МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

КОМП'ЮТЕРНА СХЕМОТЕХНІКА

ЛАБОРАТОРНИЙ ПРАКТИКУМ

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра
за освітньою програмою «Комп'ютерні системи та мережі»
спеціальності 123 «Комп'ютерна інженерія»,

Київ
КПІ ім. Ігоря Сікорського
2019

УДК 681.3

К

Комп'ютерна схемотехніка: Лабораторний практикум [Електронний ресурс] : навч. посіб. для студ. освітньої програми «Комп'ютерні системи та мережі» за спеціальністю 123 «Комп'ютерна інженерія» / О. А. Верба, В. І. Жабін, І. А. Клименко, В. В. Ткаченко; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 8,64 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2019. – 110 с.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 1 від 26.09.2019 р.)
за поданням Вченої ради факультету інформатики та обчислювальної техніки
(протокол № 12 від 10.06.2019 р.)

Електронне мережне навчальне видання

КОМП'ЮТЕРНА СХЕМОТЕХНІКА

ЛАБОРАТОРНИЙ ПРАКТИКУМ

Укладачі: Верба Олександр Андрійович, канд. техн. наук
Жабін Валерій Іванович, д-р. техн. наук, проф.
Клименко Ірина Анатоліївна, д-р. техн. наук, проф.
Ткаченко Валентина Василівна, канд. техн. наук, доц.

Відповідальний редактор Кулаков Ю.О., д-р. техн. наук, проф.

Рецензент: Дичка І. А., д.т.н., професор, декан факультету прикладної математики, науковий керівник кафедри програмного забезпечення комп'ютерних систем КПІ ім. Ігоря Сікорського

Навчальний посібник призначено для вивчення системи автоматизації проектування **Quartus II** та проведення лабораторних занять з дисципліни «Комп'ютерна схемотехніка» для підготовки студентів за освітньою програмою «Комп'ютерні системи та мережі» спеціальності 123 «Комп'ютерна інженерія». Розглядаються прикладні питання проектування типових вузлів, блоків і пристроїв комп'ютерів та дослідження їх характеристик в системі автоматизації проектування. Наведено тематику лабораторних робіт, вказівки до їх виконання, список літератури.

© КПІ ім. Ігоря Сікорського, 2019

ЗМІСТ

ВСТУП **4**

ЛАБОРАТОРНІ РОБОТИ **6**

Лабораторна робота № 1 ВИВЧЕННЯ СИСТЕМИ АВТОМАТИЗАЦІЇ ПРОЕКТУВАННЯ *QUARTUS II*. СТВОРЕННЯ ПРОЕКТУ 6

Лабораторна робота № 2 РОЗРОБЛЕННЯ МОДУЛЯ КОМБІНАЦІЙНОГО СУМАТОРА 35

Лабораторна робота № 3 КОМПІЛЯЦІЯ ПРОЕКТУ В САПР *QUARTUS* II. АНАЛІЗ РЕЗУЛЬТАТІВ КОМПІЛЯЦІЇ 45

Лабораторна робота № 4 СИМУЛЯЦІЯ В САПР *QUARTUS* II. МОДЕЛЮВАННЯ 56

Лабораторна робота № 5 РОЗРОБЛЕННЯ МОДУЛІВ ДЛЯ ВИКОНАННЯ АРИФМЕТИЧНИХ ОПЕРАЦІЙ В САПР *QUARTUS* II. ПРОГРАМУВАННЯ ПЛІС В САПР *QUARTUS* II 63

Лабораторна робота № 6 РОЗРОБЛЕННЯ МОДУЛІВ АРИФМЕТИКО-ЛОГІЧНИХ БЛОКІВ В САПР *QUARTUS* II. ВИВІД РЕЗУЛЬТАТІВ ВИКОНАННЯ ОПЕРАЦІЙ НА СЕМИСЕГМЕНТНІ ІНДИКАТОРИ 72

РЕКОМЕНДОВАНА ЛІТЕРАТУРА **76**

ДОДАТКИ **77**

Додаток А ВСТАНОВЛЕННЯ ДРАЙВЕРА USB-BLASTER 77

Додаток Б ІНСТРУКЦІЯ ДЛЯ КОРИСТУВАЧА DE2 VERSION 1.4 ALTERA DE2 BOARD 83

ВСТУП

Дисципліна «Комп'ютерна схемотехніка» є нормативною дисципліною підготовки фахівців ступеня бакалавр за освітньою програмою «Комп'ютерні системи та мережі» спеціальності 123 «Комп'ютерна інженерія» і призначена для вивчення принципів побудови типових вузлів, блоків, пристроїв та машин; способів оцінки їх якості; методів їх проектування; структур комп'ютера та комп'ютерних систем. Дисципліна вивчається на третьому курсі, тому вважається, що студенти засвоїли курси «Прикладна теорія цифрових автоматів» та «Комп'ютерна електроніка».

Цикл лабораторних робіт складається з шести робіт. Метою проведення циклу лабораторних робіт є набуття студентами практичних навичок проектування типових вузлів, блоків, пристроїв на великих інтегральних схемах, що програмуються за потребами споживачів, та визначати їх характеристики.

Кожній лабораторній роботі повинна передувати самостійна підготовка, в процесі якої докладно вивчається опис лабораторної роботи, відповідний розділ конспекту лекцій і літературні джерела. В процесі підготовки складається звіт про лабораторну роботу, в якому повинні бути відображені виконані пункти теоретичного завдання, а також заготовлені необхідні для експериментальної частини лабораторної роботи таблиці, діаграми та ін.

За результатами роботи в лабораторії студент представляє оформлений звіт, який повинний містити короткі теоретичні відомості, необхідні для виконання завдання і відповідей на контрольні питання, усі схеми, формули, таблиці, діаграми, графіки, отримані при виконанні завдання та в процесі експериментального дослідження схем, а також висновки по роботі.

Протокол виконання кожної лабораторно роботи включає:

- титульний аркуш;
- завдання на лабораторну роботу;
- проектування пристрою заданого варіантом;
- програми моделювання розробленого пристрою на ЕОМ;
- висновки.

Лабораторна робота вважається зарахованою, якщо студент пройшов співбесіду з тематики виконаної роботи.

Лабораторні роботи виконуються з використанням системи автоматизації проектування QUARTUS II.

ЛАБОРАТОРНІ РОБОТИ

Лабораторна робота № 1

ВИВЧЕННЯ СИСТЕМИ АВТОМАТИЗАЦІЇ ПРОЕКТУВАННЯ *QUARTUS II*. СТВОРЕННЯ ПРОЕКТУ

Мета роботи:

- *Вивчення системи автоматизації проектування* ***Quartus II****. Отримання навичок створення проекту, вводу проекту в схемотехнічному режимі, роботи в графічному редакторі, розроблення функціональних блоків комбінаційних схем.*

1. Теоретичні відомості

Програмне середовище ***Quartus®* II** компанії ***Altera®*** є повною мультиплатформною системою для автоматизації проектування (САПР), що містить набір інструментів для проектування цифрових пристроїв та систем на програмовному кристалі (*SOPC*). Середовище *Quartus* II включає в себе всі утиліти, необхідні для роботи з мікросхемами *FPGA* і *CPLD*. На рис. 1.1 показані основні етапи проектування (*Design Flow*) в середовищі *Quartus* II.



Рис. 1.1. Етапи стандартного процесу проектування в САПР *Quartus* II

Стандартний процес проектування включає наступні етапи.

I. Введення опису проекту

1. Створення нового проекту, вибір сімейства й типу мікросхеми. Використовується майстер створення проектів, який запускається командою **New Project Wizard** із меню **File**.

2. Введення опису проекту в САПР. Можна використовувати текстовий або графічний опис файлів проекту. За обома способами необхідно створити вихідні файли проекту, в тому числі й файл верхнього рівня ієрархії проекту (*Top Level*). За першим способом, вихідні файли створюється на мовах проектування апаратури *Verilog HDL*, *VHDL*, або *Altera Hardware Description Language (AHDL)* за допомогою текстового редактора (*Text Editor*). За використання графічного вводу блок-схема проекту створюється в графічному редакторі (*Block Editor*) з використанням уніфікованих символів, які представляють собою окремі блоки схеми, що описуються іншими вихідними файлами, та примітиви. В графічному режимі вводу проекту за допомогою майстра **MegaWizard® Plug-In Manager** (меню **Tools)** можна створювати різні мегафункції й IP-ядера і включати їх в файли проекту.

3. Вказання початкових налаштувань проекту. Використовується редактор призначень (**Assignment Editor**), диалогового вікна **Settings** (меню **Assignments**), редактор топології (**Floorplan Editor**), та/або технологія фіксованих логічних блоків (LogicLock™).

4. Проект може бути створено за допомогою спеціальних засобів для створення систем-на-кристалі (*SOPC Builder*) або для створення систем цифрової обробки сигналів (ЦОС) (*DSP Builder*). Програмні файли для процесорного ядра *Nios*® створюються за допомогою редактора програмного забезпечення (*Software Builder*).

II. Синтез проекту

5. Синтез проекту. Використовується модуль аналіза й синтеза (**Analysis & Synthesis**).

6. Функціональне моделювання проекту. Використовується сімулятор (**Simulator**) и команда **Generate Functional Simulation Netlist**.

7. Розміщення і трасування проекту. Використовується модуль трасування (**Fitter**).

8. Попередній аналіз споживної потужності. Використовується утиліта **PowerPlay Power Analyzer**.

9. Аналіз часових затримок. Використовується утиліта аналізатор часових затримок (**Timing Analyzer**).

III. Моделювання проекту

10. Моделювання проекту з урахуванням часових затримок. Використовується симулятор (**Simulator**).

11. Покращення часових характеристик проекту. Здійснюється повторний фізичний синтез проекту, налаштування в діалоговому вікні **Settings** та в редакторі призначень, фіксовані логічні блоки.

IV. Програмування мікросхеми

12. Створення файлу для програмування пристрою на мікросхемі. Використовується модуль асемблера (**Assembler**).

13. Програмування мікросхеми за допомогою утиліти програматора (**Programmer**) й устаткування *Altera*; або перетворення формату файла для програмування.

V. Внутрішньокристальне налагодження проекту

14. Налагодження проекту на мікросхемі. Використовується вбудований логічний аналізатор (*SignalTap*® II *Logic Analyzer*), генератор контрольних сигналів (*SignalProbe*™).

Проект є основною одиницею в САПР *Quartus* II. Проект створюється за допомогою майстра створення нового проекту **New Project Wizard**, що знаходиться в меню **File**. Майстер дозволяє задати робочу директорію, де будуть зберігатися файли проекту, призначити ім'я проекту і ім'я файлу верхнього рівня ієрархії проекту. До проекту можна підключити інші вихідні файли, користувацькі бібліотеки, САПР сторонніх компаній, що використовуються, обрати сімейство або тип мікросхем для реалізації проекту або використати автоматичний підбір мікросхем компілятором *Quartus* II.

2. Порядок виконання роботи

Частина 1. Створіть новий проект

1. Запустіть САПР *Quartus* II. Для виконання циклу лабораторних робіт використовується САПР *Quartus* II *Version* 9.1 *Build* 222 10/21/2009 SJ *Full Version.*

Після запуску з'явиться головне вікно , яке має декілька робочих зон (рис. 1.2).

2. Створіть новий проект за допомогою майстра **New Project Wizard**.

Послідовність створенная нового проекту:

2.1 Створіть в робочій директорії жорсткого диска папку з назвою проекту, наприклад ***lab1_Name_SM***, інакше майстер запитає про створення нової папки на наступному кроці (***Name*** – прізвище студента)

2.2. В меню **File** оберіть команду **New Project Wizard** (рис. 1.3). Відкриється вікно майстра. При першому запуску відображається вікно заставки (**Introduction**), в якому відображена послідовність дій, що супроводжує відкриття нового проекту. Якщо на екрані з'явиться сторінка **Introduction**, натисніть кнопку **Next.**

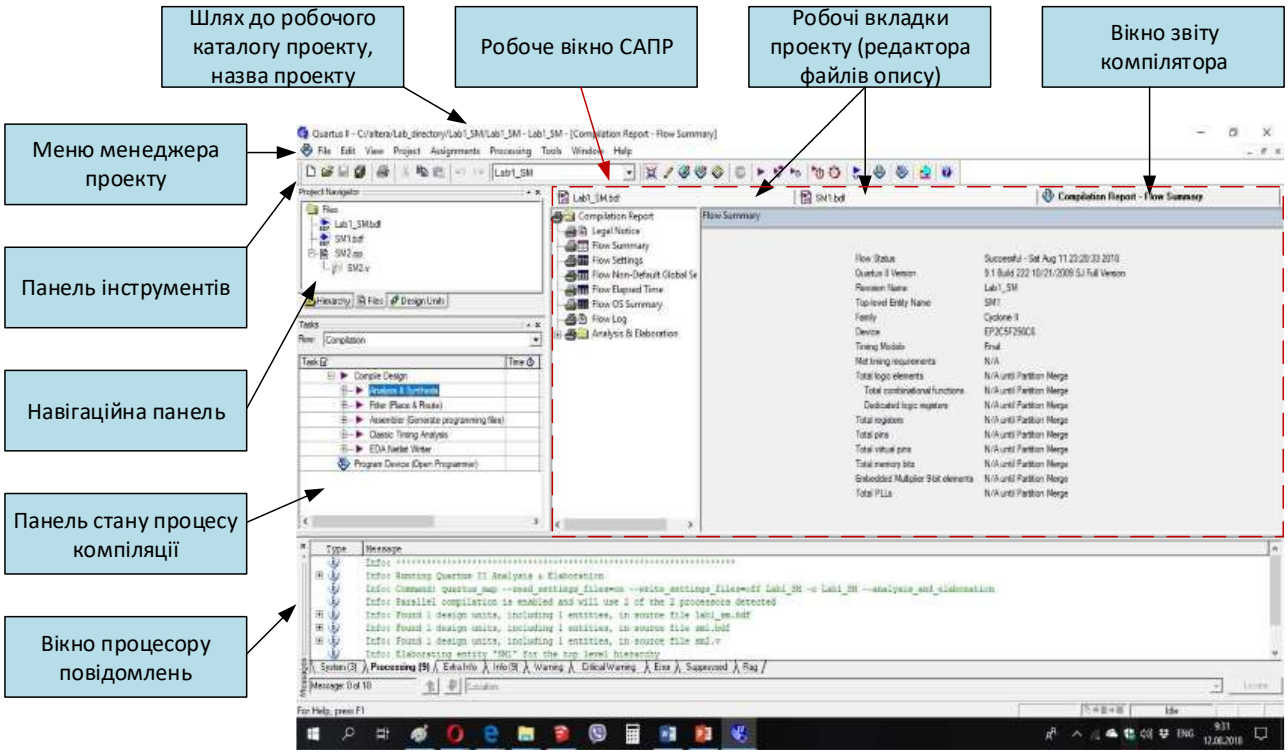


Рис. 1.2. Головне вікно середовища *Quartus* II.

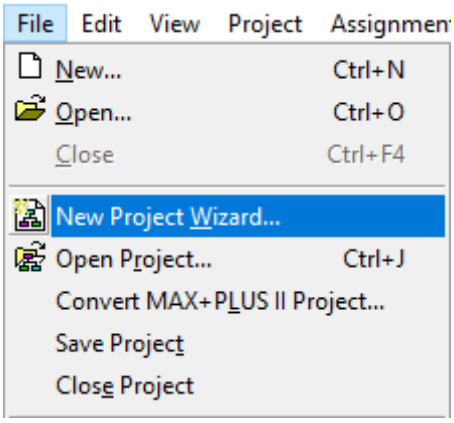


Рис. 1.3. Визивання майстра **New Project Wizard**

2.3. На першій сторінці майстра створення проекту (рис. 1.4) введіть наступну інформацію в текстові поля (приклад заповнення наведено в табл. 1):

Таблиця 1.1

Приклад для налаштування першої сторінки
майстра **New Project Wizard**

Робочій каталог проекту	***..\Lab_directory\Lab1_Name_SM***
Ім’я проекту	***Lab1_Name_SM***
Ім’я об’єкту верхнього рівня розробки (файлу проекту)	***Lab1_Name_SM***

• Назва робочого каталогу проекту. Каталог буде містити всі робочі файли проекту та інші файли, що пов'язані з даним проектом. Якщо каталог раніше не був створений, *Quartus* II створить його автоматично. Якщо каталог вже був створений необхідно вказати шлях до нього, натиснувши на піктограму ... праворуч від текстового поля (рис. 1.4). Введене ім'я автоматично відображається в інших текстових полях вікна;

• Ім'я проекту.

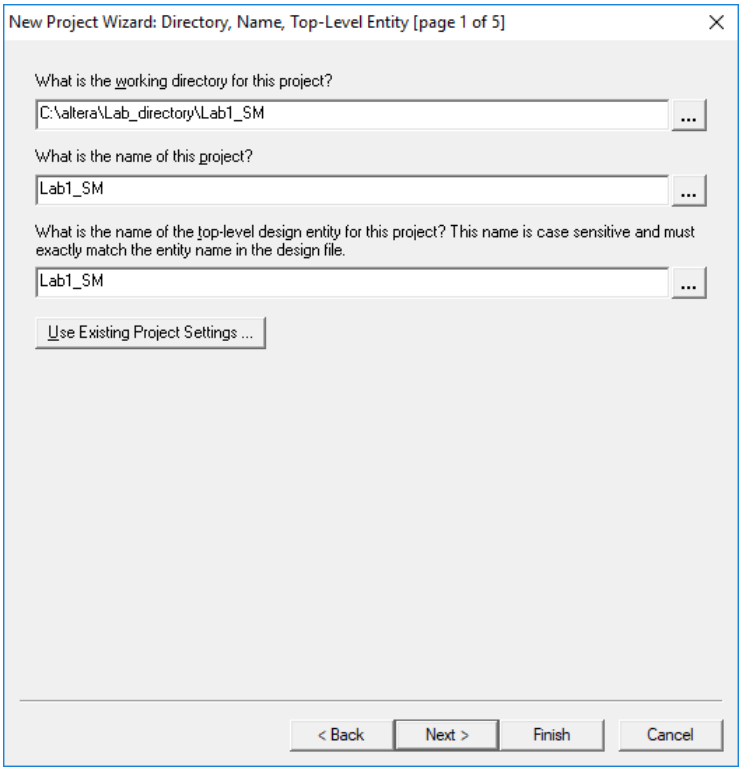


Рис. 1.4. Приклад налаштування першої сторінки майстра **New Project Wizard**

• Ім'я файлу верхнього рівня проекту. *Quartus* II автоматично створює установку компіляції та моделювання для файлу верхнього рівня проекту (головного файлу проекту) з зазначеним ім'ям. Задане ім'я проекту автоматично присвоюється головному файлу проекту в відповідному текстовому полі вікна. Припустимо задавати імена, відмінні від імені файлу проекту, але не рекомендується. Після створення проекту Ви зможете додати інші файли проекту і створити для них установки компіляції та моделювання за допомогою меню **Processing**.

Для переходу в наступне вікно натисніть кнопку ***Next***;

2.4. Друге вікно майстра створення проекту (рис. 1.5) призначене для підключення до нового проекту раніше створених файлів. Для цього натисніть піктограму ... і оберіть файл, який Ви хочете додати до Вашого проекту. Натисніть кнопку ***Open***, після чого кнопку ***Add***. Якщо Ви хочете підключити до проекту декілька файлів, виберіть їх імена і натисніть кнопку ***Add All***, щоб додати їх в робочий каталог проекту.

Файл верхнього рівня не треба додавати до проекту, цей файл знаходиться в робочій директорії проекту і автоматично доданий до проекту майстром на попередньому кроці. Додавання файлів і каталогів на цьому кроці виконується, якщо файли не знаходяться в робочій директорії проекту, були створені раніш, наприклад у складі інших проектів. Якщо Ви не маєте файлів проекту в інших каталогах або файлів, ім'я яких не співпадають з імені проекту, то додавати файли у цьому вікні не обов'язково.

Слід звернути увагу, що до проекту додаються лише шляхи до приєднаних файлів, фізично файли знаходяться в своїх вихідних директоріях.

Окрім файлів на другій сторінці можна додати бібліотеки спеціалізованих функцій, вказавши шлях до них. Це блоки, або функції, які були розроблені у складі інших проектів або сторонніми розробниками.

Під час виконання даної лабораторної роботи непотрібно підключати додаткові файли і бібліотеки, таким чином, для завершення роботи з другим вікном натисніть кнопку ***Next***.

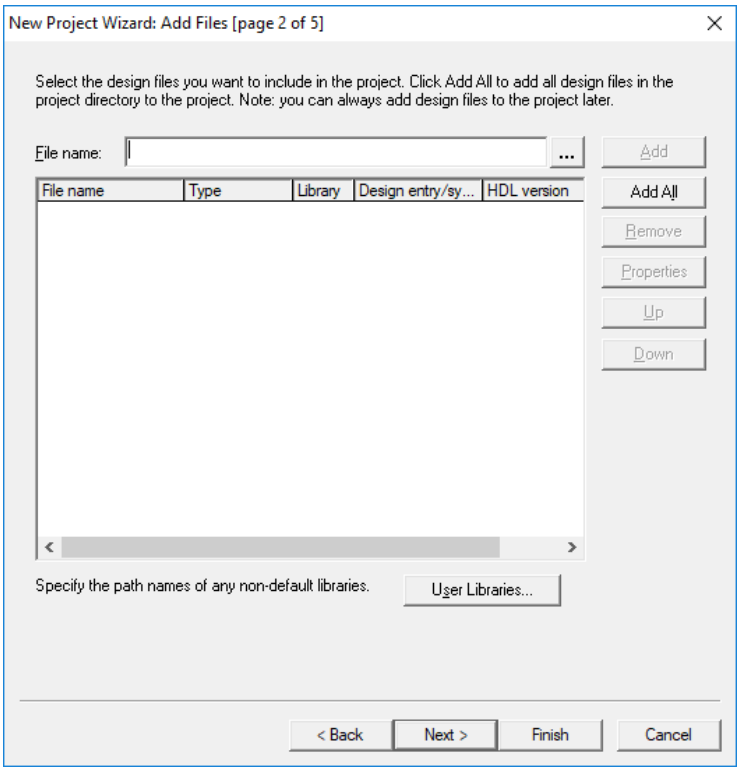


Рис.1.5. Налаштування другої сторінки майстра **New Project Wizard**

2.5. Третє вікно призначене для вибору сімейства і типу мікросхеми ПЛІС, які будуть використані під час компіляції проекту (рис.1.6).

Виберіть сімейство (**Family**) ***Cyclone* II**.

Для обмеження списку доступних мікросхем, можна використати спеціальні фільтри, якщо Вам відомі певні параметри мікросхеми. Наприклад, у частині вікна, що розміщена праворуч зверху, в розділі **Show in 'Available device' list**, встановіть наступні значення: в рядку **Package** оберіть **FBGA**, в рядку **Pin count** (кількість пінів) – **256**, **Speed grade** (швидкодія) – **Fastest**. У вікні **Available devices**

оберіть мікросхему **EP2C5F256C6**. Натисніть **Next** для переходу на наступну сторінку.

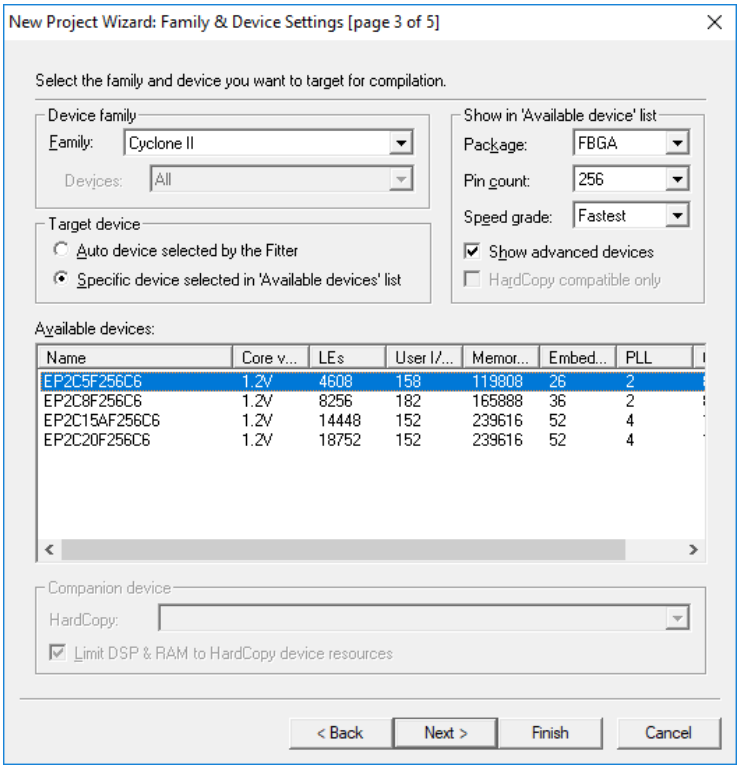


Рис. 1.6. Вибір типу мікросхеми

2.6. Четверте вікно майстра дозволяє указати САПР Quartus II додатково використовувані САПР сторонніх виробників під час створення даного проекту (рис. 1.7). Під час виконання лабораторних робіт використовується тільки САПР Quartus II, тому пропустить цей крок, натиснувши кнопку **Next** для продовження.

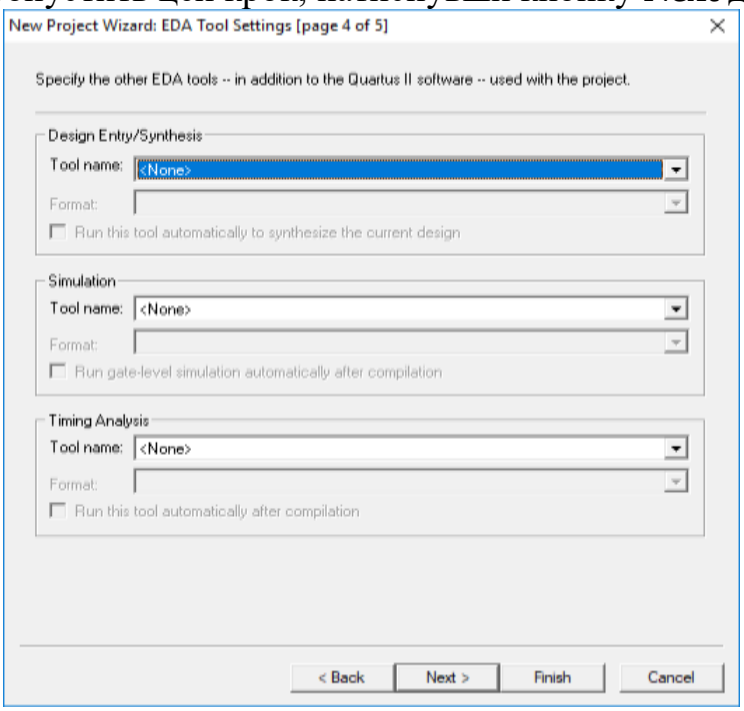


Рис. 1.7. Налаштування четвертої сторінки майстра **New Project Wizard**

2.7. П'ята сторінка містить повну інформацію про зроблені призначення. При необхідності внесення виправлень, можна повернутися до попередніх сторінок, натиснувши кнопку **Back**.

На рис 1.8. представлена підсумкова сторінка. Натисніть кнопку **Finish**. Проект створений.

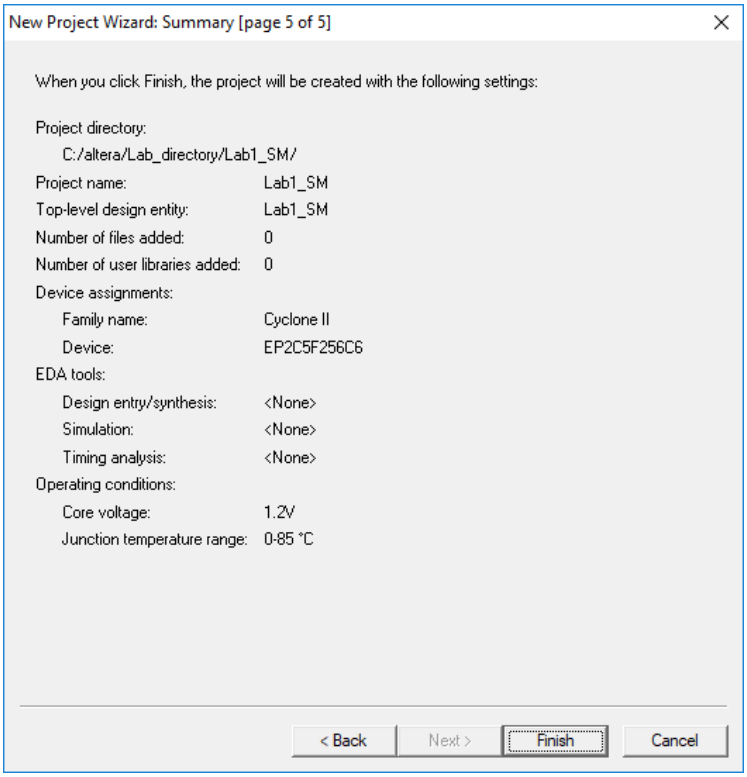


Рис. 1.8. Підсумкова сторінка роботи майстра створення проектів.

2.8. Після закінчення роботи з майстром **New Project Wizard** в основному вікні програми **Project Navigator** в рядку **Hierarchies** з'явиться імена використовуваного сімейства ПЛІС і головний файл проекту. Одночасно у верхній частині основного вікна проекту з'являться шлях до проекту, назва проекту і файлу проекту (рис. 1.10)

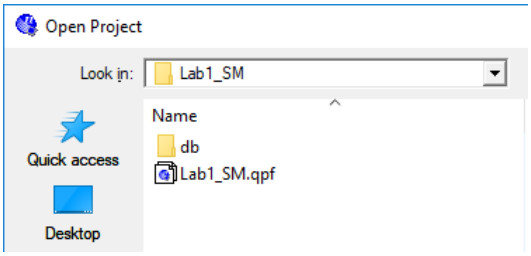


Рис. 1.9. Відкривання існуючого проекту

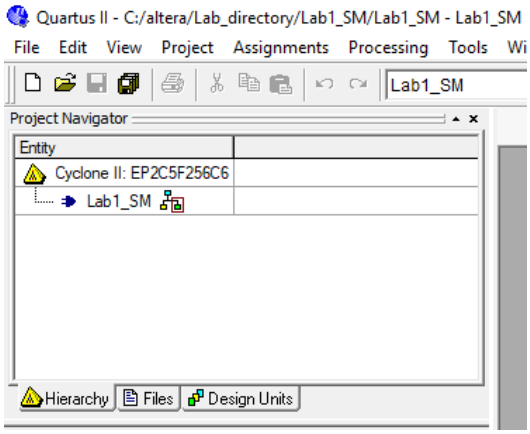
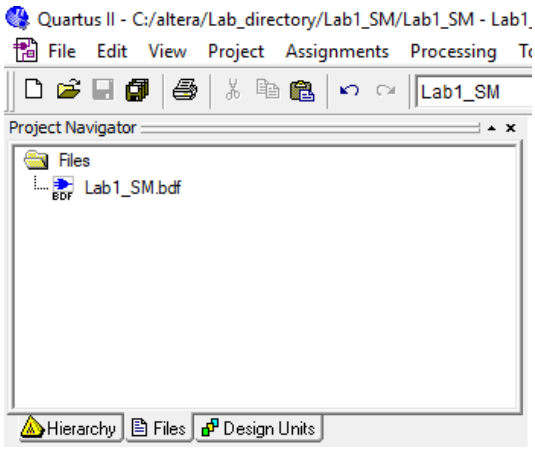


Рис. 1.10. Навігаційна панель після створення нового проекту



Частина 2. Розроблення функціональних блоків комбінаційних схем

Синтезувати комбінаційні схеми за варіантами завдання до лабораторної роботи 1 курсу «Комп'ютерна логіка-1». Етапи синтезу комбінаційних схем розглянуті в теоретичній частині до лабораторної роботи.

3. Створіть файл верхнього рівня опису проекту. Для чого виконайте наступну послідовність дій:

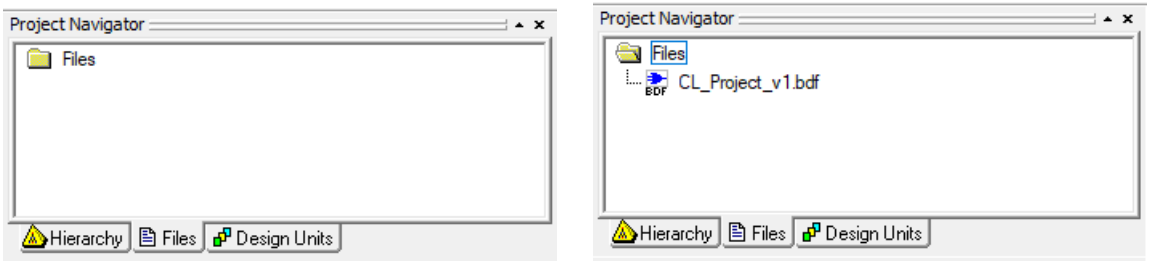
Файли опису проекту (***Design File***) описують логіку проекту і поєднані між собою ієрархічними зв'язками. Файли опису проекту створюються користувачем в графічному або текстовому редакторі. Файли створені в графічному редакторі зберігаються користувачем в директорії проекту з розширення ***.bdf***. Завжди існує файл верхнього рівня ієрархії опису проекту, який не слід путати з головним файлом проекту. Назву файлу верхнього рівня ієрархії опису проекту задають під час процедури створення проекту, але сам файл створюється користувачем на початку проектування, після чого йому надається вже задане ім'я. Файл проекту і файли з налаштуваннями проекту не можуть бути файлами верхнього рівня опису ієрархії проекту.

Якщо виникає необхідність інший файл призначити головним файлом опису проекту, слід в навігаційній панелі натиснути правою кнопкою миші на назві файлу і вибрати команду **Set as Top-Level Entity File**. Таким чином інший користувацький файл буде призначений головним файлом опису проекту.

Файл верхнього рівня опису проекту містить блок-схему верхнього рівня проекту – це блок, або декілька блоків, виводи яких приєднуються до виводів мікросхеми. Кожен блок може містити певну кількість вкладених блоків, які в свою чергу можуть мати ще вкладені блоки. Для опису вкладених блоків створюють файли опису проекту наступних рівнів ієрархії. Такий принцип опису проекту називається *структурним описом*.

3.1. В даному циклі лабораторних робіт для вводу проекту в САПР використовуємо графічний редактор. Якщо на навігаційній панелі безпосередньо

після створення проекту вибрати вкладку **Files**, ми побачимо, що у складі проекту немає жодного файлу опису проекту (рис. 1.11, а).



а б

Рис. 1.11. Вкладка **Files**: а - після створення проекту; б – після створення файлу опису верхнього рівня ієрархії проекту

Для створення файлу опису проекту верхнього рівня (оберіть в меню **File** команду **New**, відкриється вкладка **New**, де виберіть тип створюваного файлу (рис. 1.12). Необхідно вибрати вкладку **Design Files** (Файли опису проекту) і у списку, що випадає, обрати рядок **Block Diagram/Schematic File** – графічний файл. Після натискання кнопки **OK** відкриється вікно графічного редактора в головному вікні програми (рис. 1.13). В робочому полі графічного редактора вводять блок-схеми пристроїв та вузлів проекту. При створенні нового файлу проекту в графічному редакторі цей файл отримує ім’я за замовчанням ***Block.bdf*** (рис. 1.13). Збережіть створений файл опису проекту в дирекоторії проекту з ім’ям файлу проекту – ***CL_Name_Project_v1.bdf*** (***Name*** – прізвище студента). Під час першого збереження файлу опису САПР запропонує обрати саме це ім’я, як ім’я файлу верхнього рівня ієрархії опису проекту.

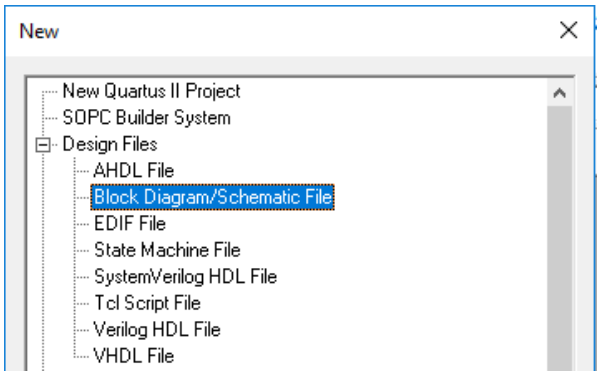


Рис. 1.12. Вікно вибору типу нового файлу проекту

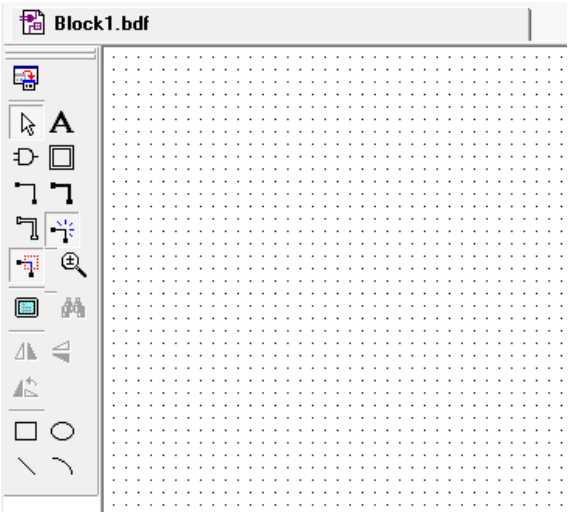


Рис. 1.13. Фрагмент вікна графічного редактора

4. В робочому полі головного вікна введіть блок схему проекту, для цього використайте панель інструментів, призначення яких наведено на рис. 1.14.

Назва інструменту	Назва інструменту
– від'єднання робочого вікна;	– масштабування об'єктів;
– виділення;	– відображення робочого вікна на весь екран;
–введення тексту;	– пошук об'єкту;
–введення стандартних компонент із бібліотеки	– коригування відображення об'єкту;
–створення блоку;	
–створення провідників;	
– створення шин;	– малювання прямокутника;
– створення каналів зв'язку;	– малювання еліпса;
– автоматичне під'єднання провідників;	– малювання прямих ліній;
	– малювання дуг.

Рис. 1.14. Призначення піктограм панелі інструментів графічного редактора

4.1 Виберіть піктограму «Створення блоку» (**Block Tool**) для створення робочого блоку функціонального елементу – для даної лабораторної роботи це функціональний блок (ФБ), що реалізує певну логічну функцію. Натиснути на білій частині поля проекту ліву кнопку миші, і накреслити прямокутну область. Можна використовувати команди відміни (**Undo**) і повторення (**Redo**) дій із меню **Edit**. Вигляд створеного блоку показаний на рис. 1.15.

4.2. Кожному блоку та елементу на схемі надайте ім'я. Виберіть піктограму «Виділення» (**Selection Tool**) і клацніть правою кнопкою миші на створеному блоці. Відкриється діалогове вікно «Властивості блоку» (**Block Properties**), де на вкладці **General** в полі **Name** введіть ім'я ФБ, наприклад, ***LC1*** (*Logic Circuit*) (рис. 1.16). Поле **Instance name** можна залишити без змін, це внутрішній ідентифікатор елементу в проекті.

На вкладці **I/Os** призначте вхідні і вихідні сигнали блоку ***LC1*** (рис. 1.17) – в полі **Name** задайте ім'я сигналу, а в полі **Type** його тип: вхідний (**INPUT**), вихідний (**OUTPUT**) або двоспрямований (**BIDIR**). Після заповнення текстових полів натисніть кнопку **Add** і перейдіть до внесення наступного сигналу. Після внесення всіх сигналів натисніть кнопку **OK**. Налагоджений блок зображений на рис. 1.18.

> Для позначення провідника в полі **Name** вікна **Block Properties** достатньо вказати лише ім'я сигналу, для позначення шини в квадратних дужках слід вказати її розрядність, наприклад **Data[3..0]**.

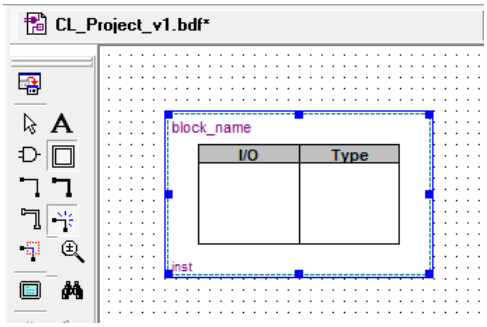


Рис. 1.15. Створення нового блоку

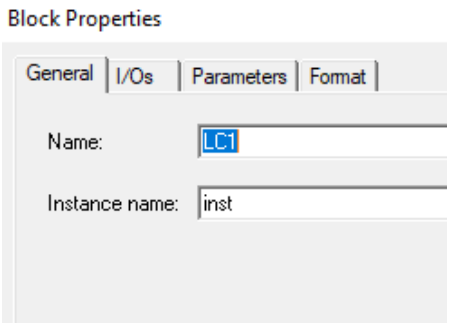


Рис. 1.16. Призначення ім’я новому блоку

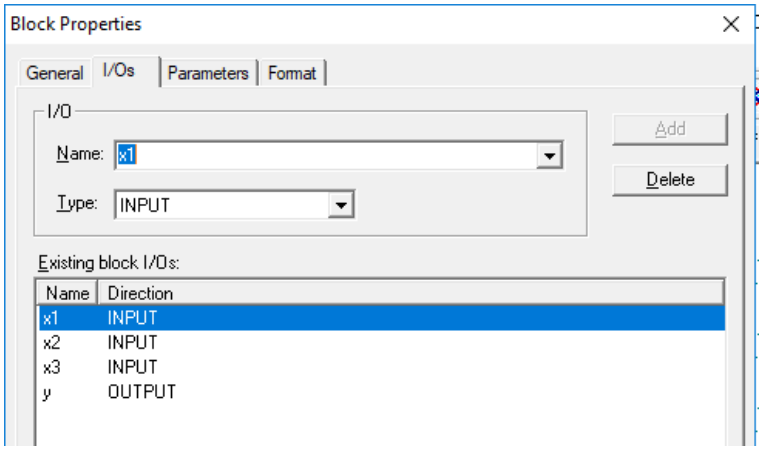


Рис. 1.17. Призначення сигналів введення/ виведення

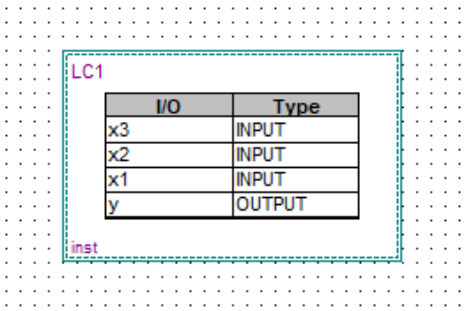


Рис. 1.18. Налагоджений блок

4.3. Створіть вхідні та вихідні контакти - піни (**Pins**) функціонального блоку.

> *Примітив Quartus* це найдрібніша одиниця схеми, яка реалізована апаратно на кристалі FPGA. Будь яка схема, що введена користувачем в САПР, спеціальними засобами САПР буде перетворена у примітиви, які в подальшому будуть розміщені в комірки мікросхеми.

Контакти введення/виведення (**Pins**) є бібліотечними компонентами, які знаходяться в бібліотеці стандартних компонентів *Quartus* II в розділі ***../libraries/primitives/pin*** (рис. 1.19).

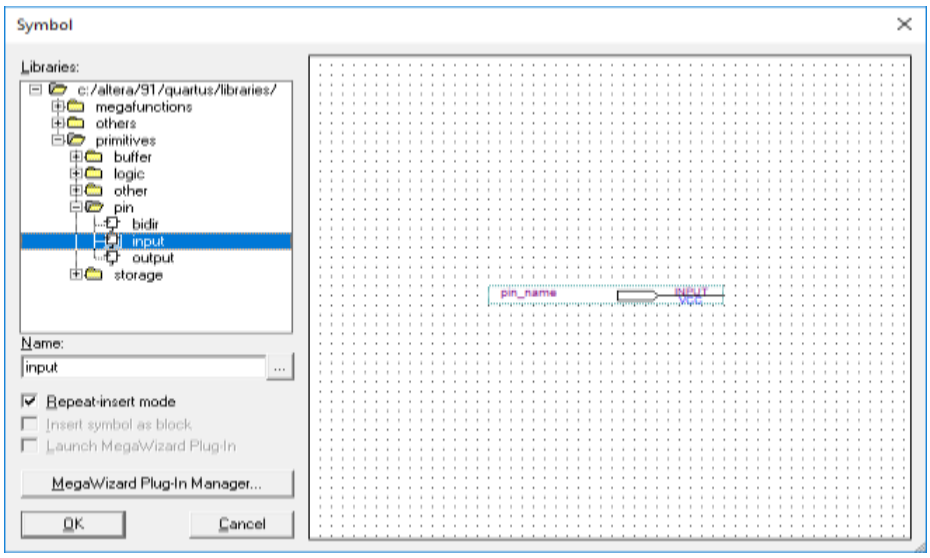


Рис. 1.19. Вікно інструменту для додавання бібліотечних символів функціональних елементів

Виберіть необхідний елемент із списку (для виконання завдання необхідно вхідні контакти (**Input**) для вводу аргументів і вихідні (**Output**) для виводу результату). Зображення вибраного елементу відобразиться в головному полі вікна **Symbol** (рис. 1.19). Натисніть кнопку ОК. Після натискання кнопки ОК вікно **Symbol** закривається, за натисканням у робочому полі редактора вибраний елемент відобразиться в визначеному місці. Встановлення прапорцю **Repeat-Insert Mode** дозволяє розмістити в робочому полі редактора декілька вибраних бібліотечних компонентів. Для відкріплення компонента від курсора миші натисніть правої кнопкою миші на робочому полі і виберіть команду **Cansel** в контекстному меню.

Розмістить необхідну кількість контактів на робочому полі. Додайте три вхідних контакти (**Input**) для вводу аргументів і один вихідний (**Output**) для виводу результату виконання логічної функції. Результат додавання контактів зображений на рис. 1.20.

