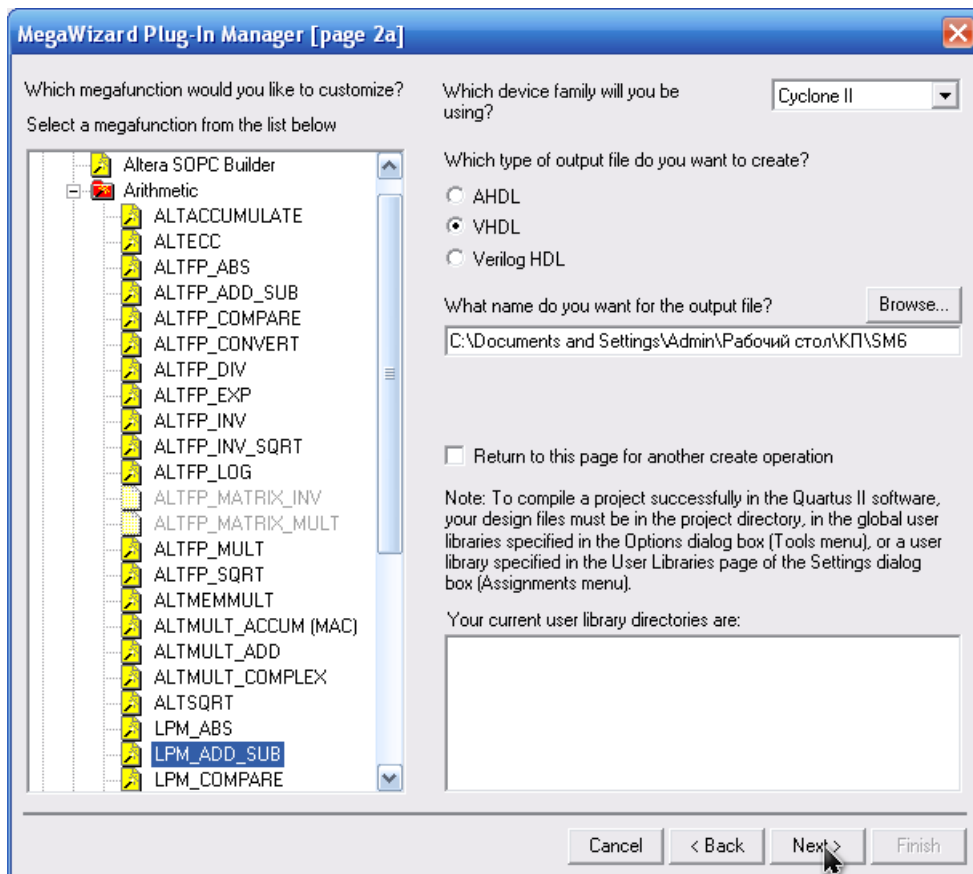


Рисунок 3.17-Створення суматора

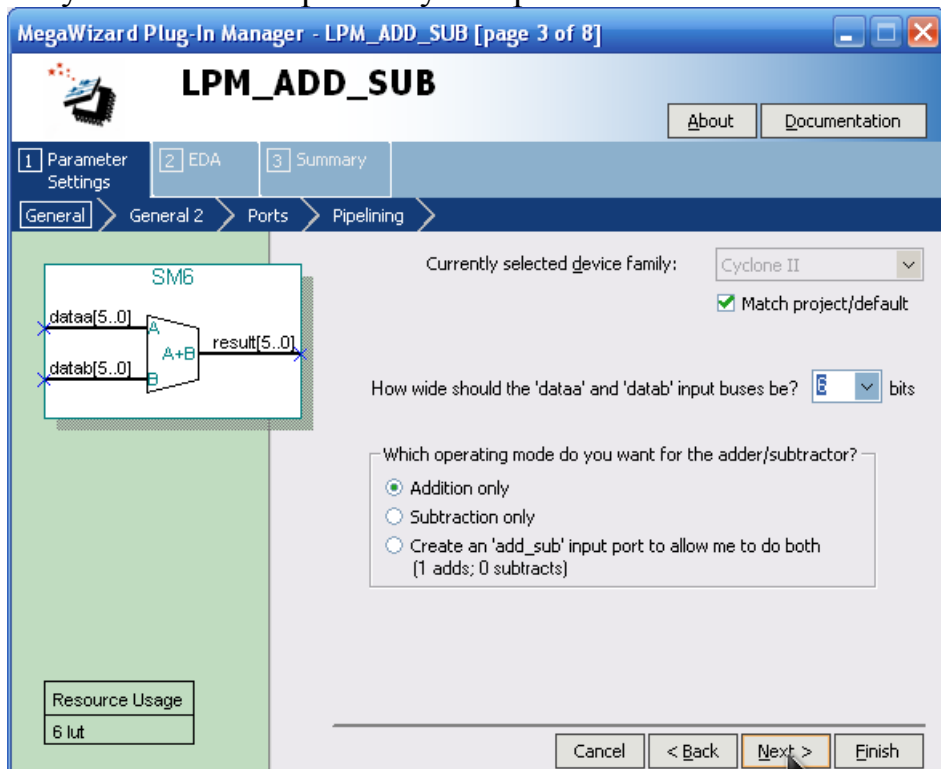


Рисунок 3.18-Вибір розрядності та функціональності суматора

У наступному вікні діалогу можна задати параметри вхідних операндів - змінні чи константи, зі знаками чи без знаків (рисунок 3.19).

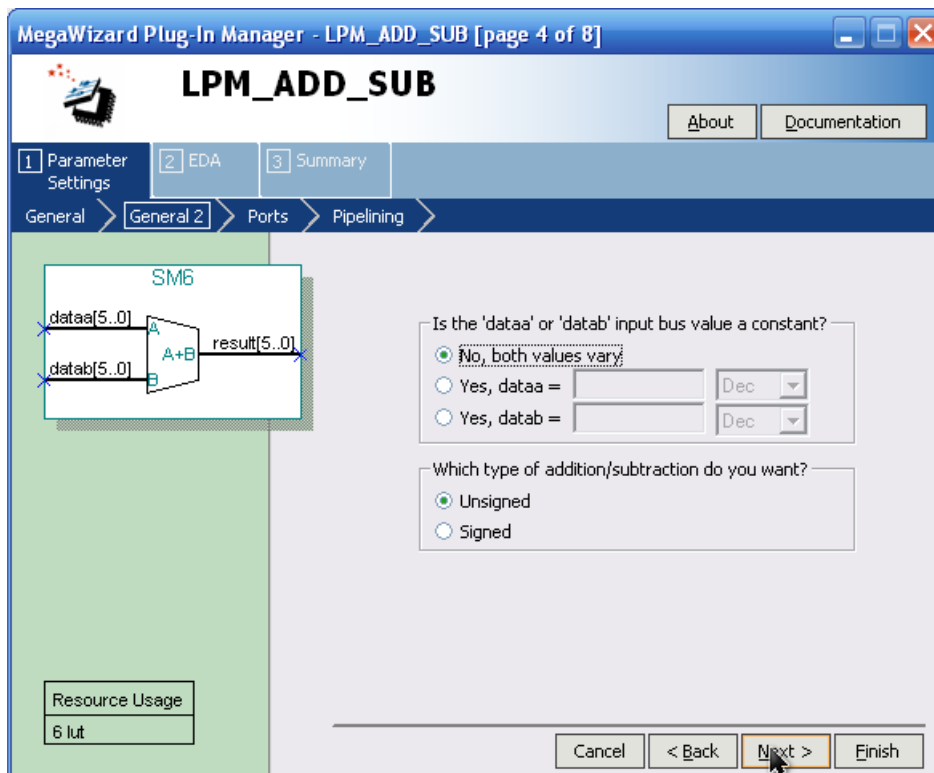


Рисунок 3.19-Вибір формату операндів

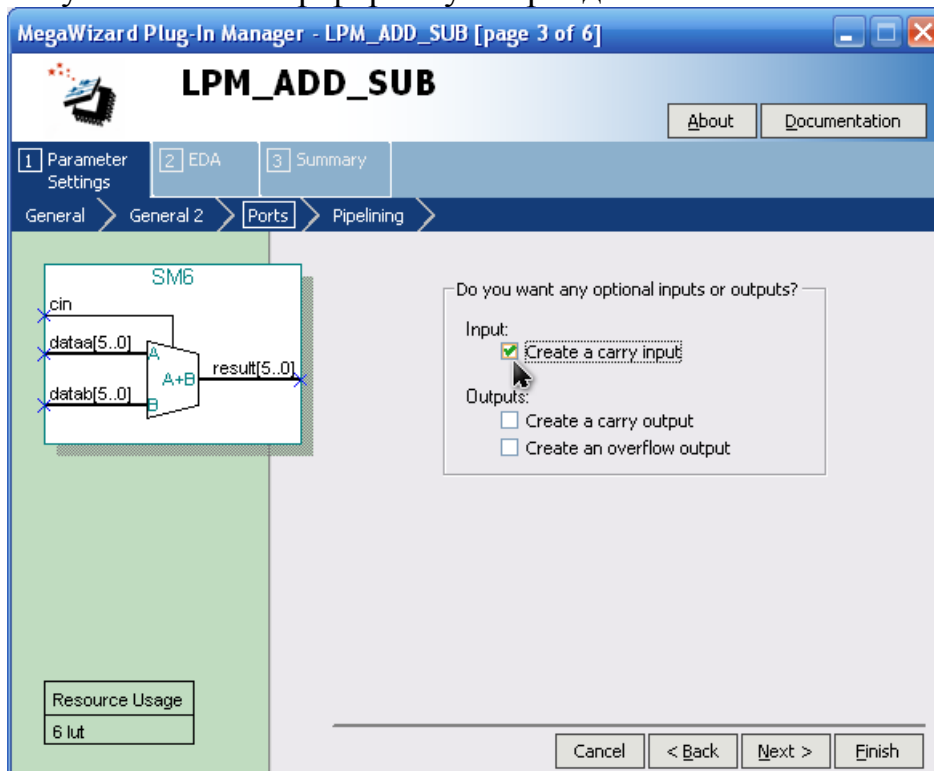


Рисунок 3.20-Вибір вхідного/вихідного переносу та переповнення

Наступне вікно призначене для додавання вхідного/вихідного переносу та виходу переповнення суматора (рисунок 3.20). У останньому вікні конфігурування елементу можна вказати залежність чи незалежність виконання операції (підсумовування) від тактового генератора (рисунок 3.21).

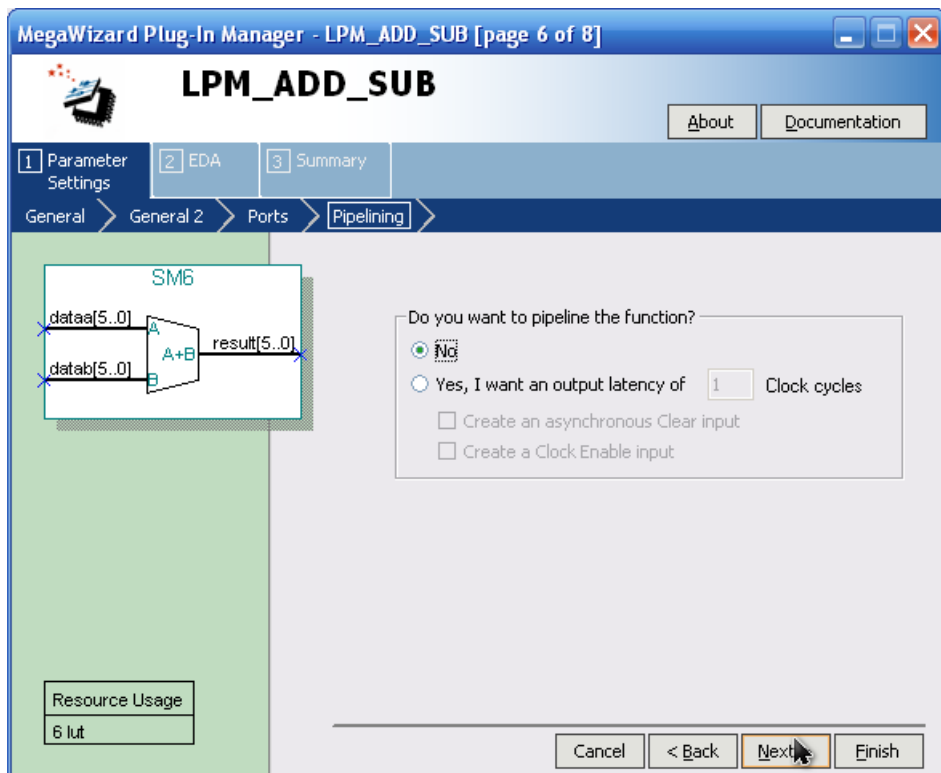


Рисунок 3.21-Вказування залежності роботи від тактового сигналу

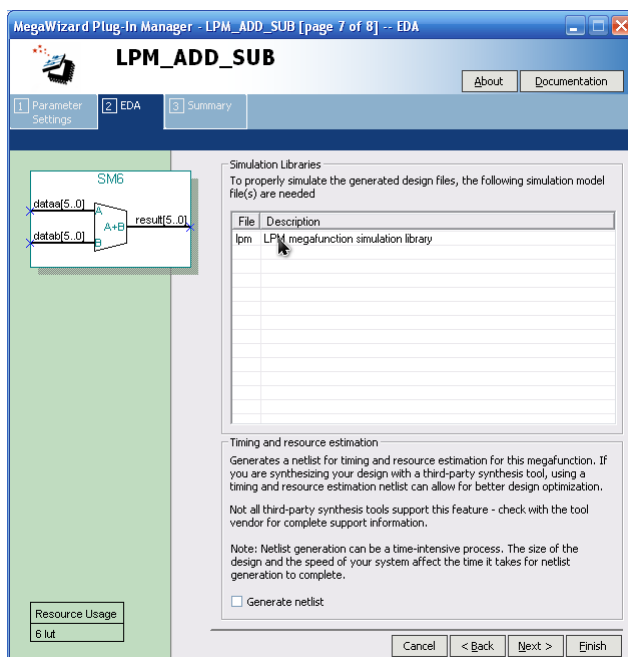


Рисунок 3.22-Список бібліотек, необхідних для створення елемента

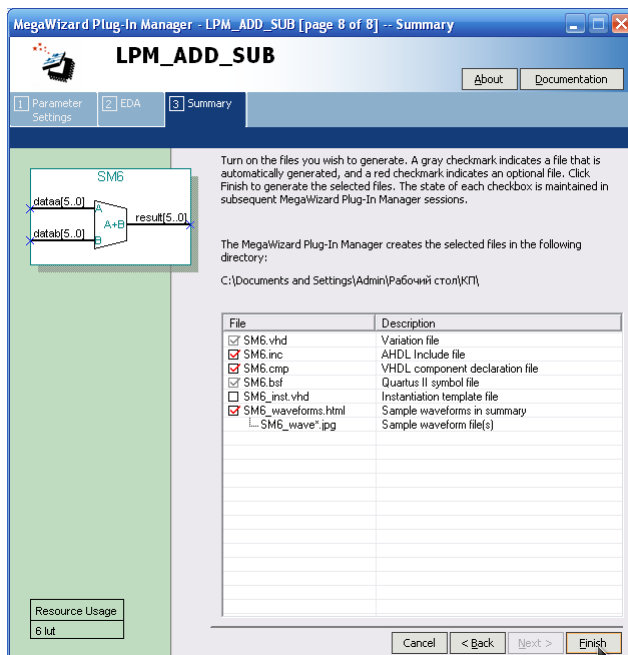


Рисунок 3.23-Список файлів, що генеруються при створенні елементу

Наступні два вікна містять інформацію про бібліотеки, що використовуються та про файли, що створюються (рисунки 3.22 та 3.23). Після натискання кнопки Finish потрібно додати створений елемент до проекту (рисунок 3.24). Після цього створений елемент з'являється у вікні вибору елементу і його можна вставляти як звичайний елемент (рисунок 3.25). Створений елемент можна додавати вільне число раз, він знаходиться у папці з ім'ям проекту у вікні вибору елемента (рисунок 3.26). Якщо при конфігурування була допущена помилка, її можна виправити, двічі натиснувши на елемент, після чого з'явиться діалог конфігурування.

