

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ
до виконання курсового проекту
з дисципліни
"Комп'ютерне моделювання і проектування
пристроїв цифрової електроніки"
для студентів спеціальності 8.090801
"Мікроелектроніка і напівпровідникові прилади"
денної і заочної форм навчання

Методичні вказівки до виконання курсового проекту з дисципліни "Комп'ютерне моделювання і проектування пристроїв цифрової електроніки" для студентів спеціальності 8.090801 "Мікроелектроніка і напівпровідникові прилади" денної і заочної форм навчання /Укл. Н.М. Нагорна. – Запоріжжя: ЗНТУ, 2006. – 38 с.

Рекомендовано до видання НМО спеціальності як методичні вказівки до виконання курсового проекту з дисципліни "Комп'ютерне моделювання і проектування пристроїв цифрової електроніки" усіх форм навчання.

Укладач: Н.М. Нагорна, ст. виклад.

Рецензент: В.О. Томашевський, доц., канд.техн. наук

Відповід. за випуск: Г.В.Сніжной, доц., канд.фіз.-матем. наук

Затверджено
на засіданні кафедри
"Мікроелектроніка і
напівпровідникові прилади"
Протокол №7 від 29.06.2006

ЗМІСТ

1 Мета і задачі курсового проекту	4
2 Завдання на курсовий проект	5
3 Організаційні вказівки	5
4 Методичні вказівки до виконання розділів курсового проекту	5
4.1 Основні розділи курсового проекту	5
4.2 Рекомендації до виконання етапів курсового проекту	6
4.2.1 Відомості про використовувану елементну базу	6
4.2.2 Маршрут проектування цифрового пристрою	7
4.2.3 Методика синтезу послідовнісних схем на основі моделей керуючих мікропрограмих автоматів з жорсткою логікою	10
4.2.3.1 Мікропрограмих автомати з жорсткою логікою	10
4.2.3.2 Розробка граф-схеми алгоритму (ГСА)	11
4.2.3.3 Побудова графа автомата	12
4.2.3.4 Структурний синтез цифрового автомата	13
4.2.3.5 Приклад синтезу цифрового автомата	14
4.3 Опис послідовнісних пристроїв автоматними моделями у VHDL – кодах	16
4.4 Методика проектування кільцевих лічильників	18
5 Вказівки щодо оформлення і захисту курсового проекту	21
6 Рекомендована література	22
Додаток А. Варіанти завдань до курсового проекту	24
Додаток Б. Приблизний календарний план виконання курсового проекту	28
Додаток В. Процедура розробки проекту в САПР MAX+PLUS II	29
Додаток Г. Ілюстрація використання методики проектування схем послідовнісного типу	32
Додаток Д. Роздруковка VHDL-програми "Автомат Мілі з п'ятьма станами"	38

1 МЕТА І ЗАДАЧІ КУРСОВОГО ПРОЕКТУ

Метою курсового проекту (КП) є поглиблення і розширення теоретичних знань, вироблення у студентів системного підходу до вирішення задач проектування схемотехніки мікроелектронних пристроїв, отримання навичок по вибору і науковому обґрунтуванню прийнятих проектних рішень, по застосуванню систем автоматизованого проектування в процесі розробки апаратури, по користуванню спеціальною науковою і довідковою літературою, діючими стандартами.

Важлива роль даного КП полягає в пристосуванні знань, набутих студентами при попередньому вивченні дисциплін "Інформатика", "Твердотільна електроніка", "Моделювання в електроніці", "Цифрові автомати", "Цифрова схемотехніка".

При роботі над курсовим проектом студент набуває досвіду для подальшої роботи над дипломним проектом і в майбутній інженерній діяльності.

В результаті виконання даного курсового проекту студент повинен засвоїти:

- прийоми самостійного аналізу літературних джерел за заданою темою;
- підходи до вибору перспективної елементної бази, придатної для реалізації заданих характеристик проектуємого пристрою;
- методику процесу проектування цифрових пристроїв на основі послідовнісних схем;
- методику реалізації проекту засобами САПР;
- прийоми функціонального моделювання цифрових пристроїв.

Після виконання курсового проекту студент повинен уміти:

- синтезувати структурну схему цифрового пристрою за заданим алгоритмом його роботи;
- виконувати проектування пристроїв з використанням сучасної САПР;
- проводити комп'ютерний аналіз роботи цифрового пристрою;
- критично оцінювати результати аналізу.

2 ЗАВДАННЯ НА КУРСОВИЙ ПРОЕКТ

Тема КП задається студенту керівником курсового проекту. Студенту надається право вибору теми КП. Студент може також запропонувати свою тему з обґрунтуванням доцільності її розробки.

Завдання на КП видається студенту на початку семестру за типовою формою. Приблизний перелік тем КП наведений у додатку А.

3 ОРГАНІЗАЦІЙНІ ВКАЗІВКИ

Робота над курсовим проектом повинна проводитись згідно з календарним планом. Приблизний календарний план виконання курсового проекту приводиться у додатку Б.

Фактичний календарний план розробляється керівником проекту сумісно зі студентом після видачі завдання на проект.

Керівник проекту визначає зміст і об'єм розрахункової і графічної частин проекту, надає допомогу студенту у виборі літературних джерел, проводить консультації, виявляє помилки, допущені в ході проектування, систематично здійснює контроль виконання календарного плану. Після закінчення курсового проекту керівник перевіряє пояснювальну записку і графічну частину.

У процесі виконання проекту студент повинен доповідати керівникові про поетапне виконання ним робіт.

4 МЕТОДИЧНІ ВКАЗІВКИ ДО ВИКОНАННЯ РОЗДІЛІВ КУРСОВОГО ПРОЕКТУ

4.1 Основні розділи курсового проекту

Курсовий проект може включати в себе нижченазвані розділи.

- 1 Основні теоретичні відомості.
- 2 Обґрунтування вибору елементної бази для реалізації пристрою.
- 3 Опис принципів роботи проектного пристрою.
- 4 Синтез цифрового пристрою заданого типу.
- 5 Реалізація цифрового пристрою на мікросхемі програмованої логіки.
- 6 Верифікація проекту схеми.
- 7 Аналіз результатів проектування.

4.2 Рекомендації до виконання етапів курсового проекту

4.2.1 Відомості про використовувану елементну базу

В проекті розробляється цифровий або цифро-аналоговий пристрій на базі програмованих логічних інтегральних схем (ПЛІС).

При цьому передбачається використання ПЛІС типів *CPLD* і *FPGA*.

CPLD (Complex Programmable Logic Device) є програмованими комутованими матричними блоками. Це ПЛІС, що містять декілька матричних логічних блоків, об'єднаних комутаційною матрицею. Кожен блок – це програмована матриця "I", фіксована матриця "АБО" і макрокомірки. До цього класу, наприклад, відносяться ПЛІС сімейства MAX3000, MAX7000 і MAX9000 фірми Altera, схеми XC7000 і XC9500 фірми Xilinx тощо.

FPGA (Field Programmable Gate Array) - програмовані користувачем вентиляльні матриці. Це ПЛІС з матрицею комірок, які мають від двох до п'яти входів і складаються з логічних елементів, тригерів, відрізків ліній зв'язку, що з'єднуються перемичками з польових транзисторів.

ПЛІС програмуються зміною рівня електричного поля (field) в затворах польових транзисторів. Затвори всіх "програмованих" польових транзисторів підключені до виходів тригерів одного довгого регістра зсуву, в який записується інформація при програмуванні ПЛІС. Деякі з ділянок цього регістра можуть також виконувати роль комірок постійних ЗП.

Отримання конкретного проекту на базі *FPGA*, як і на основі інших ПЛІС, реалізується дією на програмовані міжз'єднання, внаслідок чого забезпечується замкнений стан одних ділянок і розімкнений – інших.

Архітектура *FPGA* розробляється фірмами Xilinx, Actel, Altera, Atmel тощо. До класу *FPGA*, наприклад, відносяться ПЛІС сімейства FLEX10K, FLEX8000, FLEX6000 фірми Altera, XC2000, XC3000, XC4000, Spartan, Virtex фірми Xilinx.

4.2.2 Маршрут проектування цифрового пристрою

При виконанні проекту необхідно дотримуватись маршруту проектування цифрового пристрою, який включає в себе нижченаведені етапи.

Етап 1. Вибір елементної бази і типу САПР. Необхідно виявити специфічні вимоги проекту шляхом аналізу завдання (у загальному випадку технічного завдання, ТЗ) з метою вибору певного сімейства ПЛІС певної фірми. Вказаний вибір здійснюється на основі аналізу логічних, конструктивних, експлуатаційних, вартісних характеристик ВІС ПЛ, а також на основі оцінки можливостей САПР, що підтримує проектування на основі вибраної ПЛІС. У разі, коли вибір САПР визначений наявним технічним заділом і досвідом проектувальника, уточнюється тільки сімейство ПЛІС. Вибір сімейства повинен задовольняти вимогам ТЗ, наприклад, відповідності певним інтерфейсним стандартам, об'єму вбудованої пам'яті, швидкісним параметрам.

