МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

# МЕТОДИЧНІ ВКАЗІВКИ

**до виконання курсового проекту**

з дисципліни

**„КОМП'ЮТЕРНЕ МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ ПРИСТРОЇВ ЦИФРОВОЇ ЕЛЕКТРОНІКИ"**

для студентів спеціальності
153 "Мікро- та наносистемна техніка", освітня програма
"Мікро- та наноелектронні прилади та пристрої"
денної і заочної форм навчання

2020

Методичні вказівки до виконання курсового проекту з дисципліни "Комп’ютерне моделювання та проектування пристроїв цифрової електроніки" для студентів спеціальності 153 "Мікро- та наносистемна техніка", освітня програма "Мікро- та наноелектронні прилади та пристрої" денної й заочної форм навчання / Укл. Н. М. Нагорна. – Запоріжжя: НУ «Запорізька політехніка», 2020. – 56 с.

Укладач: Н.М. Нагорна, ст. виклад.

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

Відповідальний за випуск: Коротун А.В., доц., канд.фіз.-матем. наук

Затверджено
на засіданні кафедри
мікро- та наноелектроніки

Протокол № 8
від “ 26 “ лютого 2020 р.

Рекомендовано до видання
НМК ФРЕТ
Протокол № 7
від “ 27 “ лютого 2020 р.

# ЗМІСТ

# 1 МЕТА І ЗАДАЧІ КУРСОВОГО ПРОЕКТУ

Метою курсового проекту (КП) є поглиблення і розширення теоретичних знань, вироблення 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.

Задачами КП є отримання досвіду проектування цифрового пристрою на базі програмованих логічних інтегрованих схем (ПЛІС) в системі автоматизованого проектування Quartus II Web Edition або MAX+PLUS II; набуття вміння настроювання середовища проектування ПЛІС під апаратну платформу вказаних САПР; отримання знань з класифікації, параметрів та характеристик ПЛІС, основного функціонального складу ПЛІС.

При роботі над курсовим проектом студент набуває досвіду для подальшої роботи над магістерською роботою і в майбутній інженерній та викладацькій 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 пристрою, що проектується;
- методику процесу проектування цифрових пристроїв на основі послідовнісних схем;
- прийоми функціонального моделювання цифрових пристроїв.

Після виконання курсового проекту студент повинен вміти:

- синтезувати структурну схему цифрового пристрою за заданим алгоритмом його роботи;
- виконувати проектування пристроїв з використанням сучасної САПР;
- реалізовувати типові алгоритми на мові програмування апаратних засобів VHDL і в схемному редакторі САПР;
- моделювати різноманітні режими роботи спроектованого пристрою і оцінювати його часові параметри.

# 2 ОРГАНІЗАЦІЙНІ ВКАЗІВКИ

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.

Робота над курсовим проектом повинна проводитись згідно з календарним планом. Приблизний календарний план виконання курсового проекту приводиться табл. 2.1.

Таблиця 2.1 – Приблизний календарний план виконання курсового проекту

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

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 (або презентацію).

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

# 3 МЕТОДИЧНІ ВКАЗІВКИ ДО ВИКОНАННЯ РОЗДІЛІВ КУРСОВОГО ПРОЕКТУ

## 3.1 Зміст пояснювальної записки курсового проекту

Приблизний зміст пояснювальної записки курсового проекту приведений нижче.

Титульний лист.

Завдання на курсовий проект.

Реферат.

Зміст.

Вступ.

Розділ 1. Основні теоретичні відомості.

Підрозділ 1.1. Обґрунтування вибору елементної бази для реалізації пристрою.

Підрозділ 1.2. Опис принципів роботи проектованого пристрою.

Розділ 2. Синтез цифрового пристрою заданого типу.

Підрозділ 2.1. Реалізація цифрового пристрою на мікросхемі програмованої логіки.

Підрозділ 2.2. Верифікація проекту схеми.

Підрозділ 2.3. Аналіз результатів проектування.

Висновки.

Перелік посилань.

Додатки.

## 3.2 Рекомендації до виконання етапів курсового проекту

### 3.2.1 Відомості про використовувану елементну базу

В проекті розробляється цифровий або цифро-аналоговий пристрій на базі програмованих логічних інтегрованих 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 "*І*", 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 вентильні 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 запам’ятовувальних пристроїв.

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.

### 3.2.2 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 програмованої логіки, 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 проектованого 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 (нова назва Intel) 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 та Quartus II фірми Altera, WebPack фірми Xilinx, Actel DeskTOP фірми Actel, які можуть використовуватися в проекті.

***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 блок операційний (БО), 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. 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 використовуванні 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 спрацьовування 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.

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

**3.2.3.1** Мікропрограмні автомати з жорсткою логікою. При описі широкого класу цифрових систем доцільне їх представлення у вигляді сукупності блоку операційного (БО) і блоку керування (БК). БО 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 БО послідовність керуючих сигналів, формування яких в БК залежить від зовнішніх сигналів і результатів операцій в БО.

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 завдання 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.

**3.2.3.2** 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.3.1, а представлена змістовна граф-схема алгоритму виконання операції $D = 2A^2 + 0{,}5B$, що приводиться у [11]. У початковому стані операнд *A* записаний в регістрах *P*1 і *C*, операнд *B* – в регістрі *P*2. У першому такті виконується подвоєння *A* в *P*1 і ділення *B* на 2 в *P*2 шляхом зсуву слів на один розряд ліворуч і праворуч відповідно. Далі до вмісту *P*2 *A* разів додається слово, записане в *P*1. Після кожного додавання вміст *C* зменшується на 1. Обчислення закінчуються при виконанні умови *C* = 0. Результат операції формується в *P*2.

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 слідкує за початковою вершиною; а також вхід кінцевої вершини;

-символами *S1*, …, *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.

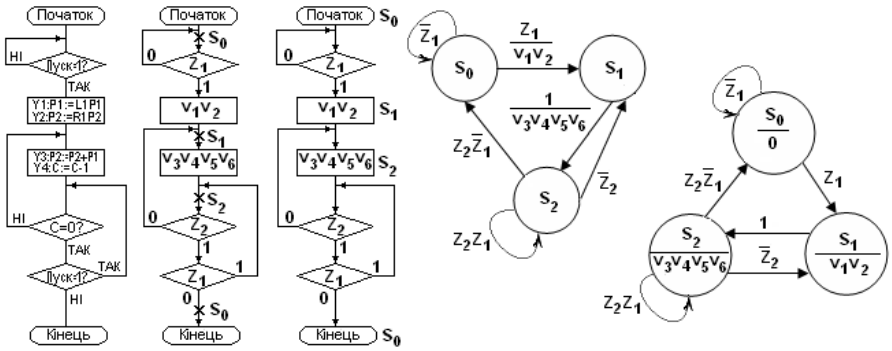
Відмічені ГСА автоматів Мілі й Мура, що відповідають змістовній граф-схемі алгоритму, представленому на рис. 3.1, а, показані на рис. 3.1, б і рис. 3.1, в відповідно. При їх формуванні враховувалося, що мікрооперації *Y*1, *Y*2 і *Y*4 керуються сигналами $v_1$, $v_2$, $v_6$. Для виконання мікрооперації *Y*3 необхідно подати три сигнали: $v_3$, $v_4$, $v_5$.

**3.2.3.3** Побудова графа автомата. Після отримання відміченої ГСА будується граф автомата Мілі або Мура. Число вершин графа рівне числу станів автомата, тобто *Sm+1*.