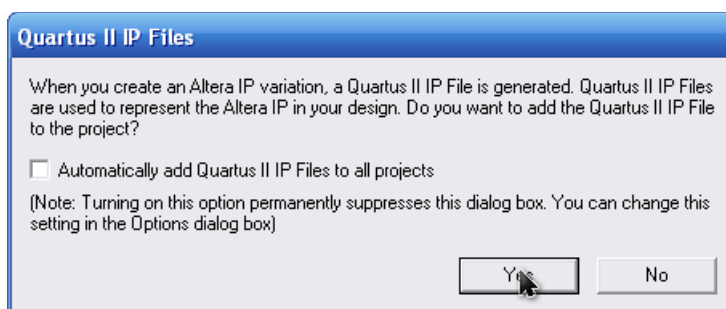


Рисунок 3.24-Підтвердження про приєднання створеного елементу до поточного проекту

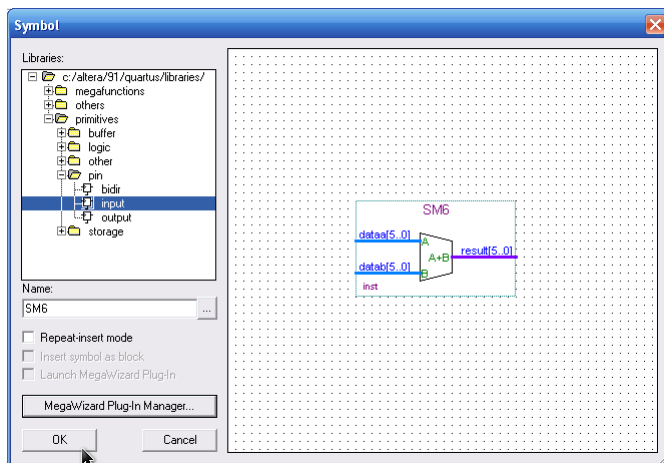


Рисунок 3.25-

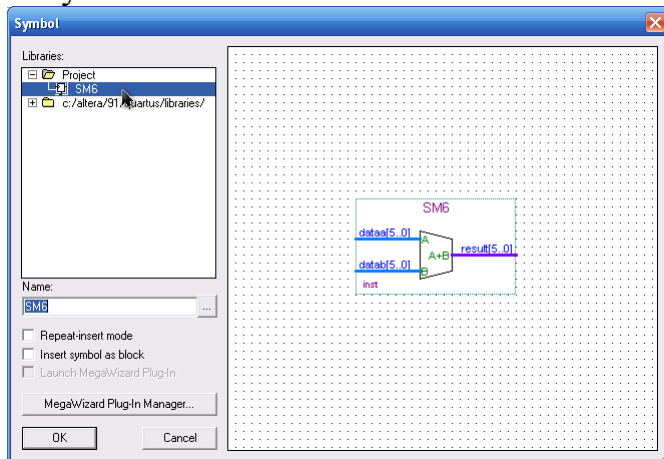


Рисунок 3.26-

Усі інші елементи створюються за аналогічною схемою із наступних мегафункцій:

- лічильник-із Arithmetic-LPM_COUNTER;
- регістр без зсуву-із Storage->LPM_FF;
- мультиплексор-із Gates->LPM_MUX;
- елемент збереження константи-із Gates->LPM_CONSTANT;
- регістр із зсувом-із Storage->LPM_SHIFTREG;
- пам'ять мікрокоманд-із Memory Compiler->ROM: 1-PORT;
- суматор за модулем 2-із Gates->LPM_XOR.

Створення пам'яті має один нюанс: при конфігурації потрібно вказати вміст пам'яті (рисунок 3.27). Це робиться за допомогою файлів пам'яті (розширення .hex або .mif). Для створення файлу пам'яті з розширенням .mif необхідно вибрати тип Memory Initialization File у вікні вибору типу створюваного файлу (рисунок 3.28). Після цього потрібно вказати кількість слів у пам'яті та розмір слів (рисунок 3.29).

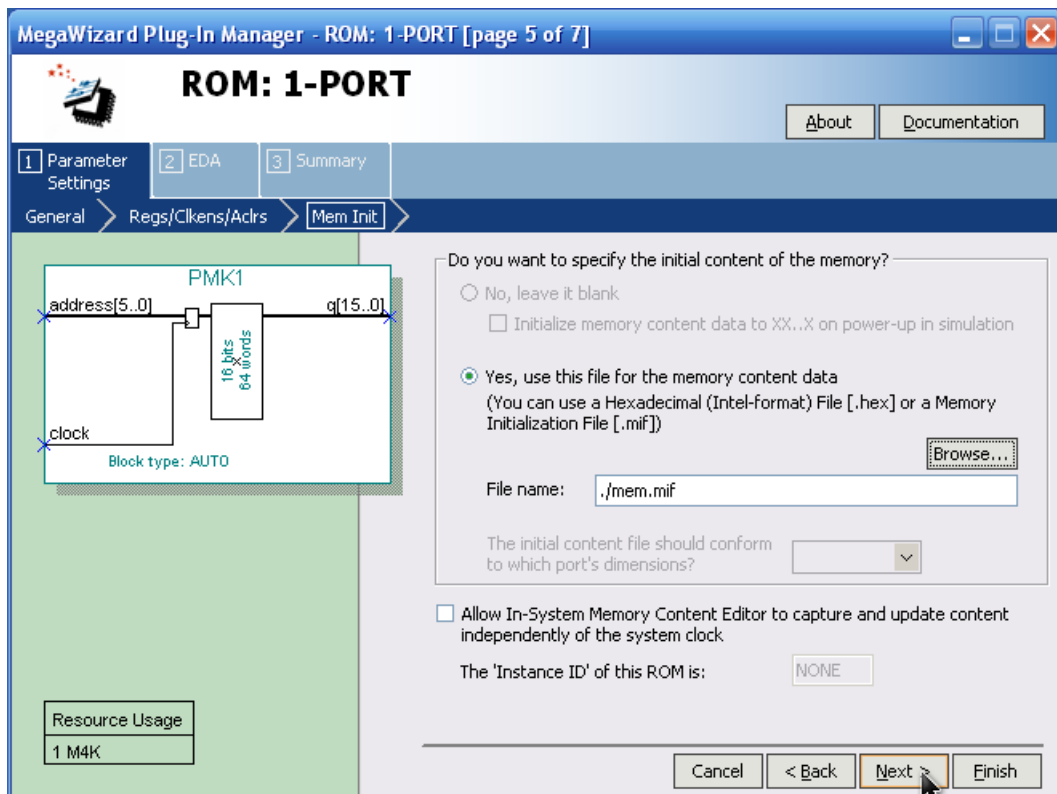


Рисунок 3.27-Задання вмісту оперативної пам'яті

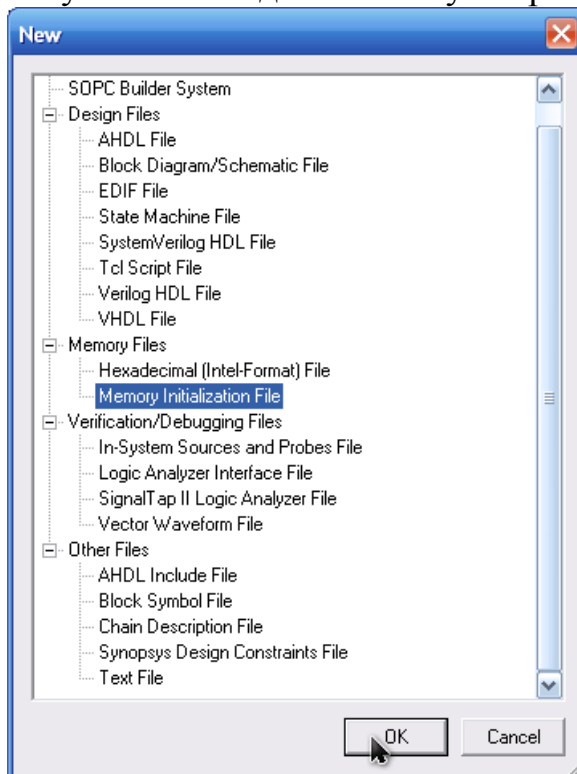


Рисунок 3.28-Створення файлу вмісту пам'яті

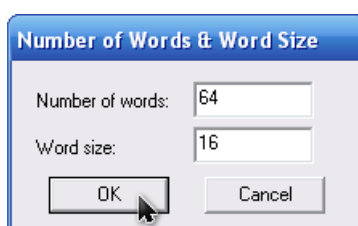
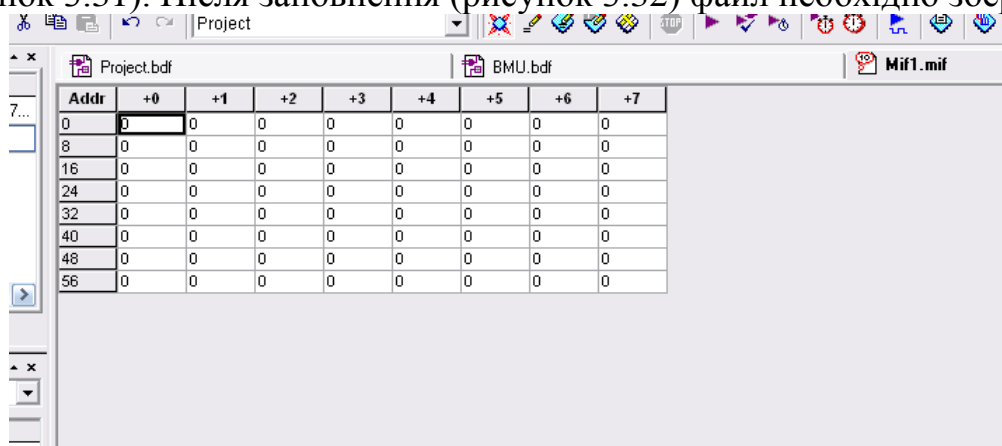


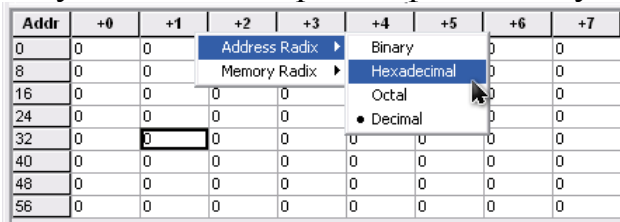
Рисунок 3.29-Задання кількості та розміру слів у пам'яті

Створений файл має наступний вигляд (рисунок 3.30). Для зміни системи числення необхідно натиснути на полі набору правою клавішею, вибрати, для пам'яті чи для адресації потрібно змінити систему числення, а потім вибрати її (рисунок 3.31). Після заповнення (рисунок 3.32) файл необхідно зберегти.



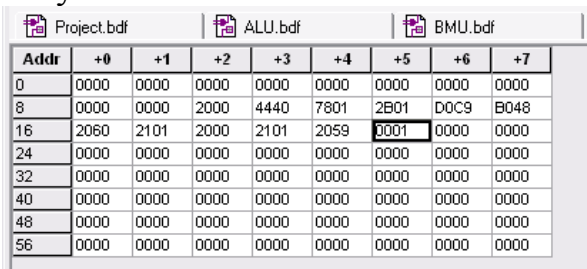
Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0
48	0	0	0	0	0	0	0	0
56	0	0	0	0	0	0	0	0

Рисунок 3.30-Створений файл вмісту пам'яті



Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	0	0					0	0
8	0	0					0	0
16	0	0	0	0			0	0
24	0	0	0	0			0	0
32	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0
48	0	0	0	0	0	0	0	0
56	0	0	0	0	0	0	0	0

Рисунок 3.31-Зміна системи числення



Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	0000	0000	0000	0000	0000	0000	0000	0000
8	0000	0000	2000	4440	7801	2B01	D0C9	B048
16	2060	2101	2000	2101	2059	0001	0000	0000
24	0000	0000	0000	0000	0000	0000	0000	0000
32	0000	0000	0000	0000	0000	0000	0000	0000
40	0000	0000	0000	0000	0000	0000	0000	0000
48	0000	0000	0000	0000	0000	0000	0000	0000
56	0000	0000	0000	0000	0000	0000	0000	0000

Рисунок 3.32-Файл пам'яті із мікропрограмою

Іноді необхідно подавати на вхід елементу проінвертований сигнал. Для цього можна виділити елемент, у контекстному меню вибрати пункт Properties, та вибрати вкладку Ports (рисунок 3.33) у якій можна проінвертувати будь-який вхід/вихід (після змін потрібно натиснути кнопку Change).