Етап 2. Формалізований опис проєктованого пристрою. ТЗ є сумішшю словесного і технічного опису. Тому необхідно виконати його формалізацію. Формалізований опис представляється у вигляді сукупності основних блоків пристрою (або алгоритму) і їх взаємозв'язків. При цьому реалізується декомпозиція об'єкту проектування з формуванням узагальненої структурної схеми. На цьому етапі САПР використовуються порівняно рідко. Проте сучасні САПР типу САПР Quartus фірми Altera виконують декомпозицію проекту за допомогою спеціальних блокових редакторів.

При виконанні курсового проекту декомпозиція виконується проектувальником при наявності необхідності в цьому.

Існують безкоштовні версії САПР: MAX+Plus II Baseline фірми Altera, WebPack фірми Xilinx, Actel DeskTOP фірми Actel, які можуть використовуватися в проєкті.

В даних методичних вказівках надана інформаційна підтримка для виконання курсового проекту з використанням САПР MAX+Plus II Baseline (додаток В).

Етап 3. Розробка структури проекту. Цей етап є найбільш творчою частиною створення проекту. Проектувальник представляє об'єкт проектування у вигляді, який сприймається САПР. Опис проекту може бути структурним або поведінковим.

Структурний опис відображає структуру блоків і дозволяє представити об'єкт проектування у вигляді сукупності компонент і зв'язків між ними. Структурний опис з використанням примітивів і мегафункцій, що знаходяться в бібліотеках САПР, просто реалізується в графічному редакторі САПР. Цей стиль зручний при описі простих пристроїв (суматорів, мультиплексорів тощо). Графічне представлення проекту в САПР може реалізуватися також у базисі графічних символів, створюваних проектувальником.

Поведінковий опис відображає правила функціонування кожного з блоків узагальненої структури. У цьому стилі зручно розробляти пристрої високого рівня складності (складні послідовнісні схеми, арбітри шин, контролери тощо).

Мови опису цифрової апаратури високого рівня (наприклад, VHDL) дозволяють виконувати як структурний, так і поведінковий опис об'єкту. Ці описи представляються у вигляді сукупності алгоритмічних виразів. Робота з VHDL-файлами здійснюється в текстовому редакторі САПР.

Вибір виду опису проекту способами САПР робить істотний вплив на якість кінцевого результату і час проектування. Проектувальнику необхідно вибирати між наочністю графічного способу опису і ефективністю текстового опису на мові високого рівня. Можна використовувати комбінований підхід, при якому окремі блоки описуються на мові високого рівня, а їх об'єднання виконується в графічному редакторі.

При розробці цифрових пристроїв іноді буває доцільним розбиття їх на операційний блок (ОБ), що здійснює перетворення даних, і блок керування (БК), що забезпечує виконання операційним блоком заданої послідовності операцій.

При описі структури операційного блоку зручно користуватися описами типових функціональних вузлів (регістрів, буферних схем, багатофункціональних логічних блоків), що присутні в бібліотеці САПР. Опис роботи БК зручно виконувати як в графічній, так і в текстовій формах.

Етап 4. Компіляція проекту. Синтез проектованого пристрою для його фізичної реалізації в ПЛІС виконується поетапно, декількома компіляторами. На етапі логічного синтезу виробляється представлення проектованого пристрою на вентильному рівні, тобто у вигляді

логічної схеми пристрою з урахуванням базису реалізації. На цьому етапі виконується і оптимізація логічного проекту. Результатом логічного синтезу є список зв'язків (Netlist), що є текстовою формою кодування логічної схеми.

Процес компіляції складається з ряду послідовних дій, які САПР виконує автоматично. При компіляції відбувається реалізація проекту в кристалі (виконуються етапи розміщення, трасування). Результатом компіляції при використуванні САПР фірм-виробників ВІС програмованої логіки є завантажувальний файл, що дозволяє виконувати конфігурацію вибраної ПЛІС. Створюється також файл звіту з інформацією про процес компіляції і його результати.

При наявності помилок в описі проекту в процесі компіляції автоматично генеруються попередження (Warnings) і повідомлення про помилки (Error messages). Проектувальнику необхідно усунути помилки в описі та прийняти до відома попередження.

Етап 5. Верифікація проекту. З метою верифікації проекту виконується часове моделювання при різних вхідних тестових наборах, які задаються проектувальником в редакторі часових діаграм, що входить до складу САПР. Для тестування можна розробити також спеціальну тестову програму.

Результатом часового моделювання є часові діаграми роботи пристрою.

Етап 6. Визначення часових характеристик розробленого пристрою. У САПР MAX+PLUS II передбачене автоматичне обчислення наступних часових параметрів:

- мінімальних і максимальних затримок між джерелами (вхідними сигналами) і приймачами (вихідними сигналами), інформація про які зображується у вигляді матриці затримок;
- максимально допустимої частоти синхронізації елементів пам'яті, використовуваних в проекті;
- часів передустановки і утримання сигналів, що гарантують надійне спрацювання схем при фіксації сигналів в елементах пам'яті.

При проектуванні необхідно виконати аналіз вказаних параметрів, що дозволяє знайти в проекті потенційно збійні місця і з'ясувати причини непередбаченої поведінки проектованого пристрою.

4.2.3 Методика синтезу послідовнісних схем на основі моделей керуючих мікропрограмних автоматів з жорсткою логікою

4.2.3.1 Мікропрограмні автомати з жорсткою логікою. При описі широкого класу цифрових систем доцільне їх представлення у вигляді сукупності операційного (ОБ) і керуючого (КБ) блоків. ОБ здійснює перетворення даних (наприклад, виконує операції множення, додавання), а БК забезпечує виконання операційним блоком заданої у часі послідовності операцій, для чого БК передає на входи ОБ послідовність керуючих сигналів, формування яких в БК залежить від зовнішніх сигналів і результатів операцій в ОБ.

При описі алгоритму роботи БК зручно використовувати моделі структурних автоматів різних типів.

Мікропрограмний автомат (МПА) є однією з форм завдання структурного автомата для випадку, коли вхідні і вихідні сигнали автомата закодовані булевими змінними.

Мікропрограмний автомат – це керуючий автомат, що формує мікрокоманди. Вхідний алфавіт МПА співпадає з множиною сигналів логічних умов, вихідний алфавіт – з множиною сигналів на виконання окремих мікрооперацій. Алфавіт станів і функцій переходів задається графом автомата.

Мікрокоманда – це сукупність мікрооперацій, виконуваних одночасно за один такт автоматного часу.

Мікрооперація – це елементарний неподільний акт обробки інформації на одному такті.

Логічні умови – це змінні для задання порядку проходження мікрокоманд. Перевірка значення логічної умови на кожному такті роботи автомата дозволяє визначити мікрокоманду, яка буде реалізовуватись в наступному такті. Тобто послідовність виконання мікрокоманд визначається функціями переходу - логічними функціями, аргументами яких є вказані логічні умови.

Сукупність мікрокоманд і функцій переходу утворює *мікропрограму*.

Щоб побудувати схему БК, необхідно задати мікропрограму роботи ОБ.

Якщо структура МПА включає схему, що складається з елементів пам'яті і комбінаційних вузлів, то такий автомат є автоматом з жорсткою логікою.

4.2.3.2 Розробка граф-схеми алгоритму (ГСА). При проектуванні різних пристроїв заздалегідь може бути складена змістовна граф-схема алгоритму. Розробка змістовної граф-схеми є достатньо інтуїтивною.

Як ілюстрація на рис. 4.1,а представлена змістовна граф-схема алгоритму виконання операції $D = 2A^2 + 0,5B$, що приводиться в [19]. У початковому стані операнд A записаний в регістрах $P1$ і C , операнд B – в регістрі $P2$. У першому такті виконується подвоєння A в $P1$ і ділення B на 2 в $P2$ шляхом зсуву слів на один розряд ліворуч і праворуч відповідно. Далі до вмісту $P2$ A разів додається слово, записане в $P1$. Після кожного додавання вміст C зменшується на 1. Обчислення закінчуються при виконанні умови $C = 0$. Результат операції формується в $P2$.

При синтезі МПА на основі змістовної ГСА спочатку формується *відмічена ГСА*.

Відмічена ГСА – це орієнтований зв'язний граф, що включає вершини чотирьох типів: початкову, кінцеву, операторну і умовну. Входи і виходи вершин з'єднуються дугами, направленими від виходу однієї вершини до входу іншої.

Відмічена ГСА повинна задовольняти наступним умовам: кожен вихід вершини з'єднується тільки з одним входом іншої вершини; дозволяється в різних умовних вершинах записувати однакові логічні умови; один з виходів умовної вершини може з'єднуватися з її входом, що неприпустимо для операторної вершини; в кожній операторній вершині записується оператор (мікрокоманда).