При побудові графа автомата Мілі кожній відмітці ГСА ставиться у відповідність вершина графа. Дві пойменовані 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 мікропрограмного автомата Мілі приведений на рис. 3.1, г, граф мікропрограмного автомата Мура – на рис. 3.1, д.




а б в г 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 3.1 - 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.

**3.2.3.4** Приклад синтезу мікропрограмного 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. 3.1.

Таблиця 3.1 – Операції суматора і мікропрограма його роботи

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

У таблиці 3.1 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 3.1 складається граф-схема мікропрограми (рис. 3.2, а) і виконується її відмітка. На підставі відміченої ГСА складається граф переходів (рис. 3.2, б) і структурна таблиця МПА Мура (табл.3.2), в якій вказуються всі можливі переходи.

Відповідно таблиці 3.3 складаються логічні рівняння для функцій збудження тригерів:

$$D_1 = s_0H \vee s_1z_0z_2 \vee s_2\bar{z}_1;$$
$$\overline{D}_2 = s_0 \vee q_6 \vee q_3z_4; \text{ тоді } D_2 = \neg(s_0 \vee q_6 \vee q_3z_4);$$
$$D_3 = s_0Hz_5 \vee s_1z_0 \vee s_2z_1 \vee s_3 \vee s_4 \vee s_5.$$

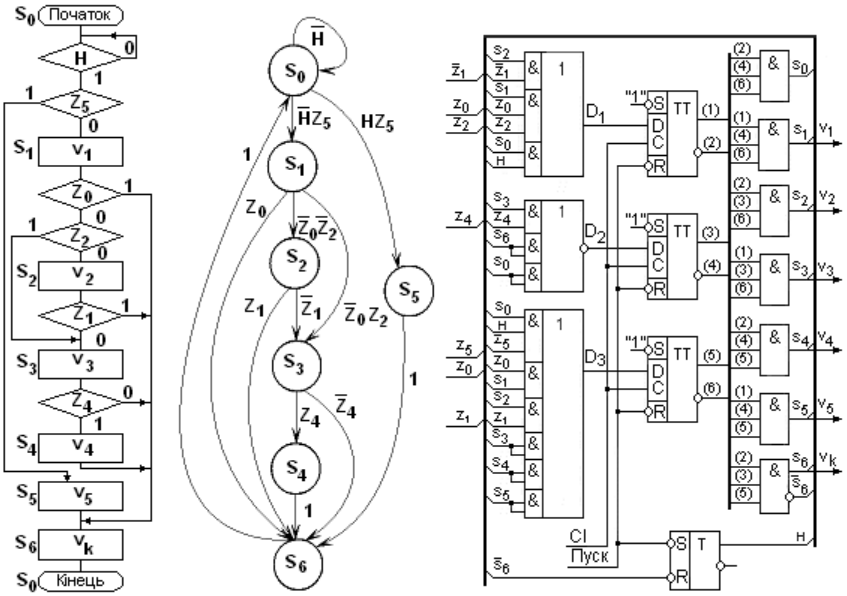
Логічна схема керуючого мікропрограмного автомата Мура приведена на рис. 3.2, в. Розшифровку станів автомата виконує перетворювач, побудований на логічних елементах $I$, який підключений до виходів $D$-тригерів і виробляє сигнали:

$y_1 = s_1; y_2 = s_2; y_3 = s_3; y_4 = s_4; y_5 = s_5$. Для управління схемою використаний $\overline{R}\overline{S}$-тригер, який по сигналу "Пуск" встановлюється в одиничний стан ($H$ = 1) і повертається в нульовий стан ($H$ = 0) по сигналу кінця операції $(\overline{y}_k = \overline{s}_6)$.

Таблиця 3.2 – 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 | $Hz_5$ | 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.




а б в

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

Рисунок 3.2 – Мікропрограмний автомат Мура

## 3.3 Проектування синхронних цифрових автоматів

### 3.3.1 Структурний синтез цифрового автомата

Для 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: $0 \to 0$, $0 \to 1$, $1 \to 0$, $1 \to 1$. Реалізація цих переходів виконується під впливом відповідних сигналів збудження (табл. 3.3).

Таблиця 3.3 – Сигнали збудження тригерів

| Переходи | Сигнали збудження тригерів | | | |
|---|---|---|---|---|
| | J , K | S , R | D | T |
| 0→0 | 0 , * | 0 , * | 0 | 0 |
| 0→1 | 1 , * | 1 , 0 | 1 | 1 |
| 1→0 | * , 1 | 0 , 1 | 0 | 1 |
| 1→1 | * , 0 | * , 0 | 1 | 0 |

У табл. 3.3 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-НЕ", "АБО-НЕ".

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.

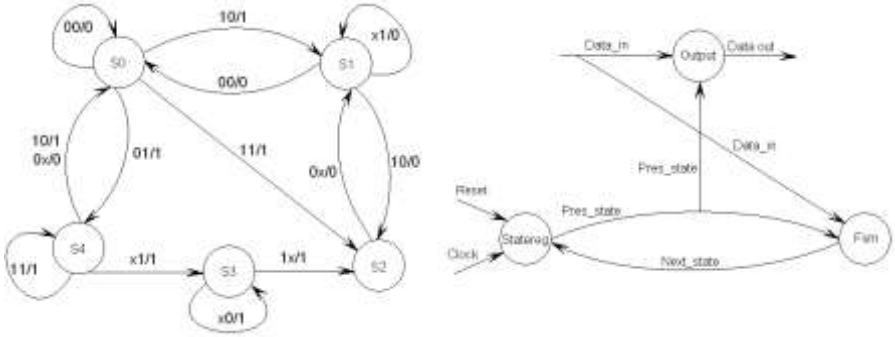
Ілюстрація використання методики проектування схем послідовнісного типу на основі автоматних моделей розглянута у прикладах, наведених у додатку Б.

### 3.3.2 Опис послідовнісних пристроїв автоматними моделями у VHDL – кодах

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

Нижче розглядається приклад проектування послідовнісного пристрою на базі автомата Мілі.

VHDL – код реалізує поведінку кінцевого автомата, заданого графом (рис. 3.3, 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. 3.3, б).




а б

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

Рисунок 3.3 – 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*.

У додатку В приведена роздруківка VHDL-програми "Автомат Мілі з п’ятьма станами".

### 3.4 Методика проектування кільцевих лічильників

Кільцеві лічильники використовуються в системах керування 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. 3.4) 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.

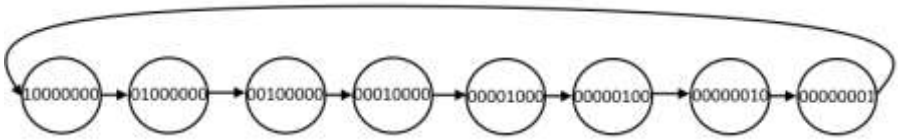



Рисунок 3.4 - Діаграма станів кільцевого лічильника

Початковий стан лічильника встановлюється за допомогою асинхронних установних сигналів. Схема лічильника, відповідна приведеній діаграмі станів, показана на рис. 3.5, а часова діаграма його роботи – на рис. 3.6.

Якщо сигнал з інверсного виходу останнього тригера подати на вхід першого тригера, то утворюється перехресний кільцевий лічильник, званий лічильником Джонсона (рис. 3.7).