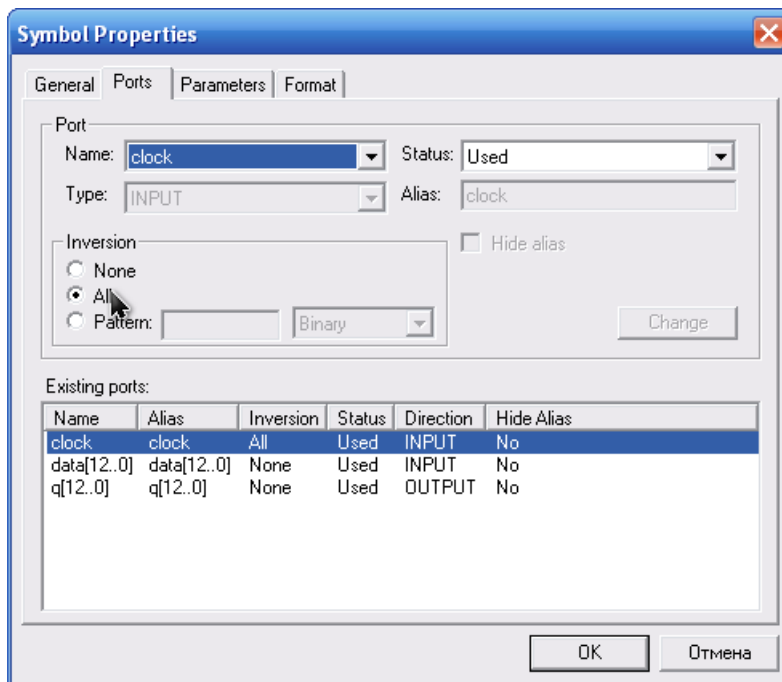


Рисунок 3.33-Інвертування виводів

Елементи з'єднуються між собою за допомогою провідників (тонка лінія) або за допомогою шин (товста лінія). Шини дозволяють значно спростувати проектування, зменшуючи візуальну складність схеми. Необов'язково з'єднувати елементи напряду, можна розірвати провідник або шину. При цьому необхідно виділити кінець шини та присвоїти їй ім'я (рисунок 3.34). Аналогічно роблять і з іншим кінцем шини.

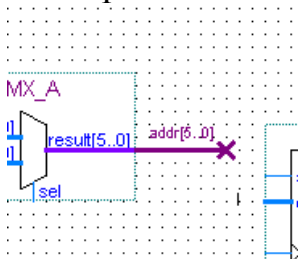


Рисунок 3.34-Іменування шини

Після набору схеми її необхідно відкомпілювати. Це робиться за допомогою кнопки Start Compilation (рисунок 3.35). За станом компіляції можна спостерігати на панелі Tasks (рисунок 3.36). В разі успішної компіляції виведеться Compilation Report (рисунок 3.37).

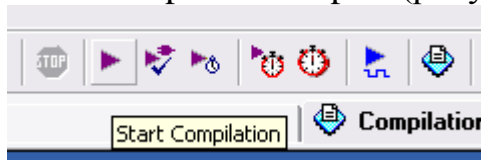


Рисунок 3.35-Кнопка Start Compilation

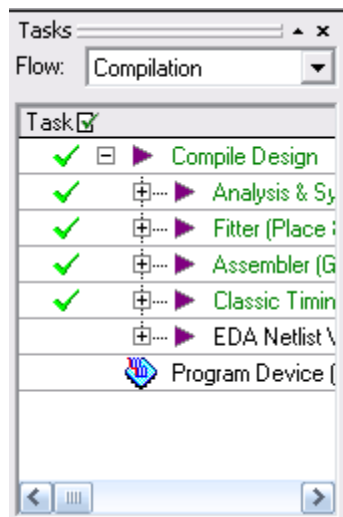


Рисунок 3.36-Стан компіляції

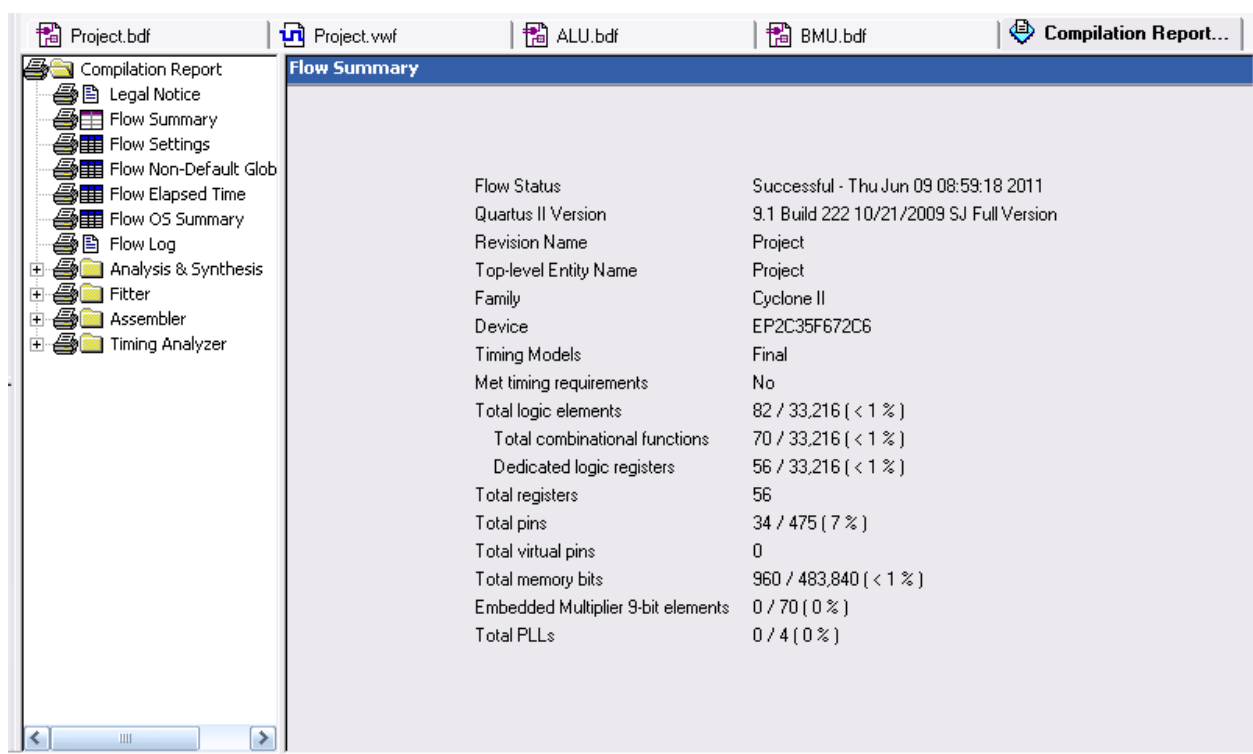


Рисунок 3.37- Compilation Report

В разі виникнення помилки вона буде описана у полі Messages, що знаходиться в нижній частині вікна.

Відлагодження схем виконується з використанням вбудованих у Quartus II засобів симуляції. Результатом симуляції є часові діаграми сигналів, що присутні на вхідних та вихідних пінах (рисунок 3.38).

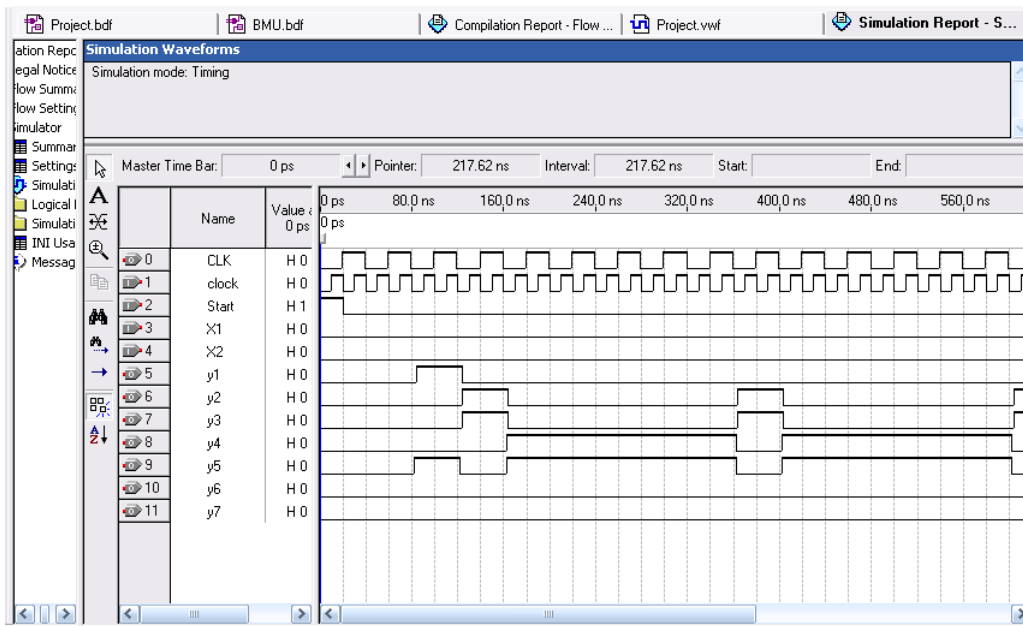


Рисунок 3.38-Часові діаграми-результати симуляції

Для симуляції потрібно створити файл типу Vector Waveform File (рисунок 3.39). Файл потрібно зберегти з ім'ям проекту. Потім потрібно додати піни, стани яких будуть відслідковуватися при симуляції. Це робиться наступним чином:

- двічі натиснути лівою клавішею миші у колонці пінів (рисунок 3.40);
- натиснути Node Finder (рисунок 3.41);
- у полі Filter вибрати Pins: all, у полі Look in - схему, робота якої симулюється і натиснути List. Отримаємо список доступних пінів. Для вибору піна потрібно його виділити і натиснути кнопку з стрілкою право (рисунок 3.42), для відміни вибору потрібно виділити пін та натиснути кнопку зі стрілкою вліво. Підтвердити свій вибір;
- у попередньому вікні вибрати систему числення (Radix) та підтвердити вибір. Після цього вибрані піни будуть додані на робоче поле файлу симуляції (рисунок 3.43).

