МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

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

Проєктування спеціалізованих комп’ютерних систем на ПЛІС. Лабораторний практикум

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського як навчальний посібник для студентів, які навчаються за спеціальністю 123 Комп’ютерна інженерія

Київ

КПІ ім. Ігоря Сікорського

2021

Рецензенти: *Заболотня Т.М., к.т.н., доцент.*
Корочкін О.В. , к.т.н., доцент.

Відповідальний редактор *Тарасенко В.П., д.т.н.,професор.*

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського(протокол № 7 від 13.05 2021 р.)
за поданням Вченої ради факультету прикладної математики (протокол № 10 від 29.03.2021 р.)
Електронне мережне навчальне видання

Клятченко Ярослав Михайлович, к.т.н., доцент.
Тарасенко-Клятченко Оксана Володимирівна, к.т.н., доцент.
Тесленко Олександр Кирилович, к.т.н., доцент.

Проєктування спеціалізованих комп'ютерних систем на ПЛІС. Лабораторний практикум

Проєктування спеціалізованих комп'ютерних систем на ПЛІС. Лабораторний практикум. [Електронний ресурс] : навч. посіб. для студ. спеціальності 123 Комп'ютерна інженерія / КПІ ім. Ігоря Сікорського ; Я.М. Клятченко, О.В. Тарасенко-Клятченко, О.К. Тесленко. – Електронні текстові дані (1 файл: 2,3 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2021. – 54 с.

Навчальний посібник надає короткі теоретичні відомості, які необхідні для виконання лабораторних робіт, докладні приклади на кожну тему, блоки лабораторних робіт, контрольні запитання, відповідаючи на які студент може перевірити свої знання теоретичного та практичного матеріалу, рекомендовану літературу, яка дозволяє ширше вивчити теоретичний матеріал з тем курсу. Навчальне видання призначене для студентів, які навчаються за спеціальністю 123 «Комп'ютерна інженерія» факультету прикладної математики НТУУ «КПІ ім. Ігоря Сікорського».

ЗМІСТ

Дисципліна «ПРОЄКТУВАННЯ СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ НА ПЛІС»

Вступ

Спеціалізовані комп'ютерні системи (СКС) – це комп'ютерні системи (КС) для розв'язку великого числа відносно вузького класу задач, що оптимізовані в певній критеріальній сукупності» [1]. Приклади критеріїв – продуктивність, швидкість, складність, вартість, технологічність та ін.

Досвід показує, що в багатьох випадках досягнення в СКС становляться здобутком при створенні універсальних КС. З цієї точки зору СКС можна розглядати як розвідувальні засоби комп'ютерної інженерії.

Загальна характеристика етапів проєктування мікросхем та апаратури

Перелік основних загальних етапів розробки апаратури та мікросхем.

1. Розробка технічного завдання та його погодження з перспективними споживачами та користувачами.

2. Розробка поведінкової моделі цифрового пристрою чи мікросхеми (модель, яка відображає об'єкт проєктування в формі правил перетворення вхідних даних в вихідні або правил генерування вихідних даних).

2.1. Опис поведінкової моделі.

2.2. Створення часових діаграм сигналів, визначення найбільш критичних моментів та варіантів синхронізації.

2.3. Верифікація поведінкової моделі (функціональне моделювання) з врахуванням моделей затримок сигналів.

3. Варіанти розробки:

3.1. Розробка зверху – вниз (Ієрархічна декомпозиція – розбиття задачі на підзадачі, які в свою чергу також розбиваються на підзадачі ще нижчого рівня, і так до тих пір поки не вийдуть на вже існуючі рішення. Тут глибина ієрархії – рівень базових елементів).

3.2. Розробка знизу вгору (ієрархічна композиція - компоновка існуючих, або створених на їх основі, рішень, з метою вирішення поставленої задачі).

3.3. Реальна розробка - в реальних проєктах (не залежно, чи апаратура, чи програма) використовується та чи інша комбінація вказаних методів.

4. Створення тестових послідовностей та верифікація проєкту на рівні базових елементів.

5. Створення дослідного зразка.

6. Верифікація дослідного зразка.

Особливості розробки мікросхем.

В випадку проєктування мікросхем рівнем базових елементів є транзистори, тобто створюється схема електрична принципова на рівні транзисторів, хоча можуть використовуватись фірмові бібліотеки типових блоків на транзисторах. На основі електричної принципової схеми,

створюються креслення для масок (пошарова топологія), які будуть використовуватись в технологічному процесі виготовлення мікросхем. Кількість таких масок може значно перевищувати 10. Для зменшення трудомісткості процесу створення топології використовують фірмові бібліотеки типових топологічних вирішень які відповідають бібліотечним

схемам електричним принциповим на рівні транзисторів. Одержану пошарову топологію мікросхеми перевіряють на відповідність схемі електричній принциповій та створюють програми для управління відповідним технологічним обладнанням. Після цього виготовлюється дослідна партія мікросхем. Дослідний зразок перевіряється на тестах. Якщо

в результаті були виявлені помилки (в процесі виготовлення, при розробці топології, схеми електричної чи навіть структурної) виконується корекція та процес повторюється з відповідного етапу (рис. 1 та 2).

Рисунок 1 – Приміщення із виготовлення мікросхем

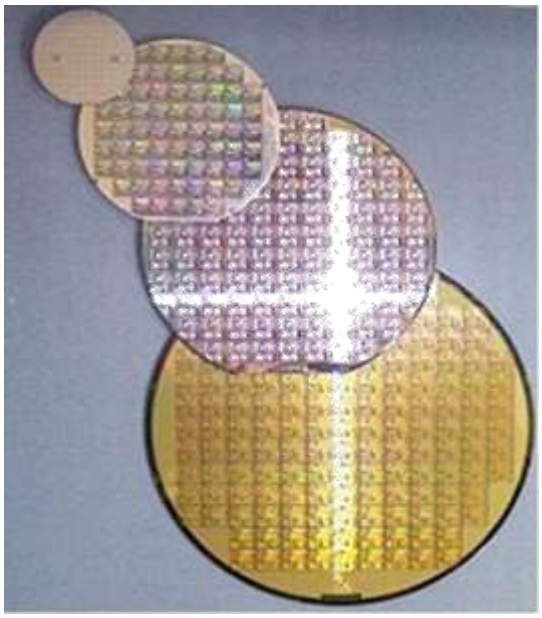


Рисунок 2 – Пластини із кристалами

Особливості розробки апаратури.

Після створення схеми електричної принципової на рівні базових елементів (в якості базових елементів можуть бути мікропроцесори, однокристальні ЕОМ тощо) конструктори розробляють пошарову топологію друкованої плати. Після цього виготовляють дослідну партію друкованих плат (процес виготовлення друкованих плат може зайняти від декількох днів до декількох тижнів), та проводять монтаж базових елементів на друкованих платах. Враховуючи, що сучасні мікросхеми можуть мати декілька сотень виводів (пінів), то монтаж може бути виконаний лише на спеціальному обладнанні (обладнання монтажу на поверхню). Якщо в результаті в дослідному зразку були виявлені помилки (в процесі виготовлення, при розробці топології друкованої плати, схеми електричної чи навіть структурної) виконується корекція та процес повторюється з відповідного етапу.

Таким чином процес розробки та виготовлення апаратури та особливо мікросхем характеризується значними затратами на основні засоби (технологічне обладнання), значною трудомісткістю та досить тривалими строками. Для ефективної амортизації затрат необхідне крупносерійне, а то і масове виробництво, що характерно для універсальних схем та структур. В випадку спеціалізованих схем та структур виникають суттєві проблеми, оскільки такі схеми та структури, далеко не завжди використовуються в великих кількостях.

Особливо це стосується створення дослідного зразка для перевірки правильності прийнятих рішень. Розробники апаратури повинні чекати місяці, доки буде виготовлений дослідний зразок. Програмістам це тяжко усвідомити, оскільки «дослідний зразок» для тестування вони отримують практично миттєво – за час, необхідний для компіляції та редагування зав’язків.

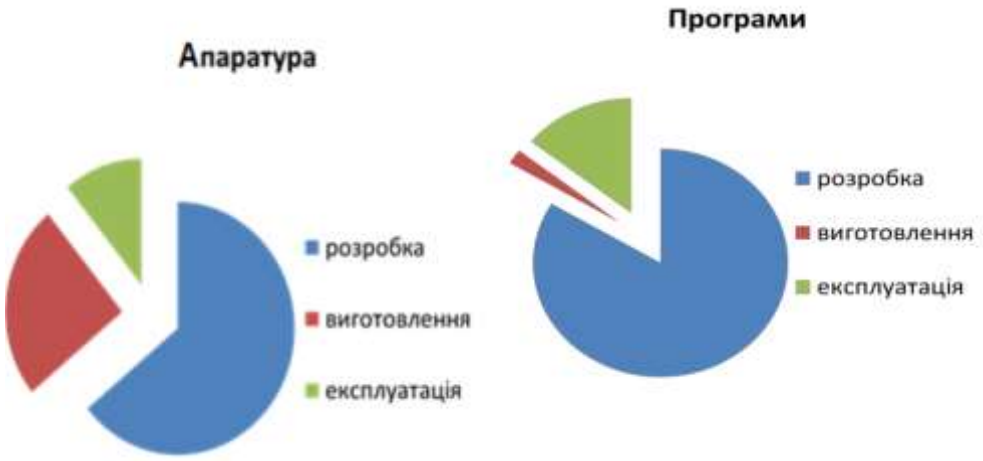


Рисунок 3–Відносні витрати при створенні та експлуатації апаратних та програмних продуктів

Важливим напрямком в вирішенні проблем створення дослідного зразка є використання програмовних логічних інтегральних схем (ПЛІС).

На ПЛІС можна створювати будь-які (у відповідних межах по складності) дослідні зразки мікросхем буквально в лабораторних або домашніх умовах, а також це робиться майже миттєво. Тобто, майже так, як «дослідні зразки» програм. В свою чергу, мікросхеми ПЛІС виготовлюються масовими тиражами і мають малу ціну.

ПЛІС типу Field Programmable Gate Array

ПЛІС типу Field Programmable Gate Array (FPGA) базуються на універсальних логічних елементах, які можуть бути налаштовані на виконання будь якої функції від деякої кількості аргументів.

Приклад функціональної схеми універсального логічного елементу для реалізації будь якої функції від 3 аргументів (рис.4).

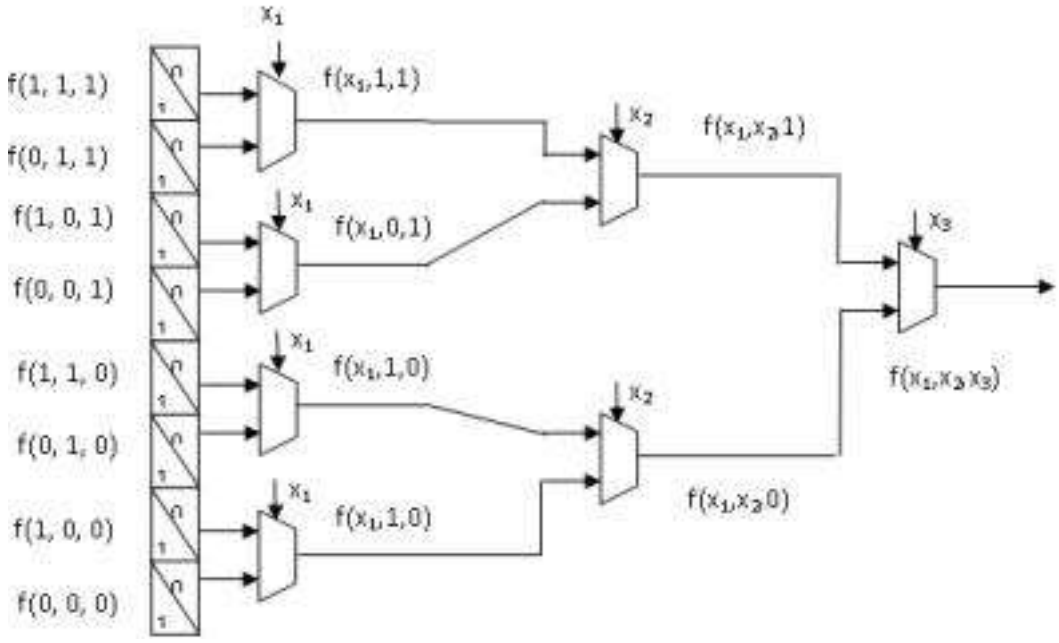


Рисунок 4 Функціональна схема універсального логічного елементу

В схемі використовується регістр розрядністю 2^n (n-кількість змінних, в прикладі n=3), а також мультиплексори налаштування, які реалізуються як показано на рис. 5.

