ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КЕРУЮЧИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Кафедра спеціалізованих комп'ютерних систем

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт
з дисциплін

«ТЕХНОЛОГІЇ ТА АВТОМАТИЗАЦІЯ ПРОЕКТУВАННЯ ПРИСТРОЇВ КОМП'ЮТЕРНИХ СИСТЕМ», «ТЕОРІЯ І ПРОЕКТУВАННЯ КОМП'ЮТЕРНИХ СИСТЕМ»

Харків – 2019
Методичні вказівки розглянуто і рекомендовано до друку

на засіданні кафедри спеціалізованих комп’ютерних систем 25 лютого 2019 р., протокол № 9.

Описано основи проектування і верифікації складних цифрових систем та їх імплементації в кристали програмувальної логіки, використання мов опису апаратури (HDL та Verylog) з технологіями структурного і поведінкового опису проектів.

Методичні вказівки призначено для студентів напряму 123 «Комп'ютерна інженерія», що вивчають курс «Технології та автоматизація проектування пристроїв комп’ютерних систем» та «Теорія і проектування комп’ютерних систем» денної та заочної форм навчання.

Укладачі:

проф. М. А. Мірошник,
доц. Л. А. Клименко

Рецензент

доц. Н. А. Корольова

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт
з дисциплін
«ТЕХНОЛОГІЇ ТА АВТОМАТИЗАЦІЯ ПРОЕКТУВАННЯ ПРИСТРОЇВ КОМП’ЮТЕРНИХ СИСТЕМ»,
«ТЕОРІЯ І ПРОЕКТУВАННЯ КОМП’ЮТЕРНИХ СИСТЕМ»

Відповідальний за випуск Мірошник М. А.

Редактор Еткало О. О.

Підписано до друку 25.03.19 р.

Формат паперу 60x84 1/16. Папір писальний.

Умовн.-друк. арк. 4,0. Тираж 50. Замовлення №

Видавець та виготовлювач Український державний університет залізничного транспорту,
61050, Харків-50, майдан Фейєрбаха, 7.
Свідоцтво суб’єкта видавничої справи ДК № 6100 від 21.03.2018 р.

ЗМІСТ

ВСТУП

Методичні вказівки призначені для проведення лабораторних робіт, метою яких є оволодіння основними прийомами проектування цифрових пристроїв і використанням програм моделювання Active-HDL і синтезу Synplify Pro, які входять до складу САПР цифрових пристроїв на програмувальних логічних інтегральних схемах (ПЛІС).

У цей час ПЛІС набули великого поширення завдяки універсальності, простоті програмування, скороченню циклу проектування кінцевого пристрою, гнучкості, доступності засобів розробки.

На сьогодні найбільш актуальними вважаються системи, які працюють з декількома методами формалізації, що дають змогу виконувати строгий аналіз у реальному часі. Кілька видів формалізації необхідно при проектуванні складних систем, що складаються з компонентів з різними методами опису. Також це необхідно в ситуації, коли система розробляється різними групами проектувальників або розробників, що використовують різні методи формалізації, або при використанні раніше створених компонентів, наданих різними мовами опису. Багато вбудованих систем проектуються для роботи в реальному часі. Тому деякі методи включають алгоритми, що дають можливість виконувати оцінку часових параметрів розроблюваних пристроїв. Існуючі системи використовують широкий діапазон мов опису, деякі використовують орієнтовані на реалізацію мови, подібні до C і VHDL. Сучасні SoCs містять як мінімум один процесор, який програмно реалізує додаткові функції та управління пристроєм, що підвищує гнучкість системи. Програмне забезпечення описується та розробляється до появи апаратних компонентів. Це потребує віртуальних прототипів hardware для налагодження software і коштів для виконання одночасної верифікації програмної та апаратної частей проекту.

Мета лабораторного практикуму – ознайомитися з основами проектування і верифікації складних цифрових систем та їх імплементації в кристали програмувальної логіки, навчитись використовувати мови опису апаратури (HDL) з технологіями структурного і поведінкового опису проектів.

Прикладом загальної мови опису проектів є VHDL (Hardware Description Language – мова опису апаратури на базі надшвидкісних інтегральних схем), яка є формальним записом призначеної для опису функції і логічної організації цифрових систем. Мова VHDL у цей час використовується як міжнародний стандарт опису обчислювальних систем (ОС) будь-якого рівня складності (мікросхема, плата, блок, пристрій, комплекс).

ЛАБОРАТОРНА РОБОТА 1
Ознайомлення із середовищем Project Navigator пакета Xilinx ISE і програмування FPGA на платі Spartan-3E Starter Kit

Мета роботи

Ознайомлення із середовищем Project Navigator пакета Xilinx ISE на прикладі проектування простої комбінаційної схеми.

1.1 Вказівки до виконання лабораторної роботи

1.1.1 Огляд Project Navigator

Project Navigator розділений на чотири вікна (рисунок 1.1). У лівому верхньому куті розташовується вікно Sources (джерела), яке ієрархічно відображає елементи, включені в проект для поточної стадії процесу проектування. Під ним розташовується вікно Processes (процеси), яке відображає наявні процеси для джерела, обраного в цей момент. Третє вікно внизу – Console відображає повідомлення, помилки і попередження. Четверте вікно справа вгорі – це вікно для перегляду і редагування файлів проекту. Кожне вікно може бути переміщене або змінене в розмірах у межах Project Navigator. Початкове розташування може бути завжди відновлено шляхом вибору опцій View → Restore Default Layout.

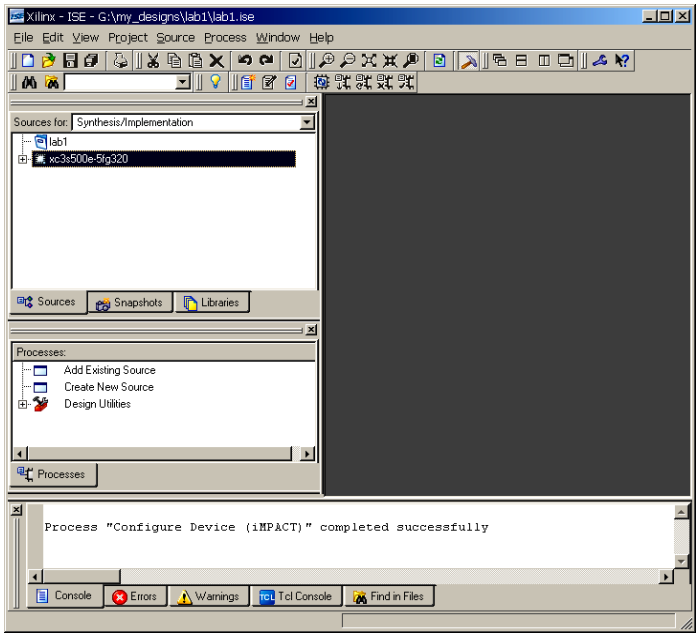


Рисунок 1.1 – Основне вікно середовища Project Navigator

Вгорі вікна **Sources** є рядок вибірки, який дає змогу користувачеві вказувати поточну фазу процесу проектування – Synthesis / Implementation (синтез / імплементація), Behavioral Simulation (моделювання поведінки), Post Route Simulation (моделювання після трасування). Це дає можливість для Project Navigator зменшувати плутанину (непорядок) шляхом відображення істотних файлів і процесів. Вікно Sources містить три закладки, що надають інформацію користувачеві.

Закладка **Sources** (ім'я проекту) призначена для визначення документів користувача, тип пристрою і вихідних файлів проекту. Кожен файл у закладці Sources має пов'язану з ним іконку. Іконка показує тип файлу. Якщо файл має у собі більш низький рівень ієрархії, зліва від іконки стоїть знак «+». VHDL-файли можуть мати знак «+», щоб показати безліч модулів усередині одного файлу. Ви можете розширювати ієрархію натисканням на «+». Ви можете відкрити файл для редагування подвійним натисканням на ім'я файлу.

Закладка **Snapshot** (моментальний знімок) відображає всі фрагменти, пов'язані з поточним проектом у Project Navigator. Моментальний знімок є копією проекту, включаючи всі файли в робочій директорії і в піддиректоріях синтезу і моделювання. Моментальний знімок зберігається в проекті, для якого він був обраний, і може бути переглянутий у закладці Snapshot. Уся

інформація, відображена в Snapshot, має статус «тільки для читання».

Закладка **Library View** показує всі бібліотеки, пов'язані з проектом, відкритим у Project Navigator.

Вікно **Processes** містить закладку Processes. Вона є контекстно залежною і чутливою до змін, основаних на обраних фазах процесу проектування та обраному джерелі вікна Sources. У Processes є доступ до: Project File Management, Design Entry Utilities, User Constraints, Synthesis, Implement Design, Generate Programming File, Behavioral (Functional) Simulation, Post Route (Timing) Simulation.

Вікно **Processes** містить у собі технологію «auto make». Це дає можливість користувачеві запускати будь-який процес у потоці проектування і програмне забезпечення автоматично запускає будь-які попередні процеси, необхідні для виконання бажаного кроку. Наприклад, коли ви запускаєте процес Implementation (імплементація), також запуститься процес синтезу – Synthesis, якщо необхідно, оскільки імплементація залежить від оновлених результатів синтезу.

Вікно **Console** показує помилки, попередження та інформаційні повідомлення. Повідомлення про помилки та попередження можуть переглядатися окремо від текстових повідомлень у Console за допомогою вибору або закладки Warnings, або закладки Errors в нижній частині вікна Console. Ви можете переходити від повідомлення про помилку (або попередження) у вікні Console до локалізації цієї помилки у вихідному файлі. Щоб зробити це, виберіть повідомлення про помилку (попередження), натисніть на ньому правою кнопкою миші і виберіть у меню Goto Source. При цьому відкриється вихідний файл і курсор опиниться на лінії з помилкою.

Ви також можете переходити від повідомлень про помилки та попередження до реєстрації серйозних проблем для отримання допомоги в їх вирішенні на сайті Xilinx. Такий тип помилок може бути ідентифікований web-іконкою зліва від помилки. Щоб перейти до реєстрації проблеми, виберіть повідомлення про помилку (або попередження), натисніть на ньому правою кнопкою миші і виберіть у меню Go to Solution Record. Web

browser відкриє і покаже всі роз'яснення, що стосуються цього повідомлення.

У четвертому вікні є доступ до текстового редактора ISE Text Editor і зразків мовних конструкцій ISE Language Templates. Текстовий редактор дає можливість редагувати вихідні файли, а також доступ до ISE Language Templates, які можна використовувати і модифікувати.

1.1.2 Створення нового проекту

Проект, описаний у цій роботі, містить простий двовходовий елемент XOR. Проект буде описуватися мовою опису апаратури VHDL. Виберіть Start → Programs → Xilinx ISE → Project Navigator. З Project Navigator виберіть File → New Project. На рисунку 1.2 показані 5 діалогових вікон (ДВ) для відкриття нового проекту (НП), рисунок 1.2, а – (1 з 5), рисунок 1.2, б – (2 з 5), рисунок 1.2, в – (3 з 5), рисунок 1.2, г – (4 з 5), останнє вікно – (5 з 5).

Можливо перемістити проект в іншу папку. Не використовуйте в іменах файлів і папок символ пропуску. Якщо все влаштовує, натисніть кнопку «Далі». Наступне ДВ (рисунок 1.2, б) дає змогу встановлювати додаткові опції проекту. Перша група установок, поданих на цьому вікні, стосується FPGA, наявного на платі Spartan-3E Starter Kit. Друга група установок стосується мови опису, засобів синтезу та установок моделювання. Виберіть опції, як показано на рисунку 1.2, б, і натисніть кнопку «ОК». Наступне ДВ (рисунок 1.2, в) дає можливість створити новий вихідний файл у рамках нового проекту. Не створюйте новий файл зараз, просто натисніть кнопку «Далі». Наступне ДВ, наведене на рисунку 1.2, г, дає можливість додавати існуючі файли, як частину нового проекту. Не додавайте файл зараз, просто натисніть кнопку «Далі».

Останнє ДВ при відкритті нового проекту є підсумковим при створенні проекту, що базується на ваших установках. Перевірте підсумкову інформацію, щоб переконатися, що все правильно. Якщо щось не так, поверніться, натиснувши «Back», і

виправте помилки. Інакше натисніть на «Finish», щоб завершити процес.

На цьому етапі проект уже створений, але він не містить жодного вихідного файлу. Інакше натисніть на «Finish», щоб завершити процес. Після цього створіть новий вихідний файл для двовходового елемента XOR або виберіть Project → New Source з головного меню, або використовуйте еквівалентний процес у вікні Processes. Відкриється перше вікно з ДВ для нового вихідного файлу New Source (рисунок 1.3, а), потім друге (рисунок 1.3, б) та третє.

Виберіть VHDL Module, щоб указати, що ви створюєте модуль проекту, описаного на VHDL. Потім введіть ім'я проекту, яке зазначено на рисунку 1.3, а. Ви не повинні змінювати локалізацію цього файлу, який має бути всередині директорії, створеної раніше. Натисніть «Далі».

Наступне ДВ дає змогу вам визначити (призначити) порти модуля. Це може бути зроблено також у текстовому редакторі, коли створюється модуль, щоб пропустити цей крок. Просто підтвердіть установки відповідно до тих, що показані на рисунку 1.3, б, і натисніть «Далі».

Заключне в цьому ланцюжку ДВ забезпечує підбиття підсумків за вказаними установкам. Перевірте інформацію, щоб переконатися в її правильності. Для коригування установок натисніть «Назад», інакше натисніть «Готово», щоб завершити процес. Новий створений файл повинен бути автоматично відкритий у текстовому редакторі, але якщо цього не сталося, встановіть у вікні Sources у рядку Sources for Synthesis / Implementation і натисніть двічі на іконці вихідного файлу у вікні Sources.

У відкритому в текстовому редакторі файлі вже є базова файлова структура, але нам доведеться замінити все, що автоматично згенеровано.

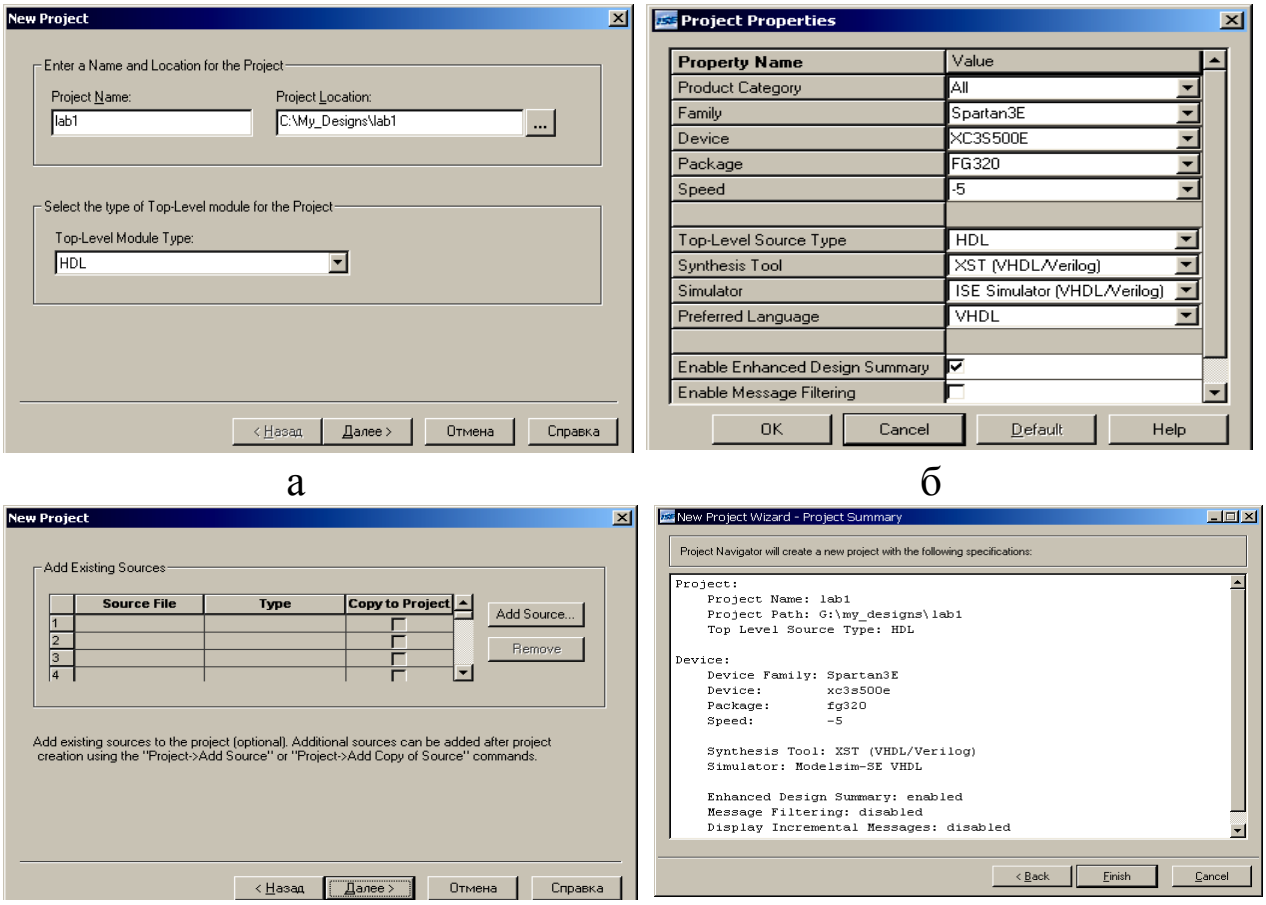


Рисунок 1.2 – ДВ для нового проекту

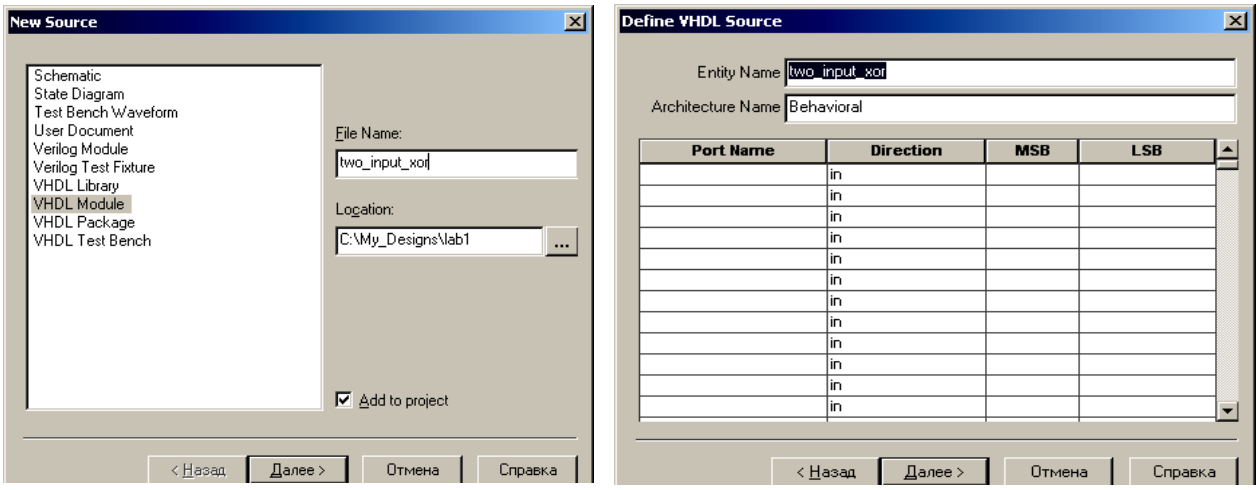


Рисунок 1.3 – ДВ для нового вихідного файлу

Для прикладу візьміть двовходовий елемент I-HI. Скопіюйте лістинг (VHDL-опис елемента XOR), розташований нижче (виділений синім кольором) і вставте його у відкритий очищений

файл. В отриманому таким чином файлі синім кольором будуть виділені ключові слова, рожевим – тип даних, зеленим – коментарі, чорним – значення. Таке колірне виділення покращує читабельність тексту і розпізнавання друкованих помилок.

```
-- Файл: two_input_xor.vhd
-- Підключення бібліотеки IEEE.
library IEEE;
-- Підключення пакетів бібліотеки ieee.
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
-- Опис інтерфейсу пристрою (входи a,b, вихід
c)
-- two_input_xor – ім’я пристрою
entity two_input_xor is
    port (a, b: in std_logic;
        c: out std_logic);
end two_input_xor;
-- Опис архітектури пристрою
-- Behavioral – ім’ я архітектури пристрою
architecture Behavioral of two_input_xor is
   begin    c <= a xor b;
end Behavioral;
```

Тепер вікно буде мати вигляд як на рисунку 1.4. Збережіть проект. Існують опції головного меню, що дають змогу зберігати окремі файли і проект цілком.