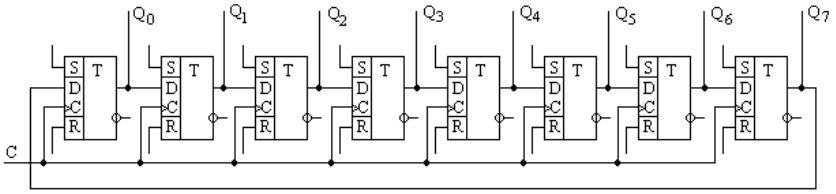



Рисунок 3.5 – Кільцевий лічильник з входами скидання і установки, що забезпечують установку лічильника в початковий стан

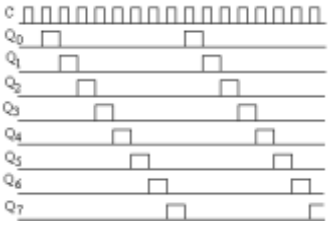



Рисунок 3.6 – Формування імпульсів керування за допомогою кільцевого лічильника

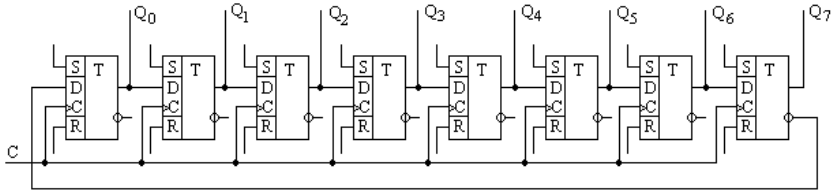



Рисунок 3.7 – Перехресний кільцевий лічильник

Лічильник Джонсона, ініціалізований в нульовому стані, формує повторювальну послідовність: 00000000, 100000000, 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 викликатимуть 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. 3.8, 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. 3.8, б, а схема – на рис. 3.8, в.

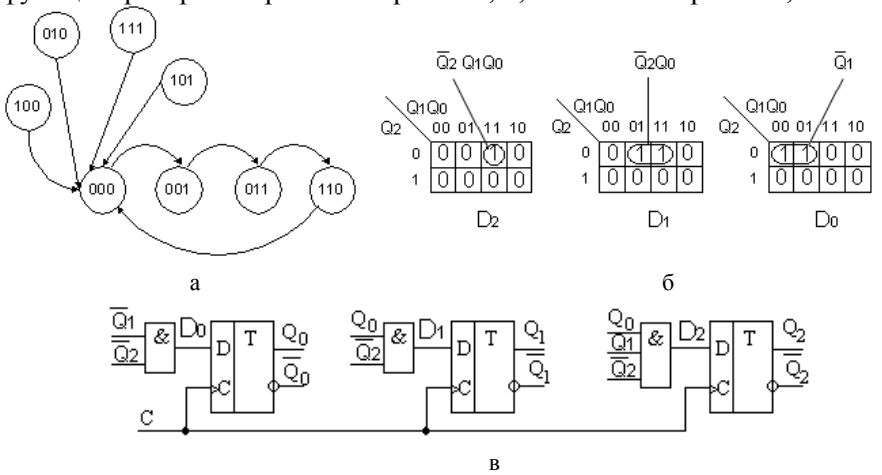



Рисунок 3.8 – Лічильник з установкою початкового стану

# 4 РОЗРОБКА ПРОЕКТІВ ПРИСТРОЇВ НА ПЛІС ІЗ ВИКОРИСТАННЯМ САПР MAX+PLUS II

## 4.1 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 у 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 та 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, що знаходиться за адресою ***c:\program file\maxplusiiv10.2\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*** та ***output***. Далі необхідно привласнити імена всім портам.

Приклад створення графічного файлу з шинною організацією приведений у додатку Д.

**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*** та ***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 вибираних сигнальних ліній. Після введення *ОК* 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 значення сигналу за допомогою кнопки символом "1" або "0" 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.

### 4.2 Аналіз часових параметрів проекту

### 4.2.1 Робота з аналізатором часових параметрів Timing Analyzer

З меню **MAX+plus II** виберіть команду ***Timing Analyzer*** (аналізатор часових параметрів). Запуск ***Timing Analyzer*** приводить до відкриття його вікна і появи на верхній панелі трьох додаткових піктограм: ***Delay Matrix*** (матриця затримок), ***Setup\Hold Matrix*** (матриця часів передустановки і утримання сигналів) і **Registered *Performance*** (швидкодія регістрової логіки). При цьому піктограма ***Delay Matrix*** є активізованою. Після натиснення кнопки ***Start*** аналізатор часових параметрів обчислює затримки розповсюдження сигналів між вхідними і вихідними контактами поточного проекту.

Якщо шляхи розповсюдження сигналу мають різну довжину, то в комірці ***Delay Matrix*** з'являються два значення затримок, відповідні найдовшому і найкоротшому шляхам. Це означає, що в схемі є

змагання сигналів. Коли в початковому проектному файлі джерело і приймач сигналу розділяються інформаційним входом *D*-тригера, затримка обчислюється через ***Clock*** (тактуючий) або ***Preset*** (встановлюючий) входи, а не через *D* (інформаційний) вхід.

При активізації ***Setup\Hold Matrix*** і кнопки ***Start*** визначте мінімально допустимі значення часів передустановки і утримання сигналів для інформаційних входів тригерів.

При активізації ***Registered Performance*** і кнопки ***Start*** визначте затримки в логіці між регістрами, мінімальний період і максимальну частоту тактового сигналу.

### 4.2.2 Робота в редакторі фізичного розміщення Floorplan Editor

Викличте редактор з основного меню **MAX+plus II**. В меню ***Layout*** представлені два варіанти зображення мікросхеми: ***Device View*** (показує всі контакти на корпусі мікросхеми та їх функції) і ***LAB View*** (вид логічних блоків, що показує логічні блоки ***LAB*** і логічні комірки ***LC*** усередині блоків, а також комірки вводу-виводу - ***I/O cell*****).**

Виберіть команду ***LAB View***. Прогляньте на екрані для кожного задіяного елементу вхідні і вихідні зв'язки. Для цього встановіть курсор на відповідну комірку і активізуйте її клацанням клавіші миші.

Використовуючи піктограми  , розташовані зліва, виведіть на екран вхідні і вихідні зв'язки кожного елементу, розташованого в одній зайнятій комірці, а також зв'язки між елементами.

### 4.2.3 Отримання Delay Matrix

Для розрахунку ***Delay Matrix*** того вигляду, який Вам потрібен (бажано з більшою кількістю стовпців і рядків), на екрані редактора ***Floorplan Editor*** послідовно активізуйте вибрані комірки, викличте клацанням правої клавіші миші спливаюче меню, виберіть команду ***Timing Analysis*** та одну з вкладок: ***Sourse*** (джерело сигналу, розташовується у рядку матриці), ***Destination*** (приймач сигналу, розташовується у стовпці матриці), ***Cutoff*** (вирізати з матриці). Потім запустіть ***Timing Analyzer***.

# 5 РОЗРОБКА ПРОЕКТІВ ПРИСТРОЇВ НА ПЛІС ІЗ ВИКОРИСТАННЯМ САПР QUARTUS II

## 5.1 Короткі відомості про систему автоматизованого проектування цифрових пристроїв - Quartus II

### 5.1.1 Класифікація файлів проекту в Quartuse II

Файли в проекті бувають логічними і допоміжними. Логічні файли описують поведінку або структуру окремих модулів проектованого пристрою. Допоміжні файли найчастіше програмою створюються автоматично (наприклад, файли звітів з розширенням ***.rpt***, символьні файли з розширенням ***.bsf***).