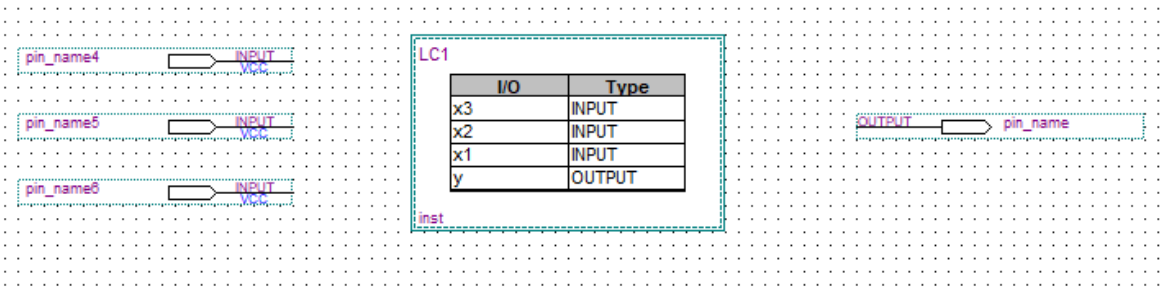


Рис. 1.20. Додавання контактів введення/виведення

4.4. Призначте ім’я кожному контакту. Для цього натисніть правою кнопкою миші на одному з контактів. Оберіть команду **Properties** із контекстного меню. У вікні, що відкрилося, на вкладці **General** введіть ім’я контакту в текстовому полі **Name** (рис. 1.21).

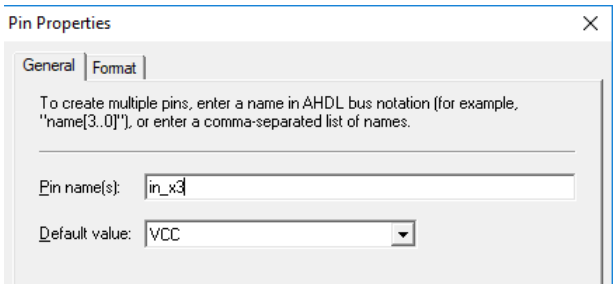


Рис. 1.21. Інструмент для призначення імені контакту

В полі **Default value** задається початкове значення сигналу: логічна одиниця **VCC** (встановлено за замовченням) або логічний нуль (**GND**). У випадку багаторозрядних контактів слід вводити ім'я контакту в форматі шини – із вказанням розрядності контакту в квадратних дужках, наприклад, **name[3..0]** або безпосередньо список всіх контактів.

Призначте ім’я всім контактам функціонального блоку згідно табл. 1.3. Результат перейменування контактів представлений на рис. 1.22.

Таблиця 1.3.

Приклад призначення імен контактам функціонального блоку

Ім’я контакту (*Name*)	**Тип контакту (*Type*)**	**Коментар**
in_x3	INPUT	Аргументи логічної функції
in_x2	INPUT	
in_x1	INPUT	
out_y	OUTPUT	Результат логічної функції

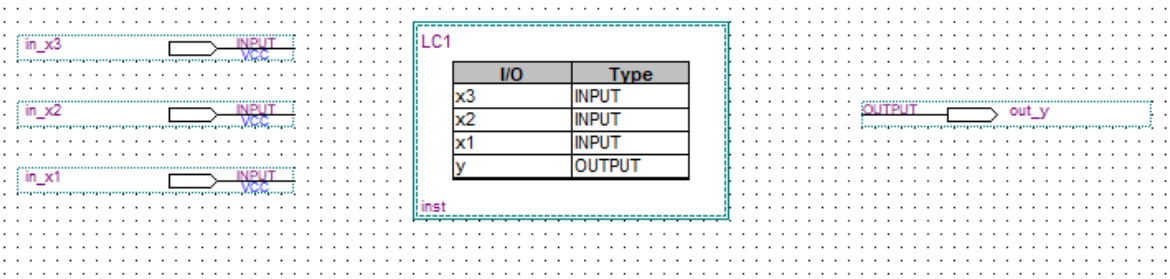


Рис. 1.22. Призначення імені контактам введення/виведення

4.5. Додайте зв’язки між функціональним блоком і контактами. Для створення провідників натисніть на піктограму визивання інструмента для створення провідників (**Node Tool**). Для під’єднання вхідних контактів до входів функціонального блоку використайте інструмент **Conduit** (Канал зв’язку), або просто під’єднайте вхідний контакт до входу блоку, після чого відтягніть його – канал зв’язку створиться автоматично. Результат з’єднання зображено на рис. 1.23. Всім провідникам і шинам (**Node** і **Bus**) задайте ім’я. Для цього відкрийте вікно **Properties**, із контекстного меню кожної шини. Рекомендовано, щоб назва

провідника/шини співпадали з назвою сигналу на виводі блоку (необов'язково).

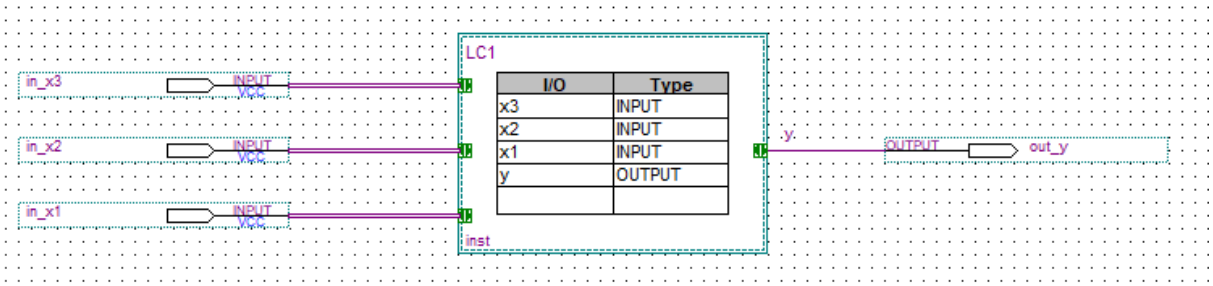


Рис. 1.23. Приклад створення з'єднань між функціональними елементами.

> Для створення шин використайте піктограму інструменту для створення шин (**Bus Tool**).

Створіть карту з'єднань. Для створення карти з'єднань натисніть правою кнопкою миші на одному з зелених прямокутників і місці приєднання провідника до функціонального блоку. Відкривається вікно «Властивості з'єднання» (**Mapper Properties)** (рис. 1.24).

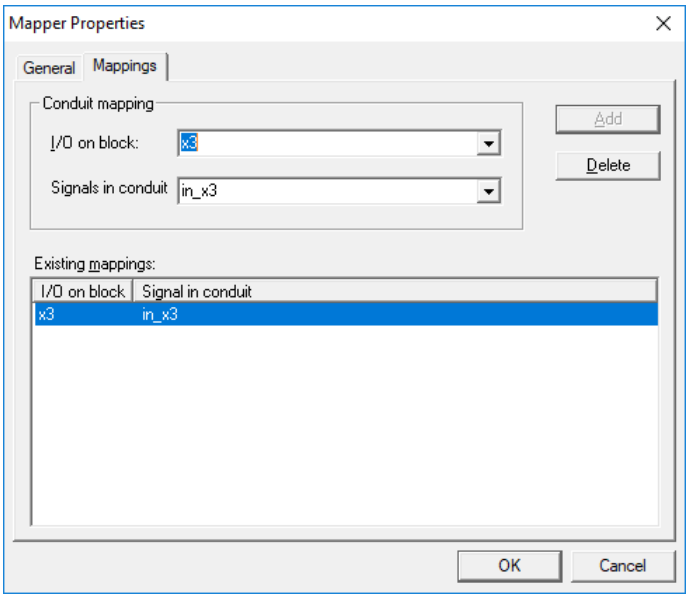


Рис.1.24. Інструмент для створення карти з'єднань

На вкладці **General** виберіть тип виводу блоку із списку **Type** (рис. 1.24). На вкладці Mapping задайте карту з'єднань. Для цього в випадаючому списку «Входи і виходи блоку» (**I/0 on block**) виберіть ім'я відповідного виводу блоку. В випадаючому списку «Сигнали в каналі» (**Signals in conduit**) виберіть ім'я сигналу в лінії зв'язку. Натисніть кнопку **Add**, після чого відповідний запис з'явиться в полі «Наявні з'єднання» (**Existing mappings**) (рис. 1.24). Задайте карти з'єднань для всіх інших виводів функціонального блоку. Результат зображений на рис. 1.25.

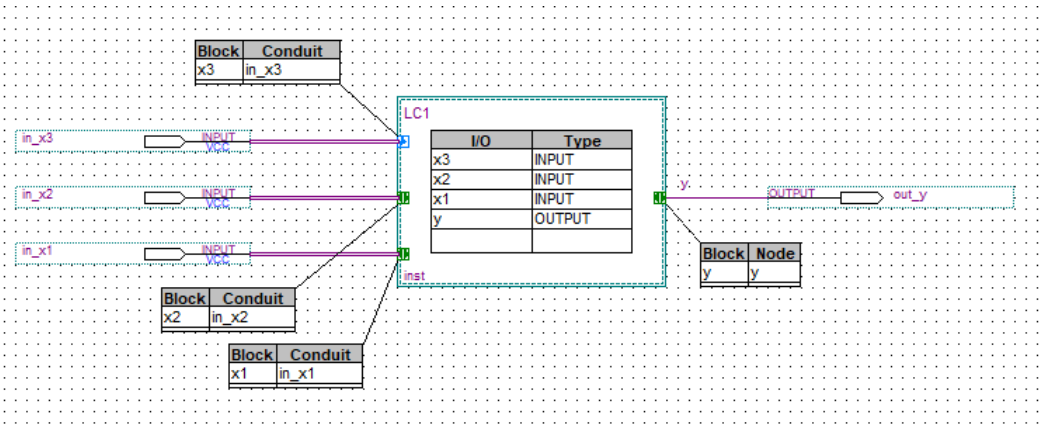


Рис. 1.25. Карти з’єднань

5. Створіть файл опису наступного рівня ієрархії

5.1. Створіть файл опису роботи функціонального блоку ***LC1***. Для цього натисніть правою клавішею мишки на блоці ***LC1*** та в контекстному меню виберіть команду «**Create Design File from Selected Block…**», відкриється вікно створення нового файлу опису виділеного блоку. В списку **File type** виберіть спосіб створення нового файлу, в нашому прикладі, це **Schematic** для створення файлу опису в графічному редакторі (рис. 1.26). Інші пункти списку призначенні для створення файлів опису на відповідних мовах програмування апаратури. Зверніть увагу, що САПР пропонує назву файлу опису відповідно до назви блоку, що описується. Можна задати власне ім’я файлу. Далі відкриється нова вкладка графічного редактора з заданим ім’ям (рис. 1.27). Новий файл з’явиться в навігаційній панелі (рис. 1.28).

Зверніть увагу, що знову створений файл опису містить контакти для приєднання до файлу верхнього рівня ієрархії, які були створені під час створення блоку верхнього рівня (рис. 1.27).

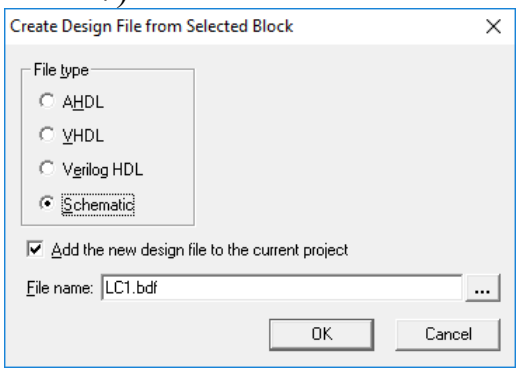
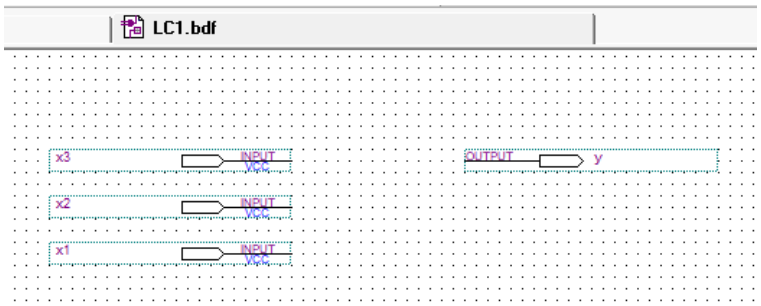


Рис.1.26.



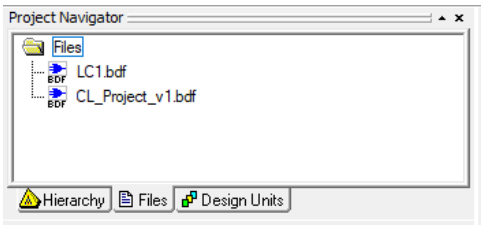
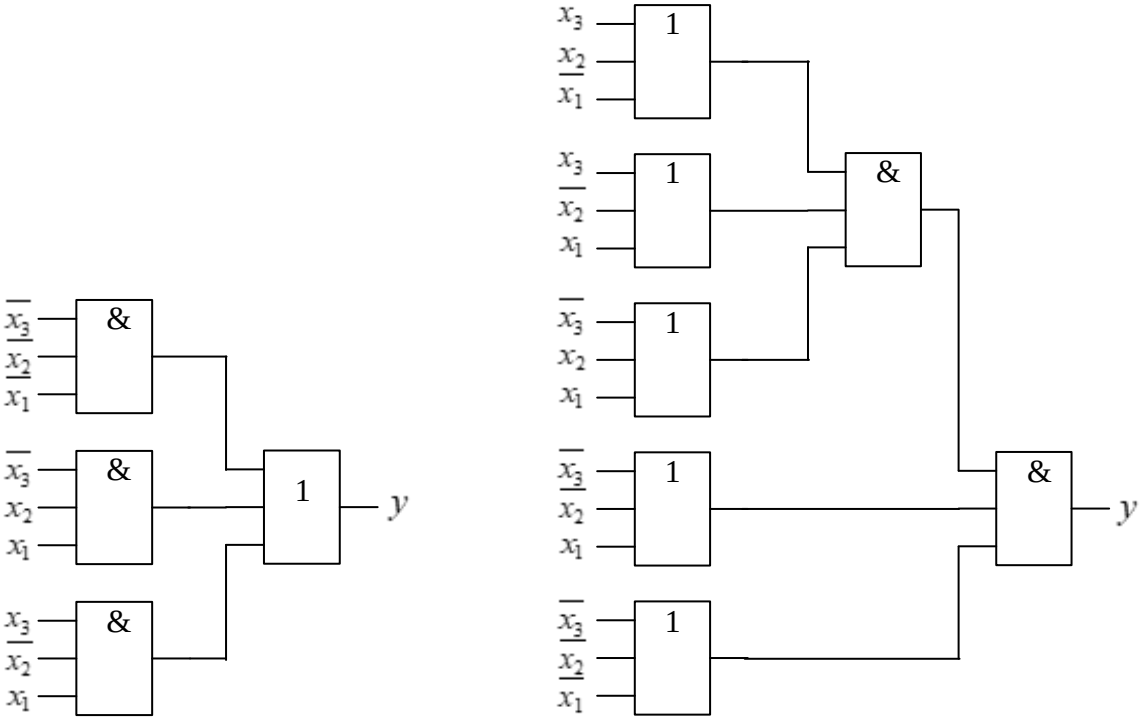


Рис. 1.27. Рис. 1.28.

5.2. Створіть логічну (комбінаційну) схему ДДНФ перемикальної функції *f*4, що розроблена під час виконання теоретичного завдання. Для цього додайте на робоче поле файлу ***LC1.bdf*** елементи логічної схеми – логічні елементи І, АБО, та інші. Приклад комбінаційної схеми в елементному базисі 3І / 3АБО наведено на рис. 1.29, а.



а б

Рис. 1.29. Вихідні комбінаційні схеми: а – ДДНФ перемикальної функції; б – ДКНФ перемикальної функції.

Логічні елементи є бібліотечними компонентами, які знаходяться в бібліотеці стандартних компонентів Quartus II. Для відкривання бібліотеки стандартних компонентів використайте вікно **Symbol**. Логічні елементи є примітивами *Quartus*. Логічні елементи знаходяться в розділі ***../libraries/primitives/logic***. Виберіть необхідні елементи із списку. Для виконання завдання виберіть логічні елементи І та АБО з трьома входами – **3and** та **3or** відповідно. Зображення вибраного елементу відобразиться в головному полі вікна **Symbol**. Після натискання кнопки **OK** вікно **Symbol** закривається, за натисканням у робочому полі редактора вибраний елемент відобразиться в визначеному місці. Розмістить на робочому полі три елемента **3and** і один елемент **3or**. Аналогічним чином розмістить на робочому полі п’ять інверторів – елемент **not** також знаходиться в бібліотеці примітивів в списку ***../libraries/primitives/logic.***

За допомогою інструмента (**Node Tool**) створити провідники між логічними елементами і виводами блоку. В результаті, на робочому полі створена логічна

схема, яка зображена на рис. 1.30.

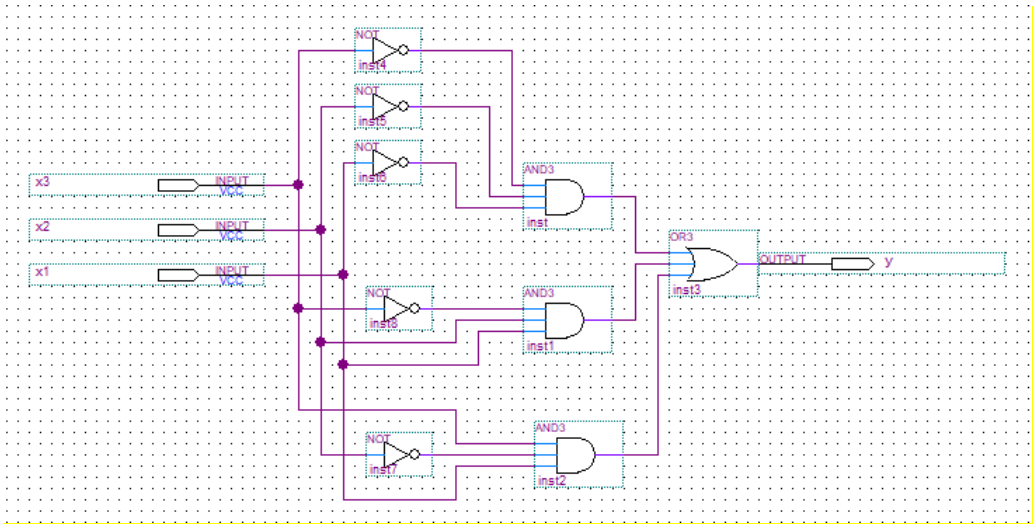


Рис. 1.30. Логічна схема перемикальної функції

Збережіть створену схему. Створений файл можна редагувати, відкривши файл із навігаційної панелі, або із контекстного меню, яке відкривається подвійним натисканням правої кнопки миші на блоці верхнього рівня опису проекту.

Схематично ієрархічний принцип поєднання об’єктів проекту представлений на рис. 1.31.

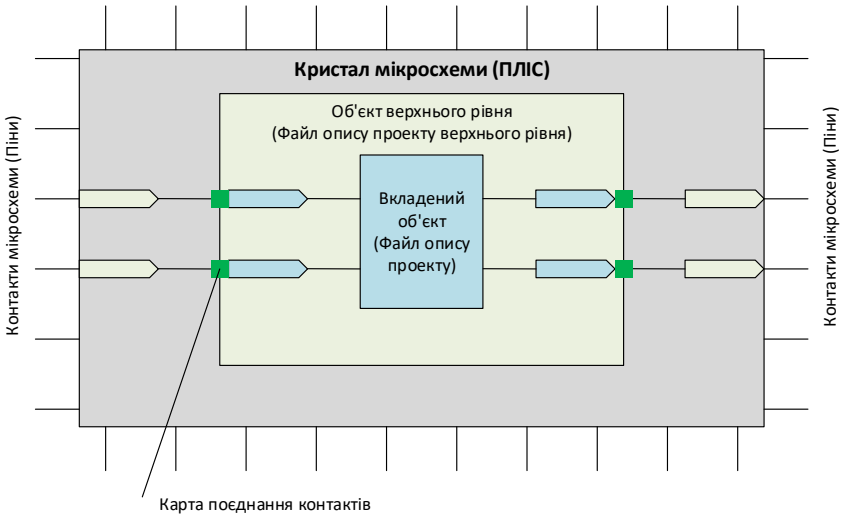


Рис. 1.31. Принцип ієрархічного поєднання об’єктів проекту

6. Збережіть і перевірте проект.

Для перевірки коректності створеної схеми виконайте перший етап компіляції ***Analysis & Elaboration***. Для цього в меню ***Processing*** оберіть команду ***Start ⇒ Start Analysis & Elaboration***.

Натисніть кнопку OK у вікні повідомлення, коли аналіз завершиться. Якщо були виявлені будь-які помилки, перевірте всі зв'язки схеми для виправлення помилок.

7. Виконайте компіляцію проекту

7.1. Якщо перевірка пройшла успішно запустіть повну компіляцію проекту командою ***Start Compilation*** меню ***Processing*** або натисніть піктограму , розташовану на панелі інструментів. Після завершення процесу компіляції відкриється діалогове вікно, в якому натисніть кнопку ***OK***.

7.2. Проаналізуйте інформацію про використані ресурси і часові затримки зі звіту компілятора:

– З розділу ***Flow Summary*** звіту компілятора ***Compilation Report*** (рис. 1.32), випишіть в таблицю (табл. 1.4) наступні значення: ***Total logic elements***, ***total memory bits***, ***total embedded multiplier 9-bit elements і total pins***.

Total logic elements	2 / 4,608 (< 1 %)
Total combinational functions	2 / 4,608 (< 1 %)
Dedicated logic registers	0 / 4,608 (0 %)
Total registers	0
Total pins	5 / 158 (3 %)
Total virtual pins	0
Total memory bits	0 / 119,808 (0 %)
Embedded Multiplier 9-bit elements	0 / 26 (0 %)
Total PLLs	0 / 2 (0 %)

Рис. 1.32. Фрагмент розділу ***Flow Summary*** звіту компілятора ***Compilation Report***

Аналіз результатів показує, що для реалізації проекту були задіяні лише логічні ресурси мікросхеми, спеціалізовані ресурси (тобто вбудована пам'ять, вбудований помножувач) не задіяні.

– Розкрийте папку ***Fitter*** звіту компілятора ***Compilation Report***. Виберіть розділ ***Resource Section***. Із звіту ***Resource Utilization by Entity***, випишіть в таблицю (табл. 1.4) використані ресурси для реалізації функціональних блоків (рис. 1.33).

Fitter Resource Utilization by Entity

	Compilation Hierarchy Node	Logic Cells	Dedicated Logic Registers
1	⊟ |Lab1_SM	2 (1)	0 (0)
2	⊟ |SM2:inst2|	1 (0)	0 (0)
3	⊟ |lpm_add_sub:lpm_add_sub_component|	1 (0)	0 (0)
4	|add_sub_03i:auto_generated|	1 (1)	0 (0)

Рис. 1.33. Фрагмент розділу ***Resource Utilization by Entity*** звіту компілятора ***Compilation Reporttpd***

– Оцініть часові затримки за результатами звіту компілятора ***Timing Analyzer*** із розділів ***Timing Analyzer Summary*** і ***tpd*** (рис. 1.34).

Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Failed Paths
1	Worst-case tpd	N/A	None	9.813 ns	iny	outs	--	--	0
2	Total number of failed paths								0

tpd

	Slack	Required P2P Time	Actual P2P Time	From	To
1	N/A	None	9.813 ns	iny	outs
2	N/A	None	9.806 ns	iny	outp
3	N/A	None	9.041 ns	inz	outs
4	N/A	None	9.027 ns	inz	outp
5	N/A	None	8.834 ns	inx	outs
6	N/A	None	8.827 ns	inx	outp

Рис. 1.34. Аналіз швидкодії на підставі звіту компілятора ***Timing Analyzer***: а – найдовший шлях проходження сигналу; б – всі шляхи проходження сигналів

Таблиця 1.4

Аналіз звіту компілятора

Звіти компілятора	***LC1***	***LC2***
Total logic elements	2 (<1%)	
Total pins	5 (3%)	
LC Block	1 Logic Cells	
Actual Time	9,813 ns	
Найдовший шлях проходження сигналу		
Найкоротший шлях проходження сигналу		
Затримка на логічних елементах	2ns	2ns
Затримка на вхідних контактах		
Затримки на вихідних контактах		
Кількість задіяних логічних комірок		
Додатково		

8. Виконайте симуляцію проекту ***CL_Name_Project_v1.***

8.1. Симуляцію починають зі створення стимуляційного файлу (**Vector Wave Form File**) – ***Test Bench*** файл, що містить стимули для моделювання. Для цього оберіть команду **New** меню **File**. В розділі **Verification/Debugging Files** оберіть тип файлу **Vector Wave Form File**. (рис. 1.35). Натисніть кнопку ОК. В головному вікні з'явиться вкладка з назвою файлу за замовчанням ***Waveform.wvf***. На вкладці знаходиться інструмент для створення часової діаграми для моделювання роботи проекту. Збережіть новий файл з ім'ям проекту ***CL_Name_Project_v1.vwf***.

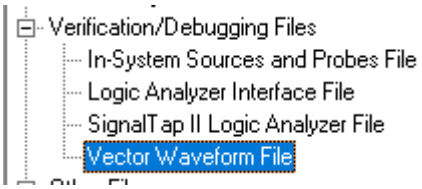


Рис. 1.35. Створення стимуляційного файлу **Vector Wave Form File**

8.2. Додайте вхідні сигнали для моделювання роботи схеми. Для цього в полі **Name** натисніть два рази лівою кнопкою миші. Відкриється вікно **Insert Node or Bus** (рис. 1.36). Якщо назви сигналів відомі введіть їх в полі **Name**.

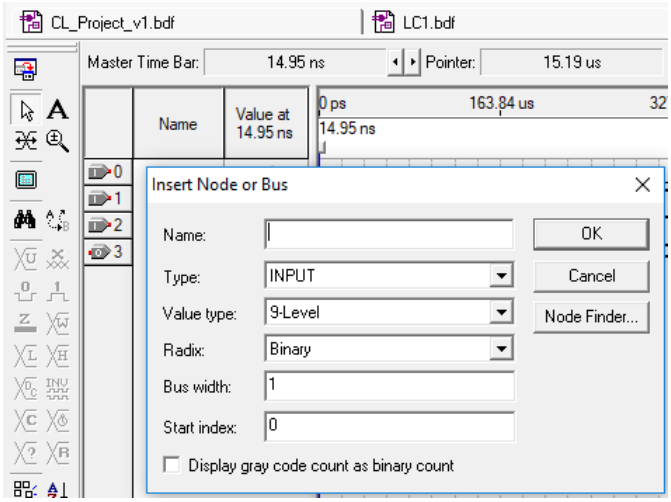
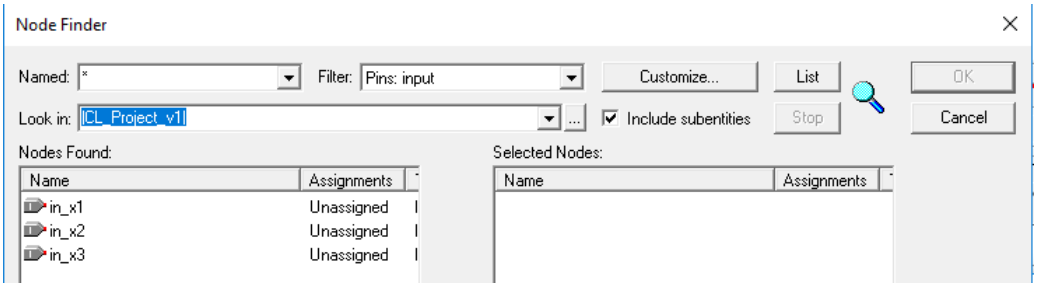


Рис. 1.36. Інструмент **Insert Node or Bus** для додавання сигналів для моделювання.

Інакше скористайтеся інструментом **Node Finder** для пошуку сигналів. Для цього натисніть кнопку **Node Finder** у вікні **Insert Node or Bus**, відкриється відповідне вікно **Node Finder** (рис. 1.37). Скористайтеся фільтром (**Filter**) для вибору групи сигналів (**Pins: input**). Натисніть кнопку **List**. Перелік необхідних вхідних сигналів відкриється в полі **Nodes Found**. Виберіть необхідні сигнали і перенесіть їх у поле **Selected Nodes** за допомогою кнопки >, або всі сигнали за допомогою кнопки >>. Натисніть кнопку **OK**.

У вікні **Insert Node or Bus** можна встановити формат вхідних та вихідних сигналів, для чого із переліку **Radix** виберіть, наприклад **Binary** (двійковий формат), після чого натисніть кнопку **OK**. У робочому полі часової діаграми з'являться вхідні сигнали для моделювання (рис. 1.38, а). Аналогічно додайте вихідні сигнали (**Pins: output**), після чого в робочому вікні ***CL_Name_Project_v1.vwf*** з'являться і вихідні сигнали (рис. 1.38, б).



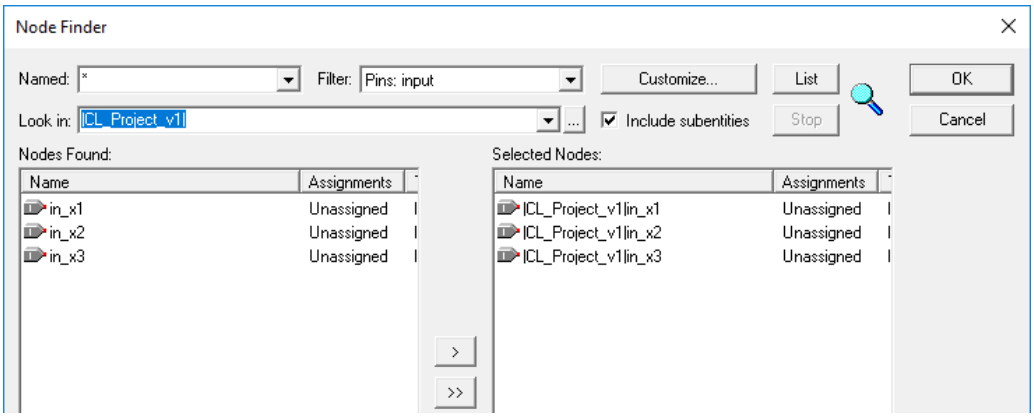
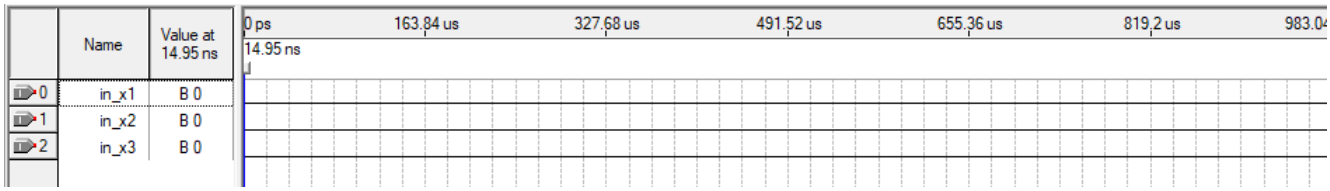
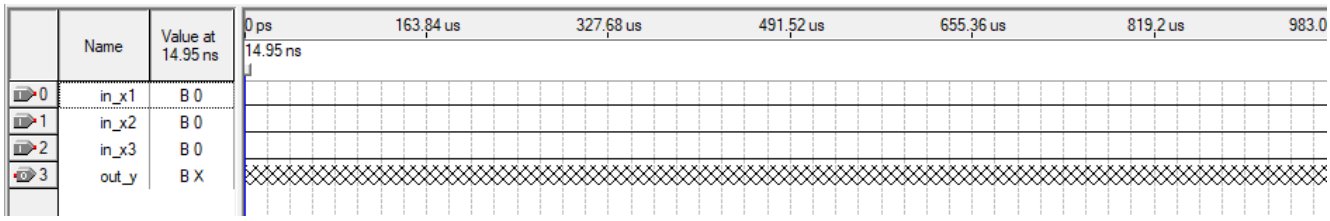


Рис. 1.37. Інструмент **Node Finder** для пошуку сигналів для моделювання



а



б

Рис. 1.38. Вхідні та вихідні сигнали для моделювання роботи схеми

Формат виводу сигналів на діаграмі можна змінити в будь який момент після формування діаграми, а також і під час перегляду результатів модулювання. Для цього слід вибрати необхідний сигнал в полі **Name** і натиснути правою кнопкою миші на назві сигналу або на виділеній ділянці. У вікні **Node Properties**, яке відкриється, із переліку **Radix** слід вибрати необхідний формат сигналу, після чого натиснути кнопку **OK**. Зручними для перегляду результатів моделювання є формати **Unsigned Decimal** –десяткове число без знаку і **Signed Decimal** – десяткове зі знаком.

8.3. Задайте інтервал для моделювання, для цього скористайтеся командою **End Time** із головного меню **Edit**. У вікні, що відкрилося, у полі **Time** задайте значення 1***ms*** (**s** – секунда, **m** – мілі, **u** – мікро, **n** – нано, **p** – піко). Натисніть кнопку ОК. Загальний інтервал для моделювання встановиться в 1***ms***. За допомогою інструменту можна зменшувати або збільшувати масштаб часової діаграми: натискання лівої кнопки миші збільшує зображення, правої – зменшує.

8.4. Задайте значення вхідних сигналів для моделювання. Для цього оберіть інструмент і виберіть необхідний сигнал, натиснувши на його імені в полі **Name**. Для завдання значень сигналів використайте піктограми на панелі інструментів.

> Якщо треба встановити весь діапазон часу модулювання в визначене значення слід натиснути на імені сигналу в полі **Name**. Для встановлення виділеного діапазону в значення логічної 1 або логічного 0 використовують піктограми та відповідно. Для встановлення значень сигналів на визначеному проміжку часу використовують інструмент **Arbitrary Value**, який визивається піктограмою . У вікні, що відкривається, можна задати точний проміжок часу і значення сигналу в необхідному форматі.

На часовій діаграмі для сигналу ***in_x3*** встановіть значення логічної 1 на проміжку часу тривалістю від 40*ns* до 80 *ns*. Для цього скористуйтеся інструментом **Arbitrary Value**. Для цього натисніть піктограму , у вікні, що відкрилося, у полі ***Start Time*** задайте значення 40*ns*, а у полі ***End Time*** – 80 *ns*. У полі ***Radix*** оберіть значення ***Binary***, у полі ***Numeric or Name Value*** задайте значення логічної 1. Натисніть кнопку **OK**.

> Для завдання регулярних послідовностей сигналів використовують інструмент - **Count Value** на панелі інструментів або в контекстному меню, що випадає під час натискання правої кнопки миші на виділеному полі часової діаграми. Для цього в контекстному меню слід вибрати команду **Count Value** в списку команд **Value**. У вікні, що відкривається на вкладці **Counting** слід задати початкове і наступне значення послідовності, а на вкладці **Timing** слід задати початок періодичної послідовності – **Start Time**, кінець періодичної послідовності – **End Time** і тривалість кожного проміжку переключення сигналу – **Count every**.

За допомогою інструменту **Count Value** на проміжку часу від 0*ns* до 80*ns* задайте періодичні значення двійкових сигналів від 0 до 1 на кожних 20*ns* лінії сигналу ***in_x2*** і на кожних 10*ns* лінії сигналу ***in_x1***. (рис. 1.39). Отримана часова діаграма зображена на рис. 1.40. Збережіть файл ***CL_Name_Project_v1.vwf***.

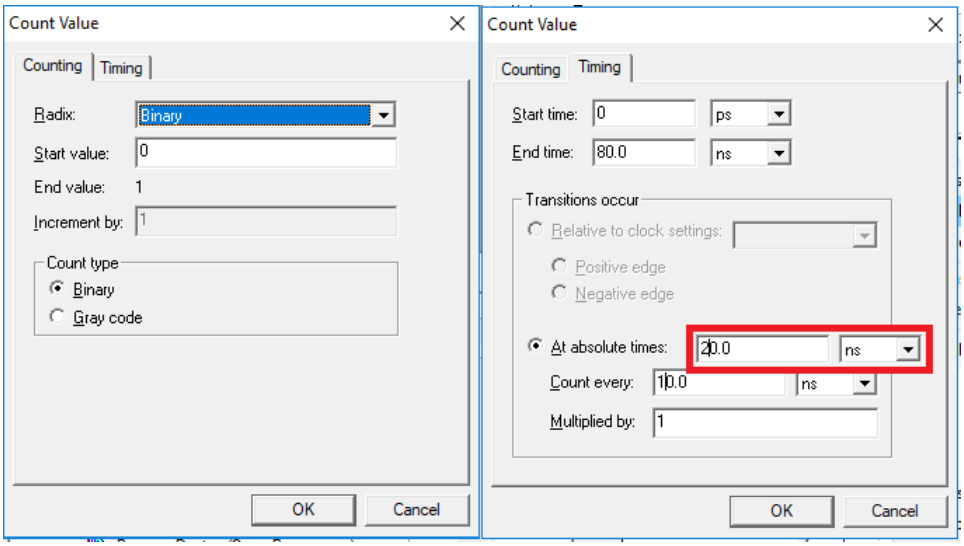


Рис. 1.39. Встановлення періодичних значень вхідних сигналів.

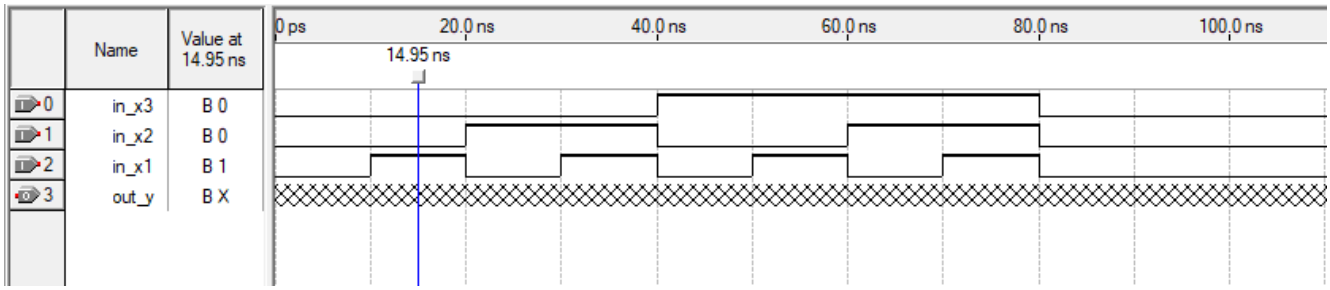


Рис. 1.40. Встановлення вхідних сигналів для моделювання на часовій діаграмі

8.5. Для запуску симуляції використайте піктограму в панелі задач, або виберіть команду **Start Simalation** в меню **Processing**. За успішно виконаного процесу симуляції відкривається відповідне повідомлення і відкривається вікно звіту симулятора **Simalation Report**. Результат виконаної симуляції проекту ***CL_Name_Project_v1*** представлений на рис. 1.41.

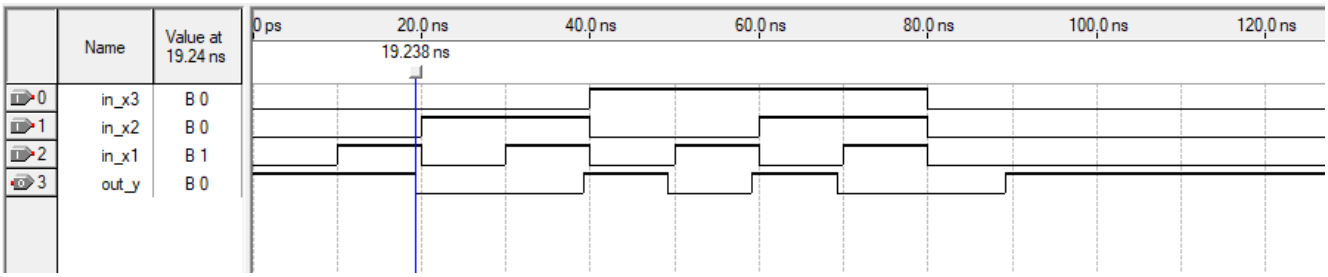
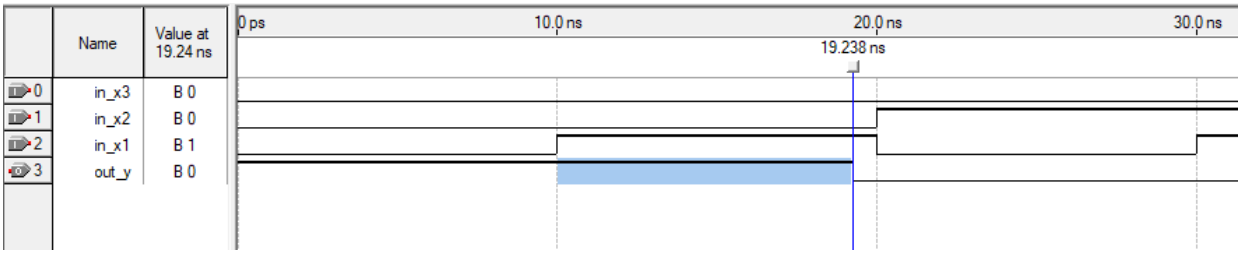


Рис. 1.41. Результати симуляції



1.42. Аналіз затримки формування результату

8.6. Проаналізуйте правильність роботи схеми. На діаграмі (рис. 1.41) можна виміряти затримку формування результату – приблизно 9***ns***, впевниться, що це співпадає з теоретичними оцінками, які були сформовані під час компіляції (рис. 1.34).

Під час моделювання можна відключити врахування затримок формування вихідних сигналів на елементах пристроїв, для цього слід вибрати режим **Functional** у вікні налаштування симуляції, для цього в головному меню слід обрати команду **Assignment** і в меню, що відкрилося, вибрати команду **Setting** (піктограма на панелі інструментів). У вікні, що відкрилося у списку **Simulation Mode** необхідно вибрати режим **Functional**. Перед виконанням функціональної симуляції необхідно виконати команду **Generate Functional Simulation Netlis** із головного меню **Processing** для автоматичного налаштування симулятора.

Функціональне моделювання відображує правильність роботи схеми і рекомендоване для виконання першого етапу симуляції складних схем. Функціональне модулювання зручно використовувати для пошуку функціональних помилок в роботі пристрою. Після функціонального моделювання виконують етап моделювання з врахуванням часових затримок, для цього у списку **Simulation Mode** необхідно вибрати режим **Timing** (в меню **Assignment** вибрати команду **Setting**). Режим встановлений за замовчанням в налаштуваннях симулятора.

8.7. Перейдіть в режим функціонального моделювання **Functional**. Для цього в меню **Assignment** виберіть команду **Setting** і у списку **Simulation Mode** виберіть режим **Functional**. Виконайте команду **Generate Functional Simulation Netlis** із головного меню **Processing**. Для запуску симуляції використайте піктограму в панелі задач, або виберіть команду **Start Simalation** в меню **Processing**. Результат виконаної функціональної симуляції проекту ***CL_Name_Project_v1*** представлений на рис. 1.43.

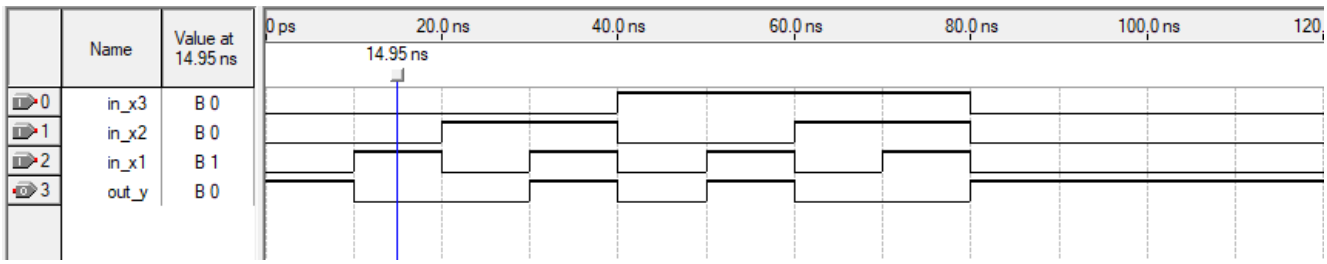


Рис. 1.43. Результати функціональної симуляції

9. Виконайте етапи введення, синтезу та симуляції комбінаційної схеми заперечення ДДНФ (або ДКНФ) функції f4.

9.1. ***Перший спосіб.*** Для реалізації ДКНФ функції f4 можна створити окремий проект (з назвою ***CL_Name_Project_v2***) і виконати пункти **1-8** для проектування комбінаційної схеми, приклад якої наведений на рис. 1.29, б.

9.2. ***Другий спосіб.*** Комбінаційну схему для реалізації ДКНФ функції f4 (рис. 1.29, б) можна ввести безпосередньо в файлі опису верхнього рівня вже створеного робочого проекту ***CL_Name_Project_v1***. Виберемо цей спосіб для експериментів в даній лабораторній роботі.

9.3. Відкрийте проект ***CL_Name_Project_v1***, відкрийте файл ***CL_Name_Project_v1.bdf*** (рис. 1.25). Додайте на робоче поле вхідні та вихідні контакти, наприклад ***in2_x3***, ***in2_x2***, ***in2_x1***, ***out2_y*** (рис. 1.44).