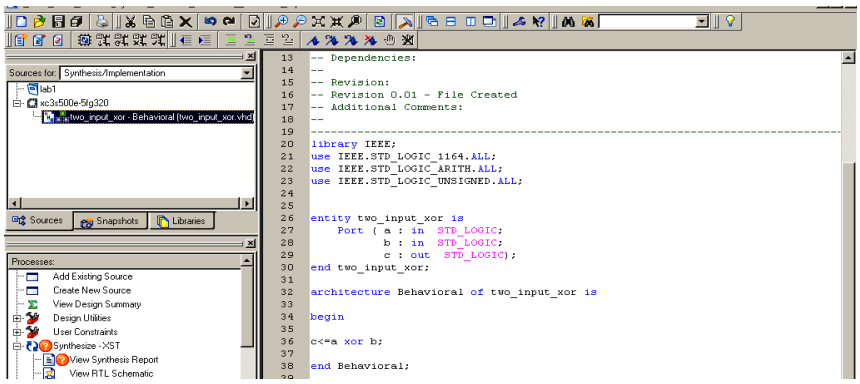


Рисунок 1.4 – Завершений проект

1.1.3 Синтез готового проекту

Тепер наступним кроком буде перехід від функціонально коректного опису проекту мовою VHDL до списку з'єднань (netlist), тобто до машинно-орієнтованої схематики. Буде використовуватися засіб, що іменується XST, який інтегрований у Project Navigator і націлений тільки на пристрої фірми Xilinx.

Щоб задати процес синтезу, встановіть у вікні Sources Synthesis / Implementation. Якщо ви виберете вихідний файл two_input_xor, ви повинні побачити Synthesize-XST, як процес, наявний у вікні Processes. На замітку – опції синтезу можна змінити до власне синтезу натисканням правої кнопки миші на Synthesize-XST і вибором Properties. Однак у цей момент не потрібно нічого змінювати, залиште опції, встановлені за замовчуванням. Подвійне натискання на Synthesize-XST запустить синтез. Коли синтез завершиться, ви повинні побачити зелену контрольну мітку як на рисунку 1.5.

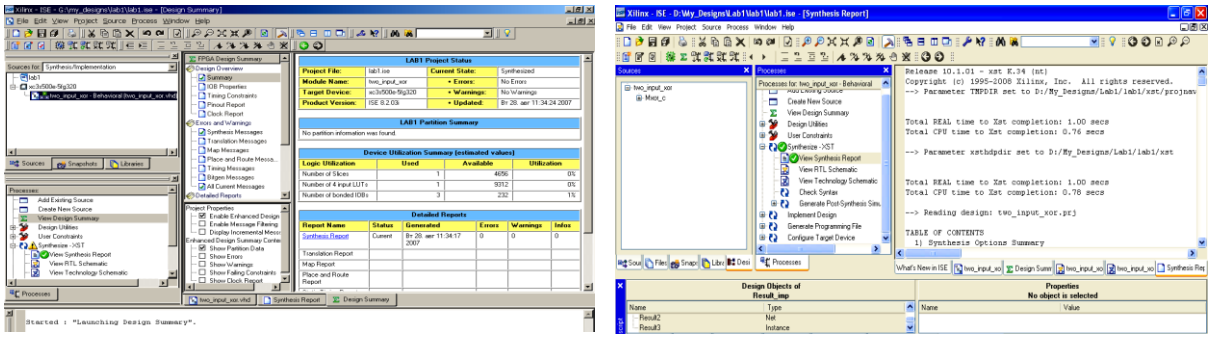


Рисунок 1.5 – Завершення синтезу проекту

Натиснувши на «+» біля Synthesize-XST, відкрити меню синтезу, у якому View Synthesize Report, View RTL Schematic, View technology Schematic. Натиснувши на рядок View Design Summary у вікні Processes, ви побачите вікно як на рисунку 1.5. У меню View Synthesize Report можна переглянути звіт. Меню View RTL Schematic показує схему до логічних елементів і меню View technology Schematic показує схему тільки до LUTтов (рисунки 1.6, 1.7).

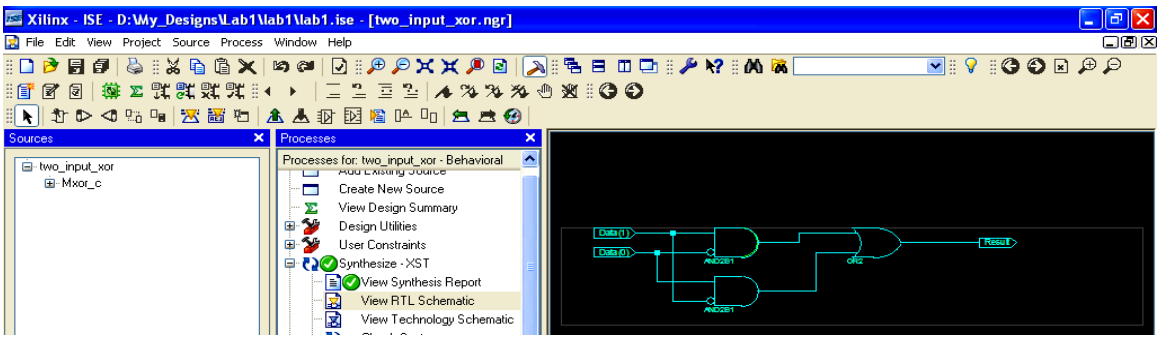


Рисунок 1.6 – Схема тільки до LUTтов

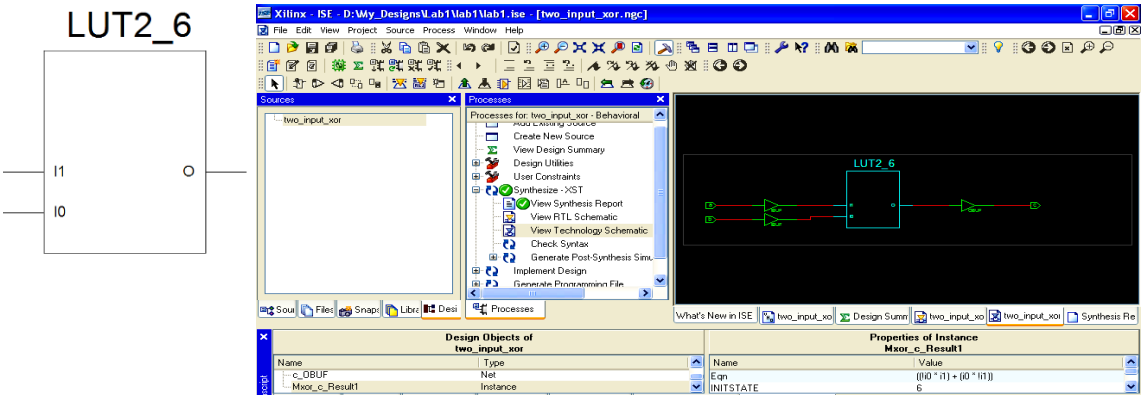


Рисунок 1.7 – Вікно View technology shcanbic

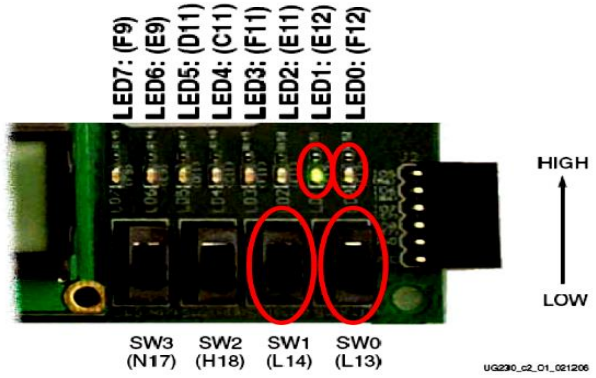
Ви не повинні зараз бачити жодну помилку у вікні Console, однак, ви завжди повинні переглядати log file, який можна побачити, розкриваючи процес Synthesize-XST натисканням на «+». Виберіть View Synthesis Report. Якщо вам незрозуміле певне повідомлення, не ігноруйте його. Замість цього зверніться на сайт підтримки Xilinx або до інструктора.

Зі звіту можна з'ясувати якого типу ресурси і як багато використовували засоби синтезу. Ви також можете дізнатися про інші проблеми, що виникли на цьому шляху. Наприклад, якщо ви виявили, що опис проекту реалізовано на тригерах, а не на look-up table, вам краще повернутися назад і з'ясувати, де допущена помилка. Ось чому ви повинні розбиратися в проектуванні апаратних засобів, які ви намагаєтеся створити, коли пишете мовний опис проекту.

1.1.4 Імплементація створеного проекту

Імплементація проекту – це послідовність подій, які переводять ваш netlist у файл програмування для пристрою FPGA. Проект, який ви зараз синтезували, має ряд портів на верхньому рівні. Засобам імплементації необхідно знати, яким чином підводити порти проекту під фізичні виводи мікросхеми FPGA, яка приєднується до різних ресурсів плати Spartan-3E Starter Kit. Якщо ви не зробите явних призначень, засоби будуть підключені до виводів випадково. Однак це погана ідея, оскільки випадкове призначення може бути неправильним.

Наш проект має два вхідних порти і один вихідний порт. Нам потрібно буде два перемикачі SW0 і SW1, що приєднуються до входів. Крім того, необхідно буде вихід підключити до LED (світлодіодів) таким чином, щоб ми могли його спостерігати – для цього призначається індикатор LD0 (рисунок 1.8).



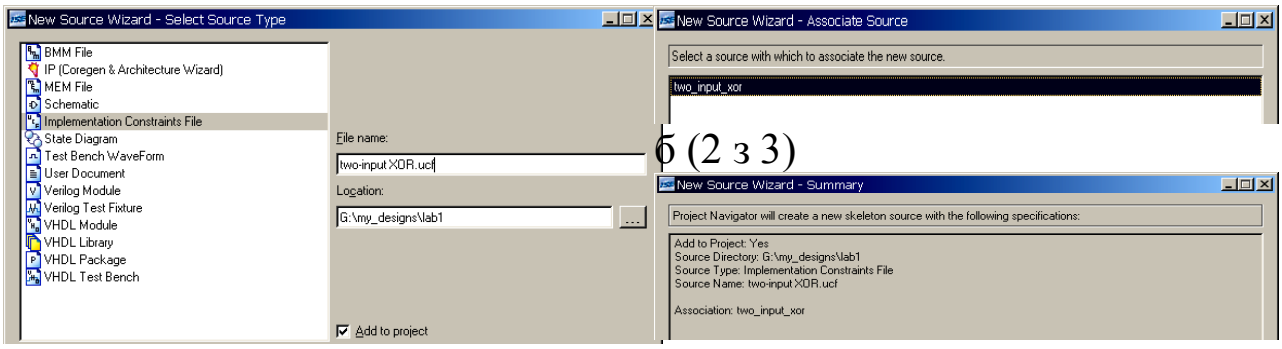
- SW0 → FPGA Pin L13
- SW1 → FPGA Pin L14
- LD0 → FPGA Pin F12

Рисунок 1.8 – Використовувані перемикачі та індикатори

Якщо подивитися на плату, то на ній видно, що багато ресурсів мають забезпечені коментарі з текстовою індикацією, яка пов'язана з виходами FPGA. Ця інформація є в Spartan-3E Starter Kit User Guide у вигляді таблиць і схематичних форм. Спробуйте визначити, який вивід FPGA використовується для SW0, SW1 і LD0, і порівняйте ваші результати з наведеною нижче інформацією. Вам необхідно бути здатними зробити це самостійно у своїх лабораторних роботах. Зараз маємо достатньо інформації для створення user constraint file (UCF) (файл призначених для користувача обмежень). Цей файл містить у собі обмеження, які не були визначені у VHDL-описі, наприклад,

розташування виводів, обмеження на характеристики проекту. Швидше це зручніше робити в UCF, ніж у VHDL- описі. Наприклад, якщо ви допускаєте помилку в призначенні виводів, не потрібно повертатися і заново синтезувати проект.

Ви можете додати UCF у проект, використовуючи той же процес, що й для додавання файлу в проект. Створіть новий вихідний файл, виберіть Project → New Source з головного меню або використовуйте еквівалентний процес у вікні Processes. З'явиться перше з ДВ New Source (рисунок 1.9, а–в).



а (1 з 3) б (2 з 3) в (3 з 3)

Рисунок 1.9 – ДВ нового джерела

Виберіть Implementation Constraints File, щоб указати, що ви створюєте файл обмежень. Потім введіть ім'я файлу, як показано на рисунку 1.9. Не потрібно змінювати раніше визначену локалізацію файлу, який повинен бути всередині проектної директорії, створеної раніше. Натисніть «Next». Друге ДВ показано на рисунку 1.9, б. Ідентифікуйте проектний модуль, з яким повинен бути пов'язаний файл обмежень. Виберіть двовходовий XOR, як показано на рисунку, і натисніть «Next». Останнє ДВ на рисунку 1.9, в підсумовує установки вихідного файлу. Перегляньте його, якщо є необхідність, поверніться і виправте помилки, натиснувши «Back», інакше натисніть «Finish», щоб завершити процес. Потрібно зазначити, що цей файл не відкривається автоматично ні в якому редакторі.

Якщо ви вибрали файл обмежень у вікні Sources і розкрили User Constraints натиснувши на «+», ви побачите ряд варіантів для редагування файлу користувача обмежень, зокрема і текстовий редактор. Відкрийте текстовий редактор натиснувши

на Edit Constraints (Text). Вставте в поле редактора нижченаведені 3 рядки, які здійснюють призначення сигналів виводів мікросхеми. NET "a" LOC="L13"; NET "b" LOC = "L14"; NET "c" LOC = "F12".

Існує графічний спосіб призначення сигналів виводів мікросхеми. У XILINX ISE виберіть у вікні Sources файл Two_input_xor.vhd, потім відкрийте у вікні Processes User Constraints і подвійним натисканням на Foorplan IO- Pre-Synthesis запустіть редактор призначення виводів і області обмежень PACE. Просто подивіться, нічого не змінюючи. Підвікно PACE, показане на рисунку 1.10, було пересунуто з його початкових позицій для того, щоб отримати поліпшену копію екрана. Натисніть на I / O Pins у вікні Design Browser. Вікно Design Object List покаже вам імена портів верхнього рівня і напрямок їх сигналів. У цьому вікні заповнення в LOC областях ґрунтується на заздалегідь визначених установленнях виводів FPGA. Заповнення в інших областях для порівняння показано на рисунку 1.10. Ці опції встановлюють електричні характеристики I / O виводів.

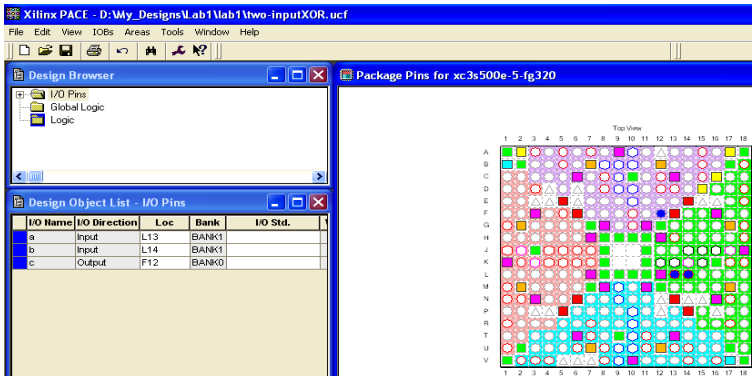


Рисунок 1.10 – Призначення обмеження розташування виводів з PACE

Після установлення кожного входу, ви можете помітити, що відповідний набір виводів, показаний у Package Pins window, буде маркований кольором для порівняння у вікні Design Object List. Ця діаграма подає ресурси частини FPGA, що належить до ваших обмежень, – у цьому випадку вхід і вихід буферів і контактні площадки входів / виходів (I / O). Натисніть на вкладку Package View у правому вікні Device Architecture і побачите Top View.