Логічні файли можуть бути файлами з текстовим описом на якій-небудь мові опису апаратури (з розширеннями ***.vhd, v., .tdf***), файлами з графічним описом схем (***.bdf***), файли з представленням у вигляді кінцевого автомата.

### 5.1.2 Використання текстового редактора Quartuse II для формування вихідного файлу

Виберіть ***File\New\VHDL File***. Відкриється вікно текстового редактора. Виберіть ***File/Save as***. У вікні введіть назву і розширення файлу ***.vhd***. Встановіть галочку ***Add File to current progect*** (Додати файл в поточний проект). Збережіть файл.

### 5.1.3 Використання графічного редактора

Для створення файлу, що містить принципову схему пристрою, виберіть меню ***File \ New ....*** У діалоговому вікні на вкладці ***Design Files*** відзначте тип файлу ***Block Diagram/Schematic File*** і натисніть кнопку ***OK.***

У вікні графічного редактора створіть схему. Для введення елемента схеми слід «клацнути» по ***Symbol Tool***. В результаті відкриється вікно з бібліотеками елементів. Наприклад, для введення логічного елемента «І» слід вибрати бібліотеку ***primitives \ logic***. Після розміщення компонентів на схемі слід розмістити вхідні (input) і вихідні (output) порти, які перебувають в папці ***primitives\pin.***

Збережіть графічний файл з розширенням ***.bdf***, вибравши меню ***File\Save as ...***. Якщо ім'я файлу збігається з ім'ям проекту, то даний файл буде файлом вищої ієрархії.

### 5.1.4 Використання планувальника чіпів *Chip Planner*

Для візуального контролю використовуваних ресурсів в Quartus II використовується ***Chip Planner***, що показує розташування і використання елементарних блоків в загальній архітектурі ПЛІС. Запуск Chip Planer: ***Tools\Chip Planer***. При цьому формується карта ресурсів ПЛІС, на якій показані логічні елементи ***LE*** (Logic Elements), об'єднані в логічні блоки ***LAB***, апаратні помножувачі (***DSP block***), комірки пам'яті, буфери введення-виведення, генератори ***PLL***. Світлим кольором зафарбовані невикористовувані блоки, темним - максимально завантажені. При збільшенні масштабу (Лупа на панелі або ctrl + коліщатко мишки) доступна деталізація на рівні ***LE***.

Більшу деталізацію можна отримати, якщо на панелі ***Layers Settings*** змінити ***Basic*** на ***Detailed***. У цьому режимі при наближенні видно локальні і глобальні ланцюга сполук, глобальні лінії керування та керуючі сигнали ***LAB***.

Якщо в ***LE*** виділити ***LUT*** (***Look-Up Table***, таблиця істинності логічної функції, релизована у вигляді комбінаційної схеми) або тригер, то на панелі ***Node Properties*** можна побачити схему ***LE*** і опис властивостей і режимів роботи. Подвійний клік миші по ***LUT*** або про триггеру запустить в новому вікні інструмент ***Resource Property Editor***, в якому можна досліджувати з'єднання всередині логічного елемента. Синім виділяються використовувані ланцюги.

Компілятор самостійно призначає контакти ПЛІС для входів і виходів схеми. Щоб вручну призначити номери контактів, виберіть пункт меню ***Assignments\Pin Planner***.

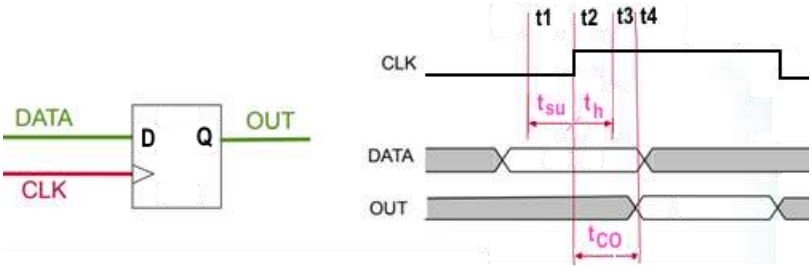
Відкривається вікно з розташованою внизу таблицею зі списком входів і виходів схеми. У порожню колонку ***Location*** впишіть назви виводів, які бажано використовувати в проекті (наприклад, N26, AE23).

### 5.1.5 Підключення невикористовуваних ніжок мікросхеми

За замовчуванням всі невикористовувані контакти мікросхеми підключаються до земляної шини. Якщо ж на платі з мікросхемою, наприклад, помилково невикористаний контакт мікросхеми підключений до живлення плати, то станеться коротке замикання, тобто виникне аварійна ситуація. Для її виключення виберіть пункт меню ***Assignments\Device*** .... Натисніть на кнопку ***Device and Pin Options*** .... З'явиться ще одне вікно, в якому в списку ліворуч виберіть закладку ***Unused Pins***. У списку ***Reserve all unused pins*** виберіть опцію ***As input tri-stated***. В цьому випадку невикористовувані виводи будуть мати високий опір. Це убезпечить мікросхему і плату.

### 5.1.6 Визначення часових параметрів проекту

Ілюстрація методики підрахунку часових параметрів показана на рис. 5.1.




а б

а – умовне позначення D-тригера;
б – часові діаграми його роботи

Рисунок 5.1 - Ілюстрація методики підрахунку часових параметрів

Основні часові параметри проекту:

$t_{SU}$ = $t2$ – $t1$ – мінімальний інтервал часу, протягом якого дані повинні не змінюватися, відраховується від моменту $t1$ до фронту синхроімпульсу (індекс *SU* означає установку - Setup);

$t_h = t3 - t2$ - мінімальний інтервал часу, протягом якого дані не повинні змінюватися, відраховується від фронту синхроимпульсу до моменту $t2$ (індекс $h$ означає утримування – Hold);

$t_{CO} = t4 - t2$ – затримка появи сигналу на виході тригера від моменту появи фронту синхроимпульсу.

$t_{pd}$ – затримка появи сигналу на виході по відношенню до входу (***pin to pin delay*** або ***input/output poin-to point delay***).

Сума параметрів $t_{SU}$ та $t_h$ визначає часове вікно навколо фронту тактового сигналу, протягом якого вхідні дані повинні бути стабільними.

Недотримання вимог до часових параметрів може викликати метастабільний (нестійкий) стан тригера, коли вихід тригера знаходиться в проміжному стані з напругою приблизно відповідною половині напруги між «0» та «1». Час перебування тригера у метастабільному стані є ймовірнісною величиною і залежить від технологічних параметрів елементної бази, температури тощо.

Стан сам по собі невизначений породжує невідомий результат в наступних елементах. Більш того, у багатьох випадках вихідний сигнал одного тригера подається відразу на кілька інших елементів, які через неідеальність технологічної бази можуть мати трохи різні пороги спрацьовування. А значить, один і той же сигнал може трактуватися як «1» в одному місці і як «0» в іншому. Це може привести до розсинхронізації роботи різних частин пристрою і загального збою. Збої в роботі пристрою, викликані метастабільністю, діагностуються вкрай складно.

Швидкодія роботи проектованого пристрою оцінюється за допомогою максимальної частоти синхронізації $\boldsymbol{f_{max}}$, яка, в свою чергу, залежить від максимальної затримки сигналу на будь-якому з шляхів між двома довільно взятими регістрами, на які надходить один і той же тактовий сигнал. При роботі часового аналізатора ***Classic Timing Analyzer*** розраховуються с розкидом на найгірший випадок параметри: $t_{SU}$, $t_{CO}$, $t_{pd}$, $t_h$.

