ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КЕРУЮЧИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Кафедра спеціалізованих комп'ютерних систем

МЕТОДИЧНІ ВКАЗІВКИ

до курсового проекту та самостійної роботи з дисципліни

«ТЕХНОЛОГІЇ ТА АВТОМАТИЗАЦІЯ ПРОЕКТУВАННЯ ПРИСТРОЇВ 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 спеціальностей (напрямів): 123 «Комп'ютерна інженерія» за освітньою програмою «Спеціалізовані комп'ютерні системи» та 126 «Інформаційні системи і технології» за освітніми програмами «Інтелектуальні інформаційні технології», «Технології штучного інтелекту», що вивчають дисципліну «Технології та автоматизація проектування пристроїв комп'ютерних систем», денної та заочної форм навчання.

Укладач

проф. М. А. Мірошник

Рецензент

проф. С. І. Доценко

МЕТОДИЧНІ ВКАЗІВКИ
до курсового проекту та самостійної роботи

з дисципліни
«ТЕХНОЛОГІЇ ТА АВТОМАТИЗАЦІЯ ПРОЕКТУВАННЯ ПРИСТРОЇВ КОМП'ЮТЕРНИХ СИСТЕМ»

Відповідальний за випуск Мірошник М. А.

Редактор Буранова Н. В.

Підписано до друку 22.03.19 р.

Формат паперу 60x84 1/16. Папір писальний.

Умовн.-друк. арк. 5,5. 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……... 3
1 Мета і завдання навчальної дисципліни…… 5
2 Робоча програма з дисципліни…… 8
2.1 Перелік тем лекційних занять……. 8
2.2 Перелік тем практичних занять……... 9
2.3 Перелік тем лабораторних робіт……. 9
2.4 Мета курсового проекту…….. 10
2.5 Форми самостійної роботи…….. 10
2.6 Характеристика підручників і навчальних посібників. 11
3 Курсовий проект……... 12
3.1 Мета і завдання курсового проекту…… 12
3.2 Загальні вимоги до тематики робіт……. 12
3.3 Структура і зміст курсового проекту…….. 14
3.4 Вказівки щодо виконання курсового проекту……... 22
3.5 Вимоги до оформлення курсового проекту……... 54
3.6 Організація виконання та захисту курсового проекту.. 58
4 Методичні вказівки до самостійного вивчення дисципліни 60
5 Індивідуальні розрахункові завдання, контрольні завдання 61
6 Приклади розв’язання типових завдань……. 66
Список літератури……. 75
Додаток А Форма титульного аркуша пояснювальної записки 77
Додаток Б Зразок заповнення аркуша завдання на курсовий проект 78
Додаток В Зразок оформлення реферату курсового проекту… 80
Додаток Г Зразок оформлення змісту курсового проекту……. 81

ВСТУП

На сьогодні найбільш актуальними вважаються системи, які працюють з декількома методами формалізації, що дають змогу виконувати строгий аналіз у реальному часі. Кілька видів формалізації потрібно при проектуванні складних систем, що складаються з компонентів з різними методами опису. Також це потрібно 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 і управління пристроєм, що підвищує гнучкість системи. Програмне забезпечення описується та розробляється до появи апаратних компонентів. Це потребує віртуальних прототипів hardware для налагодження software і засобів для виконання одночасної верифікації програмної та апаратної частин проекту.

На всіх етапах проектування, створення і експлуатації комп'ютерних систем потрібен їх всебічний аналіз за допомогою ефективного засобу автоматизації – комп'ютерного математичного моделювання. Використання моделювання дає змогу істотно скоротити терміни проектування системи, зменшити час прийняття рішень у процесі експлуатації системи, підвищити їх якість, спрогнозувати наслідки. Моделювання у наш час є обов'язковою складовою усіх етапів проектування, створення і експлуатації комп'ютерних систем різного призначення. Найбільшого поширення при цьому набуло математичне комп'ютерне моделювання.

Вивчення курсу спрямоване на освоєння студентами сучасної технології вирішення завдань моделювання із застосуванням комп'ютерних засобів, пакетів і систем моделювання, закріплення і поглиблення теоретичних знань з

дисципліни, набуття навичок вирішення практичних завдань дослідження засобами математичного моделювання та самостійного розв'язання задачі моделювання і передбачає створення, модернізацію або деталізацію запропонованої моделі, проведення експериментів з машинною моделлю, аналіз результатів моделювання. Кожне завдання курсового проекту передбачає самостійне з'ясування студентом питань, пов'язаних з розробленням комп'ютерної моделі досліджуваного об'єкта, вибором методу моделювання, принципу побудови моделювання алгоритму, розробленням схеми алгоритму і програми моделювання. Для програмної реалізації моделі студент повинен зробити обґрунтований вибір між сучасними мовами програмування загального призначення, мовами імітаційного моделювання, пакетами або системами моделювання. При цьому обов'язковою є наявність у пояснювальній записці курсового проекту роздруківок тексту програми і результатів моделювання, а також оцінок точності отриманих результатів.

1 МЕТА І ЗАВДАННЯ НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Метою навчальної дисципліни є формування у студентів спеціальності «комп'ютерна інженерія» базових знань у галузі технологій автоматизованого проектування, необхідних для освоєння моделей, методів, алгоритмів аналізу і синтезу апаратних і програмних засобів проектування цифрових обчислювальних систем і мереж різного призначення, а також вивчення загальних відомостей про об'єкти, моделі та задачі автоматизованого проектування. Студенти мають з'ясувати принципи та особливості функціонування різних систем автоматизованого проектування, надавати цілісну картину процесу автоматизованого проектування систем та навчитися робити вибір ефективних РІШЕНЬ щодо моделей комп'ютерних систем, методики розв'язання проектних завдань, математичного забезпечення процедур аналізу та синтезу проектних РІШЕНЬ, методики концептуального проектування складних систем, що покладені в основу технології CALS, питання інтеграції САПР з автоматизованими системами.

Завданням курсу є формування у студентів базових знань про існуючі у світі технології проектування комп'ютерних систем на ПЛІС, необхідних для освоєння методів аналізу і синтезу апаратних засобів цифрових обчислювальних систем різного призначення.

Дисципліна викладається з метою формування у студентів базових знань у сфері побудови моделей цифрових об'єктів, методів моделювання несправностей пристроїв, методів побудови табличних та мовних моделей цифрових пристроїв, технологій автоматизованого проектування комп'ютерних систем, необхідних для освоєння моделей, методів, алгоритмів аналізу і синтезу апаратних і програмних засобів проектування цифрових обчислювальних систем та проектування вузлів і блоків спеціалізованих комп'ютерних систем на базі пристроїв програмованої логіки із застосуванням мов опису апаратури.

У результаті вивчення цього курсу студенти повинні

знати:

– теоретичні основи створення автоматизованих систем проектування комп’ютерних систем: математичного, лінгвістичного, інформаційного, технічного, програмного, методичного забезпечення;

– стратегії проектування цифрових систем, методи синтезу та аналізу, використовувані в системах автоматизованого проектування цифрових пристроїв;

– ієрархічні моделі задання цифрових систем і відповідні їм мови опису апаратури;

– формальні засоби опису моделей цифрових об’єктів з використанням різних рівнів деталізації та алфавітів опису моделей, алгоритми справного та несправного моделювання, методи побудови табличних та мовних моделей цифрових пристроїв, методики використання методів моделювання при проектуванні цифрових пристроїв;

– сучасні засоби автоматизованого проектування провідних компаній світу і схемотехнічне забезпечення проектованих цифрових систем;

– перспективні технології створення високопродуктивних універсальних і спеціалізованих обчислювальних систем на основі логіки, що програмується;

- основні типи та класифікацію існуючих БІС, базові концепції мов програмування PLD;

- навчитися проводити аналіз розрахункових задач з метою складання ТЗ на проектування БІС потрібної архітектури. Складати опис проекту мовою VDHL. Працювати з програмним пакетом САПР БІС ACTIVE-VHDL, компелювати вхідний опис, задавати необхідний режим роботи, аналізувати внутрішню модель опису проекту. Розв'язувати типові задачі проектування на всіх рівнях за допомогою пакета САПР БІС Active-VDHL;

Вміти:

– формулювати і розв'язувати практичні задачі синтезу та аналізу систем проектування цифрових виробів на основі вибору найбільш раціональних методів і засобів (мови опису апаратури, формати внутрішнього подання даних, моделі опису цифрових об'єктів, методи й алгоритми моделювання, верифікації, синтезу, імплементації в кристали, тестування і діагностування проектованих виробів) з метою їхнього оптимального розв'язання;

– використовувати формальні методики опису при проектуванні моделей цифрових об'єктів різного рівня деталізації, використовувати різні методи моделювання при верифікації проектів, аналізу повноти тестів, аналізу часових співвідношень з використанням інтерпретативних та компілятивних систем логічного моделювання;

– використовувати сучасні комп'ютерні системи автоматизованого проектування комп'ютерних систем на прикладі Xilinx ISE та Activ-HDL;

– використовувати різні форми та рівні опису пристроїв, які проектуються, із застосуванням мов опису апаратури (VHDL), проводити функціональне моделювання з використанням середовища Active-HDL, використовувати програмовану логіку при проектуванні різних спеціалізованих пристроїв обчислювальної техніки;

мати уяву про перспективи розвитку сучасних комп'ютерних систем, тенденції розвитку засобів автоматизації, проектування і автоматизації проектування комп'ютерних систем.

Матеріал дисципліни вивчається у такому порядку: елементи мови Verilog; синтез схем моделей, які написані мовою

Verilog, технології проектування ASIC та PLD; сучасні інструменти проектування ASIC та PLD.

2 РОБОЧА ПРОГРАМА З ДИСЦИПЛІНИ

2.1 Перелік тем лекційних занять

Навчальний матеріал дисципліни вивчається частково в аудиторному режимі і в основному – в режимі самостійної роботи.

Модуль 1. Основи проектування комп'ютерних систем.

Змістовий модуль 1. Основи проектування комп'ютерних систем.

Тема 1. Загальні відомості про системи автоматизації проектних робіт.

Тема 2. Цифрові системи та методи наведення інформації.

Тема 3. Цифрові схеми та логічні елементи.

Тема 4. Принципи проектування комбінаційних логічних схем.

Тема 5. Проектування цифрових схем за допомогою VHDL.

Тема 6. Практичне розроблення схем комбінаційної логіки.

Тема 7. Приклади проектування комбінаційних схем.

Модуль 2. Автоматизація проектування складних комп'ютерних систем.

Змістовий модуль 2. Автоматизація проектування комп'ютерних систем.

Тема 8. Принципи проектування послідовних логічних схем.

Тема 9. Практичне розроблення послідовних схем.

Тема 10. Приклади проектування послідовних схем.

Тема 11. Проектування пристроїв на програмованих логічних інтегральних схемах (ПЛІС). Огляд існуючої елементної бази. Введення в архітектуру ПЛІС.

Тема 12. Пам'ять та мікросхеми типу CPLD та FPGA. Принципи програмування і зберігання конфігурації.

Тема 13. Засоби автоматизації проектування. Мови проектування та верифікації. Синтез схем із використанням мов опису апаратури VHDL і Verilog.

Тема 14. Імплементація пристроїв і аналіз тимчасових параметрів.

Тема 15. Тестування логічних схем.

Тема 16. Проектування арифметичних пристроїв.

Тема 17. Системний рівень проектування комп'ютерних систем. Перспективи розвитку систем автоматизації проектування комп'ютерних систем.

2.2 Перелік тем практичних занять

1 Мови опису апаратури: VHDL Verilog. Побудова мовних моделей цифрових пристроїв.

2 VHDL-код 4-розрядного суматора.

3 Побудова мовних моделей мультиплексора та демультиплексора.

4 Побудова мовних моделей автомата Милі та Мура за допомогою VHDL.

5 Проектування різних типів пам'яті на ПЛІС.

6 Дослідження і налагодження VHDL-описаних окремих фрагментів комп'ютерних систем.

7 Створення на основі ієрархічних систем VHDL-описів проектних модулів, VHDL-описів реальних систем у цілому.

8 Моделювання на VHDL різних мікросхем і мікропроцесорних систем.

2.3 Перелік тем лабораторних робіт

1 Ознайомлення із середовищем Project Navigator пакета Xilinx ISE.

2 Синтез комбінаційних схем у Project Navigator пакета Xilinx ISE. Вивчення CIC компонентів комбінаційного типу.

3 Синтез тригерів різних типів за допомогою середовища Project Navigator пакета XILINX ISE. Вивчення мовного опису моделей.

4 Створення схеми мультиплексора та демультиплексора.

5 Структурний синтез мікропрограмного автомата Милі та Мура канонічним способом в САПР Xilinx ISE.

6 Створення Verilog-моделей та автоматизований синтез цифрових пристроїв. Проектування арифметичних пристроїв. Автоматизація процесу тестування пристроїв.

7 Реалізація пристроїв на мікросхемі програмованої логіки. Розроблення цифрових схем та різних типів пам'яті на базі ПЛІС.

8 Проектування пристроїв цифрової обробки сигналів (FiR-filter). Розроблення структурних та поведінкових і Data-flow Verilog-моделей пристроїв.

9 Створення синтезованих моделей автоматів. Проектування і реалізація арифметичних пристроїв та середовища верифікації за допомогою SystemVerilog.

2.4 Мета курсового проекту

Метою курсового проекту є закріплення і поглиблення знань, отриманих у процесі вивчення теоретичного курсу, вироблення навичок самостійного створення (вибору) засобів математичного комп'ютерного моделювання для розв’язання завдань планування або оперативного управління, аналізу або синтезу комп'ютерних систем, об'єктів управління або їх окремих підсистем.

Тематика курсових робіт відповідає змісту і завданням дисципліни. Вона може бути пов'язана з тематикою НІРС, дипломним проектуванням, тематикою науково-дослідних і навчально-методичних робіт, виконуваних на кафедрі (факультеті).

2.5 Форми самостійної роботи

Поглиблене вивчення лекційного матеріалу.

Самостійне вивчення теоретичного матеріалу за допомогою рекомендованої літератури, список якої подано наприкінці цих методичних вказівок. Підготовка до практичних занять.

Підготовка до виконання лабораторних робіт.

Виконання курсового проекту.

Програмне забезпечення:

1 Система моделювання цифрових пристроїв – Active-HDL фірми Aldec. Synplify Pro фірми Sinplicity, ISE WEBPack фірми Xilinx.

2 Компілятор мови ABEL. Транслятори мов VHDL і Verilog. Графічний редактор схем та засоби моделювання. Пакети Active-CAD™ і Active-HDL™ фірми Aldec, Inc. Програмний продукт FPGA Express™ фірми Synopsys.

2.6 Характеристика підручників і навчальних посібників

Наприкінці цих методичних вказівок подано список рекомендованої літератури, який умовно поділено на чотири частини: основна література, додаткова (друковані видання), методичні вказівки (в електронному вигляді) та програмне забезпечення.

Основна література використовується для вивчення теорії. Вона висвітлює практично весь досліджуваний матеріал.

[1] можна використовувати для вивчення класифікації інтегральних схем та інтегральних схем програмованої логіки (тема 2, 3). Є стислий опис мови Verilog. [2] – міжнародний стандарт. [3] – підручник, містить інструкції користувачеві для роботи в системі Active-HDL, яка застосовується для проведення лабораторних робіт. Крім того, підручник містить інформацію про елементну базу програмованої логіки, а також методи тестопридатного проектування. [4, 5] – посібники, що містять інформацію щодо проектування цифрових систем інструкції користувачеві для роботи в системі Active-HDL.

Додаткова література використовується для виконання лабораторних та практичних робіт. [9, 10] – гіпертекстові навчальні матеріали. [11, 12] – підручники, додаткові до [3].

Методичні вказівки мають бути використані в повному обсязі при виконанні лабораторних робіт, вивченні теоретичного матеріалу і виконанні розрахункового завдання. [13] – вказівки до виконання лабораторних робіт. [14] – вказівки до самостійної роботи. [15] – конспект теоретичного матеріалу, містить інформацію щодо всіх досліджуваних тем, а також екзаменаційні питання.

3 КУРСОВИЙ ПРОЕКТ

3.1 Мета і завдання курсового проекту

Метою виконання курсового проекту є закріплення і поглиблення знань, отриманих у процесі вивчення теоретичної частини дисципліни; набуття навичок самостійного використання методів і засобів комп'ютерного моделювання при розв’язанні задач автоматизації проектування комп'ютерних систем. Основне завдання курсового проекту – підготовка студентів до самостійної практичної інженерної діяльності щодо вирішення всього комплексу завдань проектування – від розроблення комп'ютерної моделі системи до інтерпретації результатів моделювання та оформлення документації. Успішне виконання курсового проекту передбачає:

- систематизацію і поглиблення знань по суті завдань усіх етапів автоматизованого проектування, особливостей підходів (методів) формалізації і алгоритмізації, можливостей дослідження різних класів моделей з використанням сучасних програмних і технічних засобів;
- набуття навичок самостійного розв’язання основних інженерних задач моделювання, пов'язаних з розробленням концептуальних моделей, їх формалізацією, вибором методу моделювання, плануванням і обробкою результатів машинних експериментів;
- закріплення і розвиток навичок з вибору принципів побудови і розроблення моделювальних алгоритмів, роботи з мовами і пакетами програм моделювання, індивідуального розроблення моделювальних програм.

Курсовий проект є самостійною роботою студента, за яку він несе відповідальність за достовірність усіх даних і прийняті в ній технічні рішення.

3.2 Загальні вимоги до тематики робіт

Тематика виконуваних курсових робіт має відповідати основним розділам робочої програми дисципліни «Технології та автоматизація проектування пристроїв складних комп'ютерних

систем». Теоретична частина кожної роботи має базуватися на лекційному матеріалі дисципліни «Технології та автоматизація проектування пристроїв складних комп'ютерних систем». Практична частина курсового проекту має бути присвячена розробленню та аналізу комп'ютерної моделі досліджуваної системи або процесу відповідно до обраної теми.

Загальні завдання на курсовий проект (таблиця 3.1) передбачають

- розробку моделі і проведення машинного експерименту з метою аналізу або синтезу систем і підсистем систем обробки інформації та управління;
- розробку програм для планування і проведення експериментів;
- створення програмних засобів моделювання для НДР або навчального процесу.

В окремих випадках студентам, які виявили здатність до наукових досліджень, можуть бути запропоновані завдання теоретичного характеру, пов'язані з розв’язанням завдань автоматизованого проектування (таблиця 3.1).

Тема і завдання курсового проекту можуть бути як навчальними, так і безпосередньо пов'язаними з реальними завданнями, що розробляються на кафедрі, підприємствах або в організаціях, бути частиною або логічним продовженням робіт, раніше виконуваних студентом. Теми і завдання на курсовий проект формуються керівником і є індивідуальними. Завданням є спроектувати схему синхронного реверсивного двійково-десяткового лічильника, закодувати виходи за допомогою шифратора «2 з 5», промоделювати її мовою VHDL у середовищі моделювання ActiveHDL і запрограмувати плату Basys2 серії Spartan 3E в середовищі автоматизованого проектування фірми Xilinx Web Pack ISE.