Після цього збережіть вашу роботу і вийдіть з програми PACE. Якщо ця програма була запущена вперше, вона може дати вам підказку визначити I / O Bus Delimiter. Виберіть "XST Default" і "Do not show this dialog again ...".

Тепер у вашому проекті є файл установлення відповідності (файл обмежень), отже, можна приступати до імплементації проекту.

Виберіть двовходовий елемент xor у вікні Sources. Потім подвійним натисканням виберіть Implement Design процес у вікні Processes. Project Navigator буде імплементувати проект і видавати інформацію про хід імплементації у вікні Console. Інформація на замітку – опції імплементації можна замінити безпосередньо перед імплементацією натисканням правої кнопки миші на Implement Design, вибравши Properties. Після закінчення імплементації ви побачите інформацію, подібну до поданої на рисунку 1.11.

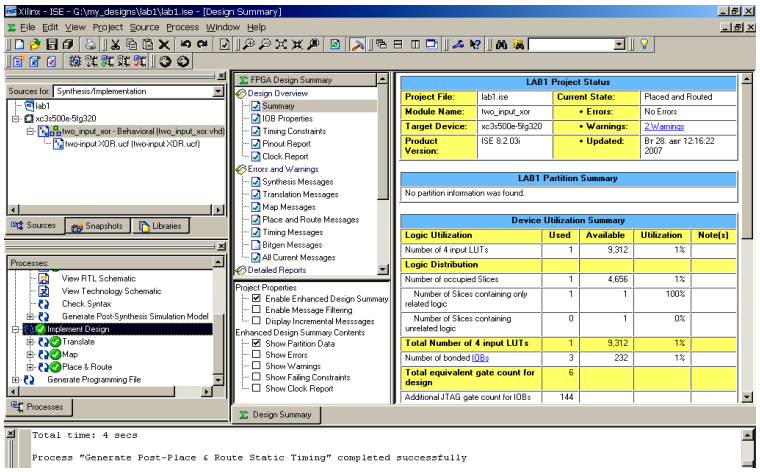


Рисунок 1.11 – Завершення імплементації проекту

Ви не повинні побачити ніяких помилок у вікні Console. Однак ви повинні завжди переглядати три log файлу, що формуються в процесі ranslateT, Map, and Place and Route. Якщо ви не зрозуміли якесь повідомлення, не ігноруйте його, зверніться в службу підтримки на сайті Xilinx. На цьому етапі (після імплементації) ви повинні переконатися в наявності зелених маркерів біля Implement Design process.

1.1.5 Програмування безпосередньо FPGA

Програмування FPGA безпосередньо – зручний спосіб випробувати проект. Цей метод корисний для прототипування, коли проект не є остаточним, щоб упевнитися в правильності проекту. Однак складні проекти не завжди працюють «з першої спроби». Одна з чудових переваг FPGAs перед ASICs схемами полягає в тому, що розплата за помилку при першій спробі мінімальна.

Тепер потрібно створити файл для програмування FPGA. Виберіть двовходовий елемент xor у вікні Sources. Потім подвійним натисканням виберіть Generate Programming File процес у вікні Processes. Project Navigator буде генерувати файл, який програмує FPGA, і видавати інформацію про хід імплементації у вікні Console. Перед тим, як рухатися далі, візьміть плату Spartan-3E Starter Kit, пристрій живлення і USB кабель. Підключіть USB кабель до USB порту вашого комп'ютера (рисунок 1.12).

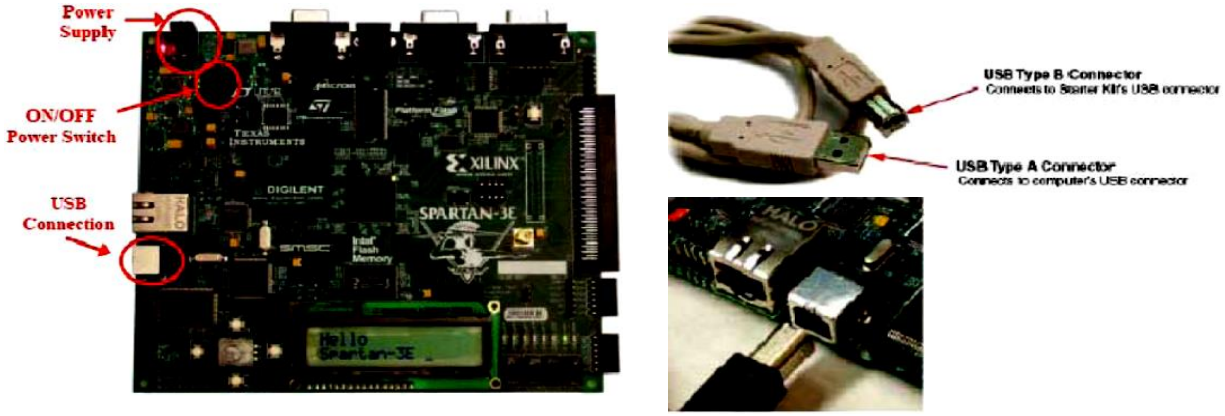


Рисунок 1.12 –Підключення плати Spartan-3E Starter Kit до USB порту

Підключіть пристрій живлення до розетки.

Забезпечте живлення перемикачами JP6 і JP7.

Встановіть перемикач режимів JP30 в одне з трьох положень master serial; батарея 0 В JP9; робоча пам'ять JP11.

Далі підключіть кабель живлення в гніздо живлення та увімкніть блок живлення. Майте на увазі, якщо файл для програмування FPGA був попередньо розміщений на плату, він

буде автоматично завантажуватися і може привести до активності плати як flashing LED, це може бути небезпечно. На завершення приєднайте USB кабель до його конектора. Зауваження – якщо ви вперше скористалися платою, Windows New Hardware Wizard буде запускатися кілька разів, поки не встановляться драйвери для плати.

Щоб завантажити ваш bitstream в FPGA, розкрийте Configure Target Device натисканням на «+» і потім подвійним натисканням завантажте Manage Configuration Project (iMPACT). Це запустить програму iMPACT в іншому вікні. Частина завдання, що залишилася, виконається в iMPACT. Після запуску iMPACT ви побачите серію ДВ, як показано на рисунку 1.13.

Плата має функцію програмування JTAG, яка називається Boundary Scan. Виберіть цю опцію і натисніть «Finish». Далі отримаєте послідовність трьох файлів замовника. У першому файлі замовника, показаному на рисунок 1.13, виберіть файл two_input_xor.bit, який був сформований у процесі імплементації. Він є файлом для програмування FPGA. Після вибору файлу для FPGA ви можете отримати попередження, що Startup Clock був змінений. Ігноруйте це ДВ. Метою наступних двох замовників файлів є ідентифікація файлів для програмування Xilinx пристроїв на платі. Ми поки ще нічого не програмували, тому оберіть Bypass (обхід) для них обох.

Нарешті, ви бачите вікно, зображене на рисунку 1.14. IMPACT готова для програмування FPGA. Виберіть іконку FPGA у вікні і потім використовуйте праву кнопку миші, щоб активізувати меню, як показано, і виберіть Program опцію. Зауважте, що Assign New Configuration File опція може бути використана, щоб змінити установлення вашого файлу конфігурації, якщо це необхідно зробити.

Далі з'явиться вікно з опціями програмування. Більшість з цих опцій є побічними для програмування FPGA і нас у цей момент не стосуються. Скасуйте Verify опцію, якщо вона обрана, і натисніть «Ok», щоб почати програмування. З'явиться індикатор виконання команд (рисунок 1.15).

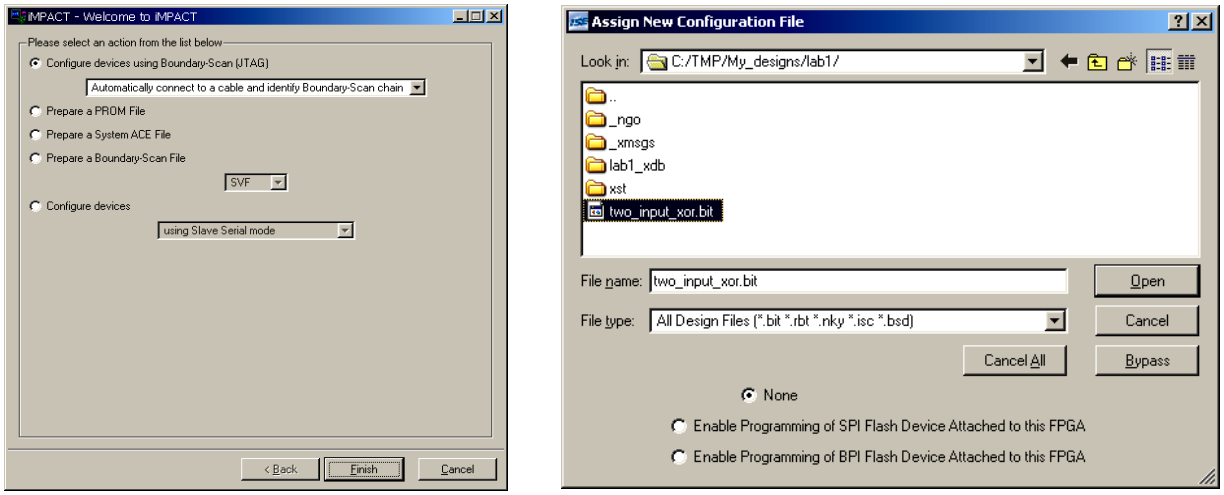


Рисунок 1.13 – Вибір режиму конфігурації та файлу для програмування FPGA

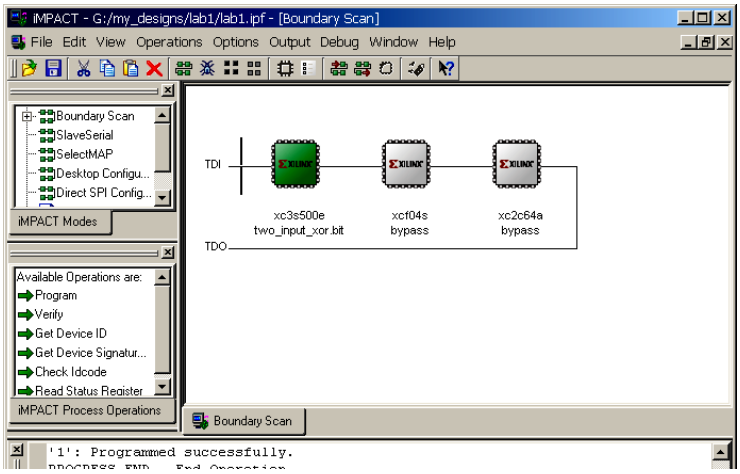


Рисунок 1.14 – Вибір пристрою для програмування

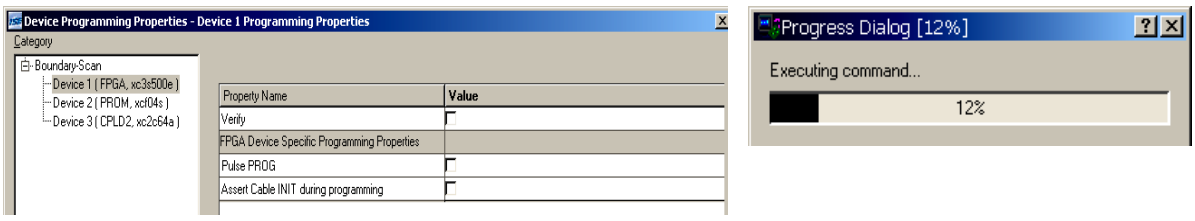


Рисунок 1.15 – Опції програмування та індикатор виконання команд

Коли програмування завершиться, система дасть знати, успішно воно пройшло, чи ні. Якщо виникли проблеми, перевірте підключення кабелів, положення перемикачів і спробуйте

програмування знову. Якщо проблеми не зникли, шукайте помилки.Тепер ми можемо протестувати наш проект на платі.

Встановлюйте перемикачі SW0 і SW1 в 4 можливих положення – 00, 01, 10, 11 – і спостерігайте реакцію схеми на ці вхідні значення на індикаторі LD0-LED0 (F12) (рисунок 1.16). Зробивши це один раз, ви готові до реалізації проекту на Xilinx Platform Flash пристрої. Цей останній крок дає можливість завантажувати ваш проект при вмиканні живлення без використання USB кабелю.

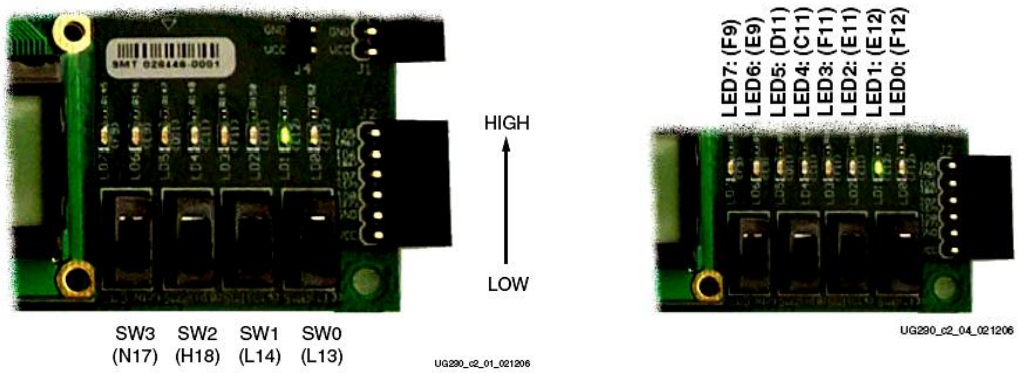


Рисунок 1.16 – Перемикачі вхідних впливів та індикатори вихідних значень

1.1.6 Програмування Xilinx Platform Flash

Повернемося до iMPACT. Виберіть опцію PROM File Formatter у вікні Flows, двічі натисніть на ньому лівою кнопкою миші. Це ініціює серію ДВ для збору додаткової інформації. Спочатку з'явиться вікно як на рисунку 1.17. Установіть опції, як показано на рисунку 1.17, і перейдіть до наступного діалогового вікна, яке зображено на рисунку 1.18. У ДВ на рисунку 1.18 ви повинні вибрати xcf04s, 4-megabit Platform Flash і натиснути на «Add». Далі перейдіть до наступного ДВ (рисунок 1.19), яке показує підсумок того, що ви вибирали. Натисніть «Back» для коригування помилок, якщо вони є. Інакше натисніть на «Finish», щоб продовжити, iMPACT буде пропонувати вам додавати файли пристрою до потоку даних пристрою. Коли з'явиться ДВ замовника файлів, виберіть two_input_xor.bit файл.

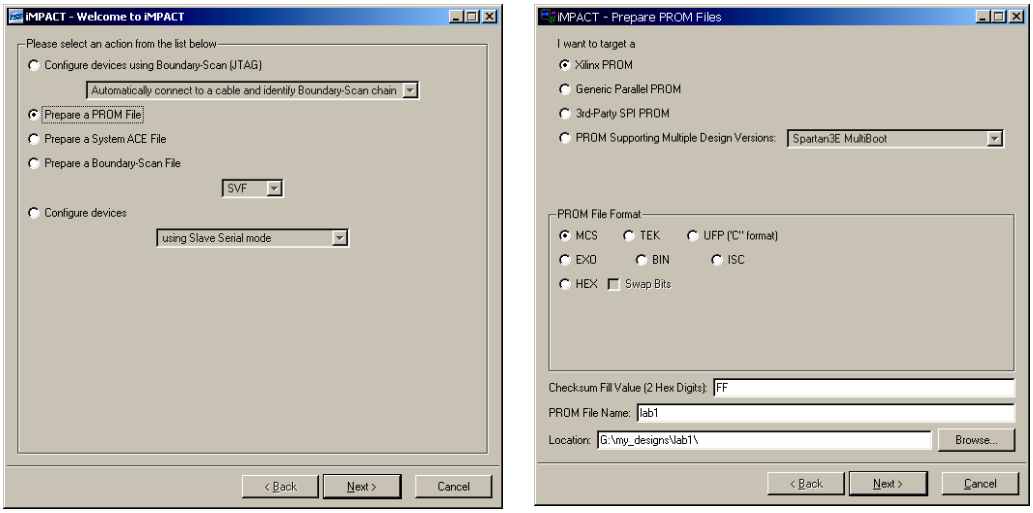


Рисунок 1.17 – Підготовка PROM файлів

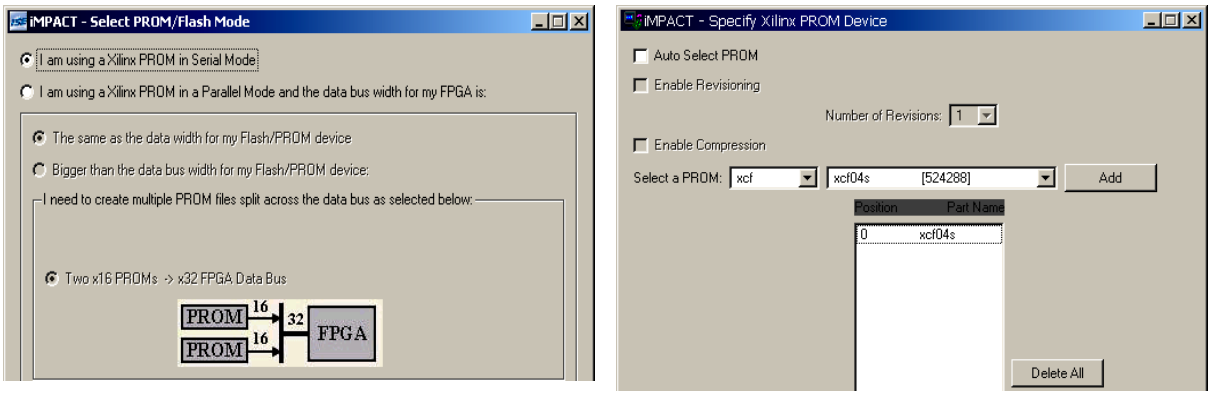


Рисунок 1.18 – Призначення Xilinx PROM пристрої

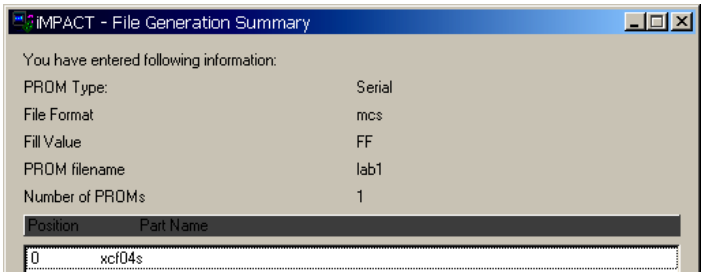
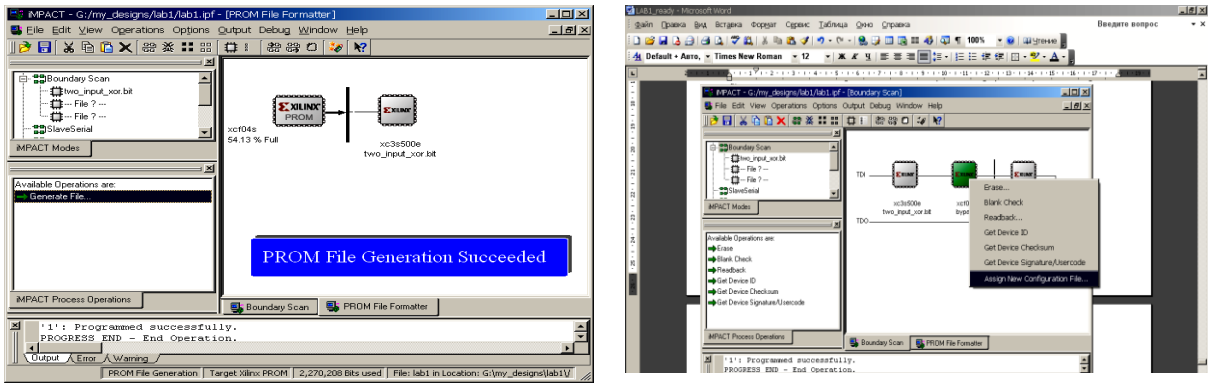


Рисунок 1.19 – Підсумки введення інформації

Після додавання two_input_xor.bit файлу iMPACT запропонує додати інший пристрій – натисніть на «No». Потім ви побачите інше ДВ, яке покаже, що введення файлу було завершено. Ігноруйте це ДВ.