Одна і та ж ГСА може реалізовуватися різними структурними схемами – автоматом Мілі або автоматом Мура. Отримання різних структурних схем пов'язане з різними методиками відмітки ГСА.

Відмітка ГСА автомата Мілі виконується так:

- символом $S0$ відмічається вхід вершини, що слідує за початковою вершиною; а також вхід кінцевої вершини;
- символами $S1, \dots, Sm$ відмічаються входи всієї решти вершин, що йдуть за операторними;
- змістовні терміни мікрооперацій і логічних умов замінюються їх умовними позначеннями.

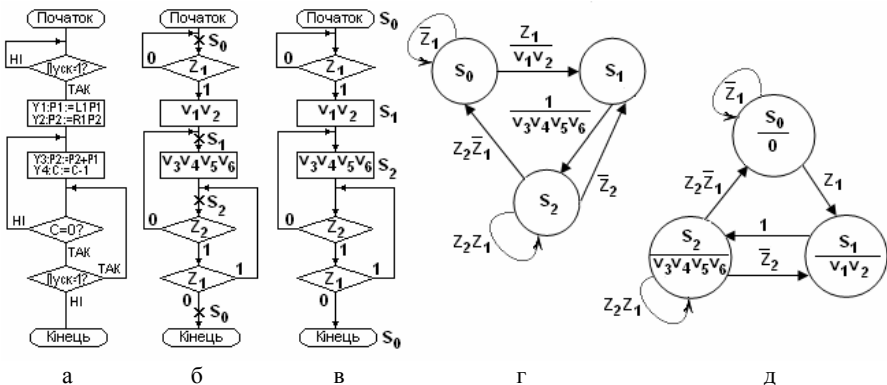
При побудові автомата Мура виконуються такі самі дії, але відмічаються не входи вершин, а самі вершини.

Відмічені ГСА автоматів Мілі й Мура, що відповідають змістовній граф-схемі алгоритму, представленою на рис. 4.1,а, показані на рис. 4.1,б і рис. 4.1,в відповідно. При їх формуванні враховувалося, що мікрооперації Y1, Y2 і Y4 керуються сигналами v_1 , v_2 , v_6 . Для виконання мікрооперації Y3 необхідно подати три сигнали: v_3 , v_4 , v_5 .

4.2.3.3 Побудова графа автомата. Після отримання відміченої ГСА будується граф автомата Мілі або Мура. Число вершин графа рівне числу станів автомата, тобто $S_m + 1$.

При побудові графа автомата Мілі кожній відмітці ГСА ставиться у відповідність вершина графа. Дві поименовані вершини графа з'єднуються дугою, якщо існує шлях переходу в ГСА від однієї відмітки до іншої з такими ж іменами. Поряд з дугою записується кон'юнкція логічних умов, відповідна цьому шляху, а так само множина мікрооперацій з операторної вершини, через яку проходить цей шлях. Якщо шляхів декілька, то записується диз'юнкція кон'юнкцій. Якщо між операторними вершинами перехід безумовний, то замість кон'юнкції ставиться "1", що означає, що даний перехід відбувається під час надходження синхросигналу.

Граф автомата Мілі приведений на рис. 4.1,г, граф автомата Мура – на рис. 4.1,д.



а - змістовна граф-схема алгоритму; б - відмічена ГСА автомата Мілі; в - відмічена ГСА автомата Мура; г - граф автомата Мілі; д - граф автомата Мура
Рисунок 4.1 - Граф-схеми і графи автоматів

При описі МПА з великим числом станів і переходів наочність графічного уявлення втрачається, при цьому переважно задавати граф автомата у вигляді списку – структурної таблиці МПА, на підставі якої формуються вирази функцій збудження і функцій переходів.

4.2.3.4 Структурний синтез цифрового автомата. Для подальшого синтезу автомата використовується метод структурного синтезу, в якому можна умовно виділити наступні етапи:

- кодування;
- вибір типів елементів пам'яті автомата;
- вибір структурно-повної системи елементів;
- формування логічних функцій виходів автомата і функцій збудження тригерів;
- побудова логічної структурної схеми автомата.

Кодування – це процес заміни символів вхідного, вихідного алфавітів і алфавіту внутрішніх станів автомата двійковими кодами.

Елементами пам'яті послідовнісних автоматів найчастіше є синхронізовані фронтом синхроімпульсів JK-, RS-, D- і T-тригери. Для тригерів характерні чотири типи переходів: $0 \rightarrow 0$, $0 \rightarrow 1$, $1 \rightarrow 0$, $1 \rightarrow 1$. Реалізація цих переходів виконується під впливом відповідних сигналів збудження (таблиця 4.1).

Таблиця 4.1 – Сигнали збудження тригерів

Переходи	Сигнали збудження тригерів			
	J , K	S , R	D	T
$0 \rightarrow 0$	0 , *	0 , *	0	0
$0 \rightarrow 1$	1 , *	1 , 0	1	1
$1 \rightarrow 0$	*, 1	0 , 1	0	1
$1 \rightarrow 1$	*, 0	*, 0	1	0

У таблиці (4.1) знак "*" означає, що відповідний перехід здійснюється як при нульовому, так і при одиничному значенні сигналу.

Функції збудження тригерів є булевими функціями. Для їх реалізації у вигляді комбінаційних схем використовується яка-небудь функціонально-повна система логічних елементів.

Після опису сигналів збудження і вихідних сигналів логічними функціями та їх мінімізації здійснюється представлення вказаних функцій у вибраному базисі, наприклад, "І-НІ", "АБО-НІ".

На основі отриманих мінімізованих булевих функцій розробляється логічна структура автомата, яка складається з комбінаційних схем функцій збудження, комбінаційних схем формування вихідних сигналів і елементів пам'яті.

Ілюстрація використання методики проектування схем послідовного типу на основі автоматних моделей розглянута у прикладах, наведених у додатку Г.

4.2.3.5 Приклад синтезу цифрового автомата. Нехай необхідно синтезувати схему керуючого автомата Мура для управління роботою *n*-розрядного суматора, що має дві вхідні *n*-розрядні шини для прийому чисел *A* і *B*, *n*-розрядну вихідну шину для видачі результату *C*. На суматор подаються керуючі сигнали: v_1 – установка суматора в 0; v_2 – прийом і порозрядне складання вмісту суматора з числом *A*; v_3 – прийом і порозрядне складання вмісту суматора з числом *B*; v_4 – вироблення перенесень і складання їх з вмістом суматора; v_5 – видача результату.

Рішення. Операції заданого суматора можна описати у вигляді мікропрограми, представленої в табл.4.2.

Таблиця 4.2 – Операції суматора і мікропрограма його роботи

Операція	Умова	Мікропрограма
Скидання суматора	Z_0	$МП_0 = Hv_1v_k$
Прийом числа <i>A</i>	Z_1	$МП_1 = Hv_1v_2v_k$
Прийом числа <i>B</i>	Z_2	$МП_2 = Hv_1v_3v_k$
$C = A \oplus B$	Z_3	$МП_3 = Hv_1v_2v_3v_k$
$C = A + B$	Z_4	$МП_4 = Hv_1v_2v_3v_4v_k$
Видача результату	Z_5	$МП_5 = Hv_5v_k$

У таблиці 4.2 символом *H* позначається початкова логічна умова, що ініціює виконання заданої операції при $H = 1$; символом v_k позначається порожня мікрооперація, за допомогою якої керуючий автомат сигналізує зовнішньому пристрою про завершення операції.

На підставі таблиці 4.2 складається граф-схема мікропрограми (рис. 4.2,а) і виконується її відмітка. На підставі відміченої ГСА складається граф переходів (рис. 4.2,б) і структурна таблиця МПА Мура (табл.4.3), в якій вказуються всі можливі переходи.

Таблиця 4.3 – Структурна таблиця МПА Мура

Поточний стан	Код поточного стану	Наступний стан	Код наступного стану	Умови переходу	Функції збудження тригерів
	$Q_3^n Q_2^n Q_1^n$		$Q_3^{n+1} Q_2^{n+1} Q_1^{n+1}$		$D_3 D_2 D_1$
S_0	0 0 0	S_0	0 0 0	$\overline{H}$	0 0 0
S_0	0 0 0	S_1	0 0 1	$H\overline{z}_5$	0 0 1
S_0	0 0 0	S_5	1 0 1	$H z_5$	1 0 1
S_1	0 0 1	S_2	0 1 0	$\overline{z}_0 \overline{z}_2$	0 1 0
S_1	0 0 1	S_3	0 1 1	$\overline{z}_0 z_2$	0 1 1
S_1	0 0 1	S_6	1 1 0	z_0	1 1 0
S_2	0 1 0	S_3	0 1 1	$\overline{z}_1$	0 1 1
S_2	0 1 0	S_6	1 1 0	z_1	1 1 0
S_3	0 1 1	S_4	1 0 0	z_4	1 0 0
S_3	0 1 1	S_6	1 1 0	$\overline{z}_4$	1 1 0
S_4	1 0 0	S_6	1 1 0	1^*	1 1 0
S_5	1 0 1	S_6	1 1 0	1^*	1 1 0
S_6	1 1 0	S_0	0 0 0	1^*	0 0 0

* - безумовний перехід.