Таблиця 3.1 – Завдання

Варіант	Код	Послідовність	перенос	Тригер	Студент
1	8-4-2-1	5,2,8,0,3,9,6,1,4,7	7-5, 5-7	Типу D	
2	7-4-2-1	3,1,7,4,6,5,0,9,3,8	3-8, 8-3	Типу D	
3	5-4-2-1	1,0,9,6,3,8,5,2,4,7	1-7, 7-1	Типу D	
4	5-2-1-1	8,6,4,9,0,5,7,3,1,2		Типу D	
5	+3	5,2,4,1,9,6,8,0,7,3		Типу D	
6	8-4-2-1	3,2,5,6,0,9,8,7,4,1		Типу D	
7	7-4-2-1	1,6,8,4,2,0,3,5,9,7		Типу D	
8	5-4-2-1	3,7,1,0,6,5,9,4,2,8		Типу D	
9	5-2-1-1	2,5,1,6,0,9,7,8,4,3		Типу D	
10	+3	9,7,4,8,5,0,1,6,3,2		Типу D	
11	8-4-2-1	2,6,5,9,7,4,8,0,3,1		Типу D	
12	7-4-2-1	2,5,9,6,4,7,8,0,1,3		Типу D	
13	5-4-2-1	4,3,5,1,6,9,7,2,0,8		Типу D	
14	5-2-1-1	6,1,7,2,8,3,9,4,5,0		Типу D	
15	+3	7,3,8,4,9,1,5,0,6,2		Типу D	
16	8-4-2-1	5,0,9,1,2,8,3,7,4,6		Типу D	
17	7-4-2-1	1,3,2,4,5,7,6,8,9,0		Типу D	
18	5-4-2-1	8,2,9,3,7,1,6,0,5,4		Типу D	
19	5-2-1-1	0,9,1,8,2,7,3,4,6,5		Типу D	
20	+3	0,1,2,3,4,5,6,7,8,9		Типу D	

3.3 Структура і зміст курсового проекту

Курсовий проект являє собою пояснювальну записку, що складається з текстової (теоретична частина, тексти і описи програм) і графічної частини (діаграми потоків, схеми алгоритмів, програм, відеограми, отримані графічні залежності).

Структурні частини пояснювальної записки – реферат; зміст; перелік умовних позначень, символів, одиниць, скорочень і термінів; вступ; основна частина; висновок; перелік посилань; додатки (розміщуються на окремих сторінках і мають заголовки симетрично до тексту великими літерами).

Пояснювальна записка є основним документом, що надається студентом під час захисту. Вона має давати достатньо повне уявлення про об'єкт дослідження, принцип побудови моделювального алгоритму, про схему проведення експериментів

і отримані результати. Зміст пояснювальної записки має повністю відповідати завданням на курсовий проект. Загальний обсяг пояснювальної записки має складати близько 25–30 сторінок. Пояснювальна записка до курсового проекту має містити: титульний аркуш – 1с.; завдання на курсовий проект – 2 с.; реферат – 1 с.; зміст – 1 с.; вступ – 1–2 с.; основну (спеціальну частину) – 15–20 с.; висновки – 1–2 с.; перелік посилань – 0,5 с.; додатки. Основну увагу необхідно приділити опису практичних або теоретичних результатів, отриманих при виконанні роботи. Титульний аркуш є першою сторінкою, містить інформацію про тему курсового проекту, її автора і керівника. Форму титульного аркуша наведено в додатку А.

Технічне завдання є другою сторінкою пояснювальної записки. Завдання на курсовий проект містить назву роботи, прізвище автора і керівника, основні вихідні дані для її виконання і список рекомендованої літератури. Вказують також дату видачі завдання на курсовий проект і дату подання закінченої роботи, а також узгоджений календарний план виконання роботи. Форму аркуша завдання на курсовий проект подано в додатку Б.

Реферат має відповідати вимогам ДСТУ 3008-2015 і містити відомості про обсяг пояснювальної записки, кількість ілюстрацій, таблиць, додатків, використаних джерел згідно з переліком посилань, текст і перелік ключових слів. Текст реферату має відображати об'єкт дослідження, мету роботи, метод дослідження (розв’язання задачі), отримані результати, їх практичну значущість і рекомендації щодо використання. Відомості записують за формою: «Записка 100 с.; 18 ілюстрацій, 14 таблиць, 5 додатків, 15 використаних джерел згідно з переліком посилань». Текст реферату має відображати інформацію, подану в пояснювальній записці, в такій послідовності: об'єкт розробки або дослідження; мета роботи; використані математичні методи, технічні та програмні засоби; результати і їх новизна; основні характеристики об'єкта розробки або дослідження; ступінь реальності курсового проекту; комплексність проекту; рекомендації щодо використання результатів роботи; можлива галузь застосування; економічна ефективність; висновки. Реферат не повинен перевищувати обсяг однієї сторінки, і поміщають його після технічного завдання.

Обсяг реферату – не більше 500 слів. Ключові слова, які є важливими для розкриття суті роботи, поміщають після тексту реферату. Перелік ключових слів містить від 5 до 15 слів, надрукованих великими літерами в називному відмінку в рядок через коми.

Зміст розташовують безпосередньо після реферату, починаючи з нової сторінки. До змісту включають: перелік умовних позначень, символів, одиниць, скорочень і термінів; вступ; послідовно перелічені назви всіх розділів, підрозділів, пунктів і підпунктів документа (якщо вони мають заголовки). Нумерація сторінок змісту має бути виконана так, щоб розряди чисел були розташовані один під іншим. У змісті відображаються розділи (підрозділи) роботи із зазначенням їхніх назв і сторінок у тексті пояснювальної записки. Перелік умовних позначень, символів, одиниць, скорочень і термінів має розташовуватися стовпцем, в якому зліва в алфавітному порядку наводяться умовні позначення або скорочення, а справа – їх докладне розшифрування в називному відмінку.

Оформлення основної частини пояснювальної записки курсового проекту (роботи), що виконується за комп'ютерною технологією, має відповідати вимогам стандарту ДСТУ 3008-2015 «Документація. Звіти у сфері науки і техніки. Структура і правила оформлення». Пояснювальна записка оформляється на аркушах формату А4 (210х297 мм) електронним способом на одному боці аркуша білого паперу відповідно до вимог ДСТУ 3008-2015. Пояснювальна записка має бути виконана акуратно, грамотно. При написанні тексту слід дотримуватися таких полів: верхнє, ліве і нижнє – не менше 20 мм, праве – не менше 10 мм. Скорочення слів і словосполучень виконувати відповідно до ДСТУ 3008-2015.

Структурні елементи «Реферат», «Зміст», «Перелік умовних позначень, символів, одиниць, скорочень і термінів», «Вступ», «Висновки», «Перелік посилань» не нумерують, а їх назви правлять за заголовки структурних елементів.

Нумерацію сторінок робити наскрізною у межах усього документа. Нумерувати сторінки слід арабськими цифрами. На титульному аркуші номер не ставлять, але мають на увазі, на наступних сторінках номер проставляється в правому верхньому

куті без крапки, біля поля, на відстані півтора міжрядкового інтервалу від тексту (6 мм). У вступі стисло викладають актуальність теми і завдання курсового проекту, мету роботи, сфери її застосування, вказують взаємозв'язок з іншими завданнями автоматизації проектування.

Основна частина складається з розділів, у яких викладається суть обраних рішень на всіх етапах проектування, починаючи з опису об'єкта і закінчуючи отриманням і інтерпретацією результатів. При цьому наводяться постановка задачі проектування, стислий огляд існуючого інструментарію і фірм розробників, обґрунтування вибору підходу до проектування, розглядаються питання розробки системи, вибору методу проектування і принципу побудови алгоритму, мови програмування і пакета програм автоматизованого середовища проектування, надається аналіз результатів проектування та моделювання, у тому числі порівняння їх з результатами аналітичної оцінки і оцінок можливих поліпшень у роботі системи (відповідно до варіанта завдання). Текст основної частини поділяють на розділи. Якщо цього вимагає виклад, останні поділяють на підрозділи, пункти і підпункти. Кожен розділ і підрозділ має бути із заголовком, який відповідає його змісту. Пункти і підпункти також можуть мати заголовки. Перенесення слів у заголовках не допускається, крапку в кінці заголовка не ставлять. Якщо найменування структурної частини документа складається з двох речень, їх розділяють крапкою. Заголовки підрозділів, пунктів і підпунктів слід починати з абзацного відступу і виконувати малими літерами, крім першої великої, не підкреслюючи, без крапки в кінці. Текст пункту, що не має заголовка, разом з порядковим номером записують з абзацу, в кінці тексту ставлять крапку. Номери пунктів поставляються з абзацу. Вміщені в пунктах переліки записують з абзацу. В кінці кожної позиції переліку ставиться крапка з комою, після останньої – крапка.

Переліки за потреби можуть бути подані всередині пунктів або підпунктів. Перед переліком ставлять двокрапку. Перед кожною позицією переліку слід ставити малу літеру української абетки з дужкою або дефіс (перший рівень деталізації). Для

подальшої деталізації переліку слід використовувати арабські цифри з дужкою (другий рівень деталізації).

Абзацний відступ має бути однаковим упродовж усього тексту пояснювальної записки, що дорівнює п'яти знакам (1,24 мм). Відстань між заголовком і подальшим чи попереднім текстом має бути не менше двох рядків (15 мм). Не допускається розміщувати назву розділу, підрозділу, а також пункту і підпункту в нижній частині сторінки, якщо після неї розміщено тільки один рядок тексту.

Текст викладати українською мовою стисло, чітко, з однозначним його тлумаченням. Мова викладу має бути простою, характерною для наукових і технічних документів. Застосовувана термінологія має відповідати встановленій у стандартах; за їх відсутності вона має бути загальноприйнятою в науково-технічній літературі. Якщо в документі прийнято специфічну термінологію, то надається перелік специфічних термінів з відповідними роз'ясненнями.

Розрахунковий матеріал викладається таким чином: описуються принципові схеми математичної моделі і прийнятого методу розрахунку; наводяться розрахункові формули з розшифруванням символів і коефіцієнтів (алгоритм розрахунку) або вказується послідовність обчислень (блок-схема розрахунку, програма розрахунку); записуються вихідні дані для розрахунку; результати розрахунків наводяться в таблицях і на графіках.

Для пояснення тексту допускається включати в нього ілюстрації (графіки, діаграми, схеми, копії креслень, фотознімки та ін.). Ілюстрації, що поміщаються в тексті, називають рисунками. Їх слід розташовувати в записці безпосередньо після тексту, в якому вони згадуються вперше, або на наступній сторінці. Рисунки слід нумерувати порядковою нумерацією в межах розділу арабськими цифрами. Номер рисунка складається з номера розділу і порядкового номера рисунка в розділі, між якими ставлять крапку. Номер записують після слова «Рисунок». Рисунок за потреби може мати найменування і пояснення (підрисунковий текст). Якщо ілюстрація не вміщується на одній сторінці, можна переносити її на інші сторінки, при цьому назву ілюстрації поміщають на першій сторінці, пояснювальні дані на кожній сторінці і під ними вказують: «Рисунок_, аркуш__».

Написи на рисунках виконують шрифтом, прийнятим у тексті пояснювальної записки. Виконання графічної частини має відповідати вимогам стандартів ЄСПД. При виконанні діаграм, графіків та інших аналогічних документів слід користуватися рекомендаціями, викладеними в правилах виконання діаграм. При виконанні ілюстрацій слід уникати перетину написів і ліній. Це правило не поширюється на діаграми.

Цифрові та інші дані рекомендується поміщати в таблиці. Розміри таблиць вибирають довільно залежно від вміщуваного матеріалу. Горизонтальні і вертикальні лінії, які розмежовують рядки таблиці, а також лінії, що обмежують таблицю зліва, справа і знизу, можна не проводити, якщо їх відсутність не утруднює користування таблицею. Якщо таблицю розбито на рядки, то висота їх має бути не менше 8 мм. Записи в рядках проводять в один ряд. Рядки граф не можна залишати порожніми. Таблицю слід розташовувати безпосередньо після тексту, в якому вона згадується вперше, або на наступному аркуші. На всі таблиці мають бути посилання в тексті пояснювальної записки. Діагональний розподіл головки таблиці не допускається. Заголовки і підзаголовки граф виконують малими літерами (крім першої великої). Якщо підзаголовок складає одне речення із заголовком, то його починають з малої літери. В кінці заголовків і підзаголовків розділові знаки не ставлять. Для скорочення тексту заголовків і підзаголовків окремі поняття, якщо вони пояснені в тексті або наведені на ілюстраціях, дозволяється замінювати літерними позначеннями. Показники з одним і тим самим літерним позначенням групують послідовно, в порядку зростання індексів. У лівому боці таблиці поміщають найменування показників, параметрів та інші дані. Текст рядків виконують малими літерами (крім першої великої). Окрему графу «№ з/п» вводити не дозволяється. За потреби нумерації показників, параметрів або інших даних порядкові номери вказують у лівому боці таблиці перед їхнім найменуванням. Таблиці мають порядкові номери. Нумерація виконується в межах розділу арабськими цифрами без знака «№». При нумерації номер таблиці складається з номера розділу і порядкового номера таблиці, між якими ставиться крапка. Таблиця може мати назву. Її виконують малими літерами (крім

першої великої) і вміщують над таблицею. При перенесенні таблиці на наступну сторінку головку таблиці повторюють і над нею пишуть: «Продовження таблиці ...» із зазначенням номера. Таблицю з великою кількістю граф ділять на частини і поміщають одну частину під іншою, вказуючи над подальшими частинами слова «Продовження таблиці ...». Найменування в цьому випадку поміщають тільки над першою частиною таблиці.

Примітки формують у разі потреби пояснення змісту тексту, таблиці або ілюстрації. Примітки розташовують безпосередньо після тексту, таблиці, ілюстрації, яких вони стосуються. Одну примітку не нумерують. Слово «Примітка» друкують з великої літери з абзацного відступу, не підкреслюють, після слова «Примітка» ставлять крапку і з великої літери в тому ж рядку подають текст примітки. Кілька приміток нумерують послідовно арабськими цифрами з крапкою. Після слова «Примітки» ставлять двокрапку і з нового рядка з абзацу після номера примітки з великої літери подають текст примітки.

Використані джерела записуються до переліку посилань відповідно до вимог ДСТУ 3008–2015 і розташовуються в порядку появи на них посилань у тексті пояснювальної записки. У тексті документа допускаються посилання на стандарти (крім стандартів підприємств); технічні умови та інші документи за умови, що вони повністю і однозначно визначають відповідні вимоги і не викликають труднощів у користуванні документами. Якщо є підозра на використання будь-якого нормативно-технічного документа, посилатися треба на документ у цілому або на його розділи і додатки. Посилання на графічний матеріал курсового проекту, якщо їхніх копій не наведено в пояснювальній записці, наводяться за позначеннями у відомостях курсового проекту. При посиланнях на стандарти і технічні умови вказують тільки позначення; при посиланнях на інші документи – найменування документа; при посиланні на розділ або додаток – їх номери і найменування, при повторних посиланнях – тільки номер. Посилання в тексті на джерела слід вказувати порядковим номером за переліком посилань, виділеним двома квадратними дужками. Допускається наводити посилання на джерела у виносках, при цьому оформлення посилання має відповідати його бібліографічному опису за

переліком посилань із зазначенням номера. При посиланнях на розділи, підрозділи, пункти, підпункти, ілюстрації, таблиці, формули, рівняння, додатки зазначають їх номери.

У висновках вказують ступінь повноти розв'язання поставленого завдання, наводять характеристику отриманих результатів (розроблених засобів), оцінюють їхню соціальну значущість і ефективність, перспективи використання, вказують можливі напрямки вдосконалення.

Додатки слід оформляти як продовження пояснювальної записки на аркушах формату А4 (допускається – на аркушах формату А3), розташовуючи додатки в порядку появи посилань на них у тексті документа. У додатки можуть бути включені: додаткові ілюстрації та таблиці; матеріали, які через великий обсяг, специфіку викладення або форму подання не можуть бути внесені до основної частини; додаткові джерела, на які не було посилань у тексті пояснювальної записки; опис новітніх систем автоматизації та візуалізації програмування, новітніх технічних і програмних засобів, використаних при виконанні курсового проекту.

Кожна програма має починатися з нової сторінки і мати заголовок, наведений вгорі малими літерами з першої великої симетрично відносно тексту сторінки. Посередині рядка над заголовком малими літерами з першої великої пишеться слово «Додаток __» і велика літера без крапки, що позначає додаток. Додатки слід позначати послідовно великими літерами української абетки, за винятком Ґ, Є, З, І, Ї, Й, О, Ч, Ь, наприклад: «Додаток А», «Додаток Б» і т. д. Один додаток позначається як додаток А. Додатки мають спільну з іншою частиною пояснювальної записки наскрізну нумерацію сторінок. Текст, ілюстрації і таблиці в додатках оформляють за встановленими правилами. Ілюстрації і таблиці нумерують у межах кожного додатка, наприклад: рисунок А.3 – третій рисунок додатка А, таблиця Б.2 – друга таблиця додатка Б. Якщо в додатку одна ілюстрація, одна таблиця, їх нумерують, наприклад, рисунок А.1, таблиця А.1. При посиланнях у тексті додатка на ілюстрації та таблиці рекомендується писати: «… на рисунку А.3 …» ; «... в таблиці Б.2 ...». Як додаток можна використовувати документ, що має самостійне значення і оформлюється згідно з вимогами до

документа такого виду. Його копію поміщають у пояснювальній записці без змін в оригіналі. Перед копією документа поміщають аркуш, на якому посередині друкують слово "Додаток ___" і його найменування (за наявності), в правому верхньому куті аркуша проставляють порядковий номер сторінки. Сторінки копії документа нумерують, продовжуючи наскрізну нумерацію сторінок пояснювальної записки.

3.4 Вказівки щодо виконання курсового проекту

3.4.1 Загальні рекомендації щодо реалізації етапів моделювання

При виконанні курсового проекту слід виділяти такі основні етапи:

- аналіз існуючих сучасних САПР для автоматизації проектування комп'ютерних систем і фірм-виробників даних САПР, їх місце на ринку. Особливе місце фірми Xilinx з її вільно поширюваним продуктом WebPack ISE;
- теоретичний опис сучасних інструментаріїв для налагодження новостворених програмних продуктів, таких як Spartan-3E і Basys2;
- теоретичне обґрунтування використання мови високого рівня VHDL (або VeryLog) для опису даного проекту. Перелік основних операторів мов і методів опису моделей;
- проектування двійково-десяткового реверсивного лічильника і кодування його виходів за допомогою шифратора «2 з 5»;
- розробка VHDL-опису спроектованих пристроїв;
- створення проекту в середовищі Project Navigator автоматизованого середовища Xilinx WebPack ISE, опис файлу обмежень .ucf для зіставлення входів і виходів розроблювального пристрою з перемикачами, кнопками і світлодіодами реальної плати Basys2;
- генерування програмованого файлу для плати Basys2 і її прошивка.

Кожен з етапів у свою чергу складається з безлічі взаємозалежних завдань.

Перелік і суть основних завдань:

Проектування цифрової системи починається з розробки технічного завдання, на основі якого потім будується функціональна схема, що послідовно перетворюється в реальний пристрій. Моделювання та синтез доповнюють одне одного процедурами, використовуваними в процесі проектування, хоча їх точне взаємовідношення залежить від планованої реалізації (рисунок 3.1).

Рисунок 3.1 – Основні етапи спадного проектування цифрових систем

Перший етап – створення технічного завдання, яке зазвичай містить вимогу до роботи даного продукту, опис інтерфейсу, вартісні обмеження і інші фізичні вимоги, такі як метричні характеристики системи і розсіювання потужності. На основі технічного завдання створюється попередній функціональний проект. Моделювання на цьому етапі використовується для приведення цієї схеми у відповідність до технічного завдання. Потім вона перетворюється в модель рівня регістрових передач (RTL – register transfer level), що описує регістри, модулі пам'яті, операційні та керувальні автомати.