Виберіть опцію Generate File в iMPACT Process Operation і двічі натисніть на ньому. iMPACT буде генерувати файл

(рисунок 1.20, а). Натисніть двічі на опції Boundary Scan у вікні iMPACT modes. Це поверне вас до режиму програмування пристрою таким чином, що ви зможете запрограмувати пристрій Platform Flash. Виберіть xcf04s, як показано на рисунку 1.20, б, і натисніть на ньому правою кнопкою – таким чином, ви можете призначити ваш заново створений файл конфігурації. Якщо пристрої не відображаються як на рисунку 1.20, б, правою кнопкою миші на порожньому полі виберіть у меню Initialize Chain, потім ви можете призначити ваш заново створений файл конфігурації.



а б

Рисунок 1.20 – Призначення нового файлу конфігурації

Використовуючи замовник файлів, що з’являється, виберіть файл lab1.mcs. Коли ви завершите цей крок, вікно iMPACT буде мати вигляд як на рисунку 1.21.

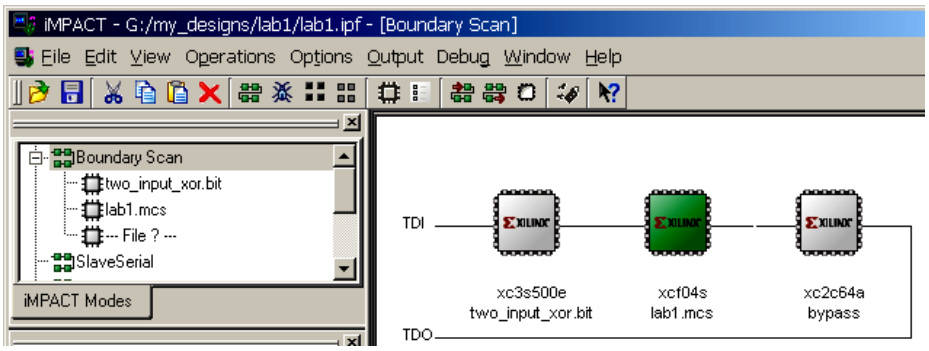


Рисунок 1.21 – Готовність до програмування

Потім виберіть іконку xcf04s, натисніть на ній правою кнопкою і виберіть опцію Program. З'явиться ДВ з опціями

програмування. Більшість цих опцій допоміжні для програмування Platform Flash, і в цей момент неістотні. Перевірте опції, щоб вони збігалися з рисунком 1.22, і натисніть «Ok», щоб почати програмування. З'явиться індикатор процесу програмування.

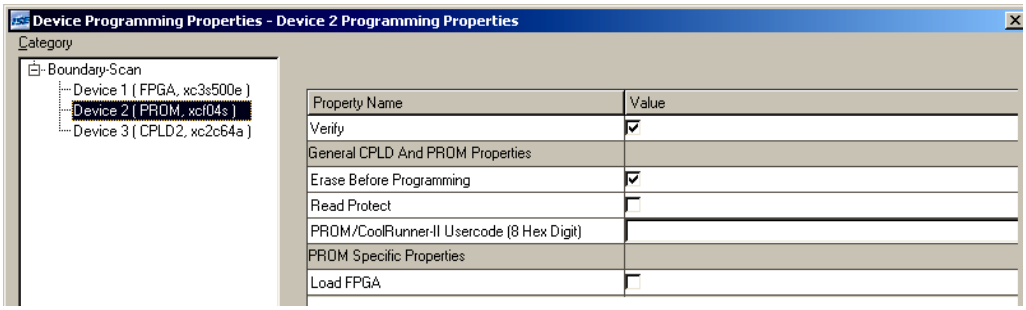


Рисунок 1.22 – Опції для програмування

Коли програмування завершиться, система дасть знати, успішно воно пройшло, чи ні. Якщо виникли проблеми, перевірте підключення кабелів, положення перемикачів і спробуйте програмування знову. Якщо проблеми не зникли, шукайте помилки.

У процесі програмування iMPACT налаштовує FPGA на перезавантаження в нього інформації, накопиченої в пристрої Platform Flash.

Ви можете відазу ж протестувати свій проект. Вийдіть з iMPACT, вимкніть живлення і відключіть USB кабель від плати. Зачекайте 3 с і знову ввімкніть живлення. FPGA має завантажити ваш проект автоматично з пристрою Platform Flash. Перевірте, чи це так.

1.2 Опис лабораторної установки

У лабораторній роботі використовується пакет Xilinx ISE Project Navigator і плата Xilinx Spartan-3E Starter Kit.

1.3 Порядок виконання лабораторної роботи

Рекомендується записувати проекти на особисту USB Flash пам'ять. Ніколи не залишайте і не накопичуйте свої проекти на комп'ютерах у лабораторії.

1.3.1 Виконати синтез логічної схеми XOR вручну.

1.3.2 Ознайомитися із середовищем Project Navigator пакета Xilinx ISE.

1.3.3 Розібратися в VHDL-описі логічної схеми XOR.

1.3.4 Створити проект у середовищі Project Navigator, додати в проект VHDL- опис логічної схеми xor.

1.3.5 Виконати синтез проекту.

1.3.6 Виконати імплементацію проекту.

1.3.7 Виконати безпосереднє програмування FPGA на платі Xilinx Spartan-3E Starter Kit і перевірити правильність роботи проекту.

1.3.8 Виконати програмування FPGA через пристрій Platform Flash на платі Xilinx Spartan-3E Starter Kit і перевірити правильність роботи проекту.

Завдання 1. Кожен студент індивідуально виконує запропонований варіант завдання. Виконати лабораторну роботу відповідно до нижченаведених пунктів.

1 Вибрати варіант завдання з таблиці 1.1 за останніми двома цифрами номера залікової книжки і виконати завдання 1. Згідно з варіантом завдання записати структурну VHDL-модель пристрою (рисунок 1.23).

Таблиця 1.1 – Варіанти завдань до завдання 1

Варіант	Завдання		Варіант	Рисунок
1	Рисунок 1.23,а		8	Рисунок 1.23, и
2	Рисунок 1.23,б		9	Рисунок 1.23, а
3	Рисунок 1.23, в		10	Рисунок 1.23,б
4	Рисунок 1.23,г		11	Рисунок 1.1,в
5	Рисунок 1.23, д		12	Рисунок 1.23,г
6	Рисунок 1.23, е		13	Рисунок 1.23,д
7	Рисунок 1.23, ж		14	Рисунок 1.23,е

В описі використовувати стандартні примітиви, як це було зроблено раніше для двовходового елемента АБО-НІ.

2 Виконати синтез заданої логічної схеми вручну, (рисунок 1.23).

3 Створити проект у середовищі Project Navigator, додати в проект VHDL-опис своєї логічної схеми.

4 Виконати синтез проекту.

5 Виконати імплементацію проекту.

6 Виконати безпосереднє програмування FPGA на платі Xilinx Spartan-3E Starter Kit і перевірити правильність роботи проекту.

7 Виконати програмування FPGA через пристрій Platform Flash на платі Xilinx Spartan-3E Starter Kit і перевірити правильність роботи проекту.

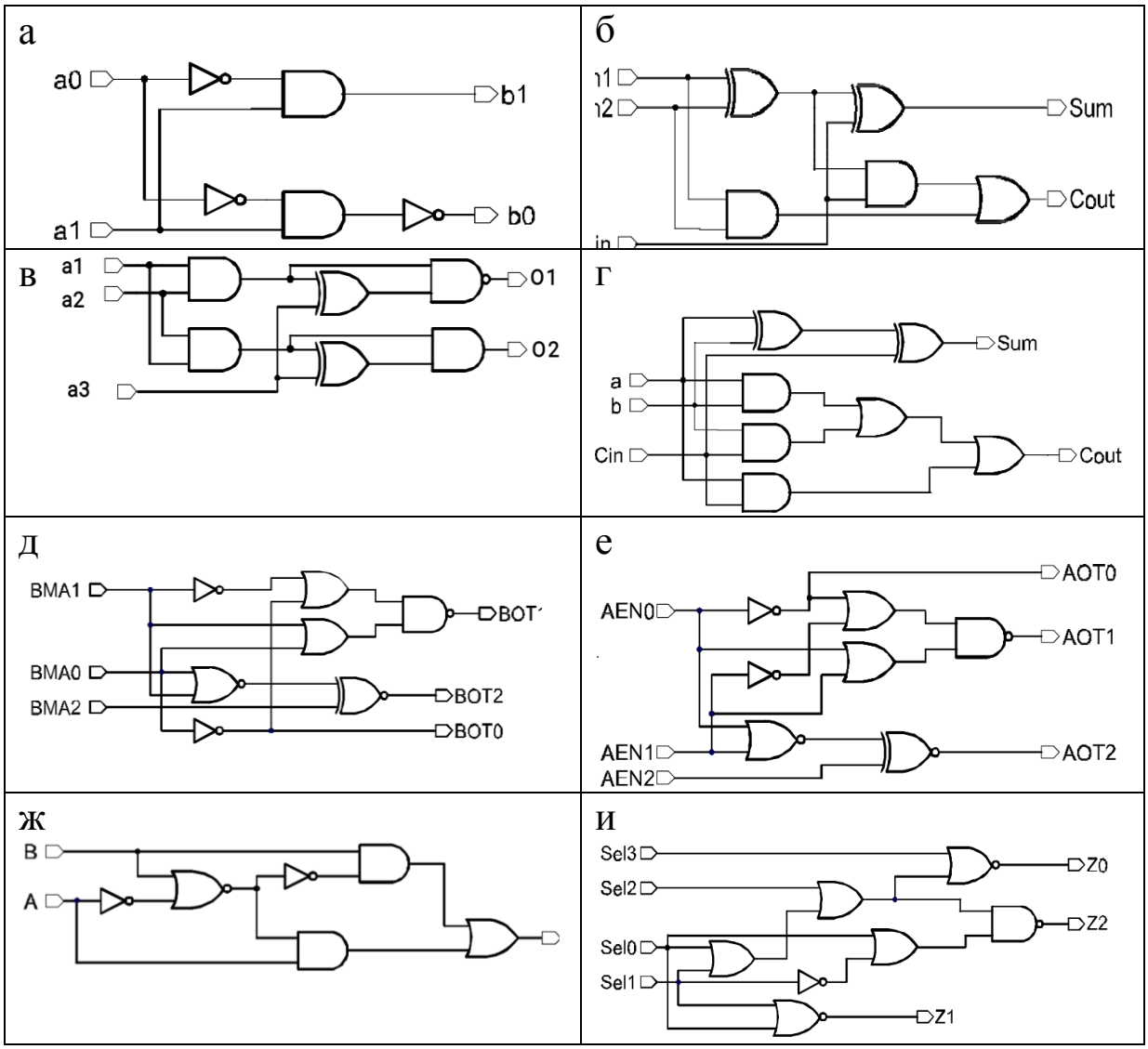


Рисунок 1.23 – Варіанти завдань

Завдання 2. Вибрати варіант завдання 2 з таблиці 1.2 за останніми двома цифрами номера залікової книжки і виконати завдання 2. Відповідно до варіанта завдання записати структурну

VHDL-модель пристрою. В описі використовувати стандартні примітиви, як це було зроблено для двовходового елемента АБО-НІ.

Таблиця 1.2 – Варіанти завдань до завдання 2

Варіант	Значення вихідних змінних					
	y1		y2		y3	
	=1	=x	=1	=x	=1	=x
1	0,3,8,14	1,9,11,12	0,1,8,12	10,13,14,15	2,3,8,10	0,4,9,14
2	1,4,5,14	0,8,13,15	1,3,4,5	0,2,12,13	3,4,7,11	2,5,8,9
3	0,4,5,7	2,6,12,15	4,5,10,13	9,11,14,15	2,10,14,15	0,5,6,13
4	2,5,7,8	3,4,6,9	1,10,12,13	3,7,9,15	4,9,12,13	6,7,11,15
5	1,8,10,15	3,6,12,13	1,5,8,15	7,9,10,11	0,1,2,3	4,9,10,11
6	0,2,6,7	1,3,5,10	3,4,5,11	12,13,14,15	3,4,5,13	2,9,12,14
7	9,12,13,15	1,7,11,14	6,7,12,13	8,9,11,14	10,13,14,15	0,4,8,12
8	1,8,9,14	3,10,11,13	2,3,4,7	0,1,6,12	6,7,12,14	3,4,5,15
9	1,5,11,15	2,8,10,12	12,13,14,15	1,3,7,11	1,3,4,12	5,8,9,10
10	2,6,9,12	4,7,8,10	2,4,7,13	3,6,8,9	1,11,12,15	4,8,13,14
11	4,6,13,14	5,7,8,10	1,2,4,13	0,3,9,11	6,10,12,15	7,11,13,14
12	3,4,6,8	9,10,11,12	4,5,6,12	0,1,7,8	2,6,7,14	0,4,8,12
13	0,1,3,4	2,5,7,8	1,6,7,9	2,3,5,10	3,6,7,13	4,10,11,12
14	3,9,12,13	1,6,11,14	1,3,5,10	4,13,14,15	6,8,9,14	3,5,12,13
15	7,9,10,14	2,6,8,11	0,1,3,12	8,9,13,15	4,5,13,14	0,2,7,8
16	2,4,5,11	7,9,10,13	10,11,12,14	2,4,6,8	3,8,9,12	0,2,4,13
17	0,2,7,9	4,10,11,14	4,5,13,14	2,7,10,12	0,2,6,11	1,3,4,13
18	4,5,13,14	1,6,8,12	1,2,4,8	0,6,12,13	2,7,9,12	0,4,5,13
19	0,2,6,7	1,3,5,12	3,4,5,11	2,7,13,15	1,5,9,13	0,2,3,4
20	0,5,7,13	1,2,3,4	2,6,12,13	4,7,8,15	3,12,13,14	2,4,5,15
21	4,7,8,9	10,11,13,15	0,7,12,13	1,6,9,15	2,3,7,8	1,9,10,11
22	0,1,3,15	2,5,12,13	1,3,6,9	0,10,12,13	0,1,4,10	2,8,9,11
23	0,1,6,13	2,3,12,14	5,10,11,13	3,4,7,8	0,3,12,13	1,2,14,15
24	1,4,10,11	8,9,12,15	4,9,12,14	0,5,10,15	4,8,10,11	1,12,14,15
25	0,6,9,10	8,12,13,15	1,2,3,14	0,7,8,15	0,7,9,10	4,6,8,11
26	1,8,10,14	0,11,12,13	4,11,12,13	2,5,6,7	0,10,12,13	5,6,9,11
27	5,11,12,13	1,4,6,8	1,7,12,13	0,2,4,14	2,9,12,13	1,3,14,15
28	2,6,10,14	5,7,8,13	2,5,6,14	8,10,11,15	4,5,6,13	0,2,7,12
29	6,10,11,13	2,3,14,15	5,11,12,13	3,4,6,7	1,2,9,11	8,10,12,15

1 Мінімізувати 3 булеві функції чотирьох змінних y1, y2 і y3, які задаються набором конституент одиниці і нуля (інші набори невизначені і можуть довизначатися одиницями і нулями) за допомогою карт Карно.

2 Отримати МДНФ і МКНФ.

3 За результатами синтезу побудувати комбінаційну схему.

4 Синтезувати функції.

5 Створити проект у середовищі Project Navigator, додати в проект VHDL-опис своєї логічної схеми.

6 Виконати синтез проекту.

7 Виконати імплементацію проекту.

8 Виконати безпосереднє програмування FPGA на платі Xilinx Spartan-3E Starter Kit і перевірити правильність роботи проекту.

9 Виконати програмування FPGA через пристрій Platform Flash на платі Xilinx Spartan-3E Starter Kit і перевірити правильність роботи проекту.

10 Оформити звіт про виконану лабораторну роботу, використовуючи шаблон.