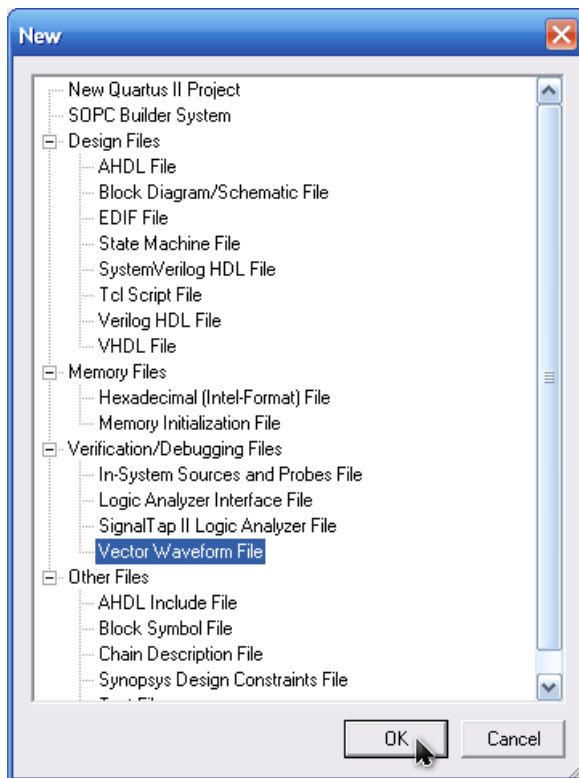


Рисунок 3.39-Створення файлу симуляції

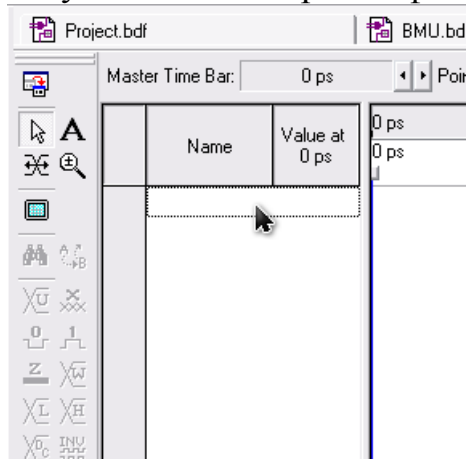


Рисунок 3.40-Виклик вікна додавання пінів

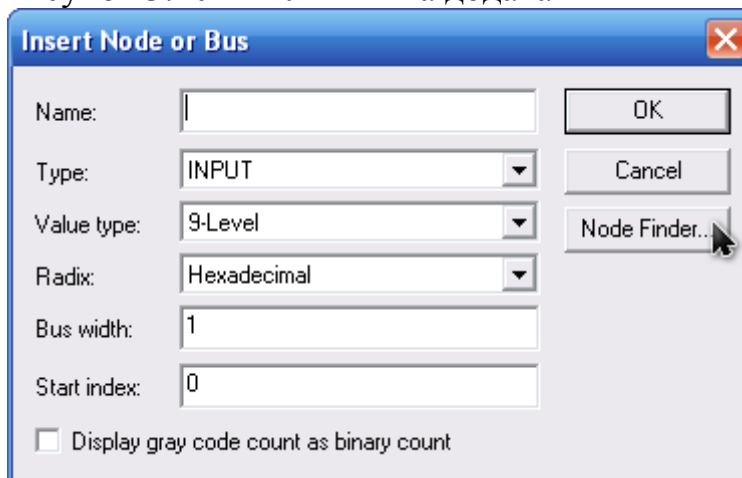


Рисунок 3.41-Вікно додавання пінів

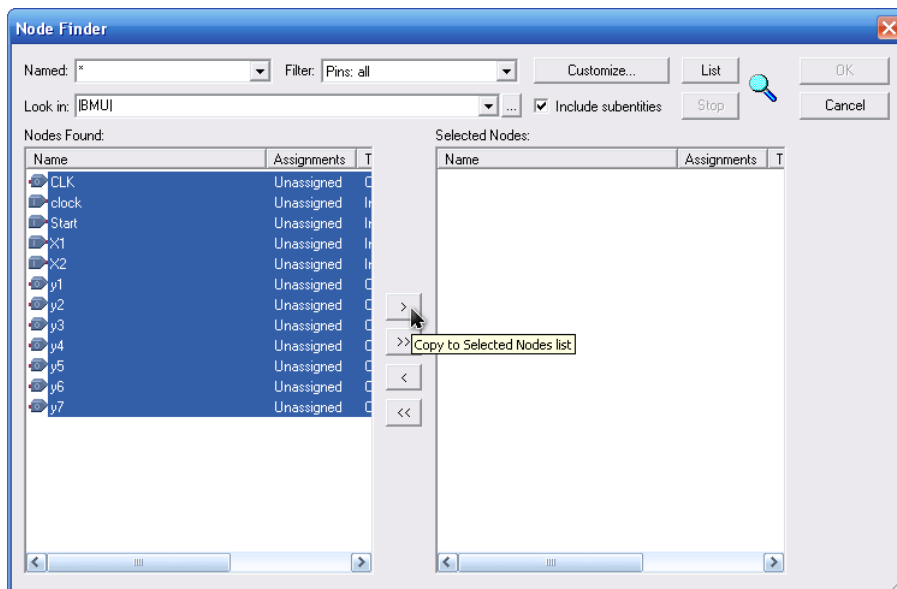


Рисунок 3.42-Вікно пошуку пінів

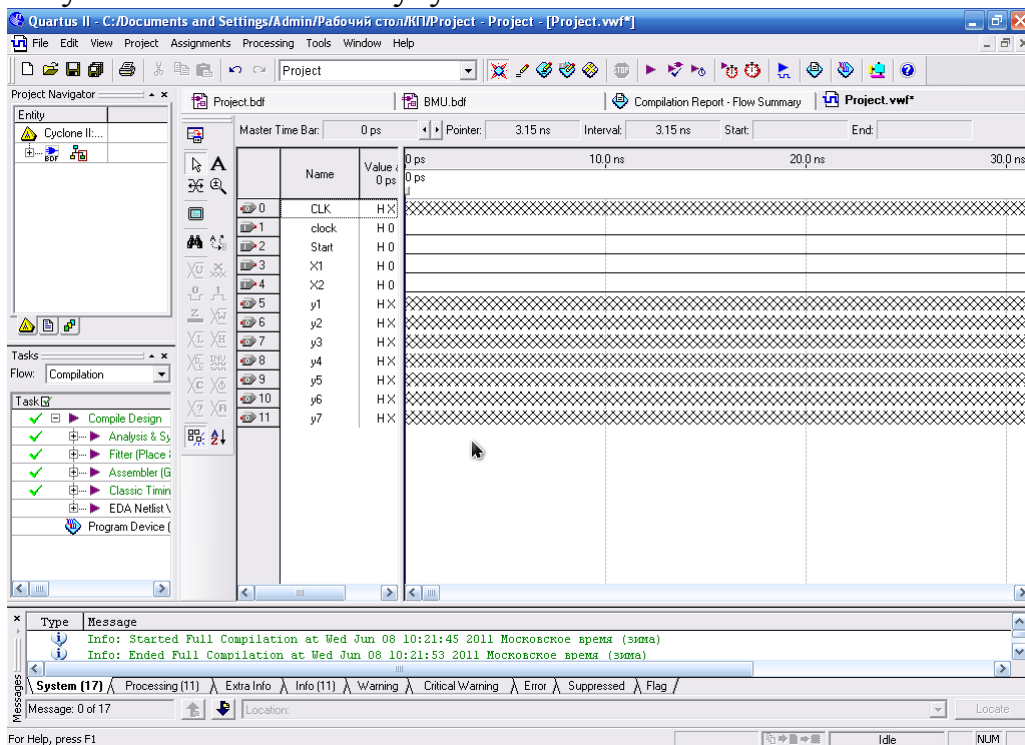


Рисунок 3.43-Піни, додані до файлу симуляції

По горизонталі на робочому полі відмічений час. Для встановлення стану вхідного піна у певний час можна скористатися інструментом Count Value (рисунок 3.44). Для цього потрібно виділити необхідний пін (натиснувши на його номер або виділивши стрілочкою) натиснути на Count Value та у формі, що з'явиться (рисунок 3.45), вибрати початкове значення та значення, на яке буде змінюватися початкове значення кожні 10нс (рисунок 3.46).

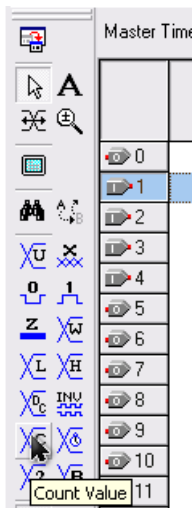


Рисунок 3.44-Інструмент Count Value

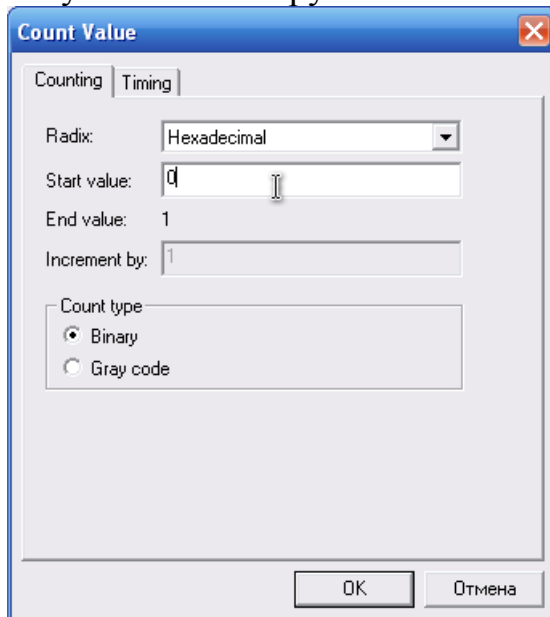


Рисунок 3.45-Форма встановлення значення на пінові

Також для встановлення значення піна на інверсне можна скористатися інструментом Waveform Editing Tool (рисунки 3.47 та 3.48).

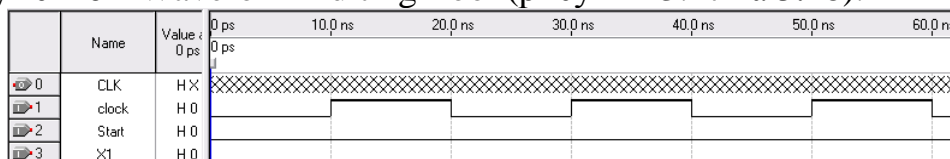


Рисунок 3.46-Встановлення стану піну clock

Для запуску симуляції потрібно натиснути на кнопку Start Simulation (рисунок 3.49). Після завершення симуляції відкриється Simulation Report (рисунок 3.50). Для зміни загального часу симуляції необхідно виконати при активній вкладці файлу симуляції Edit->End Time і у вікні, що відкриється

змінити час (рисунок 3.51).



Рисунок 3.47-Інструмент Waveform Editing Tool

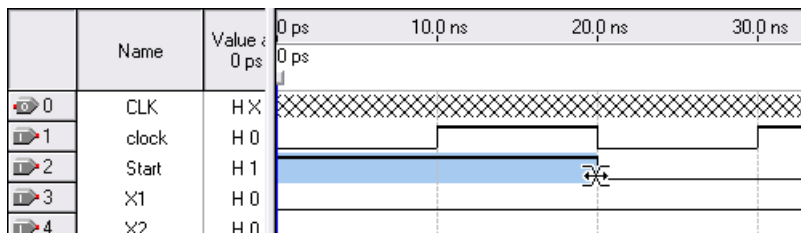


Рисунок 3.48-Використання інструменту Waveform Editing Tool

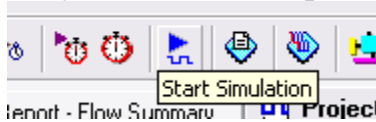


Рисунок 3.49-Кнопка Start Simulation

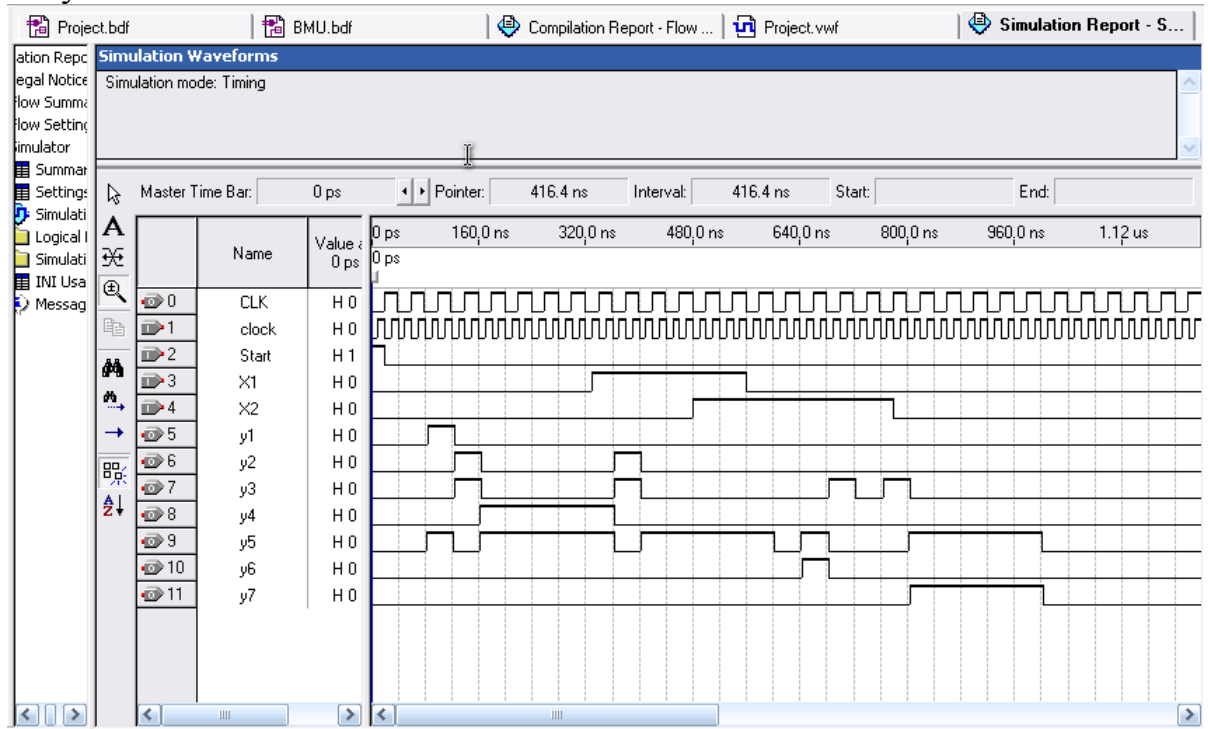


Рисунок 3.50-Simulation Report

3.4 Набір та відлагодження розробленого пристрою в цілому

Для об'єднання БМУ та арифметико-логічного пристрою створимо із них елементи та розмістимо на набірному полі головного схемного файлу Project.bdf. Для того, щоб сформувати із схемного файлу елемент потрібно відкрити потрібний схемний файл та виконати File->Create/Update->Create Symbol Files for Current File.

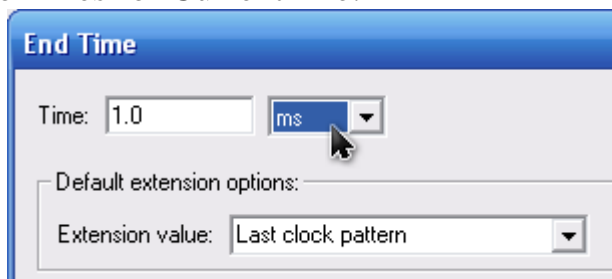


Рисунок 3.51-Зміна тривалості симуляції

Після проведення цієї операції потрібно відкрити головний схемний файл (Project.bdf) викликати вікно вибору елементу, поставити прапорець Insert symbol as block (рисунок 3.52) та вибрати елемент БМУ або арифметико-логічного пристрою. Блоки відрізняються від елементів тим, що у них можна

використовувати не всі їхні входи/виходи, з'єднувальні лінії та шини можуть під'єднуватися до будь-якої точки на блоці.

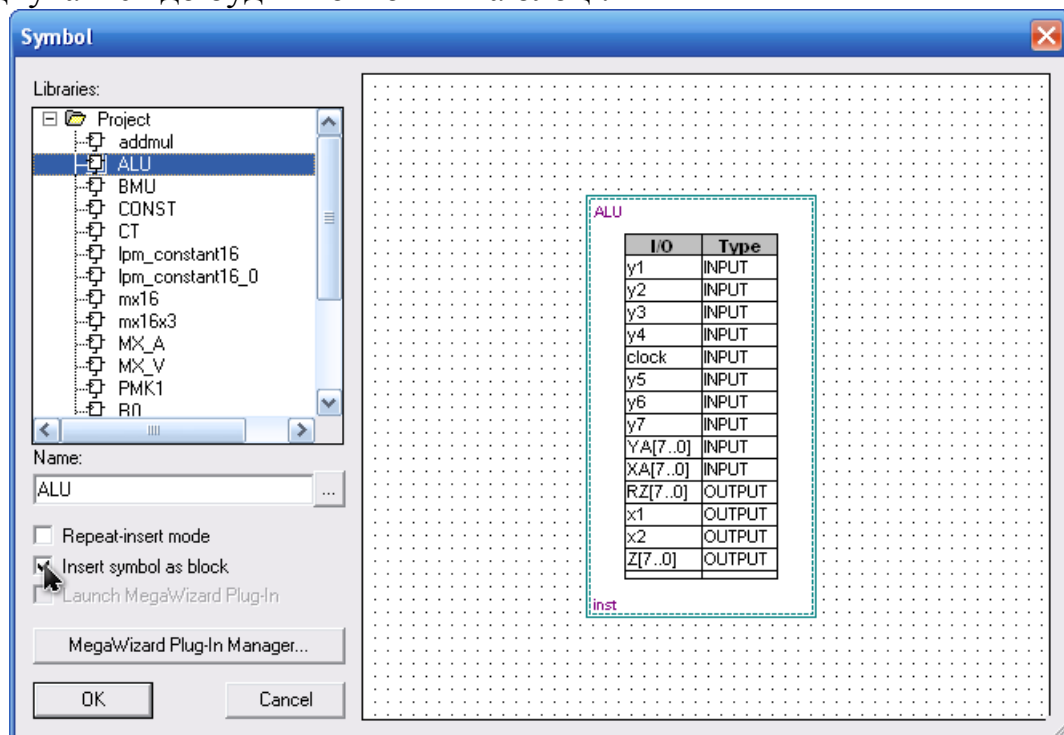


Рисунок 3.52-Вибір блоку

Правила з'єднання блоків між собою, з пінами та з елементами такі самі, як і для елементів за винятком декількох особливостей. До блоків потрібно підводити шини, що мають вигляд, як на рисунку 3.53. З'єднання потрібно іменувати іменами у подвійних «стрілочках». Так як блок не має явних виводів, кожна точка під'єднання до блоку повинна описуватися вручну. Для цього потрібно викликати пункт Mapper Properties контекстного меню стрілочки в місці під'єднання з'єднувальної лінії до блоку. У вкладці General потрібно вибрати тип виводу (вхід, вихід чи двонаправлене з'єднання-рисунок 3.54), а у вкладці Mapping – додати відношення між назвою виводу (I/O on block) та назвою з'єднання (Signals in conduit) (рисунки 3.55).