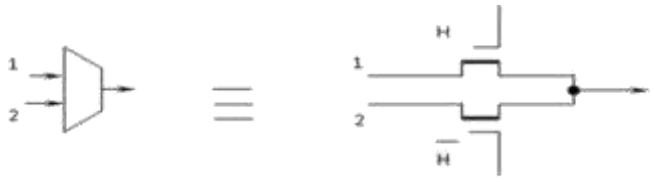
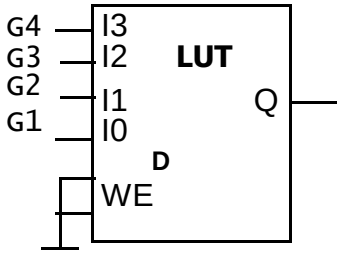


Рисунок 5 – Мультиплексор налаштування

При цьому використовується відомий розклад Шенона:

$$f(x_1,x_2,x_3)=(f(x_1,x_2,1)\ \text{and}\ x_3)\ \text{or}\ (f(x_1,x_2,0)\ \text{and}\ (\text{not}\ x_3)).$$

Основним логічним елементом в ПЛІС є логічна таблиця (ЛТ, англ. look-up table, LUT), що представляє собою регістр зсуву на 16 розрядів (рис. 6). У LUT за адресою G3, G2, G1, G0 записана одиниця, якщо код адреси являє собою мінтерм заданої чотиривходової логічної функції. Наприклад, якщо за адресою 1,1,1,1 записана одиниця, а за рештою адрес – нулі, то LUT реалізує чотиривходову функцію І. На рис.6 показано приклад кодування функції ВИКЛЮЧНЕ АБО за допомогою LUT на чотири входи.



Q:=G4 **xor** G3 **xor** G2 **xor** G1

G4	G3	G2	G1	Q
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Рисунок 6 – Кодування функції ВИКЛЮЧНЕ АБО із чотирма аргументами

Тригери **LUT** входять до складу програмуючого регістру зсуву, і їх початковий стан заповнюється під час конфігурування ПЛІС. Взагалі на поданому **LUT** реалізується будь-яка булева функція від 4 розрядів.

В сучасних ПЛІС використовуються LUT підвищеної ємності, які реалізують будь-які логічні функції від 6 аргументів (або дві будь-які функції від 5 аргументів). Будь-які булеві функції від більшої кількості аргументів реалізуються на ЛТ6 згідно розкладу Шеннона.

Крім того, інколи окремі частини пам’яті блочного ОЗП (BlockRAM) конфігуруються як 10..12-аргументна логічна функція.

Звичайно, проєктування схем в такому елементному базисі безумовно потребує використання спеціальних САПР

Лабораторна робота №1

" САПР WebPACK ISE фірми Xilinx"

Мета роботи: Вивчення особливостей роботи з САПР WebPACK ISE. Створення проєкту та завантаження його на ПЛІС.

Теоретичні відомості

Програмні засоби WebPACK ISE представляють собою систему наскрізного проєктування, яка реалізує всі етапи створення цифрового пристрою на базі ПЛІС, включаючи програмування кристала: розробка проєкту, синтез, моделювання, трасування та завантаження в кристал.

Проєктом у САПР WebPACK ISE називається сукупність модулів (файлів), які містять інформацію, необхідну для виконання всіх етапів процесу розробки цифрового пристрою на базі ПЛІС Xilinx. У його структурі можна виділити наступні групи модулів:

- вихідні описи проєктованого пристрою в графічній або текстовій формі;
- документація, що супроводжує проєкт;
- проміжні результати, які використовуються в якості вихідних даних для подальших кроків проєктування;

- звіти про виконання основних етапів проєктування;
- опису тестових впливів, необхідних для моделювання пристрою;
- остаточні результати проєктування, використовувані для конфігурування ПЛІС.

Всі модулі проєкту розташовуються в одній папці, назва якої збігається з назвою проєкту. Спочатку проєкт представлений тільки заголовком і модулем, в якому вказуються параметри проєкту. У подальшому до нього додають модулі опису проєктованого пристрою, а після виконання кожного етапу процесу розробки пристрою - результати, отримані на цьому етапі, і відповідний звіт. Крім того, розробник може включити в проєкт необхідну текстову документацію.

Створення нового проєкту в середовищі пакету WebPACK ISE

Запустити WebPACK ISE можна через меню «Пуск», або за допомогою посилання на «Робочому столі». Після запуску відкриється робоче середовище WebPACK ISE (див. рис. 1).

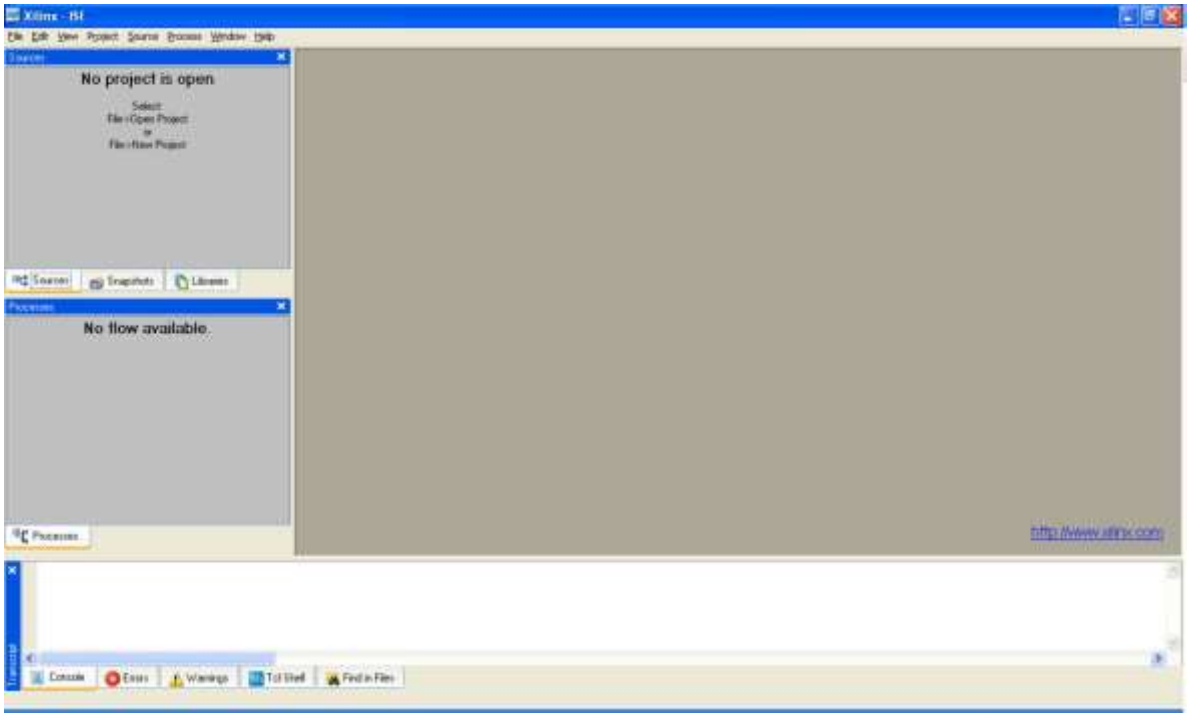


Рис.1 Робоче середовище WebPACK ISE

Для створення нового проєкту слід виконати команду New Project в меню File Навігатора проєкту (див. рис. 2). У результаті відкривається діалогова панель, в якій повинні бути введені вихідні дані для нового проєкту:

- його назва;
- диск і каталог, в якому передбачається його розташувати;
- сімейство ПЛІС, на базі якого розробляється пристрій;
- тип кристала;
- засоби синтезу пристрою.

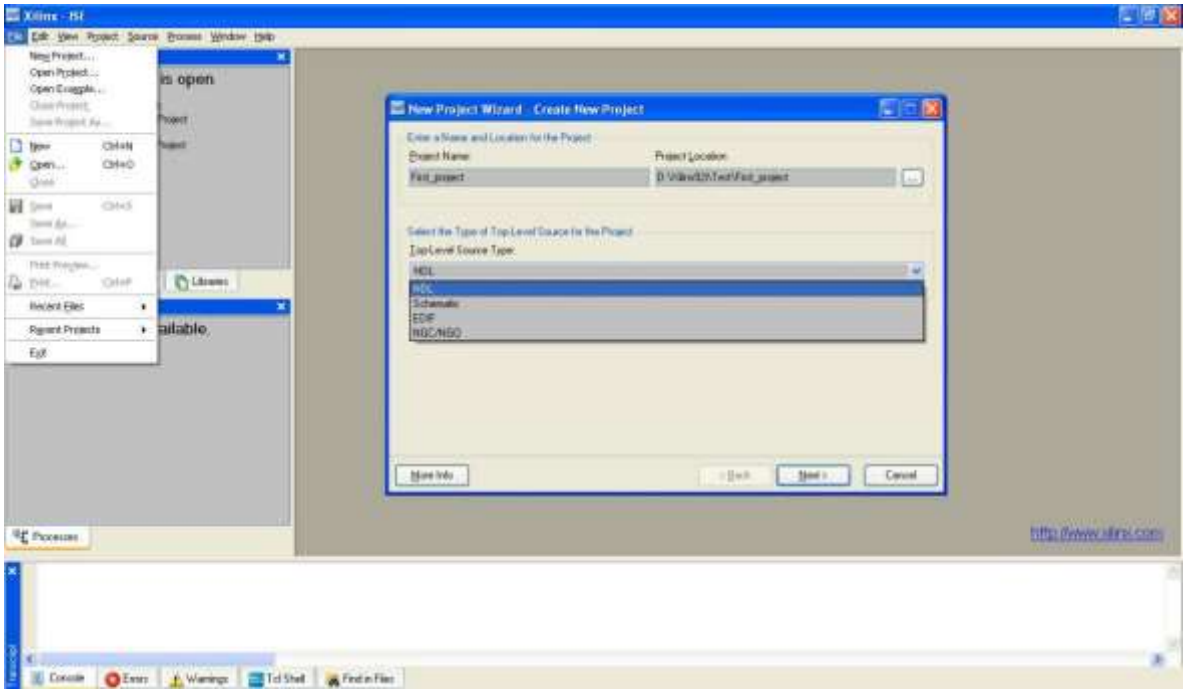


Рис.2 Створення нового проєкту

В першу чергу рекомендується визначити папку, в якій буде розташовуватися робочий каталог проєкту. Його місце на диску вказується в полі редагування Project Location (рис. 2). Доцільно зберігати всі проєкти в спеціально створеному для цих цілей каталозі, наприклад, C:\Project. За замовчуванням у поле редагування Project Location пропонуються диск і каталог, що використовувались у попередньому проєкті. Змінити місце

каталогу створюваного проєкту можна, або використовуючи клавіатуру, або панель навігації по дисках комп'ютера. У першому випадку слід помістити курсор на поле редагування Project Location, клацнути лівою кнопкою миші і ввести з клавіатури ім'я диска і каталогу. Якщо каталог новий, він створюється автоматично. В іншому випадку для вибору існуючого каталогу зручніше натиснути кнопку з піктограмою «...», розташовану праворуч від поля редагування Project Location (див. рис.2) та в відкрилася діалогової панелі навігації за допомогою миші вибрати необхідний диск і каталог, а потім підтвердити зроблений вибір натисненням кнопки ОК. Після закриття панелі навігації вибрані параметри автоматично відображаються в полі редагування Project Location.