1.5 Зміст звіту

Звіт оформляється кожним студентом індивідуально і повинен містити;

- титульний аркуш з номером і назвою роботи;
- мету роботи;
- результати синтезу схеми XOR вручну, а саме: таблицю істинності, карту Карно і рівняння функції у вигляді МДНФ;
- VHDL- опис логічної схеми XOR;
- лістинг UCF файлу;
- результати автоматизованого синтезу проекту у вигляді графічного відображення схеми XOR;
- висновки до роботи.

Контрольні запитання

1 Які типи мають сигнали у Verilog?

2 Як описуються вектори у Verilog?

3 Що таке стандартний примітив?

4 Як описуються структурні, поведінкові та DataFlow моделі у Verilog?

5 Як описуються затримки у Verilog?

6 Для чого використовується директива 'timescale?
7 Як працює блок initial?
8 Для чого використовується блок always?
9 Що таке синтез?
10 Навіщо використовується FSM Compiler?
11 У чому відмінність між схемами RTL і Technology?

ЛАБОРОТОРНА РОБОТА 2
Синтез цифрових схем у середовищі Project Navigator пакета Xilinx ISE

Мета роботи

Вивчення методів синтезу і мінімізації комбінаційних схем (КС); отримання практичних навичок у побудові синтезованих схем.

2.1 Вказівки до виконання лабораторної роботи

Комбінаційною схемою прийнято називати схему з n входами і m виходами, у якій сукупність вхідних сигналів у цей момент часу повністю визначається сукупністю вхідних сигналів у цей момент часу і не залежить від вхідних сигналів, поданих у попередні моменти часу. Отже, поведінка комбінаційної схеми може бути описана системою булевих функцій (БФ).

У зв'язку з тим, що одній БФ можуть відповідати різні суперпозиції функцій функціонально повної системи, то виникає завдання знаходження такої форми запису функції, при якій кожній функції буде відповідати одна і тільки одна формула стандартного типу і кожній формулі стандартного типу буде відповідати одна і тільки одна функція. Такі форми запису функцій називаються канонічними.

Такими канонічними формами є: досконала диз'юнктивна нормальна форма (ДДНФ) і досконала кон'юнктивна нормальна форма (ДКНФ).

Термін «диз'юнктивна» вказує на те, що зовнішньою функцією виразу є диз'юнкція, а внутрішньою – кон'юнкція, бо

для обчислення значень функцій потрібно визначити значення всіх кон'юнкцій, а після цього обчислити їх диз'юнкцію.

При проектуванні цифрових автоматів, зокрема КС, широко використовують методи мінімізації БФ, які дають змогу отримувати економічні схеми цифрових автоматів. Загальна задача мінімізації БФ може бути сформульована таким чином: знайти аналітичний вираз заданої булевої функції у формі, яка містить мінімально можливу кількість букв. У такій постановці задача досить добре досліджена в класі диз'юнктивно-кон'юнктивних нормальних форм.

Синтез комбінаційної схеми проводять, з огляду на таблицю істинності, яка описує роботу синтезованої схеми. Застосовуючи різні методи мінімізації, знаходять мінімальну диз'юнктивну нормальну форму (МДНФ) функції.

З великої кількості різних методів мінімізації найбільш прийнятні три методи:

- розрахунковий метод (метод безпосередніх перетворень);
- розрахунково-табличний метод (метод Квайна Мак-Класскі);
- табличний метод (метод карт Карно).

Розглянемо більш докладно останній метод мінімізації за допомогою карт (діаграм) Карно як один з найбільш зручних методів спрощення логічних функцій при невеликій кількості змінних, розроблений у 1953 р. Морісом Карно. Карта Карно є певною таблицею істинності для двох, трьох і чотирьох аргументів. Різні види карт Карно зображені на рисунках 2.1–2.4.

Змінні, які позначають комірки діаграми, розставляються таким чином, щоб набори, записані у двох сусідніх комірках, мали загальну частину, яка б складалася з усіх змінних, крім однієї, по якій вони можуть бути склеєні.

Для мінімізації БФ за допомогою методу Карно необхідно за таблицею істинності заповнити карту, після чого провести прямокутні або квадратні контури за такими правилами:

1) усередині контуру повинні бути комірки, які містять тільки одиниці (початкова форма ДДНФ);

2) кількість комірок у контурі повинна бути кратною степеню двійки;

3) при проведенні контурів крайні нижні і крайні верхні (крайні ліві і крайні праві) рядки вважаються сусідніми;

4) контур повинен містити якомога більшу кількість комірок;

5) усі одиниці, записані на карті, повинні бути охоплені контурами та описані.

Для того, щоб зробити мінімізацію в мінімальній кон'юнктивній нормальній формі (МКНФ), необхідно провести контури, які будуть охоплювати нулі; змінні елементарних кон'юнкцій МКНФ беруться з інверсіями. Так, наприклад, для карти Карно, наведеної на рисунку 2.1, вираз для МКНФ буде: Y = X1+ X2, до того ж він збігається з МДНФ.

Для рисунка 2.1: ДДНФ: $Y = \overline{X}_1 X_2 \vee X_1 \overline{X}_2 \vee X_1 X_2$, МДНФ:Y=X1+X2.

$X_1 \setminus X_2$	0	1
0	0	1
1	1	1

Рисунок 2.1 – Карта Карно двох змінних

$X_1 \setminus X_2 X_3$	00	01	11	10
0	1	0	0	1
1	1	1	0	1

Рисунок 2.2 – Карта Карно трьох змінних

Для рисунка 2.2:

ДДНФ: $Y = \overline{X}_1 \overline{X}_2 \overline{X}_3 \vee \overline{X}_1 X_2 \overline{X}_3 \vee X_1 \overline{X}_2 X_3 \vee X_1 \overline{X}_2 \overline{X}_3 \vee X_1 X_2 \overline{X}_3$,

МДНФ: $Y = \overline{X}_3 \vee X_1 \overline{X}_2$.

$X_1X_2 \setminus X_3X_4$	00	01	11	10
00	1	1	0	0
01	0	0	0	0
11	0	0	0	1
10	1	1	0	0

а

$X_1X_2 \setminus X_3X_4$	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	0	0	0	0
10	1	0	0	1

б

Рисунок 2.3 – Карта Карно чотирьох змінних

Для рисунка 2.3, а:

ДДНФ:

$Y = \overline{X}_1\overline{X}_2\overline{X}_3\overline{X}_4 \vee \overline{X}_1\overline{X}_2\overline{X}_3X_4 \vee X_1\overline{X}_2\overline{X}_3X_4 \vee$

$\vee X_1\overline{X}_2\overline{X}_3\overline{X}_4 \vee X_1X_2X_3\overline{X}_4,$

МНФ: $Y = \overline{X}_2\overline{X}_3 \vee X_1X_2X_3\overline{X}_4.$

Для рисунка 2.3, б

ДДНФ:

$Y = \overline{X}_1\overline{X}_2\overline{X}_3\overline{X}_4 \vee \overline{X}_1\overline{X}_2X_3\overline{X}_4 \vee$

$\vee \overline{X}_1X_2\overline{X}_3\overline{X}_4 \vee X_1\overline{X}_2X_3\overline{X}_4,$

МДНФ: $Y = \overline{X}_2\overline{X}_4.$

Контрольні приклади

1 Мінімізувати БФ чотирьох змінних, яка має конституенти одиниці на наборах: 0, 2, 4, 8, 12; отримати МДНФ і МКНФ. Карта Карно для БФ чотирьох змінних має вигляд як на рисунку 2.4.

$X_1X_2\backslash X_3X_4$	00	01	11	10
00	1	0	0	1
01	1	0	0	0
11	1	0	0	0
10	1	0	0	1

Рисунок 2.4 – Карта Карно функції $F(X_1,X_2, X_3,X_4)$

МДНФ F(x1,x2, x3,x4) = $\overline{X}_3\overline{X}_4 \vee \overline{X}_2\overline{X}_4$,

МКНФ F(x1,x2, x3,x4) = $\overline{X}_4 \cdot (\overline{X}_2 \vee \overline{X}_3)$.

На рисунку 2.5 наведена КС для функції $F(X_1,X_2, X_3,X_4)$ МДНФ у булевому базисі.

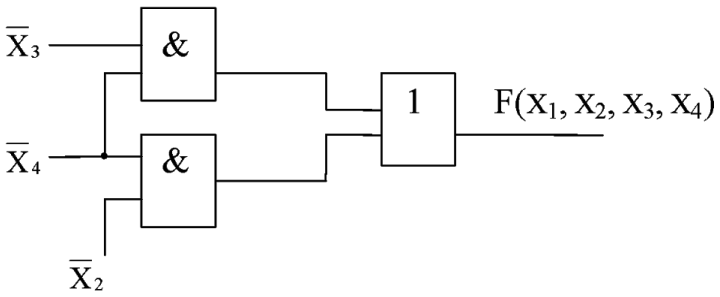


Рисунок 2.5 – Схема функції $F(X_1,X_2, X_3,X_4)$, побудована за МДНФ

Перехід від МДНФ форми до базису Шеффера простий: усі терми (логічні додатки) беруть у дужки, а всі знаки диз'юнкції і

кон'юнкції замінюють на операції Шеффера (/). Якщо в МДНФ є однобуквений терм, то в базисі Шеффера над ним ставиться заперечення.

Аналогічно виконується перехід від базису Буля до базису Пірса, якщо похідною формою є МКНФ.

При переході від МДНФ до базису Пірса схема буде триступеневою і значення всіх змінних порівняно з ДНФ змінюються на інверсні.

Нехай є функція:

$$F = X_1\overline{X}_2 \vee \overline{X}_1X_2 \vee X_1\overline{X}_3X_4 = \overline{\left(\overline{X_1\overline{X}_2}\right) \vee \left(\overline{\overline{X}_1X_2}\right) \vee \left(\overline{X_1\overline{X}_3X_4}\right)} =$$

$$= \overline{\left(\overline{\overline{X}_1 \vee X_2}\right) \vee \left(\overline{X_1 \vee \overline{X}_2}\right) \vee \left(\overline{\overline{X}_1 \vee X_3 \vee \overline{X}_4}\right)} = [(\overline{X}_1 \downarrow X_2) \downarrow (X_1 \downarrow \overline{X}_2) \downarrow (\overline{X}_1 \downarrow X_3 \downarrow \overline{X}_4)] \downarrow 0.$$

2.2 Опис лабораторної установки

У лабораторних роботах використовуються пакет Xilinx ISE Project Navigator і плата Xilinx Spartan-3E.

Методична література і додаткове програмне забезпечення розміщені на диску D кожного комп'ютера в лабораторії 3.427. Інструкція з використання програмних засобів міститься в тому ж каталозі.

2.3 Порядок виконання лабораторної роботи

2.3.1 Ознайомитися з рекомендованою літературою до лабораторної роботи. Перед виконанням роботи необхідно відповісти на контрольні запитання.

2.3.2 Кожен студент індивідуально виконує свій варіант завдання.

2.3.3 Виконати лабораторну роботу відповідно до нижченаведених пунктів.

1 Мінімізувати БФ чотирьох змінних, яка задається набором конституент одиниці (інші набори відповідають конституенті нуля) за допомогою карт Карно (таблиця 2.1).

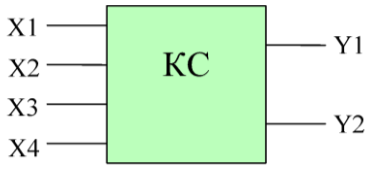
2 Отримати МДНФ і МКНФ.

3 За результатами синтезу побудувати КС у базисі Буля, Шеффера, Пірса.

Таблиця 2.1 – Варіанти завдань

Варі-ант	Конституенти одиниці функції Y(X1, X2, X3, X4)	Варі-ант	Конституенти одиниці функції Y(X1, X2, X3, X4)	Варі-ант	Конституенти одиниці функції Y(X1, X2, X3, X4)
1	1, 2, 3, 4, 5, 8, 9, 10, 11	11	1, 5, 6, 7, 8, 15	21	3, 2, 6, 8, 10, 12, 14
2	0, 1, 12, 3, 4, 15	12	3, 4, 5, 6, 7, 11, 12	22	5, 7, 13, 15
3	3, 6, 8, 9, 10, 11,12	13	0, 1, 2, 3, 4, 5, 9, 10	23	0, 1, 2, 3, 4, 6, 12, 14
4	15, 14, 13, 12, 11, 10, 9	14	0, 2, 4, 5, 6, 7, 8, 10,	24	4, 6, 12, 14, 8, 9, 11, 10
5	5, 6, 4, 13, 14, 15	15	0, 1, 5, 7, 8, 9, 10, 11	25	0, 1, 2, 3, 5, 7, 10, 11
6	1, 5, 7, 11, 12, 13, 14	16	0, 1, 5, 7, 8, 10, 12, 14	26	8, 9, 11, 12, 13, 15
7	3, 4, 5, 6, 7, 12, 13	17	2, 3, 6, 7, 9, 12 ,15	27	5, 6, 7, 13, 15
8	11, 12, 13, 14, 15	18	5, 6, 7, 8, 9, 10 ,13	28	0, 1, 5, 7, 8, 9
9	1, 2, 6, 7 10, 11, 15	19	6, 7, 10, 11, 14, 15	29	0, 2, 4, 6, 14, 10
10	0, 1, 2, 5, 6, 8, 9, 15	20	2, 3, 11, 12	30	4, 6, 10, 11, 12, 14

2.3.4 Написати VHDL-код спроектованого пристрою по МДНФ, використовуючи логічні оператори not, and, or. Як приклад можна скористатися кодом, наведеним нижче. Схема має 4 входи, 2 виходи, описана двома булевими рівняннями (рисунок 2.6).



$$Y1 = \overline{X3} \cdot \overline{X4} \cdot X2 \vee \overline{X3} \cdot X4$$
$$Y2 = \overline{X3} \cdot \overline{X4} \cdot X1 \vee X3 \cdot X4 \cdot X2$$

Рисунок 2.6 – Приклад КС

```
library ieee; - Підключення бібліотеки ieee.
use ieee.std_logic_1164.all; - Підключення
пакетів бібліотеки ieee.
```

```
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
- Опис інтерфейсу пристрою (входи X1, X2, X3, X4,
виходи Y1, Y2, Y3, Y4)
- KC - ім'я пристрою
entity KC is
port (
    X1, X2, X3, X4: in STD_LOGIC;
    Y1, Y2, Y3, Y4: out STD_LOGIC);
end KC;
architecture KC of KC is
begin
  Y1 <= (not X3 and not X4 and X2) or (not X3
and X4);
  Y2 <= (not X3 and not X4 and X1)) or (X3 and
X4 and X2);
  Y3 <= (not X3 and not X4 and X2) or (not X3
and X4);
  Y4 <= (not X3 and not X4 and X1)) or (X3 and
X4 and X2);
end KC;
```

2.3.5 Створити новий проект у середовищі Project Navigator, додати в проект VHDL-опис комбінаційної схеми перетворювача коду, розрахованого згідно з вашим варіантом (дивись підрозділ 1.2.2).

2.3.6 Виконати синтез проекту (дивись підрозділ 1.2.3).

2.3.7 Виконати імплементацію проекту, створивши попередньо файл обмежень UCF (рисунок 2.7), використовуючи для цього перемикачі та світлодіоди, зазначені в таблиці 2.2 та на рисунку 2.8.

Таблиця 2.2 – Перелік використовуваних перемикачів і світлодіодів

Перемикачі	SW3	SW2	SW1	SW0	Входи
FPGA Pin	N17	H18	L14	L13	
Світлодіоди	LD3	LD2	LD1	LD0	Виходи
FPGA Pin	F11	E11	E12	F12	

```
NET "X1" LOC = "N17";
NET "X2" LOC = "H18";
NET "X3" LOC = "L14";
NET "X4" LOC = "L13";
NET "Y1" LOC = "F11";
NET "Y2" LOC = "E11";
NET "Y3" LOC = "E12";
NET "Y4"  LOC = "F12".
```

Рисунок 2.7 – Створення файлу обмежень UCF

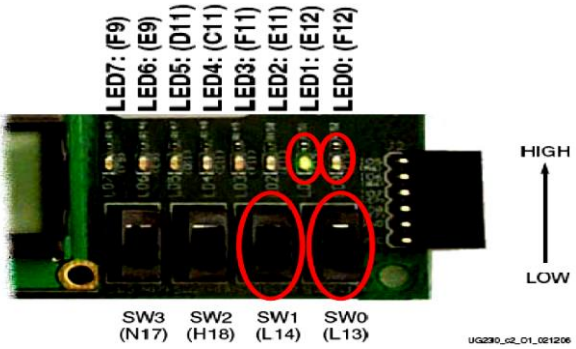


Рисунок 2.8 – Використовувані перемикачі та індикатори

Імплементація проекту розглянута у підрозділі 1.2.4.

2.3.8 Виконати безпосереднє програмування FPGA на платі Xilinx Spartan-3E Starter Kit і перевірити правильність роботи проекту.

Перед тим, як рухатися далі, візьміть плату Spartan-3E Starter Kit, пристрій живлення і USB кабель. Підключіть USB кабель до USB порту вашого комп'ютера. Підключіть пристрій живлення до розетки. Переконайтеся, що встановлено такі перемикачі (і тільки вони!).

Далі підключіть кабель живлення в гніздо живлення та увімкніть блок живлення. Майте на увазі, що якщо програмований файл був попередньо розміщений на плату, він буде автоматично завантажуватися, і може призвести до активності плати як flashing LED, і це може бути небезпечно. На завершення приєднайте USB кабель до його конектора. Зауваження – якщо ви вперше скористалися платою, Windows New Hardware Wizard буде запускатися кілька разів, поки не встановляться драйвери для плати.

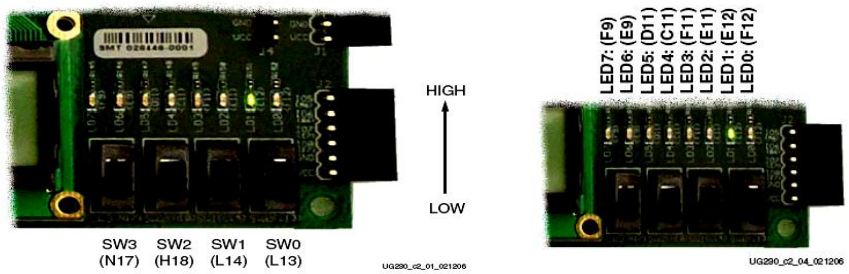
Щоб завантажити ваш bitstream (швидкий прохід за один такт) у FPGA, розкрийте Configure Target Device натисканням на «+» і потім подвійним натисканням завантажте Manage Configuration Project (iMPACT).