Наступний етап – створення логічних схем для кожного компонента.

На рівні регістрових передач і на логічному рівні можна використовувати моделювання для верифікації проекту. Моделювання несправностей дає змогу отримувати вихідні реакції схеми для заданого класу дефектів, що виникають на етапах виробництва і експлуатації. Якщо існуючі статистичні дані вказують на досить високий відсоток цих дефектів, проект слід модифікувати з метою забезпечення тестопридатності і відновлення. Нарешті, опис логічного рівня перетворюється в схемний, а потім проектується топологія кристала і обчислюються реальні фізичні властивості проекту, такі як площа кристала і розсіювання потужності. Перевіряються проектні норми, визначаються параметри схеми, після чого на цьому рівні може бути виконано верифікацію проекту. На кожному кроці ієрархічного проектування для опису структури використовуються компоненти. На найвищому абстрактному рівні – це невелика кількість складних елементів, таких як суматори або пам'ять. На нижньому – величезна кількість простих елементарних компонентів, наприклад елементів або транзисторів. Кожен рівень ієрархії проекту є певною абстракцією, має безліч операцій і підтримувальних інструментів проектування, частину з яких наведено на рисунку 3.1. За будь-якого рівня абстрактного уявлення можна передбачити поведінку об'єкта. Але фізичні властивості і можливості схеми бажано розглядати, переходячи до нижніх рівнів, хоча для них потрібен довший час аналізу і проектування.

Виправлення помилок проектування, виявлених на завершальних етапах розробки пристрою, є дуже дорогим процесом. Це може призвести до збільшення витрат часу на створення проекту, а отже, до зростання вартості готового виробу, не кажучи вже про втрати прибутку через запізнення на ринок. Завдяки можливості моделювання поведінки об'єкта на різних рівнях абстракції, несправності можна виявляти і виправляти значно раніше. Мови опису апаратури призначені для їх використання на всіх етапах проектування і тестування, вони забезпечують сумісність і відповідність між різними рівнями, а також можливості діагностування помилок проектування за наявності тестів перевірки несправностей, що належать до модельних або фізичних дефектів.

Типова процедура автоматичного синтезу залежить від обраної для імплементації схемотехніки (програмованої логіки). Розглянемо етапи проектування пристрою для реалізації на програмованих користувачем логічних матрицях FPGA (Field Programmable Gate Arrays). У цьому випадку FPGA можна розглядати як електронний пристрій з великою кількістю елементів і тригерів, зв'язки між якими можна запрограмувати як автоматично, так і ручним способом. На рисунку 3.2 показано змінену схему етапів проектування. На початку проектування на основі технічного завдання створюється функціональна схема мовою VHDL і виконується її верифікація. Потім вона перетворюється до рівня регістрових передач, визначаються потоки даних і основні апаратні функціональні модулі. Після отримання VHDL-моделі рівня регістрових передач за допомогою синтезу компілятора створюється елементний еквівалент об'єкта. На цьому етапі логічне моделювання може бути використано для попередньої оцінки можливостей спроектованого пристрою. Об'єкт на логічному рівні представлений з логічних елементів і тригерів, що входять до складу FPGA. Відповідні елементи і тригери з'єднуються за допомогою сигналів, що конфігуруються. Такий процес називається розміщенням і маршрутизацією, після закінчення якого можна отримати точні оцінки затримок на лініях і елементах. Після цього виконується моделювання реальних часових характеристик.

Узагальнену структуру стадій проектування зображено на рисунку 3.3. Основу вдосконалення цифрових систем складають досягнення мікроелектроніки. Ефективність її розвитку така, що кожного тижня тактова частота універсального процесора збільшується на 5 МГц, істотно знижується споживана потужність кристалів мікросхем і застосовуються субмікронні технології при їх виробництві. Крім того, спостерігається прогрес, пов'язаний з розширенням різноманіття пропонованих схемотехнік на ринку електронних технологій, яке можна представити трьома напрямками: мікропроцесори (CPU), замовні БІС (ASIC – Application Specific Integration Circuit, VLSI – Very Large Scale Integration) і програмована логіка (FPGA, CPLD). Перші два, поряд з високою швидкодією, мають недоліки,

пов'язані зі значними тимчасовими і фінансовими витратами проектування обчислювальних пристроїв. Щодо третього типу, то сучасний рівень розвитку субмікронних технологій мікроелектроніки дає змогу створювати на одному кристалі програмованої логічної інтегральної схеми (ПЛІС) надскладні цифрові системи, що містять 5–8 мільйонів еквівалентних елементів. Це дає можливість проектувати складні спеціалізовані обчислювальні пристрої протягом 2–4 тижнів за допомогою засобів автоматизованого проектування відомих фірм, таких як: Aldec, Cadence, Altera, Xilinx, Synopsys, Mentor Graphics.

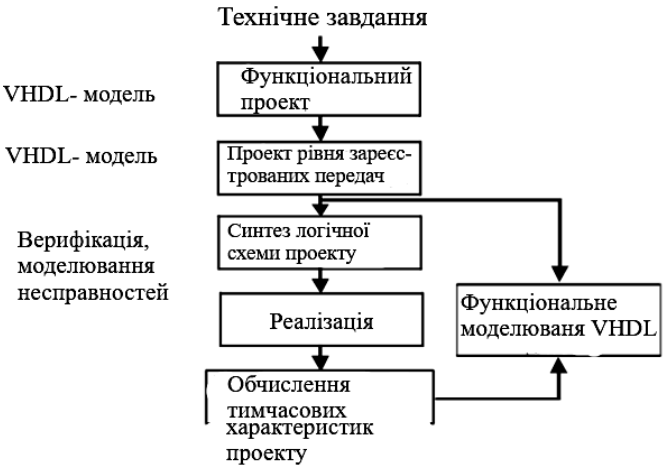


Рисунок 3.2 – Етапи розробки проекту для FPGA

Рисунок 3.3 – Узагальнена структура стадій проектування

На сьогодні світовими лідерами у сфері виробництва програмованої логіки є фірми Xilinx (FPGA – Virte, Spartan), Altera (CPLD, FLEX, Lattice / Vantis (SPLD, PAL, GAL), CPLD), Actel (FPGA), Cypress, Atmel, Quicklogic. Максимальні параметри програмованої логіки: тактова частота – 500 МГц, число еквівалентних елементів – 9 млн, обсяг пам'яті – 400 Кбіт, підтримує Boundary Scan. Обсяг продажу програмованої логіки на ринку мікроелектроніки становить понад 60 млрд дол. на рік. Лідером є фірма Xilinx – 28 млрд дол. Спільний розподіл ринку ПЛІС між цими фірмами складає: Xilinx – 52 %, Altera – 34 %, Lattice – 8 %, Actel – 6 %.

Щодо коштів автоматизованого проектування (Electronic Design Automation), тут лідирують фірми, що працюють у трьох

напрямках: 1) введення і моделювання цифрових проектів (Design Entry & Simulation): Aldec (20000 instalations), Mentor Graphics (12000 instalations); 2) синтез і імплементація (Synthesis): Synopsys (74 % продажу на ринку САПР), Mentor Graphics (15 %), Synplicity (11 %); 3) тестування і верифікація (Digital System Testing): Logic Vision, Mentor Graphics, Synopsys, Syntest, Simucad, Cadence.

Суттєвим моментом при сучасному проектуванні на програмованій логіці є ієрархічне подання проекту у взаємозв'язку з двома найбільш популярними мовами опису апаратури. Типовою класифікацією є поділ рівнів, відповідний до технологій проектування (рисунок 3.4):

Рисунок 3.4 – Етапи проектування і мови опису апаратури

1 Системне (алгоритмічне) проектування. Характеризується високим рівнем абстракції опису моделей мовами: VHDL, Verilog, System C. Швидкість виконання проектів – більше 1 млн елементів на місяць на одного проектувальника, складність проектів – 105–106 елементів. Обсяг проектів на ринку – 60 %. Конструктор спільного апаратно-програмного проектування Hardware-Software Cooperation Design і стратегія повторного застосування готових проектів – IP-core reuse.

2 Схемотехнічне проектування. Опис моделей на RTL (Register transfer level) рівні (рівні регістрових передач) за допомогою мов: VHDL (більш універсальний і академічний, популярний в Європі, в середовищі університетських вчених), Verilog (більш проста мова, промислово-орієнтована, популярна в Японії, США, в списку користувачів фірми SUN, Apple, Motorola), System C (у стадії розробки компіляторів і тестування),

EDIF (стандарт внутрішнього опису цифрових проектів та подання схем після синтезу). Швидкість виконання проектів складає близько ста тисяч елементів на місяць на одного проектувальника, складність проектів – 104 елементів. Обсяг виконуваних на ринку проектів 40 %.

З Логічне проектування цифрових систем. Опис моделей на елементному рівні у вигляді булевих рівнянь за допомогою таких мов: VHDL, Verilog, System C, EDIF. Використовується самостійно для створення нескладних проектів (близько 104 елементів), що потребують надвисокої роздільної швидкодії і мінімальних витрат апаратури. На сьогодні цей рівень розглядається як автоматична процедура синтезу цифрової системи.

У зв'язку з постійним удосконаленням технологій виробництва інтегральних схем виникає можливість розміщувати більшу кількість елементів в одній мікросхемі. Цифрові системи все більш ускладнюються. Детальне проектування таких пристроїв на елементному (логічному) рівні стає практично неможливим. З цієї причини зростає значущість мов опису апаратури. Вони дають змогу розробляти і верифікувати цифрові пристрої на верхніх рівнях до перетворення в логічний. Цей підхід подібний до написання програмних засобів мовами програмування (Сі) і використання компілятора для перекладу такої програми в машинний код. Дві найбільш популярні мови опису апаратури – VHDL і Verilog.

VHD – це абревіатура англійського виразу VHSIC Hardware Description Language. У свою чергу VHSIC походить від назви програми Very High Speed Integrated Circuit (високошвидкісні інтегральні схеми). Ця програма, фінансована Міністерством Оборони США, ставила собі за мету розвиток нового покоління високошвидкісних інтегральних схем. Перша версія мови була представлена в 1985 р. Згодом її було передано IEEE для стандартизації. У 1987 р. мову було затверджено як стандарт IEEE 1076 – 1987. Через п'ять років її було розглянуто повторно, в результаті чого нова версія 1076-93 містить низку додаткових можливостей. VHDL має популярність, що постійно зростає серед фахівців з автоматизованого проектування електронних систем. Перші VHDL-додатки виникли на початку 90-х років.

Для синтезу було розроблено пакет IEEE 1164. На практиці кожен великий виробник систем автоматизованого проектування підтримує VHDL. Мову VHDL дає змогу описувати системи на різних рівнях. Вона реалізує методологію спадного проектування, в якій система спочатку описується на високому рівні і тестується за допомогою засобів моделювання, після чого поетапно приводиться до структурного опису, тісно пов'язаного з фактичною апаратною реалізацією. Мова VHDL було розроблено незалежною для технології реалізації. Якщо якийсь проект, описаний у VHDL, реалізований для конкретної електронної технології, то цей самий VHDL-опис може бути використано як вихідний для реалізації пристрою в новій технології.

Verilog, або Verilog HDL, – мова опису апаратури, розроблена в 1985 р. Філіпом Мурба (Philip Moorby), що є простим наочним і ефективним засобом для опису цифрових схем, моделювання та аналізу їх функціонування. Мова стає власністю Gateway Design Automation, потім Cadence Design Systems. У 1990 р. Cadence відкриває мову і приводить її до стандартизації IEEE в 1995 р.

Яка мова є кращою: VHDL або Verilog? Verilog легша в розумінні і використанні. Вона застосовувалася і застосовується в промисловості, де потрібне моделювання та синтез. Але вона має недолік у конструкціях для опису рівнів системи. Verilog здобула визнання в проектуванні ASIC схем, особливо для проектів низького рівня (регістрових передач і нижче). Вона найбільш популярна в Північній Америці, Азії і особливо в Японії.

VHDL – більш складна мова, її важче вивчати і використовувати. Тим не менш, вона має більшу гнучкість, що є її перевагою і недоліком через велику кількість допустимих стилів коду. Оскільки VHDL краще підходить для роботи з дуже складними проектами, сьогодні її популярність зростає. Особливо це стосується Європи, США і Канади. Мова не має успіху у Японії. Однак у світі все більше і більше віддають перевагу мові VHDL.

Інструментальне середовище ALDEC ACTIVE-HDL – повністю інтегроване середовище для розробки і моделювання ПЛІС мовами VHDL, Verilog і EDIF заснованих проектів. Програмна оболонка Active-HDL є передовим засобом у галузі

середовищ для моделювання і верифікації HDL проектів з великою кількістю елементів. Оскільки в промисловості розмір програмованих мікросхем неухильно зростає, то потреба у високопродуктивних інструментальних засобах перевірки проектів з великою кількістю елементів набуває все більшої і більшої актуальності. Active-HDL Standard Edition (Стандартна версія) була розроблена для інженерів, які розробляють FPGA / CPLD (ПЛІС / ПЛМ) проекти і яким потрібен потужний моделювальний пристрій для опису цих проектів на рівні регістрових передач, більш того, цей засіб має бути реалізовано в операційному середовищі PC Windows. Active-HDL підтримує всі інструментальні засоби і бібліотеки незалежно від фірм-виробників інтегральних схем.

Active-HDL містить три різні програми для введення проектів, VHDL'93 і Verilog компілятори, єдине ядро моделювання, програмні модулі налагодження, графічне і текстове виведення результатів моделювання, допоміжні інструменти для зручного управління файлами ресурсів, проектами і бібліотеками

В Active-HDL містяться також кілька потужних майстрів для полегшення створення нових вихідних файлів, проектів і випробувальних стендів TestBench. Більшість операцій, які викликаються через графічний користувальницький інтерфейс, можна виконувати за допомогою команд макромови Active-HDL. Командні файли дають змогу автоматизувати процес проектування.

Інструментальне середовище **Xilinx Progect Navigator** пакет **Xilinx ISE** – повністю інтегроване середовище для здійснення синтезу, імплементації на ПЛІС, тимчасового моделювання. Підтримує VITAL-стандарт (Стандарт – IEEE P1076.4: Timing Methodology (VITAL)), представлений пакетом, що містить типи для докладного опису pin-to-pin і distributed затримок, тимчасові константи, включаючи максимальне, мінімальне та номінальні значення. Пакет також містить процедури для виконання тимчасового аналізу. Основним змістом першого етапу є перехід від словесного опису лічильника і шифратора до його математичної моделі, використовуючи карти Карно і мінімізацію.

Після цього можна переходити до аналізу об'єкта і постановки завдання проектування.

При цьому необхідно визначити істотні, з точки зору дослідження, складові процесу функціонування об'єкта і мету його дослідження. Постановка завдання є одним з найбільш важливих моментів, що визначають принципову можливість здійснення моделювання та пов'язані з цим витрати. В результаті має бути побудована функціональна схема (діаграма потоків) об'єкта, що моделюється.

Після цього можна переходити до встановлення основного змісту моделі. На підставі функцій і структури об'єкта моделювання, характеру взаємодії його елементів і впливів зовнішнього середовища вибирається підхід до вирішення задачі моделювання і відповідна типова математична схема, в рамках якої і буде здійснюватися моделювання об'єкта.

Необхідно зобразити структуру (технологію функціонування) об'єкта моделювання в символіці обраної типової математичної схеми і надати її опис, записати математичні або логічні відношення між змінними і параметрами моделі.

На етапі алгоритмізації і машинної реалізації необхідно перетворити модель, отриману на першому етапі в машинну модель (VHDL-опис). При виборі мови моделювання слід керуватися її пристосованістю для опису моделей обраного (даного) класу і доступністю (наявністю компіляторів на наданих для виконання курсового проекту комп'ютерах). У даному курсовому проекті можна використовувати VHDL, Verylog і подібні до них.

При програмуванні моделі слід використовувати блоковий принцип побудови програм. Кожен блок програми має відповідати (описувати) окремий блок об'єкта або складову частину процесу функціонування цього блока. Це зробить програму більш наочною і спростить її налагодження. Перевірку працездатності (тестування) програми доцільно проводити на серії контрольних прикладів з граничними або звичайними значеннями вихідних даних, результати моделювання для яких можуть бути легко перевірені: відсутність вхідних сигналів; надходження на вхід тільки одного сигналу; гранично висока

продуктивність елементів об'єкта; свідомо певний хід процесу (повний простий, повна відмова в обробці, перехід до детермінованого процесу та ін.).

На етапі отримання і інтерпретації результатів моделювання в першу чергу необхідно провести планування експериментів зі створеною на попередньому етапі у вигляді програми машинною моделлю. При цьому в обов'язковому порядку слід визначити необхідну кількість експериментів, число реалізацій моделювального алгоритму, здійснити вибір початкових умов або зменшити вплив початкових умов на результати моделювання, оцінити точність отриманих результатів.

Після визначення трудомісткості моделювання, необхідної пам'яті і переліку зовнішніх пристроїв проводяться комп'ютерні експерименти з моделлю. Результати моделювання можуть виводитися у вигляді таблиць, графіків, діаграм, схем тощо і можуть містити, крім характеристик, перелічених у завданні на курсову роботу, інші дані, які стосуються об'єкта моделювання та (або) дають змогу судити про коректність отриманих результатів.

Основним показником коректності результатів моделювання служить їх відповідність результатам наближених або аналітичних розрахунків.

3.4.2 Синтез двійково-десяткового лічильника

Лічильник за модулем 10 описується логічною схемою, наведеною на рисунку 3.5. JK-тригер описується логічною схемою, поданою на рисунку 3.6. Тригером типу D (Delay – затримка) називається тригер з двома стійкими станами рівноваги і одним інформаційним входом D (рисунок 3.7). Значення, що надходять на вхід D, записуються на вихід Q, тобто тригер працює як повторювач.

Повною таблицею переходів, за допомогою якої можна побудувати скорочену таблицю переходів, є таблиця 3.2. Характеристичне рівняння D-тригера має вигляд: $Q(t+1) = D(t)$. Тригером типу JK називається тригер з двома стійкими станами рівноваги і двома інформаційними входами (рисунок 3.8). Вхід J (Jarк) служить для установлення тригера в «1», вхід K (KILL) – для установлення в «0». Активним значенням сигналу на вході є

рівень 1. Одночасна подача двох активних сигналів на входи K і J не заборонена, при цьому на виході з'являється інверсне значення стану тригера $\overline{Q}^t$.

Подача двох нулів на входи тригера зберігає його внутрішній стан.