Після натискання на кнопку «Next», з'явиться діалогове вікно вибору типу ПЛІС (рис. 3) і деякі параметри проєкту, які потрібно буде вказати системі проєктування для подальшого використання та застосування.

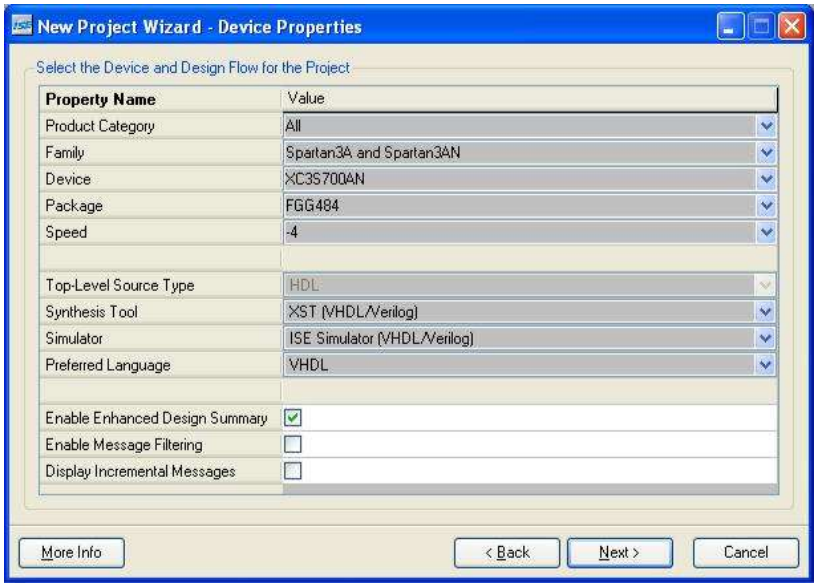


Рис.3 Діалогове вікно вибору типу ПЛІС

У наступному вікні пропонується додати нові джерела проєкту (рис.4), які можуть використовуватися в поточному проєкті. Доки додавати нічого не будемо, натискаємо на кнопку «Next» для продовження.

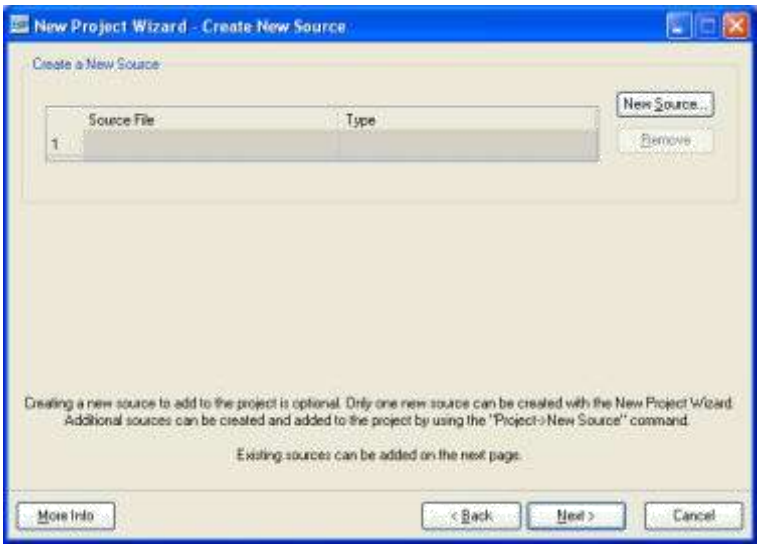


Рис. 4 Додавання джерел проєкту

У результаті одержуємо звіт про виконану роботу і натискаємо на кнопку «Finish» (рис.5).

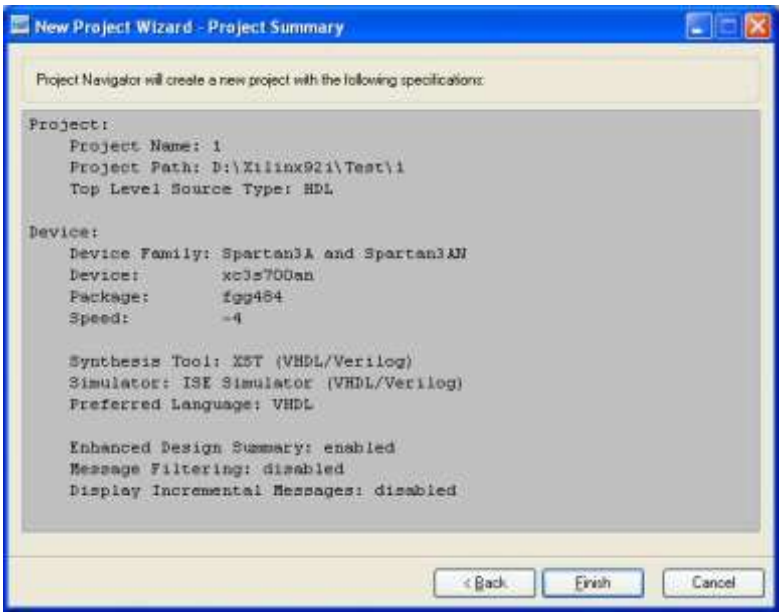


Рис.5 Звіт про створений новий проєкт

Тепер необхідно додати до нашого нового проєкту файл з VHDL – описом нашого пристрою. Для цього необхідно у вікні навігатора обрати новостворений проєкт, натиснути на ньому праву кнопку миші і обрати «New Source» (рис.6). В даному діалоговому вікні, зліва обираємо тип

файлу. В даному випадку обираємо «VHDL Module». Після цього необхідно задати ім’я нашого модулю (рис. 6). Натискаємо кнопку «Next».

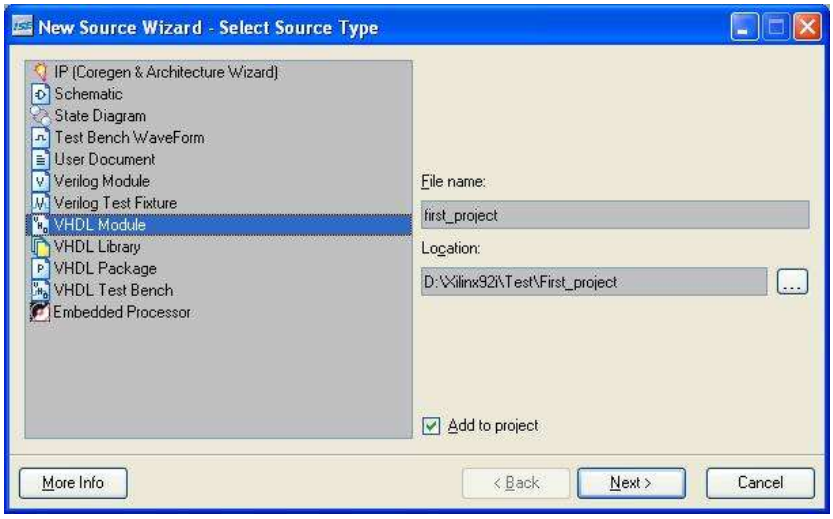


Рис. 6 Додання нового модуля до проєкту

В наступному вікні можна задати входи та виходи нашого проєктованого пристрою. Це також можна зробити і вручну. Тому просто натискаємо кнопку «Next».

Далі перед нами з’являється вікно в якому міститься вся інформація про новостворений модуль (рис. 7). Натискаємо кнопку «Finish».

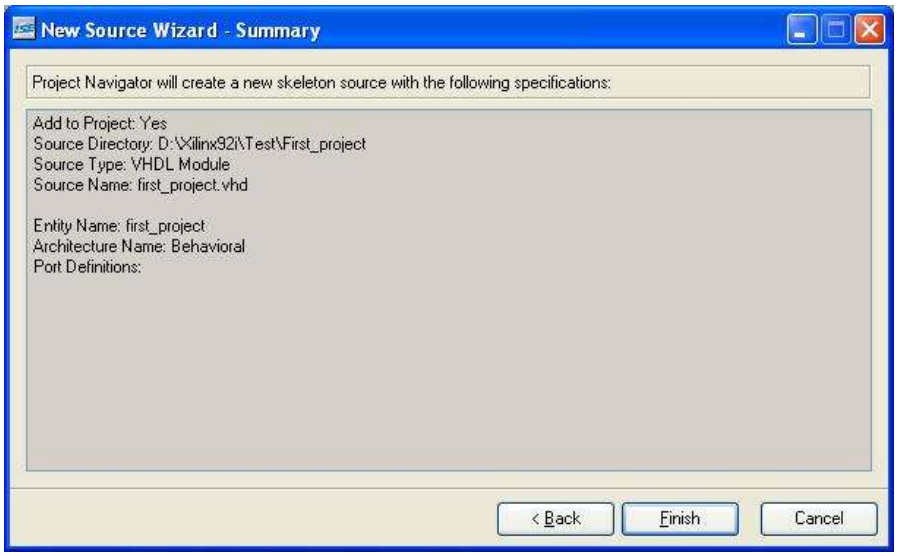


Рис. 7 Звіт про створений новий модуль

Після створення нового модуля, він автоматично відкривається в правій частині програми (рис.8). Тепер можна переходити до опису пристрою.

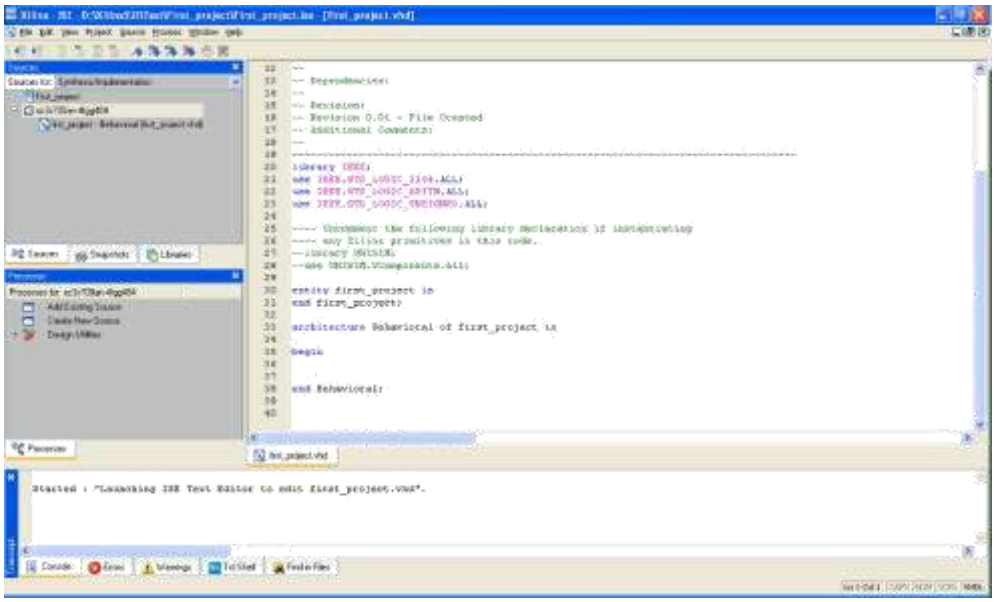


Рис.8 Робоча область пакету WebPACK ISE

В даній роботі ми будемо використовувати частотний генератор, для синхронізації, який вбудований на самій платі Spartan-3AN Starter Kit,

світлодіод та перемикач. Нам необхідно за допомогою лічильника змусити світлодіод «миготіти». За допомогою перемикача ми будемо вмикати та вимикати лічильник, відповідно це відобразиться на світлодіоді.