Відповідно таблиці 4.3 складаються логічні рівняння для функцій збудження тригерів:

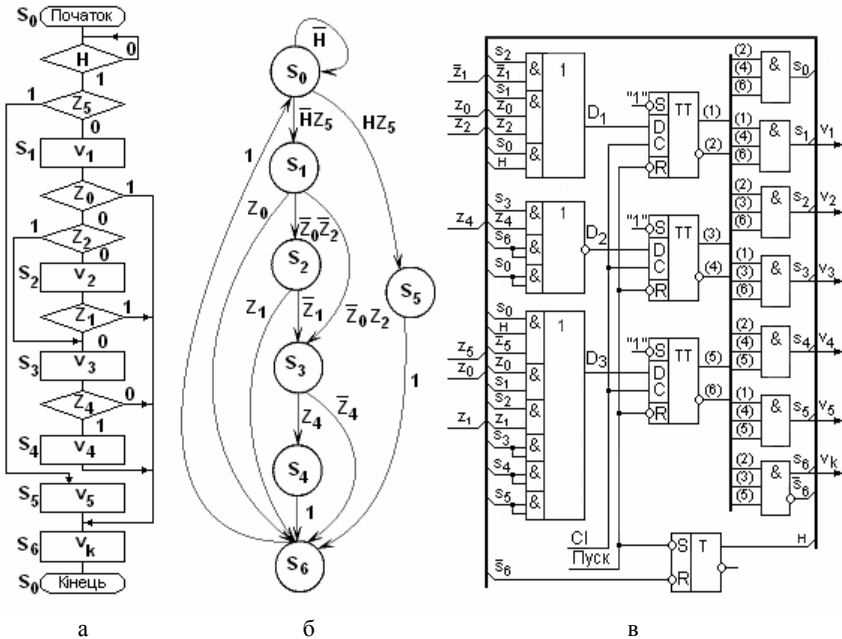
$$D_1 = s_0 H \vee s_1 z_0 z_2 \vee s_2 \overline{z}_1;$$

$$\overline{D}_2 = s_0 \vee q_6 \vee q_3 z_4; \text{ тоді } D_2 = \neg(s_0 \vee q_6 \vee q_3 z_4);$$

$$D_3 = s_0 H z_5 \vee s_1 z_0 \vee s_2 z_1 \vee s_3 \vee s_4 \vee s_5.$$

Логічна схема керуючого автомата Мура приведена на рис. 4.2,в. Розшифровку станів автомата виконує перетворювач, побудований на логічних елементах I , який підключений до виходів D -тригерів і виробляє сигнали: $v_1 = s_1; v_2 = s_2; v_3 = s_3; v_4 = s_4; v_5 = s_5$. Для управлін-

ня схемою використаний $\overline{RS}$ -тригер, який по сигналу "Пуск" встановлюється в одиничний стан ($H = 1$) і повертається в нульовий стан ($H = 0$) по сигналу кінця операції ($\bar{v}_k = \bar{s}_6$).



а - граф-схема мікропрограми; б - граф переходів автомата;
в - логічна схема автомата

Рисунок 4.2 – Автомат Мура

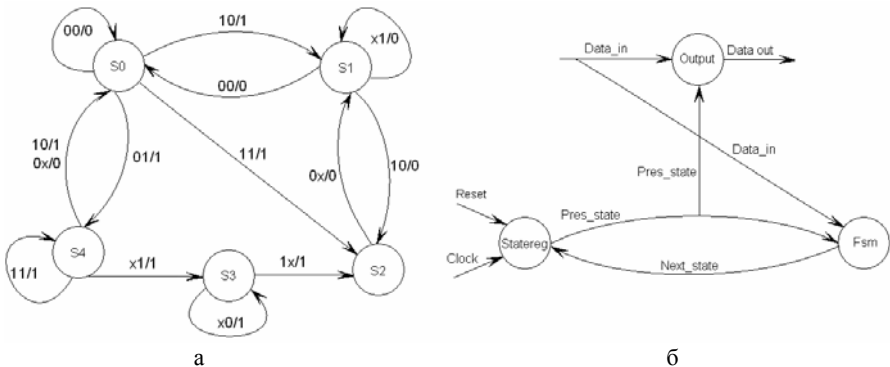
4.3 Опис послідовнісних пристроїв автоматними моделями у VHDL – кодах

ПЛІС, виконані по архітектурі FPGA, мають достатнє число тригерів, тому використання автоматних моделей дозволяє одержати достатньо швидкодіючу і в той же час наочну реалізацію пристрою при прийнятних витратах ресурсів.

Нижче розглядається приклад проектування послідовного пристрою на базі автомата Мілі.

VHDL – код (додаток Д) реалізує поведінку кінцевого автомата, заданого графом (рис. 4.3,а). Архітектурний опис цього кінцевого автомата містить два внутрішні сигнали: *Pres_state* і *Next_state*. Сигнал *Pres_state* призначений для зберігання поточного стану автомата. Фізично він реалізується у формі декількох тригерів (регістра відповідної розрядності). Сигнал *Next_state* використовується для визначення наступного стану – стану, який стане поточним в наступному такті. Значення цього сигналу визначається на базі значень вхідних сигналів і поточного стану автомата. Фізично цей сигнал реалізується у формі ліній зв'язку.

Поведінка автомата представляється у вигляді сукупності трьох процесів (рис. 4.3,б).



а –граф автомата; б - опис функціонування кінцевого автомата у вигляді сукупності процесів

Рисунок 4.3 – Кінцевий автомат Мілі з п'ятьма станами

У процесі *Fsm* визначається наступний стан автомата. Фізична реалізація такого процесу, як правило, є комбінаційною схемою.

Процес *Statereg* має список чутливості, в який входять сигнали *Reset* (обнулення) і *Clock* (синхросигнал). В процесі *Statereg* виконується власне перехід із стану в стан по відповідному фронту сигналу *Clock*. В даному випадку перехід здійснюється по висхідному фронту тактового імпульсу. По сигналу *Reset* = 0 автомат встановлюється в початковий стан *ST0*.

В процесі *Output* відповідно до поточного стану автомата визначається стан виходу *Data_out*.

4.4 Методика проектування кільцевих лічильників

Кільцеві лічильники використовуються в системах управління для формування послідовностей імпульсів. Синтезуються вони на основі регістрів зсуву, для чого останній вихід регістра зсуву з'єднується з входом першого тригера регістра.

На діаграмі станів (рис. 4.4) вхідні сигнали відсутні, оскільки єдиним вхідним сигналом для лічильника є синхронізуючий сигнал. Послідовно на кожному виході кільцевого лічильника - $Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7$ – формується одиничний імпульс тривалістю в один період тактового сигналу. Ці імпульси можуть використовуватися для управління функціональними блоками цифрової системи.

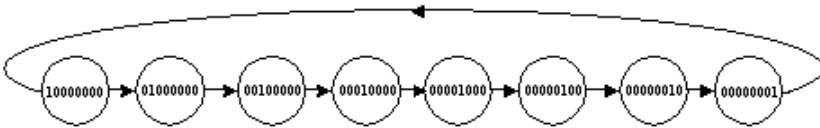


Рисунок 4.4 - Діаграма станів кільцевого лічильника

Початковий стан лічильника встановлюється за допомогою асинхронних установчих сигналів. Схема лічильника, відповідна приведеній діаграмі станів, показана на рис.4.5, а часова діаграма його роботи – на рис. 4.6.

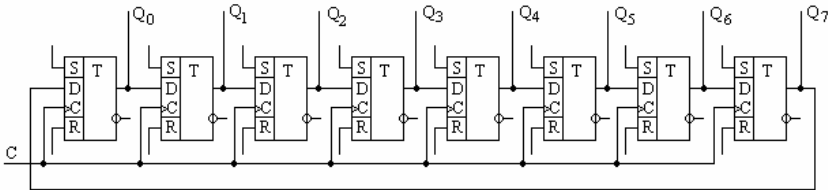


Рисунок 4.5 – Кільцевий лічильник з входами скидання і установки, що забезпечують установку лічильника в початковий стан

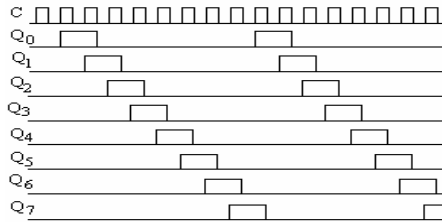


Рисунок 4.6 – Формування імпульсів управління за допомогою кільцевого лічильника

Якщо сигнал з інверсного виходу останнього тригера подати на вхід першого тригера, то утворюється перехресний кільцевий лічильник, званий лічильником Джонсона (рис. 4.7).