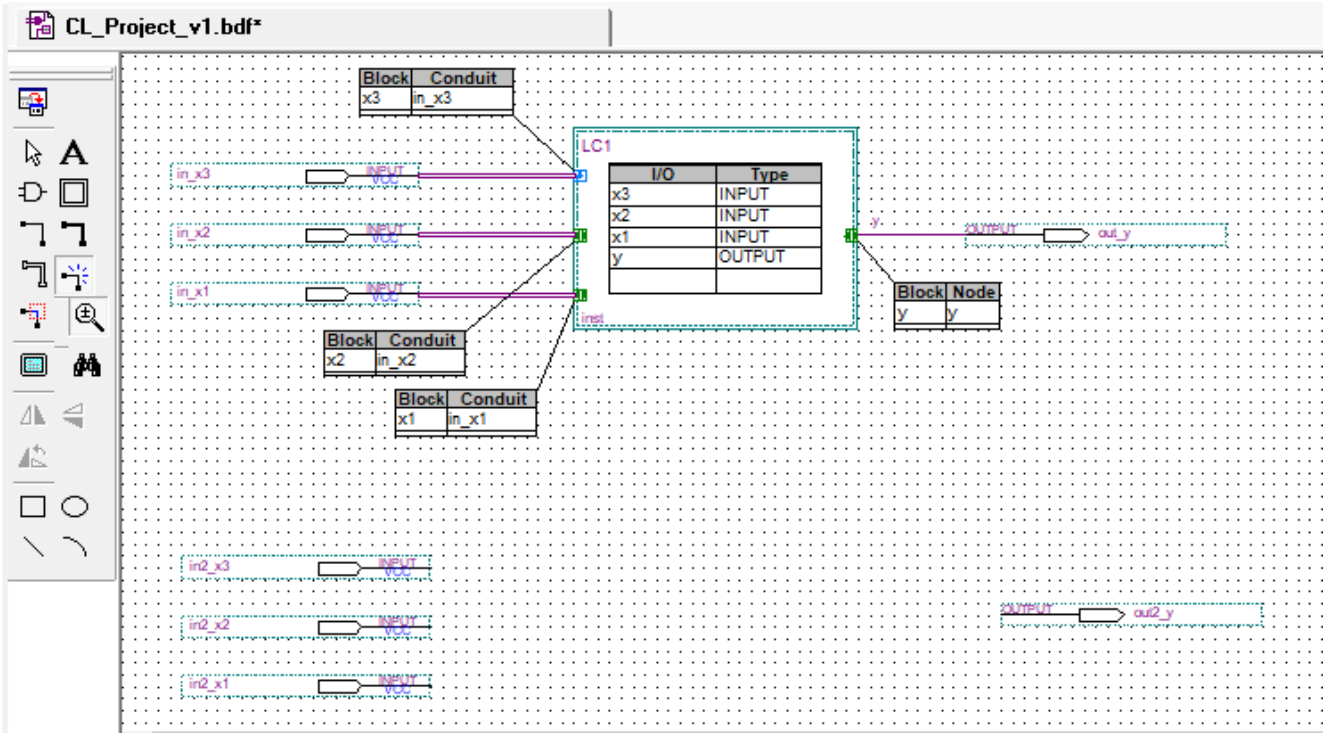


Рис. 1.44. Додавання контактів для логічної схеми ДКНФ перемикальної функції

В складних схемах припускається розривання провідників та шин. В цьому випадку провідники, які розірвані, повинні мати однакові назви. Особливо це зручно для розгалуження шин, наприклад восьмирозрядна шина ***data[7..0]*** з одного боку розгалужується на дві чотирирозрядні шини ***data[7..4]*** і ***data[3..0]*** (рис. 1.45). Компілятор поєднає провідники відповідно до призначених імен.

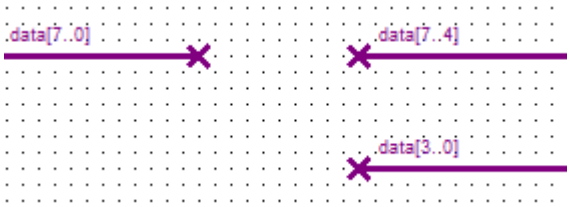


Рис. 1.45.

9.4. Введіть комбінаційну схему ДКНФ перемикальної функції f4 прямо в робочому полі файлу ***CL_Name_Project_v1.bdf.*** Результат представлений на рис. 1.46.

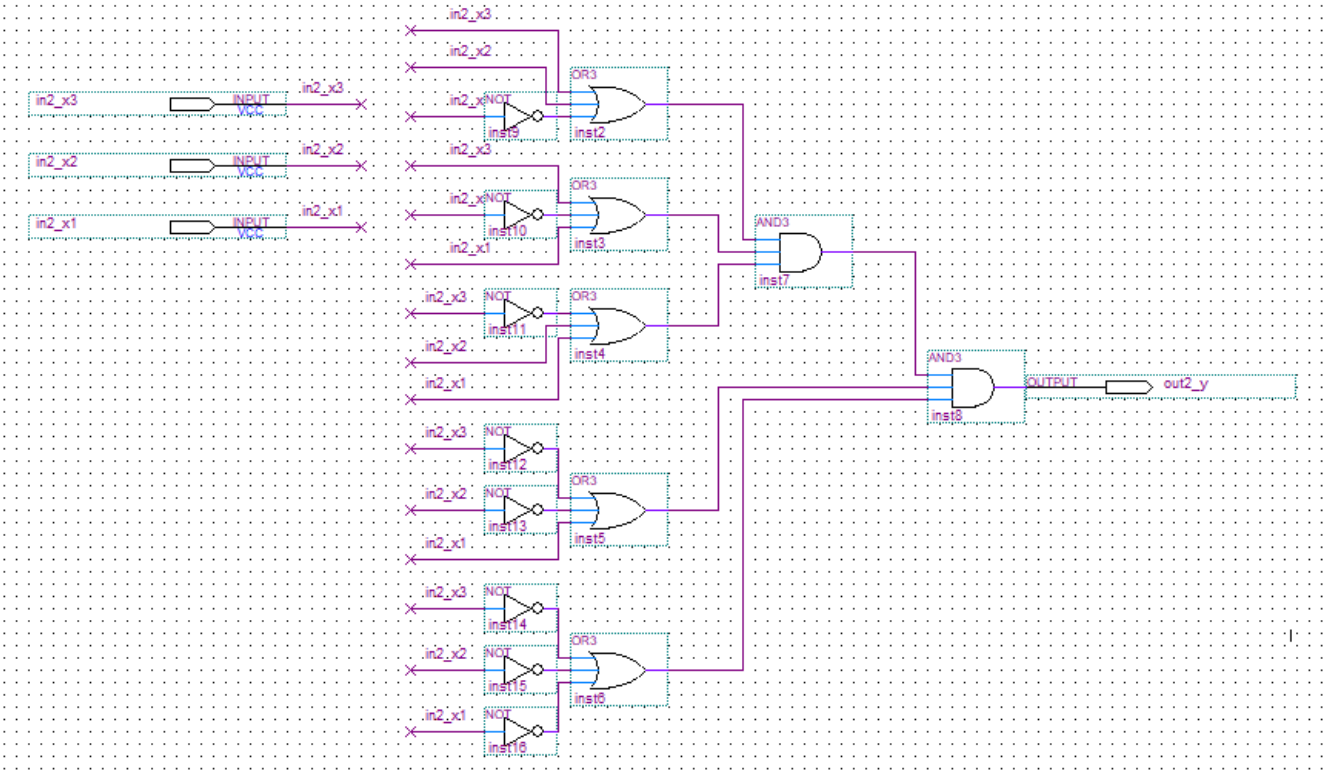


Рис. 1.46. Логічна схема ДКНФ перемикальної функції

9.5. Створіть стимуляційний файл (**Vector Wave Form File**) для моделювання. Для цього оберіть команду **New** меню **File**. В розділі **Verification/Debugging Files** оберіть тип файлу **Vector Wave Form File**. (рис. 1.35). Натисніть кнопку ОК. Збережіть новий файл з ім'ям ***CL_Name_Project_v2.vwf***.

9.6. Додайте вхідні сигнали для моделювання роботи схеми. Для цього у вікні **Insert Node or Bus** (рис. 1.36) оберіть назви сигналів, що належать до схеми ДКНФ перемикальної функції, або оберіть всі сигнали для порівняння роботи двох схем: ДДНФ і ДКНФ перемикальної функції f4.

9.10. Задайте в налаштуваннях симулятора новий вхідний файл для моделювання, для цього в меню **Assignment** виберіть команду **Setting** і у списку **Simulation Input**

задайте ім’я другого файлу зі стимулами для моделювання ***CL_Name_Project_v2.vwf***.

9.11. Виконайте етап функціональної симуляції. Для цього виконаннйте команду **Setting** головного меню **Assignment**, у вікні налаштування симулятора задайте режим симуляції **Functional**. Виконайте команду **Generate Functional Simulation Netlis** із головного меню **Processing**. Результат виконаної функціональної симуляції проекту представлений на рис. 1.47. На діаграмі представлено результати функціонального моделювання двох схем: ДДНФ і ДКНФ перемикальної функції f4. Видно, що результати збігаються, тобто цільова функція реалізована правильно.

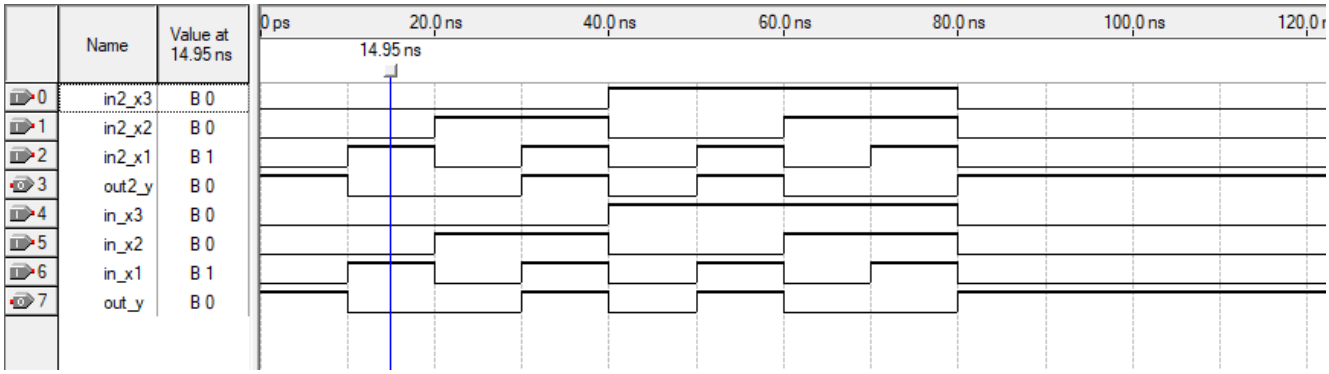


Рис. 1.47. Результати функціонального моделювання схем: ДДНФ і ДКНФ перемикальної функції f4

9.12. Дослідить часові параметри роботи розробленої схеми ДКНФ двох схем: ДДНФ і ДКНФ перемикальної функції f4. Для цього виконайте етап часової симуляції з врахуванням затримок. Перейдіть в режим симуляції Timing в налаштуваннях симулятора. Запустить симуляцію. На діаграмі (рис. 1.48) представлено результати моделювання схеми ДКНФ перемикальної функції f4 з врахуванням часових затримок передавання сигналів. Проаналізуйте результати симуляції, виміряйте час затримки формування результату, порівняйте отримані виміри з результатами компіляції. Результати часового аналізу впишіть в таблицю 1.4. Вкажіть на можливі причини виникнення короткочасних помилкових сигналів (просічок). Запропонуйте шляхи боротьби з короткочасними помилковими сигналами, виконайте корекцію комбінаційної схеми для подолання просічок.

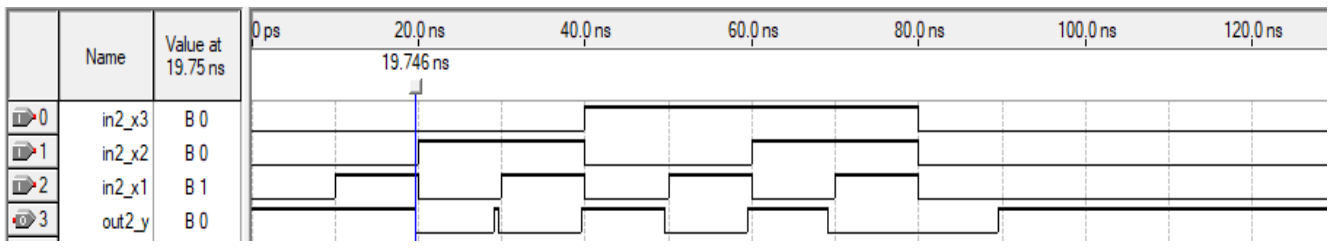


Рис. 1.48. Результати часового моделювання схем: ДДНФ і ДКНФ перемикальної функції f4

10. Зробіть загальні висновки по роботі.

Лабораторна робота № 2

РОЗРОБЛЕННЯ МОДУЛЯ КОМБІНАЦІЙНОГО СУМАТОРА

Мета роботи:

- *Вивчення особливостей функціональної побудови суматорів. Розроблення функціональної схеми суматора в САПР Quartus II, створення мегафункцій.*

За варіантами завдань із табл. 2.1. синтезувати комбінаційну схему суматора. Теоретичні відомості для виконання синтезу суматорів наведені в [1, 2].

Приклад синтезу повного однорозрядного суматора

Для прикладу виконаємо синтез повного однорозрядного суматора, що побудований на двох напівсуматорах. Функціональна схема такого суматора наведена на рис. 2.1.

Стислі теоретичні відомості для виконання завдання

Напівсуматором називають комбінаційну схему, що реалізує функції q суми по mod$k$ і переносу c при додаванні двох змінних.

Таблиця істинності напівсуматора наведена в табл. 2.2.

Таблиця 2.2

Таблиця істинності комбінаційного напівсуматора

x	y	q	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Виходячи з таблиці істинності отримаємо наступні логічні вирази:

$$q = x \cdot \bar{y} \vee \bar{x} \cdot y = (x \vee y) \cdot (\bar{x} \vee \bar{y}) = (x \vee y) \cdot \bar{c}, \quad c = x \cdot y.$$

Повний однорозрядний суматор може бути побудований із двох напівсуматорів (рис. 1.10). Комбінаційна схема напівсуматора наведена на рис. 1.11.

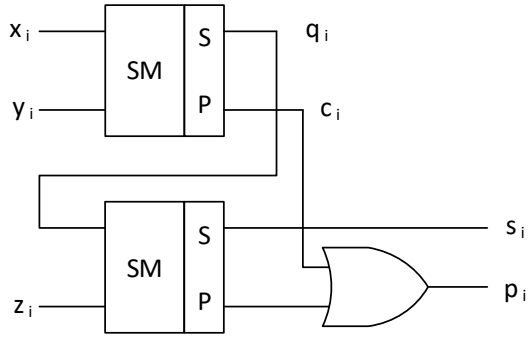


Рис. 2.1. Функціональна схема повного однорозрядного суматора, побудованого на двох напівсуматорах

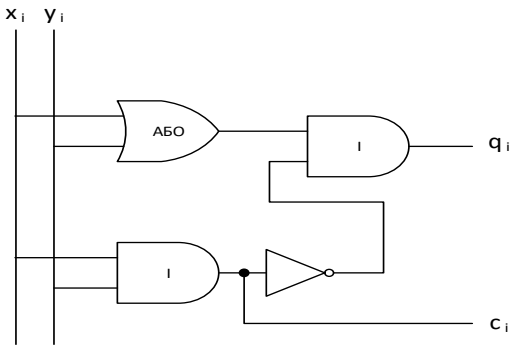


Рис. 2.2. Комбінаційна схема напівсуматора

1. Створіть файл верхнього рівня опису проекту. Для чого виконайте наступну послідовність дій:

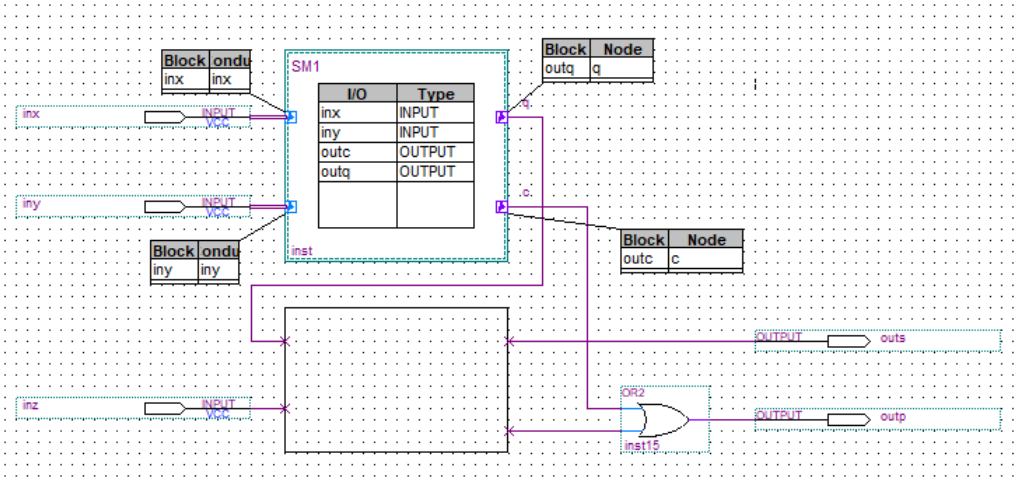


Рис. 1.26.

1.1. Створіть файли опису роботи блоку ***SM1_Name***. Для цього натисніть правою клавішею мишки на блоці напівсуматора та в контекстному меню виберіть команду «**Create Design File from Selected Block…**», відкриється вікно створення нового файлу опису виділеного блоку. В списку **File type** виберіть спосіб створення нового файлу, в нашому прикладі, це **Schematic** для створення файлу опису в графічному редакторі. Інші пункти списку призначенні для створення файлів опису на відповідних мовах програмування апаратури. Зверніть увагу, що САПР пропонує назву файлу опису відповідно до назви блоку, що описується. Можна задати власне ім’я файлу. Далі відкриється нова вкладка графічного редактора з заданим ім’ям (рис. 1.28). Новий файл з’явиться в навігаційній панелі (рис. 1.29).

Зверніть увагу, що знову створений файл опису містить контакти для приєднання до файлу верхнього рівня ієрархії, які були створені під час створення блоку верхнього рівня (рис. 1.28).

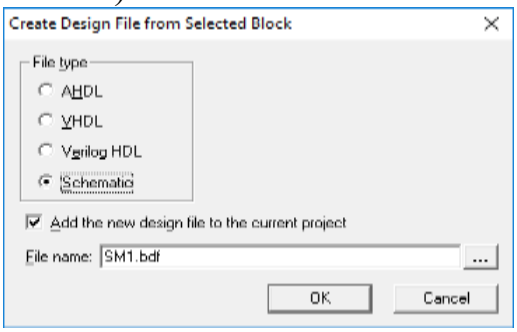


Рис.1.27.

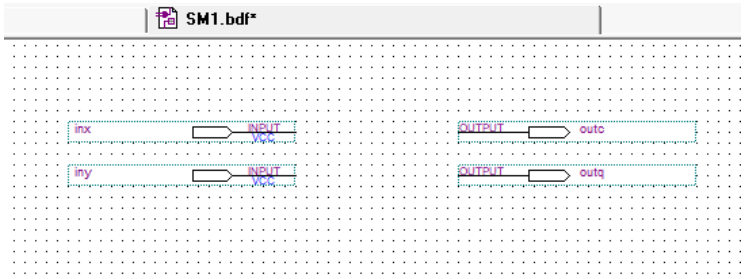


Рис. 1.28.

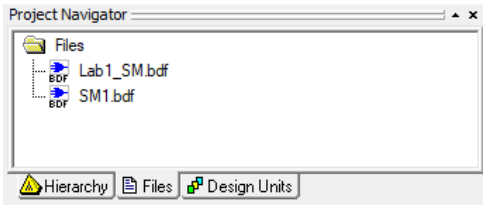


Рис. 1.29.

Розробіть на робочому полі функціональну схему напівсуматора, що зображена на рис. 1.11. Використайте стандартні компоненти: логічні елементи І, АБО. Збережіть створену схему. Результат зображений на рис. 1.29. Створений файл можна редагувати, відкривши файл із навігаційної панелі, або із контекстного меню, яке відкривається подвійним натисканням правої кнопки миші на блоці верхнього рівня опису проекту.

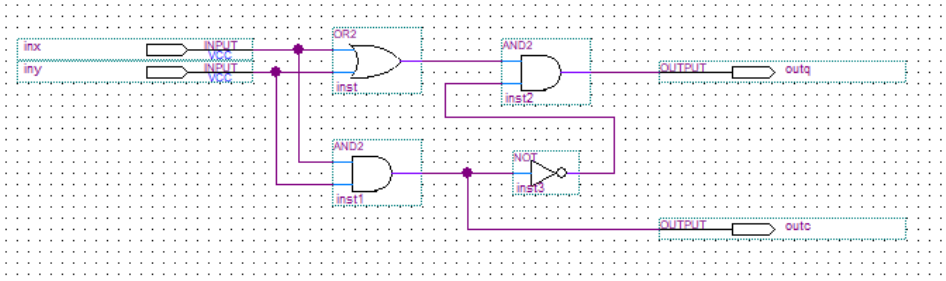


Рис. 1.29. Функціональна схема напівсуматора

Схематично ієрархічний принцип поєднання об’єктів проекту представлений на рис. 1.30.

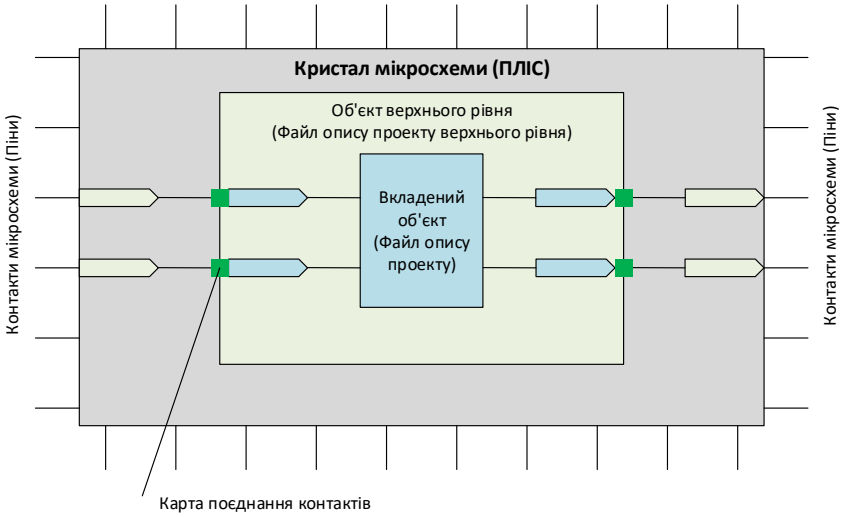


Рис. 1.30. Принцип ієрархічного поєднання об’єктів проекту

1.2. Створіть файли опису роботи блоку ***SM2_Name***.

Інший спосіб створення функціональних елементів це використання майстра **MegaWizard**. Цей майстер призначений для створення мегафункцій.

Для створення мегафункції напівсуматора виберіть команду **Mega Wizard plugin manager** в меню **Tools**. У вікні, що відкрилося (перша сторінка майстра), виберіть пункт **Create a new custom megafunction variation** (рис. 1.31). Натисніть кнопку **Next**, після чого відкриється вікно другої сторінки майстра **MegaWizard** (рис. 1.32).

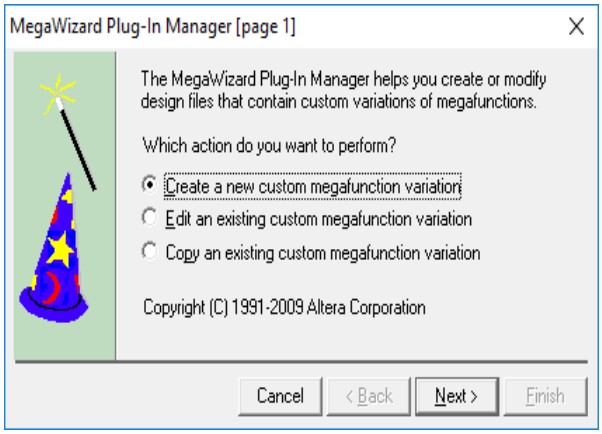


Рис. 1.31. Перша сторінка майстра **Mega Wizard plugin manager**

Базові біліотеки САПР *Quartus* II

Мегафункції (Майстр ***Mega Wizard plugin manager*)**

- Мегафункції (*ALTPLL, ALTLVDS, ALTDDIO*...) – створені компанією Altera, дозволяють налаштування модуля відповідно певних вимог користувача і особливостей обраного сімейства мікросхеми.
- Библиотека стандартних параметризовних модулів (*LPMs*):

  Параметризовні логічні модулі (*LPM_AND, LPM_DECODE*...)

  Параметризовні арифметичні модулі (*LPM_ADD_SUB, LPM_COUNTER*…)

Бібліотека стандартних компонентів (Інструмент ***Symbol Tool*)**

- Бібліотека примітивів (*Primitives*: *AND, OR, INPUT, DFFE* …);
- Бібліотека компонентів (серійні мікросхеми, налаштовані мегафункцій, IP ядра);

Користувацькі бібліотеки (*User Libraries*).

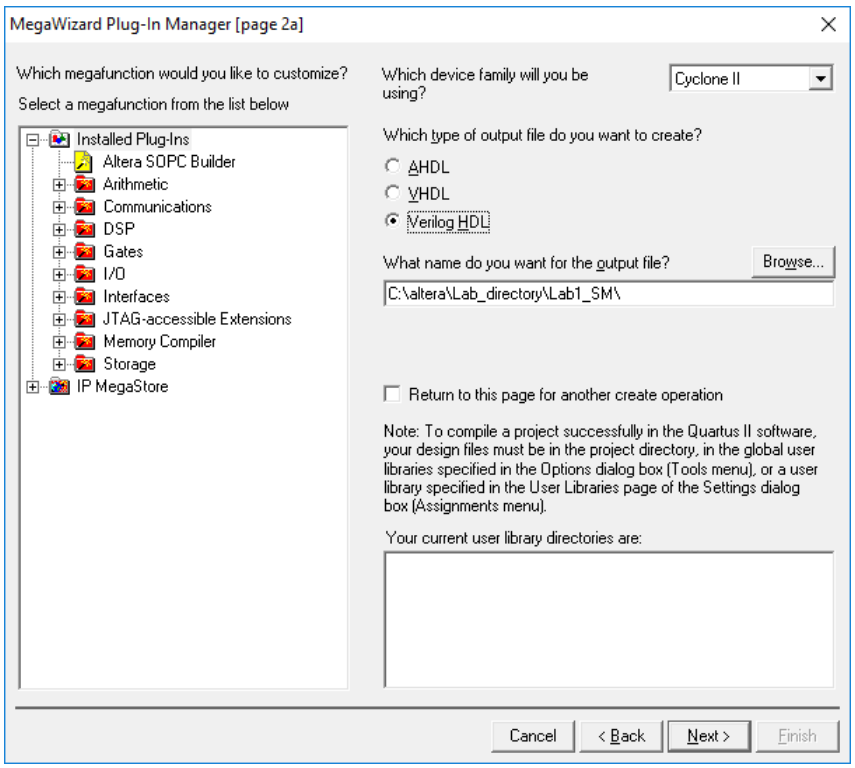


Рис. 1.32. Друга сторінка майстра **Mega Wizard**

Для створення мегафункції суматора налаштуйте наступні поля у другому вікні майстра:

– Встановіть прапорець для вибору мови опису інтерфейсу мегафункції, на якій буде додатково створено файл опису проекту – ***Verilog HDL***.

– Вкажіть ім'я файлу опису блоку суматора (***SM2_Name***) в полі **What name do you want for the output file?** Шлях до каталогу проекту вказаний у даному полі за замовчанням. Ви можете додати ім'я файлу в кінці обраного шляху до каталогу проекту або не вказувати повний шлях для автоматичного збереження файлу в робочій директорії проекту.

– Виберіть у списку бібліотечних компонентів пункт **Arithmetic**, де знаходяться мегафункції арифметичних операцій. Із бібліотеки параметризовних модулів (мегафункції з префіксом ***LPM***) виберіть модуль суматора ***LPM_ADD_SUB*** (модуль може бути також налаштований як модуль для віднімання).

– Перевірте обране сімейство мікросхем, які будуть використані під час синтезу мегафункції, у текстовому полі у верхньому куті праворуч.

– Натисніть кнопку **Next**.

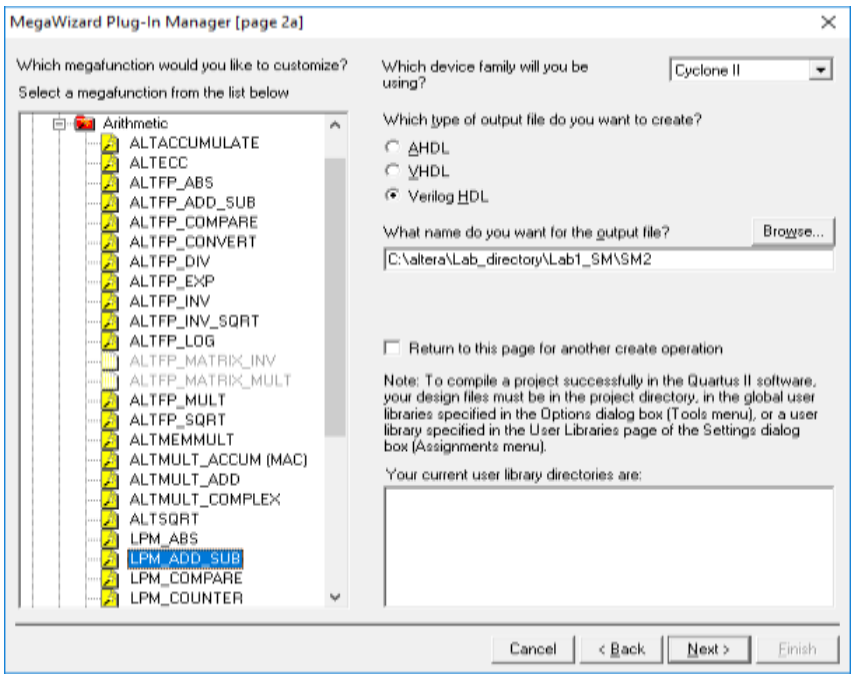


Рис. 1.33. Налаштування другої сторінка майстра **Mega Wizard**

У третьому вікні майстра (***General*)** (рис. 1.34) налаштуйте розрядність шин напівсуматора і виконувану операцію:

Для створення мегафункції суматора налаштуйте наступні поля у другому вікні майстра:

– Оберіть розрядність аргументів, для виконання лабораторної роботи це один біт. Розрядність встановіть у списку, що випадає.
– Установленням прапорця, оберіть операцію, яку слід виконати, для виконання лабораторної роботи це операція додавання ***Additional only***.
– На малюнку ліворуч зображено зовнішній вигляд блоку напівсуматора, який буде використано під час опису проекту.
– Натисніть кнопку **Next**.

У четвертому вікні майстра (***General* 2**) (рис. 1.35) залиште налаштування за замовчанням, де вказано, що на вхідні шини суматора надходять змінні – ***No, both values vary*** (не константи) та буде виконуватись операції додавання без знаку – ***Unsigned***. Натисніть кнопку **Next**.

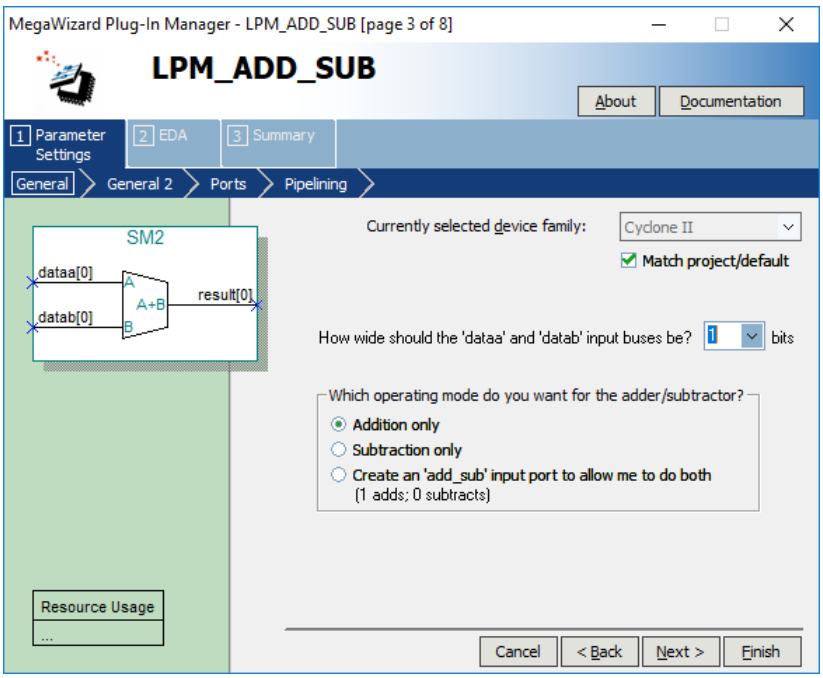


Рис. 1.34. Налаштування третьої сторінки майстра **Mega Wizard**

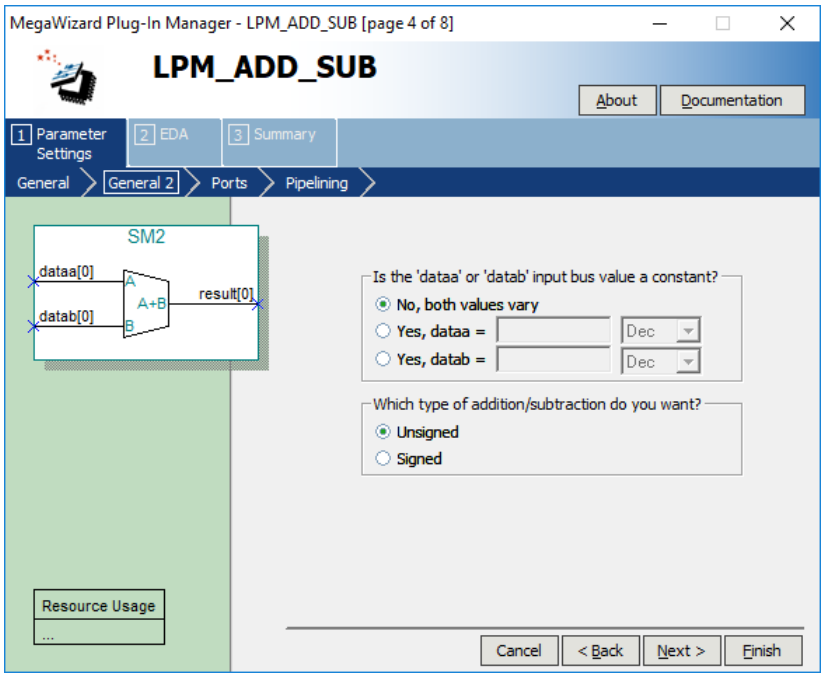


Рис. 1.35. Налаштування четвертої сторінки майстра **Mega Wizard**

У п’ятому вікні майстра (***Ports***) (рис. 1.36) додайте додатковий вихідний контакт для встановлення вихідного переносу – ***Create a carry output***. Натисніть кнопку **Next**.

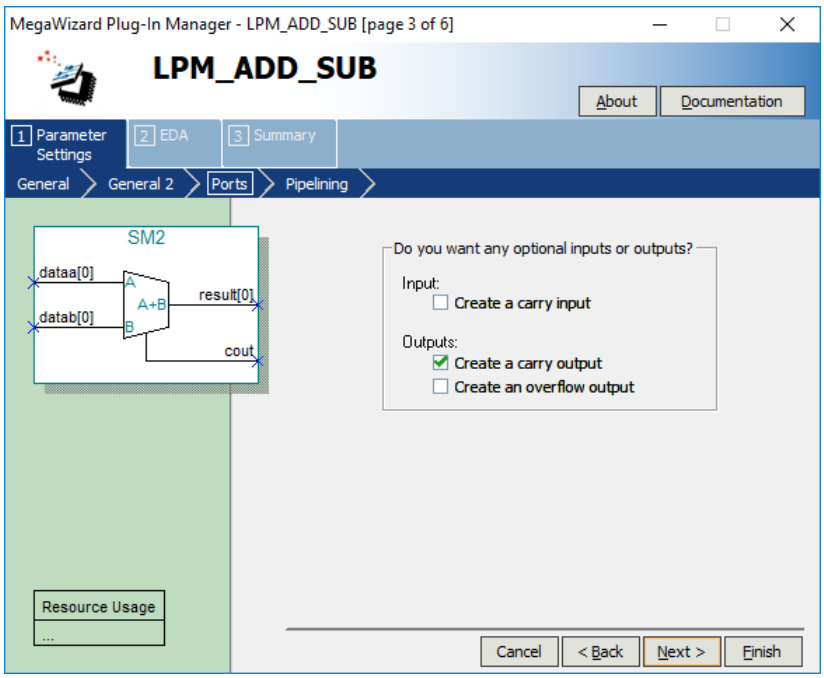


Рис. 1.36. Налаштування п’ятої сторінки майстра **Mega Wizard**

На шостій сторінці (***Pipelining***) виберіть пункт ***Yes, I want an output latency of 2 clock cycles*** (затримка видавання результату на 2 циклу тактової частоти) (рис. 1.37). Натисніть кнопку **Next**.

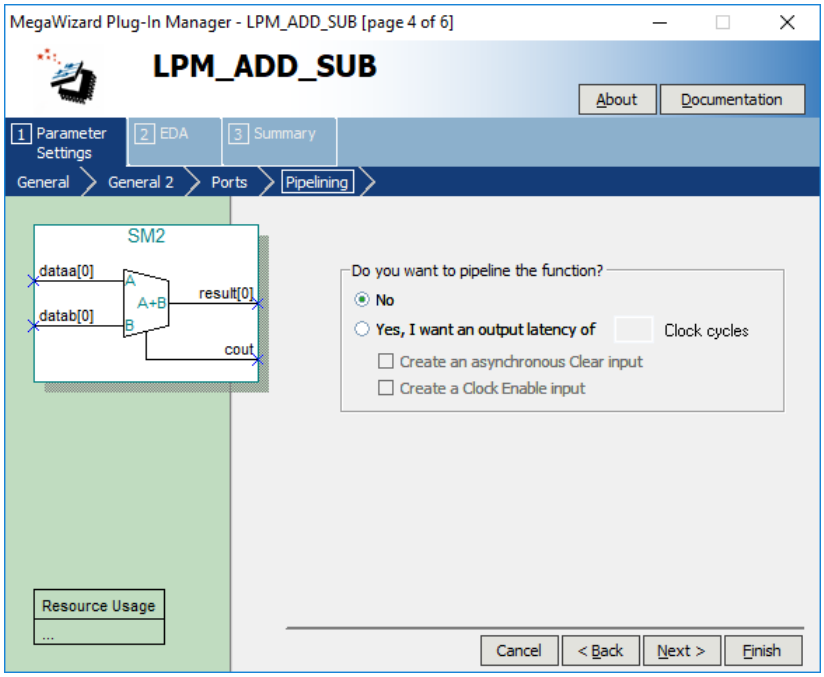


Рис. 1.37. Налаштування шостої сторінки майстра **Mega Wizard**

Надалі, на сьомій сторінці (друга вкладка ***EDA***) вказують файл **Lpm**, який використовується сторонніми САПР для моделювання мегафункції, але в даній лабораторній роботі сторонні САПР не використовуються, тому пропустить цю сторінку. Натисніть кнопку **Next**.

На восьмій сторінці (третя вкладка ***Summary***) відмітьте файли, що повинні бути створені майстром **Mega Wizard**. Для файлів типу ***Scematic*** це файли

SM2_Name.v (якщо для опису інтерфейсу мегафункції обрано мову ***Verilog***) і ***SM2_Name.bsf*** (файл із графічним зображенням (символом) блоку мегафункції) (рис. 1.38).

Натисніть кнопку ***Finish*** для закінчення створення мегафункції напівсуматора.

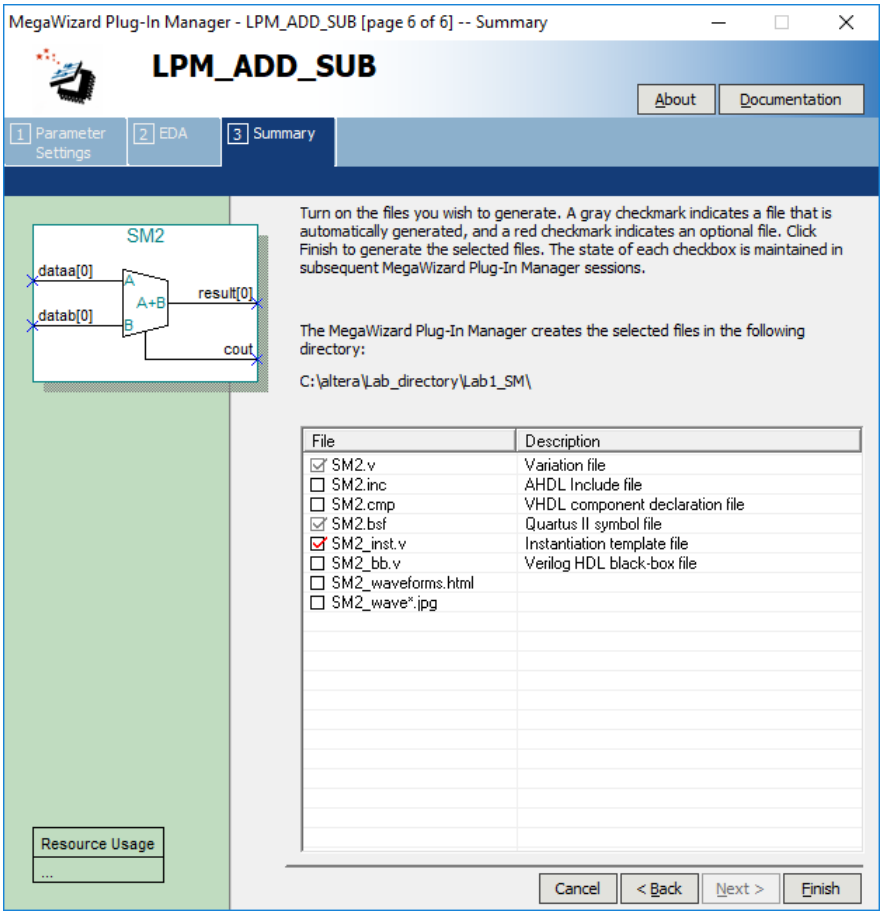


Рис. 1.38. Налаштування останньої сторінки майстра **Mega Wizard**

1.3. Додайте блок напівсуматора в проект і створіть необхідні зв'язки.

Відкрийте файл ***Lab1_Name_SM.bdf***, для цього можна використати команду **Open** в меню **File**, натиснути піктограму на панелі інструментів або два рази натиснути на імені файлу у вікні навігаційної панелі **Project Navigator**. Якщо файл не був закритий, його можна знайти на одній із вкладок робочого поля вікна САПР.

Відкрийте вікно ***Symbol*** за допомогою відповідної піктограми, або за допомогою подвійного натискання мишею в любому вільному місці робочого поля. У вікні оберіть вкладку ***Project.*** В цій папці розміщуються всі користувацькі мегафункції, створені в рамках проекту. Двічі натисніть лівою кнопкою миші на елементі ***SM2_Name***. Натисніть ліву кнопку миші в полі креслення і розмістить вибраний елемент замість прямокутника на функціональній схемі.

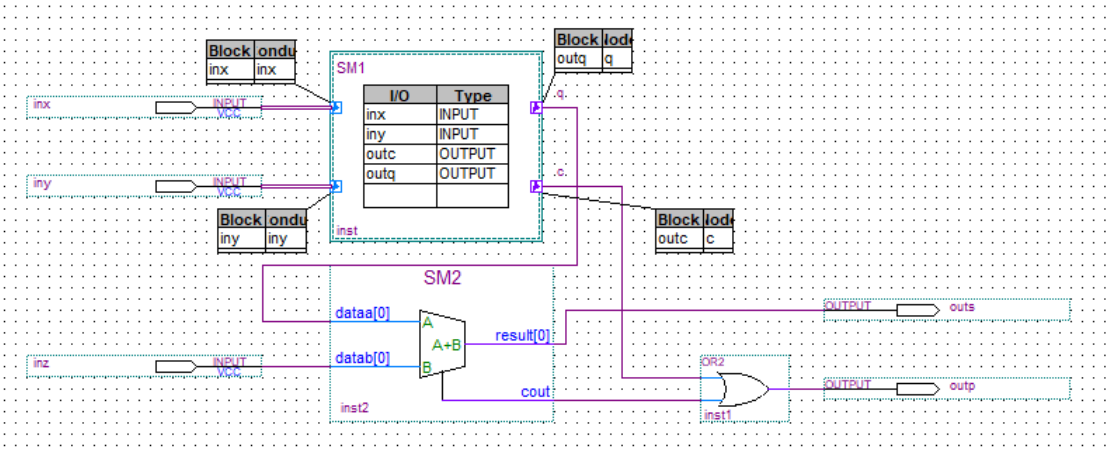


Рис. 1.38. Функціональна схема однорозрядного суматора в САПР ***Quartus* II**

1.4. Збережіть і перевірте проект. Для перевірки коректності створеної схеми виконайте перший етап компіляції ***Analysis & Elaboration***. Для цього в меню ***Processing*** оберіть команду ***Start ⇒ Start Analysis & Elaboration***.

Натисніть кнопку OK у вікні повідомлення, коли аналіз завершиться. Якщо були виявлені будь-які помилки, перевірте всі зв'язки схеми або повторно викличте **MegaWizard Plug-In Manager** для виправлення помилок в мегафункції.

1.5. Якщо перевірка пройшла успішно запустіть повну компіляцію проекту командою ***Start Compilation*** меню ***Processing***.

Лабораторна робота № 3

КОМПІЛЯЦІЯ ПРОЕКТУ В САПР *QUARTUS* II. АНАЛІЗ РЕЗУЛЬТАТІВ КОМПІЛЯЦІЇ

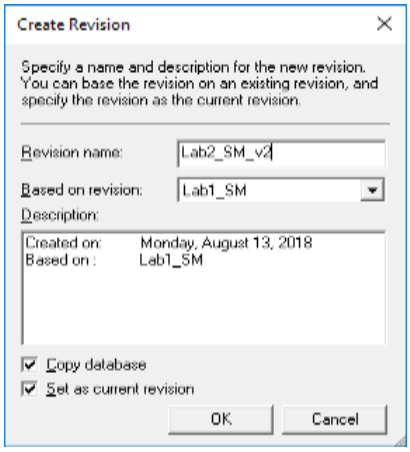
Мета роботи:

- *Вивчення процесу компіляції проектів в САПР Quartus II та отримання навиків аналізу результатів компіляції.*
- *Створення версій проектів в САПР Quartus II, порівняння результатів різних версій проектів.*
- *Вивчення різних способів введення проектів в САПР Quartus II.*
- *Отримання навиків роботи з редактором призначень.*

1. Порядок виконання роботи

1. Створіть нову версію проекту.