Це запустить програму iMPACT в іншому вікні. Частина завдання, що залишилася, буде виконана в iMPACT. Після запуску iMPACT ви побачите серію ДВ (дивись підрозділ 1.2.4).

Тепер ми можемо протестувати наш проект на платі.

Встановлюйте перемикачі SW0 і SW1 в 4 можливих положення – 00, 01, 10, 11 (рисунок 2.9, а) і спостерігайте реакцію схеми на ці вхідні значення на індикаторі LD0-LED0 (F12) (рисунок 2.9, б).

Зробивши це одного разу належним чином, ви вже готові до реалізації проекту на Xilinx Platform Flash пристрої. Цей останній крок дає можливість завантажувати ваш проект при вмиканні живлення без використання USB кабелю. У процесі програмування iMPACT налаштовує FPGA на перезавантаження в нього інформації, накопиченої в пристрої Platform Flash.



а б

Рисунок 2.9 – Перемикачі вхідних впливів та індикатори вихідних значень

Ви можете відразу ж протестувати свій проект. Вийдіть з iMPACT, вимкніть живлення і відключіть USB кабель від плати. Зачекайте 3 с і знову ввімкніть живлення. FPGA має завантажити ваш проект автоматично з пристрою Platform Flash. Перевірте, чи це так (дивись підрозділ 1.2.5).

2.3.9 Оформити звіт про виконану лабораторну роботу, використовуючи шаблон.

2.5 Зміст звіту

Звіт оформляється кожним студентом індивідуально і повинен містити;

- титульний аркуш з номером і назвою роботи;
- мету роботи;
- результати синтезу комбінаційної схеми завдання 1, а саме: карту Карно; рівняння функції у вигляді МДНФ (у базисі Буля,

Шеффера), МКНФ (в базисі Буля, Пірса); схеми в базисах Буля, Шеффера, Пірса;

- результати синтезу комбінаційної схеми завдання 2, а саме: таблицю істинності перетворювача кодів; 4 карти Карно; 4 рівняння у вигляді МДНФ; схему в базисі Буля, виконану вручну, VHDL-опис КС перетворювача; лістинг UCF файлу; результати автоматизованого синтезу проекту у вигляді графічного відображення схеми перетворювача;

- висновки до роботи.

Контрольні запитання

1 Назвіть основні ознаки КС. У чому істотність синтезу КС?

2 Поясніть застосування законів склеювання і поглинання.

3 Перерахуйте правила де Моргана.

4 Перерахуйте методи мінімізації БФ.

5 Які символи застосовуються для кодування комірок карт Карно?

6 Як відбувається перехід з базису Буля в базиси Шеффера і Пірса?

ЛАБОРОТОРНА РОБОТА 3
Синтез мультиплексорів, демультиплексорів, дешифраторів та шифраторів у середовищі Project Navigator пакета Xilinx ISE

Мета роботи

Вивчення мультиплексорів, шифраторів, дешифраторів.

3.1 Методичні вказівки з організації самостійної роботи студентів

3.1.1 Мультиплексори і демультиплексори

Мультиплексором (від англійського слова multiplex – багаторазовий) називається комбінаційний вузол, здатний комутувати (передавати) інформацію з декількох вхідних шин на

одну вихідну. За допомогою мультиплексора здійснюється тимчасовий поділ інформації, що надходить по різних каналах. На рисунку 3.1 наведено приклад мультиплексора 4 в 1.

Мультиплексори мають дві групи входів і один, рідше два взаємодоповнювальних виходи. Входи D_0-D_3 є інформаційними, входи A_1-A_2 – керівними (адресними). Набір сигналів на адресних входах *A* визначає конкретний інформаційний вхід D_0-D_3, який буде з'єднаний з вихідним каналом. У таблиці 3.1 наведені значення адрес для відповідних входів.

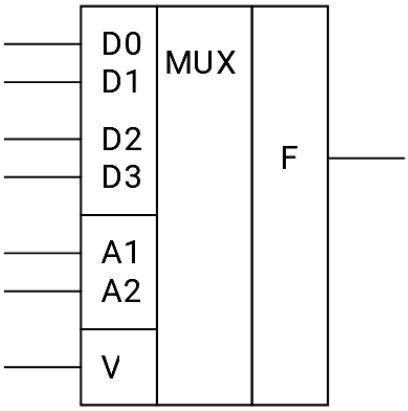


Рисунок 3.1 – Умовне позначення стробувального мультиплексора MUX 4 в 1

Таблиця 3.1 – Інформаційні входи та їх адреси

Інформаційні входи	Адреса інформаційних входів
	A_1 A_2
D_0	0 0
D_1	0 1
D_2	1 0
D_3	1 1

Вирішальний (стробувальний) вхід V керує одночасно всіма інформаційними входами незалежно від стану адресних входів. Заборонний сигнал на цьому вході блокує дію всього пристрою. Наявність дозвільного входу V розширює функціональні можливості мультиплексорів, дозволяючи синхронізувати його роботу з роботою інших вузлів.

На рисунку 3.2 наведено механічний аналог мультиплексора MUX 4 в 1. Якщо V = 0, то F = 0, тобто буде виконуватися комутація з нулем.Якщо V = 1, то F буде комутуватися з каналом відповідно до поданої адреси на входи A1 A2, тобто мультиплексор буде виконувати свою основну функцію. Мультиплексор на рисунку 3.1 реалізує функцію, подану в таблиці 3.2.

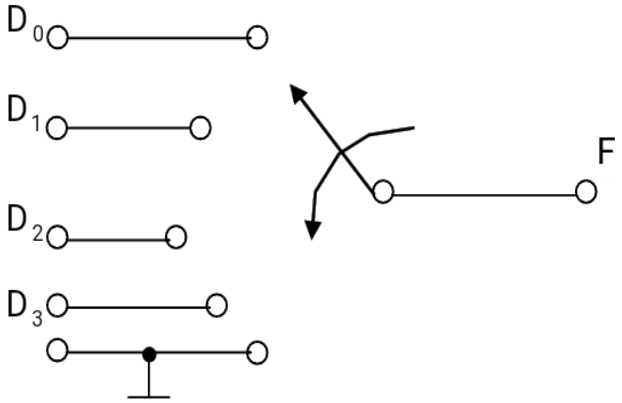


Рисунок 3.2 – Механічний аналог мультиплексора MUX 4 в 1

Таблиця 3.2 – Таблиця істинності MUX 4 в 1

V	A_1	A_2	D_0	D_1	D_2	D_3	F
1	0	0	0	x	x	x	0
1	0	0	1	x	x	x	1
1	0	1	x	0	x	x	0
1	0	1	x	1	x	x	1
1	1	0	x	x	0	x	0
1	1	0	x	x	1	x	1
1	1	1	x	x	x	0	0
1	1	1	x	x	x	1	1
0	x	x	x	x	x	x	0

Дозвільний вхід використовується також при нарощуванні кількості вхідних інформаційних каналів. Мультиплексор на рисунку 3.1 (зі стробувальним входом) згідно з таблицею 3.1 реалізує функцію

$$F = V(\bar{A}_2\bar{A}_1 \cdot D_0 \vee \bar{A}_2 A_1 \cdot D_1 \vee A_2\bar{A}_1 \cdot D_2 \vee A_2 A_1 \cdot D_3) =$$
$$= V\bar{A}_2\bar{A}_1 \cdot D_0 \vee V\bar{A}_2 A_1 \cdot D_1 \vee VA_2\bar{A}_1 \cdot D_2 \vee VA_2 A_1 \cdot D_3$$

Реалізація цієї функції в базисі Буля подана на рисунку 3.3.

Демультиплексори (DMX) виконують зворотне перетворення інформації, на відміну від мультиплексора. Демультиплексор виконує комутацію одного вхідного інформаційного каналу з одним з декількох вихідних каналів. Кількість вихідних каналів демультиплексора 2n, де n – кількість адресних входів. Як демультиплексори можна використовувати дешифратори. Демультиплексор з 1 у 2 подано на рисунку 3.4.

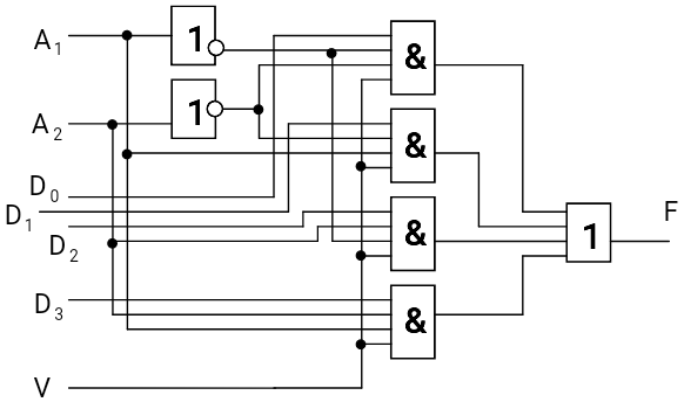


Рисунок 3.3 – Реалізація мультиплексора MUX 4 у 1 в базисі Буля

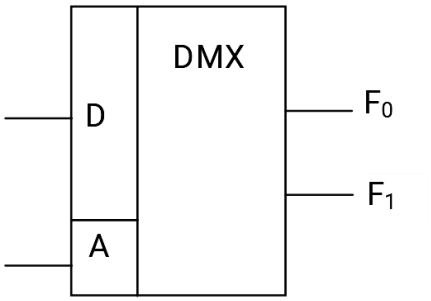


Рисунок 3.4 – Демультиплексор DMX 1 в 2

У таблиці 3.3 наведені значення адрес для відповідних виходів. На рисунках 3.4, 3.5 наведено механічний аналог демультиплексора 1 у 2. Коли A = 0, комутується D і F0, коли A = 1, комутується D і F1. У таблиці 3.4 наведено таблицю істинності DMX 1 в 2. Запис 0/z означає, що на виході 0 або z, 0 і z відповідають різним таблицям істинності. Символ z означає стан високого імпедансу або високого опору на виході (обрив зв'язку).

Таблиця 3.3 – Виходи і їх адреси в DMX 1 в 2

Інформаційні виходи	Адреса інформаційних виходів A
F_0	0
F_1	1

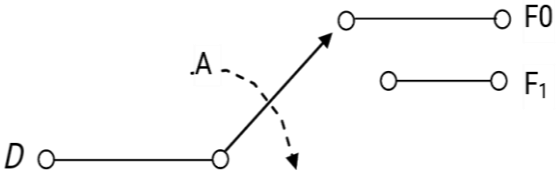


Рисунок 3.5 – Механічний аналог мультиплексора DMX 1 у 2

Незалежно від того, що на виході (0 або z), функція реалізується рівняннями. На рисунку 3.6 наведена структура демультиплексора 1 у 2.

Таблиця 3.4 – Таблиця істинності DMX 2 в 1

A	D	F_0	F_1
0	0	0	0/z
0	1	1	0/z
1	0	0/z	0
1	1	0/z	1

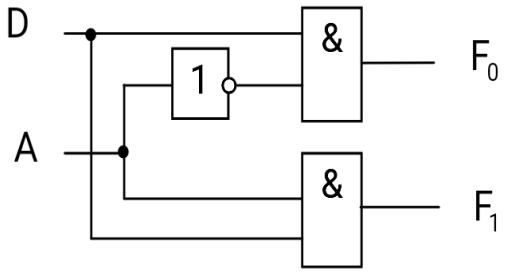


Рисунок 3.6 – Структура мультиплексора DMX 1 у 2

3.1.2 Дешифратори і шифратори

Комбінаційна логічна схема, яка перетворює двійковий позиційний код, що надходить на її входи, в активний сигнал тільки на одному з виходів (унітарний код), називається дешифратором (від англ. Decoder). Якщо кількість двійкових розрядів дешифрованого коду позначити через n, то кількість виходів дешифратора дорівнює 2n. На рисунку 3.7 зображений дешифратор з 2 у 4. Зліва – входи 1, 2 – степені двійки, далі для зручності умовно будемо їх позначати D1, D2, V – стробувальний вхід. Праворуч – виходи 0, 1, 2, 3 – десятковий еквівалент подається на входи коду, для зручності будемо далі їх позначати Q0, Q1, Q2, Q3.

Функції дешифратора подано в таблиці 3.5. Записавши МДНФ для кожної функції виходу, отримаємо такі рівняння: $Q_0 = \overline{D_1}\,\overline{D_2}$, $Q_1 = \overline{D_1} D_2$, $Q_2 = D_1 \overline{D_2}$, $Q_3 = D_1 D_2$. З урахуванням стробувального сигналу рівняння мають такий вигляд: $Q_0 = V\overline{D_1}\,\overline{D_2}$, $Q_1 = V\overline{D_1} D_2$, $Q_2 = V D_1 \overline{D_2}$, $Q_3 = V D_1 D_2$.

У ЕОМ за допомогою дешифраторів здійснюється вибірка необхідних комірок ЗП (запам'ятовувальних пристроїв), розшифрування кодів операцій з видачею відповідних керуючих сигналів, реалізація БФ. Реалізація дешифратора в базисі Буля подана на рисунку 3.8.

Якщо елементи І (кон'юнктор) у схемі дешифратора (рисунок 3.8) замінити на елементи Шеффера (І-НЕ), то отримаємо дешифратор з інверсними виходами, що показується на виходах кружками. Так як дешифратори реалізують БФ, які є конституентами одиниці, то будь-яку БФ можна реалізувати на базі дешифратора з прямими виходами і логічних схем АБО та на

базі дешифратора з інверсними виходами і логічних схем І-НЕ (рисунок 3.9).

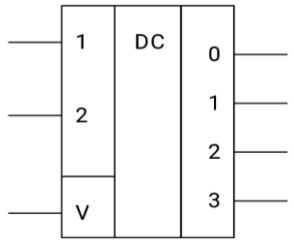


Рисунок 3.7 – Умовне позначення дешифратора 2 в 4

Таблиця 3.5 – Таблиця істинності DC

D_1, D_2	Q_0, Q_1, Q_2, Q_3.
0 0	**1** 0 0 0
0 1	0 **1** 0 0
1 0	0 0 **1** 0
1 1	0 0 0 **1**

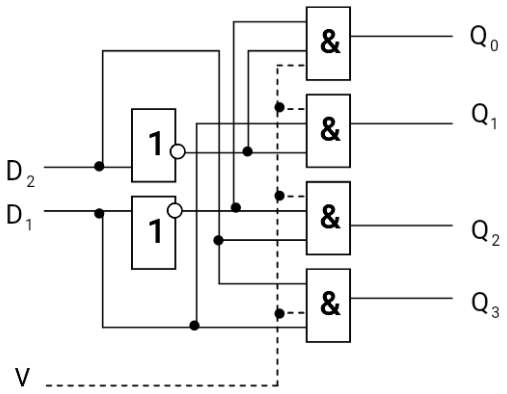
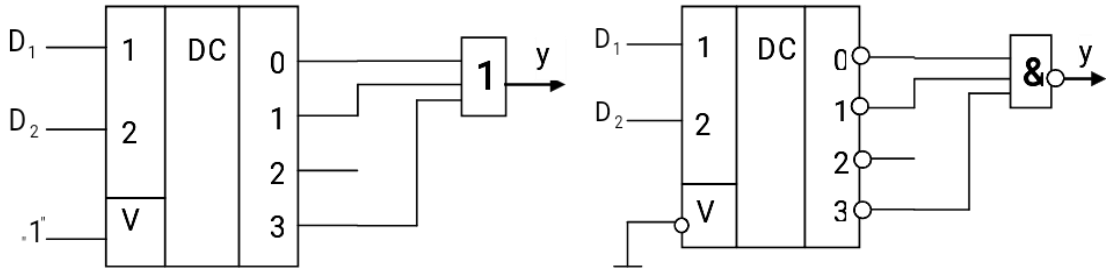


Рисунок 3.8 – Реалізація дешифратора DC 2 в 4 в базисі Буля



а б

Рисунок 3.9 – Реалізація БФ на основі дешифратора з прямими (а) та інверсними (б) виходами

Двійкові шифратори перетворять код "1 з N" на вході у двійковий позиційний код на виході, тобто виконують зворотне дешифратору перетворення інформації. При порушенні однієї з

вхідних шин шифратора на його виходах формується слово – двійковий код номера збудженої шини (рисунок 3.10).

В умовних позначеннях шифраторів використовуються літери CD (від слова coder). Таблицею, яка описує функціонування шифратора, є таблиця 3.5, з тією лише різницею, що вхідними є булеві змінні, а вихідними – БФ шифратора. Функція шифратора подана в таблиці 3.6.

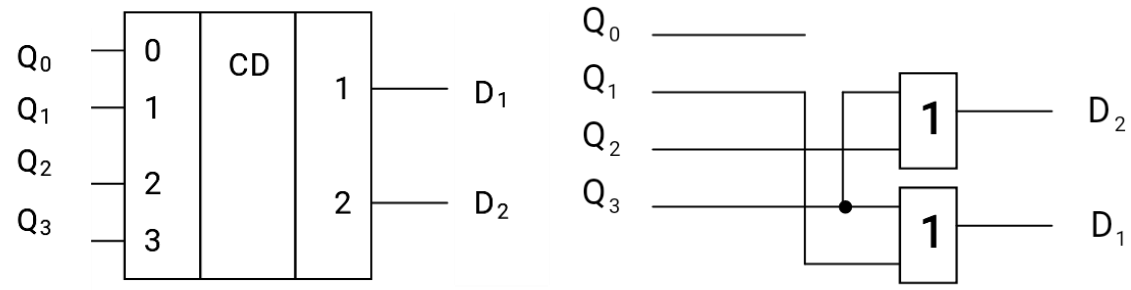


Рисунок 3.10 – Умовне позначення та структура дешифратора 4 у 2

Записавши МДНФ для кожної функції виходу, отримаємо такі рівняння:

$$D_1 = \overline{Q}_0\overline{Q}_1Q_2\overline{Q}_3 \vee \overline{Q}_0\overline{Q}_1\overline{Q}_2Q_3,\ D_2 = \overline{Q}_0Q_1\overline{Q}_2\overline{Q}_3 \vee \overline{Q}_0\overline{Q}_1\overline{Q}_2Q_3.$$

Структура шифратора 4 у 2 подана на рисунку 3.11.