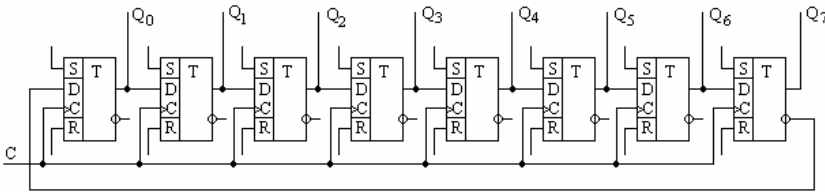


Рисунок 4.7 – Перехресний кільцевий лічильник

Лічильник Джонсона, ініціалізований в нульовому стані, формує повторювальну послідовність: 00000000, 10000000, 11000000, 11100000, 11110000, 11111000, 11111100, 11111110, 11111111, 01111111, 00111111, 00011111, 00001111, 00000111, 00000011, 00000001, 00000000, ... Кожна кодова комбінація відрізняється від сусідніх кодових комбінацій тільки одним бітом, що надзвичайно зручно для багатьох областей застосувань, оскільки при зміні кодової комбінації на наступну затримки в тригерах не викликають короткочасної появи інших комбінацій, що могло б відбутися при одночасній зміні вмісту декількох розрядів.

Початкові умови. При подачі живлення на логічну схему виходи всіх тригерів знаходяться, як правило, в довільних станах. Коли поступає на синхровхід лічильника перший синхроімпульс, може виникнути ситуація, коли формована на виходах тригерів кодова група не належить кодовій послідовності, яку необхідно сформувати. В цьому випадку виникає можливість генерації нової послідовності станів,

яка не включатиме жодної кодової комбінації з потрібної послідовності. Якщо ж на виходах лічильника з'явиться при включенні одна з комбінацій, що входять в потрібну послідовність, то лічильник почне працювати правильно.

Проблема початкових умов може бути розв'язана двома способами. Перший спосіб полягає в такому відображенні вихідних значень, коли будь-яка невизначена комбінація вихідних значень рано чи пізно переходить в одну із заданих вихідних комбінацій під час вступу чергового тактового імпульсу. У другому способі використовуються входи скидання/установки для примусової установки потрібних початкових умов.

На основі першого методу спроектований лічильник, на виході якого повторюється послідовність 000, 001, 011, 110, 000, ... Діаграма станів такого лічильника приведена на рис. 4.8,а, де показано, що всі невизначені кодові комбінації переходять в стан 000 під час вступу наступного ж тактового імпульсу. Карти Карно вхідних функцій тригерів зображені на рис. 4.8,б, а схема – на рис. 4.8,в.

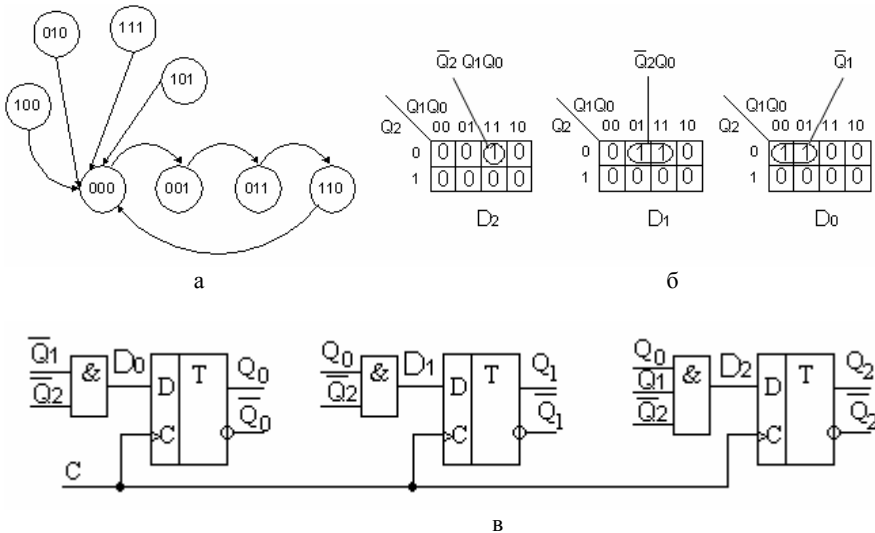


Рисунок 4.8 – Лічильник з установкою початкового стану

5 ВКАЗІВКИ ЩОДО ОФОРМЛЕННЯ І ЗАХИСТУ КУРСОВОГО ПРОЕКТУ

Курсовий проект складається з двох взаємозв'язаних частин: пояснювальної записки об'ємом 30-40 сторінок і графічної частини, яка складається з двох аркушів формату А1. В деяких випадках можуть бути представлені діючі макети, установки, вироби.

Пояснювальна записка – це документ, що вміщує опис будови та принципу дії електронного пристрою, який розробляється, а також обґрунтування прийнятих при його розробці технічних рішень.

Пояснювальна записка повинна містити: титульний лист; завдання на курсовий проект; реферат; зміст; вступ, у якому наводяться короткі загальні відомості про пристрій; теоретичні відомості, необхідні для виконання проекту, аналітичний огляд і аналіз можливих рішень; практичну (розрахункову) частину; висновки; перелік посилань.

Захист курсового проекту відбувається по графіку, затверджену кафедрою, у складі комісії з двох-трьох викладачів і в присутності студентів.

При захисті студент повинен зробити коротку доповідь (на 5-7 хвилин) по виконаному проекту і відповісти на поставлені запитання.

При оцінці курсового проекту комісія ураховує якість проекту і результати його захисту. При цьому звертається увага на повноту, якість і самостійність виконання поставленої задачі, акуратність оформлення пояснювальної записки і листів, складність завдання, наявність елементів наукових досліджень, додержання термінів захисту і якість захисту.

При захисті студент повинен:

- мати відомості про етапи проектування пристрою;
- знати особливості і архітектуру елементної бази, що використані в проекті;
- знати принцип роботи пристрою;
- володіти методикою синтезу цифрового пристрою;
- вміти критично оцінювати отримані результати.

Студент, що не подав курсовий проект у зазначений термін, або не захистив його без поважної причини, вважається таким, що має академічну заборгованість.

6 РЕКОМЕНДОВАНА ЛІТЕРАТУРА

Основна література

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

2 Стешенко В.Б. ПЛИС фирмы "Altera": Элементная база, система проектирования и языка описании аппаратуры. – М.: Издательский дом "Додэка-XXI", 2002. – 576 с.

3 Стешенко В.Б. EDA. Практика автоматизированного проектирования радиоэлектронных устройств. – М.: Издательство "Нолидж", 2002. – 768 с.

4 Бибило П.Н. Основы языка VHDL. – М.: СОЛОН-Р, 2002. – 224 с.

5 Бибило П.Н. Синтез логических схем с использованием языка VHDL. – М.: СОЛОН-Р, 2002. – 384 с.

6 Угрюмов Е.П. Цифровая схемотехника. – СПб.: БХВ-Петербург, 2000. - 528 с.

7 Норенков И.П., Маничев В.Б. Основы теории и проектирования САПР: Учеб. для втузов по спец. «Вычислительные машины, компл., сист. и сети». - М.: Высш шк., 1990.- 335 с.

8 Норенков И.П., Маничев В.Б. Системы автоматизированного проектирования электронной и вычислительной аппаратуры.: Учебн. пособие для вузов. - М.: Высш. шк., 1983.- 272 с.

9 Теоретические основы САПР: Учебник для вузов / В.П. Корячко, В.М. Курейчик, И.П. Норенков. - М.: Энергоатомиздат, 1987. - 400 с.

Додаткова література

10 Комолов Д.А., Мяльк Р.А., Зобенко А.А., Филиппов А.С. Системы автоматизированного проектирования фирмы Altera MAX+plus II и Quartus II / Краткое описание и самоучитель. – М.: ИП РадиоСофт, 2002. – 352 с.

11 Сергиенко А.М. VHDL для проектирования вычислительных устройств. – К.: ЧП "Корнейчук", ООО "ТИД "ДС", 2003. – 208 с.

12 Стешенко В.Б. ПЛИС фирмы ALTERA: проектирование устройств обработки сигналов. – М.: ДОДЭКА, 2000. – 128 с.

13 Стешенко В. Школа разработки аппаратуры цифровой обработки сигналов на ПЛИС // Chip News, 1999, № 8-10; 2000, № 1-3.

14 Савельев А.Я. Прикладная теория цифровых автоматов: Учебник. – М.: Высш. школа, 1987. – 272 с.

15 Автоматизация проектирования БИС. В 6 кн.: Практик. пособие. Кн. 1. Г.Г. Казеннов, А.Г. Соколов. Принципы и методология построения САПР БИС / Под ред. Г.Г. Казеннова. – М.: Высш. шк., 1990. – 142 с.