Правимо новостворений модуль відповідно до опису:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity first_project is
```

```
port (
      clk    : in std_logic;         -- лічильник
      nreset : in std_logic;         -- перемикач
      led1   : out std_logic         -- світлодіод
  );
end first_project;

architecture Behavioral of first_project
is signal led1_i : std_logic; begin

led_flash_proc : process(clk, nreset)
  constant RATIO : integer := 10000000;
  variable count : integer range 0 to RATIO-1; -- регістр який веде підрахунок
  begin
    if nreset='0' then -- перемикач "1" -> світлодіод працює, "0" -> не працюэ
      led1_i <= '0';        -- вимикаємо діод
      count := 0;
    elsif rising_edge(clk) then
      if count=RATIO-1 then -- лічильник досягнув заданого значення
        count := 0;       -- перезавантаження лічильника
        led1_i <= not led1_i;       -- перемикання світлодіоду
      else
        count := count + 1;
      end if;
    end if;
  end process;
  led1 <= led1_i;
```

end Behavioral;

Після того як файл відредаговано, необхідно зв’язати наші входи ти виходи нашого пристрою з входами та виходами плати. Для того, щоб знати які виходи ПЛІС відповідають за перемикач, частотний генератор та світлодіод, необхідно детально ознайомитися з документацією по інструментальному пакету Spartan-3AN Starter Kit.

В лівій частині програмного комплексу WebPACK ISE у розділі «Sources» обираємо наш створений файл. В розділі «Processes» розкриваємо пункт меню «User Constraints» та двічі натискаємо мишею по кнопці «Assign Package Pins» (рис. 9).

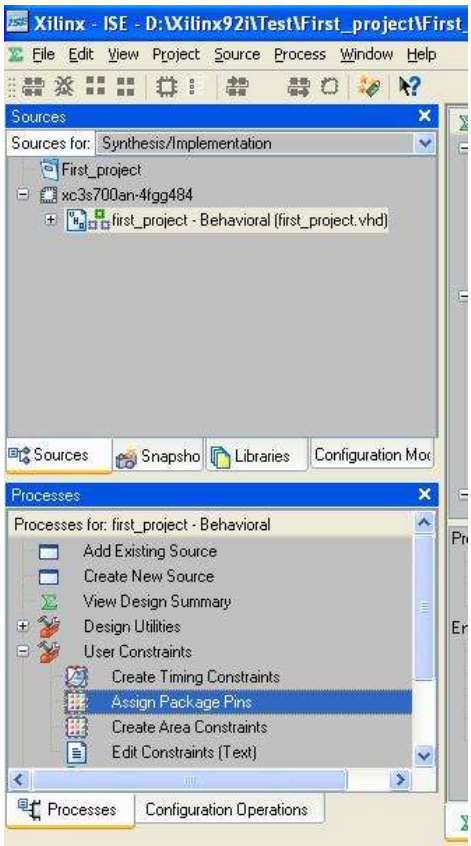


Рис. 9 Призначення виходів ПЛІС

Після чого відкриється вікно, в якому зліва відображаються входи та виходи нашого проєктованого пристрою. В правій частині ми бачимо кристал плати

Spartan-3AN Starter Kit виконаний у вигляді матриці. Простим перетягуванням відповідних входів та виходів на матрицю кристалу відбувається поєднання. Після чого відповідна комірка матриці перефарбовується в синій колір (рис 10).

Після встановлення відповідності входів та виходів нашого проєктованого пристрою необхідно задати параметри функціонування входів та виходів. В розділі «Processes» обираємо IOs, як зображено на рисунку 11. В таблиці бачимо входи та виходи нашого пристрою. Навпроти кожного входу та виходу необхідно в розділі «IOSTANDART» задати параметр «LVCMOS33» (рис. 11)

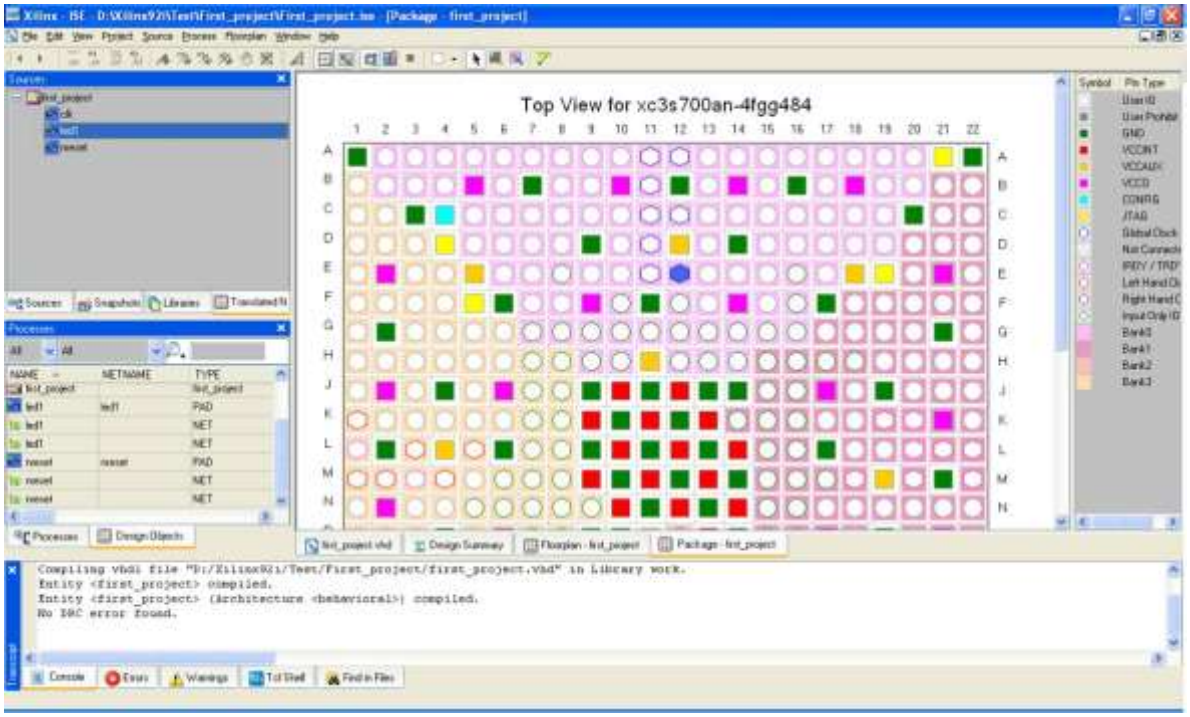


Рис.10 Встановлення відповідності входів та виходів

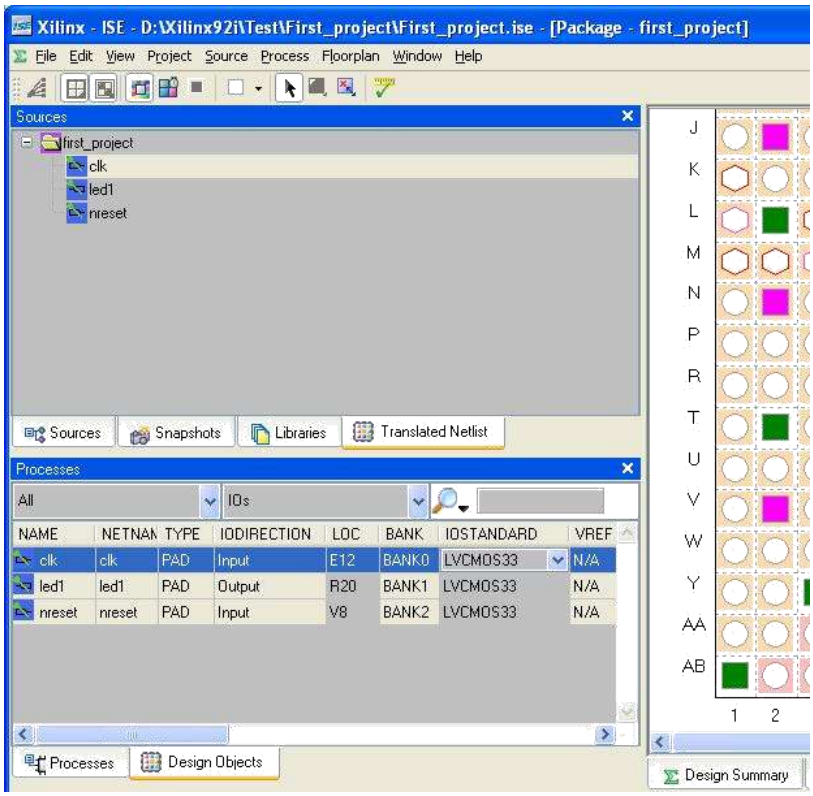


Рис.11 Задання параметрів

Після встановлення всіх відповідностей необхідно зберегти проєкт. Це можна зробити натисненням клавіш «Ctrl+S», або в меню File обрати «Save». Після збереження можна закрити вікно встановлення відповідностей.

Тепер можна переходити до наступного етапу – компіляції проєкту. Для того щоб скомпілювати проєкт необхідно в розділі «Sources» обрати файл з проєктом. У вкладці «Processes» двічі натиснути на розділ «Generate Programming File» (рис.12).

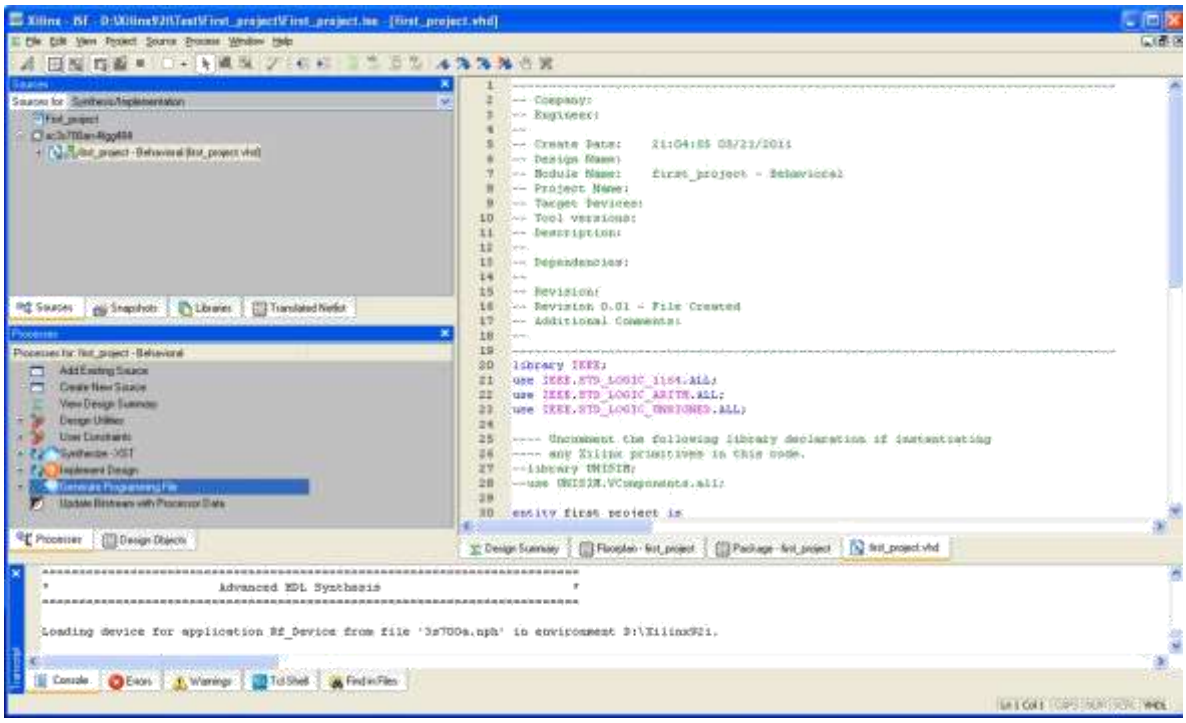


Рис.12 Компіляція проєкту

Після вдалої компіляції з’явиться діалогове вікно «Xilinx WebTalk Dialog», в якому необхідно натиснути кнопку «Decline…». В наступному діалоговому вікні обираємо «Disable the collection of device usage statistics for this project only» і натискаємо кнопку «OK» (рис.13).

Рис.13 Xilinx WebTalk Dialog

Після вдалої компіляції (рис.14) можна переходити до завантаження нашого проєкту на ПЛІС.

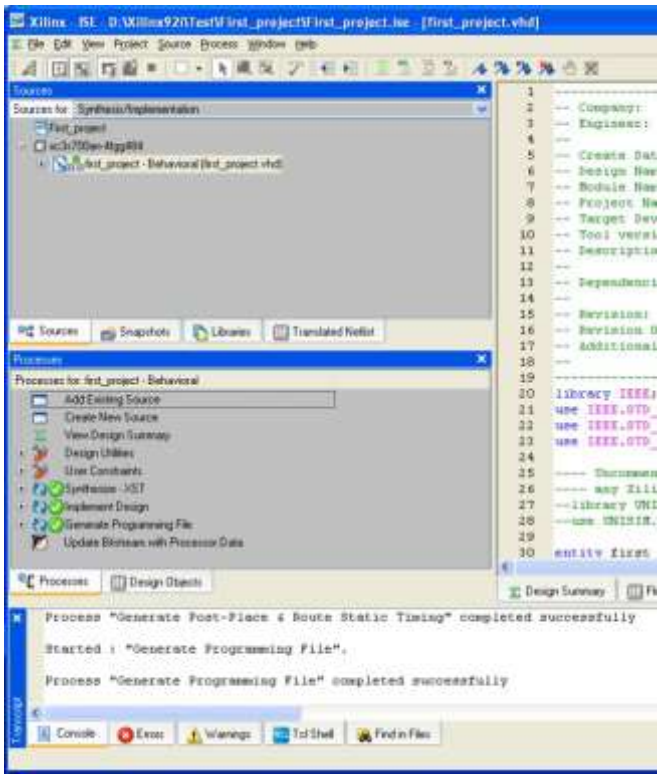


Рис.14. Вдале закінчення компіляції проєкту

Спочатку необхідно під’єднати плату Spartan-3AN до комп’ютера за допомогою USB – кабелю, та ввімкнути живлення плати. Далі в розділі «Processes» розкриваємо пункт меню «Generate Programming File», натиснувши на «+», а далі подвійним натисненням на меню «Configure Device (iMPACT)» запускаємо генерацію bit – файлу для завантаження на ПЛІС. Після генерації bit-файлу ми бачимо вікно яке зображене на рис. 15. Натискаємо кнопку «Finish» (рис. 15). Перед нами відкривається вікно для завантаження нашого проєкту на ПЛІС (рис.16).

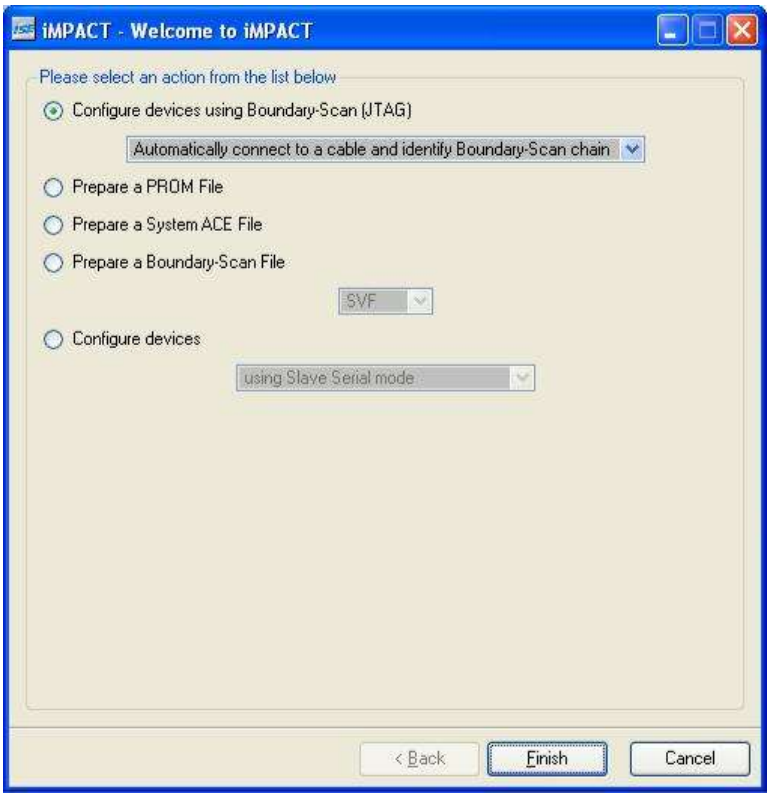


Рис.15 Закінчення генерації bit-файлу

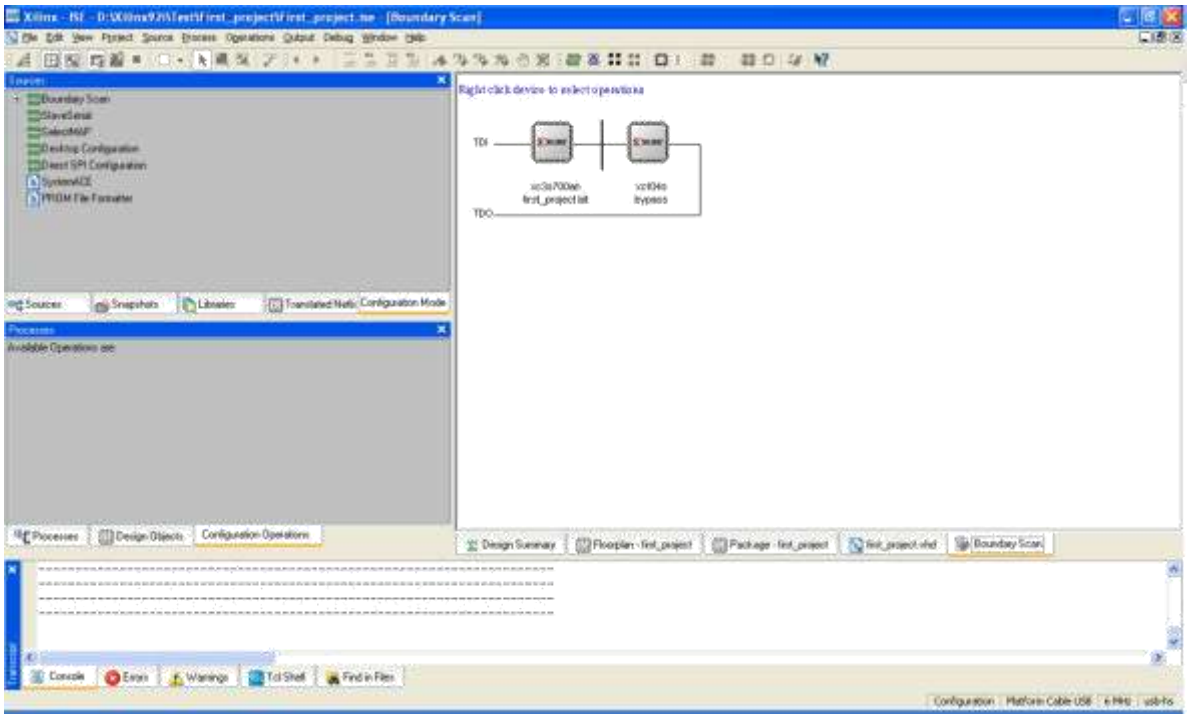


Рис.16. Підготовка для завантаження bit – файлу на ПЛІС

На даному етапі необхідно вказати наш згенерований bit – файл для xc3s700an, а для xcf04s – натиснути кнопку «Bypass».