1.1. Оберіть команду **Revisions** в меню **Project**. У вікні **Revisions**, що відкрилося, натисніть кнопку **Create.** У вікні ***Create Revision***, що відкрилося за цим (рис. 2.1), задайте ім'я нової версії проекту. У списку, що випадає, оберіть ім'я батьківського проекту. Всі інші налаштування залиште за замовчанням. Натисніть кнопку ***OK***. Вікно ***Revision*** закриється. У списку, що випадає, на панелі інструментів вікна САПР з'явиться задане ім'я нової версії проекту (рис. 2.2). За допомогою цього списку можна переключатися між створеними версіями проекту. Оберіть проект з назвою ***Lab2_SM_v2.***



закри

Рис. 2.1. Вікно створення нової версії проекту. Задавання ім'я нової версії проекту

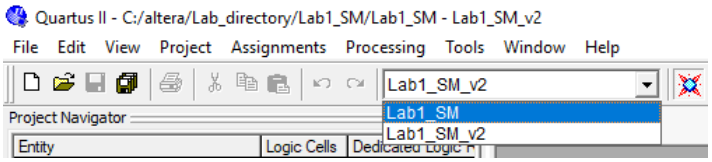


Рис. 2.2. Інструмент для переключення між версіями проекту

2. Реалізуйте повний однорозрядний суматор за допомогою мегафункції.

2.1. Створіть новий графічний файл опису проекту (**Schematic File**) за допомогою команди **New** меню **File**. Збережіть новий файл з назвою ***Lab2_SM_v2.bdf*** і встановіть його, як файл верхнього рівня ієрархії опису проекту. Це можна зробити в навігаційній панелі, натиснувши на назві файлу правою кнопкою миші. В контекстному меню оберіть команду **Set as top-level entity**. Відкрийте файл верхнього рівня опису проекту ***Lab1_SM.bdf***, скопіюйте схему і вставте її в робоче поле файлу ***Lab2_SM_v2.bdf***.

2.2. Замість двох напівсуматорів і логічного елементу АБО створіть функціональний блок повного однорозрядного суматора. Для цього видаліть непотрібні блоки і на робочому полі графічного редактора залиште вхідні та вихідні контакти. Скористуйтеся майстром **Mega Wizard plugin manager** в меню **Tools**. Детально процес створення мегафункції описано в пункті 3.10 Лабораторної роботи №1. Для створення повного суматора слід додати вхідний перенос ***Carry in*** і вихідний перенос ***Carry out***. Результат введення функціональної схеми повного однорозрядного суматора наведений на рис. 2.3.

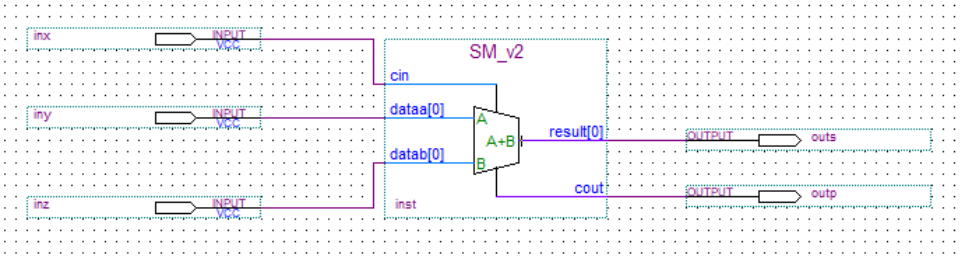


Рис. 2.3. Функціональна схема повного однорозрядного суматора

2.3. Збережіть і перевірте проект. Для перевірки коректності створеної схеми виконайте перший етап компіляції ***Analysis & Elaboration***. Для цього в меню ***Processing*** оберіть команду ***Start ⇒ Start Analysis & Elaboration***.

Натисніть кнопку OK у вікні повідомлення, коли аналіз завершиться. Якщо були виявлені будь-які помилки, перевірте всі зв'язки схеми або повторно викличте **MegaWizard Plug-In Manager** для виправлення помилок в мегафункції. Виконайте повторну перевірку.

3. Виконайте повну компіляцію версії проекту ***Lab2_SM***.

3.1. Переключиться на батьківську версію проекту ***Lab2_SM***, вибравши його назву в списку, що випадає, на панелі інструментів. Якщо під час виконання Лабораторної роботи №1 перевірка проекту пройшла успішно виконайте його компіляцію, для цього виберіть команду ***Start Compilation*** в меню ***Processing*** або натисніть піктограму , розташовану на панелі інструментів. Після завершення процесу компіляції відкриється діалогове вікно, в якому натисніть кнопку ***OK***.

3.2. Проаналізуйте інформацію про використані ресурси зі звіту компілятора:

– З розділу ***Flow Summary*** звіту компілятора ***Compilation Report*** (рис. 2.4),

випишіть в таблицю (табл. 2.1) наступні значення: ***Total logic elements***, ***total memory bits***, ***total embedded multiplier 9-bit elements і total pins***.

Total logic elements	2 / 4,608 (< 1 %)
Total combinational functions	2 / 4,608 (< 1 %)
Dedicated logic registers	0 / 4,608 (0 %)
Total registers	0
Total pins	5 / 158 (3 %)
Total virtual pins	0
Total memory bits	0 / 119,808 (0 %)
Embedded Multiplier 9-bit elements	0 / 26 (0 %)
Total PLLs	0 / 2 (0 %)

Рис. 2.4. Фрагмент розділу ***Flow Summary*** звіту компілятора ***Compilation Report***

– Розкрийте папку ***Fitter*** звіту компілятора ***Compilation Report***. Виберіть розділ ***Resource Section***. Із звіту ***Resource Utilization by Entity***, випишіть в таблицю (табл. 2.1) використані ресурси для реалізації функціональних блоків (рис. 2.5).

Fitter Resource Utilization by Entity

	Compilation Hierarchy Node	Logic Cells	Dedicated Logic Registers
1	⊟ |Lab1_SM	2 (1)	0 (0)
2	⊟ |SM2:inst2|	1 (0)	0 (0)
3	⊟ |lpm_add_sub:lpm_add_sub_component|	1 (0)	0 (0)
4	|add_sub_03i:auto_generated|	1 (1)	0 (0)

Рис. 2.5. Фрагмент розділу ***Resource Utilization by Entity*** звіту компілятора ***Compilation Reporttpd***

– З розділу ***Control Signals*** звіту компілятора ***Compilation Report***, випишіть основні керуючі сигнали і їх коефіцієнт розгалуження.

– Оцініть часові затримки за результатами звіту компілятора ***Timing Analyzer*** із розділів ***Timing Analyzer Summary*** і ***tpd*** (рис. 2.6).

Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Failed Paths
1	Worst-case tpd	N/A	None	9.813 ns	iny	outs	--	--	0
2	Total number of failed paths								0

tpd

	Slack	Required P2P Time	Actual P2P Time	From	To
1	N/A	None	9.813 ns	iny	outs
2	N/A	None	9.806 ns	iny	outp
3	N/A	None	9.041 ns	inz	outs
4	N/A	None	9.027 ns	inz	outp
5	N/A	None	8.834 ns	inx	outs
6	N/A	None	8.827 ns	inx	outp

Рис. 2.6. Аналіз швидкодії на підставі звіту компілятора ***Timing Analyzer***: а – найдовший шлях проходження сигналу; б – всі шляхи проходження сигналів

Таблиця 2.1.

Аналіз звіту компілятора

Звіти компілятора	***Lab1_SM***	***Lab2_SM_v2***
Total logic elements	2 (<1%)	
Total register	0	

Total memory bits	0	
Embedded multiplier 9-bit elements	0	
Total pins	5 (3%)	
SM2	1 Logic Cells	
Actual Time	9,813 ns	

3.3. Виконайте аналіз логічної реалізації проекту за допомогою утиліти ***RTL Viewer***

> Утиліта ***RTL Viewer*** дозволяє представити логічну реалізацію проекту в графічному вигляді. Це інструмент для аналізу результатів синтезу ***HDL*** проектів.
> Результати представлені утилітою це результати функціонального синтезу проекту, які не є його практичною реалізацією в мікросхемі ***Cyclone* II**.

Викличте утиліту ***RTL Viewer*** із меню ***Tools*** із списку ***Netlist Viewers***. Графічне відображення логічної реалізації проекту зображене на рис. 2.7. На схемі відображені контакти введення/виведення, блоки напівсуматорів.

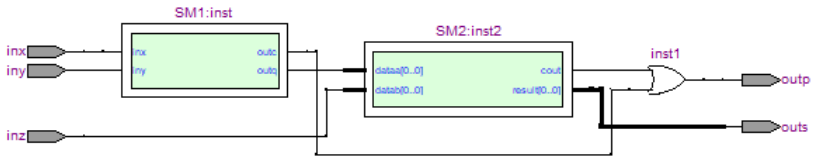


Рис. 2.7. Логічна реалізація проекту ***Lab1_SM***

Виділіть блок ***SM2.*** Натисніть на праву кнопку миші виберіть у контекстному меню команду **Hierarchy Down** (перехід на нижчій рівень ієрархії). Відкриється внутрішня структура нижнього рівня ієрархії блоку ***SM2.inst2***, з якої видно, що блок створений на базі однієї мегафункції ***lpm_add_sub***, яка створена автоматично (рис. 2.8). Натисніть ще раз мишею на блоці ***lpm_add_sub*** (можливо доведеться опуститися на декілька рівнів) і перегляньте логічну схему блоку перетворену на примітиви ***Quartus II*** (рис. 2.8).

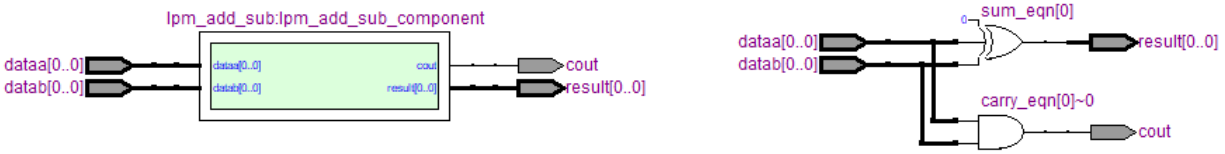


Рис. 2.8. Результати відображення логічної структури блоку ***SM2*** в утиліті ***RTL Viewer***

> Команда **Ungroup Selected Nodes** в контекстному меню корисна, якщо реалізовані багаторозрядні шини. За застосування цієї команди блоки будуть відображуватись з усіма

виводами. В цьому випадку видно, як поєднуються окремі сигнали. Для повернення до попереднього вигляду використайте команду ***Group Related Nodes.***

Для переходу на верхні рівні ієрархії використайте команду **Hierarchy Up** в тому самому контекстному меню.

Для перегляду логічної структури напівсуматора ***SM1***, використайте альтернативний спосіб – два рази натисніть на зображенні блоку лівою кнопкою миші. Відкриється нижній рівень ієрархії. Для переходу на верхній рівень, два рази натисніть мишею на вільному полі графічного вікна. Видно, що блок ***SM1*** реалізований користувачем на логічних елементах, зображення логічної структури блоку у примітивах ***Quartus II*** представлено на рис. 2.9.

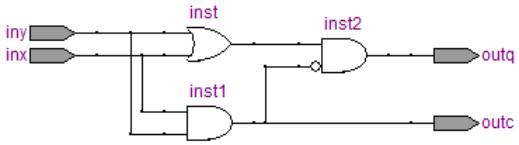


Рис. 2.9. Результати відображення логічної структури блоку ***SM1*** в утиліті ***RTL Viewer***

Виділить ще раз блок ***SM2***. В контекстному меню оберіть команду **Display Content**.

Команда ***Display Content*** дозволяє відобразити нижній рівень ієрархії в межах верхнього рівня. Виділіть блок ***lpm_add_sub*** і в його контекстному меню оберіть команду ***Display Content*** – відобразиться ще один нижній рівень ієрархії. Результат використання команди ***Display Content*** зображено на рис. 2.10. Для закривання нижніх рівнів ієрархії використайте команду ***Hide Content***.

За допомогою команди Zoom контекстного меню блоку проекспериментуйте зі зміною масштабу зображення.

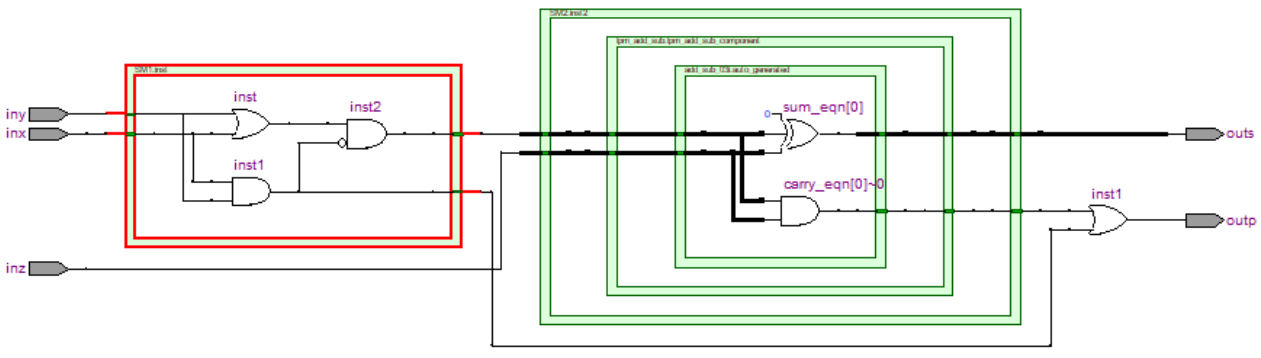


Рис. 2.10. Результат використання команди ***Display Content*** утиліти ***RTL Viewer***

Зверніть увагу, що вікно ієрархії проекту ***Hierarchy List*** утиліти ***RTL Viewer*** також відображує ієрархічну структуру кожного блоку. Для кожного рівня ієрархії додатково відображуються імена контактів і внутрішніх мереж (рис. 2.11).

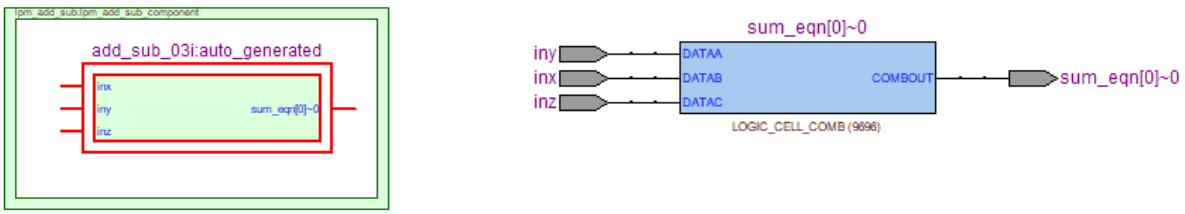


Рис. 2.11. Відображення інформації у вікні ***Hierarchy List*** утиліти ***RTL Viewer***

3.4. Виконайте аналіз фізичної реалізації проекту за допомогою утиліти ***Technology Map Viewer***

> Утиліта ***Technology Map Viewer*** дозволяє переглядати фактичну реалізацію проекту і використовувані ресурси FPGA/CPLD. Утиліта ***Technology Map Viewer*** використовується в якості допоміжного інструменту в процесі налагодження проекту для аналізу змін використаних ресурсів або змін налаштувань проекту.

Перегляньте фізичну реалізацію блоку ***SM2*** за допомогою утиліти ***Technology Map Viewer***, для цього (I спосіб) у вікні утілити **RTL Viewer** перейдіть на ієрархічний рівень, що відображує модуль мегафункції ***lpm_add_sub***, виділіть його і оберіть у контекстному меню команду **Locate**, а далі команду **Locate in Technology Map Viewer**. В результаті відкриється вікно утиліти ***Technology Map Viewer*** із зображенням виділеного блоку (рис. 2.12, а). Перейдіть на нижній рівень ієрархії будь яким способом, що описаний вище (рис.2.12, б).



а б

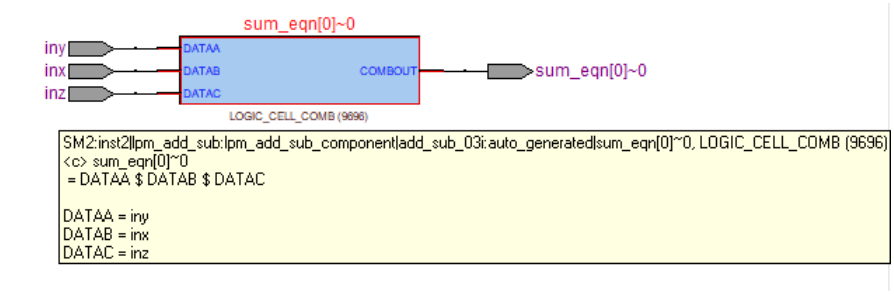
Рис. 2.12. Фізична реалізацію блоку ***SM2*** в технологічному редакторі ***Technology Map Viewer***

На відміну від примітивів, що зображуються на нижньому рівні утилітою ***RTL Viewer***, ***Technology Map Viewer*** відображає фактично використаний ресурс мікросхеми – номер логічної комірки, яка використана на мікросхемі для реалізації логічного елементу, логічну функцію, яка реалізована даним логічним елементом (рис. 2.13, а).

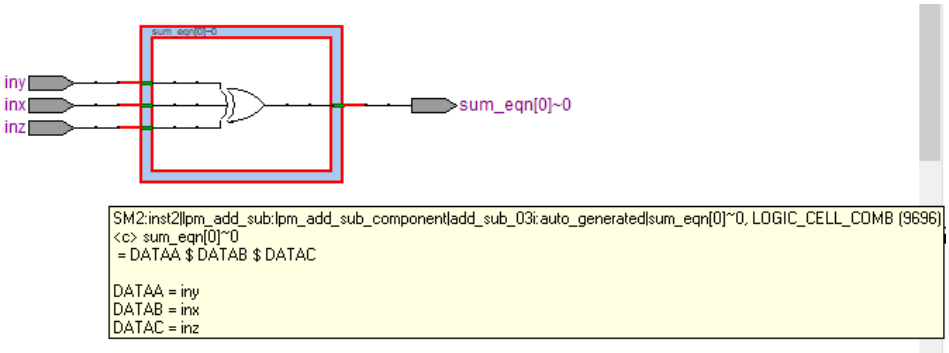
> В інших структурах утиліта ***Technology Map Viewer*** надає можливість переглянути будь які фактично використані ресурси, наприклад, регістри із складу логічної комірки, блоки

> убудованої пам’яті, помножувачі інші спеціалізовані убудовані блоки мікросхем обраних сімейств.

Двічі натисніть ліву кнопку миші на блоці ***sum*** для докладного аналізу реалізації модуля ***sum*** на базі мегафункції ***lpm_add_sub*** (рис. 2.13, б).



а



б

Рис. 2.13.

Результати аналізу функціонального і фізичного синтезу показують, що трасувальник виконав певну оптимізацію, що називається упаковуванням. Трасувальник перемістив логічний елемент АБО в блок напівсуматора, що реалізований за допомогою мегафункції ***lpm_add_sub*** для покращення продуктивності і зменшення кількості використаних логічних ресурсів мікросхеми (в утиліті видно, що логічний елемент АБО розташовувався за межами блоку SM2). Оптимізований модуль реалізований за допомогою логічного елементу ВИКЛЮЧНЕ АБО з трьома входами. Це однорівнева комбінаційна схема, яка забезпечує збільшення швидкодії у порівнянні з дворівневою реалізацією.

> В даному випадку трасувальник видає повідомлення "***Extra Info***" (додаткова інформація) у вікні повідомлень ***Message*** (або в таблиці ***Suppressed***). Це вказує на те, що відбулася оптимізація. Це можна перевірити також за допомогою редактора ресурсів ***Resource Property Editor*** в команді Locate контекстного меню певного блоку.

Перегляньте фізичну реалізацію проекту ***Lab1_SM*** цілком ***SM2*** за допомогою утиліти ***Technology Map*** **Viewer** у інший спосіб (II спосіб), для цього викличте утиліту ***Technology Map Viewer*** із меню ***Tools*** із списку ***Netlist Viewers***. Графічне відображення фізичної реалізації проекту зображене на рис. 2.14, з якого видно, що трасувальник дійсно застосував певну оптимізацію і схема відрізняється від логічної структури проекту, яка зображена на рис. 2.7.

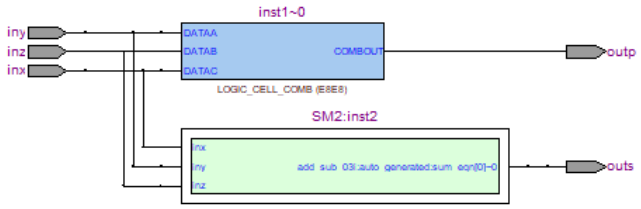


Рис. 2.7. Фізична реалізація проекту ***Lab1_SM***

Дослідить фізичну структуру нижніх рівнів ієрархії блоку SM2 у технологічному редакторі ***Technology Map Viewer***.

Перегляньте фізичну реалізацію блоку ***SM1*** за допомогою утиліти ***Technology Map Viewer.*** Аналогічно до описаної вище послідовності кроків перегляньте спосіб фізичної реалізації блоку ***inst1***, який є результатом оптимізації напівсуматора ***SM1*** трасувальником. Результат аналізу представлений на рис. 2.14.

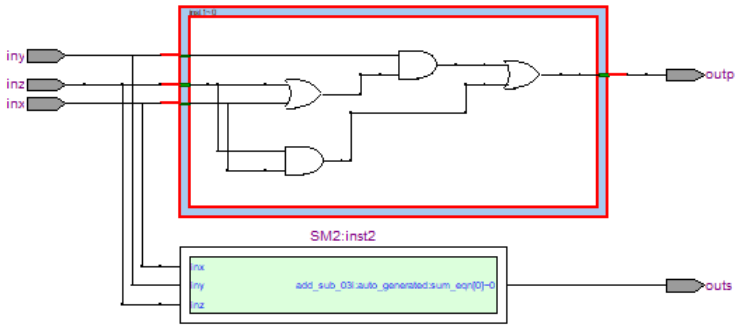


Рис. 2.8. Фізична структура оптимізованого блоку напівсуматора в технологічному редакторі ***Technology Map Viewer***.

4. Перевірте зв'язки в проекті за допомогою редактора топології кристала ***Chip Planner***.

> Редактор ***Chip Planner*** надає можливість перегляду фізичного розміщення логічних комірок вашого проекту на кристалі. Редактор ***Chip Planner*** зображує особливості реалізації проекту, а також архітектурні особливості FPGA/CPLD. Редактор ***Chip Planner*** призначений для ручного призначення логічних ресурсів, але може бути використаний й для оцінки отриманих результатів компіляції.

4.1. У вікні технологічного редактора ***Technology Map Viewer***, виділіть блок ***SM2:inst2***. Натисніть праву кнопку миші і виберіть команду **Locate ⇒ Locate in Chip Planner (Floorplan & Chip Editor)** в контекстному меню. У вікні редактора топології кристалу ***Chip Planner***, що відкрилося, оберіть піктограму на панелі інструментів, і за допомогою правої кнопки миші зменшить зображення в декілька разів. На зображенні, що відкрилося визначений блок виділений синім кольором.

> Всі інші задіяні ресурси (або, якщо зняти виділення) будуть виділені коричневим кольором. В середньому полі знаходяться логічні комірки, що угруповані в логічні блоки по 2x8 логічних комірок. Вільні комірки зображені білим кольором. У вертикальних рядах розміщені убудовані блоки ***RAM*** і помножувачів. По периметру – контакти (піни) мікросхеми.

Огляньте зображення кристалу у редакторі. Видно, що зайнято дві логічні комірки і задіяно п’ять пінів, що відповідає звіту компілятора. Позначення задіяних пінів у вікні редактора відповідає позначенням, що записано в звіті компілятора (***Compilation Report*** > ***Resource Section*** > ***Input Pins*** (***Output Pins***) (рис. 2.11).

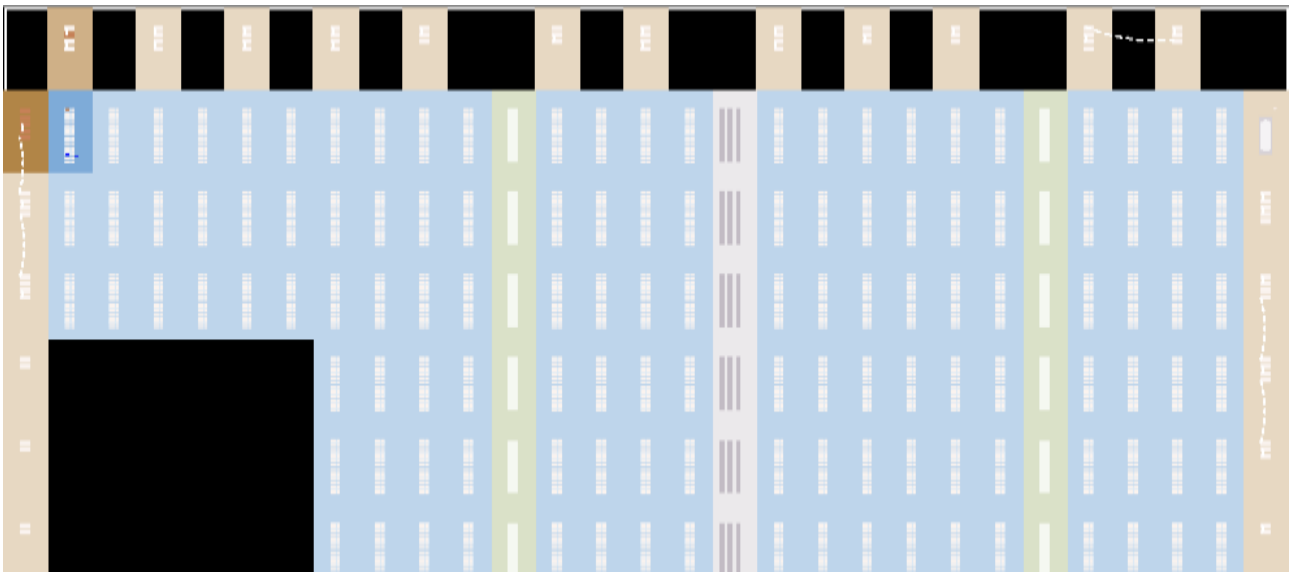


Рис. 2.9. Розміщення логічних ресурсів на мікросхемі ***Cyclon II***.

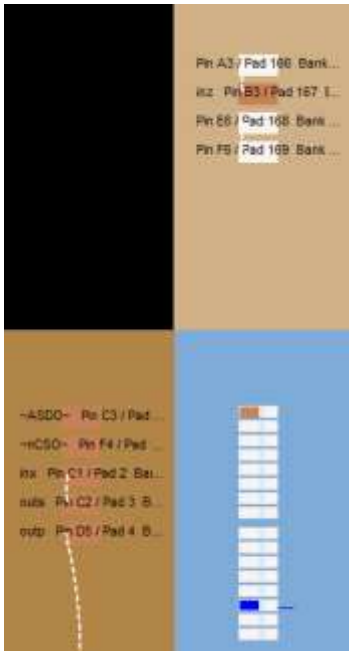


Рис. 2.10. Збільшений фрагмент кристалу у редакторі ***Chip Planner***

Input Pins

	Name	Pin #	I/O Bank	X coordinate	Y coordinate	Cell number	Combinational Fan-Out
1	inx	C1	1	0	13	2	2
2	iny	N8	4	7	0	2	2
3	inz	B3	2	1	14	1	2

Output Pins

	Name	Pin #	I/O Bank	X coordinate	Y coordinate	Cell number
1	outp	D5	1	0	13	4
2	outs	C2	1	0	13	3

Рис. 2.11. Звіт компілятора щодо задіяних контактів мікросхеми

У вікні редактора ***Chip Planner*** виділені області – це ресурси, що використовуються для розміщення логічних блоків проекту на мікросхемі ***Cyclone*** **II**. У випадку, якщо користувачем не зроблено ніяких призначень щодо розміщення логічних блоків, трасувальник самостійно призначає необхідні ресурси.

4.2. Якщо блок пам'яті виділено у вікні ***Chip Planner***, натисніть кнопку (**Generate Fan-In Connections**) на панелі інструментів (відображення вхідних

зв'язків для виділеного блоку). У вікні ***Chip Planner*** відобразяться ті логічні ресурси мікросхеми, від яких надходять вхідні сигнали до логічного блоку (блоки введення/вивення) і затримки поширення сигналів по виділеним зв'язкам.

Виділіть знову блок логічний блок ***SM2*** в ***Chip Planner*** (за необхідності зменшить масштаб зображення) і натисніть кнопку (**Generate Fan-Out Connections**) на панелі інструментів. У вікні ***Chip Planner*** одночасно відобразяться вхідні і вихідні зв'язки виділеного блоку (рис. 2.12). Вихідні зв'язку показані синім кольором.

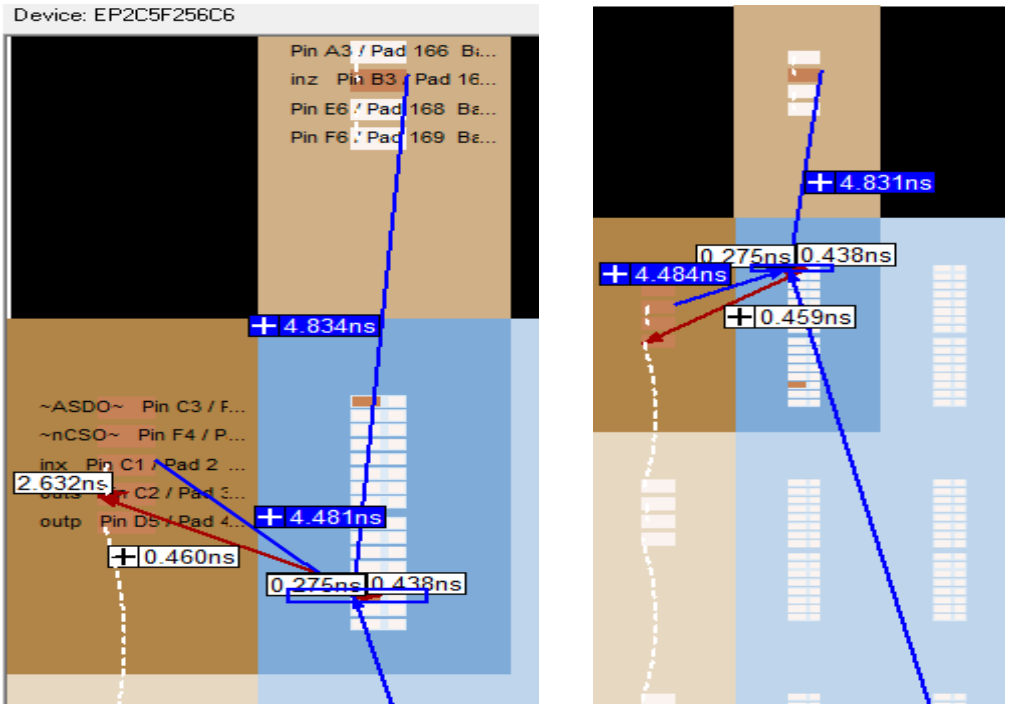
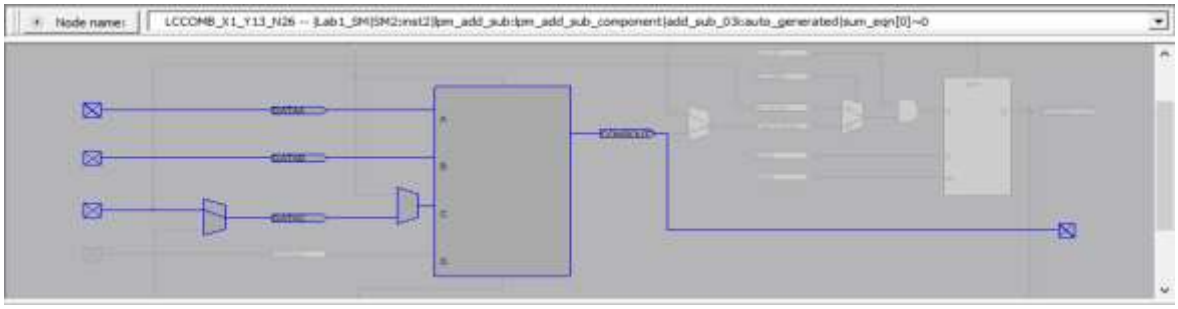


Рис. 2.12. Вхідні і вихідні зв’язки в редакторі ***Chip Planner***

Скасуйте команди **fan-in/fan-out**, натиснувши лівою кнопкою миші на будь-якому невиділеному блоці, а потім натисніть кнопку (**Clear Unselected Connections / Paths**) на панелі інструментів.

4.3. Перегляньте використані ресурси логічних комірок, які задіяні для реалізації комбінаційних схем. Для цього натисніть два рази лівою кнопкою миші на виділеному блоці. При цьому відкриється вікно редактора ресурсів ***Resource Property Editor*** (рис. 2.13).



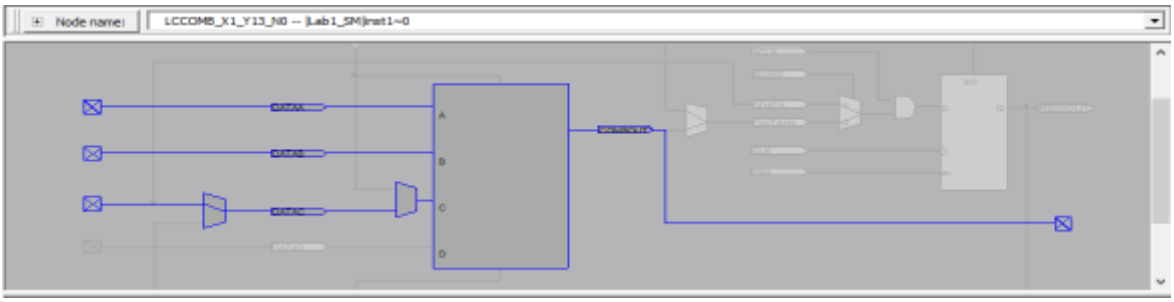


Рис. 2.13. Аналіз використаних ресурсів логічних комірок у редакторі ресурсів ***Resource Property Editor***

4.4. Зверніть увагу, що редактор топології кристалу ***Chip Planner*** можна відкрити із меню ***Tools*** із списку ***Netlist Viewers***.

5. Закрийте вікна ***RTL Viewer***, ***Technology Map Viewer*** та ***Chip Planner***.

Індивідуальне завдання

1. Намалюйте в протоколі вихідну схему проекту ***Lab1_SM***, яка була введена в САПР, логічну схему в примітивах ***Quartus* II** і схему фізичної реалізації проекту на визначеному сімействі мікросхем, виконану САПР ***Quartus* II**, на базі отриманих результатів компіляції проекту і перегляду результатів в утилітах ***RTL Viewer*** і ***Technology Map Viewer***.
2. Випишіть із звіту компілятора ***Compilation Report*** час розповсюдження сигналів за всіма ланцюгами схем (рис. 2.6). Порівняйте схеми і зробіть висновки, щодо оптимізації, виконаної трасувальником.
3. Зробіть висновки, щодо способу реалізації схеми напівсуматора – користувацька схема, або мегафункція розроблена компанією Altera. Оцініть швидкодію роботи блоків і використані ресурси.
4. Виконайте повну компіляцію другої версії проекту ***Lab2_SM_v2.***
5. Виконайте аналіз результатів компіляції за всіма кроками, що описані вище.
6. Доповніть табл. 2.1 отриманими параметрами.
7. Додайте до схем, намальованих в протоколі (згідно пункту 1 індивідуального завдання) схеми реалізації повного суматора за допомогою мегафункції Altera.
8. Випишіть із звіту компілятора ***Compilation Report*** час розповсюдження сигналів за всіма ланцюгами схем (рис. 2.6). Порівняйте схеми і зробіть висновки, щодо доцільності оптимізації, виконаної трасувальником.
9. Зробіть висновки, щодо способу реалізації схеми напівсуматора – користувацька схема, або мегафункція розроблена компанією Altera. Оцініть швидкодію роботи блоків і використані ресурси.
10. Зробіть загальні висновки

Лабораторна робота № 4

СИМУЛЯЦІЯ В САПР *QUARTUS* II. МОДЕЛЮВАННЯ

Мета роботи:

- *Вивчення процесу симуляції в САПР Quartus II та отримання навиків моделювання.*
- *Створення власних бібліотечних модулів в Quartus II.*
- *Розроблення пристроїв на базі власних бібліотечних модулів. Синтез багаторозрядних суматорів.*
- *Призначення в Quartus II. Отримання навиків роботи з редактором призначень.*
- *Програмування ПЛІС. Робота зі стендом DE2 Board Altera*

1. Порядок виконання роботи

1. Відкрийте проект ***Lab1_SM_v2***. Для цього необхідно відкрити батьківський проект ***Lab1_SM*** і за допомогою інструменту для переключення між проектами (рис. 2.2) вибрати проект ***Lab1_SM_v2***. Надалі у навігаційному меню вибрати файл проекту ***Lab1_SM_v2.qpf***. Виконайте симуляцію проекту ***Lab1_SM_v2.***

1.1. Створіть стимуляційний файл (**Vector Wave Form File**) – ***Test Bench*** файл, що містить стимули для моделювання. Для цього оберіть команду **New** меню **File**. В розділі **Verification/Debugging Files** оберіть тип файлу **Vector Wave Form File**. (рис. 3.1). Натисніть кнопку ОК. В головному вікні з'явиться вкладка з назвою файлу за замовчанням ***Waveform.wvf***. На вкладці знаходиться інструмент для створення часової діаграми для моделювання роботи проекту. Збережіть новий файл з ім'ям проекту ***Lab2_SM.vwf***.

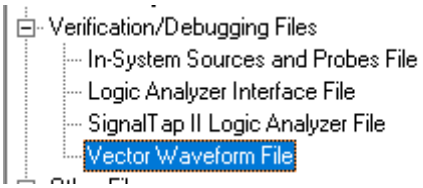


Рис. 4.1. Створення стимуляційного файлу **Vector Wave Form File**

1.2. Додайте вхідні сигнали для моделювання роботи схеми. Для цього в полі **Name** натисніть два рази лівою кнопкою миші. Відкриється вікно **Insert Node or Bus** (рис. 4.2). Якщо назви сигналів відомі введіть їх в полі **Name**.

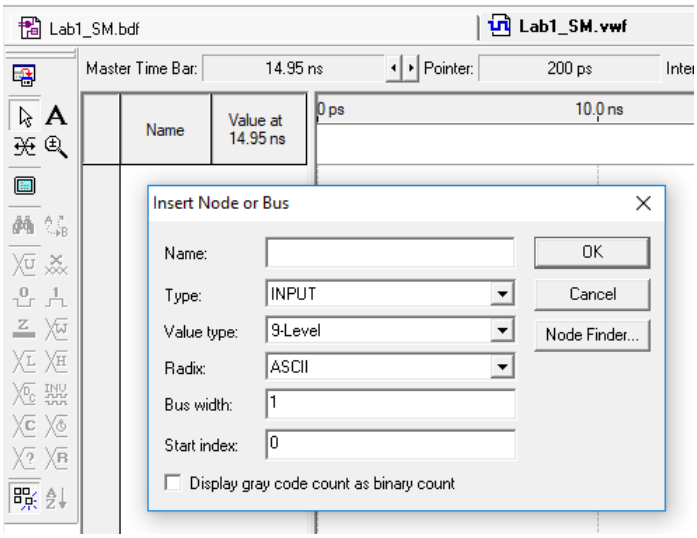


Рис. 4.2. Інструмент **Insert Node or Bus** для додавання сигналів для моделювання.

Інакше скористайтеся інструментом **Node Finder** для пошуку сигналів. Для цього натисніть кнопку **Node Finder** у вікні **Insert Node or Bus**, відкриється відповідне вікно **Node Finder** (рис. 4.3). Скористайтеся фільтром (**Filter**) для вибору групи сигналів (**Pins: input**). Натисніть кнопку **List**. Перелік необхідних вхідних сигналів відкриється в полі **Nodes Found**. Виберіть необхідні сигнали і перенесіть їх у поле **Selected Nodes** за допомогою кнопки >, або всі сигнали за допомогою кнопки >>. Натисніть кнопку **ОК**. У вікні **Insert Node or Bus** можна встановити формат вхідних та вихідних сигналів, для чого із переліку **Radix** виберіть, наприклад **Binary** (двійковий формат), після чого натисніть кнопку **ОК**. У робочому полі часової діаграми з'являться вхідні сигнали для моделювання (рис. 4.3, а). Аналогічно додайте вихідні сигнали (**Pins: output**), після чого в робочому вікні ***Lab2_SM.vwf*** з'являться і вихідні сигнали (рис. 4.4, б).

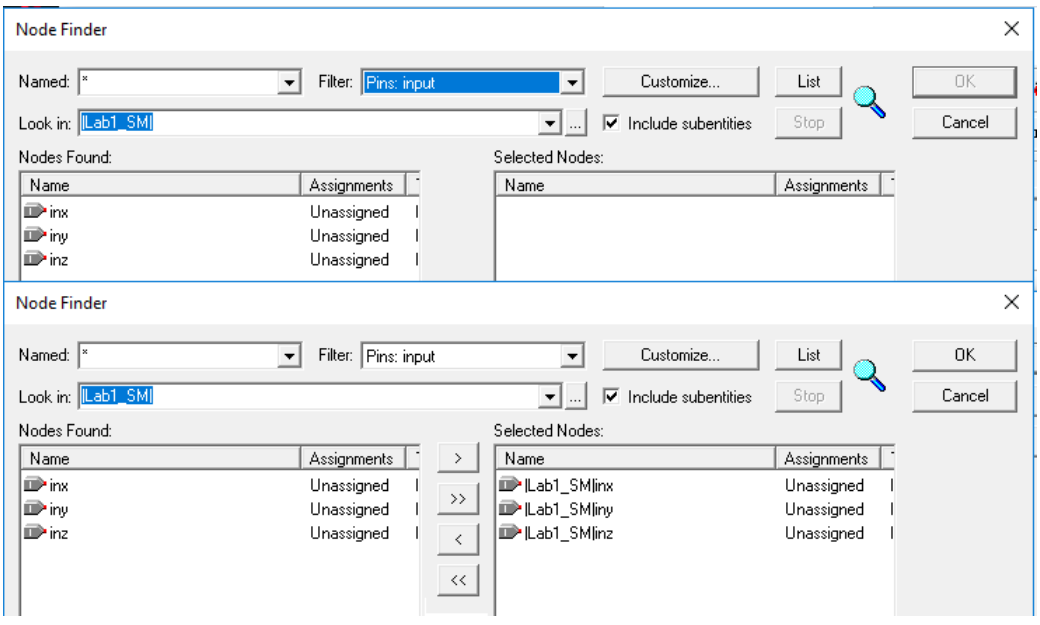
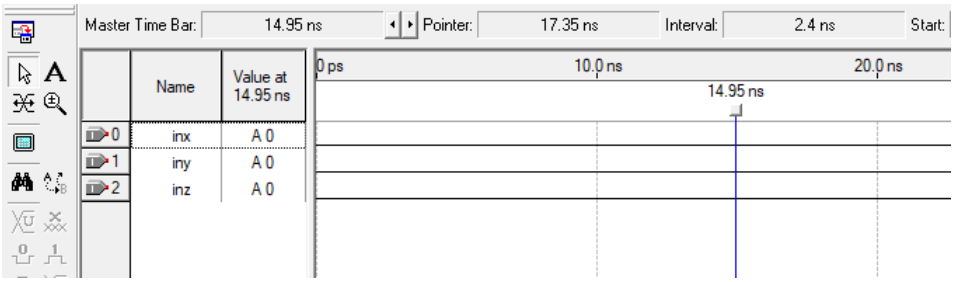
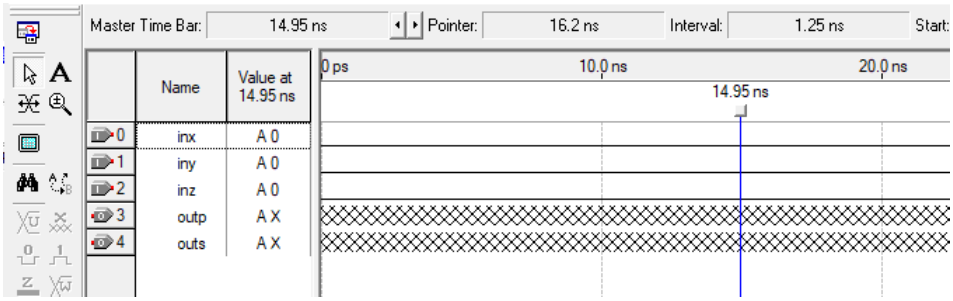


Рис. 4.3. Інструмент **Node Finder** для пошуку сигналів для моделювання



а



б

Рис. 4.4. Вхідні та вихідні сигнали для моделювання роботи схеми

1.3. Задайте інтервал для моделювання, для цього скористайтеся командою **End Time** із головного меню **Edit**. У вікні, що відкрилося, у полі **Time** задайте значення 1***ms*** (**s** – секунда, **ms** – мілі, **us** – мікро, **ns** – нано, **ps** – піко). Натисніть кнопку ОК. Загальний інтервал для моделювання встановиться в 1***ms***. За допомогою інструменту можна зменшувати або збільшувати масштаб часової діаграми: натискання лівої кнопки миші збільшує зображення, правої – зменшує.

1.4. Задайте значення вхідних сигналів для моделювання. Для цього оберіть інструмент і виберіть необхідний сигнал, натиснувши на його імені в полі **Name**. Для завдання значень сигналів використайте піктограми на панелі інструментів. Простим способом для завдання значень сигналів є виділення мишею необхідного часового проміжку і встановлення значення сигналі за допомогою інструментів на панелі інструментів. Змоделюємо роботу однорозрядного суматора. На часовій діаграмі на кожному проміжку часу тривалістю 10*ns* встановимо вхідні набори сигналів відповідно до таблиці істиності однорозрядного суматора. Для цього на лінії сигналу **inz** виділимо мишею проміжок часу від 40*ns* до 80*ns* тривалістю 40*ns* і за допомогою піктограми встановимо значення цього сигналу в 1 (рис. 4.4). Таким чином, на проміжку часу від 0*ns* до 40*ns* сигнал приймає значення 0, і від 40*ns* до 80*ns* – значення 1.

За допомогою інструменту **Count Value** на проміжку часу від 0*ns* до 80*ns* задайте періодичні значення двійкових сигналів від 0 до 1 на кожних 20*ns* лінії сигналу **inx** і на кожних 10*ns* лінії сигналу **iny**. (рис. 4.5, а, б). Отримана часова діаграма зображена на рис. 4.6.