16 Автоматизация проектирования БИС. В 6 кн.: Практик. пособие. Кн. 2. П.В. Савельев, В.В. Коняхин. Функционально-логическое проектирование БИС / Под ред. Г.Г. Казеннова. – М.: Высш. шк., 1990. – 156 с.

17 Системы автоматизированного проектирования в радиоэлектронике: Справочник / Е.В. Авдеев, А.Т. Еремин, И.П. Норенков, М.И. Песков; Под ред. И.П. Норенкова. – М.: Радио и связь, 1986. – 368 с.

18 Алексенко А.Г., Шагурин И.И. Микросхемотехника: Учебное пособие для вузов. – М.: Радио и связь, 1989. – 440 с.

19 Прикладная теория цифровых автоматов / К.Г. Самофалов, А.М. Романкевич и др.: Под ред. К.Г. Самофалова. – К.: Выща шк., 1987. – 375 с.

Додаток А

Варіанти завдань до курсового проекту

Темою курсового проекту є: "Проектування цифрового пристрою на ПЛІС".

У залежності від варіанту назва пристрою конкретизується. Наприклад, тема може бути сформульована так: "Проектування генератора послідовностей Уолша на ПЛІС".

Нижче наведені варіанти завдань до курсового проекту.

1 Генератор меандрових послідовностей.

Вказівка. Пристрій має цикл роботи, рівний 16 тактам, і видає на виході при подачі відповідної дворозрядної команди: 00, 01, 10, 11 одну з чотирьох меандрових послідовностей: 01010101010101; 0011001100110011; 0000111100001111; 0000000011111111. Схему необхідно синтезувати на основі лічильника і мультиплексора, на адресні входи якого подаються вказані команди.

2 Керований підсумовуючий лічильник з рівнобіжним переносом.

Вказівка. Якщо керуючий сигнал $a = 0$, то модуль лічби лічильника дорівнює 8, якщо $a = 1$, то модуль лічби лічильника дорівнює 5.

3 Реверсивний лічильник з рівнобіжним переносом з коефіцієнтом лічби, рівним 6.

Вказівка. X_1 і X_2 – керуючі сигнали. Якщо $X_1=1$ і $X_2=0$, то лічильник виконує підсумовування; якщо $X_1=0$ і $X_2=1$, то лічильник виконує віднімання. При $X_1=X_2$ показання лічильника не змінюються.

4 Генератор послідовностей Уолша третього порядку.

Вказівка. Схема синтезується на основі чотирирозрядного лічильника, до виходів якого підключається комбінаційний цифровий вузол (КЦВ), що формує сигнали $Y_0 \dots Y_7$ у виді рівнобіжного коду. У свою чергу, до виходів КЦВ підключається мультиплексор з 9-ю інформаційними входами, на один із яких подається стробуючий імпульс T' . Для виключення небезпечних змагань повинна бути передбачена затримка зазначеного імпульсу відносно синхроімпульсу, що

надходить на вхід лічильника. На адресні входи мультиплексора надходять коди команд.

У таблиці А.1 представлені послідовності Уолша 3-го порядку.

Таблиця А.1 - Послідовності Уолша 3-го порядку

№ команди	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
0	1	1	1	1	1	1	1	1
1	0	0	0	0	1	1	1	1
2	0	0	1	1	1	1	0	0
3	1	1	0	0	1	1	0	0
4	1	0	0	1	1	0	0	1
5	0	1	1	0	1	0	0	1
6	0	1	0	1	1	0	1	0
7	1	0	1	0	1	0	1	0

5 Шестирозрядний розподільник імпульсів.

Вказівка. Розподільник імпульсів формує "1" по черзі на кожному з виходів. Схема будується на регістрі зсуву, що формується на D-тригерах з динамічним керуванням з переключенням по фронту синхросигналу. Вихід старшого розряду регістра з'єднується з його інформаційним входом. У вихідному стані в розряди регістра записані нулі.

6 Генератор псевдовипадкової послідовності.

Вказівка. Генератор являє собою п'ятирозрядний кільцевий регістр зсуву праворуч з логічним зворотним зв'язком. На виходах регістра формуються сигнали: Q1, Q2, Q3, Q4, Q5. У ланцюг зворотного зв'язку включений логічний елемент, що реалізує функцію: $X = Q3 \oplus Q4 \oplus Q5$, де X – інформаційний вхід регістра. У вихідному стані в молодший розряд регістра записана "1".

7 Кільцевий регістр із логічним зворотним зв'язком.

Вказівка. Регістр виконує зсув інформації праворуч, формуючи на виходах сигнали Q1, Q2, Q3, Q4. Виходи регістра Q3 і Q4 з'єднані з входами логічного елемента "Виключне АБО", вихід якого, у свою чергу, підключається до інформаційного входу регістра. У вихідному стані в старший розряд регістра записана "1".

8 Керуючий автомат-розподільник імпульсів на 16 напрямків.

Вказівка. Для побудови автомата-розподільника імпульсів використовується лічильник з модулем лічби, рівним 16, і чотиривходовий дешифратор.

9 Реверсивний восьмирозрядний кільцевий лічильник.

Вказівка. Початковий стан лічильника: 00000001. Якщо керуючий сигнал $a = 1$, то здійснюється зсув праворуч, якщо $a = 0$, - то ліворуч.

10 Цифровий автомат на шість вводів.

Вказівка. Цифровий автомат має цикл із 6 тактів. На виходах автомата формуються послідовності, представлені в таблиці А.2.

Таблиця А.2 – Послідовності на виходах цифрового автомату

№ такту	Y1	Y2	Y3	Y4	Y5	Y6
0	1	0	1	0	1	0
1	0	1	0	1	1	0
2	0	1	1	0	0	0
3	0	0	0	1	0	1
4	1	1	0	1	1	1
5	0	0	1	0	1	1

11 Чотирирозрядний пристрій порівняння чисел на компараторі і регістрі.

Вказівка. Регістр будується на D-тригерах.

12 П'ятирозрядний дільник зі змінним коефіцієнтом ділення на D-тригерах.

Вказівка. Максимальний коефіцієнт ділення дорівнює 32. Для зменшення коефіцієнта ділення необхідно передбачити входи дозволу E1, E2, E3, E4, E5. Частота вихідного сигналу визначається по формулі:

$$f_{\text{вих}} = (f / 32)(E_4 \cdot 2^4 + E_3 \cdot 2^3 + E_2 \cdot 2^2 + E_1 \cdot 2^1 + E_0 \cdot 2^0).$$

13 Віднімальний двійковий лічильник з рівнобіжним переносом з модулем лічби, рівним 10.

Вказівка. Лічильник повинний мати входи установки і будуватися на JK-тригерах.

14 Двійково - десятковий лічильник, що працює в коді 2421сд (код Айкена).

Вказівка. Лічильник повинний мати входи установки і будуватися на *JK*-тригерах.

15 Лічильник Джонсона з числом станів у циклі, рівним 10.

Вказівка. Усунути неробочі цикли (два по 10 і один із двох станів). Побудувати часові діаграми для цих циклів.

16 Чотирирозрядний підсумовуючий послідовнісний функціональний вузол (ПФВ).

Вказівка. ПФВ працює під керуванням зовнішніх сигналів X_2 і X_1 або як підсумовуючий двійковий лічильник ($X_2=X_1$), або як підсумовуючий двійково-десятковий лічильник у коді 8421 ($X_2=1, X_1=0$); при $X_2=X_1=0$ повинний забезпечуватися режим збереження; комбінація $X_2=0, X_1=1$ не використовується.

17 Чотирирозрядний дільник частоти.

Вказівка. При значеннях керуючих сигналів: $X_2X_1=11, 10, 01$ дільник повинний поділяти частоту вхідних тактових імпульсів у 16, 12, 10 разів. Щільність вихідного сигналу повинна дорівнювати двом. Комбінація 00 на керуючих входах не використовується.

18 Чотирирозрядний паралельний регістр для приймання даних з двох напрямків.

19 Чотирирозрядний лічильник Джонсона.

Вказівка. Задана послідовність переходів: 0,1; 1,3; 3,7; 7,17; 17,16; 16,14; 14,10; 10,0.

20 Генератор двійкової послідовності 1101001 на регістрі зсуву.

21 Багаторежимний чотирирозрядний буферний регістр.

Вказівка. До складу регістра повинні входити: 4 D-тригери, 4 буферних каскади з трьома станами на виході, тригер запиту магістралі.

Додаток Б

Приблизний календарний план виконання
курсowego проекту

1 Проробка літературних джерел	- 1 тиждень
2 Синтез логічної структури схеми пристрою	- 2 тижня
3 Розробка VHDL-коду	- 2 тижня
4 Реалізація проекту пристрою	- 2 тижня
5 Верифікація розробленого проекту	- 2 тижня
6 Аналіз отриманих результатів моделювання	- 1 тиждень
7 Написання і оформлення пояснювальної записки	- 2 тижня
8 Виготовлення креслень	- 2 тижня
9 Захист курсового проекту	- 2 тижня

Додаток В

Процедура розробки проекту в САПР MAX+PLUS II

Процедура розробки проекту в САПР MAX+Plus II полягає у виконанні проектувальником нижченаведених поетапних дій.