Мінімальний період подачі синхросигналів

$$t_{min} = \frac{1}{f_{max}} = t_{CO} + t_{pd} + t_{SU}(+t_s),$$

де $t_s$ – короткочасна часова нестабільність періоду тактової частоти.

Завданням фізичного проектування є створення топології, при розробці якої оптимізуються часові параметри пристрою. Для оцінки значень часових параметрів використовується параметр слек (***slack***).

Термін slack позначає запас того чи іншого часового параметра. Позитивний ***slack*** означає, що запас для роботи на потрібній частоті є, негативний же говорить про те, що запас вибраний і його не вистачає для роботи на потрібній частоті. Залежно від того, що є джерелом негативного слека, його можна поліпшити зміною налаштувань синтезу і розводки, зміною часових обмежень.

### 5.1.7 Вимоги до ланцюгів розповсюдження синхроімпульсів

При проектуванні синхронних пристроїв бажано зменшувати затримку розповсюдження синхроімпульсів по ланцюгам тактування. Тому доцільно використовувати сигнальні лінії глобального розповсюдження сигналів. Якщо ж у **ланцюгу розповсюдження синхроімпульсів** використовується комбінаційна схема, то компілятор **Quartus II** може сприйняти синхросигнал як звичайний сигнальний, який буде формуватись матрицями з'єднань. Така ситуація може привести до появи перегонів сигналів.

Наприклад, в структуру ПЛІС MAX7000 входять чотири спеціалізовані входи. Ці входи можуть бути використані як входи загального призначення для обробки "швидких" сигналів. Через ці входи на кожну макрокомірку можуть бути подані глобальні керуючі сигнали (синхронізація, скидання, перехід в третій стан).

При цьому для кожного регістра може бути обраний один з трьох способів тактування:

- тактування глобальним синхросигналом - найшвидший спосіб;
- тактування глобальним сигналом із застосуванням локального сигналу дозволу тактування;
- тактування сигналом від локальної програмованої матриці.

У сімействі MAX7000 доступні два глобальних тактових сигнали з виводами GCLK1 і GCLK2.

Сучасні ПЛІС сімейства Cyclone III мають вбудовані схеми фазового автопідстроювання частоти (ФАПЧ, англ. Phase-Locked Loops або скорочено PLL), які використовуються для гнучкого керування синхроімпульсами на системному рівні. Пристрої

Cyclone III побудовані так, що вони мають на кристалі до 4 блоків PLL і до 10 системних ланцюгів синхрочастот, для того, щоб задовольнити вимогам системи, що проектується користувачем. Вони можуть використовуватися як для формування сигналів синхронізації швидкодіючих диференціальних інтерфейсів введення-виведення, так і для тактування загального призначення.

### 5.1.8 Використання мови VHDL для опису цифрових автоматів

Робота цифрового автомата може бути описана програмно мовою опису цифрової апаратури, наприклад, VHDL.

В цьому випадку при компіляції пакет Quartus II формує граф станів автомата, який можна побачити, як і таблицю станів, за допомогою команд ***Tools\Netlist Viewers\State MachineViewer***.

Результатом синтезу цифрового автомата є його функціонально-логічна схема. Для її виклику набираються команди: ***Tools\Netlist Viewers\Technology Map Viewer***. При подвійному натисканні лівою клавішею по блоку розкривається його логічна схема. При подальшому натисканні правою клавішею і виборі ***Properties*** у вспливаючому меню можна побачити таблицю істинності роботи блоку і відповідну цій таблиці карту Карно.

### 5.1.9 Часові аналізатори в Quartus II

Часовий аналізатор є інструментом, який дозволяє знаходити в синтаксично правильному проекті місця, які будуть працювати зі збоями або взагалі не будуть працювати у зв’язку з порушенням часових співвідношень між логічними елементами.

В системі ***Quartus II*** працює два часових аналізатора: ***Classic Timing Analyzer*** та ***Time Quest Timing Analyzer***. ***Classic Timing Analyzer*** простий в роботі, але з появою більш складних проектів, наприклад, з високочастотними інтерфейсами, якість часового аналізу і синтезу погіршилася. Тому була розроблена технологія ***Timing Driven Synthesis*** для ПЛІС, в основі якої проектувальником задаються часові обмеження, які впливають на результати синтезу. Задані часові обмеження виконуються при синтезі у першу чергу. Тобто синтез виконується не на шкоду часовим обмеженням.

Для новых сімейств ПЛІС (Aria II, Stratix IV/V, Cyclone II) використання ***Classic Timing Analyzer*** в системі ***Quartus II*** взагалі неможливе.

Головна відмінність від ***Classic Timing Analyzer*** аналізатора ***Time Quest Timing Analyzer*** полягає в тому, що ***Time Quest Analyzer*** перевіряє тільки ті обмеження, які задає проектувальник. Наприклад, перед компіляцією проектувальник може задати деякі часові параметри, які він хоче отримати в пристрої після проектування. До них відносяться: затримка поширення сигналу $t_{pd}$, час спрацьовування від тактового сигналу $t_{CO}$, час встановлення $t_{SU}$, внутрішня і системна частоти.

***TimeQuest Analyzer*** дозволяє працювати в двох режимах: командному і графічному. У командному режимі всі команди часового аналізу і завдання часових обмежень вводяться з консолі. При цьому створюється файл з розширенням ***.sdc***. У графічному режимі використовується графічний інтерфейс (GUI).

***Time Quest Timing Analyzer*** запускається автоматично на заключний етапі компіляції. Якщо часові обмеження проектувальником не задані, то видають критичні попередження типу:

а) *Critical Warning: Synopsys Design Constraints File file not found: 'ім'я файлу.sdc'. A Synopsys Design Constraints File is required by the Time Quest Timing Analyzer to get proper timing constraints. Without it, the Compiler will not properly optimize the design;*

б) *Critical Warning: Timing requirements not met.*

## 5.2 Процедура розробки проекту в САПР QUARTUS II

### 5.2.1 Створення нового проекту

Для створення нового проекту використовується утиліта ***New Project Wizard*** (***NPW***). Для її виклику клацніть лівою клавішею миші по кнопці ***File*** на панелі інструментів Quartus II. У вікні клацніть по рядку ***New Project Wizard.*** Далі з'явиться вікно з інформацією про кроки, які необхідно виконати при створенні нового проекту. Для переходу до наступного вікна ***NPW*** натисніть кнопку ***Next*** лівою клавішею миші. У вікні **NPW** (першому з п'яти) пропишіть шлях до директорії, в якій буде зберігатися робоча директорія проекту.

Встановіть робочу директорію проекту і введіть назву головного модуля проекту. Проект і його головний модуль повинні мати

однакові назви. Натисніть ***Next***. Якщо директорія ще не створена, то програма покаже спливаюче повідомлення, що пропонує створити вказану директорію. Виберіть ***Yes***.

У наступному вікні ***NPW*** (другому з п’яти) можна додати файли в проект, якщо вони є. Якщо їх немає, клікніть ***Next.***

У наступному вікні ***NPW*** (третьому з п’яти) визначається сімейство кристала, на основі котрого буде реалізований проект. Конкретний чіп сімейства можна вибрати самостійно або надати вибір програмі.