Все готово для завантаження проєкту на ПЛІС. Натискаємо праву кнопку миші на значку xc3s700an і в контекстному вікні обираємо позицію «Program FPGA Only» (рис. 17).

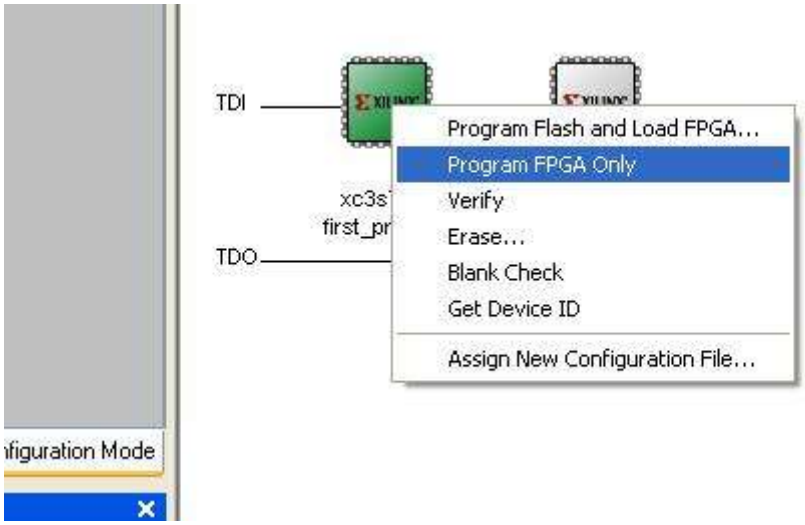


Рис.17. Завантаження проєкту на ПЛІС

Якщо Ви все правильно зробили, то після переведення перемикача в позицію «1» обраний Вами світлодіод буде миготіти. Після переведення перемикача в позицію «0», миготіння припиниться.

Завдання для роботи:

1. Ознайомитися з додатковим матеріалом по платі Spartan-3AN Starter Kit
2. Виконати самостійно всі дії, які описані в теоретичних відомостях.

Контрольні запитання

1. Що називається проєктом у САПР WebPACK ISE ?
2. Як відбувається створення проєкту у WebPACK ISE ?
3. Призначення та характеристики відлагоджувальної плати Spartan-3AN Starter Kit Board .

Лабораторна робота №2
"Робота з рідкокристалічним дисплеєм"

Мета роботи: Навчитися працювати з вбудованим рідкокристалічним дисплеєм плати Spartan-3AN Starter Kit. Закріпити отримані знання на практиці.

Теоретичні відомості

Рідкокристалічний дисплей, який використовують в складі Spartan-3AN Starter Kit, призначений для відображення алфавітно-цифрової інформації в розроблювальній вбудованій мікропроцесорній системі. Даний дисплей дозволяє відображати як стандартні символи таблиці ASCII, так і символи, що формуються розробником. Алфавітно-цифрова інформація може виводитися у вигляді двох рядків, кожна з яких містить до 16 символів. У складі рідкокристалічного дисплея використовується вбудований контролер Sitronix ST7066U, який повністю сумісний з Samsung S6A0069X, Samsung KS0066U, Hitachi HD44780 і SMOS SED1278. Інформація, яка виводиться на дисплей, може передаватися з боку ПЛІС за допомогою 4-розрядної або 8-розрядної шини даних. Схема підключення рідкокристалічного дисплею до виходів кристала XC3S700AN показана на рис. 18.

Існує два інтерфейси контролера РК-дисплея, один 8 - ми бітний та 4 - х бітний. Визначення контактів будуть використовуватися в "UCF" файлі. Якщо LCD_RW буде встановлений в «0», то це означає що додаток не зможе зчитувати дані з екрану. Встановлення LCD_E в низький рівень буде означати, що РК – дисплей буде ігнорувати всі входи. Високий рівень LCD_RS дозволяє запис, в той час як низький рівень LCD_RS вказує команду яка буде виконуватися.

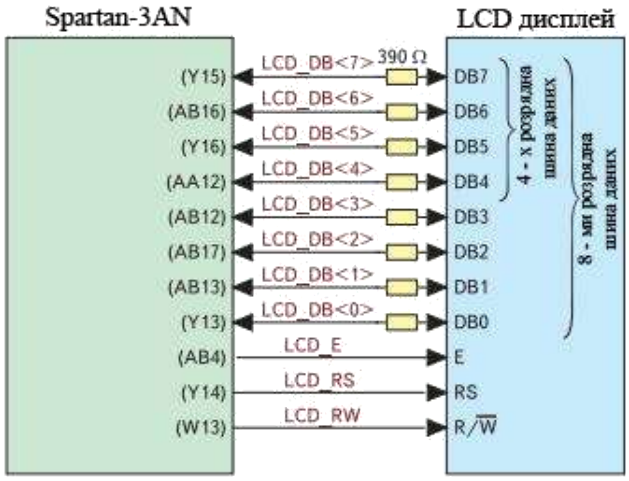


Рис.18 Схема підключення рідкокристалічного дисплею до виходів кристала XC3S700AN

Приклад UCF – файлу:

NET "LCD_E" LOC = "AB4" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;

NET "LCD_RS" LOC = "Y14" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;

NET "LCD_RW" LOC = "W13" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;

NET "LCD_DB<7>" LOC = "Y15" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;

NET "LCD_DB<6>" LOC = "AB16" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;

NET "LCD_DB<5>" LOC = "Y16" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;

NET "LCD_DB<4>" LOC = "AA12" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;

NET "LCD_DB<3>" LOC = "AB12" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;

NET "LCD_DB<2>" LOC = "AB17" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;

NET "LCD_DB<1>" LOC = "AB18" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;

NET "LCD_DB<0>" LOC = "Y13" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;

LCD екран має три внутрішніх області пам'яті. Дані зчитуються з RAM, які будуть відображатися на екрані. Character Generator оперативної пам'яті (RAM CG), в якій зберігаються шаблони і Character Generator ROM (CG ROM), який включає в себе ряд шаблонів, які відповідають ASCII символам. Значення в CG-ROM, відображені на рисунку 19 повинні бути записані у DD-RAM. Наприклад, символ "S" від CG-ROM буде мати значення "01010011".

Рис. 19 Набір РК-символів

Ініціалізація, конфігурація та відображення

Існують три основні етапи під час роботи з рідкокристалічним дисплеєм. Перший - ініціалізації з чотирьох бітного інтерфейсу (або 8 – ми бітного), другий – визначення команди для встановлення параметрів та відображення, і третій - запис даних в рідкокристалічний дисплей. Більш докладно про ці етапи можна прочитати в розділі 5 електронного документу «Spartan-3A/3AN FPGA Starter Kit Board User Guide» (див. список літератури).

Приклад роботи з рідкокристалічним екраном

В даному прикладі на рідкокристалічний екран тестової плати Spartan – 3AN Starter Kit буде виведено напис «Hello World».