Створення робочої папки для розміщення файлів проекту. У директорії *MAXWORK* необхідно створити робочу папку, наприклад, під ім'ям *vlsi_1*.

Створення директорії проекту. Директорія створюється за допомогою завдання послідовності команд *File|Project|Name* і введенням імені проекту (наприклад, *vlsi_1*). При цьому вибирається створена робоча папка.

Створення текстового файлу. Якщо текстовий файл, створений в будь-якому текстовому редакторі (наприклад, в редакторі Word), вже існує, то необхідно його вміст записати в буфер пам'яті за допомогою введення команд *Правка|Выделить все|Ctrl+C*.

Далі слід активізувати текстовий редактор, задавши послідовність команд *Max+plusII|Text Editor*, а також перемістити інформацію з буфера пам'яті за допомогою натиснення поєднання клавіш *Ctrl+V*.

Якщо введений файл є програмою, написаною на мові VHDL, то його слід зберегти в директорії проекту з розширенням *.vhd*. При цьому задається послідовність команд *File|Save As* і вводиться ім'я файлу (наприклад, *vlsi_1.vhd*).

Виконання компіляції файлу. Компіляція виконується шляхом запуску додатку *Compiler*. За наявності помилок в програмі, їх слід усунути і виконати повторну компіляцію.

Створення Include-файлу і символу бібліотечного елементу. Include-файл створюється командами *File|Create Default Include File* і записується в бібліотеку користувача. Прочитати цей файл і уточнити назви входів і виходів можна за допомогою виклику додатку *Hierarchy Display*. При цьому відображаються всі модулі проекту і їх взаємозв'язки, а також всі типи файлів, сформовані в процесі обробки проекту. Include-файл відображається з розширенням *.inc*. Його активізація дозволяє проглянути вміст вказаного файлу.

Створення графічного файлу. Для виклику графічного редактора потрібно в меню *Menager* вибрати *Max+PlusII|Grafic Editor*.

Графічному файлу із схемою необхідно командою *File|Save As* привласнити ім'я з розширенням *.gdf* (наприклад, *vlsi_1.gdf*).

Після того, як функціональні блоки введені, потрібно ввести символи вхідних і вихідних портів. Їх необхідно імпортувати з бібліотеки примітивів. Для цього необхідно двічі клацнути мишею по порожньому полю графічного редактора. Відкриється діалогове вікно, в якому в меню *Symbol Libraries* вказана бібліотека знаходиться за адресою *c:\program file\maxplusii10.2\max2lib\prim*. Після подвійного клацання за вказаною адресою в меню *Symbol File* з'явиться список логічних елементів. З вказаного списку необхідно вибрати примітиви портів, які зберігаються в бібліотеці під іменами *input* і *output*.

Далі необхідно привласнити імена всім портам.

Симуляція. Симуляція – це процес функціонального моделювання роботи схеми. Перед виконанням моделювання необхідно створити тестові вектори, тобто задати значення вхідних сигналів. Для цієї мети можна використати редактор діаграм, який вибирається послідовністю команд *Max+PlusII|Waveform Editor*.

Коли вікно редактора відкриється, створюється файл (наприклад, з назвою *vlsi_1.scf*) послідовністю *File|Save As* і вказівкою назви файлу (наприклад, *vlsi_1.scf*) в рядку *File Name* діалогового вікна, що відкрилося.

Далі визначаються вхідні і вихідні сигнальні лінії схеми для процесу симуляції. Для цього використовуються сигнальні лінії, занесені в *SNF-файл* (*Simulator Netlist File*), створений на етапі компіляції схеми. Необхідно відкрити список доступних в *SNF-файлі* сигнальних ліній за допомогою введення *Node|Enter Node from SNF*. Відкриється екран з двома вікнами: *Available Nodes & Groups* і *Selected Nodes & Groups*. Після натиснення *List* в першому вікні з'явиться список вхідних і вихідних ліній з *SNF-файлу*. Необхідно скопіювати список вхідних і вихідних ліній в друге вікно, тобто створити список вибраних сигнальних ліній. Після введення *OK* у вікні графічного редактора відобразяться вхідні і вихідні лінії.

Далі задається кінцевий час симуляції введенням *File|End Time* і інтервал часової сітки *Options|Grid Size*.

Для установки значень вхідних сигналів можна скористатися одним із способів: за допомогою вертикального репера встановити клацанням з протяганням тривалість сигналу, а потім ввести його значен-

ня за допомогою кнопки символом "1" на лівій інструментальній панелі.

Для запуску пакету моделювання потрібно або ввести *File|Simulator*, або клацнути по кнопці симулятора на головній інструментальній панелі.

Якщо результати моделювання виявилися успішними, можна за допомогою виклику додатку *File|Timing Analyzer* відобразити таблицю "*Delay Matrix*", в якій записані затримки формування вихідних сигналів щодо вхідних сигналів.

Призначення ресурсів. Для призначення ресурсів фізичних пристроїв і проглядання результатів розводки, зроблених компілятором, викликається порівневий планувальник *File|Floorplan Editor*. У вікні планувальника можна побачити тип мікросхеми, яка вибиралася в проекті автоматично (при необхідності тип ПЛІС можна вибрати) і умовне графічне зображення вибраної ПЛІС з вказівкою під'єднуваних входів і виходів схеми.

Додаток Г

Ілюстрація використання методики проектування схем
послідовнісного типу

Приклад 1. Синхронний лічильник зворотного зв'язку за модулем 5 на базі двоступеневих *JK*-тригерів.

Граф зміни станів трирозрядного лічильника приведений на рис. В.1.

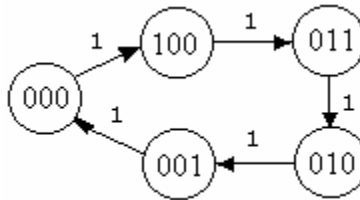


Рисунок В.1 – Граф зміни станів трирозрядного лічильника

Спочатку складається таблиця переходів і станів збудження *JK*-тригерів (таблиця В.1).

Таблиця В.1 – Переходи і стани збудження *JK*-тригерів

Попередній стан			Наступний стан			Сигнали збудження					
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	1	0	0	1	*	0	*	0	*
1	0	0	0	1	1	*	1	1	*	1	*
0	1	1	0	1	0	0	*	*	0	*	1
0	1	0	0	0	1	0	*	*	1	1	*
0	0	1	0	0	0	0	*	0	*	*	1

Далі мінімізуються функції J_1 , K_1 , J_2 , K_2 , J_3 і K_3 за допомогою карт Карно, враховуючи доvizначення на надлишкових вхідних наборах (рис. В.2).

Приклад 2. Синхронний лічильник за модулем 10 на базі регістра зсуву на D – тригерах.

Проектування дво-, три-, і чотирирозрядних лічильників на базі регістрів зсуву здійснюється шляхом реалізації графів станів, зображених на рис. В.4.

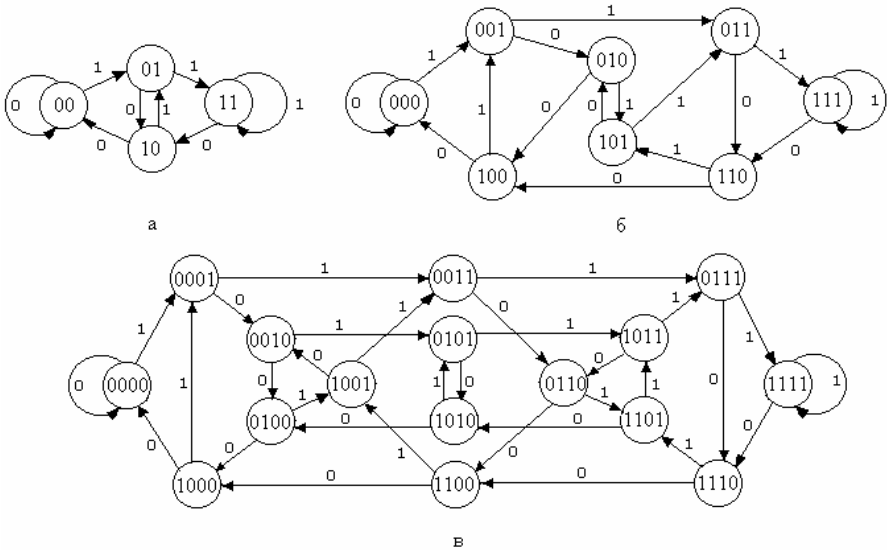


Рисунок В.4 – Графи станів: а – дво-; б – три-; в – чотирирозрядного регістрів зсуву

Щоб здобути десять різних станів, потрібний чотирирозрядний регістр зсуву, граф станів якого зображено на рис. В.4,в. На графі станів слід вибрати по напрямку обходу контур, який включає зміну 10 станів – наприклад: 0000-0001-0010-0101-1011-0111-1111-1110-1100-1000-0000. Для обраного контуру таблиця переходів і сигналів збудження D -тригерів показана нижче (табл. В.2).