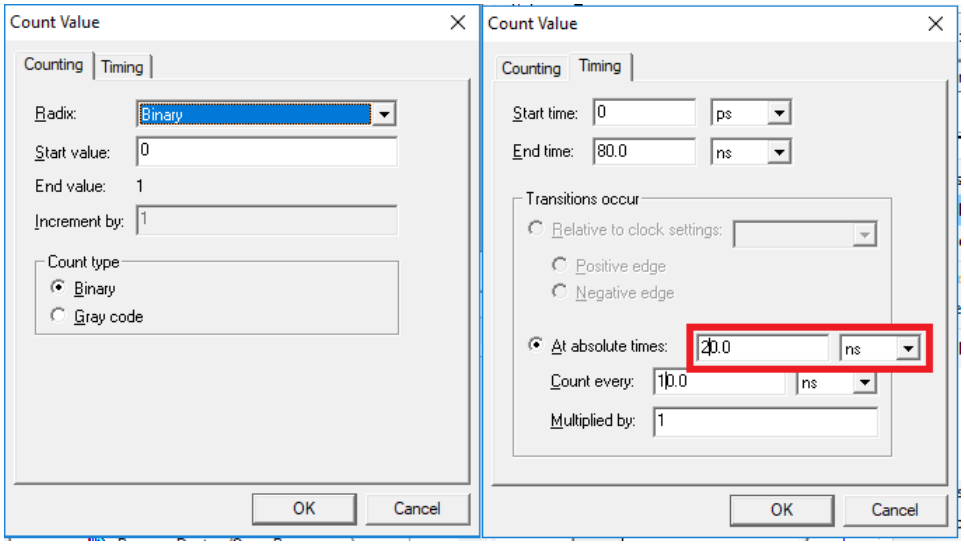


Рис. 4.5. Встановлення періодичних значень вхідних сигналів.

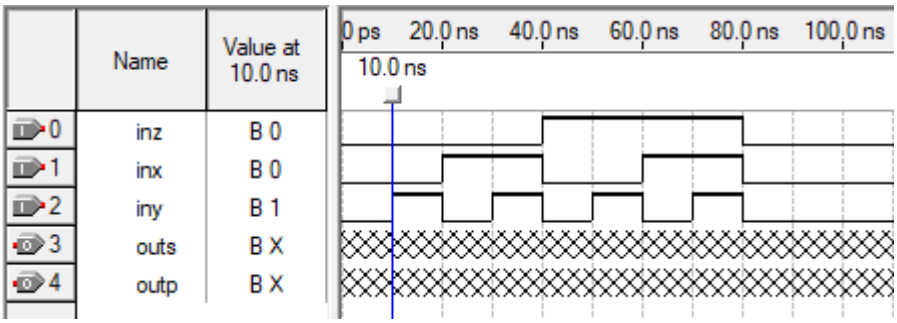


Рис. 4.6. Встановлення вхідних сигналів для моделювання на часовій діаграмі

1.5. Для запуску симуляції використайте піктограму в панелі задач, або виберіть команду **Start Simalation** в меню **Processing**. За успішно виконаного процесу симуляції відкривається відповідне повідомлення і відкривається вікно звіту симулятора **Simalation Report**. Результат виконаної симуляції проекту ***Lab2_SM_v2*** представлений на рис. 4.7.

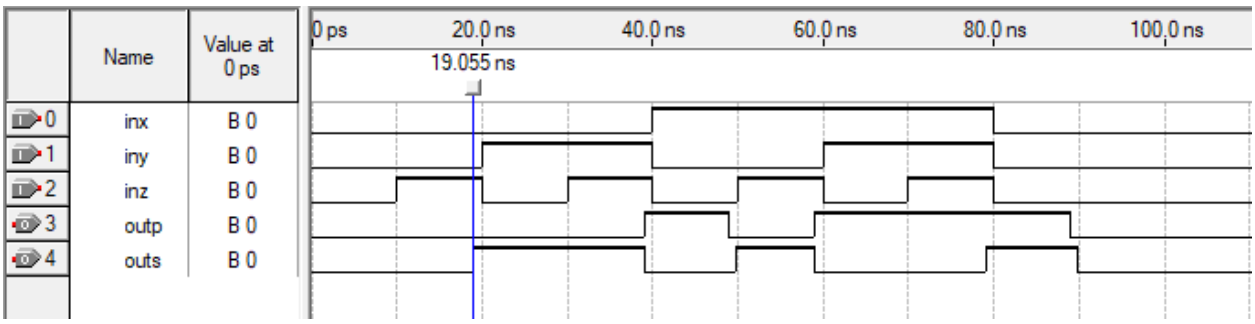


Рис. 4.7. Результати симуляції

Проаналізуйте правильність роботи суматора. На діаграмі (рис. 3.7) можна виміряти затримку формування результату – приблизно 9***ns***, що співпадає з теоретичними оцінками, які були сформовані під час компіляції (рис. 2.6).

Індивідуальне завдання

11. Виконайте симуляцію проекту ***Lab1_SM***. Для цього необхідно спочатку виконати етап функціонального моделювання і зробити висновки, чи правильно виконується цільова функція. Після чого виконати етап моделювання з врахуванням затримок передавання сигналів. Теоретичні відомості для виконання етапів функціонального (**Functional**) і часового (**Timing**) моделювання наведені в завданні ***TEST1***. Виконайте аналіз отриманої часової діаграми, визначить причини виникнення короткочасних помилкових сигналів (вкажіть ланцюги виникнення «просічок») та наведіть можливі засоби для подолання «просічок». Зверніть увагу, на те, що напівсуматор, реалізований на базі мегафункції не видає помилкових сигналів.

12. Виконайте експериментальну оцінку часу виконання операції. Порівняйте часові і апаратні витрати під час реалізації обох версій проекту. Оберіть найбільш ефективну для реалізації схему з точки зору мінімізації часових і апаратних витрат.

13. Створіть власний бібліотечний елемент – однорозрядний суматор ***bit_Name_SM*** (замість ***Name*** – Ваше прізвище) на базі розробленого Вами однорозрядного суматора з найкращими часовими і функціональними характеристиками.

14. Створіть новий проект ***Block_Name_SM***. Підключіть власний бібліотечний модуль ***bit_Name_SM*** під час створення проекту у другому вікні майстра створення проекту. На базі створених бібліотечних модулів ***bit_Name_SM*** розробіть багаторозрядний суматор. Розрядність суматора оберіть із таблиці 3.1. Виконайте компіляцію і симуляцію проекту.

15. Створіть новий проект ***Mega_Name_SM***. Розробіть багаторозрядний суматор на базі мегафункції. Виконайте компіляцію і симуляцію проекта. Порівняйте результати моделювання обох проектів (***Block_Name_SM*** і ***Mega_Name_SM)***. ***(Проект Mega_Name_SM є базовим для виконання л.р. №4)***

16. Зробіть загальні висновки

Таблиця 4.1. Розрядність суматора

Номер варіанта***	**Розрядність суматора**
000	16
001	15
010	14
011	12

100	11
101	10
110	9
111	8

*** Номер варіанта визначається по номерам розрядів номеру залікової книжки, який переведено в двійкову систему числення

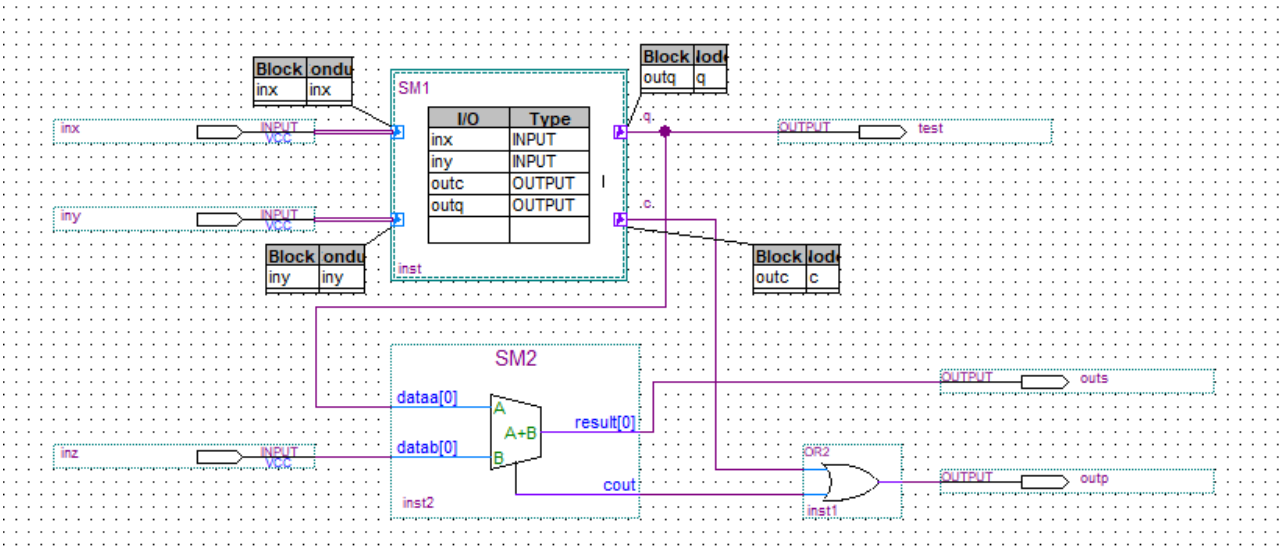
Таблиця 4.2.

Номер групи	**Розряди номера ЗК**
IO-*1	h1 h2 h3
IO-*2	h3 h2 h1
IO-*3	h2 h4 h1
IB-*1	h3 h1 h2
IB-*2	h4 h2 h3
IB-*3	h4 h2 h1

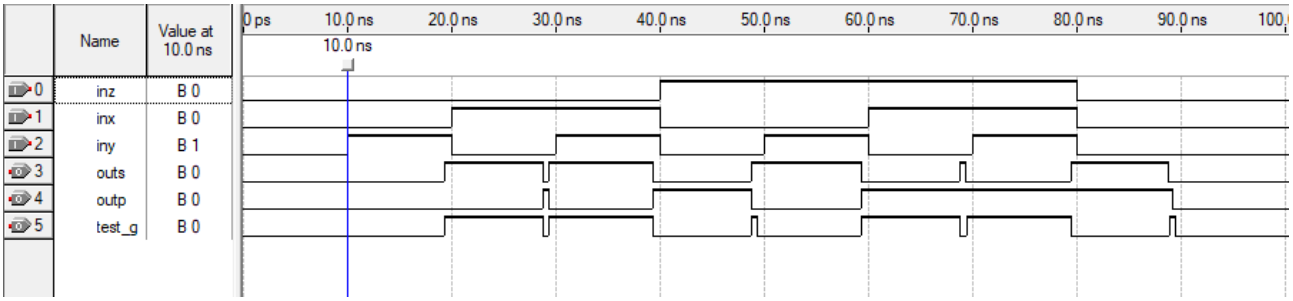
Рекомендації до виконання індивідуального завдання

Додавання вихідних контактів для тестування

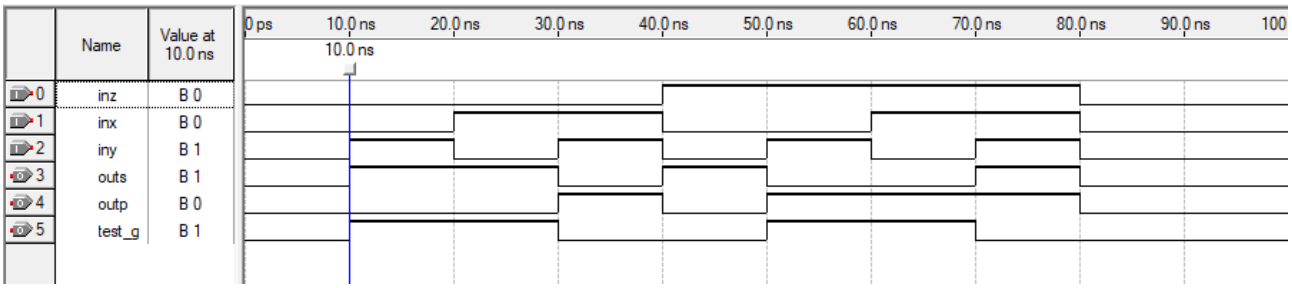
Для перегляду значень проміжних сигналів (додано контакт ***test***), але для пошуку причини просічки, треба вивести сигнали із вкладеного блоку (SM1). Застосуйте примітив **alt_buf** (повторювач, що знаходиться в бібліотеці стандартних компонентів) для подолання помилкових короткочасних сигналів, що виникли під час роботи схеми.



Результати часової симуляції (**Timing**) проекту ***Lab1_SM («просічки»)***



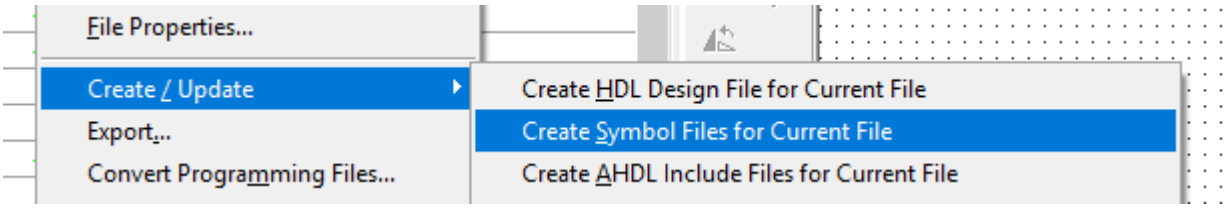
Результати функціональної симуляції (**Functional**) проекту ***Lab1_SM (ідеально)***



Детальні описи в роботі (але це не означає, що Ви не можете цього зробити!):

Створення власного бібліотечного компонента

1 спосіб: відкрити проект ***Lab1_SM_v2***, відкрити файл ***Lab2_SM_v2.bdf***. В меню File обрати команду **Create/Update** і далі команду **Create Symbol Files**.



Шукати створений компонент в бібліотеці стандартних компонентів в розділі Project. В директорії проекту вибрати всі файли, що мають назву створеного блока. Всі ці файли необхідно підключати до нових проектів на вкладці майстра створення нового проекту. Файли будуть прив'язані до нових проектів у вигляді шляхів до них. Архівувати такий проект тільки за допомогою архіватора Quartus.

2 спосіб. Створити аналогічну схему в новому проекті (в окремому файлі), після чого зробити із неї бібліотечний компонент, після чого використати його для створення файлів опису.

Підключення бібліотечних компонентів із інших проектів і бібліотек

Лабораторна робота № 5

РОЗРОБЛЕННЯ МОДУЛІВ ДЛЯ ВИКОНАННЯ АРИФМЕТИЧНИХ ОПЕРАЦІЙ В САПР *QUARTUS* II. ПРОГРАМУВАННЯ ПЛІС В САПР *QUARTUS* II

Ціль роботи:

- *Розробити модулі суматора/віднімача.*
- *Дослідження формування ознак в арифметичних блоках під час виконання арифметичних операцій.*
- *Нормалізація результату виконання арифметичних операцій.*
- *Вивчення особливостей функціональної побудови регістрів, як типових вузлів комп'ютера. Розроблення регістра в САПР Quartus II. Зсувні регістри.*
- *Призначення в Quartus II. Отримання навиків роботи з редактором призначень.*
- *Генерація файлу конфігурації засобами Compiler's Assembler Module в САПР Quartus II;*
- *Програмування ПЛІС. Використання JTAG інтерфейсу для завантаження файлу конфігурації в ПЛІС.*
- *Робота зі стендом DE2 Board Altera*

1. Порядок виконання роботи

1. Відкрийте проект ***Mega_Lab_SM***. Відкрийте головний файл опису проекту ***Mega_Lab_SM.bdf*** і відредагуйте мегафункцію ***Mega_SM***.

1.1. Відкрийте інструмент ***Symbol***. Відкрийте редактор мегафункцій кнопкой **MegaWizard Plug-In Manager** і виберіть прапорець **Edit an existing custom megafunction variation** (рис. 5.1).

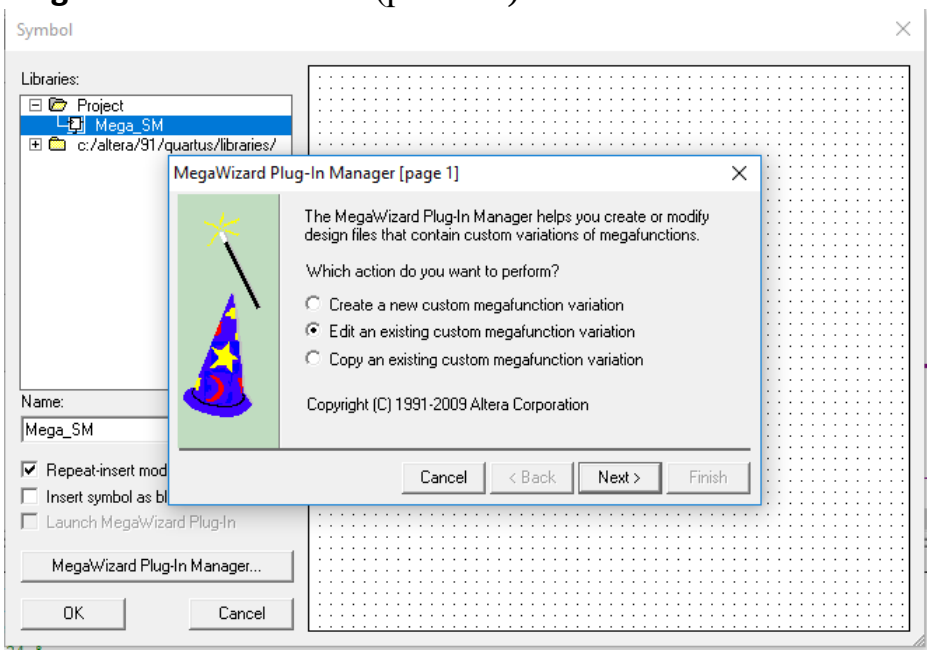


Рис. 5.1. Редактор мегафункцій

1.2. Додайте до модуля суматора керуючій вхід **add_sub** і **owerflow**. Збережіть проект.

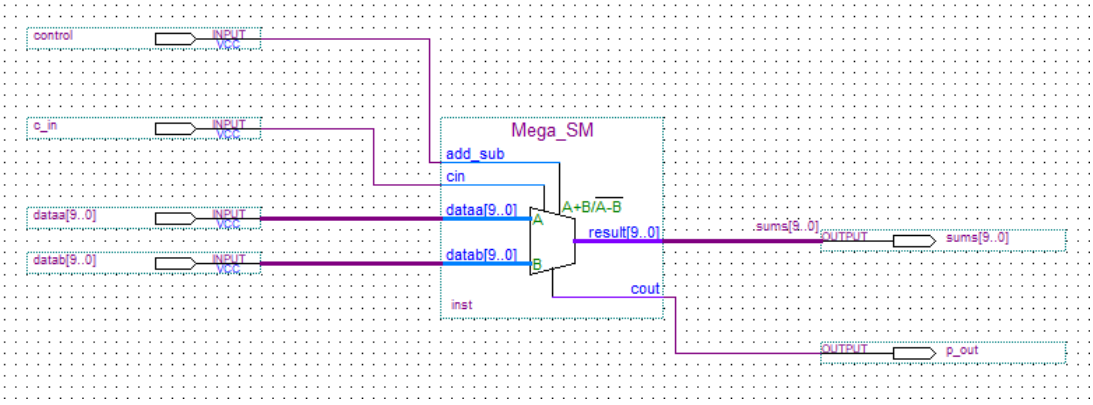


Рис. 5.2. Модуль суматора/віднімача на базі мегафункції в Quartus II

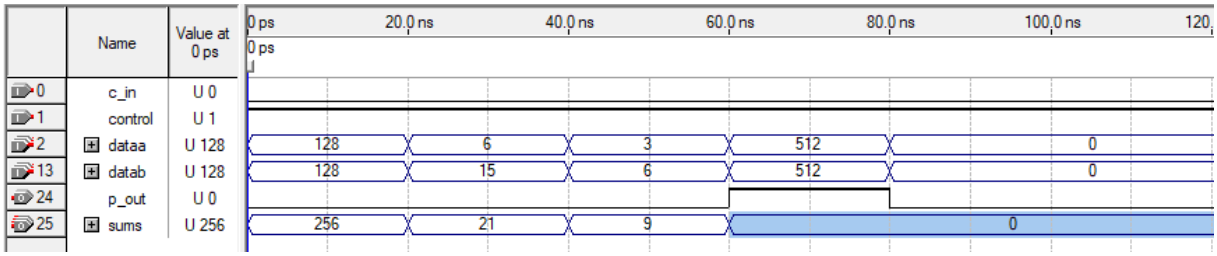
1.2. Виконайте моделювання роботи суматора/віднімача. Для аргументів оберіть двійковий формат. Вважайте, що два старших розряди є знаковими.

Дослідить формування ознак:

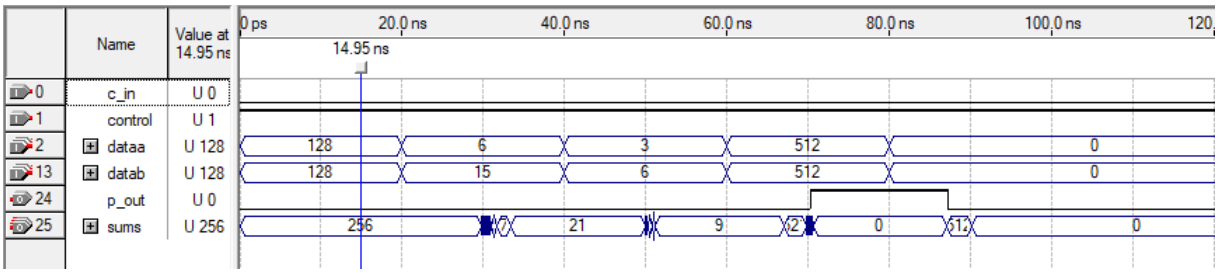
– переповнення розрядної сітки (*c*) - *cout*,
– переповнення в знакові розряди (*v*) - *overflow*.

Виведіть значення ознак на вихідні піни. Оцініть час затримки формування результату.

На рис. 5.3 зображено приклад часової діаграми виконання операції додавання. Зображено формування ознаки переповнення розрядної сітки.



а



б

Рис. 5.3. Результати моделювання: а – функціональне моделювання;
б – часове моделювання

1.3. Додайте до схеми пристрою буферний регістр для зберігання результату виконання операції додавання/віднімання (рис. 5.4.). Використайте зсувний регістр на базі мегафункції LPM_SHIFTREG.

Додайте до схеми керуючі сигнали:

– запису (*load*),

– скидання в «0» (*aclr*),
– встановлення в одиницю (a*set*),

Додайте ланцюги формування ознак:

– знак результату (n) – старший розряд результату,
– переповнення в знакові розряди (v) – ($ЗРст \oplus ЗРмл$) або вихід overflow мегафункції LPM_ADD_SUB,
– рівність результату нулю (z) – використайте логічний елемент АБО-НЕ.

Виведіть значення ознак (n, v, c, z) на вихідні піни.

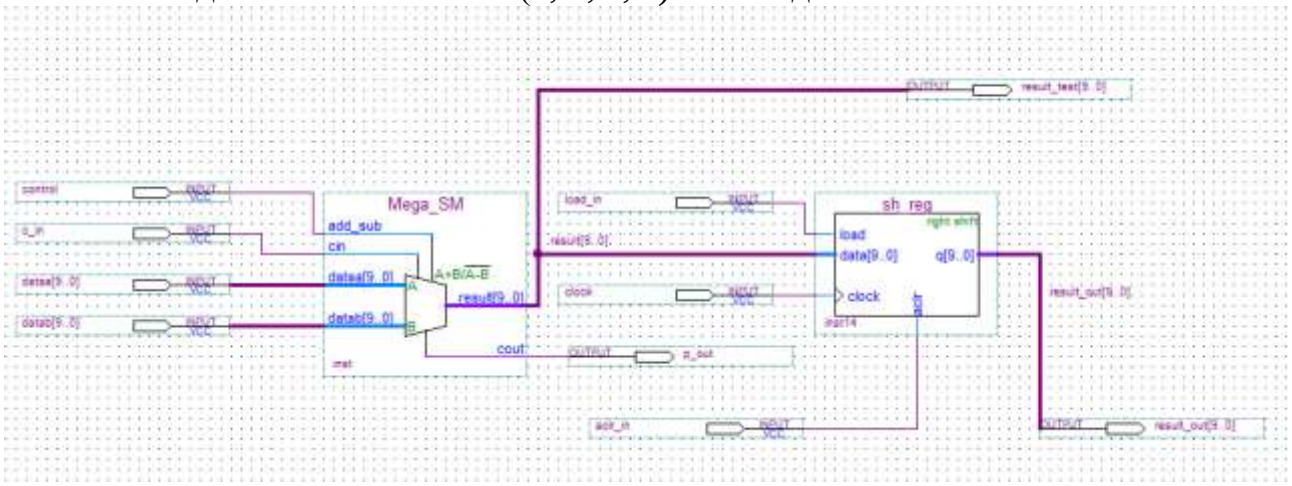


Рис. 5.4. Схема пристрою з буферним регістром

1.4. Виконайте моделювання. Оцініть час затримки формування результату. Оцініть результати формування ознак. Поясніть, як формуються ознаки під час виконання арифметичних операцій.

1.5. Дослідить процес функціонування регістру, як пристрою для зберігання інформації. Поясніть призначення керуючих сигналів. Документація на мегафункцію LPM_SHIFTREG наведена в Додатку В. Приклад часової діаграми дослідження функціонування регістру наведено на рис. 5.5. Показано, що результат виконання операції підсумовування зберігається я в буферному регістрі за фронтом керуючого сигналу *clock*.

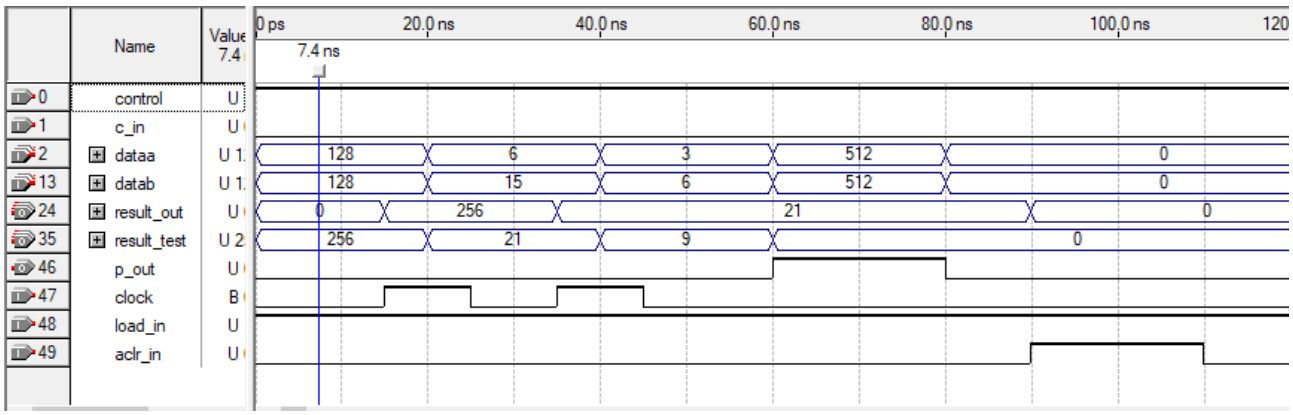


Рис. 5.5. Дослідження процесу функціонування регістру

1.6. Дослідить процес функціонування регістру, як зсувача. Відредагуйте мегафункцію, або створіть нову. Додайте керуючі сигнали керування зсувами. Виконайте моделювання з реалізацією зсувів. Змоделюйте ситуацію порушення нормалізації мантис ліворуч, або праворуч за ознаками, що формуються під час

додавання мантис. Виконайте корекцію результату за допомогою зсувів на зсуваючому регістрі.

1.7. Виконайте контрольне завдання, задане викладачем. Приклади контрольних завдань наведені в кінці лабораторної роботи.

2. Тестування пристрою на налагоджувальному стенді *DE*2.

Для тестування реалізуйте введення вихідних данных и формування результату на системі індикації стенду.

> На налагоджувальному стенді DE2 передбачені такі технічні засоби, як кнопки, вимикачі, світлодіоди, *LCD* – дисплей та ін.
>
> Документація по використанню налагоджувального стенду наведено в ДОДАТКУ Б до лабораторної роботи.

2.1. Встановіть на комп'ютер *USB* драйвер (*USB-Blaster driver*) для програматора ПЗП на платі.

> Стислий опис налагоджувального стенду і процес встановлення *USB* драйверу наведено в ДОДАТКУ А до лабораторної роботи.

2.2. Виконайте призначення контактів мікросхеми.

> Вхідним та вихідним контактам (пінам) розробленого пристрою необхідно призначити виводи кристалу ПЛІС, які вже певним чином розведені до відповідних пристроїв на стенді.
>
> Номери контактів ПЛІС *Cyclon* II та їх призначення описані в документації по використанню стенду [2].
>
> У таблиці в Додатку Б до лабораторної роботи приведені співвідношення між пристроями на платі (вимикачами ***SW*[0]** – ***SW*[17]**, світлодіодами ***LEDR*[0]** – ***LED*[17]** та ***LEDG*[0]** – ***LEDG*[7]**, кнопками ***KEY*[0]** – ***KEY*[3]**, тощо) і номерами контактів ПЛІС ***Cyclon* II**.
>
> Для призначення контактів використовують утиліту **Pin Planer**, яку можна запустити командою **Pin Planer** із меню **Assignment** або піктограмою на панелі інструментів.
>
> У вікні редактора **Pin Planer** ліворуч зверху перелічені групи пінів із схемних файлів проекту, праворуч показано умовне зображення матриці контактів ПЛІС (кольорами виділені банки контактів). В нижній частині редактору представлений інструмент для призначення реальних контактів кристалу ПЛІС віртуальним контактам пристроїв у схемних файлах проекту. Для призначення схемному піну певного контакту кристалу ПЛІС потрібно у таблиці співвідношень для схемного піна із стовбця **Node Name** прописати номер контакту кристалу ПЛІС в стовпець **Location**.

Наприклад, для того, щоб виводити стан біту ***RZ*[0]** на світлодіод ***LEDR*[0]**, потрібно піну пристрою у рядку ***RZ*[0]** стовбця **Node Name** призначити контакт ***PIN_AE*23** в стовбці **Location**.

Для тестування роботи пристрою використайте:

- **4 кнопкові перемикачі *KEY*[0] – *KEY*[3]**
- **18 тумблерів-перемикачів *SW*[0] – *SW*[17]**
- **18 червоних LED- індикаторів *LEDR*[0] – *LED*[17]**
- **9 зелених LED-індикаторів *LEDG*[0] – *LEDG*[7]**

Наприклад, для вводу аргументів використайте тумблери-перемикачі, для виводу результату – червоні LED-індикатори, для виводу ознак зелені LED-індикатори. Для формування керуючих впливів – кнопочні перемикачі.

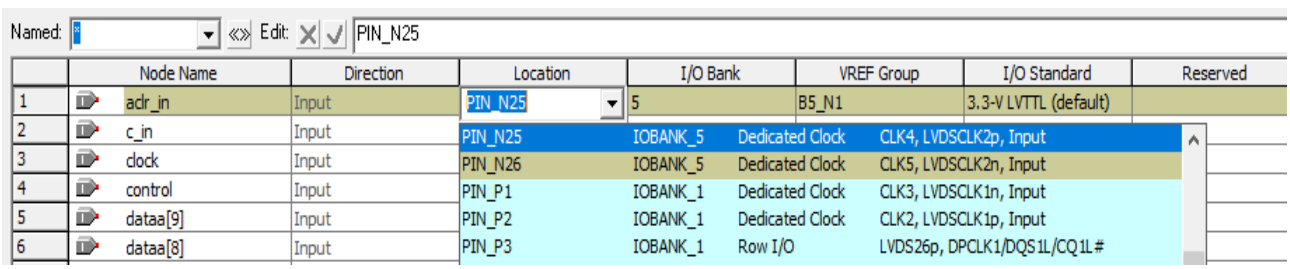
Випишіть із Додатку Б в таблицю 5.1 співвідношення між використовуваними пристроями введення/виведення на стенді і контактами кристалу ПЛІС ***Cyclon* II**.

Таблиця 5.1

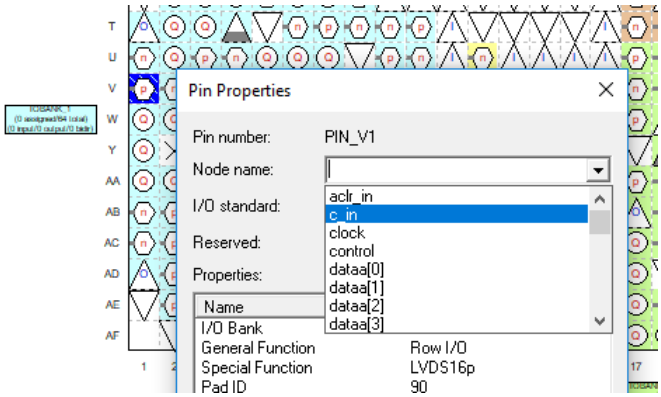
Приклад призначення контактів

Вивід схемного файлу	**Пристрій на стенді**	**Номер контакту мікросхеми**	**Коментар**
c_in	*SW*[17]	PIN_V2	Вхідний перенос суматора
control	*SW*[16]	PIN_V1	Додавання *control*=1
load_in	*SW*[15]	PIN_U4	
aclr_in	*SW*[0]	PIN_N25	Скидання регістру *reset*
aset_in	*SW*[1]	PIN_N26	Встановлення в одиницю *set*
clock	*KEY*[0]	PIN_G26	Синхросигнал
dataa[6..0]	*SW*[7] – *SW*[2]		Вхідна 6-розрядна шина *A*
datab[6..0]	*SW*[13] – *SW*[8]		Вхідна 6-розрядна шина *B*
result_out[6..0]	*LEDR*[5] – *LEDR*[0]		Вихідна 6-розрядна шина *S*
p_out	*LEDG*[0]	PIN_AE22	Ознака переповнення за межі розрядної сітки

В редакторі **Pin Planer** виконайте призначення контактів (рис. 5.6).



а



б

Рис. 5.6. Призначення контактів в редакторі **Pin Planer**: *а* – пошук контактів у списку, *б* – пошук контакту за адресою в матриці контактів, прив'язка до піну у схемі

2.3. Виконайте компіляцію проекту. Перевірьте правильність призначення контактів у звіті компілятора в секції *Resourse Section* (*Input Pin*, *Output Pin*)

2.4. JTAG завантаження конфігурації в ПЛІС зображено на рис. 4.1. документації в додатку Б. Для завантаження виконайте наступні підготовчі кроки:

- підключить живлення для керування стендом *DE2*;
- з'єднайте ***USB*** кабель з ***USB*** портом на стенді *DE2* (див. рис. 2.1 документації);
- забезпечте режим ***JTAG*** програмування встановивши вимикач ***RUN/PROG*** (з лівого боку стенду) в позицію ***RUN***;
- тепер *FPGA* можна програмувати за допомогою електронного програматора, якщо вибрати конфігураційний файл з розширенням ****.sof***.

2.5. Створіть конфігураційний файл і запрограмуйте ПЛІС за допомогою електронного програматора, для цього виберіть команду **Programmer** із головного меню **Tools**. У вікні електронного програматора, що відкрилося, натисніть кнопку **Hardware Setup**. Якщо драйвер був встановлений правильно у випадаючому списку **Currently selected Hardware** виберіть **Usb_blaster**. Натисніть кнопку **Close**. У вікні програматора замість напису **No hardware** повинно з'явитись **Usb_blaster** і стати активною кнопка **Start**. У полі **Mode** виберіть режим **JTAG**. У полі знаходиться назва файлу з розширенням ***.sof** , який по замовчанню має ім'я проекту. Виділить файл і натисніть кнопку **Start**.

2.6. Виконайте тестування розробленого пристрою на налагоджувальному стенді ***Altera DE2 board***.

3. Зробіть загальні висновки по роботі

Контрольні завдання

Завдання 1.

Знайти суму S або різницю D двох мантис A_i та B_j. Операції машинного додавання та віднімання виконати в обереному або доповнювальному кодах.

Знайти $S_1=A_1+B_1$ $D_1=A_1-B_1$, якщо $A_1 = 0110001110$
$S_2=A_1+B_2$ $D_2=A_1-B_2$ $A_2 = -0110001110$
$S_3=A_2+B_1$ $D_3=A_2-B_1$ $B_1 = 1010010111$
$S_4=A_2+B_2$ $D_4=A_2-B_2$ $B_2 = -1010010111$

Виконайте тестування роботи пристрою на заданих прикладах (виділено жовтим). Сформуйте та дослідить ознаки результату (виділено зеленим). За необхідністю виконайте нормалізацію мантис результату.

$S_1 = A_1 + B_1$

RGA	00	0110001110	$[A_1]_{пк}$
RGB	00	1010010111	$[B_1]_{пк}$
+	00	0110001110	$[A_1]_{ок}$
	00	1010010111	$[B_1]_{ок}$
SM	01	0000100101	$[S_1]_{ок}$

Додатне переповнення

Отриманий результат є денормалізованим вліво, про що свідчить незбіжність цифр у знакових розрядах мантиси.
Нормалізація результату. Зсуваємо мантису вправо на один розряд

$S_2 = A_1 + B_2$

RGA	00	0110001110	$[A_1]_{пк}$
RGB	11	1010010111	$[B_2]_{пк}$
+	00	0110001110	$[A_1]_{ок}$
	11	0101101000	$[B_2]_{ок}$
SM	11	1011110110	$[S_2]_{ок}$

$S_3 = A_2 + B_1$

RGA	11	0110001110	$[A_2]_{пк}$
RGB	00	1010010111	$[B_1]_{пк}$
+	11	1001110001	$[A_2]_{ок}$
	00	1010010111	$[B_1]_{ок}$
SM	00	0100001000	
		+1	
SM	00	0100001001	$[S_3]_{ок}$

$S_4 = A_2 + B_2$

RGA	11	0110001110	$[A_2]_{пк}$
RGB	11	1010010111	$[B_2]_{пк}$
+	11	1001110001	$[A_2]_{ок}$
	11	0101101000	$[B_2]_{ок}$
SM	10	1111011001	
		+1	
SM	10	1111011010	$[S_4]_{ок}$

Від'ємне переповнення

Виконаємо операцію машинного віднімання в оберненому коді:

$D_1 = A_1 - B_1$

RGA	00	0110001110	$[A_1]_{пк}$
RGB	11	1010010111	$[-B_1]_{пк}$
+	00	0110001110	$[A_1]_{ок}$
	11	0101101000	$[-B_1]_{ок}$
SM	11	1011110110	$[D_1]_{ок}$

$D_2 = A_1 - B_2$

RGA	00	0110001110	$[A_1]_{пк}$
RGB	00	1010010111	$[-B_2]_{пк}$
+	00	0110001110	$[A_1]_{ок}$
	00	1010010111	$[-B_2]_{ок}$
SM	01	0000100101	$[D_2]_{ок}$

Додатне переповнення

$D_3 = A_2 - B_1$

RGA	11	0110001110	$[A_2]_{пк}$
RGB	11	1010010111	$[-B_1]_{пк}$
+	11	1001110001	$[A_2]_{ок}$
	11	0101101000	$[-B_1]_{ок}$
SM	10	1111011001	
		+1	
SM	10	1111011010	$[D_3]_{ок}$

$D_4 = A_2 - B_2$

RGA	11	0110001110	$[A_2]_{пк}$
RGB	00	1010010111	$[-B_2]_{пк}$
+	11	1001110001	$[A_2]_{ок}$
	00	1010010111	$[-B_2]_{ок}$
SM	00	0100001000	
		+1	
SM	00	0100001001	$[D_4]_{ок}$

Від’ємне переповнення

Виконаємо операцію машинного додавання в доповнювальному коді:

$S_1 = A_1 + B_1$

RGA	00	0110001110	$[A_1]_{пк}$
RGB	00	1010010111	$[B_1]_{пк}$
+	00	0110001110	$[A_1]_{дк}$
	00	1010010111	$[B_1]_{дк}$
SM	01	0000100101	$[S_1]_{дк}$

$S_2 = A_1 + B_2$

RGA	00	0110001110	$[A_1]_{пк}$
RGB	11	1010010111	$[B_2]_{пк}$
+	00	0110001110	$[A_1]_{дк}$
	11	0101101001	$[B_2]_{дк}$
SM	11	1011110111	$[S_2]_{дк}$

Додатне переповнення

$S_3 = A_2 + B_1$

RGA	11	0110001110	$[A_2]_{пк}$
RGB	00	1010010111	$[B_1]_{пк}$
+	11	1001110010	$[A_2]_{дк}$
	00	1010010111	$[B_1]_{дк}$
SM	00	0100001001	$[S_3]_{дк}$

$S_4 = A_2 + B_2$

RGA	11	0110001110	$[A_2]_{пк}$
RGB	11	1010010111	$[B_2]_{пк}$
+	11	1001110010	$[A_2]_{дк}$
	11	0101101001	$[B_2]_{дк}$
SM	10	1111011011	$[S_4]_{дк}$

Від’ємне переповнення

Отриманий результат є денормалізованим вправо, про що свідчить збіжність цифр знакового і старшого розряду мантиси. Нормалізація результату. Зсуваємо мантису вліво на один розряд

Виконаємо операцію машинного віднімання в доповнювальному коді:

$D_1 = A_1 - B_1$

RGA	00	0110001110	$[A_1]_{пк}$
RGB	11	1010010111	$[-B_1]_{пк}$
+	00	0110001110	$[A_1]_{дк}$
	11	0101101001	$[-B_1]_{дк}$
SM	11	1011110111	$[D_1]_{дк}$

$D_2 = A_1 - B_2$

RGA	00	0110001110	$[A_1]_{пк}$
RGB	00	1010010111	$[-B_2]_{пк}$
+	00	0110001110	$[A_1]_{дк}$
	00	1010010111	$[-B_2]_{дк}$
SM	01	0000100101	$[D_2]_{дк}$

Додатне переповнення

$D_3 = A_2 - B_1$

RGA	11	0110001110	$[A_2]_{пк}$
RGB	11	1010010111	$[-B_1]_{пк}$
+	11	1001110010	$[A_2]_{дк}$
	11	0101101001	$[-B_1]_{дк}$
SM	10	1111011011	$[D_3]_{дк}$

$D_4 = A_2 - B_2$

RGA	11	0110001110	$[A_2]_{пк}$
RGB	00	1010010111	$[-B_2]_{пк}$
+	11	1001110010	$[A_2]_{дк}$
	00	1010010111	$[-B_2]_{дк}$
SM	00	0100001001	$[D_4]_{дк}$

Від’ємне переповнення

Завдання 2.

Одержати суму чисел A і B із плаваючою комою, поданих у прямому коді:

1. $A = 0\ 011\ 1\ 10101$, $B = 0\ 101\ 0\ 11001$;
2. $A = 0\ 101\ 1\ 10101$, $B = 0\ 011\ 0\ 11001$;
3. $A = 0\ 101\ 1\ 10101$, $B = 0\ 100\ 1\ 11001$.

Виконайте тестування роботи пристрою на заданих прикладах (виділено жовтим). Сформуйте та дослідить ознаки результату (виділено зеленим). За необхідністю виконайте нормалізацію мантис результату.

Виконання завдання

1) Знаходимо різницю порядків $B - A = 101 - 011 = 010$. Виконуємо зрівняння порядків заданих чисел, при цьому менший порядок числа A (011) збільшується до більшого порядку числа B (101), а мантиса числа A зсувається на два розряди вправо. Одержуємо:

$$A = 0\ 101\ 1\ 0010101,$$
$$B = 0\ 101\ 0\ 11001.$$

Подамо мантиси чисел у доповнювальному коді і виконаємо додавання мантис:

$$+\ \begin{array}{ll} M_{[A']ДК} & 11,\ 1101011 \\ M_{[B]ДК} & 00,\ 1100100 \\ \hline M_{[C]ДК} & 00,\ 1001111\ . \end{array}$$

Отриманий результат є нормалізованим, тому лише подамо отримане число у прямому коді і виконаємо його округлення до заданої кількості розрядів мантиси.

Відповідь: $C = 0\ 101\ 0\ 10100$.

2) Знаходимо різницю порядків $B - A = 011 - 101 = -010$. Виконуємо зрівняння порядків заданих чисел, при цьому менший порядок числа B (011) збільшується до більшого порядку числа A (101), а мантиса числа B зсувається на два розряди вправо. Одержуємо:

$$A = 0\ 101\ 1\ 10101,$$
$$B = 0\ 101\ 0\ 0011001.$$

Подамо мантиси чисел у доповнювальному коді і виконаємо додавання мантис:

$$+\ \begin{array}{ll} M_{[A]ДК} & 11,\ 0101100 \\ M_{[B']ДК} & 00,\ 0011001 \\ \hline M_{[C]ДК} & 11,\ 1000101\ . \end{array}$$

Отриманий результат є денормалізованим вправо, про що свідчить збіжність цифр знакового і старшого розряду мантиси.

Нормалізація результату. Зсуваємо мантису вліво на один розряд (11,000101) і виконуємо корекцію порядку (101 – 1 = 100). Подаємо результат у прямому коді (11,111011) і виконуємо його округлення (11,11110).

Відповідь: $C = 0\ 100\ 1\ 11101$.

3) Знаходимо різницю порядків $B - A = 100 - 101 = -001$. Виконуємо зрівняння порядків заданих чисел, при цьому менший порядок числа B (100) збільшується до більшого порядку числа A (101), а мантиса числа B зсувається на один розряд вправо. Одержуємо:

$$A = 0\ 100\ 1\ 10101,$$
$$B = 0\ 101\ 1\ 011001.$$

Подамо мантиси чисел у доповнювальному коді і виконаємо додавання мантис:

$$+\ \begin{array}{ll} M_{[A]ДК} & 11,\ 010110 \\ M_{[B']ДК} & 11,\ 100111 \\ \hline M_{[C]ДК} & 10,\ 111101\ . \end{array}$$

Отриманий результат є денормалізованим вліво, про що свідчить незбіжність цифр у знакових розрядах мантиси.