```
module lcd (clk, lcd_rs, lcd_rw, lcd_e, lcd_0, lcd_1, lcd_2, lcd_3, lcd_4, lcd_5,
lcd_6, lcd_7);
            parameter      k = 18;
 (* LOC="E12" *)   input         clk;
            reg  [k+8-1:0] count=0;
            reg          lcd_busy=1;
            reg          lcd_stb;
            reg       [5:0] lcd_code;
            reg       [6:0] lcd_stuff;
 (* LOC="Y14" *)   output reg     lcd_rs;
 (* LOC="W13" *)    output reg     lcd_rw;
 (* LOC="Y15" *)   output reg     lcd_7;
 (* LOC="AB16" *) output reg      lcd_6;
 (* LOC="Y16" *)   output reg     lcd_5;
 (* LOC="AA12" *) output reg      lcd_4;
 (* LOC="AB12" *) output reg      lcd_3;
```

```
(* LOC="AB17" *) output reg       lcd_2;
(* LOC="AB18" *) output reg       lcd_1;
(* LOC="Y13" *)  output reg       lcd_0;
(* LOC="AB4" *)  output reg       lcd_e;

always @ (posedge clk) begin
  count <= count + 1;
  case (count[k+7:k+2])
     0: lcd_code <= 6'h03;// ініціалізація живлення
     1: lcd_code <= 6'h03;
     2: lcd_code <= 6'h03;
     3: lcd_code <= 6'h02;
     4: lcd_code <= 6'h02;
     5: lcd_code <= 6'h08;
     6: lcd_code <= 6'h00;// встановлення режиму запсу
     7: lcd_code <= 6'h06;
     8: lcd_code <= 6'h00;// вмикання/вимикання дисплею
     9: lcd_code <= 6'h0C;
    10: lcd_code <= 6'h00;         // очистка дисплею
    11: lcd_code <= 6'h01;
    12: lcd_code <= 6'h24;         // H
    13: lcd_code <= 6'h28;
    14: lcd_code <= 6'h26;         // e
    15: lcd_code <= 6'h25;
    16: lcd_code <= 6'h26;         // l
    17: lcd_code <= 6'h2C;
    18: lcd_code <= 6'h26;         // l
```

```
      19: lcd_code <= 6'h2C;
      20: lcd_code <= 6'h26;       // o
      21: lcd_code <= 6'h2F;
      22: lcd_code <= 6'h22;       //
      23: lcd_code <= 6'h20;
      24: lcd_code <= 6'h25;       // W
      25: lcd_code <= 6'h27;
      26: lcd_code <= 6'h26;       // o
      27: lcd_code <= 6'h2F;
      28: lcd_code <= 6'h27;       // r
      29: lcd_code <= 6'h22;
      30: lcd_code <= 6'h26;       // l
      31: lcd_code <= 6'h2C;
      32: lcd_code <= 6'h26;       // d
      33: lcd_code <= 6'h24;
      34: lcd_code <= 6'h22;       // !
      35: lcd_code <= 6'h21;
      default: lcd_code <= 6'h10;
    endcase
    lcd_stb <= ^count[k+1:k+0] & ~lcd_rw & lcd_busy;
    lcd_stuff <= {lcd_stb,lcd_code};
    {lcd_e,lcd_rs,lcd_rw,lcd_7,lcd_6,lcd_5,lcd_4} <= lcd_stuff;
    {lcd_3,lcd_2,lcd_1,lcd_0} <= 4'b1111;
  end
endmodule
```

Завдання на роботу:

1. Ознайомитися з розділом 5документу «Spartan-3A/3AN FPGA Starter Kit Board User Guide».
2. Скомпілювати та завантажити на ПЛІС представлений приклад.
3. Перевірити роботу рідкокристалічного дисплею.
4. Вивести на РК – дисплей своє прізвище та ініціали.

Контрольні запитання

1. В чому полягає призначення рідкокристалічного дисплею в Spartan-3A/3AN FPGA Starter Kit Board?

2. Яке призначення внутрішніх області пам'яті LCD екрану?

3. Назвіть та охарактеризуйте основні етапи роботи з рідкокристалічним дисплеєм?

Лабораторна робота №3

"Робота з портом VGA "

Мета роботи: Ознайомитися з роботою VGA – порту. Навчитися програмувати VGA – порт інструментального пакету Spartan-3AN Starter Kit.

Теоретичні відомості

Інструментальний комплект Spartan-3AN FPGA Starter Kit містить VGA – порт, який виконаний у вигляді стандартного роз'єму HD-DB15. За допомогою цього порту можна підключати більшість CRT та LCD моніторів, які містять стандартний VGA – кабель. VGA – порт розміщується на платі як показано на рис. 20.

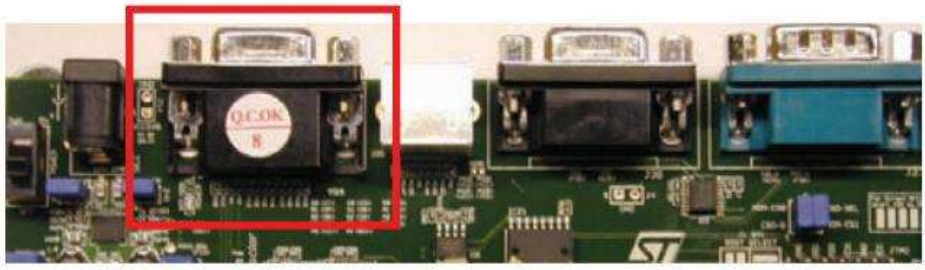


Рис.20 Розміщення порту на платі Spartan-3AN Starter Kit

Схема підключення VGA – порту до тестової плати Spartan – 3AN Starter Kit зображена на рис. 21. FPGA безпосередньо управляє п'ятьма сигналами VGA через резистори. Кожен червоний, зелений і блакитний сигнали мають чотири виходи від FPGA, які подаються через дерево резисторів. Цей підхід забезпечує 4-розрядну роздільну здатність, генеруючи 12-розрядний колір, або 4096 можливих кольори. Послідовний резистор, в комбінації з 75 Ом резистором вбудованим в VGA кабель, гарантує, що кольорові сигнали залишаються в VGA діапазоні 0V до 0.7V. Управління VGA_R[3:0], VGA_[3:0], і VGA_B[3:0] сигналами (високий, низький рівень), забезпечує генерування бажаного кольору. Кожен індивідуальний кольорний вихід підтримує 16 можливих значень, як описано в формулі :

$$COLOR_{OUT} = \frac{VGA[3:0]}{15} \times COLOR .$$

Три окремих значення для червоного кольору, зеленого і синього підтримують максимум 12 - розрядний колір, або 4096 можливих значень кольорів.

Таблиця формування кольорів зображена на рисунку 22.

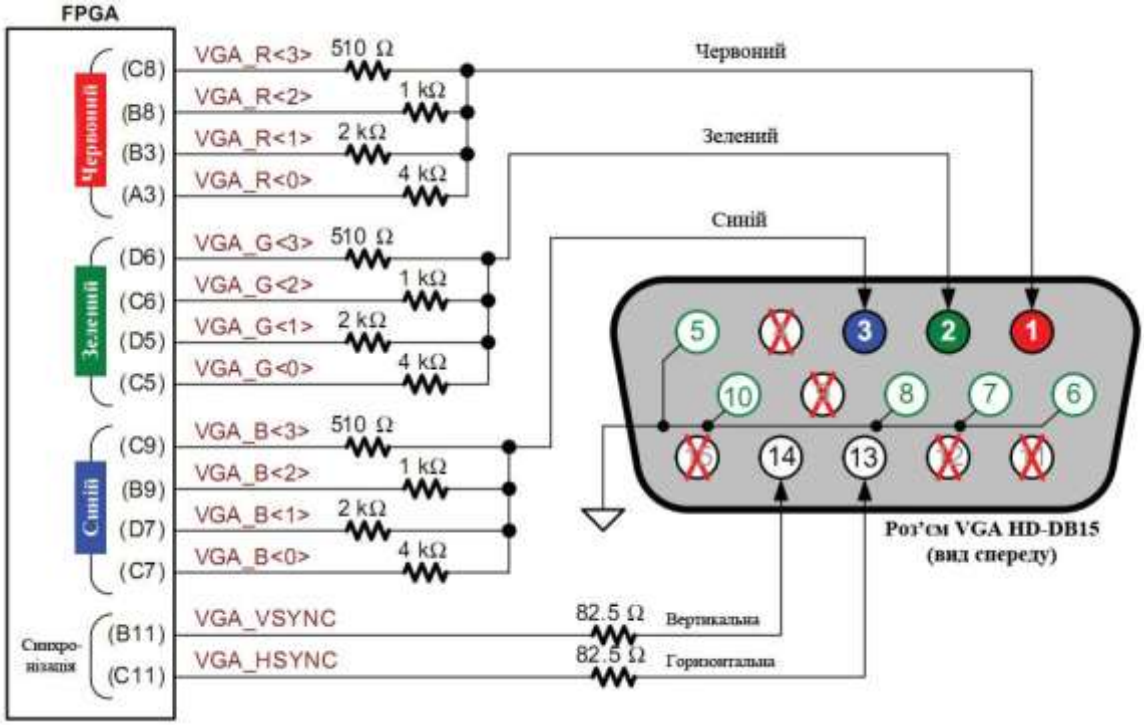


Рис.21 Схема підключення VGA порту до ПЛІС

VGA_R[3:0]	VGA_G[3:0]	VGA_B[4:0]	Колір
0000	0000	0000	Чорний
0000	0000	1111	Синій
0000	1111	0000	Зелений
0000	1111	1111	Блакитний
1111	0000	0000	Червоний
1111	0000	1111	Пурпуровий
1111	1111	0000	Жовтий
1111	1111	1111	Білий

Рис 22 Формування кольорів

Формування VGA сигналу дуже докладно описано в технічній документації «Spartan-3A/3AN FPGA Starter Kit Board User Guide», розділ 6.

Приклад UCF – файлу:

```
NET "VGA_R<3>" LOC = "C8" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;

NET "VGA_R<2>" LOC = "B8" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;

NET "VGA_R<1>" LOC = "B3" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;

NET "VGA_R<0>" LOC = "A3" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;

NET "VGA_G<3>" LOC = "D6" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;

NET "VGA_G<2>" LOC = "C6" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;

NET "VGA_G<1>" LOC = "D5" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;

NET "VGA_G<0>" LOC = "C5" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;

NET "VGA_B<3>" LOC = "C9" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;

NET "VGA_B<2>" LOC = "B9" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;

NET "VGA_B<1>" LOC = "D7" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;
```

```
NET "VGA_B<0>" LOC = "C7" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW
= FAST ;

NET "VGA_HSYNC" LOC = "C11" | IOSTANDARD = LVCMOS33 | DRIVE = 8
| SLEW = FAST ;

NET "VGA_VSYNC" LOC = "B11" | IOSTANDARD = LVCMOS33 | DRIVE = 8
| SLEW = FAST ;
```

Приклад роботи з VGA – портом.

Простий приклад, в якому буде реалізовано взаємодія з VGA – портом. Зафарбовування екрану червоним кольором.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity vga is
    port(clk50_in : in std_logic;
    red_out  : out std_logic;
    green_out : out std_logic;
    blue_out : out std_logic;
    hs_out    : out std_logic;
    vs_out    : out std_logic);
end vga;

architecture Behavioral of vga is

signal clk25            : std_logic;
signal horizontal_counter : std_logic_vector (9 downto 0);
```

```
signal vertical_counter  : std_logic_vector (9 downto 0);


begin
 -- generate a 25Mhz clock
        process (clk50_in)
        begin
                if clk50_in'event and clk50_in='1'
                        then if (clk25 = '0') then
                        clk25 <= '1';
                        else
      clk25 <= '0';
                        end if;
                end if;
        end process;


  process (clk25)
  begin
   if clk25'event and clk25 = '1' then
     if (horizontal_counter >= "0010010000" ) -- 144
                        and (horizontal_counter < "1100010000" ) -- 784
                        and (vertical_counter >= "0000100111" ) -- 39 and
                        (vertical_counter < "1000000111" ) -- 519
                                then
                                        --here you paint!!
                                        red_out <= '1';
                                        green_out <= '0';
                                        blue_out <= '0';
```

```
                    else
                        red_out <= '0';
                        green_out <= '0';
                        blue_out <= '0';
end if;
 if (horizontal_counter > "0000000000" )
                    and (horizontal_counter < "0001100001" ) --
                    96+1 then
                        hs_out <= '0';
                    else
                        hs_out <= '1';
 end if;
 if (vertical_counter > "0000000000" )
                    and (vertical_counter < "0000000011" ) --
                    2+1 then
                        vs_out <= '0';
                    else
                        vs_out <= '1';
 end if;

          horizontal_counter <= horizontal_counter+"0000000001";

 if (horizontal_counter="1100100000") then
  vertical_counter <= vertical_counter+"0000000001";
  horizontal_counter <= "0000000000";
 end if;
```

```
      if (vertical_counter="1000001001")
      then vertical_counter <= "0000000000";
      end if;
    end if;
  end process;

  end Behavioral;
```

В наступному прикладі представлено вивід квадратів зі стороною 8 пікселів. В середині кожного квадрату містяться інші різнокольорові квадрати.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity square is
  port(clk50_in : in std_logic;
       red_out  : out std_logic;
       green_out : out std_logic;
       blue_out : out std_logic;
       hs_out    : out std_logic;
       vs_out    : out std_logic);
end square;
architecture Behavioral of square is
signal clk25             : std_logic; -- the 25Mhz clock
signal horizontal_counter : std_logic_vector (9 downto
0); signal vertical_counter : std_logic_vector (9 downto
0); begin
```

```
-- generate a 25Mhz
clock process (clk50_in)
begin
 if clk50_in'event and clk50_in='1' then
  if (clk25 = '0') then
   clk25 <= '1';
  else
   clk25 <= '0';
  end if;
 end if;
end process;

process (clk25)
begin
 if clk25'event and clk25 = '1' then
  if (horizontal_counter >= "0010010000" ) -- 144
  and (horizontal_counter < "1100010000" ) --
  784 and (vertical_counter >= "0000100111" ) --
  39 and (vertical_counter < "1000000111" ) --
  519 then
   red_out <= horizontal_counter(3)
              and vertical_counter(3);
   green_out <= horizontal_counter(4)
              and vertical_counter(4);
   blue_out <= horizontal_counter(5)
              and vertical_counter(5);
  else
```

```
  red_out <= '0';
  green_out <= '0';
  blue_out <= '0';
 end if;
 if (horizontal_counter > "0000000000" )
  and (horizontal_counter < "0001100001" ) -- 96+1
 then
  hs_out <= '0';
 else
  hs_out <= '1';
 end if;
 if (vertical_counter > "0000000000" )
  and (vertical_counter < "0000000011" ) -- 2+1
 then
  vs_out <= '0';
 else
  vs_out <= '1';
 end if;
 horizontal_counter <= horizontal_counter+"0000000001";
 if (horizontal_counter="1100100000") then
  vertical_counter <= vertical_counter+"0000000001";
  horizontal_counter <= "0000000000";
 end if;
 if (vertical_counter="1000001001")
 then vertical_counter <= "0000000000";
 end if;
end if;
```

```
end process;

end Behavioral;
```

Завдання на роботу:

1. Ознайомитися з технічною документацією по роботі з VGA – портом ("Spartan-3A/3AN FPGA Starter Kit Board User Guide", розділ 6).

2. Скомпілювати та завантажити представлені приклади на ПЛІС. Переконатися в правильності роботи.

Контрольні запитання

1. Яке призначення VGA – порта ?
2. Яким чином адаптером формуються кольори?
3. Яке призначення послідовних резисторів?

Лабораторна робота №4

"Генерування звуку за допомогою ПЛІС"

Мета роботи: Навчитися генерувати звукові хвилі за допомогою ПЛІС. Перевірка правильності роботи програми на тестовій платі Spartan-3AN Starter Kit.

Теоретичні відомості

В даній лабораторній роботі ви навчитеся генерувати звукові тони. Спочатку будуть розглянуті прості приклади генерації одно тонових звуків. Далі будуть розглядатися більш складніші приклади.

Звуковий вихід розташований на тестовій платі Spartan-3AN як показано на рис. 23

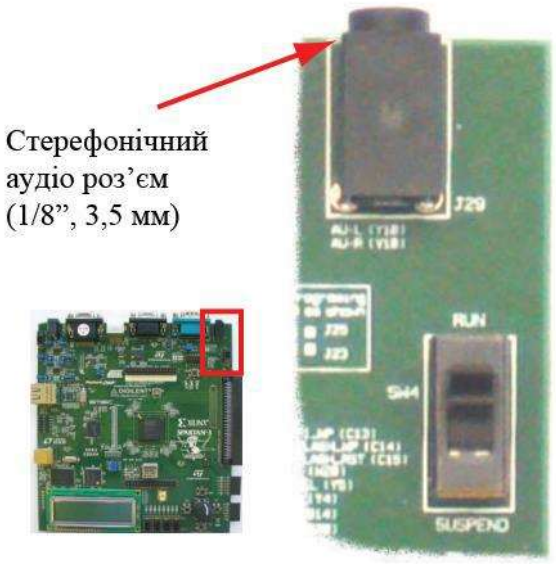


Рис. 23 Звуковий вихід Spartan-3AN Starter Kit Приклад UCF – файлу:

```
NET "AUD_L" LOC = "Y10" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;

NET "AUD_R" LOC = "V10" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;
```

Генерація звуку відбувається за рахунок зміни частоти входу/виходу.

Розглянемо простий приклад, в результаті виконання якого ми повинні почути одно тоновий сигнал.

За допомого ПЛІС можна легко реалізовувати двійкові лічильники. Почнемо з 16 – ти бітного лічильника. За допомого 25 МГц лічильника можна поділити частоту вдвічі, використовуючи при цьому 16 – ти бітний лічильник, який рахує від 0 до 65536 (65536 різних значень). Перемикання лічильника буде відбуватися на частоті 25000000/65536 = 381 Гц.

Однотоновий сигнал:

```
module music(clk, speaker);
input clk;
output speaker;
// 16 – ти бітний лічильник
reg [15:0] counter;
always @(posedge clk) counter <= counter+1;
// при високому сигналі відбувається вивід однотонового звуку
assign speaker = counter[15];
endmodule
```

Тепер давайте згенеруємо двох тоновий сигнал. Зобразимо сирену швидкої допомоги.

```
module music(clk, speaker);
input clk;
output speaker;
parameter clkdivider =
25000000/440/2; reg [23:0] tone;
always @(posedge clk) tone <= tone+1;
reg [14:0] counter;
```

```
always @(posedge clk) if(counter==0) counter <= (tone[23] ? clkdivider-1 :
clkdivider/2-1); else counter <= counter-1;
reg speaker;
always @(posedge clk) if(counter==0) speaker <=
~speaker; endmodule
```

Тепер нам потрібно згенерувати звук, що звучить як поліцейська сирена.

Використаємо перший приклад простого однотонового звуку. Ми використовуємо тільки 23 біта, щоб зробити його в два рази швидше (MSB перемикає на рівні близько 3 Гц). Ми витягуємо біт 15 до 21 біт в тон лічильника, так: тон [21:15]. Це дає нам 7 біт, які йдуть від 0 до 127 з деякою середньою швидкістю. Після того, як лічильник досягає 127 він іде до 0.

Для того, що зробити іншу частину звуку поліцейської сирени, ми використаємо тон [21:15]. Це дає нам 7 біт знову ж таки, які йдуть з 127 до 0.

Для перемикання між двома сигналами, ми використовуємо тон [22]. Як тільки перший лічильник сигналу дійшов до 127, ми переходимо до другого сигналу, поки лічильник не переходить в 0, а потім повертаємося до першого сигналу.