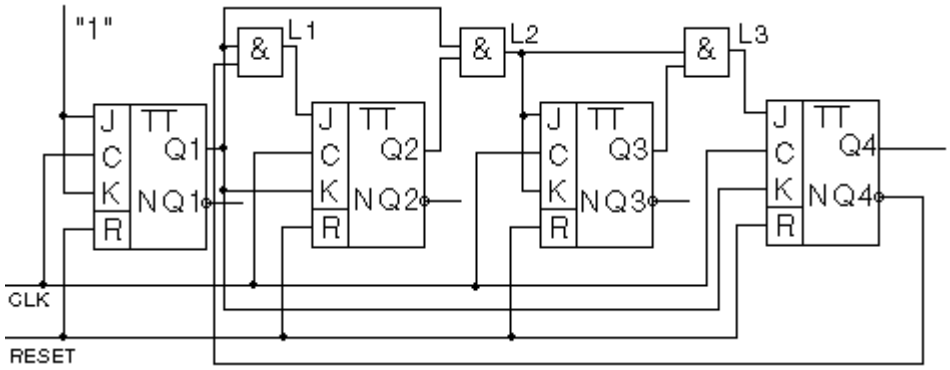
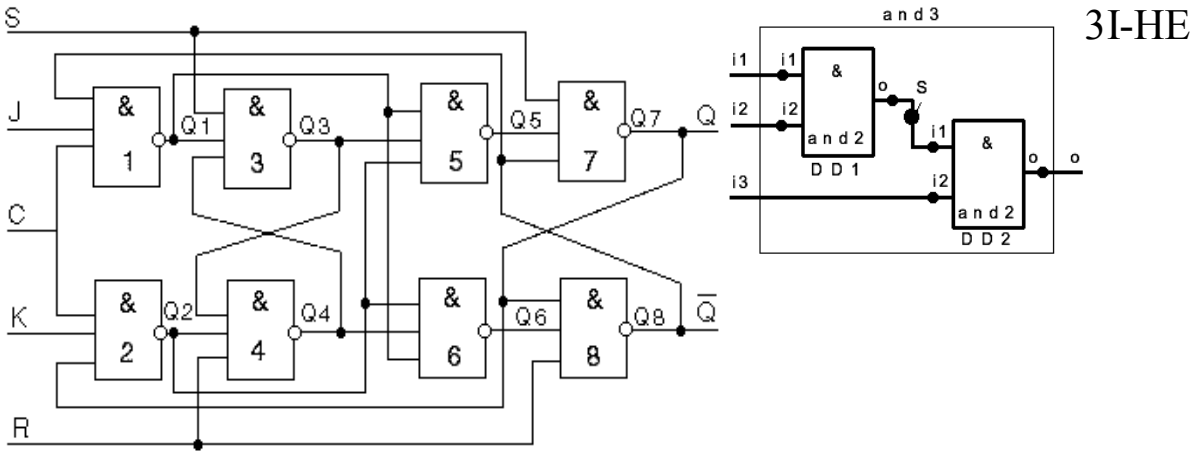


Рисунок 3.5 – Логічна схема лічильника за модулем 10



3І-НЕ

Рисунок 3.6 – Логічна схема JK-тригера

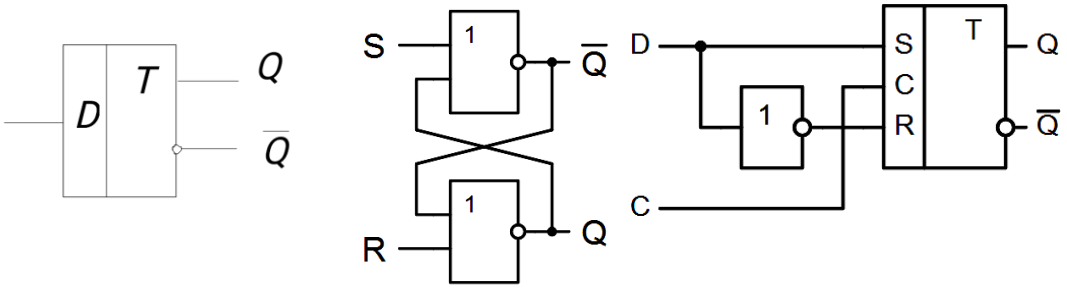


Рисунок 3.7 – Умовне позначення асинхронного D-тригера

Таблиця 3.2 – Повна і скорочена таблиця і матриця переходів D-тригера

t		$t+1$	
D	Q	Q	
0	0	0	
0	1	0	
1	0	1	
1	1	1	

D	$Q(t+1)$
0	0
1	1

$Q(t)$-$Q(t+1)$	D
0-0	0
0-1	1
1-0	0
1-1	1

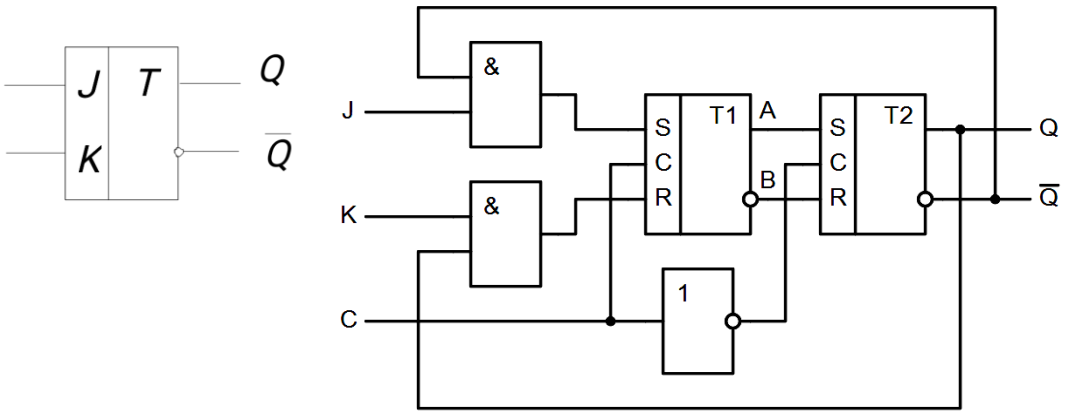


Рисунок 3.8 – Умовне позначення асинхронного JK-тригера

Повною таблицею переходів (ПТП), за допомогою якої можна побудувати скорочену таблицю переходів, є таблиця 3.3.

Таблиця 3.3 – Повна і скорочена таблиця переходів JK -тригера і карта Карно для функції переходів JK-тригера

t			$t+1$	
K	J	Q	Q	
0	0	0	0	1
0	0	1	1	4
0	1	0	1	2
0	1	1	1	4
1	0	0	0	1
1	0	1	0	3
1	1	0	1	2
1	1	1	0	3

K	J	$Q(t+1)$
0	0	$Q(t)$
0	1	1
1	0	0
1	1	$\overline{Q}(t)$

$J_tK_t \setminus Q_t$	0	1
00	0	1
01	1	1
11	1	0
10	0	0

		Q(t+1)					
Z(t)	Q(t)	Z1	Y	X=0	Z0	Y	X=1
	$Q_1Q_2Q_3Q_4$			$Q_1Q_2Q_3Q_4$			$Q_1Q_2Q_3Q_4$
	8421			8421			8421
0	0000	1	0	0001	9	1	1001
1	0001	2	0	0010	0	0	0000
2	0010	3	0	01011	1	0	0001
3	01011	4	0	0100	2	0	0010
4	0100	5	0	0101	3	0	01011
5	0101	6	0	0110	4	0	0100
6	0110	7	0	0111	5	0	0101
7	0111	8	0	1000	6	0	0110
8	1000	9	0	1001	7	0	0111
9	1001	0	1	0000	8	0	1000

Використовуючи карту Карно, можна знайти мінімальну ДНФ булевої функції для опису функціонування JK-тригера (характеристичну функцію переходів) $Q^{JK}_{(t+1)} \cdot Q^{JK}_{(t+1)} = J_t\overline{Q}_t \vee \overline{K}_t Q_t$.

Виходячи з технічного завдання потрібно спроектувати схему реверсивного двійково-десяткового лічильника, який веде рахунок у коді 8-4-2-1 у такій послідовності: 1,2,3,4,5,6,7,8,9,0. Початковий стан «1». Імпульс перенесення формується при переходах 0-1 при прямому ході і 1-0 при зворотному. Стан лічильника кодується шифратором коду 2 з 5 в наступному відповідно: 0 = 1, 1 = 2, 2= 3, 3 = 4, 4 = 5, 5 = 6, 6 = 7, 7 = 8, 8 = 9, 9 = 9. Використовувати необхідно тригер типу D або JK, логічні елементи «І-НЕ». Виконати аналіз функціональної схеми лічильника і побудувати граф-схему його аналітичної моделі. Всі отримані функції синтезувати в середовищі ISE і надати всі роздруківки.

Відповідно до технічного завдання будуємо таблицю 3.3 переходів – виходів. Q (t) – поточний стан; Q (t + 1) – подальший стан; Y (t) – імпульс перенесення; X = 0 і X = 1 – режими рахунку в прямому і зворотному порядку відповідно. Побудуємо карти Карно для функцій Q1, Q2, Q3, Q4 Y (рисунок 3.9, а-д).

а) Q_1

	X=0				X=1			
$Q_1Q_2\backslash Q_3Q_4$	00	01	11	10	00	01	11	10
00	X	X	0	X				
01	0	0	0	0				
11	1	X	X	X				

б) Q_2

$Q_1Q_2\backslash Q_3Q_4$ X	000	001	011	010	100	101	111	110
00	X	X	0	X				
01	0	0	1	0				
11	1	X	X	X				
10	0	0	0	0				

в)Q_3

$X_1X_2\backslash X_3X_4$	00	01	11	10
000	X	X	0	X
001	0	1	1	1
011	1	X	X	X
010	0	0	1	1
100				
101				
111				
110				

г) Q_4

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

Рисунок 3.9 – Карта Карно для функцій

Набори, на яких значення функції не визначено, позначимо «X» (функція в повному обсязі визначена). Об'єднавши елементи в картах Карно, складаємо мінімальну диз'юнктивну нормальну форму (МДНФ) функції:

$Q_1(t+1) = X_1$, Q2(t+1): $Q_2(t+1) = X_1X_2 \vee X_2X_3X_4$,

Q3(t+1): $Q_3(t+1) = X_1X_2 \vee X_2X_4 \vee X_2X_3 \vee X_1X_3$,

Q4(t+1): $Q_4(t+1) = X_1X_4 \vee X_2X_3 \vee X_2\overline{X}_4$, Y0(t+1): $Y_0 = X_1$.

Для D-тригера ці рівняння підставляємо без зміни, тому що його рівняння $Q_1(t+1) = D(t)$. Для JK-тригера, користуючись рівняннями, поданими вище, отримуємо $Q(t+1) = \overline{Q}(t) \cdot J + Q(t) \cdot \overline{K}$. Для D-тригера ці рівняння підставляємо без зміни. Потім перейдемо до базису І-НЕ. За цими даними будуємо електричну принципову схему синхронного реверсивного двійково-десяткового лічильника на тригерах.

Тригером називається логічна схема з позитивним зворотним зв'язком, що має два стабільних стани, що називаються одиничним і нульовим і позначаються 0 і 1 (a0 і a1 на графі). У загальному випадку тригер – це пристрій з двома стійкими станами, який містить запам'ятовувальний елемент (ЗЕ) і комбінаційну схему управління (КС) (рисунок 3.10), де X1 ... Xn – інформаційні входи тригера; C1, Cm – синхронізовані входи; Q і •Q – відповідно прямий і інверсний виходи тригера; f1 та f2 – функції збудження ЗЕ. Переведення тригера в одиничний стан шляхом впливу на його входи називають установленням (**Set**) тригера, а сигнал, який впливає на вхід, позначають S. Переведення тригера в нульовий стан називають скиданням або гасінням (**Reset**), а відповідний сигнал на вході позначають R.

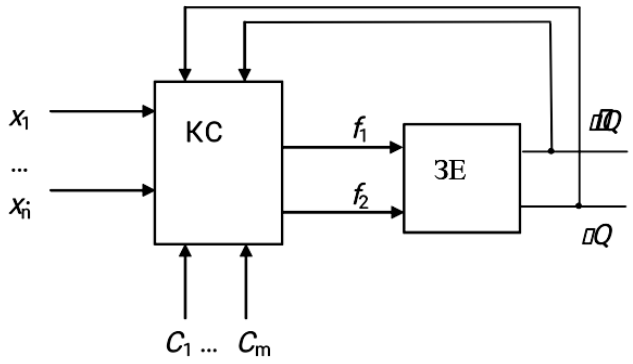


Рисунок 3.10 – Структурна схема тригера

Іншими позначеннями входів найбільш поширених тригерів можуть бути: **C** – вхід синхронізації; **D** (delay) – затримка (вхід тригера D); **T** (toggle) – перемикач (вхід тригера T); **J** (Jarк) – вхід синхронної установки універсального JK-тригера в '**1**'; **K** (KILL) – вхід синхронної установки універсального JK – тригера в "**0**"; **V** – вхід, за яким DV-тригер налаштовується на виконання функції D-тригера. За наявності n інформаційних входів можна отримати 5N (N = 2n) типів тригерів. На практиці використовують невелику кількість типів тригерів. До них можна віднести RS, JK, T, D, S, R, E-тригери. У нашому курсовому проекті ми використовуємо D і JK-тригери. D-тригер, синхронізований рівнем 1, зображено на рисунку 3.11. JK-тригер, синхронізований переднім фронтом зі скиданням в 0, зображено на рисунку 3.12.

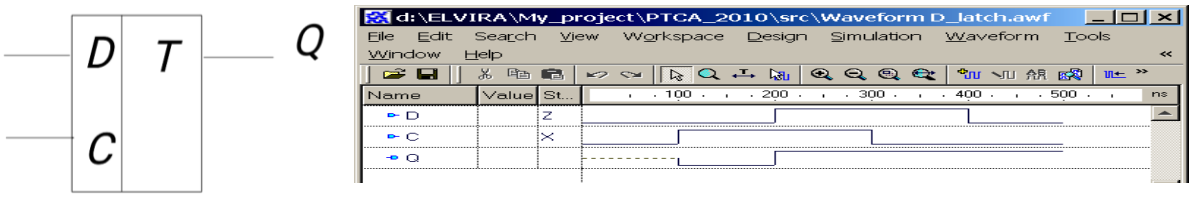


Рисунок 3.11 – *D*-тригер, синхронізований рівнем 1, та часова діаграма його роботи

Лістинг 3.1 – *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;
```

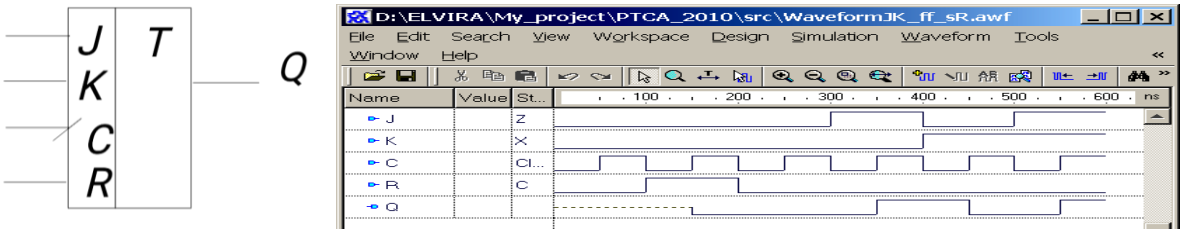


Рисунок 3.12 – *JK*-тригер, синхронізований переднім фронтом із синхронним скиданням в 0, та часова діаграма його роботи

Лістинг 3.2 – *JK*-тригер, синхронізований переднім фронтом зі 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 передньому 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;
```

3.4.3 Проектування шифратора

Розробляємо перетворювач кодів – шифратор. Структурну схему шифратора двійково-десяткових чисел подано на рисунку 3.13. Кодуємо виходи лічильника шифратором «2 з 5» (таблиця 3.3). Будуємо графоаналітичну модель реверсивного двійково-десяткового лічильника для всіх наборів (таблиця 3.4). Відповідно до цієї таблиці будуємо графоаналітичну модель лічильника для всіх наборів.

Рисунок 3.13 – Структурна схема шифратора двійково-десяткових чисел

Таблиця 3.4 – Таблиця істинності кодувальника коду – шифратора «2 з 5»

Y	Y_1	Y_2	Y_3	Y_4	Y_5
0	1	1	0	0	0
1	0	1	1	0	0
2	0	0	1	1	0
3	0	0	0	1	1
4	1	0	0	0	1
5	1	0	1	0	0
6	0	1	0	1	0
7	0	0	1	0	1
8	1	0	0	1	0
9	0	1	0	0	1

Z(t)	Q(t)	Z1	Y_1	Y_2	Y_3	Y_4	Y_5
	$Q_1Q_2Q_3Q_4$						
	8421						
0	0000	1	0	1	1	0	0
1	0001	2	0	0	1	1	0
2	0010	3	0	0	0	1	1
3	0011	4	1	0	0	0	1
4	0100	5	1	0	1	0	0
5	0101	6	0	1	0	1	0
6	0110	7	0	0	1	0	1
7	0111	8	1	0	0	1	0
8	1000	9	0	1	0	0	1
9	1001	0	1	1	0	0	0

	Z(t)	Q(t)	Z1	Y_0	Q(t+1),y(t)		Q(t)	Q(t+1),y(t)		Y_1	Y_2	Y_3	Y_4	Y_5
		$Q_1Q_2Q_3Q_4$			X=0	X=1	8421	X=0	X=1					
Основний	0	0000	1	0	0001,1	1	a0	0	1	0	1	1	0	0
	1	0001	2	0	0010,0	1	a1	0	2	0	0	1	1	0
	2	0010	3	0	0011,0	0	a2	1	3	0	0	0	1	1
	3	0011	4	1	0100	0	a3	1	4	1	0	0	0	1
	4	0100	5	1	0101	1	a4	0	5	1	0	1	0	0
	5	0101	6	0	0110	0	a5	0	6	0	1	0	1	0
	6	0110	7	0	0111	1	a6	1	7	0	0	1	0	1
	7	0111	8	1	1000	0	a7	0	8	1	0	0	1	0
	8	1000	9	0	1001	0	a8	1	9	0	1	0	0	1
	9	1001	0	1	0000	0	a9	0	0	1	1	0	0	0
Не основний	10	1010												
	11	1011												
	12	1100												
	13	1101												
	14	1110												
	15	1111												

3.4.4 Основні етапи автоматизованого проектування пристроїв на ПЛІС

Синтез проекту – створення схеми, тобто перехід від функціонально коректного опису проекту мовою VHDL до списку з'єднань (netlist), тобто до машинно-орієнтованої схематики.

Імплементація проекту – розміщення схеми на цільовій мікросхемі. Імплементація проекту – це послідовність подій, яка переводить netlist у файл програмування для пристрою FPGA. Синтезований проект має ряд портів на верхньому рівні. Засобам імплементації необхідно знати, яким чином відводити порти проекту під фізичні виводи мікросхеми FPGA, яка приєднується до різних ресурсів плати Spartan-3E Starter Kit. Якщо не зробити явних призначень, засоби будуть підключені до виводів випадковим чином. Однак це поганий варіант, оскільки випадкове призначення може бути не правильним (рисунок 3.14).