У вікні ***NPW*** четвертому з п’яти вибираються інструменти сторонніх розробників, які будуть використовуватися. У Вашому проекті зазначені інструменти використовуватися не будуть, тому слід натиснути ***Next.***

У вікні п'ятому ***NPW*** з п’яти дається інформація про встановлені налаштування. Слід натиснути ***Next.***

### 5.2.2 Компіляція проекту

Запуск процесу компіляції**: Processing\Start Compilation**.
***Компіляція проекту*** включає наступні етапи:

- аналіз і синтез;
- розміщення і трасування;
- часовий аналіз;
- аналіз споживаної потужності.

Вихідними даними при компіляції проекту можуть бути файли з розширеннями ***.vhd***, ***.v***, ***.tdf***, ***.bdf***, а також файли бібліотек (***.lmf***).

Головний модуль проекту може бути файлом з текстовим описом. В цьому випадку при компіляції конструкції мови представляються у вигляді апаратної реалізації, тобто у вигляді функціональних перетворювачів ***LUT***: логічних елементів, тригерів, помножувачів, блоків пам’яті.

На ***етапі аналізу і синтезу*** створюється база проекту, тобто сукупність файлів опису проекту. Вихідними даними при цьому є файли звітів (***.rpt***, ***.htm***), файли бази даних (***.rdb***), файли з розширенням ***.cdb,*** файли з установками (з розширенням ***.qsf***).

Після закінчення процесу компіляції відкривається вкладка ***Compilation Report - firstproject***, яка містить звіт про результати виконання аналізу і синтезу.

У розділі ***Flow Summary*** у рядку ***Flow Status*** виводиться узагальнений результат компіляції: ***Successful*** - успішно, або ***Flow Failed*** - компіляція не виконана. У розділі ***Flow Summary*** також наводяться параметри процесу компіляції, час її виконання і детальні результати синтезу.

У вікні ***Flow Messages*** виведені повідомлення про помилки, які розділені на чотири групи. Зеленим кольором позначаються інформаційні повідомлення, які виводяться під час будь-якої компіляції. Синім кольором позначаються попередження ***Warnings*** та ***Critical Warnings*** про проблемні ситуації при компіляції, що дозволяють продовжувати процес компіляції. Критичні попередження ***Critical Warnings*** викликаються помилками, які можуть серйозно вплинути на роботу проектованого пристрою. Вони усуваються автоматично, але при цьому робота пристрою може сильно відрізнятися від передбачуваної. Тому критичні попередження вимагають додаткового аналізу і, по можливості, усунення їх причини. Червоним кольором виділяються повідомлення про помилки ***Errors***, що перешкоджають успішному виконанню компіляції.

На ***етапі розміщення і трасування*** виконується оптимізаційна процедура розміщення функціональних перетворювачів на кристалі на основі критерію оптимізації: мінімізації часів поширення сигналів.

Вихідними файлами для виконання етапу розміщення і трасування є файли з розширенням ***.cdb***, ***.qsf***. Вихідними файлами є файли звітів про компіляцію (***.rpt***, ***.htm***) і оновлена база даних (***.rdb***).

Після виконання етапу розміщення і трасування у користувача є можливість завдання власних призначень. У разі завдання власних призначень повторно виконується оптимізаційна процедура. Результати остаточних розміщення і трасування можна проаналізувати за допомогою планувальника кристалів ***Chip Planner***.

Ці результати можна вручну скоригувати за допомогою редактора топології ***Resource Property Editor***.

На ***етапі часового аналізу*** виконується перевірка на відповідність вимогам швидкодії.

На ***етапі аналізу споживаної потужності*** виконується перевірка на відповідність вимогам по споживаної потужності.

Система Quartus II включає засіб аналізу енергоспоживання ***PowerGauge***, який використовує файли, створені в процесі моделювання для того, щоб зв'язати оцінку споживання енергії з заданими параметрами пристрою. Аналізатор енергоспоживання дозволяє проектувальникам встановити і оптимізувати споживання енергії ранній стадії процесу розробки.

### 5.2.3 Верифікація проекту

При верифікації проекту використовується сигнальний редактор ***Waveform Editor***.

Спочатку необхідно створити файл часових діаграм за допомогою команд ***File\New\Other Files\Vector Waveform File***. Створений файл необхідно зберегти і включити в проект. Для формування часових діаграм необхідно в поле ***Name*** робочого вікна сигнального редактора натиснути на праву кнопку миші і в контекстному меню вибрати пункт ***Insert Node or Bus…***

У діалоговому вікні слід натиснути на кнопку ***Node Finder…*** У наступному діалоговому вікні слід натиснути на ***List → OK***. Якщо після виконаних дій в поле ***Nodes Found*** не з'являться рядки з іменами контактів пристрою, то в поле ***Filter*** необхідно встановити ***Pins: all*** та натиснути ***List*** повторно.

Панель інструментів дозволяє задати значення сигналів: **лог.0**, **лог.1**, невизначений стан ***X*** вузла, третій стан ***Z***, невизначений стан ***W*** шини, значення «Не ініціалізовано» ***U***. Піктограма ***R*** дозволяє сформувати випадкові значення (**лог.0** або **лог.1**) на кожному кроці сітки. Клацання по піктограмі ***INV*** інвертує сигнал на виділеному часовому інтервалі. Піктограма у вигляді годинника дозволяє ввести синхроімпульс у вигляді меандра. Піктограма ***C*** дозволяє сформувати циклічну послідовність із **лог.0** та **лог.1** із заданим часовим інтервалом за допомогою параметру ***Increment by***. Піктограма «**?**»дозволяє порівнювати часові діаграми.

При завданні на виділеному часовому інтервалі значення сигналів на обраній шині в поле ***Radix*** задається система числення: двійкова - ***Binary***, вісімкова - ***Octal***, десяткова з числами зі знаком-

***Signed Decimal***, десяткова з числами без знаку - ***Unsigned Decimal***, шістнадцяткова - ***Hexadecimal***.

Для запуску симуляції в меню ***Processing*** вибирається пункт ***Start Simulation***. Результати зберігаються у файлі з розширенням ***.vwf.***

Для оцінки швидкодії проектованого пристрою використовуються маркери, які поміщаються на початку і кінці вимірюваного часового інтервалу. У полі ***Interval*** фіксується значення інтервалу.

### 5.2.4 Використання RTL-переглядача

***RTL***-переглядач використовується для полегшення розробки коду ***VHDL*** для вже розроблених схем. Перегляд зображення дозволяє спростити знаходження втрачених елементів, невірних з'єднань і інших типових помилок на ранньому етапі розробки проекту.

Для відображення схеми на рівні регістрових передач клікніть на ***Tools\Viewers\RTL Viewer.***

Інструмент «***Лупа***» збільшує зображення виділеної частини схеми. Для перегляду вмісту будь-якої підсхеми необхідно двічі клікнути на відповідному їй блоці.

### 5.2.5 Функціональне моделювання

Функціональне моделювання (***Functional Simulation***) перевіряє правильність логічної роботи синтезованих схем. При цьому часові затримки в елементах не враховуються.

За замовчуванням Quartus II цей режим не вибирає. Для його запуску введіть ***Assignments\Setting\Simulator Setting\Functional***. Виберіть ***Run simulation until all vector stimuli are used.***

# 6 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. Графічна частина може бути замінена презентацією. 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, затвердженому 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.

# РЕКОМЕНДОВАНА ЛІТЕРАТУРА

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

2 **Стешенко, В. Б.** EDA. Практика радиоэлектронного проектирования радиоэлектронных устройств [Текст] / В. Б. Стешенко. – М.: Издательство «Нолидж», 2002. – 768 с.