Таблиця 3.6 – Таблиця істинності CD 4 в 2

Q_0, Q_1, Q_2, Q_3.				D_1, D_2
1	0	0	0	0 0
0	**1**	0	0	0 1
0	0	**1**	0	1 0
0	0	0	**1**	1 1
На всіх інших наборах				0 0

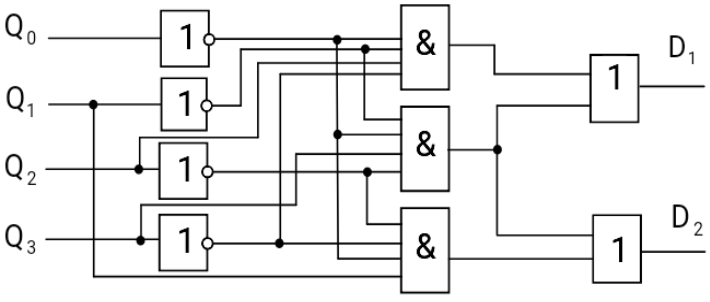


Рисунок 3.11 – Структура шифратора 4 у 2

3.2 Опис лабораторної установки

У лабораторних роботах використовуються пакет Xilinx ISE Project Navigator і плата Xilinx Spartan-3E Starter Kit.

Методична література і додаткове програмне забезпечення розміщені на диску D кожного комп'ютера в лабораторії 3.427. Інструкція з використання програмних засобів міститься в тому ж каталозі.

3.3 Порядок виконання лабораторної роботи

3.3.1 Ознайомитися з літературою до лабораторної роботи. Перед виконанням роботи необхідно відповісти на контрольні запитання.

3.3.2 Кожен студент індивідуально виконує свій варіант завдання.

3.3.3 Виконати лабораторну роботу відповідно до нижченаведених пунктів.

1 Нарисувати умовне графічне зображення проектованої комбінаційної схеми пристрою відповідно до варіанта (таблиця 3.7). Спроектувати відповідно до варіанта одну із запропонованих схем: мультиплексор (MUX), демультиплексор (DMX) (рисунок 3.12, таблиця 3.8), дешифратор (DC), шифратор (CD).

2 Написати таблицю 3.9 істинності для цього пристрою. Запис 0 / z означає, що на виході може бути або 0, або z, 0 і z відповідають різним таблицям істинності. Символ z означає стан високого імпедансу або високого опору на виході (обрив зв'язку).

3 Отримати МДНФ або МКНФ функції проектованого пристрою. Незалежно від того, що на виході (0 або z), функція реалізується рівняннями: $F_0 = \overline{A}D$, $F_1 = AD$.

4 За результатами синтезу побудувати комбінаційну схему в базисі Буля.

Таблиця 3.7 – Варіанти завдань

Варіант	**Завдання**	Варіант	**Завдання**	Варіант	**Завдання**
1	MUX «2 в 1»	12	DMX «1 у 2» зі стробувальним входом V	23	DC «3 в 7»
2	MUX «2 в 1» зі стробувальним входом V	13	DMX «1 у 3» зі стробувальним входом V	24	DC «3 у 8»
3	MUX «3 в 1»	14	DMX «1 у 4» зі стробувальним входом V	25	DC «1 у 2» зі стробувальним входом V
4	MUX «3 в 1» зі стробувальним входом V	15	DMX «1 у 5» зі стробувальним входом V	26	DC «2 в 3» зі стробувальним входом V
5	DMX «1 у 2»	16	DMX «1 у 6» зі стробувальним входом V	27	DC «3 в 5» зі стробувальним входом V
6	DMX «1 у 3»	17	DMX «1 у 7» зі стробувальним входом V	28	DC «3 в 6» зі стробувальним входом V
7	DMX «1 у 4»	18	DMX «1 у 8» зі стробувальним входом V	29	DC «3 в 7» зі стробувальним входом V
8	DMX «1 у 5»	19	DC «1 у 2»	30	DC «3 у 8» зі стробувальним входом V
9	DMX «1 у 6»	20	DC «2 в 3»	31	CD «2 в 1»
10	DMX «1 у 7»	21	DC «3 в 5»	32	CD «2 в 1» зі стробувальним входом V
11	DMX «1 у 8»	22	DC «3 в 6»	33	CD «4 у 2»

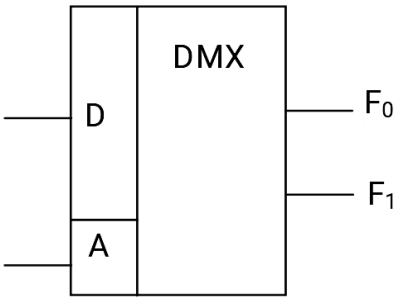


Рисунок 3.12 – Демультиплексор DMX 1 у 2

Таблиця 3.8 – Виходи і їх адреси в DMX 1 у 2

Інформаційні виходи	Адреса інформаційних виходів A
F_0	0
F_1	1

На рисунку 3.13 наведена структура демультиплексора 1 у 2.

Таблиця 3.9 – Таблиця істинності DMX 2 в 1

A	D	F_0	F_1
0	0	0	0/z
0	1	1	0/z
1	0	0/z	0
1	1	0/z	1

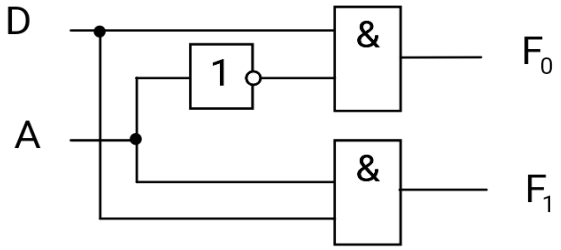


Рисунок 3.13 – Структура демультиплексора DMX 1 у 2

3.3.4 Написати VHDL-код спроектованого пристрою по МДНФ, використовуючи логічні оператори not, and, or. Скористатися кодом комбінаційної схеми, наведеним нижче. Схема має 4 входи, 2 виходи, описана двома булевими рівняннями (рисунок 3.14): $Y1 = \overline{X3} \cdot \overline{X4} \cdot X2 \vee \overline{X3} \cdot X4$, $Y2 = \overline{X3} \cdot \overline{X4} \cdot X1 \vee X3 \cdot X4 \cdot X2$

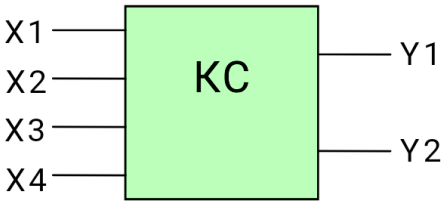


Рисунок 3.14 – Приклад комбінаційної схеми

```
-- Підключення бібліотеки ieee.
library ieee;
-- Підключення пакета бібліотеки ieee.
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
-- Опис інтерфейсу приладу (входи X1, X2, X3,
X4, виходи Y1, Y2)
-- DMX – ім’я приладу
entity DMX is
port ( X1, X2, X3, X4: in STD_LOGIC;
    Y1, Y2: out STD_LOGIC);
end DMX;
```

```
architecture DMX of DMX is
begin
  Y1 <= (not X3 and not X4 and X2) or (not X3
and X4);
  Y2 <= (not X3 and not X4 and X1)) or (X3 and
X4 and X2);
end KC;
```

Напишемо VHDL-код спроектованого вище пристрою (демультиплексор DMX 1 у 2, (рисунок 3.15) по МДНФ, отриманий за Картами Карно, використовуючи логічні оператори not, and, or. Скористаємося кодом, наведеним вище, для нашої схеми, яка має 2 входи, 2 виходи, описана двома булеві рівняннями: $F_0 = \overline{A}D$, $F_1 = AD$.

Рисунок 3.15 –Схема демультиплексора

```
-- Підключенння бібліотеки ieee.
library ieee;
-- Підключення пакета бібліотеки ieee.
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
--  Опис  інтерфейсу  пристрою   (входи  D,  A,
виходи F0, F1)
-- DMX – ім’я пристрою
entity DMX is
port (
    D, A: in STD_LOGIC;
    F0, F1: out STD_LOGIC);
end DMX;
architecture DMX of DMX is
begin
```

```
  F0 <= (not A and D);
  F1 <= (A and D);
end DMX;
```

3.3.5 Створити проект у середовищі Project Navigator, додати в проект VHDL-опис комбінаційної схеми (дивись підрозділ 1.2.2).

3.3.6 Виконати синтез проекту (дивись підрозділ 1.2.3).

3.3.7 Виконати імплементацію проекту, створивши попередньо UCF файл (дивись підрозділ 1.2.4). Використовуйте для цього перемикачі та кнопки, зазначені в таблиці 3.10 і на рисунку 3.16 і світлодіоди, зазначені в таблиці 3.11 і на рисунку 3.17.

Таблиця 3.10 – Перелік перемикачів і кнопок для задання вхідних впливів

Перемикачі	SW3	SW2	SW1	SW0	
FPGA Pin	N17	H18	L14	L13	
Кнопки	BTN_WEST	BTN_NORTH	BTN_EAST	BTN_SOUTH	ROT_CENTER
FPGA Pin	D18	V4	H13	K17	V16

Таблиця 3.11 – Перелік світлодіодів для спостереження вихідних реакцій

Світлодіоди	LED7	LED6	LED5	LED4
FPGA Pin	F9	E9	D11	C11
Світлодіоди	LED3	LED2	LED1	LED0
FPGA Pin	F11	E11	E12	F12

Приклад UCF файлу при використанні кнопок для подачі вхідних впливів. Для того, щоб не натиснута кнопка відповідала «0», а натиснута – «1», додайте | IOSTANDARD = LVTTL | PULLDOWN після кожного призначення сигналу виведення мікросхеми, наприклад:

```
NET "a"  LOC = "D18" | IOSTANDARD = LVTTL | PULLDOWN ;
NET "b"  LOC = "V4"  | IOSTANDARD = LVTTL | PULLDOWN ;
NET "c"  LOC = "H13" | IOSTANDARD = LVTTL | PULLDOWN ;
NET "d"  LOC = "K17" | IOSTANDARD = LVTTL | PULLDOWN ;
```

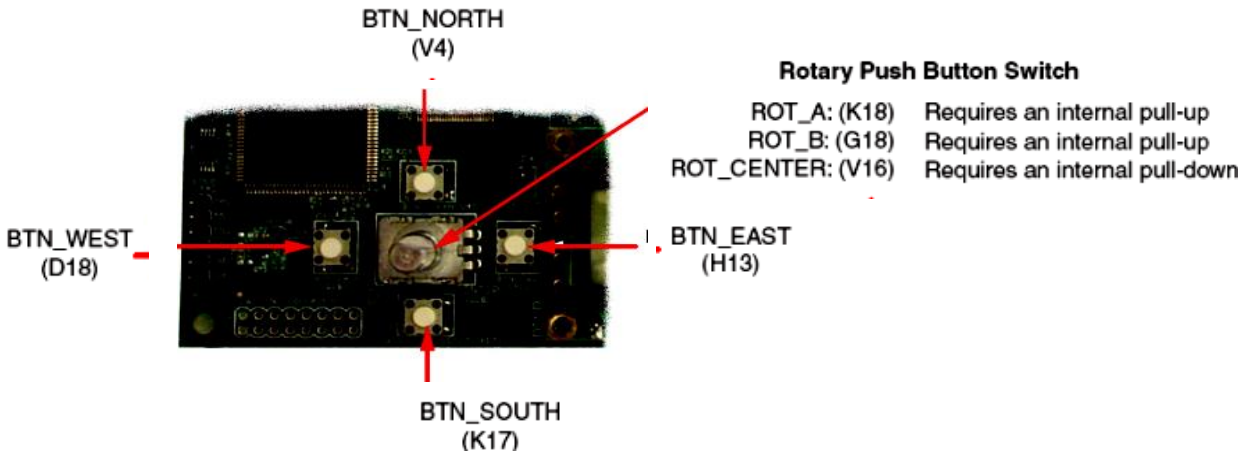


Рисунок 3.16 – Кнопки для задання вхідних впливів

Рисунок 3.17 – Перемикачі вхідних впливів та індикатори вихідних значень

Для нашого випадку F0 <= (not A and D); F1 <= (A and D);

```
NET "A"  LOC = "N17" | IOSTANDARD = LVTTL | PULLDOWN ;
NET "D" LOC = "H18" | IOSTANDARD = LVTTL | PULLDOWN ;
NET "F0"  LOC = "E12" | IOSTANDARD = LVTTL | PULLDOWN ;
NET "F1"  LOC = "F12" | IOSTANDARD = LVTTL | PULLDOWN ;
```

3.3.8 Виконати безпосереднє програмування FPGA на платі Xilinx Spartan-3E Starter Kit і перевірити правильність роботи проекту (дивись підрозділ 1.2.5).

Безпосереднє програмування FPGA – зручний спосіб випробувати проект. Цей метод корисний для прототипування, коли проект не є остаточним, щоб упевнитися в правильності проекту. Однак складні проекти не завжди працюють «з першої спроби». Однією з переваг FPGAs перед ASICs схемами, є те, що ціна за помилку при першій спробі мінімальна.

Тепер потрібно створити файл для програмування FPGA. Виберіть DMX_2_1 у вікні Sources. Потім подвійним натисканням виберіть Generate Programming File процес у вікні Processes. Project Navigator буде генерувати файл для програмування FPGA і видавати інформацію про хід імплементації у вікні Console.

3.3.9 Оформити звіт про виконану лабораторну роботу, використовуючи шаблон.

3.4 Зміст звіту

Звіт оформляється кожним студентом індивідуально і повинен містити:

- титульний аркуш з номером і назвою роботи;
- мету роботи;
- умовне графічне зображення проектованої комбінаційної схеми, таблицю істинності для цього пристрою, МДНФ або МКНФ функції проектованого пристрою, комбінаційну схему в базисі Буля;
- VHDL-опис комбінаційної схеми; лістинг UCF файлу; результати автоматизованого синтезу проекту у вигляді графічного відображення схеми;
- висновки до роботи.

Контрольні запитання

1 Визначення системи-на-кристалі. Узагальнена архітектура системи-на-кристалі.

2 Які пристрої належать до класу програмованої логіки та до спеціалізованих мікросхем?

3 На якому етапі проектування відбувається перетворення опису пристрою, що є схемою з елементів цільової мікросхеми, у двійковий файл для програмування мікросхеми?

ЛАБОРАТОРНА РОБОТА 4
Вивчення мовного опису моделей тригерів

Мета роботи

Отримання навичок у побудові VHDL-опису моделей тригерів.

4.1 Вказівки до виконання лабораторної роботи

4.1.1 D-тригер, синхронізований рівнем 1, зображено на рисунку 4.1.

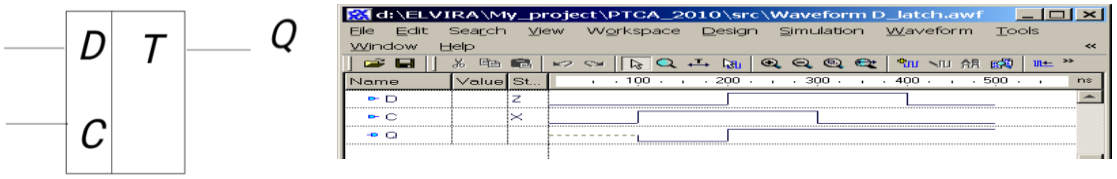


Рисунок 4.1 – D-тригер, синхронізований рівнем 1, та його часова діаграма

Лістинг 4.1 – D-тригер, синхронізований рівнем 1

```
library IEEE;
use IEEE.std_logic_1164.all;
entity D_latch is
port   (D: in STD_LOGIC;
C: in STD_LOGIC;
Q: out STD_LOGIC);
end D_latch;

architecture D_latch of D_latch is
   begin
process (C, D)
begin
if C='1' then  -- синхронний запис Q за рівнем 1
Q <= D;
end if;
end process;
end D_latch;
```

4.1.2 D-тригер, синхронізований рівнем 1 з асинхронним скиданням в 0, зображено на рисунку 4.2.

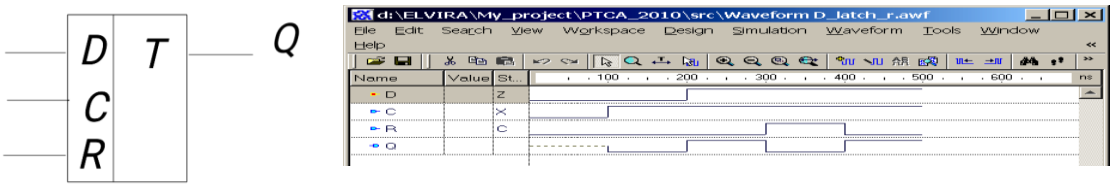


Рисунок 4.2 – D-тригер, синхронізований 1 з асинхронним скиданням в 0, та його часова діаграма

Лістинг 4.2 – D-тригер, синхронізований 1 з асинхронним скиданням в 0

```
library IEEE;
use IEEE.std_logic_1164.all;
entity D_latch_R is
port   (D: in STD_LOGIC;
C: in STD_LOGIC;
R: in STD_LOGIC;
Q: out STD_LOGIC);
end D_latch_R;
architecture D_latch_R of D_latch_R is
   begin
process (C, R)
begin
if R='1' then     --асинхронне скидання в 0
      Q <= '0';
elsif  C='1'  then   --  синхронний  запис  Q  за
рівнем 1
      Q <= D;
end if;
end process;
end D_latch_R;
```

4.1.3 D-тригер, синхронізований переднім фронтом з асинхронним скиданням в 0, зображено на рисунку 4.3.

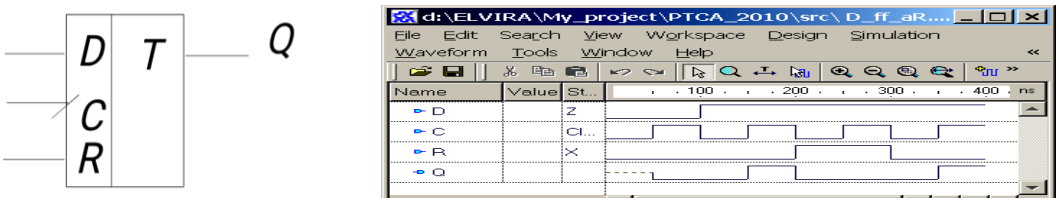


Рисунок 4.3 – D-тригер, синхронізований переднім фронтом з асинхронним скиданням в 0, та його часова діаграма