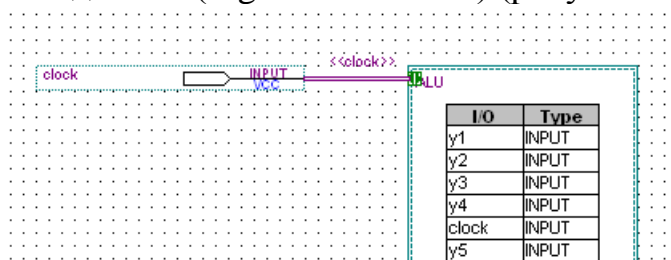


Рисунок 3.53-З'єднання блоку та піна

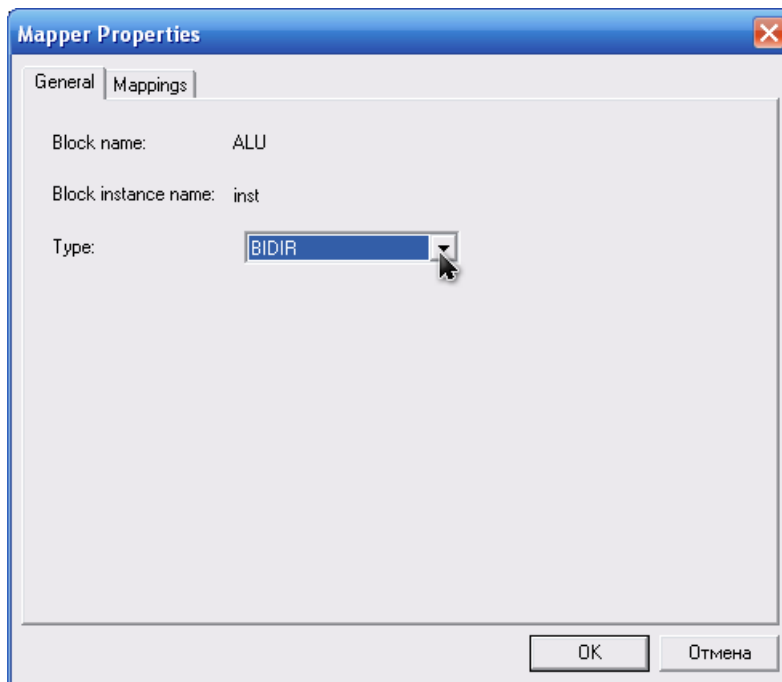


Рисунок 3.54-Вибір типу виводу

Після цього поряд із місцем під'єднання до блоку з'явиться відношення вивід блоку-назва з'єднання (рисунок 3.56). Компіляція та відлагодження пристрою в цілому не відрізняється від компіляції та відлагодження окремого схемного файлу.

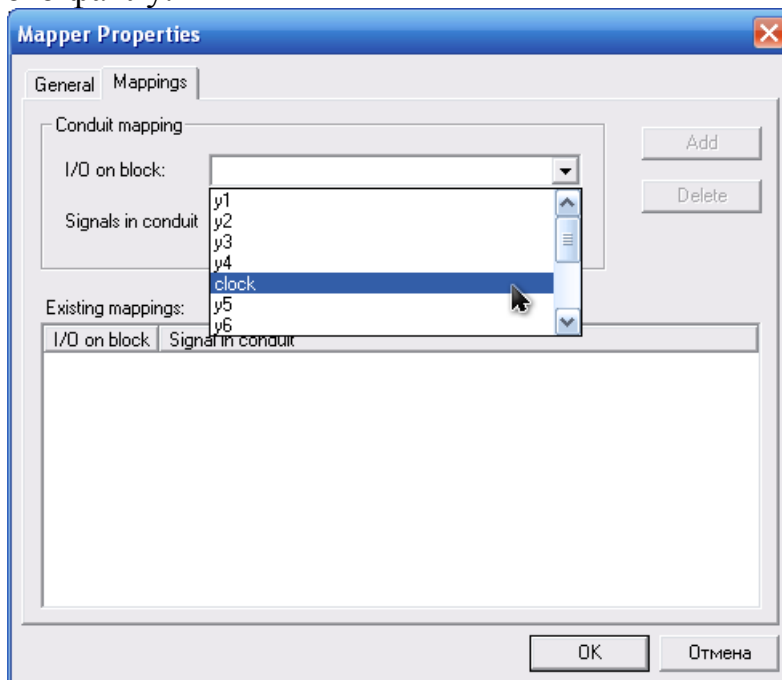


Рисунок 3.55-Вибір імені піну блока

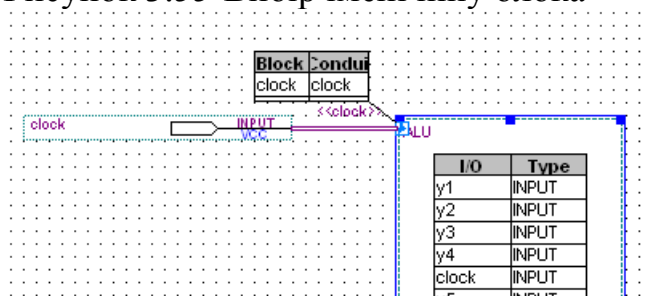


Рисунок 3.56-Встановлене відношення вивід блоку-назва з'єднання

3.5 Опис роботи пристрою

Опишемо роботу схеми БМУ.

Схема БМУ зображена на рисунку 3.57.

Часові діаграми роботи БМУ зображені на рисунку 3.58.

Так як потрібно забезпечити запис спочатку у РАМК, і тільки потім – у РМК, потрібно поділити тактову частоту для інших елементів (у тому числі, й для АЛП) на два. Це досягається використанням Т-триггеру (inst30). Вводиться сигнал CLK, частота коливань якого вдвічі менша, ніж у clock. При зміні CLK на 1, clock змінюється на 0, але зміна CLK відбувається трохи раніше за clock. Це призводить до того, що спочатку записується адреса МК, а за мить – вона подається на лічильник та RMC. По переходу CLK у 0 відбувається запис до лічильника та RMC. До наступного переходу

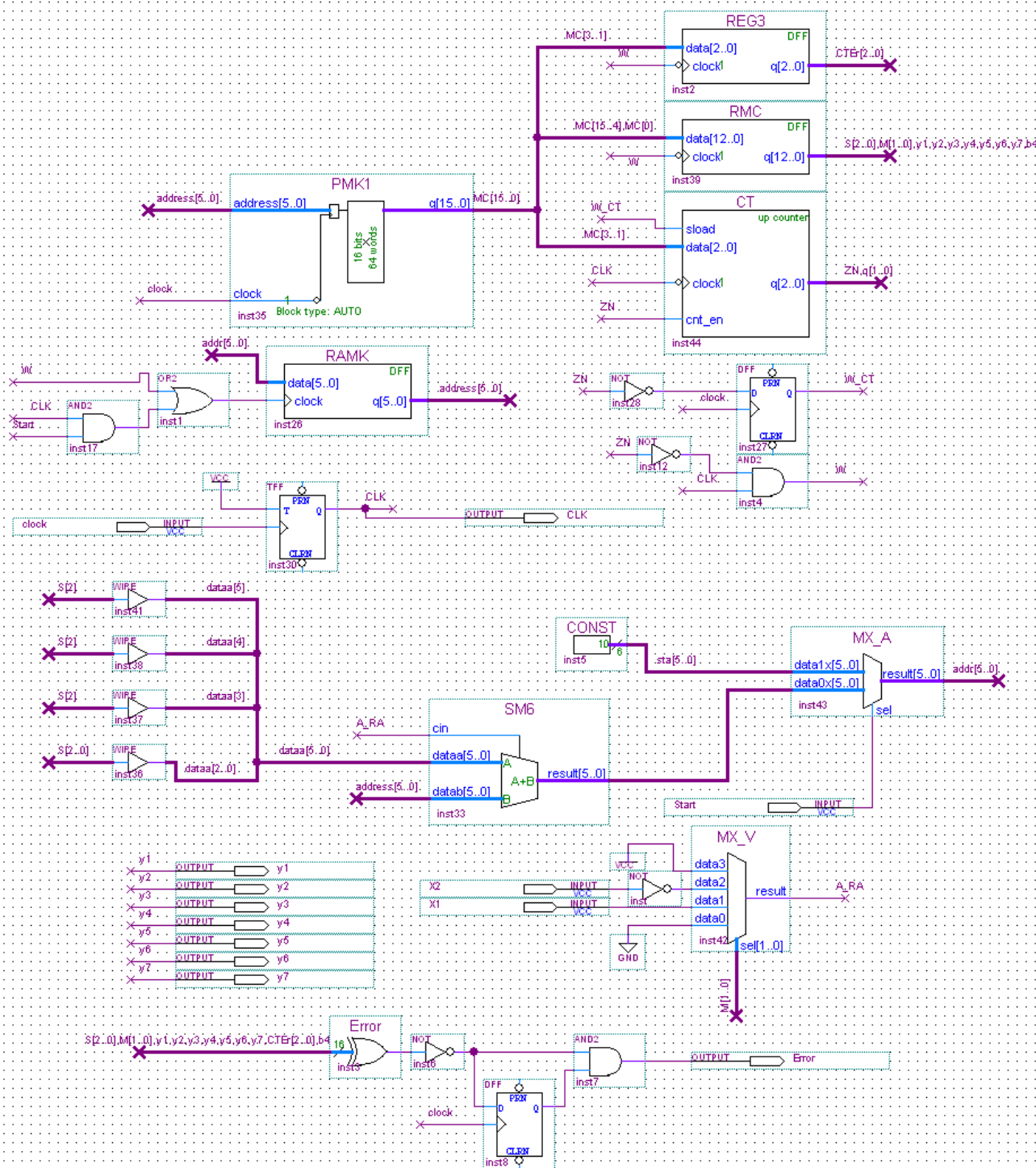


Рисунок 3.57-Схема БМУ

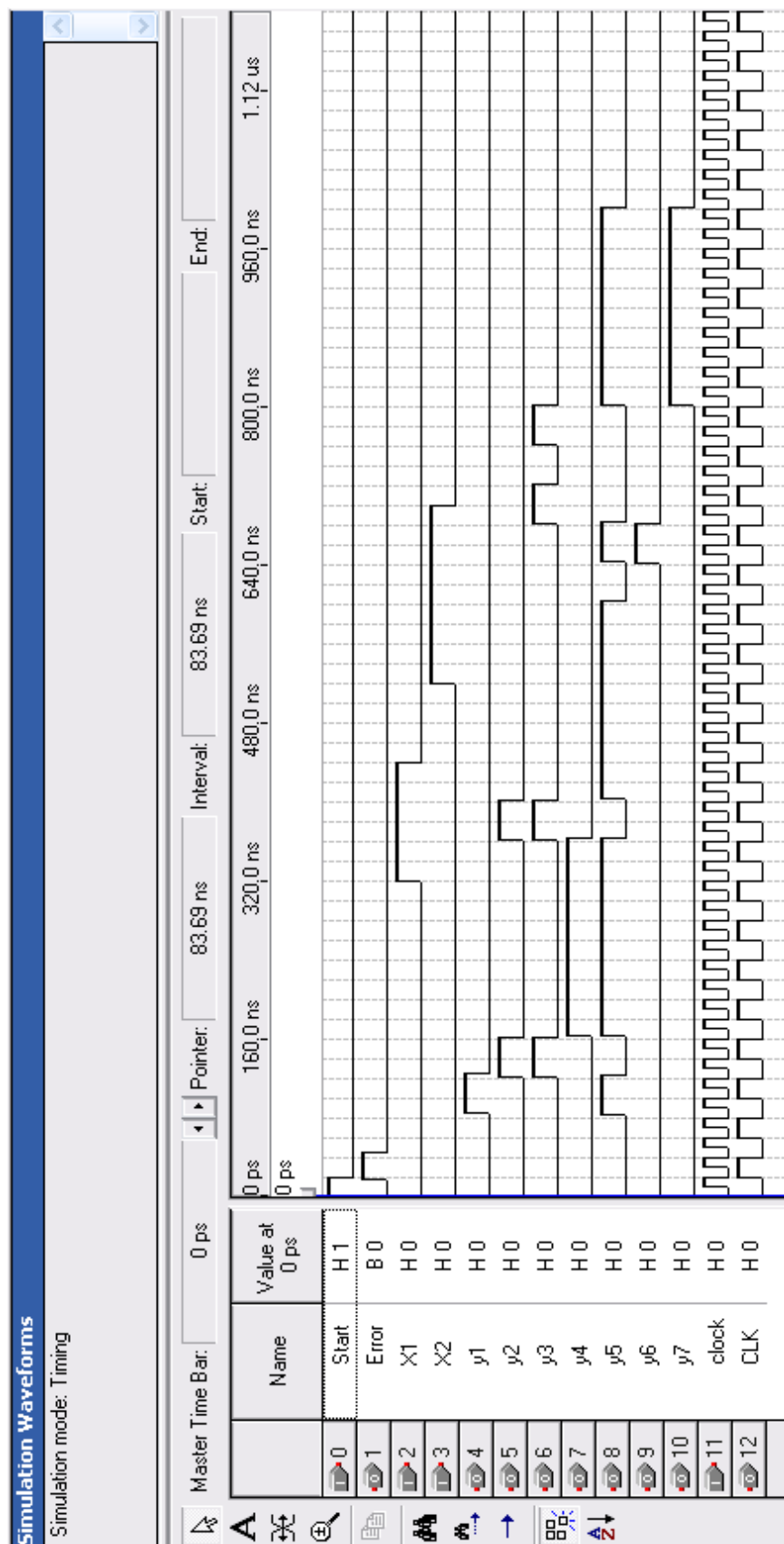


Рисунок 3.58-Часові діаграми роботи БМУ

CLK у 1 вже буде відомо, чи завантажувати наступну адресу чи ні (завдяки біту ZN).

Для того, щоб записати якесь значення у лічильник СТ (inst44), необхідно спочатку подати сигнал clock на РМК1(inst35) та sload на СТ, а потім – clock на СТ. Те саме відбувається і при записі у регістр без зсуву RMC (inst39) і при роботі суматора у АЛУ та регістрів у АЛУ. Затримку сигналу забезпечують D-тригер (inst2).

Логічна схема з I2 та АБО2 перед РАМК забезпечує запис наступної адреси лише при старті роботи пристрою (подається сигнал Start і початкова адреса мікропрограми через мультиплексор MX_A (inst43) записується у РАМК) або при наявності сигналу W. Сигнал W з'являється лише по сигналу CLK при нульовому стані лічильника (див. inst12 та inst4).

Для виділення сигналу помилки використовується суматор по модулю 2 на 16 однобітних входів (inst3) з інвертором (inst6). Для усунення короточасних одиничних вихідних сигналів використовується фільтр на основі елемента затримки (D-тригер inst8) та I2 (inst7).

Опишемо роботу схеми АЛП

Схема АЛП зображена на рисунку 3.59.

Часові діаграми роботи АЛП зображені на рисунку 3.60.