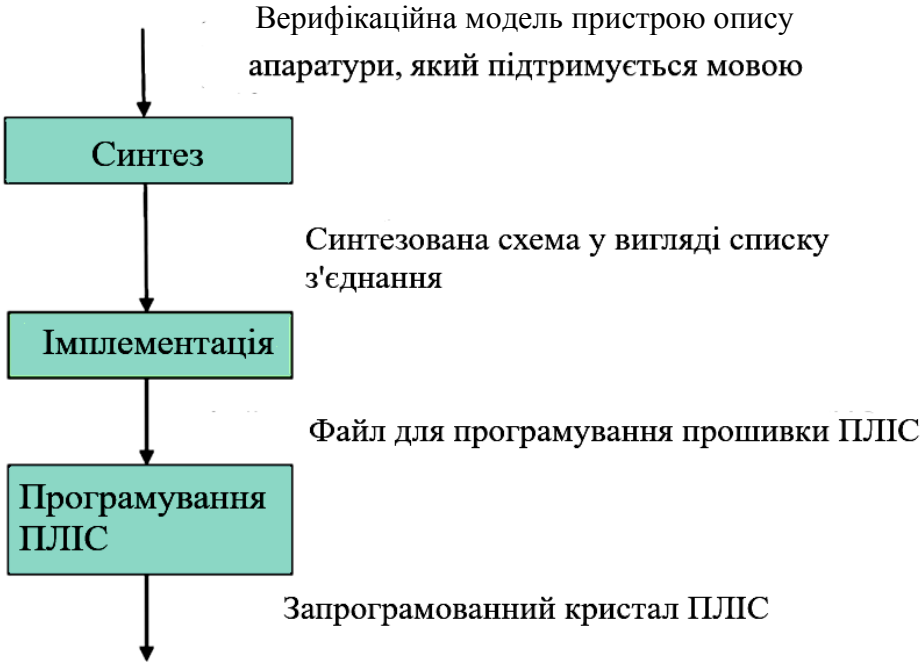


Рисунок 3.14 – Основні етапи автоматизованого проектування пристроїв на ПЛІС

3.4.5 Створення VHDL-проектів спроектованих пристроїв лічильника і шифратора

На вхід системи автоматизації проектування надходить верифікаційний опис пристрою на підтримуваній мовою опису

апаратурі (МОА), наприклад, VHDL-опис (файл з розширенням vhd). Верифікація – перевірка правильності роботи моделі об'єкта. Мова VHDL служить для опису моделі цифрового пристрою (приладу, системи). Опис мовою VHDL визначає зовнішні зв'язки пристроїв («вид зовні» або інтерфейс) і один або кілька «видів зсередини» (див. рисунок 3.15).

Вигляд зовні задає інтерфейс пристрою, набір сигналів, якими обмінюється із зовнішнім світом. Цей вигляд описує абстрактне уявлення пристрою «в цілому» і позначається англійським терміном entity, що в дослівному перекладі означає «сутність» і найбільш точно відображає зміст зображуваного. Однак у літературі термін «сутність» широко не застосовується, для позначення зовнішнього опису об'єкта використовуються терміни «інтерфейс об'єкта», «декларативна частина» та інші. У цьому посібнику буде використовуватися термін «інтерфейс об'єкта» або просто «інтерфейс».

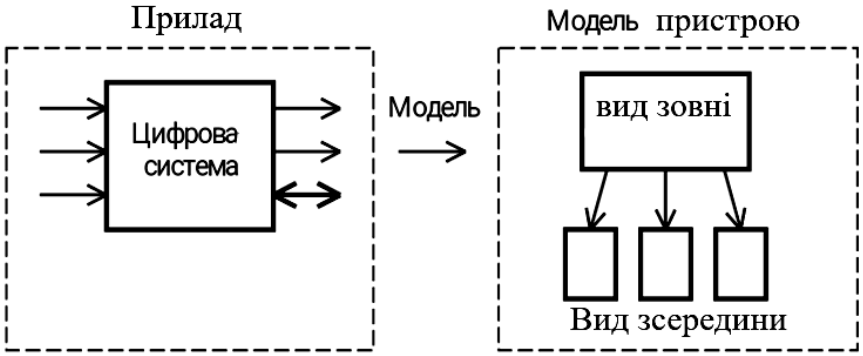


Рисунок 3.15 – Цифровий пристрій і його модель

Вигляд зсередини визначає функціональні можливості пристрою або його структуру. Внутрішню будову об'єкта визначає архітектура (architecture body).

Як і в мовах програмування, мова VHDL має свої правила, в тому числі правила опису імен змінних, об'єктів, типів даних і інших параметрів. Основні правила мови VHDL описано в наступних розділах.

Спочатку мову VHDL задумували як мову моделювання, тому пізніше при створенні автоматизованого синтезу схеми з моделі мовою опису апаратури довелося ввести низку обмежень, оскільки не всі конструкції мови могли бути надані апаратурою.

Різні програми синтезу, створені різними фірмами, можуть працювати з різними синтезованими підмножинами мови. У 1999 р. американським товариством стандартизації IEEE випущено стандарт щодо виконання синтезу з VHDL (IEEE Std 1076.6-1999IEEE Standard for VHDL Register Transfer Level Synthesis). З удосконаленням програм синтезу деякі обмеження, можливо, будуть змінюватися або повністю зніматися.

Усі конструкції мови, які не рекомендується використовувати в моделях, призначених для синтезу, діляться на ігноровані і на такі, що не підтримуються. Ігноровані конструкції вилучаються з моделі в момент її аналізу, а такі, що не підтримуються, – призводять до виникнення помилки синтезу.

Інтерфейс (Entity) підтримується, проте використовувані в інтерфейсі оператори ігноруються, тому що вони описують пасивну поведінку схеми при її моделюванні і не впливають на зміну значення сигналів, які не підтримують декларативну частину інтерфейсу.

```
entity identifier is
entity_header
entity_declarative_part
[ begin
entity_statement_part ]
end [entity] [entity_simple_name];
```

Задаються в інтерфейсі generic-константи і порти, які підтримуються, а ініціалізація значення портів ігнорується.

Архітектура (Architecture) підтримується. Також дозволяється виконувати множинні архітектури, відповідні одному інтерфейсу, але не підтримується глобальна взаємодія сигналів між архітектурою. У декларативної частини архітектури декларації файлів, псевдонімів (Alias), конфігурацій, Disconnection і визначені користувачем атрибути ігноруються. Не підтримуються сумісно використовувані змінні, шаблони і декларації груп. У конфігурації підтримується тільки її декларація, всі інші її складові не підтримуються.

```
    configuration identifier of entity_name is
    ~~configuration_declarative_part~~
    block_configuration end [configuration]
[configuration_simple_name];
    ~~configuration_declarative_part~~    ::=
use_clause | attribute_specification |
group_declaration
```

Декларація конфігурації має підтримуватися, щоб дати можливість виконувати опис архітектури, пов'язаної з інтерфейсом верхнього рівня синтезованого проекту.

Напишемо VHDL-код спроектованих вище пристроїв (лічильника (рисунок 3.16) і шифратора) за МДНФ, отриманим за картами Карно (рисунок 3.9, а–д) використовуючи логічні оператори not, and, or. Скористаємося кодом для нашої схеми лічильника, яка має 5 входів, 5 виходів і описана булевими рівняннями:

$$Z_1 = X_1,$$
$$Z_2 = X_1X_2 \vee X_2X_3X_4,$$
$$Z_3 = X_1X_2 \vee X_2X_4 \vee X_2X_3 \vee X_1X_3,$$
$$Z_4 = X_1X_4 \vee X_2X_3 \vee X_2\overline{X}_4,$$
$$Y_0 = X_1.$$

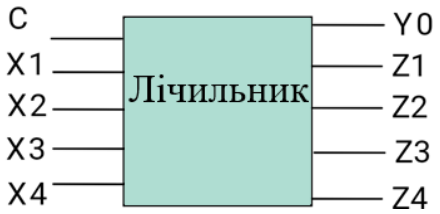


Рисунок 3.16 – Приклад схеми лічильника

Аналогічно створюємо VHDL-проект спроектованого пристрою дешифратора.

3.4.6 Створення VHDL-проектів спроектованих пристроїв лічильника і шифратора в автоматизованому середовищі Xilinx ISE

Створіть новий проект у середовищі Project Navigator, виберіть Start → Programs → Xilinx ISE → Project Navigator. У Project Navigator виберіть File → New Project.

Проект вже створений, але він не містить жодного вихідного файлу. Інакше натисніть на «Finish», щоб завершити процес. Після цього створіть новий вихідний файл для свого лічильника.

Або виберіть Project → New Source з головного меню, або використовуйте еквівалентний процес у вікні Processes. Відкриється перше вікно з діалогових вікон для нового вихідного файлу New Source.

Виберіть VHDL Module, щоб вказати, що ви створюєте модуль проекту, описаного VHDL. Потім введіть ім'я проекту. Ви не повинні змінювати локалізацію цього файлу, який має бути всередині директорію, створеного раніше. Натисніть «Далі». Наступне діалогове вікно дає вам змогу визначити (призначити) порти модуля. Це може бути зроблено також у текстовому редакторі, коли створюється модуль, щоб пропустити цей крок. Просто підтвердьте установки відповідно до тих. Заключне в цьому ланцюжку діалогове вікно забезпечує підведення підсумків за вказаними установками. Перевірте інформацію, щоб переконатися в її правильності. Для коригування установок натисніть «Назад», інакше натисніть «Готово», щоб завершити процес. Новий створений файл має 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 є базова файлова структура, але доведеться замінити всі, що автоматично згенеровано. Скопіюйте лістинг (VHDL опис елемента XOR), розташований нижче, 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.

Лістинг 3.3 – VHDL опис елемента XOR

```
- Підключення бібліотеки IEEE.
library IEEE;
-- Подключення 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 C, X1, X2,
X3, X4, виходи Z1, Z2, Z3, Z4, Y0,)
```

```
-- Counter – ім’я пристрою
entity Counter is
port (
    C, X1, X2, X3, X4: in STD_LOGIC;
    Z1, Z2, Z3, Z4, Y0: out STD_LOGIC);
end Counter;
architecture Counter of Counter is
begin
  Z1 <= X1;
  Z2 <= (X1 and X2) or (X2 and X3 and X4);
  Z3 <= (X1 and X2) or (X2 and X4) or (X2 and
X3) or (X1 and X3);
  Z4 <= (X1 and X4) or (X2 and X3) or (X2 and
not X4);
  Y0 <= X1;
end Counter;
```

Збережіть проект. Існують опції головного меню, що дозволяють зберігати окремі файли і проект цілком. Виконайте синтез 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 називається XST, який інтегровано в Project Navigator і націлено тільки на пристрої фірми Xilinx.

Щоб задати процес синтезу, встановіть у вікні Sources Synthesis / Implementation. Якщо ви виберете вихідний файл counter, ви маєте побачити Synthesize-XST як процес, наявний у вікні Processes. Опції синтезу можна змінити до власне синтезу натисканням правої клавіші миші на Synthesize-XST і вибором Properties. Однак у даний момент не потрібно нічого змінювати, залиште опції, встановлені за замовчуванням. Подвійним натисканням на Synthesize-XST запустіть синтез. Коли синтез завершиться, ви побачите зелену контрольну мітку.

Натиснувши на + біля Synthesize-XST, відкрити меню синтезу, в якому View Synthesize Report, View RTL Schematic, View technology Schematic. Натиснувши на рядок View Design Summary у вікні Processes, ви побачите вікно. 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 LUT's. Ви маєте зараз побачити, що жодної помилки у вікні Console немає, однак ви завжди повинні переглядати log file, який можна побачити, розкриваючи процес Synthesize-XST натисканням на «+». Виберіть View Synthesis Report. Якщо вам незрозуміло певне повідомлення, не ігноруйте його. Замість цього, зверніться на сайт підтримки Xilinx або зверніться до інструктора.

Переглядаючи звіт, можна з'ясувати, якого типу (і як багато) ресурсів 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 допущено помилку. Ось чому ви маєте знатися на проектуванні апаратних засобів, які ви намагаєтеся створити, коли пишете мовний опис проекту.

3.4.7 Імплементація проекту

Імплементація проекту – це послідовність подій, які переводять ваш netlist у файл програмування для пристрою FPGA. Ваш проект, який ви зараз синтезували, має ряд портів на верхньому рівні. Засобам імплементації необхідно знати, яким чином відводити порти проекту під фізичні виводи мікросхеми FPGA, яка приєднується до різних ресурсів плати Basys2. Якщо ви не зробите явних призначень, засоби будуть підключені до виводів випадковим чином. Однак це неслушна ідея, оскільки випадкове призначення може бути неправильним.

Наш проект має два вхідних порти і один вихідний порт. Нам потрібно буде два перемикачі SW0 і SW7, що приєднуються до входів.

Якщо ви уважно подивитеся на плату, то звернете увагу на те, що багато ресурсів дбайливо забезпечені коментарями з текстовою індикацією, яка пов'язана з виходами FPGA. Ця інформація також є в Basys2 у вигляді таблиць і схематичних форм. Виконати імплементацію проекту, створивши попередньо UCF файл.

Використання перемикачів та світлодіодів, зазначених у таблиці 3.5, зображено на рисунку 3.17.

Таблиця 3.5 – Перелік перемикачів і світлодіодів для плати Basys2

Перемикачі	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0	входи
FPGA Pin	N3	E2	F3	G3	B4	K3	L3	P11	
Світлодіоди	LD7	LD6	LD5	LD4	LD3	LD2	LD1	LD0	виходи
FPGA Pin					N5	N4	P4	G1	

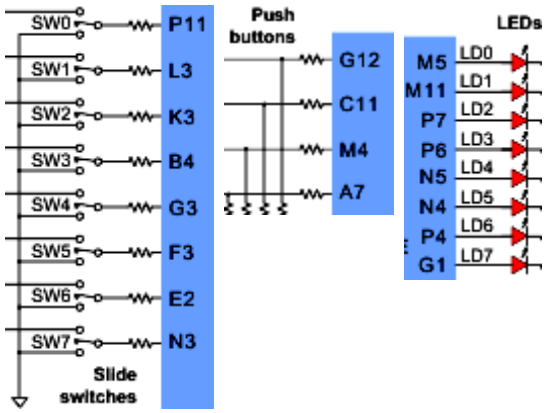


Рисунок 3.17 – Використовувані перемикачі та індикатори

3.4.8 Створення файлу обмежень

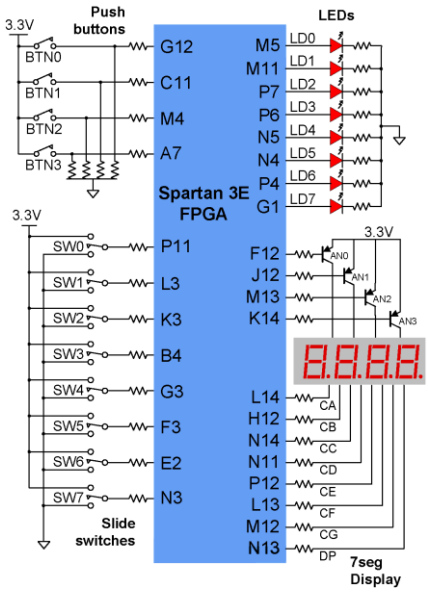
Зараз ви маєте достатньо інформації для створення того, що називається user constraint file (UCF) (файл призначених для користувача обмежень). Цей файл містить обмеження, які не були визначені в VHDL-описі, наприклад, розташування виводів, обмеження на характеристики проекту – це зручніше робити в UCF, ніж у VHDL-описі. Якщо ви допускаєте помилку в призначенні виводів, не потрібно повертатися і заново синтезувати проект.

Ви можете додати UCF у проект, використовуючи той самий процес, що використовується 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 діалогових вікон New Source.

Останнє діалогове вікно підсумовує установки вихідного файлу. Перегляньте його, якщо є потреба, поверніться і виправте

помилки, натиснувши «Back», інакше натисніть «Finish», щоб завершити процес. Потрібно зазначити, що цей файл не відкривається автоматично в жодному редакторі. Якщо ви вибрали файл обмежень у вікні Sources і розкрили User Constraints шляхом натискання на «+», ви побачите низку варіантів для редагування файлу користувача обмежень, включаючи і текстовий редактор. Відкрийте текстовий редактор шляхом натискання на Edit Constraints (Text). Вставте в поле редактора нижченаведені рядки, які здійснюють призначення сигналів виводів мікросхеми (рисунок 3.18).

У версії пакета 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.




```
NET "C"  LOC = "N3";
NET "X1"  LOC = "P11";
NET "X2"  LOC = " L3";
NET "X3"  LOC = " K3";
NET "X4"  LOC = "B4";
NET "Z1"  LOC = "G1";
NET "Z2"  LOC = "P4";
NET "Z3"  LOC = " N4";
NET "Z4"  LOC = " N5";
NET "Y0"  LOC = "M5";
```

Рисунок 3.18 – Використовувані перемикачі та індикатори

Підвікно PACE було пересунуто з його первинних 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 спрямування їхніх сигналів. У цьому вікні заповнення в LOC- областях грунтується на заздалегідь визначених установках виводів FPGA. Заповнення в інших областях здійснюється для порівняння. Ці опції встановлюють електричні характеристики I / O виводів.

Після введення установки 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 являє 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 в 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», а також «Do not show this dialog again ...».

Тепер у вашому проекті є файл установки відповідності (файл обмежень), можна розпочати імплементацію проекту.

Виберіть двоходовий 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 клавіші миші на Implement Design, вибравши Properties. Після закінчення імплементації ви побачите інформацію.

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 LUT’s.

Ви не побачите жодних помилок у вікні Console. Однак ви маєте завжди переглядати три log файлу, що формуються в процесі Translate, Map, 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 маєте переконатися в наявності зелених маркерів біля Implement Design process.

3.4.9 Програмування FPGA безпосередньо

Виконати безпосереднє програмування FPGA на платі Xilinx Basys2 і перевірити правильність роботи проекту (рисунок 3.18).

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 програмувальний файл для FPGA.

Виберіть свій елемент у вікні Sources. Потім подвійним натисканням виберіть Generate Programming File процес у вікні Processes. Project Navigator буде генерувати програмувальний файл і видавати інформацію про хід імплементації у вікні Console.

Якщо ви вперше скористалися платою, її потрібно встановити. Для встановлення драйверів плати потрібно запустити Adept програми Digilent. Перед тим як рухатися далі, візьміть плату Basys2 і USB кабель.

Після цього під'єднати USB кабель плати до USB порту вашого комп'ютера (рисунок 3.19).

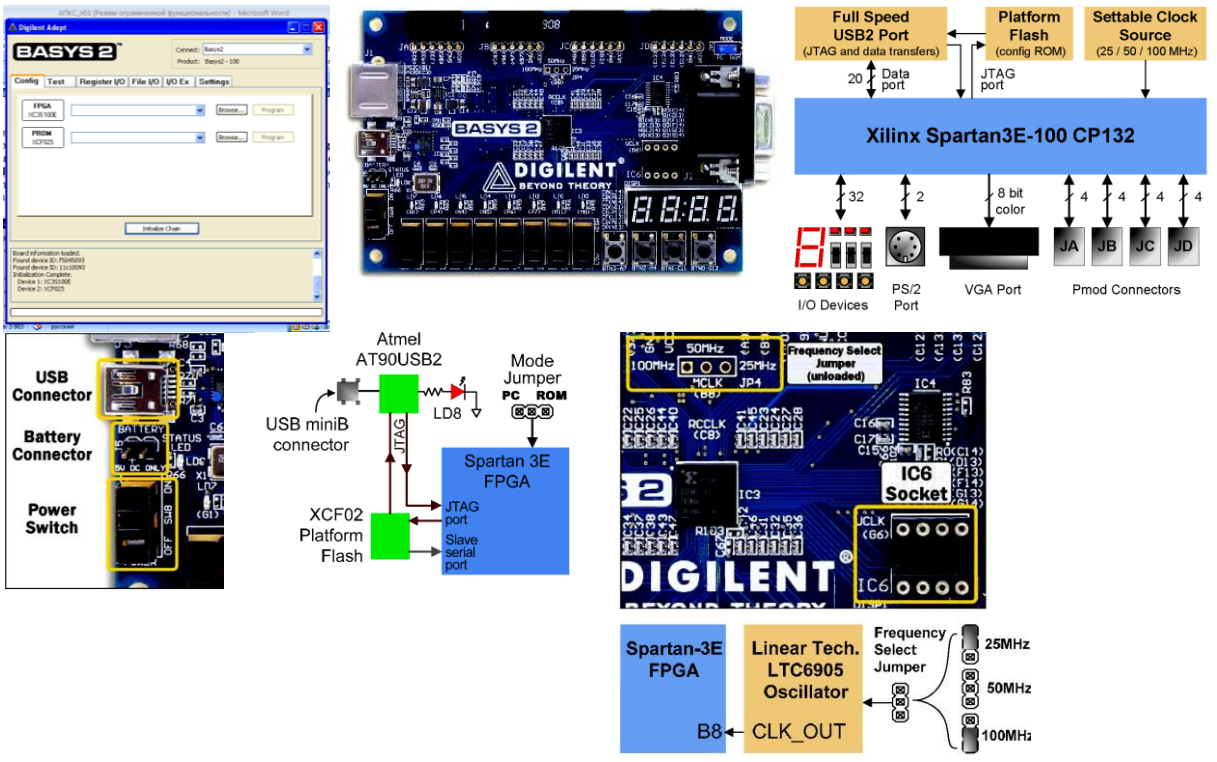


Рисунок 3.19 – Установлення драйверів плати

У меню Digilent Adept у вікні Connect з'явиться назва нашої плати Basys2. Після цього натиснути клавішу Inicialize. На платі блимне діод R66, це означає, що драйвери встановлено. Плата може бути запрограмована просто з цієї програми. Щоб запрограмувати FPGA, треба натиснути праворуч від неї кнопку Browse і вибрати свій проект. Натиснути кнопку Program праворуч від Browse і прошити FPGA.