3 **Стешенко, В. Б.** ПЛИС фирмы "Altera": Элементная база, система проектирования и языки описании аппаратуры [Текст] / В. Б. Стешенко. – М.: Издательский дом "Додэка-XXI", 2002. – 576 с.

4 **Бибило, П. Н.** Основы языка VHDL [Текст] / П. Н. Бибило.– М.: СОЛОН-Р, 2002. – 224 с.

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

6 **Угрюмов, Е. П**. Цифровая схемотехнка [Текст] / Е. П. Угрюмов. – СПб.: БХВ-Петербург, 2000. - 528 с.

7 **Комолов, Д. А.** Основы автоматизированного проектирования фирмы Altera MAX+plus II, Quartus II. Краткое описание и самоучитель [Текст] / Д. А. Комолов, З. А. Мяльк, А. А. Зобенко, А. С. Филиппов. – М.: ИП РадиоСофт, 2002. – 352 с.

8 **Уилкинсон, Б.** Основы проектирования цифровых схем [Текст] / Б. Уилкинсон. – М.: Издательский дом «Вильямс», 2004. – 320 с.

9 **Савельев, А. Я.** Прикладная теория цифровых автоматов [Текст] / А. Я. Савельев. – М.: Высш. шк, 1987. - 272 с.

10 **Норенков, И. П**. Основы автоматизированного проектирования [Текст] / И. П. Норенков – М.: Изд-во МГТУ им. Н. Э. Баумана, 2002. – 336 с.

11 **Савельев, А. Я.** Прикладная теория цифровых автоматов [Текст] / А. Я. Савельев. – М.: Высш. школа, 1987. - 272 с.

12 **Сергиенко, А. М.** VHDL для проектирования вычислительных устройств [Текст] / А. М. Сергиенко. – К.: ЧП "Корнейчук", ООО " ТИД "ДС", 2003. – 208 с.

13 **Бабич, М. П.** Комп'ютерна схемотехніка [Текст] / М. П. Бабич, І. А. Жуков. - К.: «МК-Прес», 2004. – 412 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: 0101010101010101; 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 12, якщо $a$ = 1, то модуль лічби лічильника дорівнює 5.

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

Вказівка. X1 і X2 – керуючі сигнали. Якщо X1=1 і X2=0, то лічильник виконує підсумовування; якщо X1=0 і X2=1, то лічильник виконує віднімання. При X1=X2 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 Y0…Y7 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 кожнім 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:

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

**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. Щілинність 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.

**22 Пристрій підсумовування співпадаючих кодів на базі ПЛІС.**

Вказівка. Чотирирозрядні числа знаходяться в регістрах пам'яті. Потактово вони надходять на входи компаратора. Компаратор формується на основі дешифратора і мультиплексора. Якщо коди співпадають, на виході компаратора формується логічна 1. До виходу компаратора під'єднується лічильник з $К_{ліч}$ = 10, який підраховує кількість співпадінь кодів на протязі 10 тактів. Лічильник необхідно синтезувати за методикою, представленою в методичних вказівках.

**23 Пристрій підсумовування кількості незбігів кодів на базі ПЛІС.**

Вказівка. Чотирирозрядні числа знаходяться в регістрах пам'яті. Потактово вони надходять на входи компаратора. Компаратор

формується на основі логічних елементів «виключне АБО» та кон'юнктора. Якщо коди не збігаються, на виході компаратора формується логічна 1. До виходу компаратора під'єднується лічильник з $К_{ліч}$ = 12, який підраховує кількість незбігів кодів на протязі 12 тактів. Лічильник необхідно синтезувати за методикою, представленою в методичних вказівках.

**24 Пристрій зсуву кодів при порівнянні їх з константою на базі ПЛІС.**

Вказівка. Чотирирозрядний код записується зі входів схеми в регістр пам'яті. Потактово він надходить на входи компаратора. Компаратор формується на основі логічних елементів «виключне АБО» та кон'юнкторів. Значення константи: 0110. Якщо представлений код менше константи, то на виході компаратора формується логічний 0, інакше – логічна 1. До виходу компаратора під'єднується вхід керування реверсивного регістра. В реверсивний регістр спочатку записується код, що надійшов на входи схеми. Далі в регістрі цей код зсувається на один розряд праворуч при появі лог. 1 на виході компаратора або ліворуч при появі лог. 0. Реверсивний регістр необхідно синтезувати за методикою, представленою в методичних вказівках.

**25 Пристрій множення на 2 на базі ПЛІС.**

Вказівка. Чотирирозрядний код записується зі входів схеми в регістр пам'яті. Потактово він надходить на входи компаратора. Компаратор формується на основі логічних елементів «виключне АБО» та кон'юнкторів. Значення константи: 1001. Якщо представлений код менше константи, то на виході компаратора формується логічний 0, інакше – логічна 1. До виходу компаратора під'єднується вхід керування регістра зсуву. В регістр зсуву спочатку записується код, що надійшов на входи схеми. Далі в регістрі цей код зсувається на один розряд при появі лог. 1 на виході компаратора або залишається незмінним при появі лог. 0. Регістр необхідно синтезувати за методикою, представленою в методичних вказівках.

**26 Пристрій ділення на 2 на базі ПЛІС.**

Вказівка. Чотирирозрядний код записується зі входів схеми в регістр пам'яті. Потактово він надходить на входи компаратора. Компаратор формується на основі логічних елементів «виключне АБО» та кон'юнкторів. Значення константи: 0101. Якщо представлений код менше константи, то на виході компаратора

формується логічний 0, інакше – логічна 1. До виходу компаратора під'єднується вхід керування регістра зсуву. В регістр зсуву спочатку записується код, що надійшов на входи схеми. Далі в регістрі цей код зсувається на один розряд при появі лог. 0 на виході компаратора або залишається незмінним при появі лог. 1. Регістр необхідно синтезувати за методикою, представленою в методичних вказівках.

**27 Цифровий пристрій контролю витрати матеріалу в бункері на базі ПЛІС.**

Вказівка. Цифровий пристрій керує клапаном подачі матеріалу в бункер. У системі є два датчики, один з яких визначає настання події досягнення матеріалом рівня $L1$, а другий - настання події досягнення матеріалом рівня $L2$. Відсутності події відповідає лог. 0, настанню події - лог. 1. Клапан подачі матеріалу керується сигналом з двома станами: лог. 1 - "ВВІМКНЕНО", лог. 0 - "ВИМКНЕНО". Клапан повинен відкриватися, коли рівень матеріалу в бункері опускається нижче відмітки $L1$, і залишатися відкритим до тих пір, поки рівень матеріалу не підніметься вище за відмітку $L2$, після чого він повинен закритися. Крім того, клапан залишається закритим, поки рівень матеріалу не падає нижче відмітки $L1$. Пристрій повинен виробляти сигнал управління роботою клапана, а також сигнал про помилку. Крім того, система повинна рахувати кількість відчинень клапана за заданий проміжок часу, виражений в кількості періодів синхроімпульсу, що надходить на вхід лічильника. Чотирирозрядний лічильник необхідно синтезувати за методикою, представленою в методичних вказівках.