У схемі для затримки сигналу clock також використовуються D-тригери (inst14, inst15, inst17), сигнал clock перетворюється у сигнали W_RG0, W_RGX, W_RGZ, відповідно, для регістрів R0, RX та RZ. Так як можлива ситуація, коли підряд йдуть два управляючі сигнали (y5, y3 та y1, y2), призначені одному регістру, необхідно переривати сигнали W_RG0, W_RGZ у точці зміни керуючого сигналу (якщо цього не зробити, перший керуючий сигнал виконає свою дію у АЛП, а другий – ні). Для цього один керуючий сигнал зсувається (y5 та y1) за допомогою D-тригеру (inst25 та inst26), а інший залишається нерухомим. Потім вони попарно кон'юктуються з своїми тактовими сигналами (W_RG0 або W_RGZ) та сумуються за модулем 2. Цей остаточний сигнал подається на вхід clock регістру. Тепер, якщо два керуючих сигнали слідуєть один за одним, один з них зсувається,

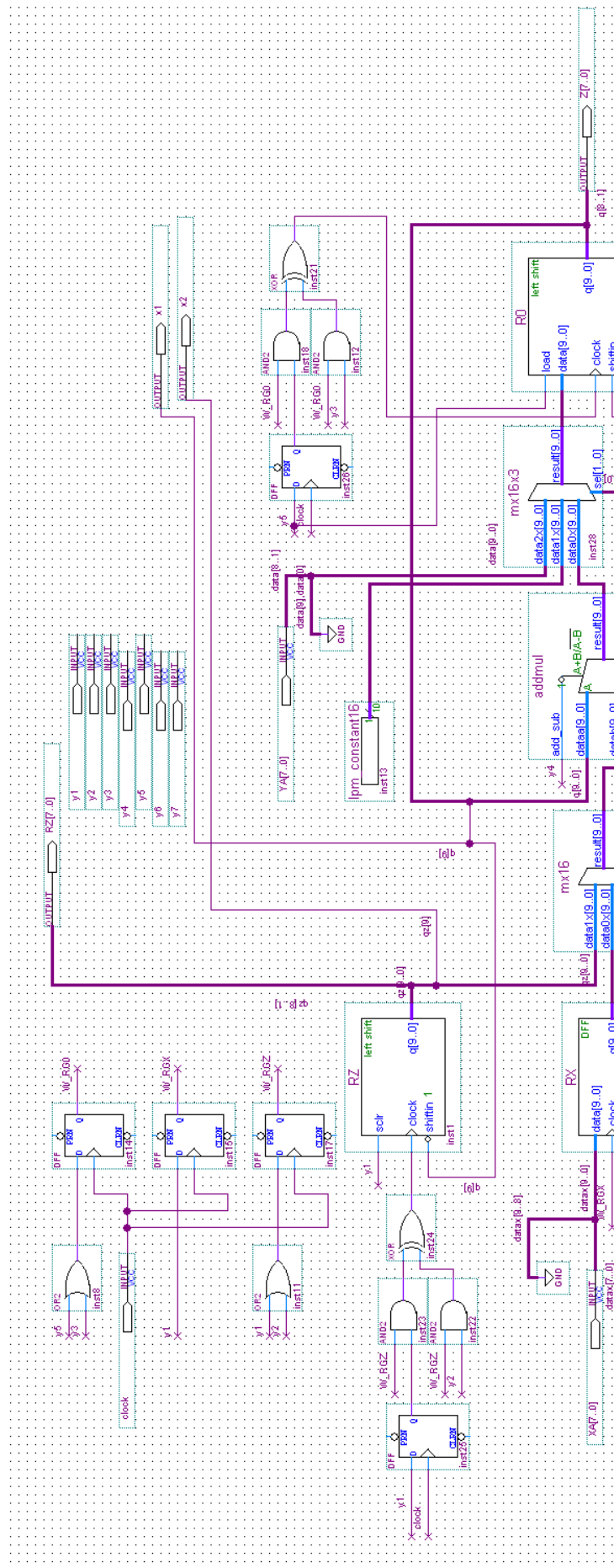


Рисунок 3.59-Схема АЛП

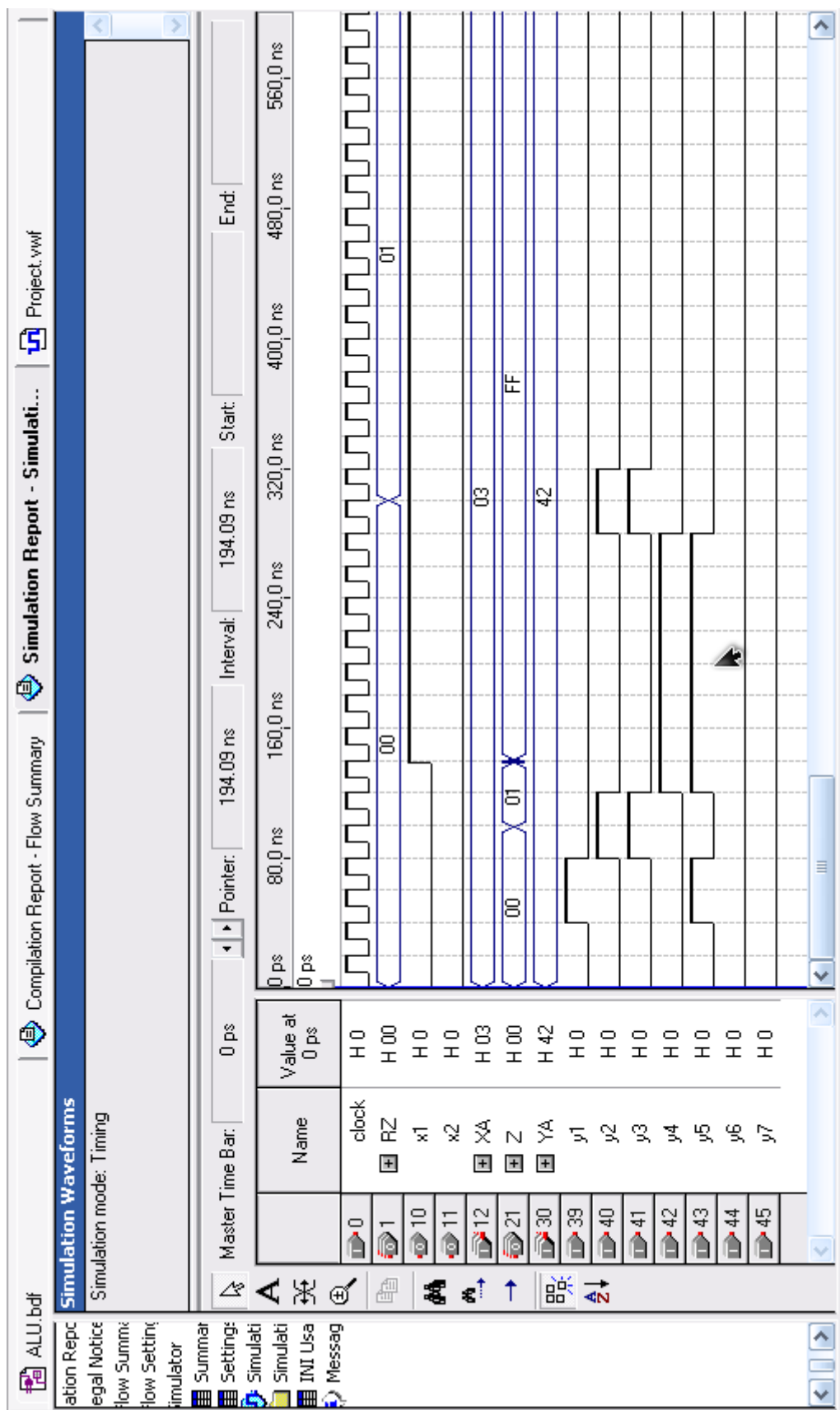


Рисунок 3.60-Часові діаграми роботи АЛП

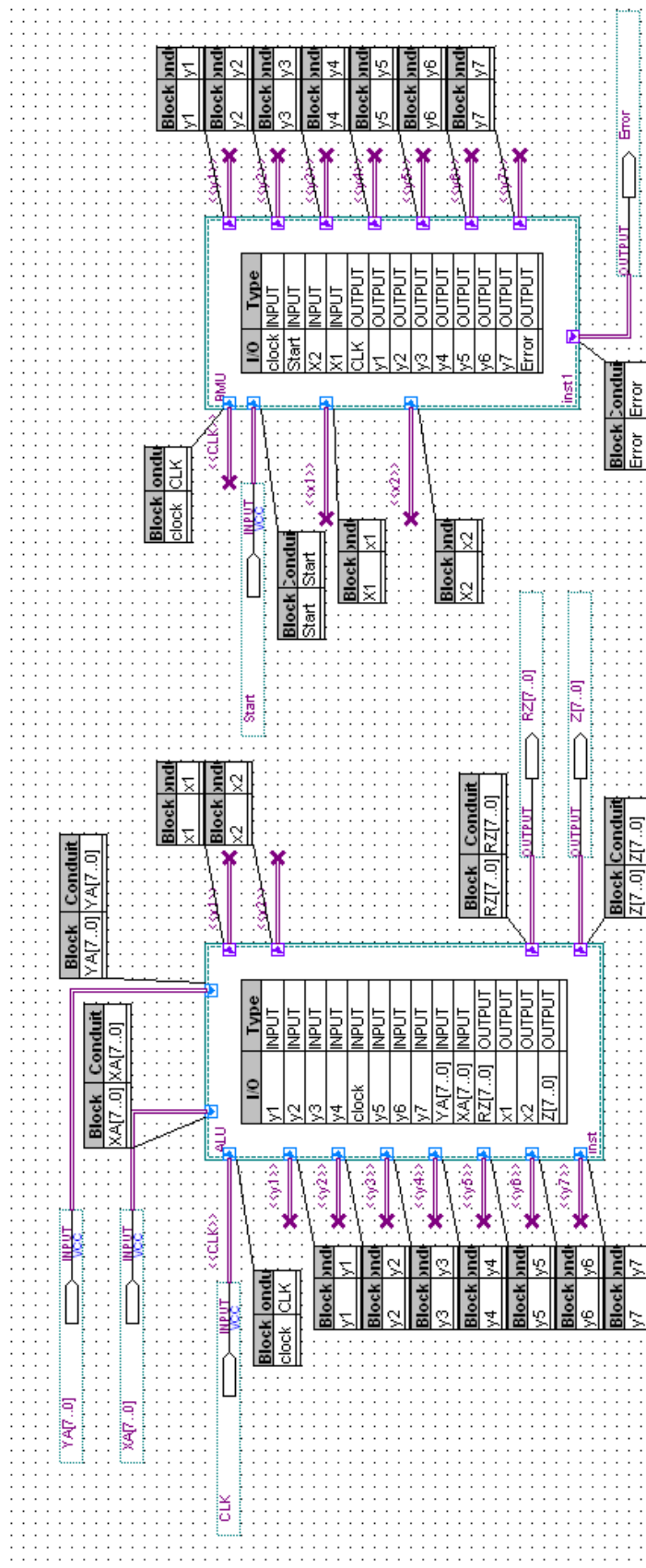


Рисунок 3.61-Схема разоблюванного прибора

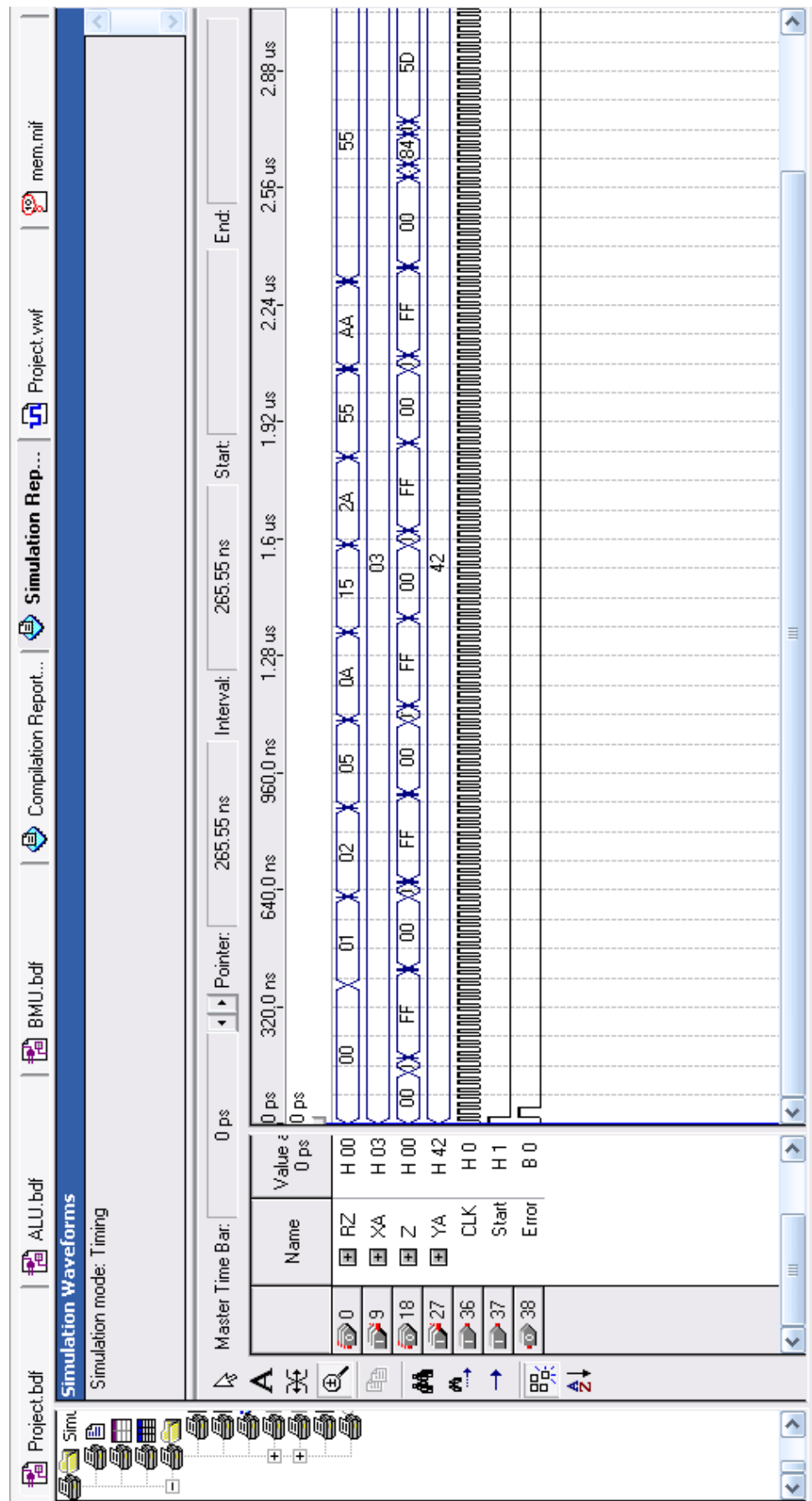


Рисунок 3.60-Часові діаграми роботи розроблюваного пристрою

і суматор по модулю 2 розділяє ці сигнали у часі на достатню відстань, щоб регістр міг виконати обидві команди.