Щоб запрограмувати пам'ять PROM, треба натиснути праворуч від неї кнопку Browse і вибрати свій проект. Натиснути кнопку Program праворуч від Browse і прошити PROM. Але в нашому проекті ми повертаємося в програму ISE і продовжуємо роботу з нею. Щоб завантажити ваш bitstream в FPGA, розкрийте Configure Target Device натисканням на «+». Потім подвійним натисканням завантажте Manage Configuration Project (iMPACT).

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, виконується в iMPACT. Після запуску iMPACT ви побачите серію діалогових вікон.

Плата має функцію програмування JTAG, яка також називається Boundary Scan. Виберіть цю опцію і натисніть «Finish». Далі ви має отримати послідовність трьох файлових замовників. У першому файловому замовнику виберіть файл transformation_four.bit, який був сформований у процесі імплементації. Він є програмувальним файлом для FPGA.

Після вибору програмувального файлу для FPGA ви можете отримати попередження, що Startup Clock був змінений. Ігноруйте це діалогове вікно. Метою наступних двох замовників файлів є ідентифікація програмувальних файлів для Xilinx пристроїв на платі. Ми поки ще нічого не програмували, тому оберіть Bypass (обхід) для них обох.

Нарешті, ви бачите вікно iMPACT, готове 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

установки 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.

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 SW7 в усі можливі положення і спостерігайте реакцію схеми на ці вхідні значення на індикаторі LD0 – LD7 (рисунок 3.20).

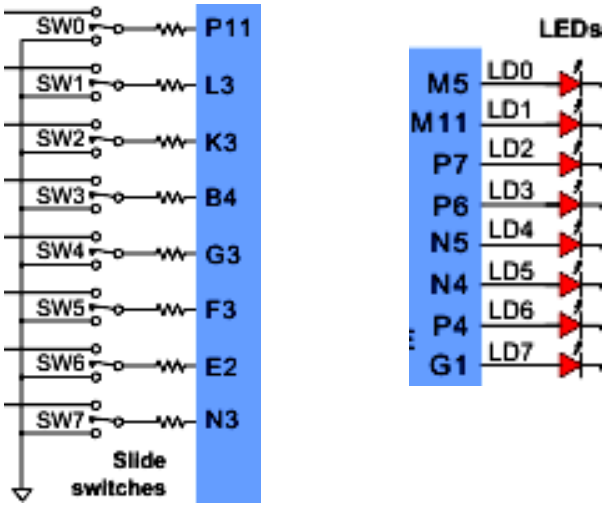


Рисунок 3.20 – Індикатори LD0 – LD7

Після цього можна запрограмувати плату через Flash пам'ять (програмування Xilinx Platform Flash). Повернімося до iMPACT, виберіть опцію PROM File Formatter у вікні Flows, двічі клацніть на ньому лівою кнопкою миші. Це ініціює серію діалогових вікон для збору додаткової інформації. У діалоговому вікні ви повинні вибрати xcf04s, 4-megabit Platform Flash і натиснути на «Add». Далі переходьте до наступного діалогового вікна.

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 діалогове вікно замовника файлів, виберіть two_input_xor.bit файл (як і при прямому програмуванні FPGA).

Після додавання two_input_xor.bit файлу iMPACT запропонує додати інший пристрій, натисніть на «No». Потім ви побачите інше діалогове вікно, яке покаже, що введення файлу було завершено. Ігноруйте діалогове вікно. Виберіть опцію Generate File в iMPACT Process Operstion і двічі натисніть на ній. iMPACT буде генерувати файл.

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 і натисніть на ньому правою кнопкою – таким чином ви можете призначити ваш заново створений файл конфігурації. Якщо пристрої не відображаються, правою кнопкою миші на порожньому полі виберіть у меню Initialize Chain, потім ви можете призначити ваш заново створений файл конфігурації.

Для подальшого використання виберіть lab1.mcs.

Потім виберіть іконку xcf04s, натисніть на ній правою кнопкою і виберіть опцію Program. З'явиться діалогове вікно з опціями програмування. Більшість цих опцій допоміжні для програмування Platform Flash і в даний момент не істотні (рисунок 3.20).

Перевірте опції, щоб вони збігалися, і натисніть «Ok», щоб почати програмування. З'явиться індикатор процесу 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 ньому інформації, накопиченої в пристрої Platform Flash (рисунок 3.21).

Ви можете зараз же протестувати свій проект. Вийдіть з iMPACT, вимкніть живлення і відключіть USB кабель від плати. Зачекайте 3 секунди і знову ввімкніть живлення. FPGA має завантажити ваш проект автоматично з пристрою Platform Flash.

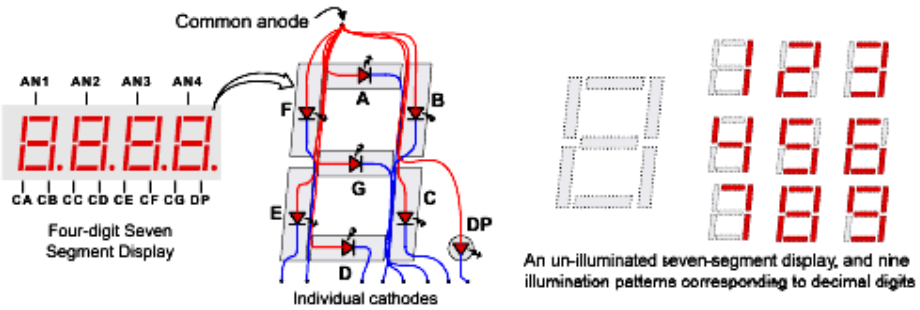


Рисунок 3.21 – Пристрий Platform Flash

3.5 Вимоги до оформлення курсового проекту

3.5.1 Оформлення пояснювальної записки

Загальними вимогами до пояснювальної записки є грамотність і логічна послідовність викладу матеріалу, стислість, чіткість і конкретність подання методу і алгоритму розв'язання задачі, а також опису отриманих результатів. У тексті допускаються тільки загальноприйняті скорочення.

Зміст пояснювальної записки розбивають на розділи, підрозділи, пункти, підпункти. Розділи мають бути з порядковими номерами, що позначаються арабськими цифрами. Заголовки розділів слід писати симетрично до тексту великими буквами. Підрозділи мають порядкові номери в межах кожного розділу. Номер підрозділу складається з номера розділу і порядкового номера підрозділу, відокремлених крапкою.

Заголовки підрозділів пишуть з абзацу малими літерами. Підрозділи можна розбити на пункти. Пункти нумерують у межах підрозділу. Номер пункту має складатися з номера розділу, підрозділу і пункту, між якими ставлять крапку.

Перенесення слів у заголовках не допускаються. Крапку в кінці заголовка не ставлять. Якщо заголовок складається з двох речень, їх розділяють крапкою. Відстань між заголовками і текстом має бути не менше, ніж три інтервали (два рядки).

Сторінки нумерують арабськими цифрами, додержуючись наскрізної нумерації впродовж всієї записки. На титульному аркуші номер не ставлять, на наступних сторінках номер проставляють у верхньому кутку, протилежному полю зшивання, без крапки, у полі над текстом.

Формули в пояснювальній записці нумерують арабськими цифрами в межах розділу. Номер формули складається з порядкового номера розділу і номера формули в розділі, між якими ставлять крапку. Номер вказується з правого боку аркуша на рівні формули в круглих дужках, наприклад: (1.2) – друга формула першого розділу. Якщо рівняння не вміщується в один рядок, його переносять після знака рівності (=) або після знака (+), мінус (-), множення (·), ділення (:).

Структурні схеми алгоритмів необхідно складати відповідно до ДСТУ-3008-2015 «Схеми алгоритмів, програм, даних і систем. Умовні позначення і правила оформлення».

Посилання в тексті на джерела слід наводити в порядку появи їх в тексті записки, вказуючи порядковий номер, виділений двома квадратними дужками. У переліку посилань наводяться прізвища та ініціали авторів, назва роботи, найменування і номер збірника чи журналу, назва видавництва, рік видання, номери сторінок, на які робиться посилання.

Додатки слід оформити як продовження записки на наступних її сторінках, розташовуючи їх у порядку появи посилань у тексті. Кожну програму слід починати з нового аркуша (сторінки) із зазначенням посередині сторінки слова «Додаток» із заголовком. Якщо в записці більше одного додатка, їх позначають послідовно з великої літери, наприклад: Додаток А, Додаток Б тощо.

3.5.2 Опис програмного забезпечення

Текст комп'ютерної програми моделювання та її стислий опис оформлюють в одному документі «Опис програми» і поміщають у додатку. Основний зміст цього документа визначається вимогами є ЄСКД. Документ "Опис програми" у роботі містить титульний аркуш і наступні розділи: текст програми; загальні відомості; функціональне призначення; опис логічної структури; використовувані технічні засоби; виклик і завантаження; вхідні дані; вихідні дані.

Залежно від особливостей програми допускається вводити додаткові розділи або об'єднувати окремі розділи.

Титульний аркуш документа містить відомості про тему курсового проекту, найменування програми (якщо воно відрізняється від теми курсового проекту), найменування документа, його кодове позначення і обсяг. Кодове позначення документа записується в строго визначеному форматі, наприклад, АБВГ 123.02.07 13, де АБВГ – код УкрДУЗТ; 123 – шифр спеціальності СКС; 02 – індекс кафедри СКС на факультеті ІКСТ; 07 – порядковий номер студента у списку групи; 13 – код виду документа «Опис програми». Форма заповнення титульного аркуша документа згідно ДСТУ-3008-2015 «Позначення програм і програмних документів» подано в додатку В.

Текст програми мовою оригіналу моделювання роздруковується на аркушах формату А4 або А3.

У розділі «Загальні відомості» мають бути вказані: позначення і найменування програми; програмне забезпечення (операційні системи, пакети програм моделювання і т. д.), необхідне для функціонування програми моделювання; мова програмування, якою написана програма.

У розділі «Функціональне призначення» вказуються класи розв'язуваних завдань і (або) призначення програми, а також відомості про функціональні обмеження на її застосування.

Розділ «Опис логічної структури» має містити опис або схему методу розв’язання і алгоритму програми; структури програми з описом функцій складових частин (підпрограм, процедур, компонент) і зв'язків між ними; зв'язків програми з іншими програмами. Опис логічної структури програми виконують з урахуванням тексту програми мовою оригіналу. Якщо метод розв’язання, схема моделювального алгоритму або блок-діаграма детально подані в розділах основної частини пояснювальної записки або в графічній частині, то обмежуються посиланням на них і відповідні описи не надають.

У розділі «Використовувані технічні засоби» вказуються типи комп'ютерів і пристроїв, які необхідні для проведення програмного машинного експерименту. У розділі «Виклик і завантаження» вказуються: способи виклику програми з відповідних носіїв даних і вхідні точки в програму; відомості про використовувану оперативну пам'ять і обсяг програми.

У розділі «Вхідні дані» зазначаються: характер, організація і попередня підготовка вхідних даних; формат, опис і спосіб кодування вхідних даних.

У розділі «Вихідні дані» вказуються: характер і організація вихідних даних; формат, опис і спосіб кодування вихідних даних. При цьому необхідно зробити посилання на додаток, в якому наводяться роздруківки результатів моделювання.

Наведені в документі описи мають бути досить лаконічними, їх загальний обсяг, як правило, не перевищує 2–3 сторінки.

3.5.3 Оформлення графічної частини

Графічна частина роботи може складатися з креслень, схем або демонстраційних плакатів. Характер і зміст графічної частини визначаються керівником і вказуються в завданні на курсовий проект. Креслення і схеми є складовою частиною курсового проекту, що містить структурні або функціональні схеми об'єктів моделювання, схеми моделювальних алгоритмів, схеми функціональних структур програм моделювання і т. д. Ілюстрації (рисунки) схем і креслень, подані у графічній частині, в пояснювальній записці не виконуються, в тексті записки на них мають бути в обов'язковому порядку посилання. Наприклад, «... дивись креслення АБВГ 123.02.07.5А», де 5А – код графічного документа. Основні графічні документи мають такі коди: 4А – схема функціональної структури; 4Б – схема організаційної структури; 4В – схема автоматизації; 4Г – схема структурного комплексу технічних засобів; 5А – креслення форми документа (відеограми).

На схемах і кресленнях наводять основні елементи, зв'язки між елементами, необхідні пояснювальні написи. Елементами схем є умовні графічні позначення об'єктів з їх кодами або найменуваннями. Виділення групи елементів схеми за будь-якою ознакою слід виконувати штрих-пунктирною лінією з пояснювальним підписом у лівому верхньому кутку рамки. Зв'язки між елементами відбивають відношення між об'єктами. Лінії зв'язку, як правило, виконують паралельними лініям зовнішньої рамки схеми.

Схеми слід виконувати компактно при збереженні зрозумілості і зручності читання. При виконанні креслень і схем слід керуватися вимогами ДСТУ «Загальні вимоги до виконання схем», ДСТУ «Позначення умовні графічні технічних засобів», ДСТУ «Вимоги до виконання креслень» Єдиної системи стандартів автоматизованої системи управління. Схеми алгоритмів і програм виконуються відповідно до ДСТУ «Схеми алгоритмів, програм, даних і систем. Умовні позначення і правила оформлення». Схеми і креслення виконуються олівцем або чорним чорнилом.

Плакати використовуються як ілюстративний матеріал під час захисту курсового проекту. На них можуть бути представлені схеми, що містяться в пояснювальній записці, інші схеми, графіки і матеріали, що доповнюють доповідь і відображають суть виконаної роботи. Кожен плакат має містити заголовок, що поміщається у верхній частині аркуша.

3.6 Організація виконання та захисту курсового проекту

Для здійснення загального керівництва і надання методичної допомоги при виконанні курсового проекту кожному студенту серед викладачів, аспірантів або наукових співробітників кафедри призначається керівник.

Керівник видає студенту індивідуальне завдання, в якому вказується тема роботи, наводиться текстовий опис об'єкта моделювання, визначаються вихідні дані, перелік розроблюваних питань, зміст розрахунково-пояснювальної записки та графічної частини, а також міститься список рекомендованої літератури. Термін видачі завдання не повинен перевищувати двох тижнів від початку семестру, термін здачі виконаної роботи на перевірку керівникові, як правило, – не пізніше двох тижнів до закінчення семестру. Виходячи із зазначених термінів, студент складає і погоджує з керівником календарний план виконання курсового проекту, в якому вказуються основні етапи роботи і терміни їх виконання. Загальний обсяг позааудиторної самостійної роботи студента над виконанням завдання становить близько 20 годин. До основних етапів виконання проекту належать такі:

- вибір теми, отримання та з'ясування завдання;

- аналіз існуючих методів розв’язання;
- розробка концептуальної моделі;
- вибір принципу побудови моделювального алгоритму і мови моделювання;
- розробка схеми моделювального алгоритму;
- планування машинного експерименту і підготовка вихідних даних;
- розробка програмного машинного забезпечення;
- перевірка програм моделювання на контрольних прикладах;
- проведення машинних експериментів;
- аналіз і інтерпретація результатів моделювання;
- оформлення пояснювальної записки та графічної частини роботи.

Завдання і календарний план виконання курсового проекту оформляються згідно з додатком Б.

Керівник проводить консультації зі студентами, визначає порядок і режим використання обчислювальної техніки, контролює хід виконання курсового проекту. Для систематизації цієї роботи розробляються графіки групових та індивідуальних консультацій, форми і графіки контролю ходу виконання роботи.

На початковому етапі студент повинен усвідомити поставлену задачу моделювання, ознайомитися з рекомендованою літературою, вибрати підхід до розв’язання завдання. Метою контролю з боку керівника на цьому етапі є оцінка підготовленості студента до розв’язання поставленого завдання, перевірка правильного розуміння постановки задачі, намічених шляхів моделювання, вміння проводити всебічний аналіз можливих підходів до розв’язання.

На проектному етапі студент повинен обґрунтовано вибрати шлях рохзв’язання задачі моделювання, використовуючи при цьому такі критерії оцінки ефективності, як точність, достовірність і економічність отримання результатів. Метою перевірки керівником є оцінка правильності вибору методу розв'язання задачі, принципу побудови моделювального алгоритму, мови моделювання, можливості реалізації моделі з використанням наявних комп'ютерних засобів.

На експериментальному етапі студент повинен перевірити адекватність моделі об'єкта, що вивчається, провести модельні

експерименти, виконати всебічний аналіз (коректність, точність, достовірність) і інтерпретацію отриманих результатів. Метою перевірки є контроль коректності і правильності отриманих результатів.

На завершальному етапі метою перевірки є контроль знань з оформлення пояснювальної записки та графічної частини, а також підготовленості студента до захисту курсового проекту.

Перед захистом закінчений курсовий проект подається керівнику для перевірки. Керівник перевіряє відповідність виконаної роботи завданням, ступінь повноти виконання, коректність основних рішень і отриманих результатів, а також якість оформлення роботи. Якщо робота містить розв’язки з усіх питань завдання, не має принципових помилок і оформлена за встановленими вимогами, то керівник допускає студента до захисту роботи. У противному випадку робота повертається студенту для доопрацювання або виправлення помилок.

Захист курсового проекту проводиться студентом – автором роботи перед комісією згідно із заздалегідь складеним графіком. Захист складається з доповіді студента по суті виконаної роботи і його відповіді на запитання членів комісії з теорії виконаної роботи, тексту доповіді, за матеріалами пояснювальної записки та графічної частини, отриманими результатами. Доповідь має містити опис об'єкта і постановку задачі його моделювання, вибір методу моделювання та принципу побудови моделювального алгоритму, опис алгоритму і аналіз отриманих результатів.

Оцінка роботи ведеться за стобальною системою на підставі теоретичних знань з дисципліни, якості та повноти виконання роботи, достовірності отриманих результатів, вміння студента відповісти і захистити результати роботи.

4 МЕТОДИЧНІ ВКАЗІВКИ ДО САМОСТІЙНОГО ВИВЧЕННЯ ДИСЦИПЛІНИ

У цьому розділі наведено рекомендації щодо вивчення як теоретичного матеріалу, що викладається на лекціях, так і матеріалу, винесеного на самостійне вивчення.

Порядок вивчення тем: 1, 2, 3, 4, 5.