Мінімізацію функцій D_1 , D_2 , D_3 і D_4 можна провести за допомогою карт Карно (рис. В.5).

Логічна схема проектованого лічильника зображена на рис. В.6.

Таблиця В.2 –Переходи і сигнали збудження D-тригерів

Попередній стан				Наступний стан				Сигнали збудження			
Q4	Q3	Q2	Q1	Q4	Q3	Q2	Q1	D4	D3	D2	D1
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	0
0	0	1	0	0	1	0	1	0	1	0	1
0	1	0	1	1	0	1	1	1	0	1	1
1	0	1	1	0	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	0	1	1	1	0
1	1	1	0	1	1	0	0	1	1	0	0
1	1	0	0	1	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0

		Q_2Q_1			
Q_4Q_3		00	01	11	10
	00	1	0	*	1
	01	*	1	1	*
	11	0	*	0	0
	10	0	*	1	*

$$D_1 = Q_2 \wedge \bar{Q}_3 \vee \bar{Q}_4 \wedge (\bar{Q}_1 \vee Q_3)$$

		Q_2Q_1			
Q_4Q_3		00	01	11	10
	00	0	1	*	0
	01	*	1	1	*
	11	0	*	1	0
	10	0	*	1	*

$$D_2 = Q_1$$

		Q_2Q_1			
Q_4Q_3		00	01	11	10
	00	0	0	*	1
	01	*	0	1	*
	11	0	*	1	1
	10	0	*	1	*

$$D_3 = Q_2$$

		Q_2Q_1			
Q_4Q_3		00	01	11	10
	00	0	0	*	0
	01	*	1	1	*
	11	1	*	1	1
	10	0	*	0	*

$$D_4 = Q_3$$

Рисунок В.5 - Мінімізація функцій лічильника за модулем 10

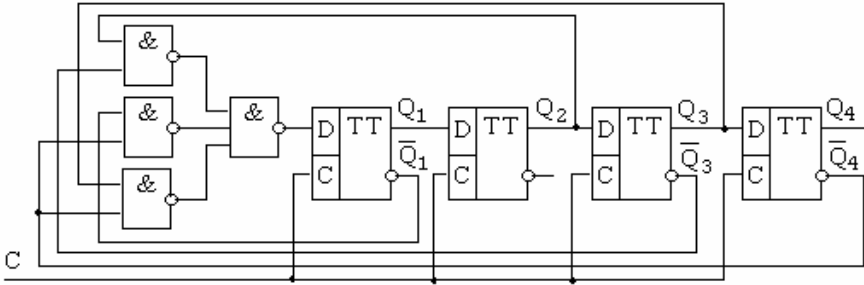


Рисунок В.6 – Синхронний лічильник за модулем 10 на базі регістра зсуву

Приклад 3. Генератор двійкової послідовності 0-1-1-1-0 на базі регістра зсуву на *RS*-тригерах.

Функціональна схема генератора двійкової послідовності складатиметься з реалізованого на базі регістра зсуву лічильника з відповідним модулем лічби і комбінаційного автомата, призначеного для перетворення вихідних сигналів тригерів у потрібну двійкову послідовність. При проектуванні генераторів двійкових послідовностей, що вміщують до 16 біт, використовуються граfi станів дво-, три-, і чотирьорозрядних регістрів зсуву (рис. В.4).

Щоб отримати двійкову послідовність, яка вміщує 6 біт, потрібний трирозрядний лічильник. Він може бути реалізований на базі регістра зсуву, граfi станів якого зображено на рис. В.4,б. На граfi станів слід вибрати за напрямом обходу контур, що включає зміну 6 станів. Наприклад: 000-001-011-111-110-100-000. Для вибраного контуру таблиця переходів і сигналів збудження *RS*-тригерів приведена нижче.

Таблиця В.3 –Переходи і сигнали збудження *RS*-тригерів

Попередній стан			Наступний стан			Сигнали збудження						Y
Q ₃	Q ₂	Q ₁	Q ₃	Q ₂	Q ₁	S ₃	R ₃	S ₂	R ₂	S ₁	R ₁	
0	0	0	0	0	1	0	*	0	*	1	0	0
0	0	1	0	1	1	0	*	1	0	*	0	1
0	1	1	1	1	1	1	0	*	0	*	0	1
1	1	1	1	1	0	*	0	*	0	0	1	1
1	1	0	1	0	0	*	0	0	1	0	*	1
1	0	0	0	0	0	0	1	0	*	0	*	0

Логічна функція Y визначає потрібну двійкову послідовність.

Мінімізація функцій S_1 , R_1 , S_2 , R_2 , S_3 , R_3 і Y проводиться за допомогою карт Карно з урахуванням довизначення на надлишкових наборах (рис. В.7).

Q_2Q_1		00	01	11	10
Q_3	0	0	*	*	1
	1	1	*	*	*

$$J_1 = Q_2 \vee Q_3$$

Q_2Q_1		00	01	11	10
Q_3	0	*	1	1	*
	1	*	*	*	*

$$K_1 = 1$$

Q_2Q_1		00	01	11	10
Q_3	0	0	0	*	*
	1	1	*	*	*

$$J_2 = Q_3$$

Q_2Q_1		00	01	11	10
Q_3	0	*	*	0	1
	1	*	*	*	*

$$K_2 = \bar{Q}_1$$

Q_2Q_1		00	01	11	10
Q_3	0	1	0	0	0
	1	*	*	*	*

$$J_3 = \bar{Q}_1 \wedge \bar{Q}_2$$

Q_2Q_1		00	01	11	10
Q_3	0	*	*	*	*
	1	1	*	*	*

$$K_3 = 1$$

Рисунок В.7 – Мінімізація функцій генератора двійкової послідовності 0-1-1-1-1-0

Логічна структура генератора двійкової послідовності зображена на рис. В.8.

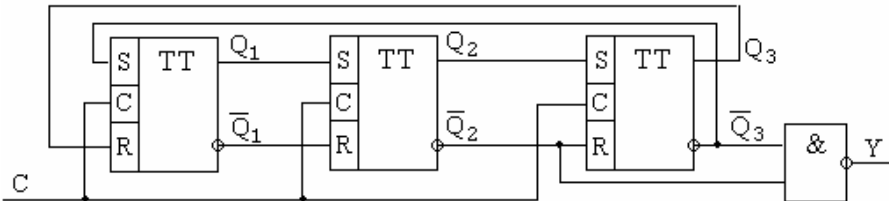


Рисунок В.8 – Генератор двійкової послідовності 0-1-1-1-1-0

Додаток Д

Роздруківка VHDL-програми "Автомат Мілі з п'ятьма станами"

```

library ieee;
-- Автомат Мілі з 5 станами
use ieee.std_logic_1164.all;
entity mealy is
port (clock, reset: in std_logic;
data out: out std_logic;
data in: in std_logic_vector (1 downto 0));
end mealy;
architecture behave of mealy is
type state_values is (st0, st1, st2, st3, st4);
signal pres_state, next_state: state_values;
begin
statereg: process (clock, reset)
--FSM perictp
begin
if (reset = '0') then
pres_state <= st0;
elsif (clock'event and clock = '1') then
pres_state <= next_state;
end if;
end process statereg;
fsm: process (pres_state, data_in)
--FSM - процес
begin
case pres_state is
when st0 =>
case data_in is
when "00" => next_state <= st0;
when "01" => next_state <= st4;
when "10" => next_state <= st1;
when "11" => next_state <= st2;
when others => next_state <= st0;
end case;
when st1 =>
case data_in is
when "00" => next_state <= st0;
when "10" => next_state <= st2;
when others => next_state <= st1
end case;
when st2 =>
case data_in is
when "00" => next_state <= st1;
when "01" => next_state <= st1;
when "10" => next_state <= st3;
when "11" => next_state <= st3;
when others => next_state <= st0;
end case;
when st3 =>
case data_in is
when "01" => next_state <= st4;
when "11" => next_state <= st4;
when others => next_state <= st3;
end case;
when st4 =>
case data_in is
when "11" => next_state <= st4;
when others => next_state <= st0;
end case;
end process fsm;
outputs: process (pres_state, data_in)
-- Процес outputs
begin
case pres_state is
when st0 =>
case data_in is
when "00" => data_out <= '0';
when others => data_out <= '1';
end case;
when st1 => data_out <= '0';
when st2 =>
case data_in is
when "00" => data_out <= '0';
when "01" => data_out <= '0';
when others => data_out <= '1';
end case;
when st3 => data_out <= '1';
when st4 =>
case data_in is
when "10" => data_out <= '1';
when "11" => data_out <= '1';
when others => data_out <= '0';
end case;
when others => data_out <= '0';
end case;
end process outputs;
end behave;

```