```
module music(clk, speaker);
input clk;
output speaker;

reg [22:0] tone;
always @(posedge clk) tone <= tone+1;
wire [6:0] ramp = (tone[22] ? tone[21:15] :
~tone[21:15]); wire [14:0] clkdivider = {2'b01, ramp,
6'b000000}; reg [14:0] counter;
```

```
always @(posedge clk) if(counter==0) counter <= clkdivider; else counter <=
counter-1;
reg speaker;
always @(posedge clk) if(counter==0) speaker <=
~speaker; endmodule
```

Завдання на роботу:

1. Ознайомитися з технічною документацією по роботі зі звуковим виходом ("Spartan-3A/3AN FPGA Starter Kit Board User Guide", розділ 16).
2. Скомпілювати та завантажити представлені приклади на ПЛІС. Переконатися в правильності роботи.

Контрольні запитання

1. Яким чином можна реалізувати двійковий лічильник на ПЛІС?
2. Назвати основні часові характеристики лічильників.
3. Сформулювати методику синтезу синхронних лічильників з неприроднім порядком лічби.
4. Нарисувати узагальнену структуру кільцевого лічильника.

Лабораторна робота №5
«Робота з VGA, PS/2 портами та звуком»

Мета роботи: Навчитися використовувати декілька периферійних пристроїв одночасно.

Теоретичні відомості

Дана робота являється комплексною, тому що в ній буде розглядатися робота декількох периферійних пристроїв. Роботу зі звуком та VGA – портом ми розглядали в попередніх роботах. В даній роботі ми ще розглянемо роботу з портом PS/2 для підключення клавіатури або миші. Всі ці знання ми закріпимо на прикладі гри «пінг - понг», в якій ми використаємо VGA та PS/2 порти, а також вивід звуку.

Робота з портом PS/2.

Роз'єм інтерфеса PS/2 дозволяє підключати одночасно два пристрої, наприклад клавіатуру та мишу. Для цього необхідно використовувати відповідний роздвоювач. PS/2 порт розташовується на платі, як показано на рис.24. Схема підключення контактів PS/2 зображена на рис. 25.

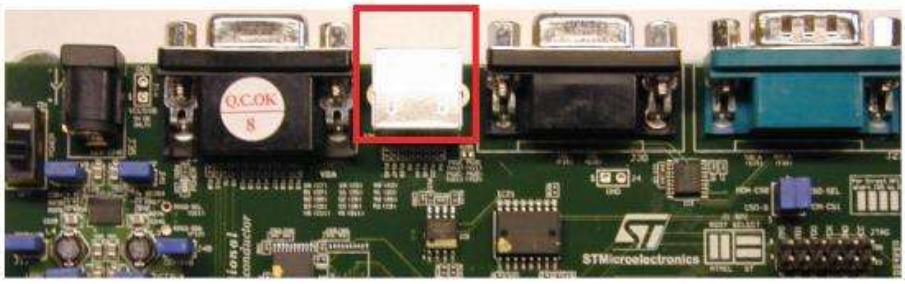


Рис.24 Розміщення порту PS/2 на платі Spartan-3AN Starter Kit

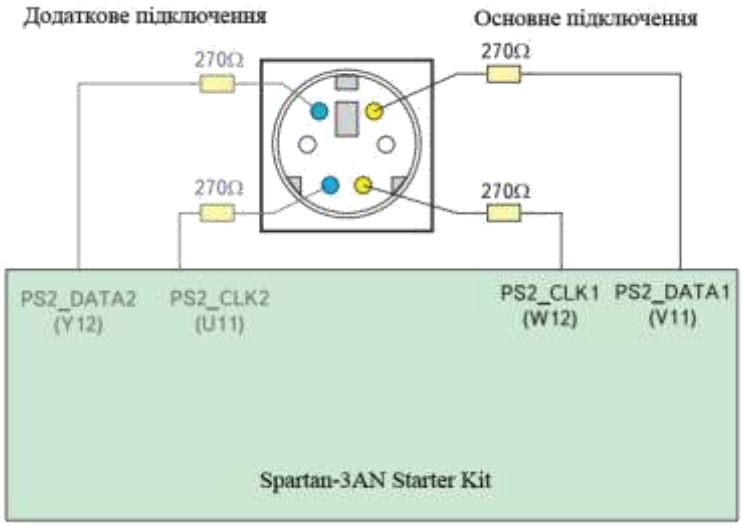


Рис. 25 Схема підключення контактів PS/2 до плати Spartan-3AN Starter Kit UCF – файл:

```
# Основне підключення

NET "PS2_CLK1" LOC = "W12" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;

NET "PS2_DATA1" LOC = "V11" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;

# Додаткове підключення

NET "PS2_CLK2" LOC = "U11" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;

NET "PS2_DATA2" LOC = "Y12" | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = SLOW ;
```

Робота протоколу PS/2

Обмін даними між клавіатурою і контролером здійснюється асинхронно за допомогою послідовного протоколу. Суть асинхронної передачі полягає в тому, що дані передаються лише тоді, коли є що передавати - натиснута / відпущена клавіша на клавіатурі і потрібно видати відповідний скан-код або контролеру потрібно видати команду клавіатурі.

Для обміну даними служать дві лінії - KBData і KBSync. При передачі скан-кодів клавіатура виставляє черговий розряд даних на лінії KBData і підтверджує передачу переведенням з "1" в "0" сигналу на лінії KBSync. При прийомі даних від контролера клавіатура зчитує розряд даних з лінії KBData і видає підтвердження прийому перекладом з "1" в "0" сигналу на лінії KBSync. Контролер може сигналізувати про свою неготовність передавати / приймати дані низьким рівнем на лінії KBSync. Весь інший час, коли немає даних для передачі, обидві лінії мають високий рівень сигналу. Частота проходження імпульсів лінії KBSync становить близько 10-25КГц.

Дані передаються в такому порядку: один стартовий біт - "0", вісім біт даних, біт парності (сума всіх розрядів +1), один стоповий біт - "1". Після прийому кожного байта даних контролер виставляє низький рівень на лінії KBSync, сигналізуючи тим самим, що зайнятий обробкою отриманих даних і не готовий приймати наступні. Це можна вважати підтвердженням прийому. Клавіатура підтверджує кожен байт прийнятої команди видачею коду 0FAh. При виникненні помилки при передачі, контролер може зажадати повторити передачу останнього байта, видачею команди 0FEh. Клавіатура ж веде себе по-іншому - просто ігнорує помилки.

PS/2 - клавіатура використовує коди сканування, щоб передати дані натиснення клавіш. У кожної клавіші є єдиний, унікальний код сканування, який відправляється щоразу, коли відповідна клавіша натиснута. Коди сканування для більшості клавіш клавіатури представлені на рис. 26

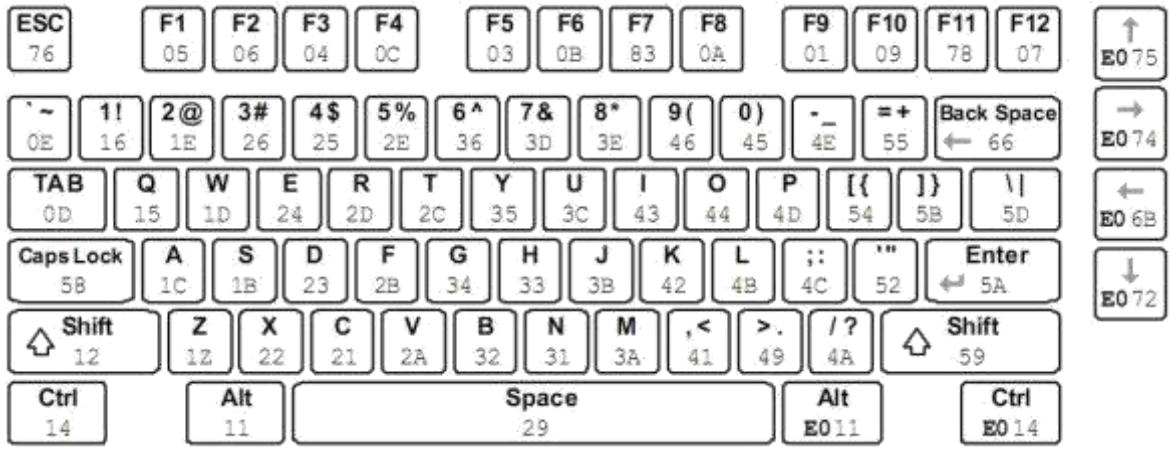


Рис. 26 Коди сканування клавіш клавіатури

Реалізація гри «пінг - понг»

Контролер клавіатури:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity KeyboardController is
   Port ( Clock : in STD_LOGIC;
               KeyboardClock : in STD_LOGIC;
          KeyboardData : in STD_LOGIC;
          LeftPaddleDirection : buffer integer;
          RightPaddleDirection : buffer integer
          );
end KeyboardController;

architecture Behavioral of KeyboardController is
```

```
signal bitCount : integer range 0 to 100 := 0;
signal scancodeReady : STD_LOGIC := '0';
signal scancode : STD_LOGIC_VECTOR(7 downto 0);
signal breakReceived : STD_LOGIC := '0';

constant keyboardA : STD_LOGIC_VECTOR(7 downto 0) := "00011100";
constant keyboardY : STD_LOGIC_VECTOR(7 downto 0) := "00011010";
constant keyboardK : STD_LOGIC_VECTOR(7 downto 0) := "01000010";
constant keyboardM : STD_LOGIC_VECTOR(7 downto 0) := "00111010";

begin

        keksfabrik : process(KeyboardClock)
        begin
                if falling_edge(KeyboardClock) then
                        if bitCount = 0 and KeyboardData = '0' then --keyboard wants
to send data
                                scancodeReady <= '0';
                                bitCount <= bitCount + 1;
                        elsif bitCount > 0 and bitCount < 9 then -- shift one bit into
the scancode from the left
                                scancode <= KeyboardData & scancode(7 downto
                                1); bitCount <= bitCount + 1;
                        elsif bitCount = 9 then -- parity bit
                                bitCount <= bitCount + 1;
                        elsif bitCount = 10 then -- end of
                                message scancodeReady <= '1';
```

```
                bitCount <= 0;
            end if;
        end if;
    end process keksfabrik;

    kruemelfabrik : process(scancodeReady, scancode)
    begin
        if scancodeReady'event and scancodeReady = '1' then
            -- breakcode breaks the current
            scancode if breakReceived = '1' then
                breakReceived <= '0';
                if scancode = keyboardA or scancode = keyboardY
                    then LeftPaddleDirection <= 0;
                elsif scancode = keyboardK or scancode = keyboardM
then
                    RightPaddleDirection <= 0;
                end if;
            elsif breakReceived = '0' then
                -- scancode processing
                if scancode = "11110000" then -- mark break for next
scancode
                    breakReceived <= '1';
                end if;

                if scancode = keyboardA then
                    LeftPaddleDirection <= -1;
                elsif scancode = keyboardY then
```

```
                    LeftPaddleDirection <= 1;
                elsif scancode = keyboardK then
                    RightPaddleDirection <= -1;
                elsif scancode = keyboardM then
                    RightPaddleDirection <= 1;
                end if;
            end if;
        end if;
    end process kruemelfabrik;
end Behavioral;
```

Завдання на роботу:

1. Ознайомитися з технічною документацією ("Spartan-3A/3AN FPGA Starter Kit Board User Guide", розділи 6, 8, 16).
2. Розібратися в представленому прикладі гри «пінг - понг».
3. Скомпілювати та завантажити на ПЛІС приклад гри. Переконатися в правильності її роботи.

Контрольні запитання

1. Назвати призначення роз’єму інтерфесу PS/2.
2. В чому полягає суть асинхронної передачі? Що для цього використовується?
3. Як в даній роботі використовується контролер клавіатури?

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. В.П. Тарасенко, А.О. Мельник Сучасні ситуативно-методологічні аспекти створення спеціалізованих комп’ютерних систем // Наукові вісті НТУУ "КПІ", 1997, -№1, -С.18-21.)
2. Spartan-3A/3AN FPGA Starter Kit Board User Guide. [Електронний ресурс]/XilinxInc.–Режимдоступу: http://www.xilinx.com/support/documentation/boards_and_kits/ug330.pdf.
3. Грушвицкий, Р.И. Проектирование систем на микросхемах программируемой логики / Р.И. Грушвицкий, А.Х. Мурсаев, Е.П. Угрюмов. -
4. Немудров В., Мартин Г. Системы на кристалле. Проектирование и развитие. — М.: Техносфера, 2004, -216 с.
5. Кузелин М.О. Современные семейства ПЛИС фирмы XILINX / М.О. Кузелин, Д.А. Кнышев, В.Ю. Зотов –М.: "Горячая линия – Телеком", 2004, 440 с.
6. Зотов Ю.В. Проектирование цифровых устройств на основе ПЛИС фирмы XILINX в САПР WebPack ISE. — М.: Горячая линия-Телеком, 2003. — 624 с.
7. Мельник А.О. Архітектура комп’ютера. Наукове видання. – Луцьк: Волинська обласна друкарня, 2008. – 470 с.
8. Книшев Д.А. ПЛИС фирмы «XILINX»: описание структуры основных семейств. М. Додэка-XXI, 2001, -238 с.
9. Максфилд К. Проектирование на ПЛИС. Курс молодого бойца. М.: Изд.дом "Додэка-XXI", 2007. – 408 с.
10. Угрюмов Е.П. Цифровая схемотехника. – СПб.: БХВ-Санкт-Петербург, 2000. – 528 с.