Порядок виконання лабораторних робіт:

▪ після теми 2 лабораторна робота 1 (завдання 1 і 2);
▪ після теми 3 і 4 лабораторна робота 2 (завдання 3);
▪ після теми 6 лабораторна робота 3;
▪ після теми 9 лабораторна робота 4;
▪ після теми 12 лабораторна робота 5;
▪ після теми 14 лабораторна робота 6.

Конспект усього теоретичного матеріалу, слайд лекції, гіпертекстові підручники і екзаменаційні питання можна також знайти в [15] і на сервері кафедри СКС.

Для проведення лабораторних робіт і виконання курсового проекту необхідний комп'ютер IBM PC, платформа Windows XP, система моделювання цифрових схем Active-HDL, автоматизоване середовище проектування Xilinx ISE.

5 ІНДИВІДУАЛЬНІ РОЗРАХУНКОВІ ЗАВДАННЯ, КОНТРОЛЬНІ ЗАВДАННЯ

Вибрати варіант завдання з таблиці 5.1 за останніми двома цифрами номера залікової книжки. Якщо останні дві цифри номера залікової книжки становлять число 17 і більше, то для формування номера варіанта необхідно з цього числа відняти 16, тобто якщо Ваше число 26, то номер варіанта буде 10.

Необхідно виконати три завдання:

- Завдання I (Завдання 1 або Завдання 2 залежно від номера варіанта) (рисунок 5.1);
- Завдання II (Завдання 3 або Завдання 4 залежно від номера варіанта) (таблиця 5.2, 5.3);
- Завдання III (Завдання 5) (рисунок 5.2).

Розшифровку схемних позначень логічних елементів дивіться в додатку в кінці файлу. Будьте уважні, у Verilog ідентифікатор **a** й **A** – не одне й те саме.

Таблиця 5.1 – Варіанти завдань

Ва-рі-ант	Завдання I	Завдання II	Завдання III	Завдання IV	Завдання V	Завдання VI
1	рисунок 5.1, а	рисунок 5.1, а	таблиця 5.2, а	таблиця 5.3, а	рисунок 5.2, а	рисунок 5.2, к
2	рисунок 5.1, б	рисунок 5.1, б	таблиця 5.2, б	таблиця 5.3, б	рисунок 5.2, б	рисунок 5.2, л
3	рисунок 5.1, в	рисунок 5.1, в	таблиця 5.2, в	таблиця 5.3, в	рисунок 5.2, в	рисунок 5.2, м
4	рисунок 5.1, г	рисунок 5.1, г	таблиця 5.2, г	таблиця 5.3, г	рисунок 5.2, г	рисунок 5.2 , н
5	рисунок 5.1, д	рисунок 5.1, д	таблиця 5.2, д	таблиця 5.3, д	рисунок 5.2, д	рисунок 5.2, а
6	рисунок 5.1, е	рисунок 5.1, е	таблиця 5.2, е	таблиця 5.3, е	рисунок 5.2, е	рисунок 5.2, б
7	рисунок 5.1, ж	рисунок 5.1, ж	таблиця 5.2, ж	таблиця 5.3, ж	рисунок 5.2, ж	рисунок 5.2 , в
8	рисунок 5.1, и	рисунок 5.1, и	таблиця 5.2, и	таблиця 5.3, и	рисунок 5.2, и	рисунок 5.2, г

Таблиця 5.2 – Варіанти лічильників до завдання II

№	Тип лічильника	Тип початкового встановлення	Тип синхронізації	Розряд-ність
а)	Віднімаючий	Синхронне скидання в 0	задній фронт	4
б)	Віднімаючий	Синхронне установлення в 1	задній фронт	3
в)	Віднімаючий	Асинхронне скидання в 0	передній фронт	5
г)	Віднімаючий	Асинхронне установлення в 1	передній фронт	6
д)	Підсумовуючий	Синхронне установлення в 1	задній фронт	4
е)	Підсумовуючий	Синхронне установлення в 1 і 0	передній фронт	5
ж)	Підсумовуючий	Асинхронне установлення в 1	передній фронт	6
и)	Підсумовуючий	Асинхронне установлення в 1 і 0	передній фронт	5

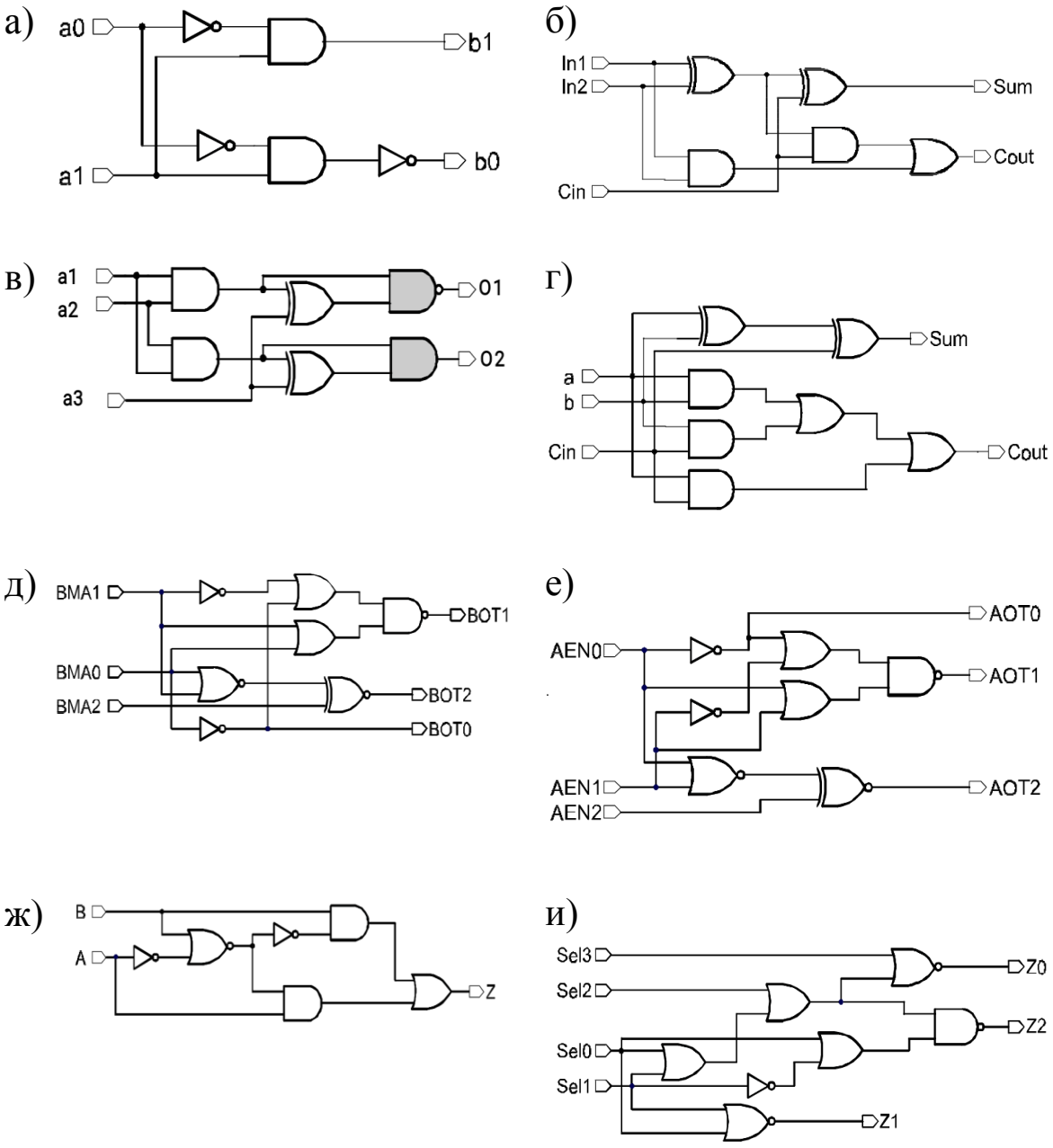


Рисунок 5.1 – Варіанти комбінаційних схем до завдання I

Таблиця 5.3 – Варіанти регістрів до завдання II

№	Установлення (Set / Reset)	Синхронізація CLK / фронт	Розрядність DataIn та DataOut	Входи управління зрушенням	Тип зсуву
а)	Асинхронне установлення в 1 і 0	передній	7	Shift	Вправо на 4 розряди із заповненням «0» старших розрядів
б)	Асинхронне установлення в 1 і 0	передній	6	Shift	Вправо на 3 розряди із заповненням «0» старших розрядів
в)	Синхронне установлення в 1 і 0	передній	4	Shift	Циклічний вправо на 1 розряд
г)	Синхронне установлення в 1 і 0	задній	3	Shift	Циклічний вліво на 1 розряд
д)	Асинхронне установлення в 1 і 0	задній	5	Shift	Вправо на 1 розряд
е)	Асинхронне установлення в 1	передній	7	ShiftR	Циклічний вправо на 1 розряд
				ShiftL	Циклічний вліво на 1 розряд
ж)	Синхронне установлення в 1 і 0	задній	6	Shift	Вліво на 1 розряд
и)	Синхронне установлення в 0	задній	5	ShiftR	Циклічний вправо на 1 розряд
				ShiftL	Циклічний вліво на 1 розряд

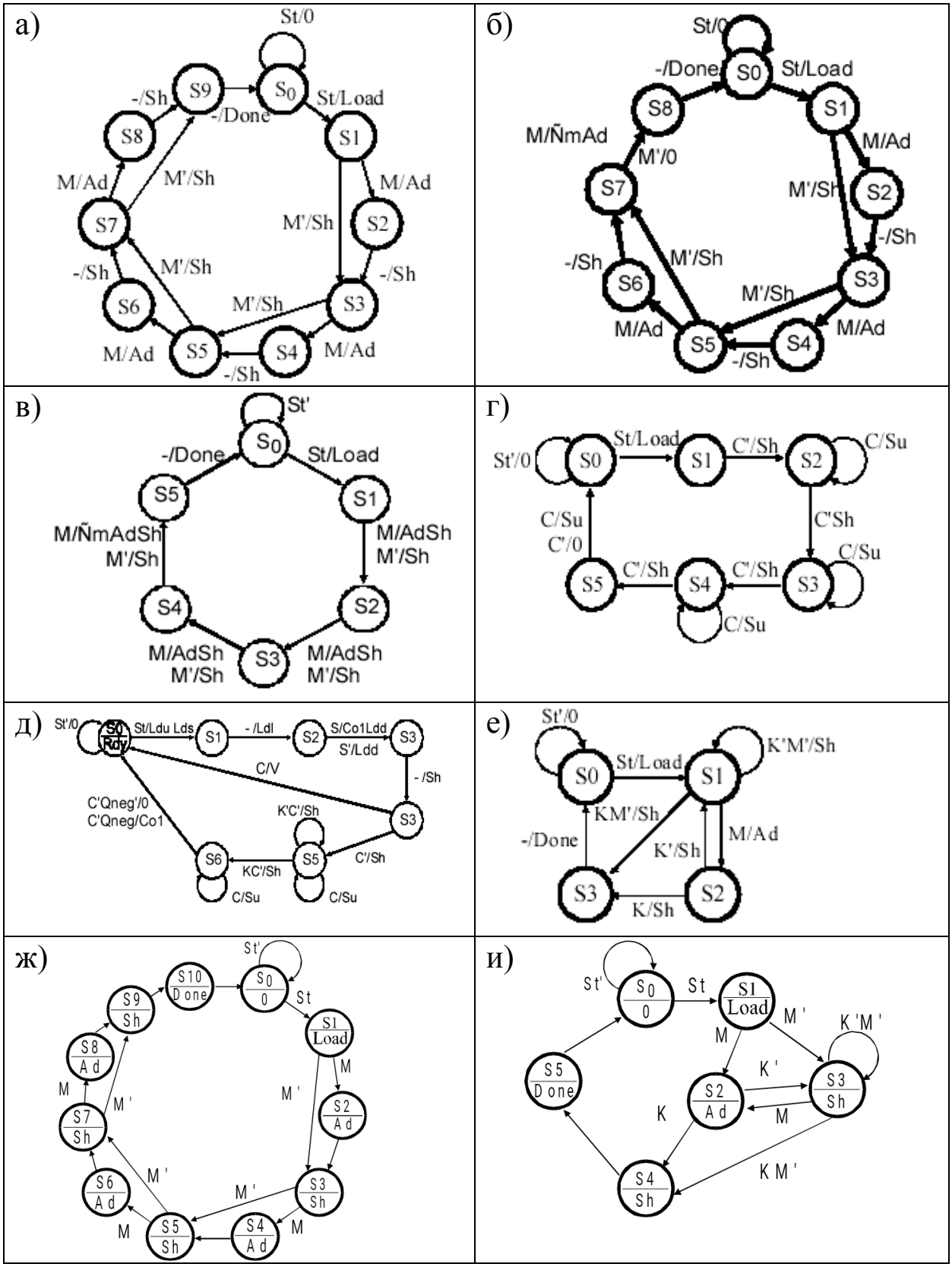


Рисунок 5.2 – Варіанти автоматів до завдання III, аркуш 1

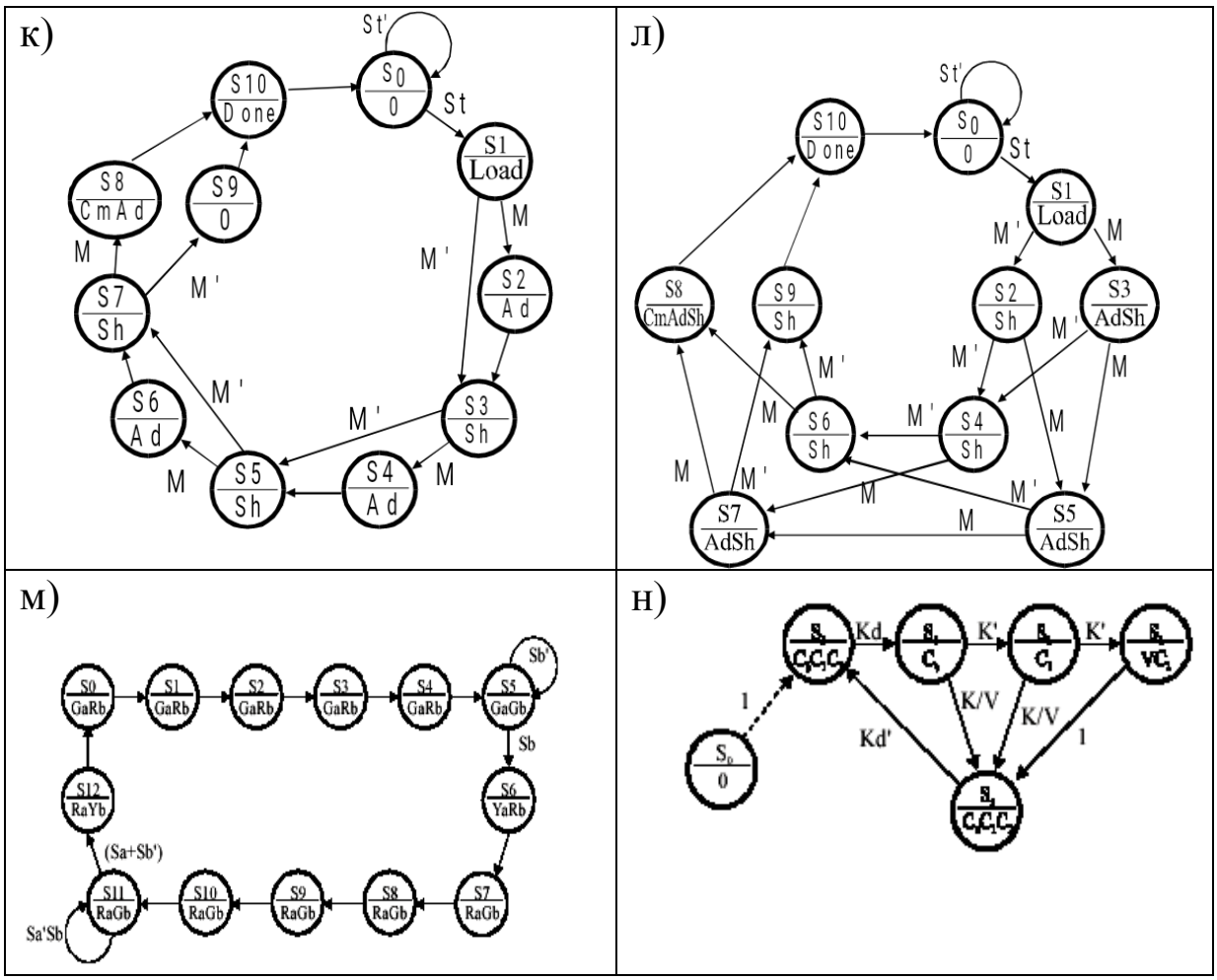


Рисунок 5.2, аркуш 2

6 ПРИКЛАДИ РОЗВ’ЯЗАННЯ ТИПОВИХ ЗАВДАННЯ

У цьому розділі наведено приклади розв'язання задач для виконання розрахункового завдання за темою 2 «Основи мови VHDL» для самостійного вивчення.

Завдання 6.1. Згідно з варіантом завдання записати структурну Verilog-модель пристрою (рисунок 6.1). В описі використовувати стандартні примітиви, як це показано на рисунку 6.1.

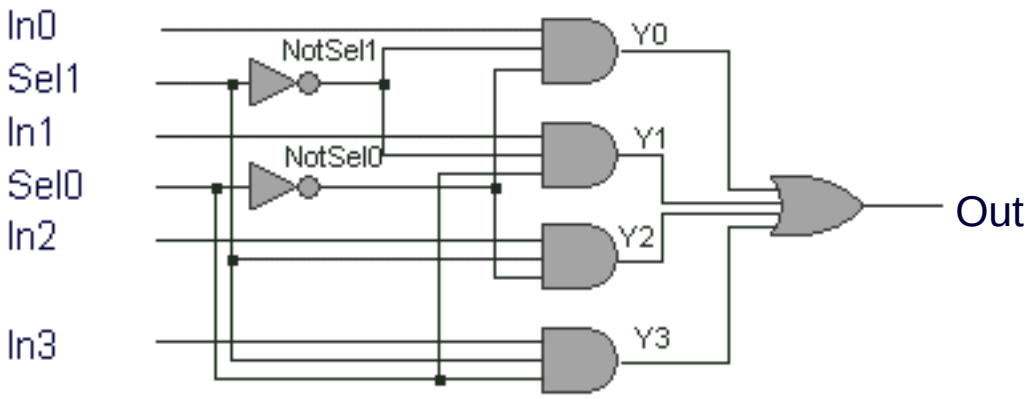


Рисунок 6.1 – Комбінаційна схема до прикладів 1 і 2

Лістинг 6.1 – Приклад виконання завдання 6.1

```
// Модель структурного опису з використанням стандартних примітивів.
// Опис елементів і зв'язків між ними.
module mux_4_to_1 (Out, In0, In1, In2, In3, Sel1, Sel0);
// Декларація портів
output Out;
input In0, In1, In2, In3, Sel1, Sel0;
// Декларація внутрішніх сигналів
wire NotSel0, NotSel1;
wire Y0, Y1, Y2, Y3;
and (Y1, In1, NotSel1, NotSel0);
not (NotSel0, Sel0);
and (Y3, In3, Sel1, Sel0);
or (Out, Y0, Y1, Y2, Y3);
and (Y0, In0, NotSel1, NotSel0);
not (NotSel1, Sel1);
and (Y2, In2, Sel1, NotSel0);
endmodule
```

Завдання 6.2. Згідно з варіантом завдання записати VHDL-модель пристрою рівня передачі даних, приклад якої наведено нижче.