Схема пристрою в цілому зображена на рисунку 3.61.

Часові діаграми роботи пристрою в цілому зображені на рисунку 3.62.

Звіти про компіляцію та симуляцію знаходяться в Додатку А.

3.6 Помилки та їх виправлення

Типова помилка – зміна конфігурації елемента (створеного з мегафункції) без оновлення його на наборному полі. При цьому здається, що елемент має одні параметри, а насправді – інші. Ця сама ситуація може виникнути при копіюванні однойменних схемних файлів з одного проекту в інший.

Також поширені помилки – вказування неправильних розмірів шин, відсутність джерела інформації для потрібного біту (наприклад, у шині 8 бітів і вона підписана як `a[8..1]`, а потім шина розділяється на гілки з назвами `a[0]` та `a[8..2]`), створення однойменних шин.

Більшість помилок та фальшпомилок виникає через неуважність та недостатні знання предмету розробки. Сюди відноситься незнання про особливості роботи регістрів, переносу у суматорах тощо.

Також потрібно пам'ятати, що САПР Quartus II – досить складна програма, яка може мати деякі «особливості» в роботі. Наприклад, якщо виділити усі елементи схеми АЛП та вставити їх у другий схемний файл, можна помітити, що імена шин `datax[7..0]` та `datax[9..0]` (підключення піну ХА до регістру RX) злилися в одне і'мя та приписані до однієї шини, що викликає помилку про несумісність розмірів шин. На деяких комп'ютерах в процесі компіляції створюються процеси з назвою «quartus_sh.exe», які не зникають ні після завершення компіляції, ні при завершенні роботи із САПР Quartus II. При тривалій розробці вказані процеси можуть заповнити усю доступну оперативну пам'ять. Щоб цього не сталося, потрібно вручну видаляти вказані процеси за допомогою диспетчера задач Windows.

4 Тестування розробленого пристрою на апаратному відлагоджувальному комплексі

4.1 Розмітка пінів

Під час програмної симуляції роботи пристрою (див підр. 3.3) вхідні та вихідні сигнали спостерігалися за допомогою часових діаграм. Для того, щоб мати можливість контролювати та спостерігати за роботою пристрою на апаратному комплексі, у ньому передбачені численні технічні засоби: кнопки, вимикачі, світлодіоди, LCD – дисплей, USB, VGA, PS/2 – порти та ін.. Для того, щоб співставити вхідний/вихідний пін на схемі з піном на ПЛІС, використовують програму Pin Planer. Для її запуску потрібно натиснути на відповідну кнопку (рисунок 4.1).



Рисунок 4.1-Кнопка Pin Planer

В верхній правій частині розташовані групи пінів, що присутні у схемних файлах проекту, у верхній лівій – умовне зображення пінів на ПЛІС (кольорами виділені банки пінів), в нижній частині – таблиця співвідношення між реальними пінами на ПЛІС та віртуальними пінами у схемних файлах проекту (рисунок 4.2).

Для того, щоб назначити якомусь схемному пінові пін ПЛІС потрібно у таблиці співвідношень для схемного піна (колонка Node Name) прописати пін ПЛІСу (колонка Location, рисунок 4.3). Номери пінів ПЛІС та їх призначення приводяться в інструкції по використанню апаратного відлагоджувального комплексу. Наприклад, для того, щоб виводити стан біту RZ[0] на світлодіод LEDR[0], потрібно у рядку RZ[0] прописати в колонці Location PIN_AE23.

У таблиці 4.1 приведені співвідношення між вимикачами SW[0]-SW[17] на платі пінами на ПЛІС. У таблиці 4.2 приведені співвідношення для кнопок KEY[0]-KEY[3], у таблиці 4.3 – для світлодіодів LEDR[0]-LED[17] та LEDG[0]-LEDG[7].

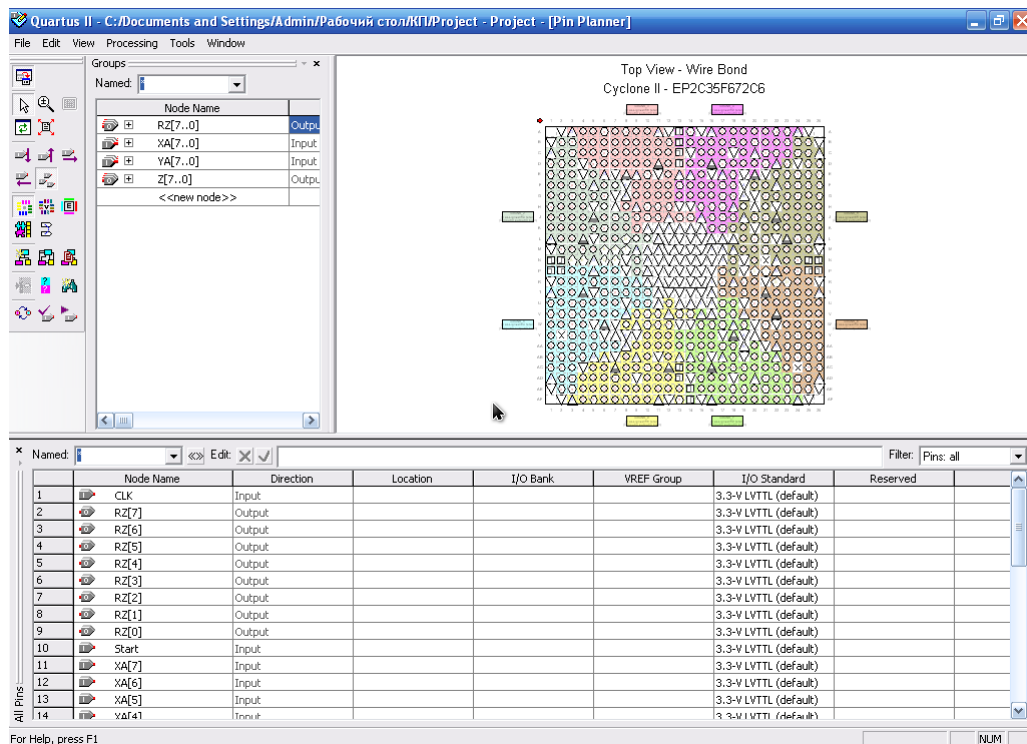


Рисунок 4.2-Вікно Pin Planer

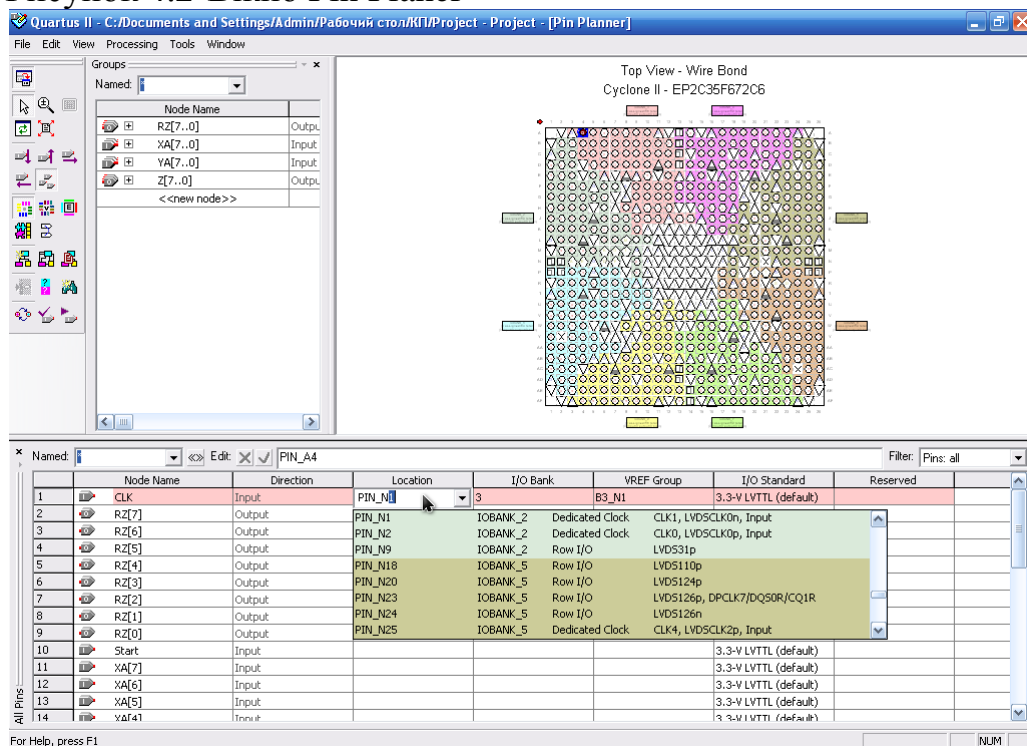


Рисунок 4.3-Притизначення віртуальному пінові реального

Таблиця 4.1-Піни, виділені на вимикачі SW[0]-SW[17]

Signal Name	FPGA Pin No.	Description
SW[0]	PIN_N25	Toggle Switch[0]
SW[1]	PIN_N26	Toggle Switch[1]
SW[2]	PIN_P25	Toggle Switch[2]
SW[3]	PIN_AE14	Toggle Switch[3]
SW[4]	PIN_AF14	Toggle Switch[4]
SW[5]	PIN_AD13	Toggle Switch[5]
SW[6]	PIN_AC13	Toggle Switch[6]
SW[7]	PIN_C13	Toggle Switch[7]
SW[8]	PIN_B13	Toggle Switch[8]
SW[9]	PIN_A13	Toggle Switch[9]
SW[10]	PIN_N1	Toggle Switch[10]
SW[11]	PIN_P1	Toggle Switch[11]
SW[12]	PIN_P2	Toggle Switch[12]
SW[13]	PIN_T7	Toggle Switch[13]
SW[14]	PIN_U3	Toggle Switch[14]
SW[15]	PIN_U4	Toggle Switch[15]
SW[16]	PIN_V1	Toggle Switch[16]
SW[17]	PIN_V2	Toggle Switch[17]

Таблиця 4.2-Піни, виділені на кнопки KEY[0]-KEY[3]

Signal Name	FPGA Pin No.	Description
KEY[0]	PIN_G26	Pushbutton[0]
KEY[1]	PIN_N23	Pushbutton[1]
KEY[2]	PIN_P23	Pushbutton[2]
KEY[3]	PIN_W26	Pushbutton[3]

Таблиця 4.3-Піни, виділені на світлодіоди LEDR[0]-LED[17] та LEDG[0]-LEDG[7]

Signal Name	FPGA Pin No.	Description
LEDR[0]	PIN_AE23	LED Red[0]
LEDR[1]	PIN_AF23	LED Red[1]
LEDR[2]	PIN_AB21	LED Red[2]
LEDR[3]	PIN_AC22	LED Red[3]
LEDR[4]	PIN_AD22	LED Red[4]
LEDR[5]	PIN_AD23	LED Red[5]
LEDR[6]	PIN_AD21	LED Red[6]
LEDR[7]	PIN_AC21	LED Red[7]
LEDR[8]	PIN_AA14	LED Red[8]
LEDR[9]	PIN_Y13	LED Red[9]
LEDR[10]	PIN_AA13	LED Red[10]
LEDR[11]	PIN_AC14	LED Red[11]
LEDR[12]	PIN_AD15	LED Red[12]
LEDR[13]	PIN_AE15	LED Red[13]
LEDR[14]	PIN_AF13	LED Red[14]
LEDR[15]	PIN_AE13	LED Red[15]
LEDR[16]	PIN_AE12	LED Red[16]
LEDR[17]	PIN_AD12	LED Red[17]
LEDG[0]	PIN_AE22	LED Green[0]
LEDG[1]	PIN_AF22	LED Green[1]
LEDG[2]	PIN_W19	LED Green[2]
LEDG[3]	PIN_V18	LED Green[3]
LEDG[4]	PIN_U18	LED Green[4]
LEDG[5]	PIN_U17	LED Green[5]
LEDG[6]	PIN_AA20	LED Green[6]
LEDG[7]	PIN_Y18	LED Green[7]
LEDG[8]	PIN_Y12	LED Green[8]

У комплексі DE2 є два генератори тактів, що працюють на частотах 27МГц та 50МГц. Плата також має SMA конектор, що дозволяє під'єднувати зовнішнє джерело тактів. У таблиці 4.4 приведені співвідношення для тактових генераторів.

Таблиця 4.4-Піни, виділені на тактові генератори

Signal Name	FPGA Pin No.	Description
CLOCK_27	PIN_D13	27 MHz clock input
CLOCK_50	PIN_N2	50 MHz clock input
EXT_CLOCK	PIN_P26	External (SMA) clock input

Після призначення віртуальним пінам імен реальних потрібно ще раз перекомпілювати проект. Результат призначення зображено на рисунках 4.4 та 4.5. Введення ХА проводиться через вимикачі SW[7]-SW[0] (старші розряди-

молодші розряди), YA – SW[15]-SW[8] (старші розряди-молодші розряди). Сигнал старту приєднано до кнопки KEY3, результат ділення виводиться на світлодіоди LEDR[7]-LEDR[0] (старші розряди-молодші розряди), результат виконання усієї функції – на світлодіоди LEDR[15]-LEDR[8] (старші розряди-молодші розряди). Тактовий сигнал має частоту 50МГц. Розводка схеми на чипі зображена на рисунку 4.6.