Нормалізація результату. Зсуваємо мантису вправо на один розряд (11, 0111101) і виконуємо корекцію порядку (101+1=110). Подаємо мантису у прямому коді і виконуємо його округлення (1,10000).

Відповідь: $C = 0\ 110\ 1\ 10000$.

Лабораторна робота № 6

РОЗРОБЛЕННЯ МОДУЛІВ АРИФМЕТИКО-ЛОГІЧНИХ БЛОКІВ В САПР *QUARTUS* II. ВИВІД РЕЗУЛЬТАТІВ ВИКОНАННЯ ОПЕРАЦІЙ НА СЕМИСЕГМЕНТНІ ІНДИКАТОРИ

Ціль роботи:

- *Розроблення та дослідження роботи арифметико-логічних блоків з загальними мікрооперaціями.*
- *Дослідження формування ознак в арифметичних блоках з загальними мікрооперaціями.*
- *Нормалізація результату виконання арифметичних операцій.*
- *Вивчення особливостей функціональної побудови мультиплексорів, як типових вузлів комп'ютера. Розроблення мультиплексора в САПР Quartus II.*
- *Робота зі стендом DE2 Board Altera*
- *Вивід результатів виконання мікрооперацій на семисегментні індикатори*

Теоретичні відомості

Арифметико-логічні блоки з загальними мікрооперaціями

Приклад функціональної схеми арифметико-логічного блоку з загальними мікрооперaціями (АЛБ) наведений на рис. 5.1. Алгоритм роботи АЛБ наведений далі.

На керуючий вхід ***I*** АЛБ надходить керуюча мікроінструкція (АЛБ_МІ) $I_5I_4I_3$. Приклад кодування мікрооперацій мікроінструкцією $I_5I_4I_3$ наведено в табл. 6.1.

Таблиця 6.1.

Мікроінструкція $I_5I_4I_3$

I_5	I_4	I_3	Мікрооперація в *ALB*
0	0	0	$R+S+CI$
0	0	1	$S-R-1+CI$
0	1	0	$R-S-1+CI$
0	1	1	$R \vee S$
1	0	0	$R \& S$
1	0	1	$\overline{R} \& S$
1	1	0	$R \oplus S$
1	1	1	$\overline{R \oplus S}$

АЛБ_МІ $I_5I_4I_3$ керує мультиплексором *MX* блока АЛБ, відповідно до табл. 5.1. На функціональній схемі АЛБ (рис. 5.1) на входи *MR* і *MS* надходять аргументи, які поступають на входи *R* і *S* всіх функціональних вузлів АЛБ. *CI* – вхід переносу, виходи *CO, VO, NO, ZO* – виходи ознак які формуються під час виконання мікрооперацій в АЛБ (відповідно: переносу, переповнювання в знакові розряди, знаку результату, рівності нулю результату), *F_ALB* – вихід результату виконуваної мікрооперації. В АЛБ всі мікрооперації на всіх функціональних вузлах виконуються паралельно. Для формування результату мікрооперацій та ознак на виході блоку *ALB* використовується мультиплексор *MX*, який відповідно до керуючої мікроінструкції на його керуючому вході ***sel*** комутує результат з виходу відповідного функціонального вузла на вихід *F_ALB* АЛБ.

Мікрооперації підсумовування і віднімання виконуються у доповняльному коді з урахуванням вхідного переносу *CI*. Логічні мікрооперації – диз'юнкція (АБО), кон'юнкція (І) і сума за модулем два (ВИКЛЮЧНЕ АБО) є порозрядними.

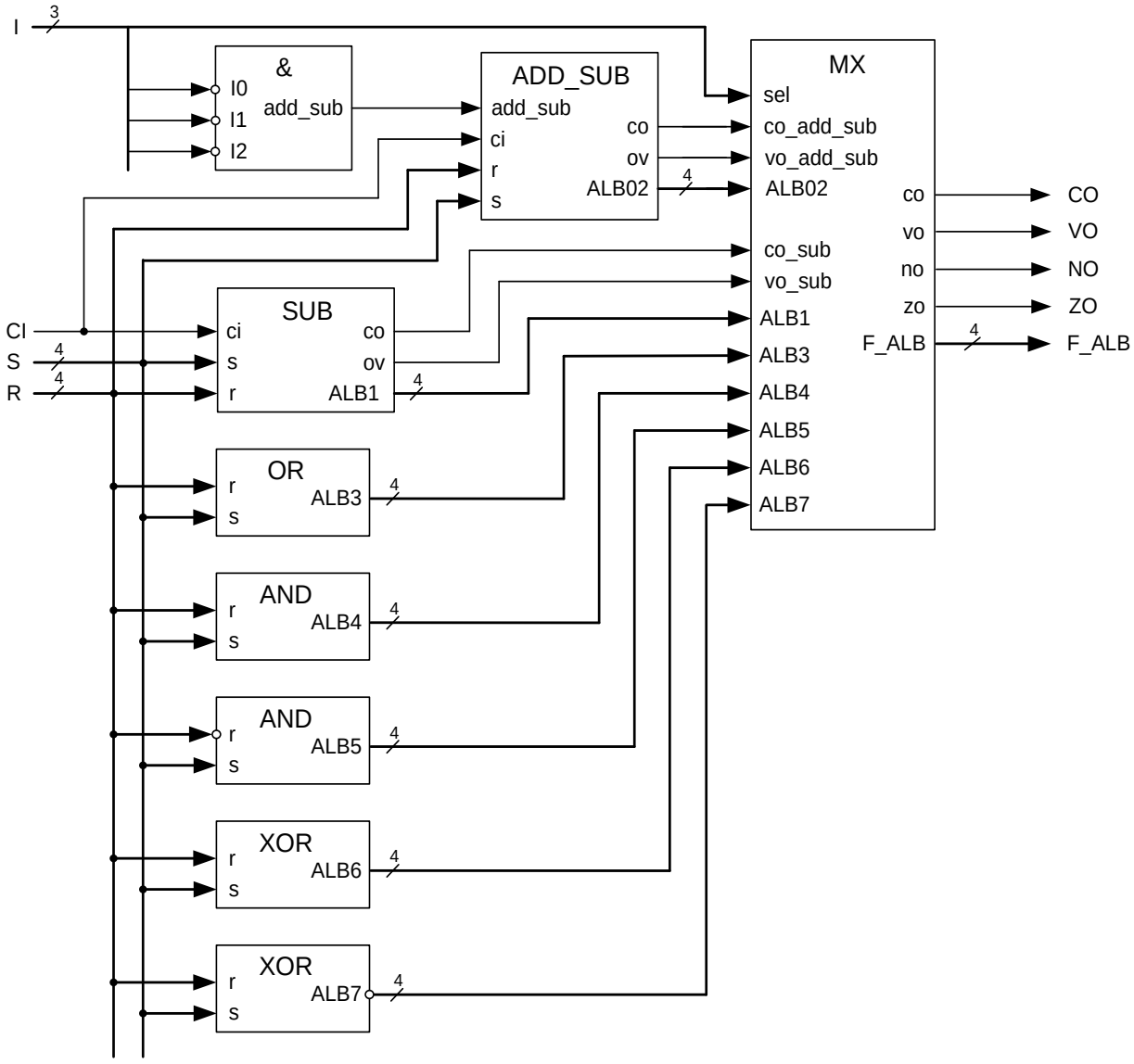


Рис. 6.1. Функціональна схема арифметико-логічного блоку з загальними мікрооперaціями

1. Порядок виконання роботи

1. Створіть новий проект ***Mega_Lab_ALB***. В даній лабораторній роботі Ви будете модифікувати схему із попередньої лабораторної роботи. Для цього Ви можете використати будь які способи перенесення вже розробленої ріниш схеми пристрою в новий проект. Наприклад, зкопіюйте головний файл опису проекту ***Mega_Lab_SM.bdf*** в папку проекту ***Mega_Lab_ALB***, перейменуйте його в файл ***Mega_Lab_ALB.bdf*** та призначте головним файлом опису проекту ***Mega_Lab_ALB***. Або Ви можете створити в проекті ***Mega_Lab_ALB*** головний файл опису проекту ***Mega_Lab_ALB.bdf***, після чого безпосередньо зкопіювати схему суматора із робочого поля файлу ***Mega_Lab_SM.bdf*** проекту ***Mega_Lab_SM*** в робоче поле файлу ***Mega_Lab_ALB.bdf***. За бажанням, Ви можете створити нову схему пристрою.

2. Розробіть АЛБ для виконання заданих у табл. 5.2. мікрооперацій. Це можна зробити шляхом модифікацій вже розробленого суматора-віднімача (див. п.1) згідно варіантам наведеним у табл. 6.2, або створення нової схеми.

Таблиця 6.2.

Вибір варіанту для розроблення АЛБ

h_2	h_3	h_1	Мікроінструкція АЛБ_МІ I_4I_3			
			00	**01**	**10**	**11**
0	0	0	$R-S-1+CI$	$R \oplus S$	$R \vee S$	$R+S+CI$
0	0	1	$\overline{R \oplus S}$	$R \vee S$	$R+S+CI$	$S-R-1+CI$
0	1	0	$R \vee S$	$R+S+CI$	$\overline{R} \& S$	$R-S-1+CI$
0	1	1	$R-S-1+CI$	$R \& S$	$R+S+CI$	$R \vee S$
0	0	0	$S-R-1+CI$	$R \vee S$	$R+S+CI$	$R \& S$
0	0	1	$R+S+CI$	$S-R-1+CI$	$R \vee S$	$\overline{R} \& S$
0	1	0	$\overline{R} \& S$	$\overline{R \oplus S}$	$R+S+CI$	$R-S-1+CI$
0	1	1	$S-R-1+CI$	$R \vee S$	$R+S+CI$	$\overline{R \oplus S}$
1	0	0	$R+S+CI$	$S-R-1+CI$	$R \& S$	$R \vee S$
1	0	1	$R \oplus S$	$R+S+CI$	$\overline{R} \& S$	$R-S-1+CI$
1	1	0	$R \vee S$	$R \oplus S$	$S-R-1+CI$	$R+S+CI$
1	1	1	$R+S+CI$	$R \& S$	$R \vee S$	$R-S-1+CI$
1	0	0	$R \vee S$	$R+S+CI$	$S-R-1+CI$	$R \vee S$
1	0	1	$R \oplus S$	$R+S+CI$	$R-S-1+CI$	$\overline{R} \& S$
1	1	0	$S-R-1+CI$	$R \& S$	$R \vee S$	$R+S+CI$
1	1	1	$R \vee S$	$R+S+CI$	$S-R-1+CI$	$\overline{R \oplus S}$

2.1. Введіть у проект функціональний модуль АЛБ структура, якого наведена на рис 6.2.

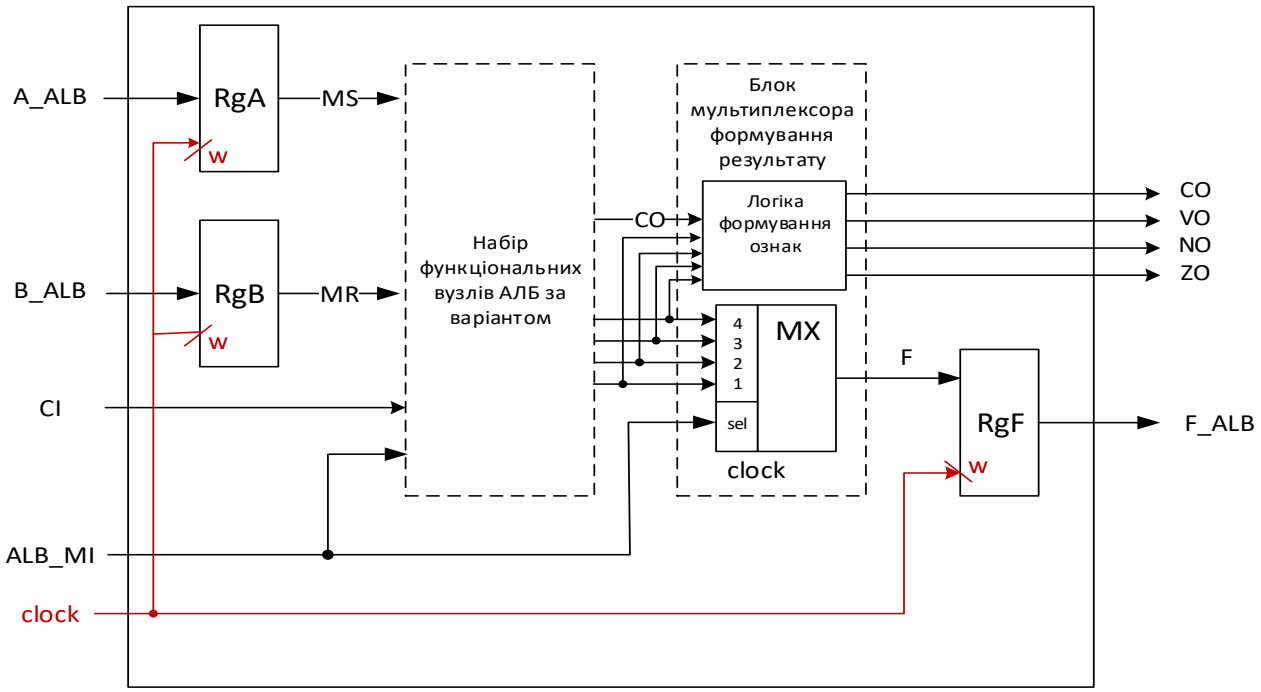


Рис. 6.2. Структурно-функціональна схема проекту ***Mega_Lab_ALB***

2.2. Для зберігання аргументів і результатів передбачте буферні регістри. Забезпечте виконання мікрооpеацій додавання, віднімання, логічних функцій за один такт роботи пристрою, для цього здійсніть запис аргументів у буферні регістри за фронтом синхросигналу, а запис результату за зрізом синхросигналу.

2.3. Для створення функціональних вузлів АЛБ використайте засоби інструменту для створення мегафункцій **MegaWizard Plug-In Manager**

3. Виконайте етапи функціонального та часового моделювання роботи АЛБ. **3.1.** Дослідіть часові параметри розробленого АЛБ і оберіть тривалість синхросигналу, таку що відповідає часу формування результатів на виходах пристрою.

3.2. Після тестування моделі АЛБ в САПР ***Quartus*** **II** створіть бібліотчний модуль АЛБ і збережіть його в користувацькі бібліотеці.

4. Виконайте тестування роботи АЛБ на налагоджувальному стенді DE2.

5. Виведіть результати роботи АЛБ на семисегментні індикатори.

5.1. Інструкція для підключення виводів розроблюваних пристроїв до семисегментних індикаторів представлена у **Додатку Г.**

5.2. Виконайте тестування роботи АЛБ, подаючи різні керуючі мікроінструкції. Виконайте тестування формування ознак.

6. Зробіть висновки по роботі.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

Базова

1. Жабін В.І., Жуков І.А., Клименко І.А., Ткаченко В.В. Прикладна теорія цифрових автоматів: Навч. посібник. – К.: Вид-во НАУ, 2009. – 364 с.
2. Матвієнко М.П., Розан В.П. Комп'ютерна схемотехніка. Навчальний посібник. – Київ: Видавництво Ліра – К, 2016. – 192 с.
3. Жабін В.І., Ткаченко В.В. Цифрові автомати. Практикум. – К.: ВЕК+, 2004.– 160 с.
4. Процюк Р. О., Корнейчук В.И. та інші. Комп'ютерна схемотехніка.- К.: Корнейчук, 2006.- 468 с.
5. Самофалов К.Г. и др. Прикладная теория цифровых автоматов. – К.: Вища школа, 1987. – 375 с.

Допоміжна

6. Арифметичні та управляючи пристрої цифрових ЕОМ: Навчальний посібник / В.І.Жабін, І.А.Жуков, І.А.Клименко, Стиренко С.Г. – К.: ВЕК +, 2008. – 176 с.
7. Жабин В.И. и др. Логические основы и схемотехника ЭВМ: Практикум. – К.: ВЕК+, 1999. – 128 с.
8. Самофалов К.Г., Корнейчук В.И., Тарасенко В.П., Жабин В.И. Цифровые ЭВМ. Практикум. – К.: Вища школа, 1989. – 215 с.
9. ДСТУ 3212-95. Мікросхеми інтегровані. Класифікація та система умовних позначень.
10. ДСТУ 2399-94. Системи обробки інформації. Логічні пристрої, схеми, сигнали. Терміни та визначення.
11. ДСТУ 2383-94. Мікросхеми інтегровані. Терміни, визначення та літерні позначення електричних параметрів.

Інформаційні ресурси

1. Quartus II Introduction Using Schematic Design ftp://ftp.altera.com/up/pub/Tutorials/DE2/Digital_Logic/tut_quartus_intro_schem.pdf
2. Прикладна теорія цифрових автоматів - http://bookre.org/reader?file=600701&pg=2.
3. Комп'ютерна схемотехніка. https://www.college-chnu.cv.ua/images/Books/Komp_N/Osn_Sh.pdf
4. Комп'ютерна схемотехніка. https://msn.khnu.km.ua/course/view.php?id=3565.

ДОДАТКИ

Додаток А
ВСТАНОВЛЕННЯ ДРАЙВЕРА USB-BLASTER

DE2 плата. Стисла інформація

Плата DE2 має багато можливостей, що дозволяють користувачу реалізувати велику кількість розроблених схем - від простих до різних мультимедійних проектів.

На рис. А.1 зображено розміщення ключових компонентів і з'єднань на платі.

На платі DE2 розміщені наступні апаратні модулі:

- Altera Cyclone® II 2C35 FPGA пристрій
- Altera Serial Configuration Device - EPCS16
- USB програматор ПЗУ (на платі) для програмування и контролю програмного інтерфейсу користувачем, при чому підтримуються JTAG, і AS(Active Serial) режими.
- 512-кілобайтний SRAM
- 8-мегабайтний SDRAM
- 4-мегабайтна флеш-пам'ять
- Гніздо SD карти
- **4 кнопочних перемикача**
- **18 колінчатих перемикача**
- **18 червоних LEDs**
- **9 зелених LEDs**
- 50-MHz генератор і 27-MHz генератор для таймера.
- 24-бітний CD звук CODEC с лінійним входом, лінійним виходом і входом для мікрофону
- VGA DAC (10-бітний швидкісний трійний DACs) з вихідним VGA роз'ємом
- TV Декодер (NTSC/PAL) і вхідний TV роз'єм
- 10/100 Ethernet контролер з роз'ємом
- USB Host/Slave контролер з роз'ємами типу A і типу B.
- Прийомопередавач RS-232 і 9ти піновий роз'єм
- Роз'єм для PS/2 мишки/клавіатури
- Прийомопередавач IRDA
- Два 40-пінових слота розширення з діодним захистом.

В додаток до цих апаратних властивостей, DE2 плата має програмне забезпечення для стандартних інтерфейсів вводу/виводу і панель управління для доступу до різних компонентів. Також, програмне забезпечення забезпечує декілька демонстрацій, які ілюструють додаткові можливості DE2 плати.

Для того, щоб використати DE2 плату користувачу необхідно познайомитись з програмним інструментом Quartus II.

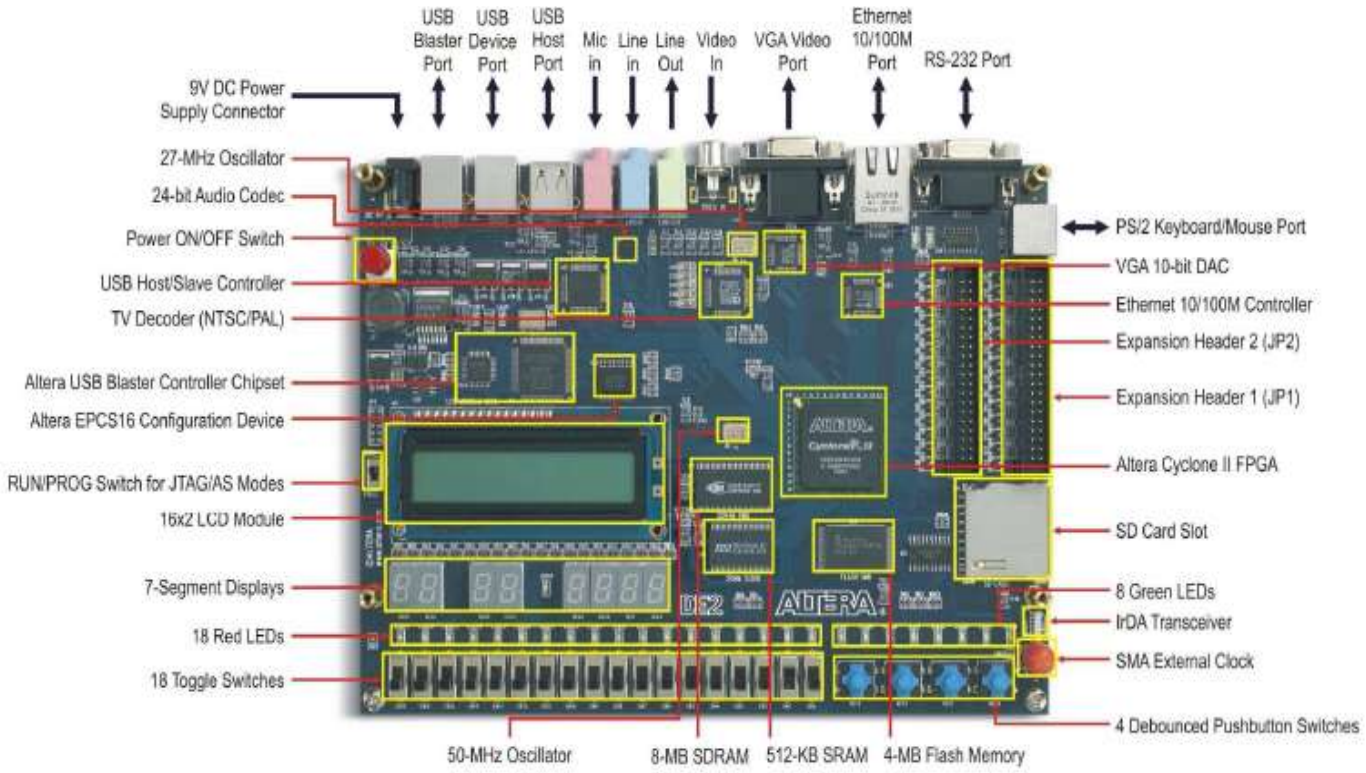


Рис. А.1. Розташування компонентів на платі

JTAG и AS режими

Altera DE2 плата дозволяє загружати файл конфігурації в пристрій двома шляхами: з допомогою JTAG режиму і з допомогою AS режиму. Конфігураційна інформація пересилається з головного комп'ютера (на якому працює інструмент Quartus II) на плату з допомогою кабеля, який підключається до головного комп'ютера через USB порт і через лівий USB роз'єм на платі. Для того, щоб використати це з'єднання, необхідно встановити драйвер USB програматора ПЗУ (USB-Blaster driver). Перед використанням плати, необхідно переконатись, що USB кабель правильно підключено і правильно підведено живлення до плати.

В JTAG (Joint Test Action Group) режимі конфігураційна інформація загружається безпосередньо в FPGA пристрій. Ця група найшла простий шлях для тестування цифрових схем і загрузки даних в них, і цей шлях став IEEE стандартом. Якщо FPGA конфігуровано таким чином, пристрій буде тримати (зберігати) цю конфігурацію до того часу, поки живлення буде включено. Конфігураційна інформація втрачається, коли живлення плати вимикається.

Можливо також використати AS (Active Serial) режим. В цьому випадку конфігураційний пристрій, який має деяку флеш пам'ять, використовується для зберігання конфігураційної інформації. Програмний комплекс Quartus II розміщує конфігураційну інформацію в конфігураційному пристрої на платі. Після чого, ця інформація загружається в FPGA під час вмикання живлення або реконфігурації. Обрати той або інший режим можна з доломогою перемикача RUN/PROG на DE2 платі. Позиція RUN призначена для JTAG режиму, позиція PROG – для AS режиму.

Практичне керівництво
Встановлення USB-Blaster driver

Для початку підключимо DE2 плату до електромережі з допомогою 9ти вольтного адаптера, який є в комплекті. Далі з'єднаєм головний комп'ютер (USB порт) з платою (лівий роз'єм USB на платі) з допомогою USB кабеля. Включим пристрій (червона кнопка на платі).

Комп'ютер розпізнає новий пристрій, що підключений до USB порту, але він не буде працювати з пристроєм, поки необхідний драйвер не буде установлено. DE2 плата програмується, використовуючи Altera USB-Blaster механізм. Якщо драйвер USB-Blaster ще не установлено, з'явиться вікно **New Hardware Wizard**, зображене на рис. А.2.



Рис. А.2. Вікно Found New Hardware Wizard

В вікні виберіть **No, not this time** і жміть **Next**. В вікні, що з'явилось, зображеному на рис. А.3, поскільки драйвер доступний без інструмента Quartus II виберіть **Install from a specific location** і жміть **Next.**

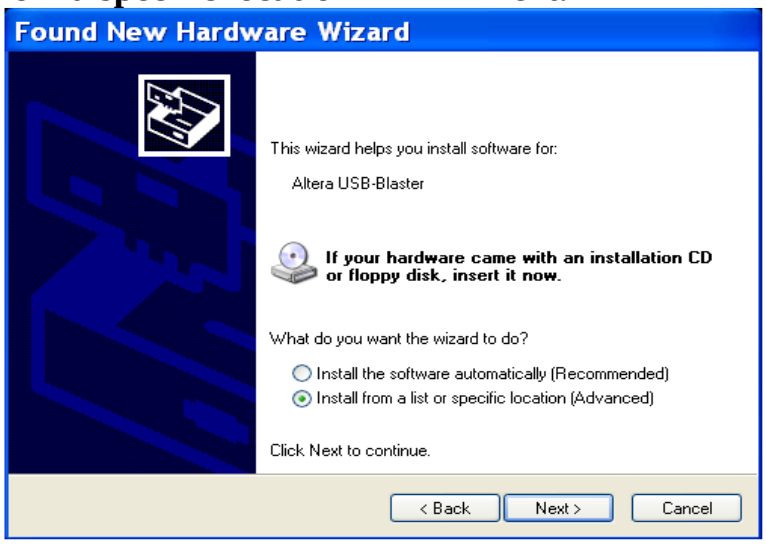


Рис. А.3. Драйвер найдено в визначеному місці.

В вікні, що з’явилося, зображеного на рис.А.4, виберіть Search fro the best driver in these locations і жміть Browse для того, щоб добратися до вікна, зображеного на рис. А.5.



Рис. А.4. Визначення місцезнаходження драйвера



Рис. А.5. Вибір місцезнаходження в дереві каталогів

Найдіть необхідний драйвер, який знаходиться за шляхом *altera\quartus72\drivers\usb-blaster*. Жміть **OK**, далі **Next**. Після цього з’явитья вікно (рис. А.6.)



Рис. А.6. Нема необхідності тестувати драйвер

Жміть**Continue Anyway**.

Тепер драйвер установлено, як показано на рис.А.7. Жміть **Finish** і можете починати роботу з платою DE2.

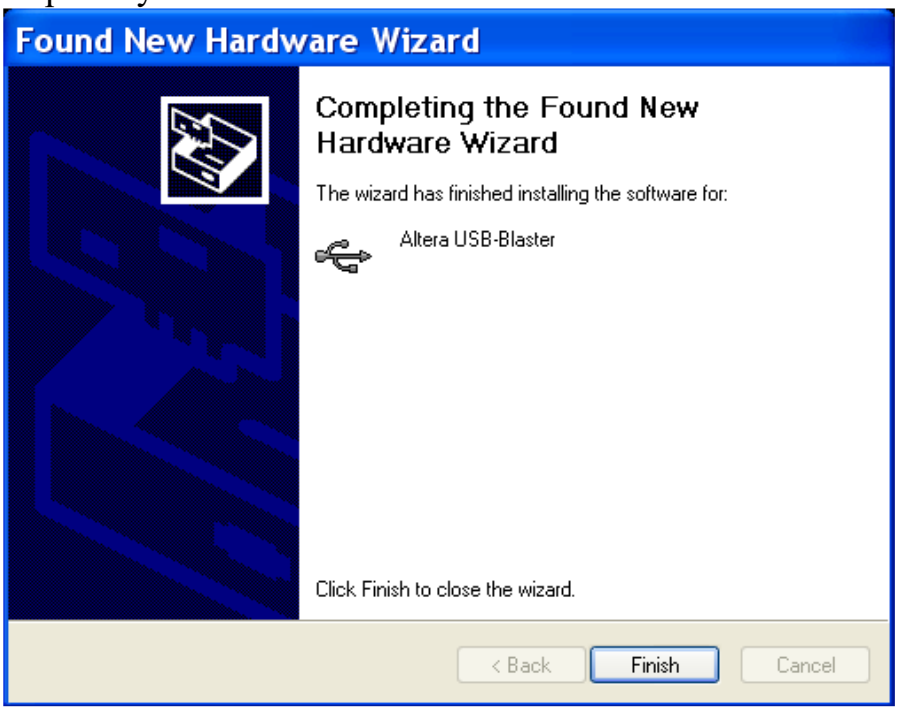


Рис. А.7. Драйвер установлено

Підключення відлагоджувального комплексу

Підключення комплексу до комп'ютера відбувається наступним чином:

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

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.

Програмування ПЛІС

Програмування ПЛІС проводиться за допомогою утиліти Programmer. Щоб її запустити, потрібно виконати Tools->Programmer. У вікні, що з'явиться 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 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 звантаження був виконаний успішно і можна тестувати роботу пристрою на відлагоджувальному комплексі.

Приклад перевірки роботи зображено на рис. А.8. Введені значення: XA=5 YA=3, RZ=0.00110011, Z=0.00111111.



Рис. А.8 – Перевірка працездатності пристою

Додаток Б

ІНСТРУКЦІЯ ДЛЯ КОРИСТУВАЧА DE2 VERSION 1.4 ALTERA DE2 BOARD

ЗМІСТ

Б.1 Пакет DE2 82
Б.1.1 Вміст Пакету 82
Б.1.2 Збір DE2стенда 82
Б.1.3 Допомога 83
Б. 2 Стенд Altera DE2 83
Б.2.1 Розташування і Компоненти 83
Б.2.2 Блок-схема стенда DE2 84
Б.2.3 Живлення стенда DE2 87
Б.3 Використання стенда DE2 87
Б.3.1 Розміщення Cyclone II FPGA 87
Б.3.2. Використання вхідних елементів і вимикачів 88
Б.3.3. Використання 7-сегментних індикаторів 91
Б.3.4. Входи таймера 93
Б.3.5. Використання модуля LCD 94
Б.3.6. Використання магістралі розширення 95
Б.3.7. Використання VGA 97
Б.3.8. Використання 24-розрядного звуку CODEC 100
Б.3.9. Порт серії RS-232 101
Б.3.10. Порт серії PS/2 101
Б.3.11. Диспетчер мережі швидкого Ethernet 102
Б.3.12. Декодер TV 103
Б.3.13. Шифратор TV 104
Б.3.14. Використання USB 104
Б.3.15. Використання IRDA 106
Б.3.16. Використання SDRAM/SRAM/Flash 106

Б.1. Пакет DE2

Пакет DE2 містить всі компоненти, що потрібно для стенда в поєднанні з комп'ютером, який має програмне забезпечення Microsoft Windows.

Б.1.1 Вміст Пакету

Пакет DE2 включає (рис. Б.1.1):

Плату DE2 стенда;

Кабель USB для ПЛІС програмування і контролю;

CD-ROM, що містить документацію DE2, у тому числі Інструкція для Користувача;

Користування Панель Управління посилаються на проекти і демонстрації, пристрій datasheets, консультації і набір лабораторних вправ;

CD-ROMS, що містить Altera's Quartus® II, Мережеве Видання і Nios® II ;

Сумка для шести кремнієвих кришок для DE2 стенда. Сумка також містить деякі шпильки розширювача, які можуть використовуватися аби полегшити дослідження з перевіряючим устаткуванням магістралей розширення управління введення/виведення;

Прозора пластикова кришка для управління;

9V прямолінійне електроживлення.



Рис. Б.1.1. Вміст пакету DE2

Б.1.2 Збір стенду DE2

Встановіть кремнієву кришку, як показано на рис. Б.1.2, для кожної з шести мідних стійок на платі DE2

Прозора пластикова кришка забезпечує додатковий захист, і підіймається над платою, використовуючи додаткові стійки і гвинти.

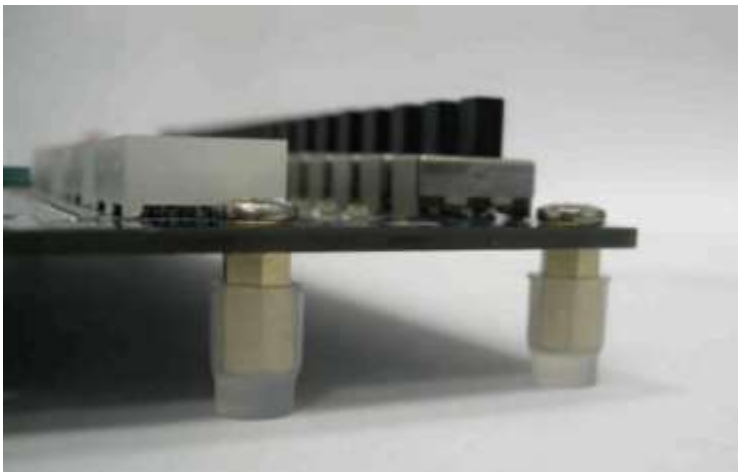


Рис. Б.1.2.Ніжки плати DE2.

Б.1.3 Допомога

Тут є адреси, за якими ви можете отримати допомогу, якщо виникнуть проблеми:

•Altera 101 Сан-Хосе, Каліфорнія, 95134 електронна пошти США: university@altera.com

•Технології Terasic

Номер 356, Сек. 1, Місто Jhubei, Графство HsinChu, Тайвань, 302

Електронна пошта: *support@terasic.com*

708-222 Spadina Торонто авеню, Онтаріо, канадська електронна Пошта M5T3A2: DE2support@archescomputing.com Web: DE2.archescomputing.com

Форум електронної ДОШКИ (Система Дошки Оголошень) оголошень для плати DE2 був створений за адресою, показаною нижче.

BBS forum: http://www.terasic.com/english/discuss.htm

Б.2 Стенд Altera DE2

Ця глава представляє особливості проектування та характеристики стенда DE2.

Б.2.1. Розташування і Компоненти

Фотографія стенда DE2 представлена на рис. Б.2.1. Вона представляє розташування стенда і вказує розташування з'єднувачів і ключових компонентів.

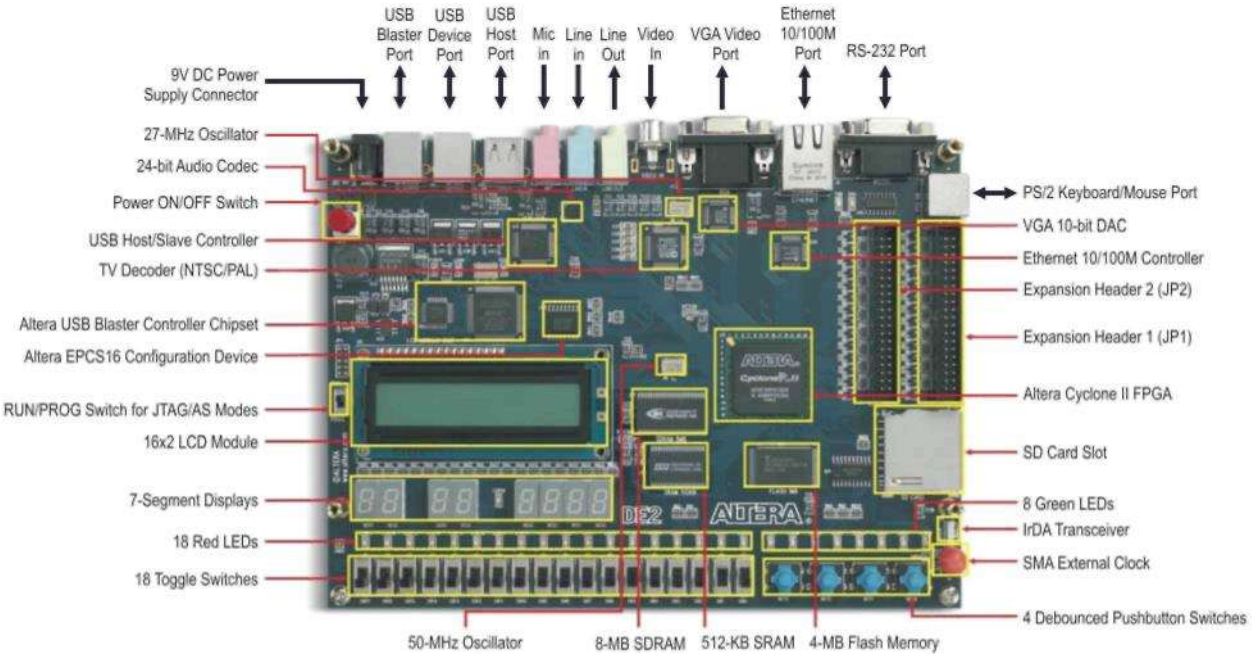


Рис. Б.2.1. Стенд DE2.

Стенд DE2 має багато особливостей, які дозволяють користувачеві здійснювати широкий діапазон задач, від простих схем до різних мультимедійних проектів.

Наступні технічні засоби забезпечені на платі DE2:

Altera Cyclone® II 2C35 пристрій FPGA;

Altera пристрій Конфігурації Серії - EPCS16;

Диспетчер (на управлінні) USB для програмування і призначений для контролю програмного ІНТЕРФЕЙСУ AS та JTAG;

512-кілобайтний SRAM;

8-мегабайти SDRAM;

- Флеш-пам'ять 4-мегабайти (1 Мегабайт на деяких стендх);
- Гніздо SD карти;
- 4 кнопочних вимикачів;
- 18 колінчастих вимикачів;
- 18 червоних LEDs;
- 9 зелених LEDs осцилятора 50-мгц і осцилятор 27-мгц для джерел таймера;
- 24-розрядний звук CD-якості CODEC з лінійним входом, лінійним виходом і включеним мікрофоном;
- VGA DAC (10-розрядний швидкохідний потрійний DACs) з Vga-зовнішнім з'єднувачем;
- Декодер (NTSC/PAL) TV і включений ТБ з'єднувач;
- 10/100 Диспетчер Ethernet із з'єднувачем;
- Host/Slave Диспетчер USB;
- Приймач RS-232 і з'єднувач;
- PS/2 мишка/клавіатура;
- Приймач IRDA;
- Дві 40-шпильні магістралі розширення з діодним захистом;
- На додаток до цих апаратних особливостей, стенд DE2 має програмну підтримку і засіб панелі управління для доступу до різних компонентів. Також, програмне забезпечення забезпечує ряд демонстрацій, які ілюструють можливості плати DE2.
- Для того, щоб використовувати стенд DE2, користувачеві доведеться ознайомитися з програмним забезпеченням Quartus II. Необхідне знання можна отримати на заняттях консультацій *Початок стенд Altera's DE2* і *Вступом до Quartus II* (який існує в трьох версіях, а саме Verilog, VHDL та схемний вхід). Ці консультації знаходяться *DE2_tutorials* на **DE2 Системі CD-ROM,** який входить до складу стенда DE2 і також можна знайти на веб-сторінці Altera's DE2.

Б.2.2 Блок-схема стенда DE2

Рис. Б.2.2 представляє блок-схему стенда DE2. Аби забезпечити максимальну гнучкість для користувача, всі зв'язки робляться через Cyclone II. Тому, користувач може розмістити ПЛІС, аби здійснювати будь-який системний проект.

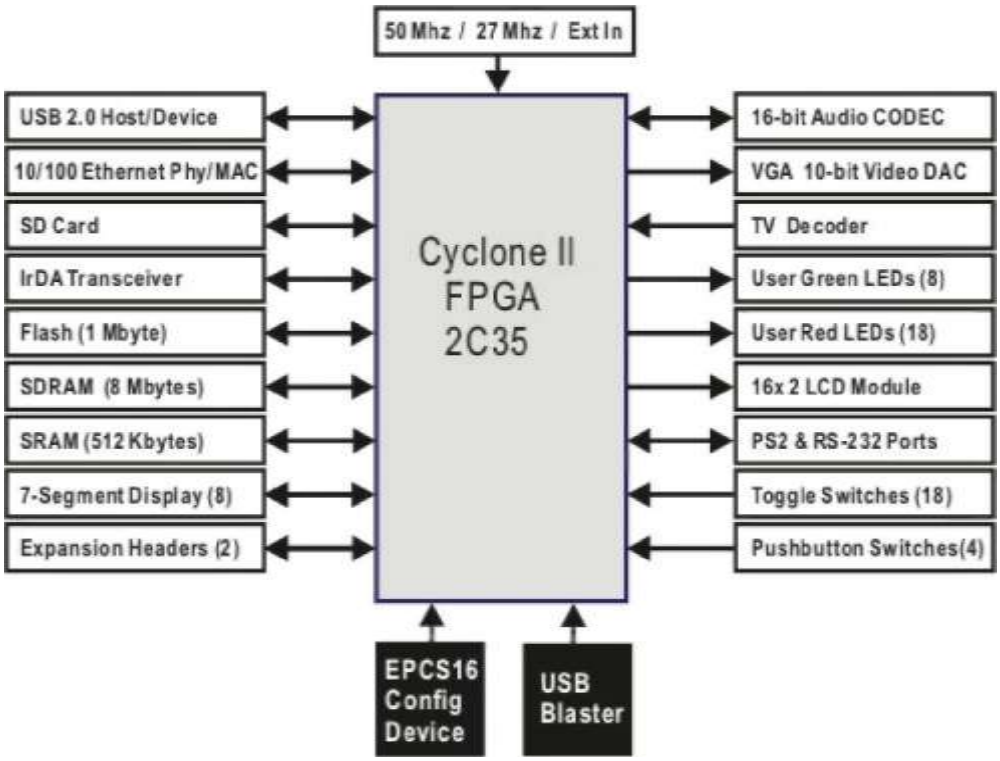


Рис. Б.2.2. Блок-схема стенда DE2.

Cyclone II 2C35 ПЛІС

33,216 ніжок;

· 105 M4K блоків оперативної пам'яті;

· 483,840 повних біт оперативної пам'яті;

· 35 вкладених множників;

· 4 PLLs;

· 475 призначених для користувача шпильок для введення/виведення;

· FineLine BGA 672-pin пакет;

Серійний пристрій конфігурації і USB диспетчер

Пристрій Конфігурації Серії Altera's EPCS16.

· USB диспетчер на платі для програмування і призначений для користувача, щоб той контролював програмний інтерфейс;

· JTAG та AS програмний метод.

SRAM (статична оперативна пам'ять**)**

512-кілобайтний блок пам'яті статичної оперативної пам'яті.

· Організовано як 256K x 16 біт;

· Доступний як пам'ять для процесора Nios II і DE2 Панелі Управління.

SDRAM

Швидкісна оперативна пам'ять.

· Організовано як 1M x 16 розрядів x 4 банків;

· Доступний як пам'ять для процесора Nios II і DE2 панелі управління.

Показна пам'ять

4-мегабайти Показної пам'яті (1 Мегабайт на деяких стендах).

· шина 8-розрядних даних;

· Доступна як пам'ять для процесора Nios II і DE2 панелі управління.

Гніздо SD карти

Забезпечує метод SPI для доступу SD карти.

· Доступний як пам'ять для процесора Nios II DE2 SD Карти.

Кнопочні вимикачі

4 кнопочні вимикачі.

· Коли вимикач натискують, він виробляє один імпульс.

Колінчасті вимикачі

18 колінчастих вимикачів, що призначені для входів.

Вимикач викликає логічний 0, коли в позиції DOWN (ближче до краю стенда DE2) і логічну 1, коли в позиції UP.

Входи синхронізації

осцилятор 50-мгц;

· осцилятор 27-мгц;

· SMA вхід таймера.

Звуковий CODEC

Звук Wolfson WM8731 24-розрядної сигматичної дельти CODEC.

· Горизонтальний для лінійного входу, рівень лінії виходу, і вхід для мікрофону;

· Випробувальна частота: від 8 до 96 кГц;

· Додатки для програвання MP3 і реєстраторів, PDAs, голосові реєстратори і т.п.

Вихід VGA

Використовує ADV7123 240-мгц та 10-розрядний швидкохідний DAC.

· Підтримує аж до 1600 x 1200 в частоті оновлення 100-гц;

· Може використовуватися з Cyclone II ПЛІС, аби здійснювати високопродуктивний Шифратор TV.

NTSC/PAL схему декодера ТБ

ADV7181B Декодер Відео багато-форматний SDTV.

· Підтримує NTSC-(M., J,4.43), PAL-(B/D/G/H/I/M/N), SECAM;

· Об'єднує три 54-мгц 9-розрядний ADCs;

· Відлік від єдиного входу осцилятора 27-мгц;

· Підтримує Комплексний RCA відеовхід;

Підтримує формати цифрової продукції (8-bit/16-bit): ITU-R output BT.656 YCrCb 4:2:2 + HS, VS.

Додатки: Реєстратори DVD, LCD TV, Цифрове телебачення, мобільні відеопристрої.

10/100 диспетчер Ethernet

Інтегрований MAC і PHY із загальним інтерфейсом процесора.

· Підтримує 100Base-T і 10Base-T додатки;

· Підтримує дуплексну дію в 10 Mb/s і 100 Mb/s, з auto-MDIX;

· Повністю поступливий із Специфікацією IEEE 802.3u;

· Підтримка IP/TCP/UDP покоління;

· Підтримує back-pressure метод для контролю потоку half-duplex методу.

Диспетчер USB Host/Slave

Підкоряється повністю Універсальному Серійному Звороту Шинної Специфікації. 2.0;

· Підтримує передачу даних на повній швидкості та при низькій швидкості;

· Підтримує двох портові USB;

Забезпечує швидкохідний паралельний інтерфейс до більш всього доступних процесорів; підтримує Nios II з драйвером Terasic;

•Підтримує Програмований I/O (PIO) і Прямий Доступ Пам'яті.

Серійні порти

Один порт RS-232;

· Один порт PS/2;

· З'єднувач серії БД-9 для порту RS-232;

· З'єднувач PS/2 для з'єднання миші PS2 або клавіатури з стендом DE2.