Лістинг 6.2 – Приклад виконання завдання 6.2

```
// Модель рівня передачі даних (data flow)
// Логічні рівняння
```

```
module mux_4_to_1 (Out, In0, In1, In2, In3,
Sel1, Sel0);
// Декларація портів
output Out;
input In0, In1, In2, In3, Sel1, Sel0;
assign Out = (~ Sel1 & ~ Sel0 & In0) | (~ Sel1
& Sel0 & In1) | (Se1 & ~ sel0 & In2) | (Sel1 &
Sel0 & In3);
endmodule
```

Завдання 6.3. Згідно з варіантом завдання записати поведінкову Verilog-модель лічильника, приклад якого наведено нижче.

Лістинг 6.3 – Приклад виконання завдання 6.3

```
// Модель синхронного чотирирозрядного
підсумовуючого лічильника з
// асинхронним установленням в 0. Синхронізація
по передньому фронту.
module counter (COUNTER, CLK, RESET);
parameter SIZE = 'd4; // оголошення константи –
розрядність
// Декларація портів
output [SIZE-1: 0] COUNTER;
input CLK, RESET;
reg [SIZE-1: 0] COUNTER;
// Передній фронт синхронізації, асинхронний
reset
always @ (posedge CLK or posedge RESET)
begin
if (RESET) COUNTER = 'd0;
else COUNTER = COUNTER + 'd1;
end
endmodule
```

Лістинг 6.4 – Ще один приклад виконання завдання 6.3

```
// Модель синхронного п’ятирозрядного
віднімаючого лічильника з
```

```
// синхронним установленням в 0. Синхронізація
по задньому фронту.
module counter_ (Q, clk, reset);
input clk, reset;
wire clk, reset;
output [4: 0] Q;
reg [4: 0] Q;
always @ (negedge clk) // Задній фронт
синхронізації
if (reset == 'b1) Q =' d0; // синхронний reset
else Q = Q - 1;
endmodule
```

Завдання 6.4. Згідно з варіантом завдання записати поведінкову Verilog-модель регістра, що виконує паралельне завантаження даних і зрушення.

Лістинг 6.5 – Приклад виконання завдання 6.4

```
/ * Модель восьмирозрядного регістра з
синхронним установленням в 0, із зсувом вправо
на 4 розряди і зрушенням вліво на 3 розряди із
заповненням нулями. Також регістр виконує
циклічний зсув вліво. Синхронізація по задньому
фронту. Використаний оператор casex * /
module register (DataOut, DataIn, ShiftR,
Shift, ShiftL, CLK, RESET);
input ShiftR, ShiftL, Shift, CLK, RESET;
input [7: 0] DataIn;
output [7: 0] DataOut;
reg [7: 0] DataOut;
always @ (negedge CLK)
casex ({RESET, ShiftR, ShiftL, Shift})
4'b1xxx: DataOut = 8'b00000000;
4'bx1xx: DataOut = DataOut >> 4;
4'bxx1x: DataOut = DataOut << 3;
4'bxxx1: DataOut = {DataOut [6: 0], DataOut [7]};
default DataOut = DataIn;
endcase
endmodule
```

Лістинг 6.6 – Ще один приклад виконання завдання 6.4

```
/ * Модель восьмирозрядного регістра з синхронним встановленням в 0, із зсувом вправо на 4 розряди і зрушенням вліво на 3 розряди із заповненням нулями. Також регістр виконує циклічний зсув вліво. Синхронізація по передньому фронту. Використаний оператор if * /
module register (DataOut, DataIn, ShiftR, Shift, ShiftL, CLK, RESET);
input ShiftR, ShiftL, Shift, CLK, RESET;
input [7: 0] DataIn;
output [7: 0] DataOut;
reg [7: 0] DataOut;
always @ (posedge CLK)
if (RESET) DataOut = 8'b00000000;
    else if (ShiftR) DataOut = DataOut >> 4;
    else if (ShiftL) DataOut = DataOut << 3;
    else if (Shift) DataOut = {DataOut [6: 0], DataOut [7]};
else DataOut = DataIn;
endmodule
```

Завдання 6.5. Згідно з варіантом завдання записати поведінкову VHDL-модель синтезованого керуючого автомата (з двома блоками always). Пояснення:

У записі X / Y X – умова переходу (вхідні сигнали), Y – виконувана дія (вихідні сигнали). Вершини відповідають станам автомата, дуги позначають переходи між станами. Для автоматів типу Мілі X / Y записані біля дуги графа, що позначає перехід між станами. Для автоматів типу Мура X записано біля дуги графа, а Y – у вершині графа під ім'ям стану. Якщо в однієї дуги записані дві умови переходів, це означає, що в дійсності там дві дуги. Якщо умову переходу не вказано, це означає, що перехід безумовний і буде виконуватися під час вступу синхросигналу.

Для пристрою Мілі з входами M, K і виходами Ad, Sh запис M / Ad означає, що перехід буде виконуватися, коли M = 1 (значення K не відіграє ролі), при цьому Ad = 1, а Sh = 0. M 'означає M = 0. Для пристрою Мура (рисунок 6.2) з входами C,

St і виходами Su, Sh запис S3 / Su означає, що коли автомат переходить у стан S3, виробляється сигнал Su = 1, при цьому Sh = 0, C 'означає C = 0, а C означає C = 1. Якщо дузі (переходу) відповідає C, то C = 1 (значення St не відіграє ролі). Всі пристрої повинні мати сигнал скидання в початковий стан RESET. Двоблокова модель автомата містить один блок always, що обчислює наступний стан і комбінаційні сигнали автомата Мілі, а другий блок – послідовні. Така модель є синтезованою.

Приклад виконання завдання 6.5: Автомат Мура (рисунок 6.2). Автомат Мілі з входом enable, визначальним умови переходів станів (рисунок 6.3).

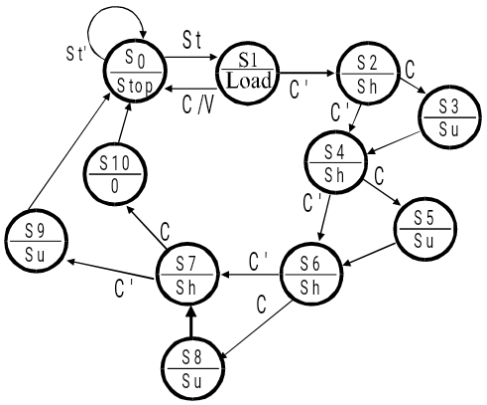


Рисунок 6.2 – Автомат Мура

Лістинг 6.7 – Приклад виконання завдання 6.5

```
module FSM (Clk, Reset, St, C, Stop, Load, Sh,
Su, V);
input Clk, Reset, St, C;
output Stop, Load, Sh, Su, V;
reg   Stop, Load, Sh, Su, V;
// Мітки станів
parameter S0 = 4'b0000;
parameter S1 = 4'b0001;
parameter S2 = 4'b0010;
parameter S3 = 4'b0011;
parameter S4 = 4'b0100;
parameter S5 = 4'b0101;
parameter S6 = 4'b0110;
parameter S7 = 4'b0111;
```

```
parameter S8 = 4'b1000;
parameter S9 = 4'b1001;
parameter S10 = 4'b1010;
reg [3:0] state, next_state; // блок для
послідовної логіки
always @ (posedge Clk or posedge Reset)
if (Reset) state <= S0;
else state <= next_state;
// блок для комбінаційної логіки
always @ (state or St or C)
begin
Stop <= 1'b0; Load <= 1'b0; Sh <= 1'b0; Su <=
1'b0; V <= 1'b0;
case (state)
S0: if (St) next_state <= S1;
else next_state <= S0;
S1: if (C) begin next_state <= S0;
V <= 1'b1; end
else next_state <= S2;
S2: if (C) next_state <= S3;
else next_state <= S4;
S3: next_state <= S4; S4: if (C) next_state <=
S5;
else next_state <= S6;
S5: next_state <= S6;
 S6: if (C) next_state <= S8;
else next_state <= S7;
 S7: if (C) next_state <= S10;
else next_state <= S9;
S8: next_state <= S7;  S9: next_state <= S0;
S10: next_state <= S0;
default: next_state <= S0;
endcase
Stop <= (state == S0)? 1'b1: 1'b0;
Load <= (state == S1)? 1'b1: 1'b0;
Sh <= (state == S2) || (state == S4) || (state
== S6) || (state == S7)? 1'b1: 1'b0;
```

```
Su <= (state == S3) || (state == S5) || (state
== S8) || (state == S9)? 1'b1: 1'b0;
End. endmodule
```

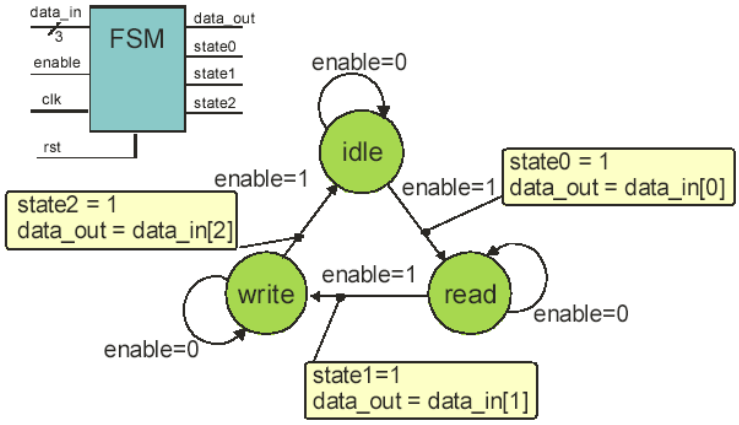


Рисунок 6.3 – автомат Мілі

Лістинг 6.8 – Ще один приклад виконання завдання 6.5

```
module FSM1_synplify (clk, rst, enable,
data_in, data_out, state0, state1, state2);
input clk, rst, enable; input [2: 0] data_in;
output data_out, state0, state1, state2;
/ * Визначено мітки станів * /
parameter deflt = 3'bxxx;
parameter idle = 3'b001;
parameter read = 3'b010;
parameter write = 3'b100;
reg data_out, state0, state1, state2;
reg [2: 0] state, next_state;
/ * Блок Always для послідовної логіки * /
always @ (posedge clk or negedge rst)
if (! rst) state <= idle;
else state <= next_state;
/ * Блок Always для комбінаційної логіки * /
always @ (state or enable or data_in) begin
/ * Значення за замовчуванням для виходів
автомата FSM * /
state0 <= 1'b0; state1 <= 1'b0; state2 <= 1'b0;
data_out <= 1'b0;
```

```
case (state)
idle: if (enable)
begin state0 <= 1'b1;
data_out <= data_in [0];
next_state <= read;
end
else next_state <= idle;
read: if (enable)
begin state1 <= 1'b1;
data_out <= data_in [1];
next_state <= write;
end
else next_state <= read;
write: if (enable)
begin state2 <= 1'b1;
data_out <= data_in [2];
next_state <= idle;
end
else next_state <= write;
/ * Default привласнення для моделювання * /
default: next_state <= deflt;
endcase
end endmodule
```

СПИСОК ЛІТЕРАТУРИ

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

2 IEEE Std 1364-2001. IEEE Standard Hardware Description Language Based on the Verilog Hardware Description Language. The Institute of Electronicx Engeneers, Inc. 345 East 47th Street, New York, NY 10017-2394, USA, 2000. 403 p.

3 Семенец В. В., Хаханов В. И., Хаханова, И. В. Проектирование цифрових систем с использованием языка VHDL. Харьков : ХНУРЕ, 2003. 510 с.

4 Лістровий С. В., Семенец В. В. Інформаційно-управляючі системи та організація паралельних обчислювань : навч. посібник. Харків : Діса плюс, 2015. 324 с.

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

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

7 Боровик В. М., Гамаюн В. П. Автоматизоване робоче місце проектування інформаційних систем і баз даних : навч. посібник. Київ : Вид-во Нац. авіац. ун-ту «НАУ-друк», 2010. 128 с.

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

9 Мірошник М. А. Арифметичні і логічні основи цифрових автоматів. Гіпертекстові навчальні матеріали (ел. навч. матеріали). 2010.

10 Мірошник М. А. Діагностика і моделювання. Гіпертекстові навчальні матеріали (ел. навч. матеріали). 2010.

11 Хаханов, В. І. Хаханова, І. В. [та ін.] Verilog & SystemVerilog. Харків : Нове слово, 2010. 528 с.

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

13 Мірошник М. А. Методичні вказівки до лабораторних робіт з дисципліни «Технології та автоматизація проектування

пристроїв складних комп'ютерних систем» для студентів спеціальності 123 «СКС». Харків : УкрДУЗТ, 2019. 48 с.

14 Мірошник М. А. Методичні вказівки до курсового проекту з дисципліни «Технології та автоматизація проектування пристроїв складних комп'ютерних систем» для студентів спеціальності 123 «СКС». Харків : УкрДУЗТ, 2019. 48 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.

16 Книшев Д. А., Кузелин М. О. ПЛИС фирмы «Xilinx» : описание структуры основных симейств. Москва : Додека, 2001.

17 Прищепа С. Л «САПР цифровых устройств» «Проектирование цифровых схем при помощи САПР WeBPACK ISE : учеб.-метод. пособие по курсу «САПР цифровых устройств» для студ. спец. «Защита информации в телекомуникациях» і «Телекомуникационные системы» дневной формы обуч. Минск : БДУИР, 2006. 56 с.

ДОДАТОК А
Форма титульного аркуша пояснювальної записки

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Український державний університет залізничного транспорту

ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КЕРУЮЧИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Кафедра "Спеціалізовані комп'ютерні системи"

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО КУРСОВОГО ПРОЕКТУ

з дисципліни:
“Технології та автоматизація проектування комп’ютерних систем ”

_____АБВГ.ХХХ.ХХХ ПЗ_____
(позначення документа)

на тему: " **ПРОЕКТУВАННЯ ШИФРАТОРА ДВІЙКОВО - ДЕСЯТКОВИХ ЧИСЕЛ ЗА ДОПОМОГОЮ САПР XILINX ISE**"
(тема проекту)

ВИКОНАВ:
студент _____СКС – __IV ___
(шифр групи)
_____
(прізвище, ініціали)

ПРИЙНЯВ:
проф. Мірошник М. А.
(посада, прізвище, ініціали)
_____
(підпис)

Роботу захищено з оцінкою "_____"
" _____ " _____ 2018 р.
Комісія: _____ (_____)
_____ (_____)
_____ (_____)
(підписи) (ПІБ)

Харків 2019

ДОДАТОК Б

Зразок заповнення аркуша завдання на курсовий проект

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Український державний університет залізничного транспорту

Факультет ІКСТ Кафедра СКС

Спеціальність Спеціалізовані комп'ютерні системи

(номер, назва)

Курс IV Група СКС – __ IV семестр VIII

ЗАВДАННЯ

НА КУРСОВИЙ ПРОЕКТ (РОБОТУ)

студентові _____

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Проектування шифратора двійково- десяткових чисел за допомогою САПР XILINX ISE

2. Термін здачі студентом закінченого проекту 29.04.2019 р.

3. Вихідні дані до проекту (роботи) реверсивний двійково- десятковий лічильник

код, у якому лічильник веде лічення

послідовність лічення

Початковий стан « »

Імпульс переносу формується під час переходів « » – « », « » – « »,

Стан лічильника кодується шифратором коду «2» з «5».

Використовувати тригер типу D та логічні елементи типу «І-НІ».

4. Зміст пояснювальної записки (перелік питань, що їх потрібно розробити)

4.1 аналіз технічного завдання

4.2 проектування та аналіз функціональної схеми лічильника

4.3 побудова граф-схеми автоматної моделі лічильника

4.4 розробка VHDL-опису лічильника

4.5 моделювання роботи лічильника в САПР Xilinx ISE

4.6 програмування плати Diligent Basys2

5. Перелік графічного матеріалу

5.1 лістинг програми опису роботи лічильника мовою VHDL

5.2 лістинг файлу обмежень

5.3 лістинг файлу, що програмує ПЛІС

КАЛЕНДАРНИЙ ПЛАН

Номер	Назва етапів курсового проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Аналіз технічного завдання	28.02 – 4.03	
2	Проектування схеми лічильника	7.03 – 18.03	
3	Розробка VHDL-опису лічильника	21.03 – 1.04	
4	Моделювання роботи лічильника в САПР Xilinx ISE	4.04 – 15.04	
5	Програмування плати Diligent Basys2	18.04 – 28.04	
6	Здача проекту на перевірку викладачу	29.04	
7	Захист курсового проекту	2.05 – 6.05	

ДОДАТОК В
Зразок оформлення реферату курсового проекту (роботи)

РЕФЕРАТ

Пояснювальна записка до курсового проекту містіть: ___ стор., ___ таблиць, ____ рисунків, ___ джерел, ___ додатків.
Ключові слова: КОМБІНАЦІЙНА СХЕМА, лічильник, шифратор, КОД, ПЛАТА, VHDL.
Текст реферату має відбивати інформацію, подану в пояснювальній записці, в такій послідовності:
Об’єктом розробки у даному курсовому проекті є реверсивний двійково-десятковий лічильник та шифратор.
Метою курсового проекту є розробка шифратора лічильника двійково-десяткових чисел, його роботи мовою високого рівня VHDL, моделювання його роботи в автоматизованому середовищі Web PACK ISE фірми Xilinx та програмування плати Basys2 типу Spartan 3E фірми Xilinx.
У цьому курсовому проекті використовувались такі математичні методи: мінімізація функції за допомогою карт Карно, моделювання роботи системи тощо.
У курсовому проекті використовувались такі технічні та програмні засоби: компілятор мови високого рівня ActivHDL, автоматизоване середовище Web PACK ISE фірми Xilinx, налагоджувальний комплекс Spartan 3E на прикладі налагоджувальної плати Basys2.
У результаті виконання даного курсового проекту було отримано такі результати: показано їх новизну; основні характеристики об'єкта розробки або дослідження.
У курсовому проекті запропоновано алгоритм лічення та кодування виходів лічильника, описано їх роботу мовою високого рівня VHDL, що реалізує цей алгоритм мовою VHDL.
Також треба вказати:

ступінь реальності курсового проекту;
комплексність проекту;
рекомендації щодо використання результатів роботи;
можливу галузь застосування;
економічну ефективність;
висновки.

ДОДАТОК Г
Зразок оформлення змісту курсового проекту

ЗМІСТ

Перелік позначень і скорочень……..
Вступ……
1 Аналіз технічного завдання……
2 Огляд сучасних САПР та різновид налагоджувальних плат та фірм-виробників……..
3 Проектування реверсивного двійково-десяткового лічильника та шифратора двійково-десяткових чисел ……
4 Використання мови високого рівня VHDL для опису моделі лічильника та шифратора ……
5 Вивчення особливостей інструментального набору Spartan 3E для проектування ЦС на ПЛІС на прикладі налагоджувальної плати Basys2 ……..
6 Автоматизація проектування комп’ютерних систем в автоматизованому середовищі WebPACK ISE фірми Xilinx ….
7 Моделювання роботи лічильника та шифратора в САПР ISE. Формування файлу обмежень. ……
8 Програмування плати Diligent Basys2 ……
Перелік використаної літератури……
Додаток А Лістинг програми лічильника мовоюVHDL ……..
Додаток Б Лістинг файлу обмежень ……..
Додаток С Лістинг файлу, що програмує ПЛІС ……