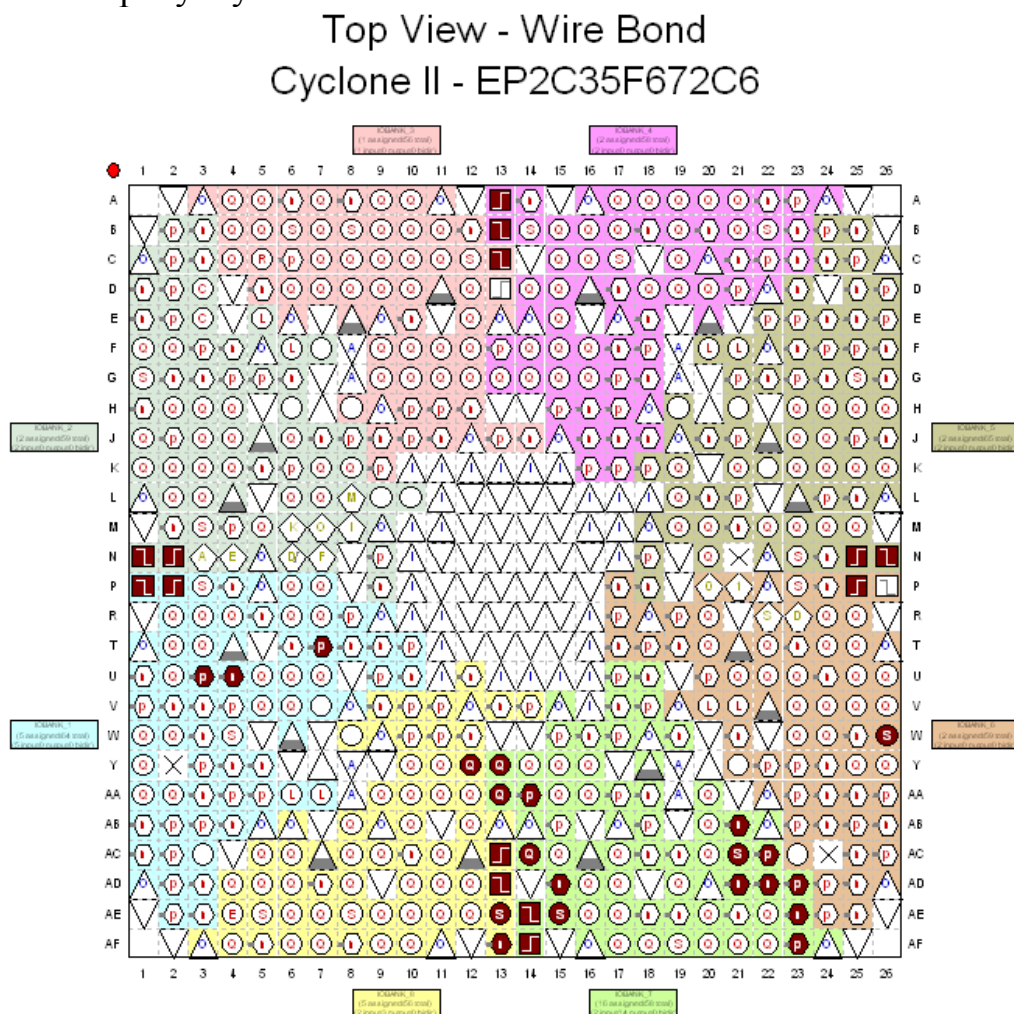


Рисунок 4.4-Результат призначення пінів

	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved
1	CLK	Input	PIN_N2	2	B2_N1	3.3-V LVTTTL (default)	
2	Error	Output	PIN_Y12	8	B8_N0	3.3-V LVTTTL (default)	
3	RZ[7]	Output	PIN_AC21	7	B7_N0	3.3-V LVTTTL (default)	
4	RZ[6]	Output	PIN_AD21	7	B7_N0	3.3-V LVTTTL (default)	
5	RZ[5]	Output	PIN_AD23	7	B7_N0	3.3-V LVTTTL (default)	
6	RZ[4]	Output	PIN_AD22	7	B7_N0	3.3-V LVTTTL (default)	
7	RZ[3]	Output	PIN_AC22	7	B7_N0	3.3-V LVTTTL (default)	
8	RZ[2]	Output	PIN_AB21	7	B7_N0	3.3-V LVTTTL (default)	
9	RZ[1]	Output	PIN_AF23	7	B7_N0	3.3-V LVTTTL (default)	
10	RZ[0]	Output	PIN_AE23	7	B7_N0	3.3-V LVTTTL (default)	
11	Start	Input	PIN_W26	6	B6_N1	3.3-V LVTTTL (default)	
12	XA[7]	Input	PIN_C13	3	B3_N0	3.3-V LVTTTL (default)	
13	XA[6]	Input	PIN_AC13	8	B8_N0	3.3-V LVTTTL (default)	
14	XA[5]	Input	PIN_AD13	8	B8_N0	3.3-V LVTTTL (default)	
15	XA[4]	Input	PIN_AF14	7	B7_N1	3.3-V LVTTTL (default)	
16	XA[3]	Input	PIN_AE14	7	B7_N1	3.3-V LVTTTL (default)	
17	XA[2]	Input	PIN_P25	6	B6_N0	3.3-V LVTTTL (default)	
18	XA[1]	Input	PIN_N26	5	B5_N1	3.3-V LVTTTL (default)	
19	XA[0]	Input	PIN_N25	5	B5_N1	3.3-V LVTTTL (default)	
20	YA[7]	Input	PIN_U4	1	B1_N0	3.3-V LVTTTL (default)	
21	YA[6]	Input	PIN_U3	1	B1_N0	3.3-V LVTTTL (default)	
22	YA[5]	Input	PIN_T7	1	B1_N0	3.3-V LVTTTL (default)	
23	YA[4]	Input	PIN_P2	1	B1_N0	3.3-V LVTTTL (default)	
24	YA[3]	Input	PIN_P1	1	B1_N0	3.3-V LVTTTL (default)	
25	YA[2]	Input	PIN_N1	2	B2_N1	3.3-V LVTTTL (default)	
26	YA[1]	Input	PIN_A13	4	B4_N1	3.3-V LVTTTL (default)	
27	YA[0]	Input	PIN_B13	4	B4_N1	3.3-V LVTTTL (default)	
28	Z[7]	Output	PIN_AE13	8	B8_N0	3.3-V LVTTTL (default)	
29	Z[6]	Output	PIN_AF13	8	B8_N0	3.3-V LVTTTL (default)	
30	Z[5]	Output	PIN_AE15	7	B7_N1	3.3-V LVTTTL (default)	
31	Z[4]	Output	PIN_AD15	7	B7_N1	3.3-V LVTTTL (default)	
32	Z[3]	Output	PIN_AC14	7	B7_N1	3.3-V LVTTTL (default)	
33	Z[2]	Output	PIN_AA13	7	B7_N1	3.3-V LVTTTL (default)	
34	Z[1]	Output	PIN_Y13	7	B7_N1	3.3-V LVTTTL (default)	
35	Z[0]	Output	PIN_AA14	7	B7_N1	3.3-V LVTTTL (default)	
36	<new node>>						

Рисунок 4.5-Призначення пінів

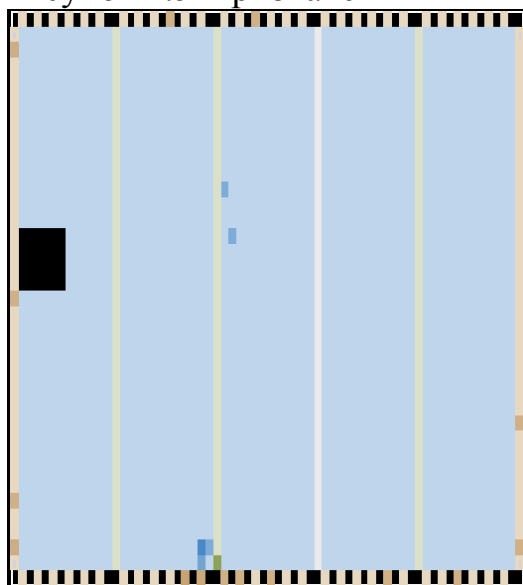


Рисунок 4.6-Розміщення схеми на чипі

4.3 Підключення відлагоджувального комплексу

Підключення комплексу до комп'ютера відбувається наступним чином:

- підключити USB type B кабель до USB Blaster Port плати DE2 (рисунок 4.7), USB type A кабель – до комп'ютера;
- підключити адаптер живлення до 9V DC Power Supply Connector;
- натиснути велику червону кнопку Power ON/OFF Switch.

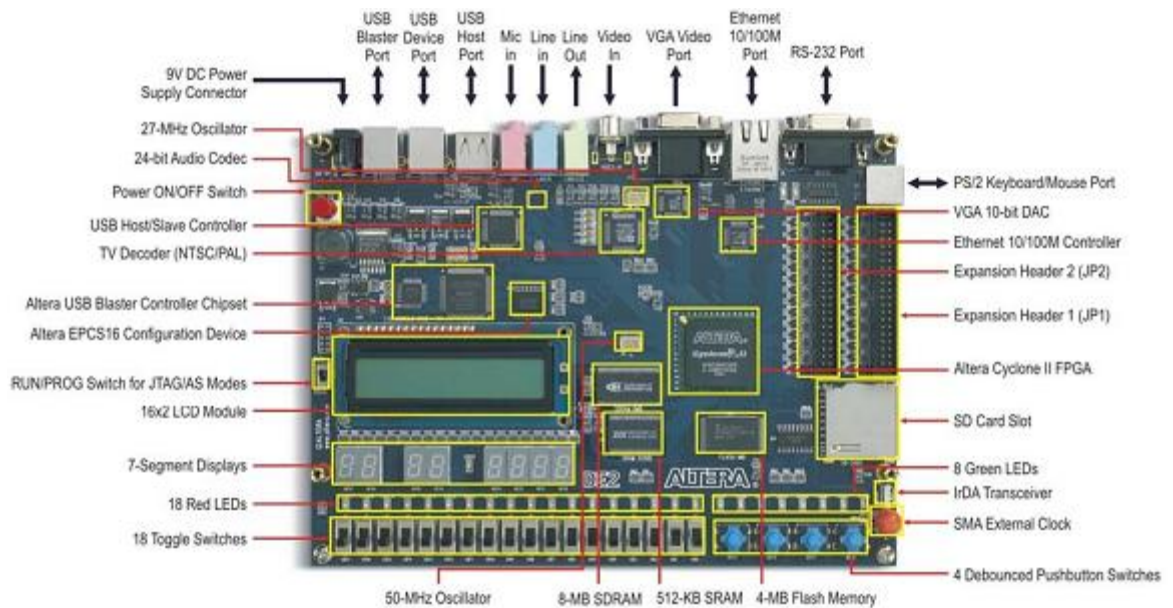


Рисунок 4.7-Шляхи взаємодії з відлагоджувальним комплексом

Комплекс DE2 програмується з використанням Altera USB-Blaster механізму. Для його роботи потрібен драйвер. При першому підключенні комплексу ОС Windows запустить Майстер нового обладнання. Потрібно вибрати установку із вказаного місця. У наступному вікні діалогу потрібно вибрати пошук драйверів у вказаному місці, заборонити пошук на змінних носіях та прописати шлях пошуку:

<Логічний диск з встановленим Quartus||>:\altera\91\quartus\drivers\usb-blaster.

Після цього продовжити. Драйвери автоматично встановляться.

4.3 Програмування ПЛІС

Програмування ПЛІС проводиться за допомогою утиліти Programmer. Щоб її запустити, потрібно виконати Tools->Programmer. У вікні, що з'явиться (рисунок 4.8) повинен розміщатися файл <Ім'я проекту>.sol, якщо його немає, потрібно вибрати його вручну, натиснувши Add File. Якщо кнопка Start неактивна, потрібно виконати наступні дії:

- у списку Mode вибрати JTAG;
- натиснути Hardware Setup..., у вікні, що з'явиться (рисунок 4.9) у вкладці Hardware Settings у списку Currently selected hardware вибрати USB-Blaster.

Після цього потрібно перевести перемикач RUN/PROG Switch у положення RUN та запустити завантаження схеми у ПЛІС, натиснувши кнопку Start. При завантаженні на платі повинен згаснути світлодіод GOOD та засвітитися LOAD. Після завершення завантаження світлодіод LOAD виключиться. Якщо при цьому засвітиться GOOD, то процес завантаження був виконаний успішно і можна тестувати роботу пристрою на відлагоджувальному комплексі.

Приклад перевірки роботи зображено на рисунку 4.10. Введені значення: XA=5 YA=3, RZ=0.00110011, Z=0.00111111.

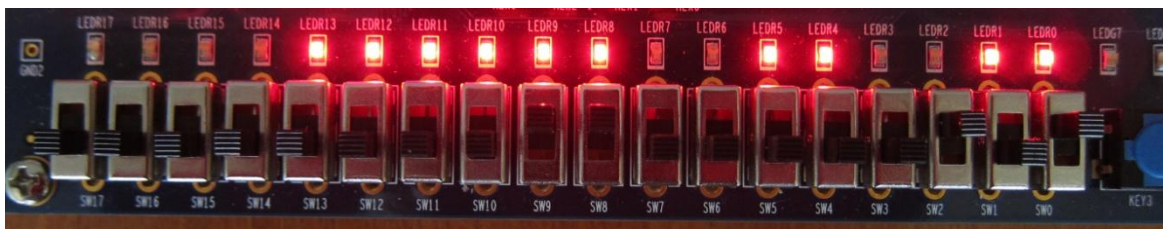


Рисунок 4.10-Перевірка працездатності пристрою

Висновки

В результаті виконання курсового проекту на елементній базі FPGA був створений фрагмент процесорного ядра, що складається з арифметико-логічного пристрою і блоку управління. Були проведені необхідні розрахунки параметрів БМУ та операційного пристрою, приведені відповідні структурні схеми.

Був виконаний набір електричної схеми пристрою у САПР Quartus II. Отримані схеми перевірено на відсутність помилок та перевірено на правильність роботи. за допомогою побудови часових діаграм роботи пристрою засобами САПР Quartus II.

Отримані електричні схеми були завантажені у апаратний відлагоджувальний комплекс Altera DE2. Отриманий пристрій працював у відповідності з поставленим завданням.

Отже, САПР Quartus II є потужним засобом розробки логічних пристроїв для подальшого завантаження у ПЛІС. Апаратний відлагоджувальний комплекс Altera DE2 добре доповнює САПР Quartus II та дозволяє впевнитись у реальній працездатності розроблюваного пристрою.

Процес виконання роботи був докладно описаний.

Список скорочень

БМУ – блок мікропрограмного управління.

АЛП – арифметико-логічний пристрій.

МК - мікрокоманда.

РМК – регістр мікрокоманд.

РАМК – регістр адрес мікрокоманд.

ПМК – пам'ять мікрокоманд.

САПР – система автоматизованого проектування.

Список літератури:

1. Жабін В.І., Жуков І.А., Клименко І.А., Стіренко С.Г. – Арифметичні та управляючі пристрої цифрових ЕОМ: Навчальний посібник. – К.:БЕК+, 2008. – 176 с.
2. Жабін В.І., Жуков І.А., Клименко І.А., Ткаченко В.В. – Прикладна теорія цифрових автоматів: Навч. посібник. – К.: Книжкове вид-во НАУ, 2007. – 364 с.
3. User Manual: Altera DE2 Board. - Altera Corporation, 2006. – 72 с.
4. Микропроцессоры. – URL:
<http://dfe3300.karelia.ru/koi/posob/microcpu/index.html>
5. Могнонов П.Б.. – Организация микропроцессорных систем: Учебное пособие. – Улан-Удэ: Изд-во ВСГТУ, 2003. – 355 с.