Приймач IRDA

Містить: 115.2-kb/s для інфрачервоного приймача;

· інтегрований щит;

· IEC825-1 Клас 1 безпечний для очей;

· Вхід edge.

Дві 40-шпильні магістралі розширення

72 Циклону II шпильки введення/виведення, лінії заземлення;

40-pin магістраль проектується, аби прийняти 40-pin стрічковий кабель стандарту для використовування жорстких дисків IDE.

Забезпечені діод і резистор захисту.

Б.2.3 Живлення стенда DE2

Для живлення стенда виконують наступні кроки:

1) З'єднайте забезпечений кабель USB з головним комп'ютером до USB з'єднувача диспетчера на платі DE2. Для комунікації між користувачем і стендом DE2, необхідно встановити програмне забезпечення Altera. Якщо не встановлюється на головному комп'ютері, це може бути встановлене як пояснено в консультації *Початок DE2 Правлінням Altera*. Ця консультація доступна на **DE2 Системі CD-ROM** і на веб-сторінці Altera DE2.

2) З'єднайте 9V адаптер з стендм DE2.

3) З'єднайте монітор VGA з портом VGA на управлінні DE2.

4) З'єднайте свій головний телефон із зовнішнім лінійним звуковим портом на платі DE2.

5) Включіть вимикач RUN/PROG з лівого боку стенда DE2, аби управляти позицією; позиція PROG використовується лише для AS програмування.

6) Натисніть на On/off перемикач на платі DE2.

Б.3 Використання стенда DE2

Ця глава дає інструкції для використання стенда DE2 і описує кожний з його пристроїв введення/виведення.

Б.3.1 Розміщення Cyclone II FPGA

Процедура для завантаження схем з головного комп'ютера до стенда DE2 описується в інструкції *Введення в Quartus II*. Ця консультація знаходиться в папці *DE2_tutorials* на **DE2 Системі CD-ROM,** і це також доступно на Altera DE2 веб-сторінці. Заохочують, аби користувач спочатку читав консультацію.

Стенд DE2 містить серію EEPROM, яка запам'ятовує дані конфігурації для Циклону II FPGA. Ці дані автоматично завантажується від EEPROM в FPGA. Використовуючи програмне забезпечення Quartus II, можливо перепрограмувати FPGA у будь-який час, і також можливо змінити непостійні дані, це запам'ятовується в EEPROM. Нижче описано два метода програмування.

JTAG програмування: У цьому методі програмування, названого на честь IEEE *Групи Випробувальних Акцій З'єднання* стандартів, дані конфігурації завантажується безпосередньо в Циклон II FPGA. FPGA зберігає цю конфігурацію, поки живлення подається до стенда; конфігурація втрачається, коли живлення вимкнене.

AS програмування: У цьому методі, *Активному Серійному* програмуванні, що звертається всередину даних конфігурації завантаження EEPROM Altera EPCS16. Це забезпечує непостійне зберігання даних, таким чином, що інформація зберігається навіть, коли електроживлення до стенда DE2 вимкнене. Коли живлення вмикається, дані конфігурації в пристрої EPCS16 автоматично завантажуються в Циклон II FPGA.

Пункти нижче описують кроки виконання як JTAG, так і AS. Для обох методів стенд DE2 з'єднується з головним комп'ютером через кабель USB. Використовуючи цей зв'язок, стенд ідентифікує головний комп'ютер як Altera *USB* пристрій *диспетчера*. Процес для установки на головному комп'ютері, необхідний драйвер пристрою програмного забезпечення, який зв'язується з диспетчером USB, описується в консультації *Початок DE2 уравлінням Altera*. Ця консультація доступна на **DE2 Системі CD-ROM** і від Altera DE2 на веб-сторінці.

Конфігурація FPGA в методі JTAG

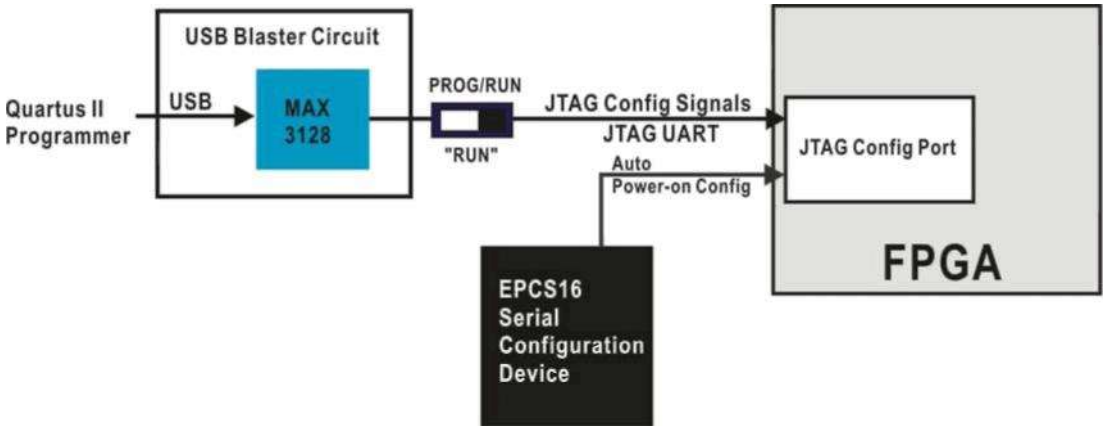
Рис. Б.3.1 ілюструє JTAG установку конфігурації. Аби завантажити дані конфігурації в Циклоні II FPGA, виконують наступні кроки:

Гарантуйте, що живлення присутнє для стенда DE2.

З'єднайте забезпечуваний кабель USB з USB портом диспетчера на управлінні DE2 (див. рис. Б.2.1).

Встановіть JTAG програмування, врегулювавши вимикач (з лівого боку сторони управління) RUN/PROG до позиції RUN.

FPGA може зараз програмувати використовуючи модуль Програміста Quartus II, аби вибрати файл конфігурації з .sof розширенням імені файлу.



- Рис. Б.3.1. JTAG схема конфігурації.

Розміщення EPCS16 в AS методі

- Рис. Б.3.2 ілюструє конфігурацію AS режиму. Аби завантажити конфігурацію даних в EPCS16, EEPROM пристрої виконують наступні кроки:
- Гарантуйте, що живлення застосовується до стенда DE2.
- З'єднайте забезпечуваний кабель USB з USB портом диспетчера на платі DE2 (див. рис. Б.2.1).
- Включіть AS програмування, врегулювавши вимикач (з лівого боку плати) RUN/PROG до позиції PROG.
- Чип EPCS16 можна зараз програмувати використовуючи модуль Програміста Quartus II, аби вибрати файл конфігурації даних з .pof розширенням імені файлу.

Як тільки дія програмування закінчена, встановлюють вимикач RUN/PROG назад до позиції RUN а потім знову встановлюють управління, вимикаючи енергетичний вимикач назад; ця акція викликає нові дані конфігурації в пристрої EPCS16, щоб був завантаженим пристрій EPCS16, який завантажує дані в чип FPGA.

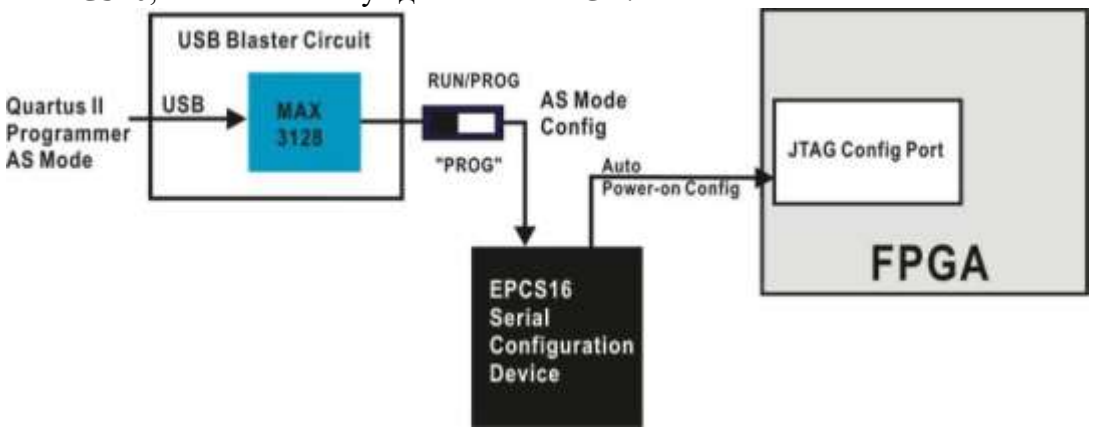


Рис. Б.3.2. AS схема конфігурації.

На додаток до його використання для JTAG і AS програмування, USB порт диспетчера на платі DE2 може також бути used, аби управляти деякими з особливостей плати віддалено від головного комп'ютера. Детально описано цей метод використання USB порту в документації стенда DE2.

Б.3.2 Використання введень і вимикачів

Стенд DE2 має чотири кнопки вимикачів. Кожен з цих вимикачів використовує Тригер Шмидта, як вказано на рис. Б.3.3. Чотири виходи під назвою *KEY0 ., KEY3* Тригера

з'єднуються безпосередньо з Циклоном II FPGA. Кожен вимикач забезпечує логічний високий рівень (3.3 Вольта), коли він не натиснутий, і забезпечує низький логічний рівень (0 Вольт), коли натиснутий.

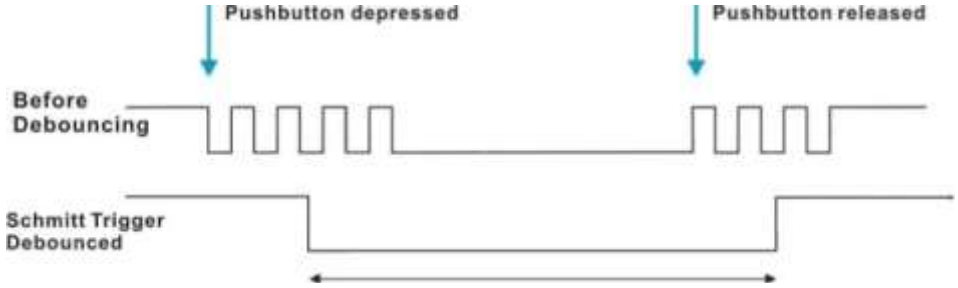


Рис. Б.3.3. Перемикання вимикачів

Є також 18 колінчастих вимикачів (засувки) на платі DE2. Ці вимикачі призначаються для введення даних. Кожен вимикач з'єднується безпосередньо з шпилькою на Циклоні II FPGA. Коли вимикач знаходиться в позиції (близько до краю плати) DOWN, це забезпечує низький логічний рівень (0 Вольт) до FPGA, і коли вимикач знаходиться в позиції UP, це забезпечує логічний високий рівень (3.3 Вольт).

Є 27 керованих користувачем введень на платі DE2. Вісімнадцять червоних введень розташовано вище за 18 колінчастих вимикачів, і вісім зелених введень знаходяться вище за кнопками вимикачами (9-е зелене ВВЕДЕННЯ посеред 7-сегментних показів). Кожне ВВЕДЕННЯ управляється безпосередньо шпилькою на Циклоні II FPGA; управляючи його асоційованою шпилькою до високих логічних рівнів LED, і управляючи pin низькими рівнями. Схематична діаграма, яка показує схему натиснених і колінчастих вимикачів, приведена на рис. Б.3.4. Схематична діаграма, яка показує схему LED, приведена на рис. Б.3.5.

Список pin імен на Циклоні II FPGA, які з'єднуються з колінчастими вимикачами, дається в таблиці Б.3.1. Так само, шпильки з'єднані з натисненими вимикачами і введення показуються в таблицях Б.3.2 і Б.3.3, відповідно.

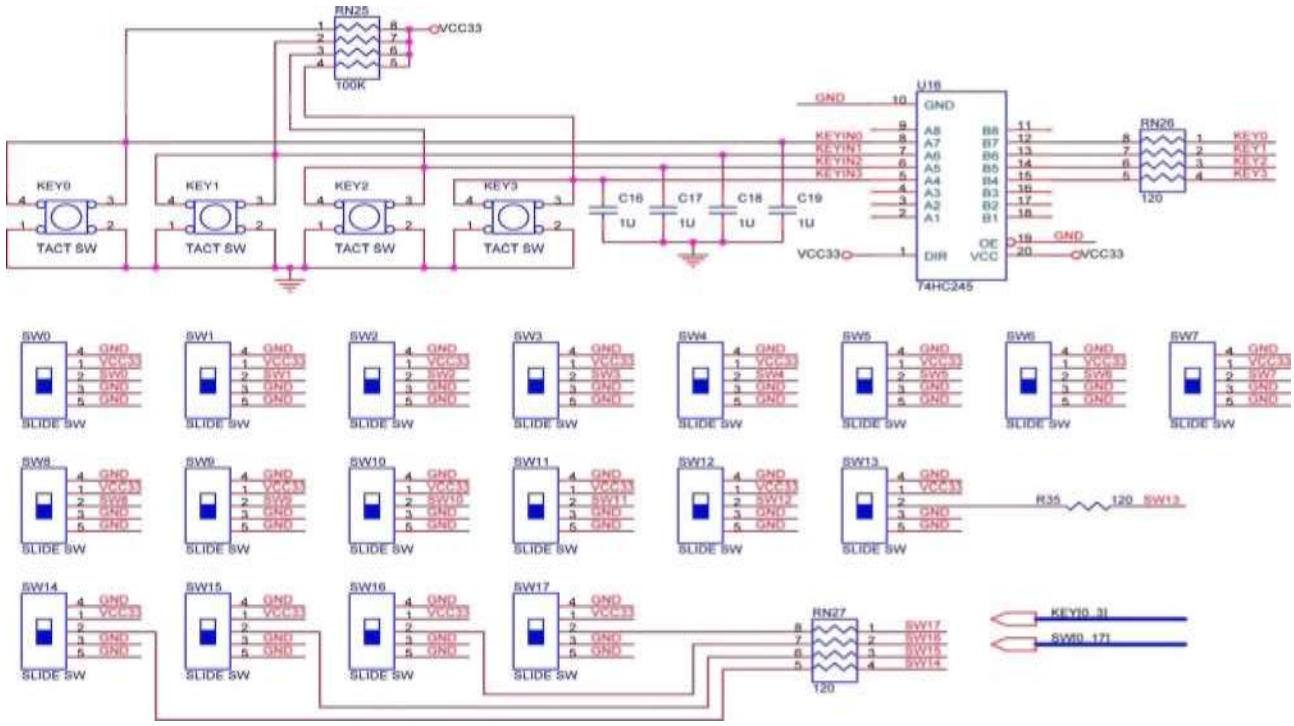


Рис. Б.3.4. Схематична діаграма натиснутих і колінчастих вимикачів.

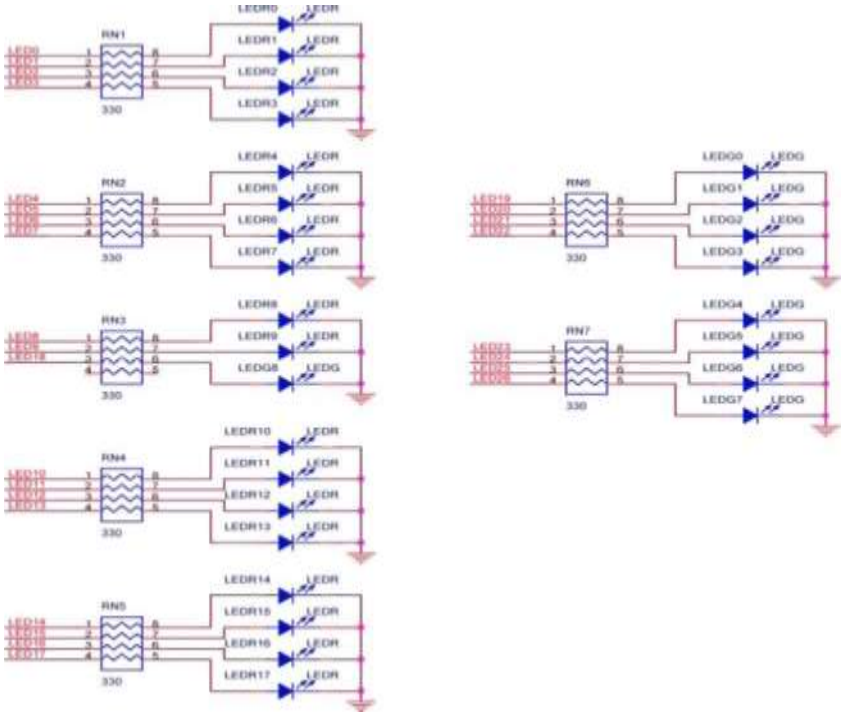


Рис. Б.3.5. Схематична діаграма LEDs.

Таблиця Б.3.1. Pin призначення для колінчастих вимикачів

Signal Name	FPGA Pin	Description
SW[0]	PIN_N25	Колінчастий Вимикач[0]
SW[1]	PIN_N26	Колінчастий Вимикач[1]
SW[2]	PIN_P25	Колінчастий Вимикач[2]
SW[3]	PIN_AE14	Колінчастий Вимикач[3]
SW[4]	PIN_AF14	Колінчастий Вимикач[4]
SW[5]	PIN_AD13	Колінчастий Вимикач[5]
SW[6]	PIN_AC13	Колінчастий Вимикач[6]
SW[7]	PIN_C13	Колінчастий Вимикач[7]
SW[8]	PIN_B13	Колінчастий Вимикач[8]
SW[9]	PIN_A13	Колінчастий Вимикач[9]
SW[10]	PIN_N1	Колінчастий Вимикач[10]
SW[11]	PIN_P1	Колінчастий Вимикач[11]
SW[12]	PIN_P2	Колінчастий Вимикач[12]
SW[13]	PIN_T7	Колінчастий Вимикач[13]
SW[14]	PIN_U3	Колінчастий Вимикач[14]
SW[15]	PIN_U4	Колінчастий Вимикач[15]
SW[16]	PIN_V1	Колінчастий Вимикач[16]
SW[17]	PIN_V2	Колінчастий Вимикач[17]

.

Таблиця Б.3.2. Pin призначення для натиснених вимикачів

Signal	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 Б.3.3. Pin призначення для LEDs.

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]

Б.3.3 Використання 7-сегментних індикаторів

Стенд DE2 має вісім 7-сегментних індикаторів. Ці індикатори влаштовуються в двох парах і чотирьох групах, з наміром показу номерів різних розмірів. Як вказано в схемному рішенні на рис. Б.3.6, сім сегментів з'єднуються з шпильками на Циклоні II FPGA. Застосування низького логічного рівня до сегменту заставляє їх запалитися.

Кожен сегмент в показі ідентифікує індекс з 0 до 6, з позиціями, даними на рис. Б.3.7. Відзначте, що крапка в кожному індикаторі незв'язана і не може бути використана. В таблиці Б.3.4 приведені призначення FPGA Pin для 7-сегментних індикаторів.

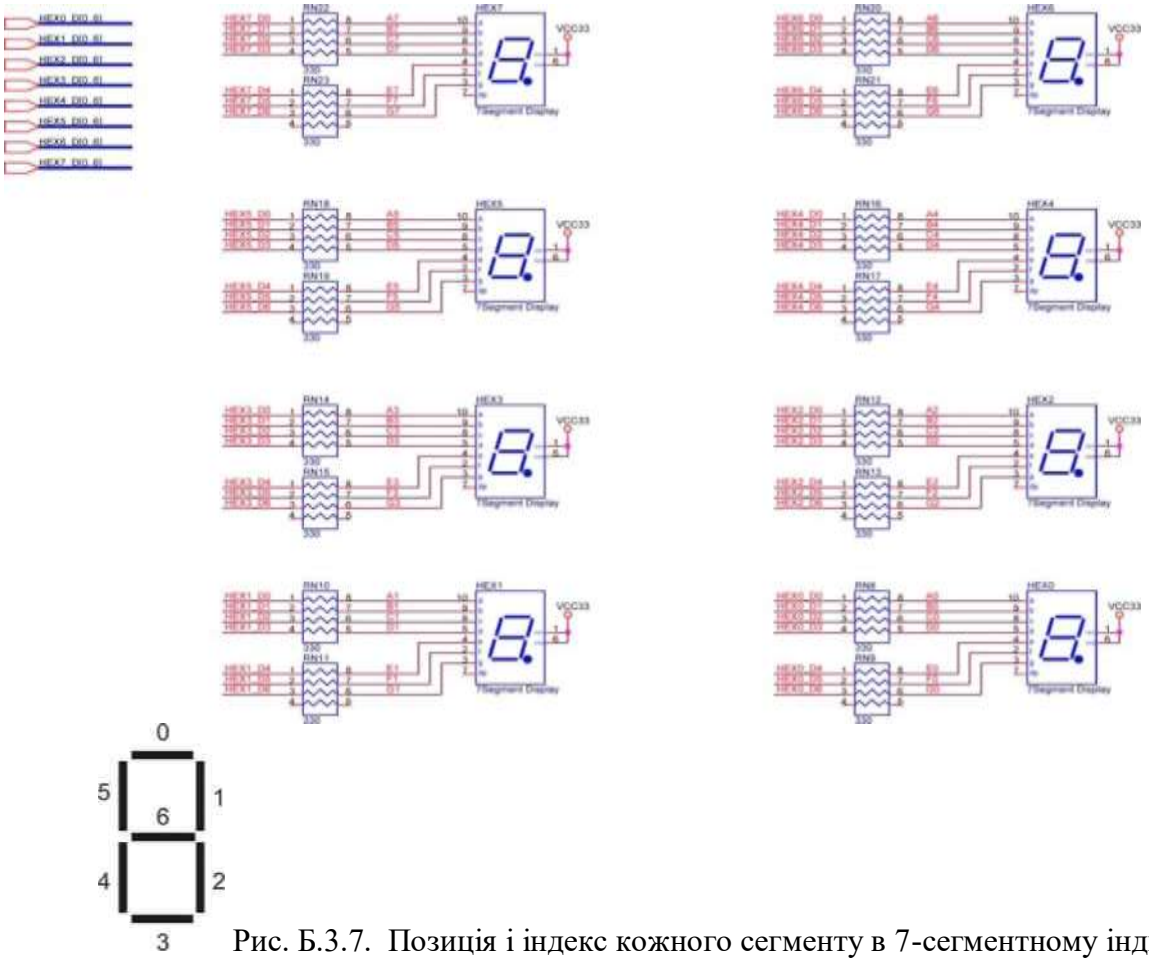


Рис. Б.3.7. Позиція і індекс кожного сегменту в 7-сегментному індикаторі.

Таблиця Б.3.4. Pin призначення для 7-сегментних показів.

Signal Name	**FPGA Pin No.**	**Description**
HEX0[0]	PIN_AF10	Seven Segment Digit 0[0]
HEX0[1]	PIN_AB12	Seven Segment Digit 0[1]
HEX0[2]	PIN_AC12	Seven Segment Digit 0[2]
HEX0[3]	PIN_AD11	Seven Segment Digit 0[3]
HEX0[4]	PIN_AE11	Seven Segment Digit 0[4]
HEX0[5]	PIN_V14	Seven Segment Digit 0[5]
HEX0[6]	PIN_V13	Seven Segment Digit 0[6]
HEX1[0]	PIN_V20	Seven Segment Digit 1[0]
HEX1[1]	PIN_V21	Seven Segment Digit 1[1]
HEX1[2]	PIN_W21	Seven Segment Digit 1[2]
HEX1[3]	PIN_Y22	Seven Segment Digit 1[3]
HEX1[4]	PIN_AA24	Seven Segment Digit 1[4]
HEX1[5]	PIN_AA23	Seven Segment Digit 1[5]
HEX1[6]	PIN_AB24	Seven Segment Digit 1[6]
HEX2[0]	PIN_AB23	Seven Segment Digit 2[0]
HEX2[1]	PIN_V22	Seven Segment Digit 2[1]
HEX2[2]	PIN_AC25	Seven Segment Digit 2[2]

HEX2[3]	PIN_AC26	Seven Segment Digit 2[3]
HEX2[4]	PIN_AB26	Seven Segment Digit 2[4]
HEX2[5]	PIN_AB25	Seven Segment Digit 2[5]
HEX2[6]	PIN_Y24	Seven Segment Digit 2[6]
HEX3[0]	PIN_Y23	Seven Segment Digit 3[0]
HEX3[1]	PIN_AA25	Seven Segment Digit 3[1]
HEX3[2]	PIN_AA26	Seven Segment Digit 3[2]
HEX3[3]	PIN_Y26	Seven Segment Digit 3[3]
HEX3[4]	PIN_Y25	Seven Segment Digit 3[4]
HEX3[5]	PIN_U22	Seven Segment Digit 3[5]
HEX3[6]	PIN_W24	Seven Segment Digit 3[6]
HEX4[0]	PIN_U9	Seven Segment Digit 4[0]
HEX4[1]	PIN_U1	Seven Segment Digit 4[1]
HEX4[2]	PIN_U2	Seven Segment Digit 4[2]
HEX4[3]	PIN_T4	Seven Segment Digit 4[3]
HEX4[4]	PIN_R7	Seven Segment Digit 4[4]
HEX4[5]	PIN_R6	Seven Segment Digit 4[5]
HEX4[6]	PIN_T3	Seven Segment Digit 4[6]
HEX5[0]	PIN_T2	Seven Segment Digit 5[0]
HEX5[1]	PIN_P6	Seven Segment Digit 5[1]
HEX5[2]	PIN_P7	Seven Segment Digit 5[2]
HEX5[3]	PIN_T9	Seven Segment Digit 5[3]
HEX5[4]	PIN_R5	Seven Segment Digit 5[4]
HEX5[5]	PIN_R4	Seven Segment Digit 5[5]
HEX5[6]	PIN_R3	Seven Segment Digit 5[6]
HEX6[0]	PIN_R2	Seven Segment Digit 6[0]
HEX6[1]	PIN_P4	Seven Segment Digit 6[1]
HEX6[2]	PIN_P3	Seven Segment Digit 6[2]
HEX6[3]	PIN_M2	Seven Segment Digit 6[3]
HEX6[4]	PIN_M3	Seven Segment Digit 6[4]
HEX6[5]	PIN_M5	Seven Segment Digit 6[5]
HEX6[6]	PIN_M4	Seven Segment Digit 6[6]
HEX7[0]	PIN_L3	Seven Segment Digit 7[0]
HEX7[1]	PIN_L2	Seven Segment Digit 7[1]
HEX7[2]	PIN_L9	Seven Segment Digit 7[2]
HEX7[3]	PIN_L6	Seven Segment Digit 7[3]
HEX7[4]	PIN_L7	Seven Segment Digit 7[4]
HEX7[5]	PIN_P9	Seven Segment Digit 7[5]
HEX7[6]	PIN_N9	Seven Segment Digit 7[6]

Б.3.4 Входи синхронізації

Стенд DE2 включає два осцилятори, які виробляють 27 мгц і 50 мгц синхронізуючих сигналів. Стенд також включає з'єднувач SMA, який може використовуватися для з'єднання зовнішнього джерела синхронізації стенда. Схематичне вирішення схеми синхронізації показано на рис. Б.3.8, і асоційовані pin призначення приведені в таблиці Б.3.5.

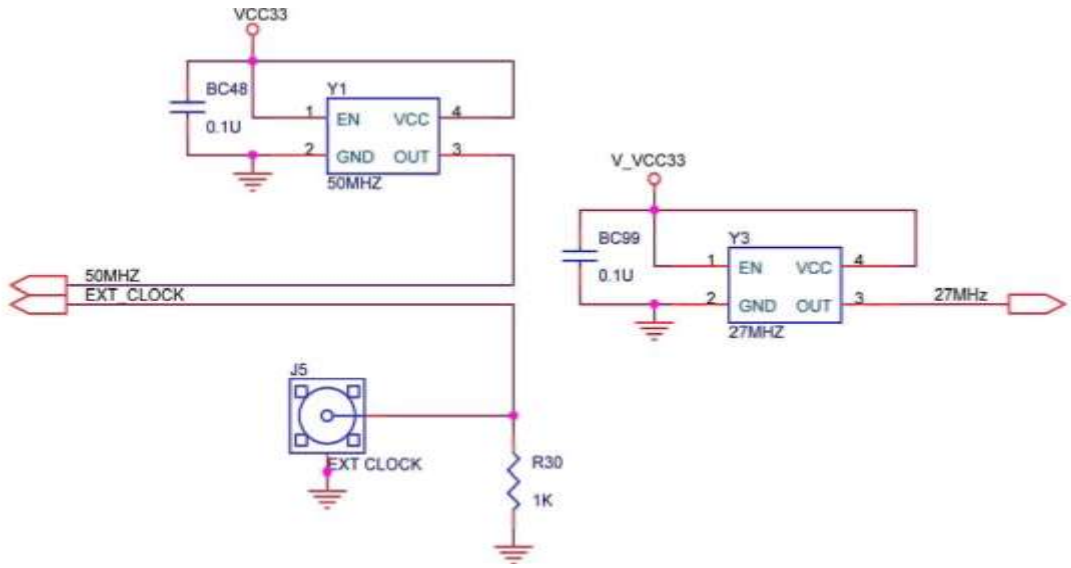


Рис. Б.3.8. Схематична діаграма синхронізації.

Таблиця Б.3.5. Pin призначення для входів синхронізації.

Signal Name	**FPGA Pin N...**	**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

Б.3.5 Використання Модуля LCD

Модуль LCD має вбудовані шрифти і може бути використанний для показу тексту, посилаючи відповідні команди до контроллера дисплея, який є під назвою HD44780. Детальна інформація для використання показу доступна в його datasheet, який може бути знайдений на веб-сайті виготівника, і в папці *Datasheet* на **DE2 Системі CD-ROM**. Схематична діаграма модуля LCD, що показує зв'язки Циклону II FPGA дається на рис. Б.3.9. Асоційовані pin призначення приведені в таблиці Б.3.6.

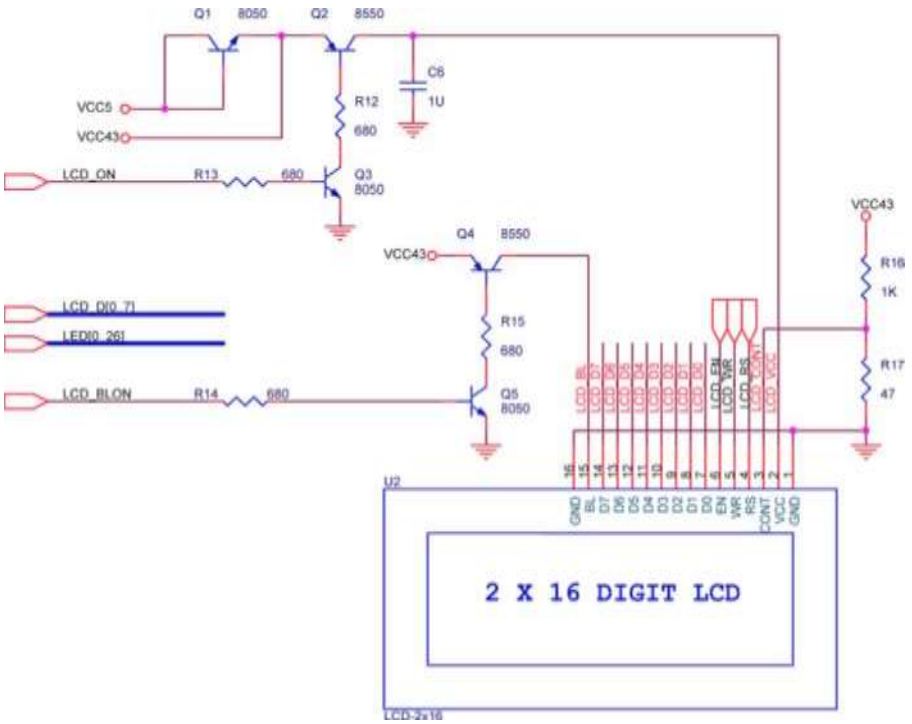


Рис. Б.3.9. Схематична діаграма модуля LCD.

Таблиця Б.3.6. Pin призначення для модуля LCD.

Signal Name	FPGA Pin	Description
LCD_DATA[0]	PIN_J1	LCD Data[0]
LCD_DATA[1]	PIN_J2	LCD Data[1]
LCD_DATA[2]	PIN_H1	LCD Data[2]
LCD_DATA[3]	PIN_H2	LCD Data[3]
LCD_DATA[4]	PIN_J4	LCD Data[4]
LCD_DATA[5]	PIN_J3	LCD Data[5]
LCD_DATA[6]	PIN_H4	LCD Data[6]
LCD_DATA[7]	PIN_H3	LCD Data[7]
LCD_RW	PIN_K4	LCD Read/Write Select, 0 = Write, 1 = Read
LCD_EN	PIN_K3	LCD Enable
LCD_RS	PIN_K1	LCD Command/Data Select, 0 = Command, 1 = Data
LCD_ON	PIN_L4	LCD Power ON/OFF
LCD_BLON	PIN_K2	LCD Back Light ON/OFF

Б.3.6 Використання магістралі розширення

Стенд DE2 забезпечує дві 40-pin магістралі розширення. Кожна магістраль з'єднується безпосередньо з 36 шпильками на Циклоні II FPGA, а також забезпечує DC +5V (VCC5), DC +3.3V (VCC33), і двох шпильок GND. Рис. Б.3.10 показує зв'язані схематичні рішення. Кожна шпилька на магістралях розширення з'єднується з двома діодами і резистором, який забезпечує захист від високої і низької напруги. Рисунок показує схему захисту лише для чотирьох шпильок на кожній магістралі, але ця схема включається для всіх 72 шпильок. Таблиця Б.3.7 дає pin призначення.

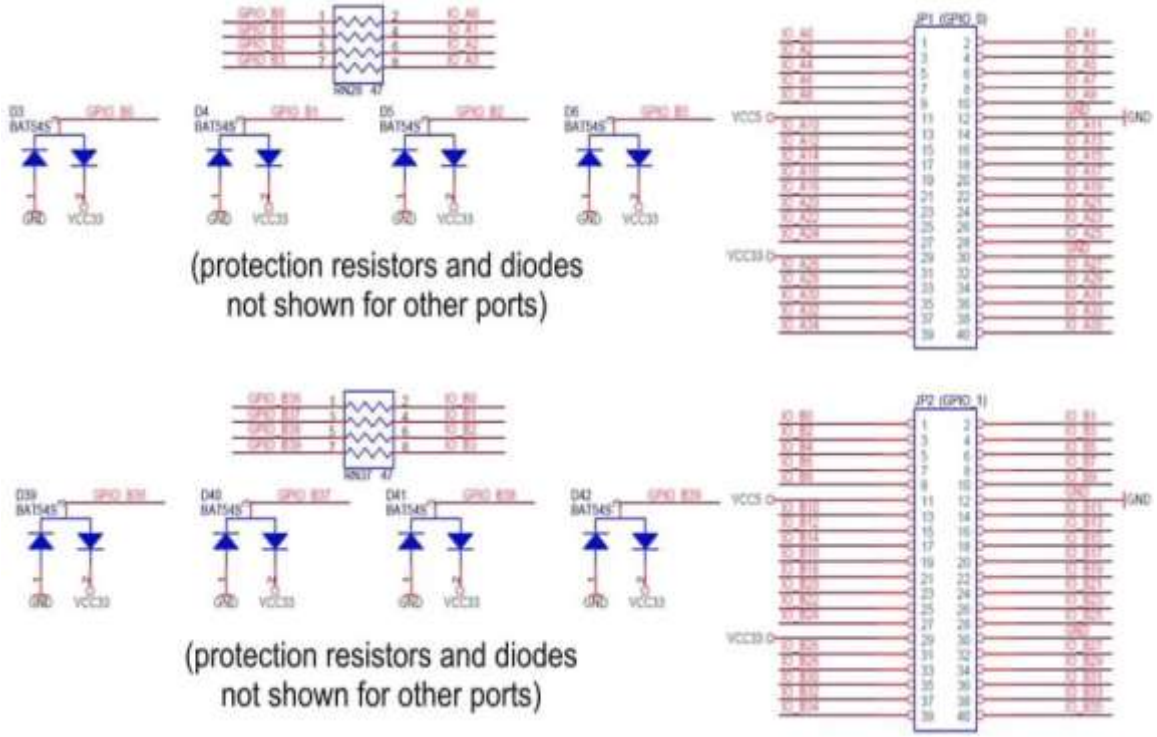


Рис. Б.3.10. Схематична діаграма магістралей розширення.

Таблиця Б.3.7. Pin призначення для магістралей розширення

Signal Name	FPGA Pin	Description
GPIO_0[0]	PIN_D25	GPIO Connection 0[0]
GPIO_0[1]	PIN_J22	GPIO Connection 0[1]
GPIO_0[2]	PIN_E26	GPIO Connection 0[2]
GPIO_0[3]	PIN_E25	GPIO Connection 0[3]
GPIO_0[4]	PIN_F24	GPIO Connection 0[4]
GPIO_0[5]	PIN_F23	GPIO Connection 0[5]
GPIO_0[6]	PIN_J21	GPIO Connection 0[6]
GPIO_0[7]	PIN_J20	GPIO Connection 0[7]
GPIO_0[8]	PIN_F25	GPIO Connection 0[8]
GPIO_0[9]	PIN_F26	GPIO Connection 0[9]
GPIO_0[10]	PIN_N18	GPIO Connection 0[10]
GPIO_0[11]	PIN_P18	GPIO Connection 0[11]
GPIO_0[12]	PIN_G23	GPIO Connection 0[12]
GPIO_0[13]	PIN_G24	GPIO Connection 0[13]
GPIO_0[14]	PIN_K22	GPIO Connection 0[14]
GPIO_0[15]	PIN_G25	GPIO Connection 0[15]
GPIO_0[16]	PIN_H23	GPIO Connection 0[16]
GPIO_0[17]	PIN_H24	GPIO Connection 0[17]
GPIO_0[18]	PIN_J23	GPIO Connection 0[18]
GPIO_0[19]	PIN_J24	GPIO Connection 0[19]
GPIO_0[20]	PIN_H25	GPIO Connection 0[20]
GPIO_0[21]	PIN_H26	GPIO Connection 0[21]
GPIO_0[22]	PIN_H19	GPIO Connection 0[22]
GPIO_0[23]	PIN_K18	GPIO Connection 0[23]
GPIO_0[24]	PIN_K19	GPIO Connection 0[24]
GPIO_0[25]	PIN_K21	GPIO Connection 0[25]
GPIO_0[26]	PIN_K23	GPIO Connection 0[26]
GPIO_0[27]	PIN_K24	GPIO Connection 0[27]
GPIO_0[28]	PIN_L21	GPIO Connection 0[28]
GPIO_0[29]	PIN_L20	GPIO Connection 0[29]
GPIO_0[30]	PIN_J25	GPIO Connection 0[30]
GPIO_0[31]	PIN_J26	GPIO Connection 0[31]
GPIO_0[32]	PIN_L23	GPIO Connection 0[32]
GPIO_0[33]	PIN_L24	GPIO Connection 0[33]
GPIO_0[34]	PIN_L25	GPIO Connection 0[34]
GPIO_0[35]	PIN_L19	GPIO Connection 0[35]
GPIO_1[0]	PIN_K25	GPIO Connection 1[0]
GPIO_1[1]	PIN_K26	GPIO Connection 1[1]

GPIO_1[2]	PIN_M22	GPIO Connection 1[2]
GPIO_1[3]	PIN_M23	GPIO Connection 1[3]
GPIO_1[4]	PIN_M19	GPIO Connection 1[4]
GPIO_1[5]	PIN_M20	GPIO Connection 1[5]
GPIO_1[6]	PIN_N20	GPIO Connection 1[6]
GPIO_1[7]	PIN_M21	GPIO Connection 1[7]
GPIO_1[8]	PIN_M24	GPIO Connection 1[8]
GPIO_1[9]	PIN_M25	GPIO Connection 1[9]
GPIO_1[10]	PIN_N24	GPIO Connection 1[10]
GPIO_1[11]	PIN_P24	GPIO Connection 1[11]
GPIO_1[12]	PIN_R25	GPIO Connection 1[12]
GPIO_1[13]	PIN_R24	GPIO Connection 1[13]
GPIO_1[14]	PIN_R20	GPIO Connection 1[14]
GPIO_1[15]	PIN_T22	GPIO Connection 1[15]
GPIO_1[16]	PIN_T23	GPIO Connection 1[16]
GPIO_1[17]	PIN_T24	GPIO Connection 1[17]
GPIO_1[18]	PIN_T25	GPIO Connection 1[18]
GPIO_1[19]	PIN_T18	GPIO Connection 1[19]
GPIO_1[20]	PIN_T21	GPIO Connection 1[20]
GPIO_1[21]	PIN_T20	GPIO Connection 1[21]
GPIO_1[22]	PIN_U26	GPIO Connection 1[22]
GPIO_1[23]	PIN_U25	GPIO Connection 1[23]
GPIO_1[24]	PIN_U23	GPIO Connection 1[24]
GPIO_1[25]	PIN_U24	GPIO Connection 1[25]
GPIO_1[26]	PIN_R19	GPIO Connection 1[26]
GPIO_1[27]	PIN_T19	GPIO Connection 1[27]
GPIO_1[28]	PIN_U20	GPIO Connection 1[28]
GPIO_1[29]	PIN_U21	GPIO Connection 1[29]
GPIO_1[30]	PIN_V26	GPIO Connection 1[30]
GPIO_1[31]	PIN_V25	GPIO Connection 1[31]
GPIO_1[32]	PIN_V24	GPIO Connection 1[32]
GPIO_1[33]	PIN_V23	GPIO Connection 1[33]
GPIO_1[34]	PIN_W25	GPIO Connection 1[34]
GPIO_1[35]	PIN_W23	GPIO Connection 1[35]

.

Б.3.7 Використання VGA

Стенд DE2 включає 16-pin з'єднувач ПІДПИСКИ Столу, бо VGA вихід. VGA сигнали синхронізації забезпечені безпосередньо від Циклону II FPGA, і Аналоговий Пристрій ADV7123 triple 10-розрядний швидкохідний DAC, що виробляє сигнали аналогових даних (червоний, зелений, і блакитний). Асоційоване схемне рішення дається на рис. Б.3.11 і може підтримувати резолюції аж до 1600 x 1200 пікселів, в 100 мгц.

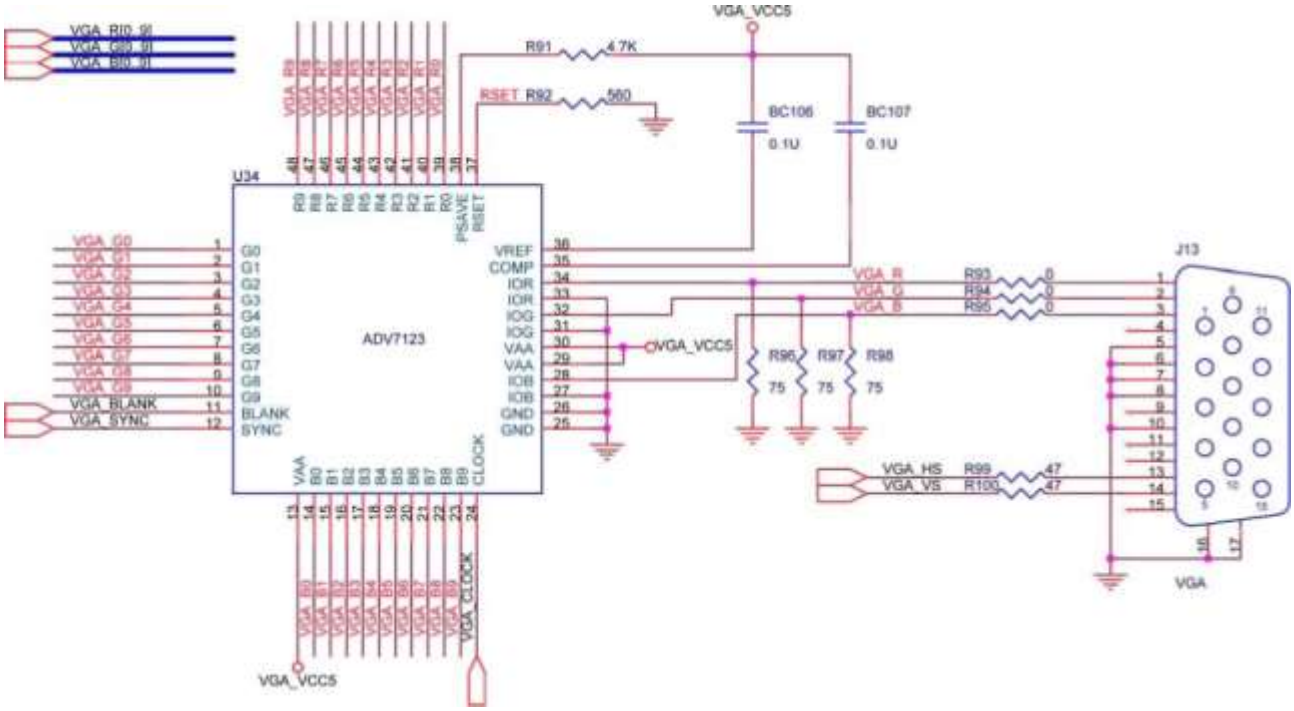


Рис. Б.3.11. Схемне рішення VGA.

Специфікація розрахунку часу для синхронізації VGA і даних RGB (червоний, зелений, блакитний) може бути знайдена на різних освітніх веб-сайтах (наприклад, знайдіть "VGA вибору часу сигналу"). Рис. Б.3.12 ілюструє основні вимоги розрахунку часу для кожного ряду (горизонталь), який показується на моніторі VGA. Низький імпульс специфічної тривалості застосовується до входу горизонтальної синхронізації (*hsync*) монітора, який означає кінець одного ряду даних і початку наступного. Введення даних (RGB) на моніторі мають бути (управляється до 0 V) на якийсь час періодом звернення до *зворотної веранди (b)* після того, як *hsync ім*пульс відбувається, який завершений інтервалом показу (c). Протягом показу даних, interval дані RGB, переїжджає кожен піксель у свою чергу ряд, що показується. Нарешті, є період часу звернувся до *передньої веранди (d),* де сигнали RGB мають знову бути перед тим, як наступний *hsync* імпульс зявиться. Вибір часу вертикальної синхронізації (*vsync*) такий же, як показано на рис. Б.3.12, за винятком того, що *vsync* імпульс означає кінець однієї рамки і початок наступної, і дані посилаються на набір рядів в рамці (горизонтальний вибір часу). Рисунки Б.3.13 і Б.3.14 показують, для різних резолюцій, тривалість періодів часу, *b, c*, і *d* як для горизонтального, так і вертикального вибору часу.