Лістинг 4.3 – D-тригер, синхронізований переднім фронтом з асинхронним скиданням в 0

```
library IEEE;
use IEEE.std_logic_1164.all;
entity D_ff_aR is
port   (D: in STD_LOGIC;
C: in STD_LOGIC;
R: in STD_LOGIC;
Q: out STD_LOGIC);
end D_ff_aR;
architecture D_ff_aR of D_ff_aR is
   begin
process (C, R)
begin
if R='1' then Q <= '0';      --асинхронне
скидання  в 0
elsif (C'event and C='1') then
--синхронний запис Q за переднім фронтом
-  передній  фронт  описаний  атрибутом  event
(подія) і C='1'
Q <= D;
end if;
end process;
end D_ff_aR;
```

4.2.4 D-тригер, синхронізований заднім фронтом з синхронним скиданням в 0, зображено на рисунку 4.4.

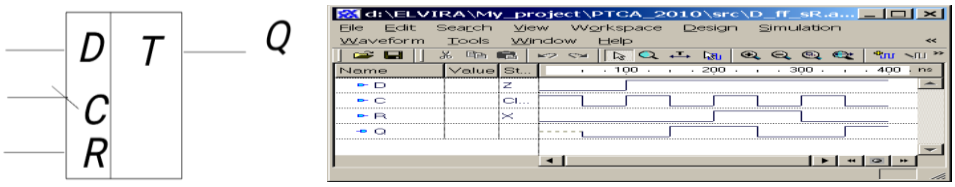


Рисунок 4.4 – D-тригер, синхронізований заднім фронтом із синхронним скиданням в 0, та його часова діаграма

Лістинг 4.4 – D-тригер, синхронізований заднім фронтом з синхронним скиданням в 0

```
library IEEE;
use IEEE.std_logic_1164.all;
entity D_ff_sR is
port   (D: in STD_LOGIC;
C: in STD_LOGIC;
R: in STD_LOGIC;
Q: out STD_LOGIC);
end  D_ff_sR;
architecture  D_ff_sR of  D_ff_sR is
begin
process (C)
begin
if (C'event and C='0') then --синхронний запис
Q по задньому фронту
if R='1' then      --синхронне скидання в 0
Q <= '0';
else
Q <= D;
end if;
end if;
end process;
end D_ff_sR;
```

4.1.5 JK-тригер, синхронізований заднім фронтом з асинхронним скиданням в 0

На рисунку 4.5 наведено умовне позначення і VHDL-модель синхронного JK-тригера, керованого заднім фронтом, з асинхронним установленням в 0. Особливістю цієї моделі є те, що для реалізації операції «інверсія стану» використовується внутрішня змінна Qint, так як сигнал Q, декларований в операторі port, не може стояти в правій частині оператора паралельного призначення сигналу <=.

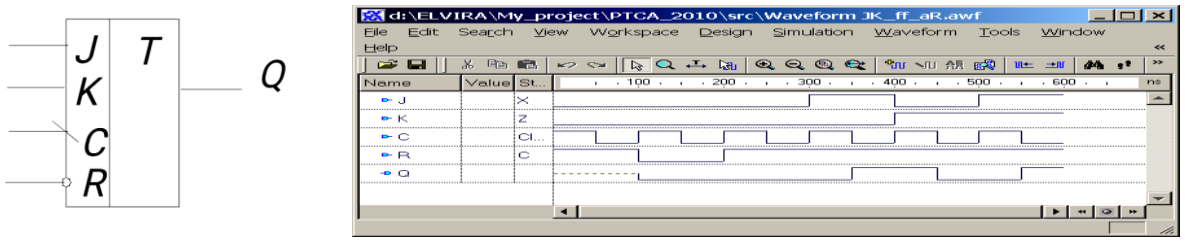


Рисунок 4.5 – JK-тригер, синхронізований заднім фронтом із асинхронним скиданням в 0, та часова діаграма його роботи

Лістинг 4.5 – JK-тригер, синхронізований заднім фронтом із асинхронним скиданням в 0

```
library IEEE;
use IEEE.std_logic_1164.all;
entity JK_tr is
port   (J: in STD_LOGIC;
K:in STD_LOGIC;
C:in STD_LOGIC;
R: in STD_LOGIC;
Q: out STD_LOGIC);
end JK_tr;
architecture JK_tr of JK_tr is
begin
process(C, R) is
variable Qint: STD_LOGIC;
begin
if   (R   ='0')   then   Qint:='0';   --асинхронне
скидання в 0 (R - інверсій)
```

```
elsif (falling_edge(C)) then --синхронний запис
Q по задньому фронту
if (J='1' and K='1') then Qint := not(Qint);
elsif (J='0' and K='0') then Qint:=Qint;
elsif (J='1' and K='0') then Qint:='1';
elsif (J='0' and K='1') then Qint:='0';
end if;
      - задній фронт описаний за допомогою
функції falling_edge (C)
end if;
Q<=Qint;
end process;
end JK_tr;
```

4.1.6 JK-тригер, синхронізований переднім фронтом із синхронним скиданням в 0, зображено на рисунку 4.6.

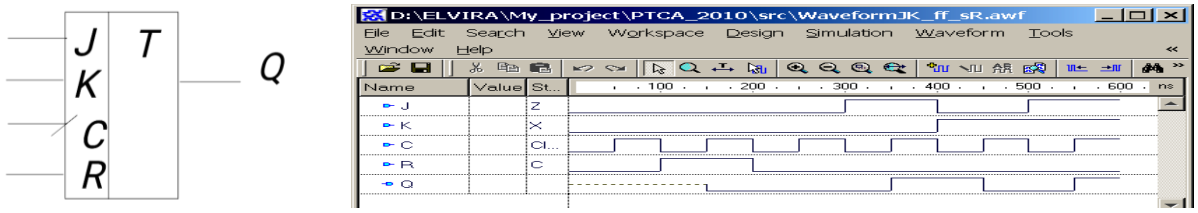


Рисунок 4.6 – JK-тригер, синхронізований переднім фронтом із синхронним скиданням в 0, та часова діаграма його роботи

Лістинг 4.6 – JK-тригер, синхронізований переднім фронтом із синхронним скиданням в 0

```
library IEEE;
use IEEE.std_logic_1164.all;
entity JK_tr is
port   (J, K, C, R: in STD_LOGIC;
Q: out STD_LOGIC);
end JK_tr;
architecture JK_tr of JK_tr is
begin
process(C, R) is
variable Qint: STD_LOGIC;
begin
```

```
if (rising_edge(C)) then    --синхронний запис
Qint по задньому фронту
if  (R  ='1')  then  Qint:='0';  --синхронне
скидання в 0 (R - прямий)
elsif (J='1' and K='1') then Qint := not(Qint);
elsif (J='0' and K='0') then Qint:=Qint;
elsif (J='1' and K='0') then Qint:='1';
elsif (J='0' and K='1') then Qint:='0';
end if;
     - передній фронт описаний за допомогою
функції rising_edge (C)
end if;
Q<=Qint;
end process;
end JK_tr;
```

4.2 Опис лабораторної установки

У лабораторних роботах використовуються пакет Xilinx ISE Project Navigator і плата Xilinx Spartan-3E Starter Kit. Методична література і додаткове програмне забезпечення розміщене на диску D кожного комп'ютера в лабораторії 3.427. Інструкція з використання програмних засобів міститься в тому ж каталозі.

4.3 Порядок виконання лабораторної роботи

4.3.1 Ознайомитися з літературою до лабораторної роботи. Перед виконанням роботи необхідно відповісти на контрольні запитання.

4.3.2 Кожен студент індивідуально виконує запропонований варіант завдання.

4.3.3 Вибрати VHDL-код D-тригера, синхронізованого переднім фронтом з асинхронним скиданням в 0 (лістинг 4.3). Виконати для нього пункти 4.3.5 – 4.3.10.

4.3.4 Написати VHDL-код тригера, обраного відповідно до варіанта з таблиці 4.1. Виконати для нього пункти 4.3.5 – 4.3.10.

4.3.5 Створити проект у середовищі Project Navigator, додати в проект VHDL-опис тригера.

4.3.6 Виконати синтез проекту.

4.3.7 Виконати імплементацію проекту, створивши попередньо UCF файл.

Таблиця 4.1 – Варіанти завдань

Варіант	Завдання
1	2
1	*J*K**-тригер, синхронізований переднім фронтом з асинхронним скиданням в 0
2	*J*K**-тригер, синхронізований переднім фронтом з асинхронним установленням в 1
3	*J*K**-тригер, синхронізований заднім фронтом з асинхронним скиданням в 0
4	*J*K**-тригер, синхронізований заднім фронтом з асинхронним установленням в 1
5	*J*K**-тригер, синхронізований переднім фронтом із синхронним скиданням в 0
6	*J*K**-триггер, синхронізований переднім фронтом із синхронним установленням в 1
7	*J*K**-тригер, синхронізований заднім фронтом із синхронним скиданням в 0
8	*J*K**-тригер, синхронізований заднім фронтом із синхронним установленням в 1
9	*T*-тригер, синхронізований переднім фронтом із асинхронним скиданням в 0
10	*T*-тригер, синхронізований переднім фронтом із асинхронним установленням в 1
11	*T*-тригер, синхронізований заднім фронтом із асинхронним скиданням в 0
12	*T*-тригер, синхронізований заднім фронтом із асинхронним установленням в 1
13	*T*-тригер, синхронізований переднім фронтом із синхронним скиданням в 0
14	*T*-тригер, синхронізований переднім фронтом із синхронним установленням в 1
15	*T*-тригер, синхронізований заднім фронтом із синхронним скиданням в 0

Продовження таблиці 4.1

1	2
16	*T*-тригер, синхронізований заднім фронтом із синхронним установленням в 1
17	*JK*-тригер, синхронізований переднім фронтом із асинхронним скиданням в 0
18	*JK*-тригер, синхронізований переднім фронтом із асинхронним установленням в 1
19	*JK*-тригер, синхронізований заднім фронтом із асинхронним установленням в 1
20	*JK*-тригер, синхронізований переднім фронтом із синхронним установленням в 1
21	*JK*-тригер, синхронізований заднім фронтом із синхронним скиданням в 0
22	*JK*-тригер, синхронізований заднім фронтом із синхронним установленням в 1
23	*T**-тригер, синхронізований переднім фронтом із асинхронним скиданням в 0
24	*T**-тригер, синхронізований переднім фронтом із асинхронним установленням в 1
25	*T**-тригер, синхронізований заднім фронтом із асинхронним скиданням в 0
26	*T**-тригер, синхронізований заднім фронтом із асинхронним установленням в 1
27	*T**-тригер, синхронізований переднім фронтом із синхронним скиданням в 0
28	*T**-тригер, синхронізований переднім фронтом із синхронним установленням в 1
29	*T**-тригер, синхронізований заднім фронтом із синхронним скиданням в 0
30	*T**-тригер, синхронізований заднім фронтом із синхронним установленням в 1

Використовувати для цього потрібно перемикачі, зазначені в таблиці 4.2 і на рисунку 4.7, і світлодіоди, зазначені в таблиці 4.3 і на рисунку 4.8, а також директиву для синхросигналу (лістинг 4.8).

Таблиця 4.2 – перелік використовуваних перемикачів і кнопок для задання вхідних впливів

Перемикачі	SW3	SW2	SW1	SW0
FPGA Pin	N17	H18	L14	L13

Таблиця 4.3 – Перелік використовуваних світлодіодів для спостереження вихідних реакцій

Світлодіоди	LED7	LED6	LED5	LED4
FPGA Pin	F9	E9	D11	C11
Світлодіоди	LED3	LED2	LED1	LED0
FPGA Pin	F11	E11	E12	F12

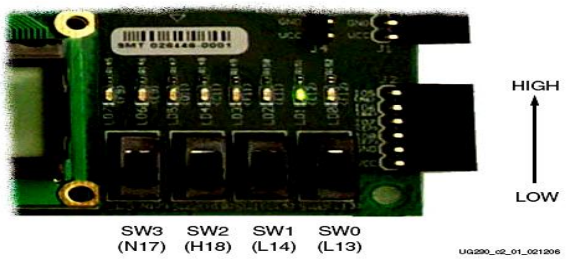


Рисунок 4.7 – Перемикачі вхідних впливів

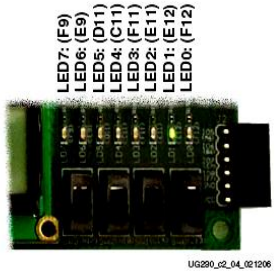


Рисунок 4.8 – Індикатори вихідних значень

У лістингу 4.7 наведений перший варіант UCF файлу для D-тригера.

Лістинг 4.7 – Перший варіант UCF файлу для D-тригера

```
NET "R"   LOC = "H18";
NET "D"   LOC = "L14";
NET "Q"   LOC = "F12";
NET "C"   LOC = "N17";
```

Якщо використовувати при імплементації UCF файл у такому вигляді, при виконанні аналізу розміщення і трасування система видасть повідомлення про помилку:

ERROR:Place:1018 - A C IOB / clock component pair have been found that are not placed at an optimal clock IOB /

Це пов'язано з тим, що сигнали скидання, вибору операнда і синхронізації визначені системою як тактувальні. Призначення сигналів поза апаратних областей тактування призводить до зниження швидкодії (детально дивіться «PAR report» - звіт про розміщення і трасування). «Обхід» такої помилки проводиться за допомогою виключення відповідної директиви. Наприклад, для сигналу C це має вигляд: `NET "C" CLOCK_DEDICATED_ROUTE = FALSE;`

У лістингу 4.8 наведено другий варіант UCF файлу для D-тригера. Саме такий файл необхідно використовувати в проекті.

Лістинг 4.8 – Другий варіант UCF файлу для D-тригера

```
NET "R"  LOC = "H18";
NET "D"  LOC = "L14";
NET "Q"  LOC = "F12";
NET "C"  LOC = "N17";
NET "C" CLOCK_DEDICATED_ROUTE = FALSE;
```

4.3.8 Виконати безпосереднє програмування FPGA на платі Xilinx Spartan-3E Starter Kit.

4.3.9 Скласти тимчасову діаграму роботи тригера, реалізованого на платі, і перевірити правильність його роботи за складеною тимчасовою діаграмою.

4.3.10 Оформити звіт про виконану лабораторну роботу, використовуючи шаблон.

4.4 Зміст звіту

Звіт оформляється кожним студентом індивідуально і повинен містити;

- титульний аркуш з номером і назвою роботи;
- мету роботи;
- умовне графічне зображення досліджуваних тригерів;
- VHDL-опис тригерів;
- тимчасові діаграми перевірки правильності роботи тригерів;
- листинги UCF файлів;

- результати автоматизованого синтезу проекту у вигляді графічного відображення схем тригерів;
- висновки до роботи.

Контрольні запитання

1 Який тригер називається асинхронним та синхронним?

2 Чим відрізняється синхронізація за рівнем від синхронізації по фронту?

3 У чому відмінність між асинхронним і синхронним установленням тригерів у початковий стан?

4 Що таке latch і flip-flop тригери?

5 У чому полягає особливість MS-структури тригерів?

6 Які конструкції VHDL використовуються для опису синтезованих моделей асинхронних тригерів, тригерів, синхронізованих рівнем, і тригерів, синхронізованих фронтом?

СПИСОК ЛІТЕРАТУРИ

1 Семенец В. В., Хаханова И. В., Хаханов В. И., Проектирование цифрових систем с использованием языка VHDL: учеб. пособие. Харьков : ХНУРЭ, 2003. 492 с.

2 Мірошник М. А. Комп'ютерні технології автоматизованого проектування : навч. посібник. Харків : ХНУРЕ, 2007. 300 с.

3 Грушвицкий Р. И., Мурсаев А. Х., Угрюмов Е. П. Проектирование систем на микросхемах программируемой логики. СПб. : БХВ-Петербург, 2002. 608 с.

4 Лістровий С. В., Мірошник М. А. Теорія автоматичного управління, штучний інтелект і автоматизація процесу прийняття рішення : навч. посібник. Харків : УкрДУЗТ, 2018. 144 с.

5 Семенець В. В., Хаханов В. І. Проектування цифрових схем з використанням мови VHDL : навч. посібник. Харків : ХНУРЕ, 2003. 492 с.

6 Среда проектирования Web PACK ISE: метод. пособие к лабораторным работам по курсу «Проектирование устройств на ПЛИС» для студ. спец. дневной формы обуч. Харьков : ХНУРЭ, 2007. 72 с.

7 Хаханов В. І., Литвинова Є. І., Гузь О.А. Проектування і тестування цифрових систем на кристалах : підручник. Харків : ХНУРЕ, 2009. 484 с.

8 Хацук В. А. Проектирование в системе Xilinx ISE. URL: http://www.scan-west.com/xilinx ise.pdf.

9 Лістровий С. В., Мірошник М. А. Інформаційно-управляючі системи та організація паралельних обчислювань : навч. посібник. Харків : Діса плюс, 2015. 324 с.

10 Соловьев В. В. Проектирование цифровых систем на основе программируемых логических интегральных схем. Москва : Горячая линия-Телеком, 2001. 636 с.

11 Бабич Н. П., Жуков И. А. Компьютерная схемотехника. Методы построения и проектирования. Киев : МК-Пресс, 2007. 576 с.

12 Кравчук С. О., Шонін В. О. Основи комп'ютерної техніки: компоненти, системи, мережі : навч. посібник. Київ : ІВЦ «Вид-во Політехніка» : Каравела, 2005. 343 с.

13 Леонов С. Ю., Загарій Г. І. Автоматизоване проектування складних систем у комп'ютерній схемотехніці : навч. посібник. Харків : ПП вид. «Нове слово», 2012. 287 с.

14 Мирошник, М. А. Проектирование диагностической инфраструктуры вычислительных систем и устройств на ПЛИС : монографія. Харьков: ХУПС, 2012. 188 с.

15 Мірошник М. А. Конспект лекцій з дисциплін «САПР пристроїв і систем автоматики» та «Основи систем автоматизації проектування», для студентів спеціальності 123 «СКС». Харків : УкрДУЗТ, 2013. 102 с.