Детальна інформація для використання ADV7123 відео DAC доступна в його datasheet, який може бути знайдений на веб-сайті виготівника, і *Datasheet* на **DE2 Системі CD-ROM**. Pin призначення між Циклоном II FPGA і ADV7123 вносяться до списку в таблицю Б.3.8. Приклад коди, яка управляє показом VGA, описується в Секціях 5.2 і 5.3.

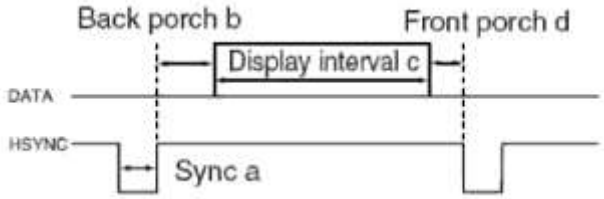


Рис. Б.3.12. Горизонтала VGA специфікація розрахунку часу.

Метод VGA		**Горизонтальна специфікація розрахунку часу**				
Конфігурація	Резолюція(HxV)	a(us)	b(us)	c(us)	d(us)	Годинник пікселя (Mhz)

VGA(60Hz)	640x480	3.8	1.9	25.4	0.6	25	(640/c)
VGA(85Hz)	640x480	1.6	2.2	17.8	1.6	36	(640/c)
SVGA(60Hz)	800x600	3.2	2.2	20	1	40	(800/c)
SVGA(75Hz)	800x600	1.6	3.2	16.2	0.3	49	(800/c)
SVGA(85Hz)	800x600	1.1	2.7	14.2	0.6	56	(800/c)
XGA(60Hz)	1024x768	2.1	2.5	15.8	0.4	65	(1024/c)
XGA(70Hz)	1024x768	1.8	1.9	13.7	0.3	75	(1024/c)
XGA(85Hz)	1024x768	1.0	2.2	10.8	0.5	95	(1024/c)
1280x1024(60Hz)	1280x1024	1.0	2.3	11.9	0.4	108	(1280/c)

Рис. Б.3.13. Горизонтальна VGA специфікація розрахунку часу.

Метод VGA		**Вертикальна специфікація розрахунку часу**			
Configuration	Resolution (HxV)	a(lines)	b(lines)	c(lines)	d(lines)
VGA(60Hz)	640x480	2	33	480	10
VGA(85Hz)	640x480	3	25	480	1
SVGA(60Hz)	800x600	4	23	600	1
SVGA(75Hz)	800x600	3	21	600	1
SVGA(85Hz)	800x600	3	27	600	1
XGA(60Hz)	1024x768	6	29	768	3
XGA(70Hz)	1024x768	6	29	768	3
XGA(85Hz)	1024x768	3	36	768	1
1280x1024(60Hz)	1280x1024	3	38	1024	1

Рис. Б.3.14. VGA вертикальна специфікація розрахунку часу.

Таблиця Б.3.8. Шпилькові призначення ADV7123.

Сигнальне Ім'я	**FPGA Pin No.**	**Description**
VGA_R[0]	PIN_C8	VGA Red[0]
VGA_R[1]	PIN_F10	VGA Red[1]
VGA_R[2]	PIN_G10	VGA Red[2]
VGA_R[3]	PIN_D9	VGA Red[3]
VGA_R[4]	PIN_C9	VGA Red[4]
VGA_R[5]	PIN_A8	VGA Red[5]
VGA_R[6]	PIN_H11	VGA Red[6]
VGA_R[7]	PIN_H12	VGA Red[7]
VGA_R[8]	PIN_F11	VGA Red[8]
VGA_R[9]	PIN_E10	VGA Red[9]
VGA_G[0]	PIN_B9	VGA Green[0]
VGA_G[1]	PIN_A9	VGA Green[1]
VGA_G[2]	PIN_C10	VGA Green[2]
VGA_G[3]	PIN_D10	VGA Green[3]
VGA_G[4]	PIN_B10	VGA Green[4]
VGA_G[5]	PIN_A10	VGA Green[5]
VGA_G[6]	PIN_G11	VGA Green[6]
VGA_G[7]	PIN_D11	VGA Green[7]
VGA_G[8]	PIN_E12	VGA Green[8]
VGA_G[9]	PIN_D12	VGA Green[9]
VGA_B[0]	PIN_J13	VGA Blue[0]
VGA_B[1]	PIN_J14	VGA Blue[1]
VGA_B[2]	PIN_F12	VGA Blue[2]
VGA_B[3]	PIN_G12	VGA Blue[3]

VGA_B[4]	PIN_J10	VGA Blue[4]
VGA_B[5]	PIN_J11	VGA Blue[5]
VGA_B[6]	PIN_C11	VGA Blue[6]
VGA_B[7]	PIN_B11	VGA Blue[7]
VGA_B[8]	PIN_C12	VGA Blue[8]
VGA_B[9]	PIN_B12	VGA Blue[9]
VGA_CLK	PIN_B8	VGA Clock
VGA_BLANK	PIN_D6	VGA BLANK
VGA_HS	PIN_A7	VGA H_SYNC
VGA_VS	PIN_D8	VGA V_SYNC
VGA_SYNC	PIN_B7	VGA SYNC

Б.3.8 Використання 24-розрядного звуку CODEC

Стенд DE2 забезпечує високоякісний 24-розрядний звук Wolfson WM8731 CODEC (enCOder/DECoder). Цей чип підтримує вхід мікрофону, лінійний вхід, і зовнішні для лінії порти, з типовою нормою, регульованою від 8 kHz до 96 kHz. WM8731 управляють серійні I2C інтерфейси шини, які з'єднується з шпильками на Циклоні II FPGA. Схематична діаграма звукової схеми показується на рис. Б.3.15, і шпилькові призначення FPGA вносяться до списку в таблицю Б.3.9. Детальна інформація для використання WM8731 codec доступна в його datasheet, який може бути знайдений на веб-сайті виготівника, і папці *Datasheet* на **DE2 Системі CD-ROM**.

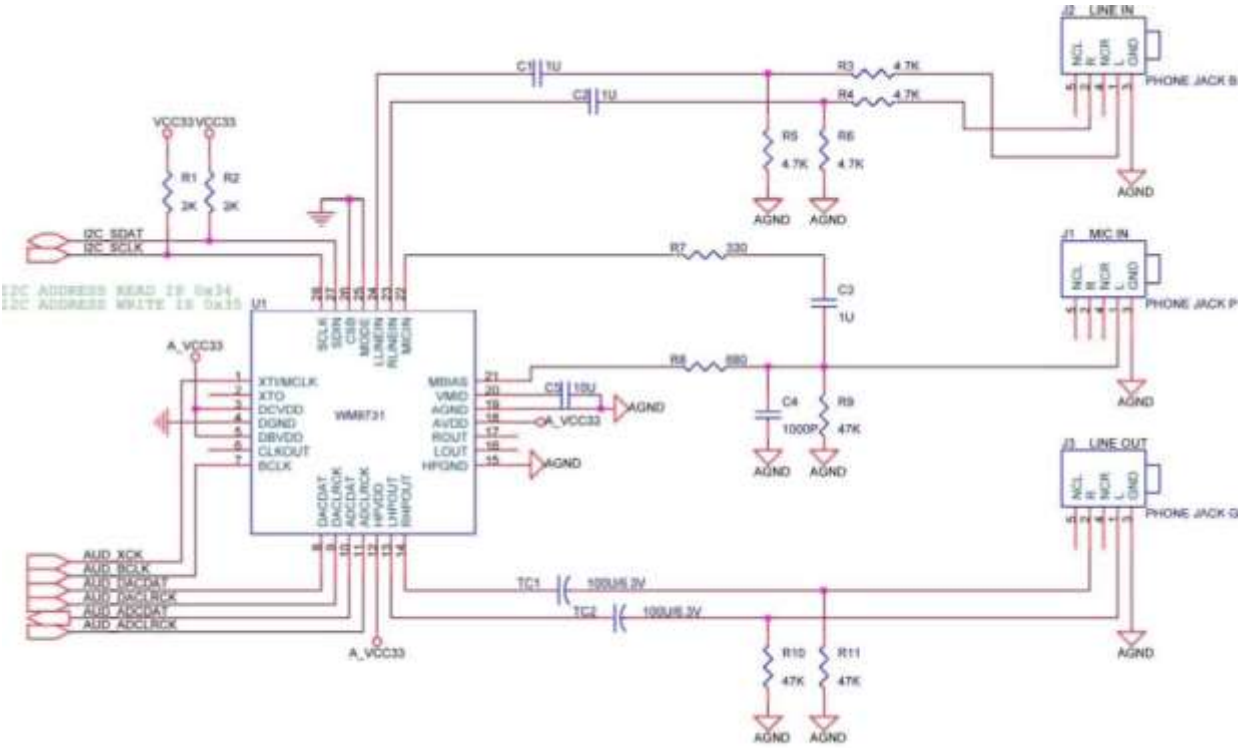


Рис. Б.3.15. Схемне рішення звукового CODEC.

Таблиця Б.3.9. Звуковий CODEC, шпилькові призначення.

Сигнальне Ім'я	**FPGA Pin**	**Опис**
AUD_ADCLRCK	PIN_C5	Audio CODEC ADC LR Clock
AUD_ADCDAT	PIN_B5	Audio CODEC ADC Data
AUD_DACLRCK	PIN_C6	Audio CODEC DAC LR Clock
AUD_DACDAT	PIN_A4	Audio CODEC DAC Data
AUD_XCK	PIN_A5	Audio CODEC Chip Clock

AUD_BCLK	PIN_B4	Audio CODEC Bit-Stream Clock
I2C_SCLK	PIN_A6	I2C Data
I2C_SDAT	PIN_B6	I2C Clock

Б.3.9 Порт серії RS-232

Стенд DE2 використовує MAX232 чип приймача і 9-pin з'єднувач для комунікацій RS-232. Для детальної інформації відносно того, як використовувати приймач посилаються на datasheet, який доступний на веб-сайті виготівника, і в папці *Datasheet* на **DE2 Системі CD-ROM**. Рисунок Б.3.16 показує зв'язані схемні рішення, і списки (таблиця Б.3.10) Циклон II pin призначення FPGA.

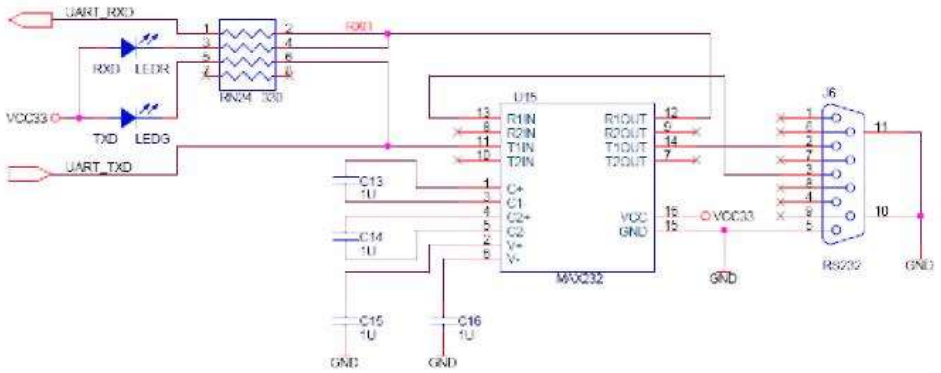


Рис. Б.3.16. MAX232 (RS-232) схемне вирішення чіпа.

Таблиця Б.3.10. Pin призначення RS-232.

Сигнальне Ім'я	FPGA Pin No.	Опис
UART_RXD	PIN_C25	UART Одержувач
UART_TXD	PIN_B25	UART Відправник

Б.3.10 Порт серії PS/2

Стенд DE2 включає стандартний PS/2 з'єднувач для клавіатури PS/2 або миші. Рисунок Б.3.17 показує схематичне вирішення PS/2. Інструкції для використання миші PS/2 або клавіатури можна знайти виконання відповідного пошуку на різних освітніх веб-сайтах. Pin призначення для асоційованого інтерфейсу показуються в таблиці Б.3.11.

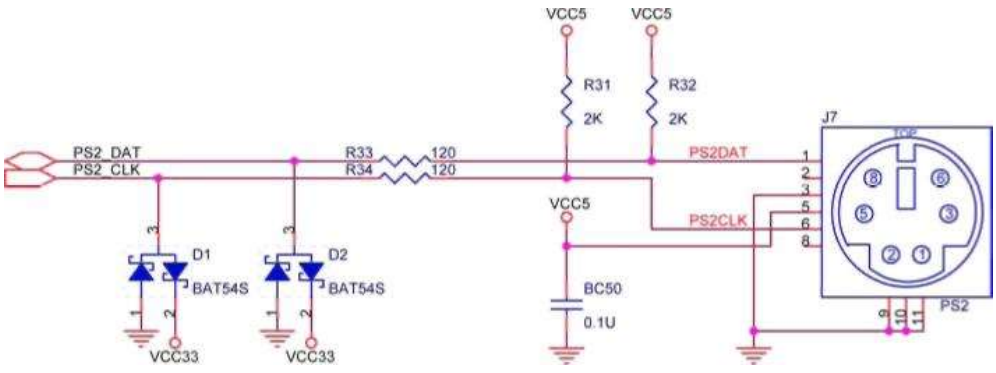


Рис. Б.3.17. Схематичне рішення PS/2

Таблиця Б.3.11. Шпилькові призначення PS/2.

Сигнальне Ім'я	FPGA Pin No.	Опис
PS2_CLK	PIN_D26	PS/2 Clock
PS2_DAT	PIN_C24	PS/2 Data

Б.3.11. Диспетчер мережі швидкого Ethernet

Стенд DE2 забезпечує підтримку Ethernet диспетчера Пост Davicom DM9000A Ethernet. DM9000A включає загальний інтерфейс процесора, 16 Кілобайт SRAM, пристрій управління (MAC) медіа-доступу і 10/100M PHY приймач. Рисунок Б.3.18 показує схематичне рішення для інтерфейсу Швидкий Ethernet, і асоційовані pin призначення вносяться до списку в таблицю Б.3.12. Для детальної інформації відносно того, як використовувати DM9000A посилаються на його datasheet і примітку по вживанню, яка доступна на веб-сайті виготівника, і в папці *Datasheet* на **DE2 Системі CD-ROM**.

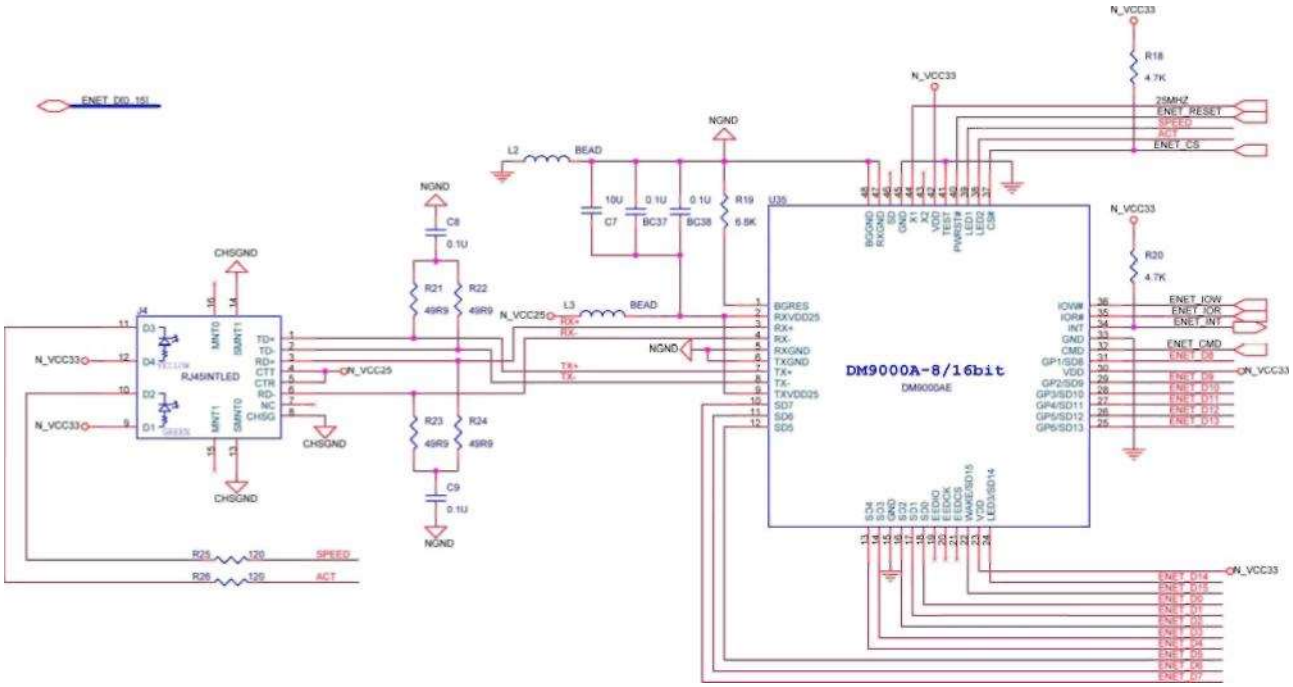


Рис. Б.3.18. Схематичне рішення Швидкий Ethernet.

Таблиця Б.3.12. Швидкий Ethernet, шпилькові призначення.

Signal Name	FPGA Pin No.	Description
ENET_DATA[0]	PIN_D17	DM9000A DATA[0]
ENET_DATA[1]	PIN_C17	DM9000A DATA[1]
ENET_DATA[2]	PIN_B18	DM9000A DATA[2]
ENET_DATA[3]	PIN_A18	DM9000A DATA[3]
ENET_DATA[4]	PIN_B17	DM9000A DATA[4]
ENET_DATA[5]	PIN_A17	DM9000A DATA[5]
ENET_DATA[6]	PIN_B16	DM9000A DATA[6]
ENET_DATA[7]	PIN_B15	DM9000A DATA[7]
ENET_DATA[8]	PIN_B20	DM9000A DATA[8]
ENET_DATA[9]	PIN_A20	DM9000A DATA[9]
ENET_DATA[10]	PIN_C19	DM9000A DATA[10]
ENET_DATA[11]	PIN_D19	DM9000A DATA[11]

ENET_DATA[12]	PIN_B19	DM9000A DATA[12]
ENET_DATA[13]	PIN_A19	DM9000A DATA[13]
ENET_DATA[14]	PIN_E18	DM9000A DATA[14]
ENET_DATA[15]	PIN_D18	DM9000A DATA[15]
ENET_CLK	PIN_B24	DM9000A Clock 25 MHz
ENET_CMD	PIN_A21	DM9000A Command/Data Select, 0 = Command, 1 = Data
ENET_CS_N	PIN_A23	DM9000A Chip Select
ENET_INT	PIN_B21	DM9000A Interrupt
ENET_RD_N	PIN_A22	DM9000A Read
ENET_WR_N	PIN_B22	DM9000A Write
ENET_RST_N	PIN_B23	DM9000A Reset

Б.3.12 Декодер TV

Стенд DE2 обладнаний аналоговим пристроєм ADV7181 мікросхемою декодера ТБ. ADV7181 - интегральний відеодекодер, що автоматично перетворює стандартний сигнал (NTSC і SECAM) телебачення основної смуги частот аналога на 4:2:2 складовий відеодані compatible з 16-bit/8-bit CCIR601/CCIR656. ADV7181 сумісний з широким діапазоном відеопристроїв, у тому числі DVD, broadcast джерела, і security/surveillance камери.

Записи в декодері TV може програмувати серійний інтерфейс I2C, який з'єднується з Циклоном II FPGA, як вказано на рисунку Б.3.19. Pin призначення вносяться до списку в таблиці Б.3.13. Детальна інформація відносно ADV7181 доступна на веб-сайті виготівника, і папці *Datasheet* на **DE2 Системі CD-ROM**.

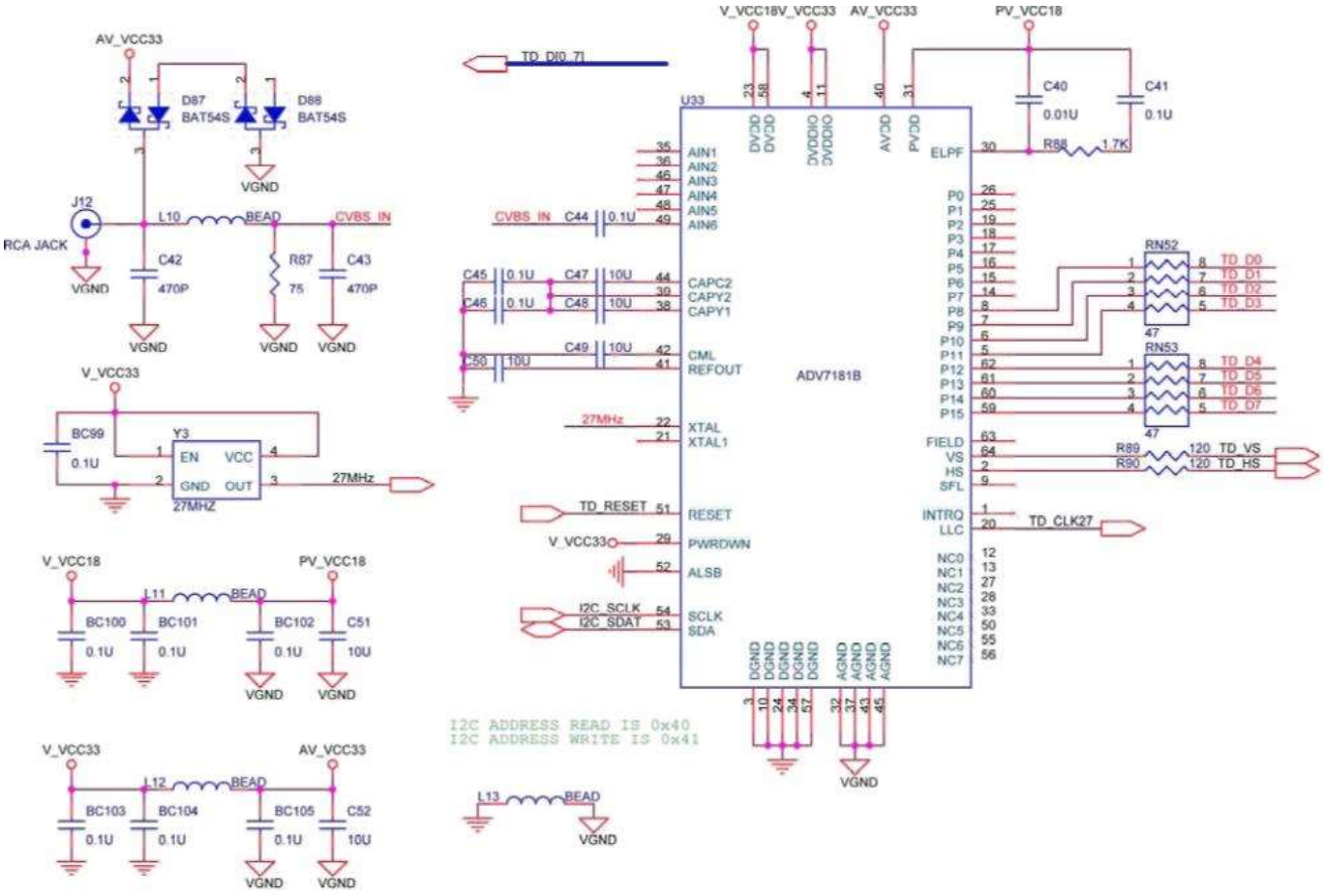


Рис. Б.3.19. Схемне вирішення декодера TV.

Таблиця Б.3.13. Шпилькові призначення Декодера TV.

Сигнальне Ім'я	**FPGA Pin No.**	**Опис**
TD_DATA[0]	PIN_J9	Дані Декодера TV[0]
TD_DATA[1]	PIN_E8	Дані Декодера TV[1]
TD_DATA[2]	PIN_H8	Дані Декодера TV[2]
TD_DATA[3]	PIN_H10	Дані Декодера TV[3]
TD_DATA[4]	PIN_G9	Дані Декодера TV[4]
TD_DATA[5]	PIN_F9	Дані Декодера TV[5]
TD_DATA[6]	PIN_D7	Дані Декодера TV[6]
TD_DATA[7]	PIN_C7	Дані Декодера TV[7]
TD_HS	PIN_D5	Декодер TV H_SYNC
TD_VS	PIN_K9	Декодер TV V_SYNC
TD_CLK27	PIN_C16	Вхід Годинника Декодера TV.
TD_RESET	PIN_C4	Скидання Декодера TV
I2C_SCLK	PIN_A6	Дані I2C
I2C_SDAT	PIN_B6	Годинник I2C

Б.3.13 Шифратор TV

Хоча стенд DE2 не включає TV чип шифратора, ADV7123 (10-розрядний швидкохідний потрійний ADCs) може бути застосовано, щоб здійснювати якісний для професіонала шифратор TV з цифровою оброблювальною частиною, здійснюваною в Циклоні II FPGA. Рисунок Б.3.20 показує блок-схему шифратора TV, що здійснена в цій формі.

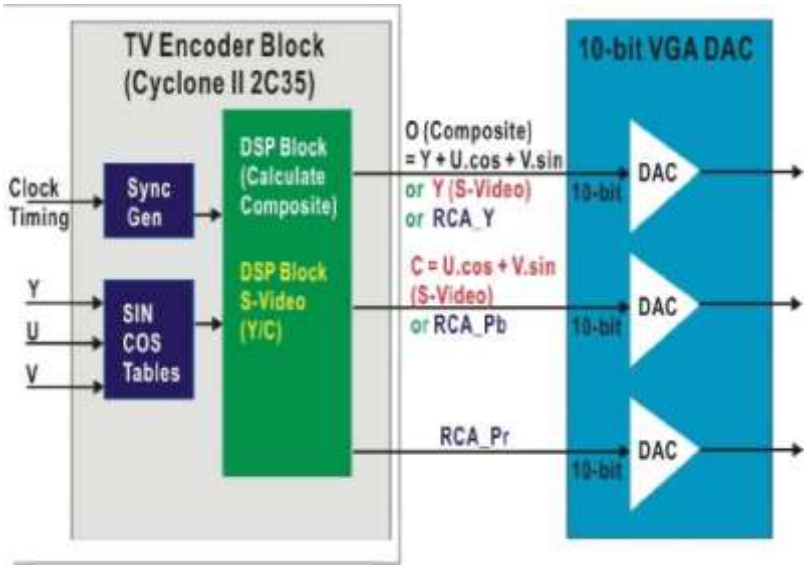


Рис. Б.3.20. Шифратор TV, який використовує Циклон II FPGA і ADV7123.

Б.3.14 Використання USB

Стенд DE2 забезпечує USB вхід, так і пристрій зв'язку використанням Philips ISP1362 USB диспетчера в одній мікросхемі. Диспетчери USB і пристрою працюють з Універсальним Серійним Обміном Шинної Специфікації 2.0, підтримуючи передачу даних full-speed (12 Mbit/s) і low-speed (1.5 Mbit/s). На рисунку Б.3.21 показана схематична діаграма схеми USB; pin призначення для асоційованого інтерфейсу вносяться до списку в таблицю Б.3.14.

Детальна інформація для використання пристрою ISP1362 доступна в його datasheet і програмному керівництві; ці документи можуть бути знайдені на веб-сайті виготівника, так і в папці *Datasheet* на **DE2 Системі CD-ROM**. Більш всього повна частина додатка USB знаходиться в проекті програмного драйвера. Два повні приклади управління USB для приймання даних, так і додатки пристрою, можуть бути знайдені в секціях 5.3 і 5.4 документації. Ці демонстрації забезпечують приклади програмних драйверів для процесора Nios II.

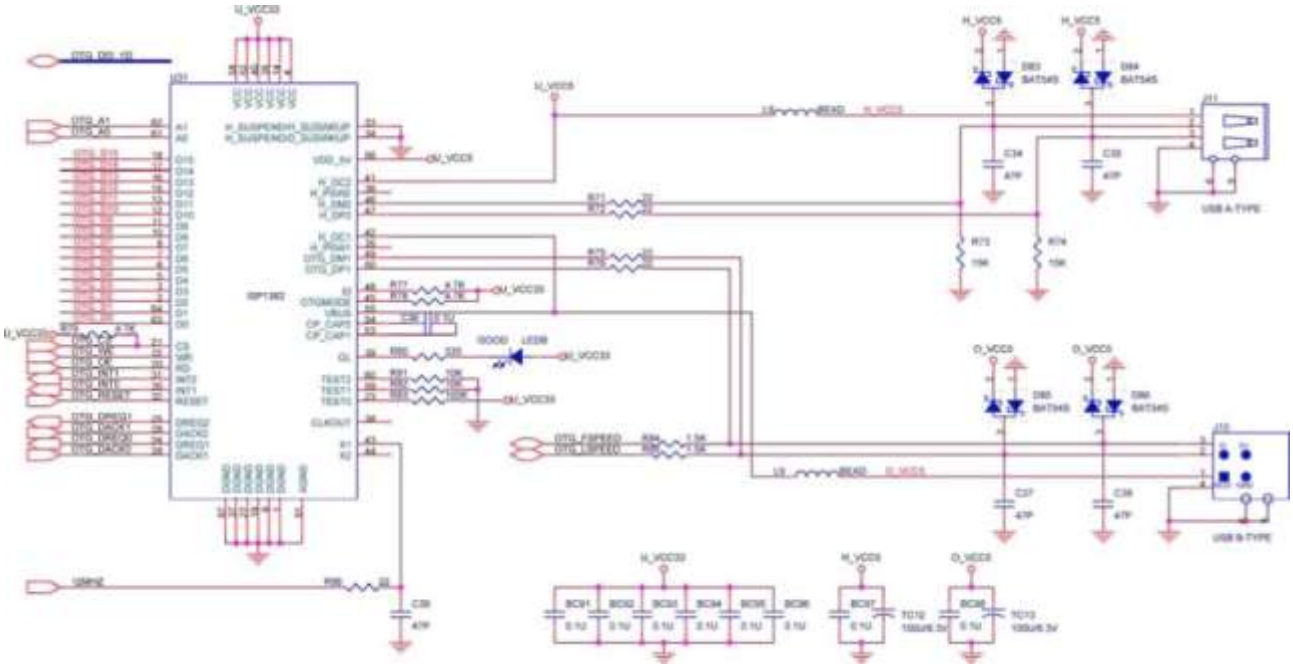


Рис. Б.3.21. Контролер USB (ISP1362) і схемне вирішення пристрою.

Таблиця Б.3.14. Шпилькові призначення USB (ISP1362).

Signal Name	**FPGA Pin**	**Description**
OTG_ADDR[0]	PIN_K7	ISP1362 Адреса[0]
OTG_ADDR[1]	PIN_F2	ISP1362 Адреса[1]
OTG_DATA[0]	PIN_F4	ISP1362 Дані[0]
OTG_DATA[1]	PIN_D2	ISP1362 Дані[1]
OTG_DATA[2]	PIN_D1	ISP1362 Дані[2]
OTG_DATA[3]	PIN_F7	ISP1362 Дані[3]
OTG_DATA[4]	PIN_J5	ISP1362 Дані[4]
OTG_DATA[5]	PIN_J8	ISP1362 Дані[5]
OTG_DATA[6]	PIN_J7	ISP1362 Дані[6]
OTG_DATA[7]	PIN_H6	ISP1362 Дані[7]
OTG_DATA[8]	PIN_E2	ISP1362 Дані[8]
OTG_DATA[9]	PIN_E1	ISP1362 Дані[9]
OTG_DATA[10]	PIN_K6	ISP1362 Дані[10]
OTG_DATA[11]	PIN_K5	ISP1362 Дані[11]
OTG_DATA[12]	PIN_G4	ISP1362 Дані[12]
OTG_DATA[13]	PIN_G3	ISP1362 Дані[13]

OTG_DATA[14]	PIN_J6	ISP1362 Дані[14]
OTG_DATA[15]	PIN_K8	ISP1362 Дані[15]
OTG_CS_N	PIN_F1	ISP1362 Chip Select
OTG_RD_N	PIN_G2	ISP1362 Read
OTG_WR_N	PIN_G1	ISP1362 Write
OTG_RST_N	PIN_G5	ISP1362 Reset
OTG_INT0	PIN_B3	ISP1362 Interrupt 0
OTG_INT1	PIN_C3	ISP1362 Interrupt 1
OTG_DACK0_N	PIN_C2	ISP1362 DMA Acknowledge 0
OTG_DACK1_N	PIN_B2	ISP1362 DMA Acknowledge 1
OTG_DREQ0	PIN_F6	ISP1362 DMA Request 0
OTG_DREQ1	PIN_E5	ISP1362 DMA Request 1
OTG_FSPEED	PIN_F3	USB Full Speed, 0 = Enable, Z = Disable
OTG_LSPEED	PIN_G6	USB Low Speed, 0 = Enable, Z = Disable

Б.3.15 Використання IRDA

Стенд DE2 забезпечує прості ЗМІ безпровідні комунікації, використовуючи Agilent енергетичний інфрачервоний приймач низького рівня HSDL-3201. Datasheet для цього пристрою забезпечений в папці *Datasheet\IrDA* на **DE2 Системі CD-ROM**. Відзначте, що найвища підтримувана норма передачі складає 115.2 Kbit/s і як TX, так і RXD сторони доведеться використати ту ж норму передачі. Рисунок Б.3.22 показує схематичне вирішення IRDA зв'язку комунікації. Зверніться до наступного веб-вузла за детальною інформацією відносно того, як послати і отримати дані, використовуючи зв'язок IRDA: <http://techtrain.microchip.com/webseminars/documents/IrDA_BW.pdf>. Pin призначення асоційованого інтерфейсу вносяться до списку в таблицю Б.3.15.

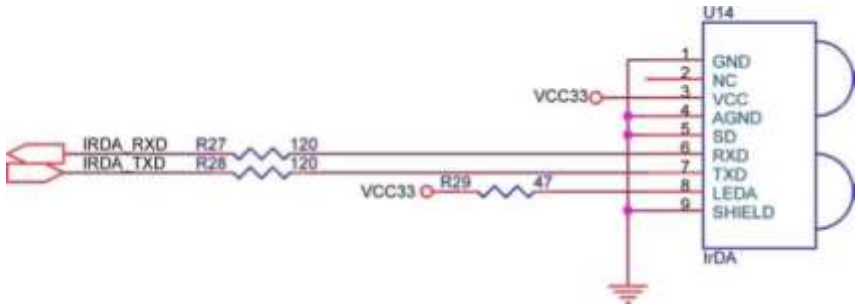


Рис. Б.3.22. Схемне рішення IRDA.

Таблиця Б.3.15. Шпилькові призначення IRDA.

Signal Name	FPGA Pin	Description
IRDA_TXD	PIN_AE24	IRDA Transmitter
IRDA_RXD	PIN_AE25	IRDA Receiver

Б.3.16 Використання SDRAM/SRAM/Flash

Стенд DE2 забезпечує 8-мегабайтний SDRAM, 512-кілобайтний SRAM, і пам'ять Flash 4-мегабайт (1-мегабайти на деяких правліннях). Рисунки Б.3.23, Б.3.24 і Б.3.25 показують схемні вирішення чіпів пам'яті. Pin призначення для кожного пристрою вносяться до списку в таблиці

Б.3.16, Б.3.17 і Б.3.18. Datasheets для чіпів пам'яті забезпечені в папці *Datasheet* на **DE2 Системі CD-ROM**.

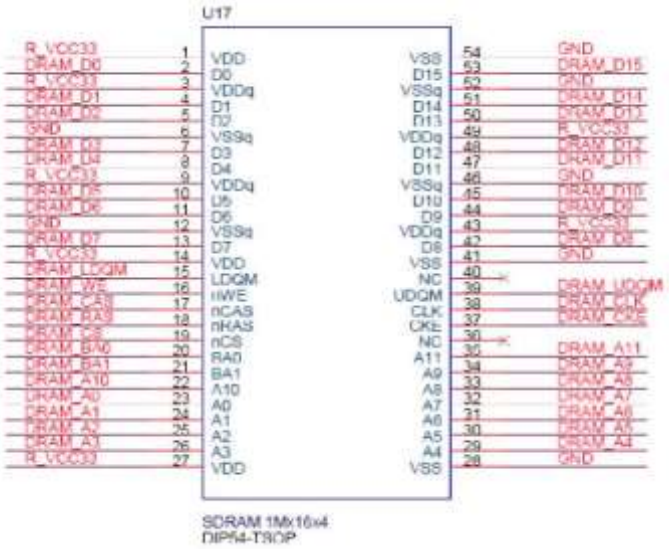


Рис. Б.3.23. Схемне рішення SDRAM.

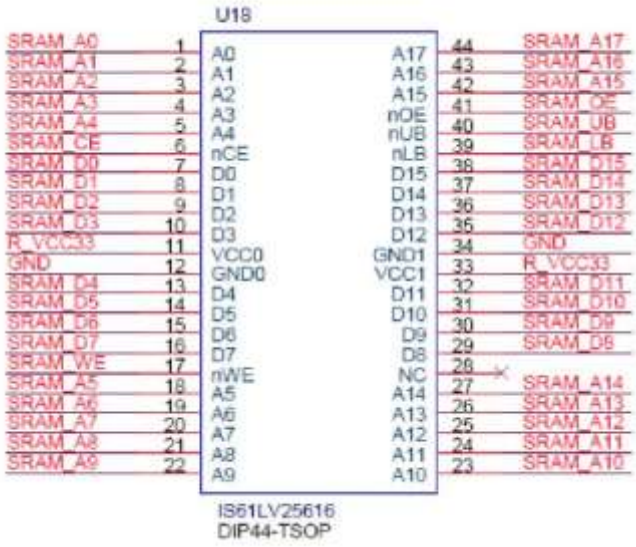


Рис. Б.3.24. Схемне рішення SRAM



Рис. Б.3.25. Flash схемне рішення.

Таблиця Б.3.16. Шпилькові призначення SDRAM.

Signal Name	**FPGA Pin No.**	**Description**
DRAM_ADDR[0]	PIN_T6	SDRAM Address[0]
DRAM_ADDR[1]	PIN_V4	SDRAM Address[1]
DRAM_ADDR[2]	PIN_V3	SDRAM Address[2]
DRAM_ADDR[3]	PIN_W2	SDRAM Address[3]
DRAM_ADDR[4]	PIN_W1	SDRAM Address[4]
DRAM_ADDR[5]	PIN_U6	SDRAM Address[5]
DRAM_ADDR[6]	PIN_U7	SDRAM Address[6]
DRAM_ADDR[7]	PIN_U5	SDRAM Address[7]
DRAM_ADDR[8]	PIN_W4	SDRAM Address[8]
DRAM_ADDR[9]	PIN_W3	SDRAM Address[9]
DRAM_ADDR[10]	PIN_Y1	SDRAM Address[10]
DRAM_ADDR[11]	PIN_V5	SDRAM Address[11]
DRAM_DQ[0]	PIN_V6	SDRAM Data[0]
DRAM_DQ[1]	PIN_AA2	SDRAM Data[1]
DRAM_DQ[2]	PIN_AA1	SDRAM Data[2]
DRAM_DQ[3]	PIN_Y3	SDRAM Data[3]
DRAM_DQ[4]	PIN_Y4	SDRAM Data[4]
DRAM_DQ[5]	PIN_R8	SDRAM Data[5]
DRAM_DQ[6]	PIN_T8	SDRAM Data[6]
DRAM_DQ[7]	PIN_V7	SDRAM Data[7]
DRAM_DQ[8]	PIN_W6	SDRAM Data[8]
DRAM_DQ[9]	PIN_AB2	SDRAM Data[9]
DRAM_DQ[10]	PIN_AB1	SDRAM Data[10]
DRAM_DQ[11]	PIN_AA4	SDRAM Data[11]
DRAM_DQ[12]	PIN_AA3	SDRAM Data[12]

DRAM_DQ[13]	PIN_AC2	SDRAM Data[13]
DRAM_DQ[14]	PIN_AC1	SDRAM Data[14]
DRAM_DQ[15]	PIN_AA5	SDRAM Data[15]
DRAM_BA_0	PIN_AE2	SDRAM Bank Address[0]
DRAM_BA_1	PIN_AE3	SDRAM Bank Address[1]
DRAM_LDQM	PIN_AD2	SDRAM Low-byte Data Mask
DRAM_UDQM	PIN_Y5	SDRAM High-byte Data Mask
DRAM_RAS_N	PIN_AB4	SDRAM Row Address Strobe
DRAM_CAS_N	PIN_AB3	SDRAM Column Address Strobe
DRAM_CKE	PIN_AA6	SDRAM Clock Enable
DRAM_CLK	PIN_AA7	SDRAM Clock
DRAM_WE_N	PIN_AD3	SDRAM Write Enable
DRAM_CS_N	PIN_AC3	SDRAM Chip Select

Таблиця Б.3.17. SRAM шпилькові призначення.

Signal Name	**FPGA Pin No.**	**Description**
SRAM_ADDR[0]	PIN_AE4	SRAM Address[0]
SRAM_ADDR[1]	PIN_AF4	SRAM Address[1]
SRAM_ADDR[2]	PIN_AC5	SRAM Address[2]
SRAM_ADDR[3]	PIN_AC6	SRAM Address[3]
SRAM_ADDR[4]	PIN_AD4	SRAM Address[4]
SRAM_ADDR[5]	PIN_AD5	SRAM Address[5]
SRAM_ADDR[6]	PIN_AE5	SRAM Address[6]
SRAM_ADDR[7]	PIN_AF5	SRAM Address[7]
SRAM_ADDR[8]	PIN_AD6	SRAM Address[8]
SRAM_ADDR[9]	PIN_AD7	SRAM Address[9]
SRAM_ADDR[10]	PIN_V10	SRAM Address[10]
SRAM_ADDR[11]	PIN_V9	SRAM Address[11]
SRAM_ADDR[12]	PIN_AC7	SRAM Address[12]
SRAM_ADDR[13]	PIN_W8	SRAM Address[13]
SRAM_ADDR[14]	PIN_W10	SRAM Address[14]
SRAM_ADDR[15]	PIN_Y10	SRAM Address[15]
SRAM_ADDR[16]	PIN_AB8	SRAM Address[16]
SRAM_ADDR[17]	PIN_AC8	SRAM Address[17]
SRAM_DQ[0]	PIN_AD8	SRAM Data[0]
SRAM_DQ[1]	PIN_AE6	SRAM Data[1]
SRAM_DQ[2]	PIN_AF6	SRAM Data[2]
SRAM_DQ[3]	PIN_AA9	SRAM Data[3]
SRAM_DQ[4]	PIN_AA10	SRAM Data[4]
SRAM_DQ[5]	PIN_AB10	SRAM Data[5]
SRAM_DQ[6]	PIN_AA11	SRAM Data[6]
SRAM_DQ[7]	PIN_Y11	SRAM Data[7]

SRAM_DQ[8]	PIN_AE7	SRAM Data[8]
SRAM_DQ[9]	PIN_AF7	SRAM Data[9]
SRAM_DQ[10]	PIN_AE8	SRAM Data[10]
SRAM_DQ[11]	PIN_AF8	SRAM Data[11]
SRAM_DQ[12]	PIN_W11	SRAM Data[12]
SRAM_DQ[13]	PIN_W12	SRAM Data[13]
SRAM_DQ[14]	PIN_AC9	SRAM Data[14]
SRAM_DQ[15]	PIN_AC10	SRAM Data[15]
SRAM_WE_N	PIN_AE10	SRAM Write Enable
SRAM_OE_N	PIN_AD10	SRAM Output Enable
SRAM_UB_N	PIN_AF9	SRAM High-byte Data Mask
SRAM_LB_N	PIN_AE9	SRAM Low-byte Data Mask
SRAM_CE_N	PIN_AC11	SRAM Chip Enable

Таблиця Б.3.18. Flash pin призначення.

Signal Name	**FPGA Pin No.**	**Description**
FL_ADDR[0]	PIN_AC18	FLASH Address[0]
FL_ADDR[1]	PIN_AB18	FLASH Address[1]
FL_ADDR[2]	PIN_AE19	FLASH Address[2]
FL_ADDR[3]	PIN_AF19	FLASH Address[3]
FL_ADDR[4]	PIN_AE18	FLASH Address[4]
FL_ADDR[5]	PIN_AF18	FLASH Address[5]
FL_ADDR[6]	PIN_Y16	FLASH Address[6]
FL_ADDR[7]	PIN_AA16	FLASH Address[7]
FL_ADDR[8]	PIN_AD17	FLASH Address[8]
FL_ADDR[9]	PIN_AC17	FLASH Address[9]
FL_ADDR[10]	PIN_AE17	FLASH Address[10]
FL_ADDR[11]	PIN_AF17	FLASH Address[11]
FL_ADDR[12]	PIN_W16	FLASH Address[12]
FL_ADDR[13]	PIN_W15	FLASH Address[13]
FL_ADDR[14]	PIN_AC16	FLASH Address[14]
FL_ADDR[15]	PIN_AD16	FLASH Address[15]
FL_ADDR[16]	PIN_AE16	FLASH Address[16]
FL_ADDR[17]	PIN_AC15	FLASH Address[17]
FL_ADDR[18]	PIN_AB15	FLASH Address[18]
FL_ADDR[19]	PIN_AA15	FLASH Address[19]
FL_ADDR[20]	PIN_Y15	FLASH Address[20]
FL_ADDR[21]	PIN_Y14	FLASH Address[21]
FL_DQ[0]	PIN_AD19	FLASH Data[0]
FL_DQ[1]	PIN_AC19	FLASH Data[1]

FL_DQ[2]	PIN_AF20	FLASH Data[2]
FL_DQ[3]	PIN_AE20	FLASH Data[3]
FL_DQ[4]	PIN_AB20	FLASH Data[4]
FL_DQ[5]	PIN_AC20	FLASH Data[5]
FL_DQ[6]	PIN_AF21	FLASH Data[6]
FL_DQ[7]	PIN_AE21	FLASH Data[7]
FL_CE_N	PIN_V17	FLASH Chip Enable
FL_OE_N	PIN_W17	FLASH Output Enable
FL_RST_N	PIN_AA18	FLASH Reset
FL_WE_N	PIN_AA17	FLASH Write Enable