**28 Пристрій контролю відновлення вхідних сигналів на базі ПЛІС.**

Вказівка. Трирозрядне слово $A_2A_1A_0$ декодується і значення унітарного коду з виходів дешифратора надходить на входи шифратора. При правильній роботі дешифратора і шифратора вхідний код $A_2A_1A_0$ має збігатися з вихідним кодом шифратора $B_2B_1B_0$. При цьому на виході схеми порівняння повинна встановитися лог.1. Потім при наявності лог.1 сигнали з виходів дешифратора в послідовному коді надходять на вхід лічильника з $К_{ліч} = 5$. Результат підсумовування іншою схемою порівняння порівнюється з кодом 100. Якщо результат менше чи дорівнює 100, то на виході схеми порівняння встановлюється лог. 1, якщо ні – то лог. 0. Лічильник необхідно синтезувати за методикою, представленою в методичних вказівках.

**29 Дільник доповняльного коду на 2 на базі ПЛІС.**

Вказівка. На входи схеми надходить чотирирозрядний прямий код, який необхідно перетворити перетворювачем кодів в доповняльний, а потім подати на входи регістра зсуву для виконання операції ділення. Регістр необхідно синтезувати за методикою, представленою в методичних вказівках.

**30 Блок керування для операції додавання на базі ПЛІС.**

Вказівка. Початкові дані:

- тип арифметичної операції – додавання двійкових чисел;
- початковий код подання операндів – доповняльний;
- розрядність операндів - 8;
- код виконання операцій у суматорі – доповняльний модифікований;
- структура операційного блоку – із закріпленими мікрооперацією;
- структура операційного блоку – із закріпленими мікрооперацією;
- тип блоку керування – автомат Мура з пам'яттю на *JK*-тригерах;
- схема логічної ознаки – переповнення розрядної сітки;
- схема логічного порозрядного додавання кодів вхідних операндів *A* і *B*.

**31 Блок керування для операції віднімання на базі ПЛІС.**

Вказівка. Початкові дані:

- тип арифметичної операції – віднімання двійкових чисел;
- початковий код подання операндів – доповняльний;
- розрядність операндів - 8;
- код виконання операцій у суматорі – доповняльний модифікований;
- структура операційного блоку – із закріпленими мікрооперацією;
- структура операційного блоку – із закріпленими мікрооперацією;
- тип блоку керування – автомат Мілі з пам'яттю на *JK*-тригерах;
- схема логічного порозрядного віднімання кодів вхідних операндів *A* і *B*.

**32 Блок керування для операції множення на базі ПЛІС.**

Вказівка. Початкові дані:

- тип арифметичної операції – множення двійкових чисел;
- початковий код подання операндів –прямий;
- розрядність операндів - 8;
- код виконання операції у суматорі – доповняльний;
- структура операційного блоку – із закріпленими мікрооперацією;
- структура операційного блоку – із закріпленими мікрооперацією;
- тип блоку керування – автомат Мілі з пам'яттю на *JK*-тригерах;
- схема логічної ознаки – переповнення розрядної сітки;
- схема логічного порозрядного множення кодів вхідних операндів *A* і *B*.

**33 Пристрій зсуву кодів з керуванням блоком підсумовування співпадаючих кодів на базі ПЛІС.**

Вказівка. Чотирирозрядні числа знаходяться в регістрах пам'яті. Потактово вони надходять на входи компаратора. Компаратор формується на основі дешифратора і мультиплексора. Якщо коди співпадають, на виході компаратора формується лог. 0, якщо ні – лог.1. До виходу компаратора під'єднується чотирирозрядний реверсивний зсувовий регістр, в який початково записується код 0110. Вихід компаратора під'єднується до входу вибору операції регістра. В регістрі при лог. 0 на цьому вході інформація зсувається праворуч, при лог. 1 – ліворуч. .Регістр необхідно синтезувати за методикою, представленою в методичних вказівках.

**34 Пріоритетний шифратор клавіатури «12→4» на базі ПЛІС.**

Вказівка. Сигнал з більшим пріоритетом блокує запити з меншим пріоритетом. На виходах шифратора необхідно утворити результат шифрування у вигляді інверсного позиційного коду. В схемі шифратора необхідно передбачити також вихід, сигнал на якому вказує на надходження вхідного сигналу. До цього виходу під'єднується синхронний лічильник, за допомогою якого можна визначити швидкість роботи людини-оператора.

**35 Пристрій дешифрування адреси операнда на базі ПЛІС.**

Вказівка. Структура дешифратора повинна бути прямокутною. Адреса операнда до дешифрування є чотирирозрядною, двійковою. Формування адреси операнда передбачити за допомогою лічильника Джонсона. Лічильник необхідно синтезувати за методикою, представленою в методичних вказівках.

**36 Двоканальний реверсивний лічильник на базі ПЛІС.**

Вказівка. Початкові дані:

– кількість розрядів лічильника – 12;
– лічильник має групову структуру (три однакові групи по 4 розряди;
– між групами перенесення послідовне;
– в групі початкові дані *D*3-*D*0 завантажуються паралельно при
$\overline{L} = 0$;
– підсумовування імпульсів *U* відбувається за входом ”+1”;
– віднімання імпульсів *U* відбувається за входом ”-1”;
– передбачається наявність виходів ”міжрозрядне перенесення” та”міжрозрядна позика” в групі;
– скидання в початковий стан сигналом лог. 0 на $\overline{R}$ - вході;
– тривалість імпульсів на лічильних входах має бути не меншою 20 нс.

**37 Зсувовий пристрій на базі ПЛІС.**

Вказівка. Восьмирозрядний код записується зі входів схеми в регістр пам’яті. Потактово він надходить на входи компаратора. Компаратор формується на основі логічних елементів «виключне АБО» та кон’юнкторів. Значення константи: 10010011. Якщо представлений код менше константи, то на виході компаратора формується лог. 1, інакше – лог. 0. До виходу компаратора під’єднується вхід керування кільцевого регістра. В регістр спочатку записується код, що надійшов на входи схеми. Далі в регістрі цей код зсувається на один розряд праворуч при появі лог. 1 на виході компаратора або не змінюється при появі лог. 0. Кільцевий регістр необхідно синтезувати за методикою, представленою в методичних вказівках.

**38 Генератор послідовності 9-18-21-5-12-7 на базі ПЛІС.**

Вказівка. Генератор необхідно побудувати на основі лічильника на *JK*-тригерах. Лічильник необхідно синтезувати за методикою, представленою в методичних вказівках.

**39 Схема порівняння з циклічним зсувом.**

Вказівка. Схема порівнює два чотирирозрядних двійкових числа без знаку, що надходять на входи з регістрів пам'яті. У залежності від сигналу керування *z* на виходи схеми подається число, що є більшим (*z*=0) або число, що є меншим (*z*=1), зсунуте циклічно праворуч на два розряди.

**40 Генератор парності з кодовою корекцією.**

Вказівка. Схема підраховує число нулів у восьмирозрядному операнді. Якщо їх кількість парна, то нуль у самому старшому розряді зі всіх розрядів, що мають нулі, замінюється на одиницю. Результат зберігається у регістрі пам'яті.

**41 Охоронний пристрій з сигналом тривоги.**

Вказівка. Пристрій виробляє сигнал тривоги, коли на його входи надходить 8-розрядний код, в якому присутні три поруч розташовані нулі. Він також підраховує число нулів у коді. На вході пристрою необхідно передбачити регістр пам'яті.

# Додаток Б
# 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. Б.1.

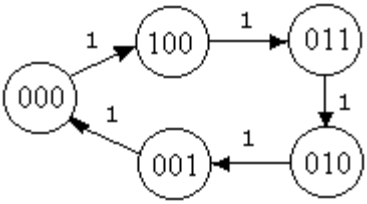



Рисунок Б.1 – Граф зміни станів трирозрядного лічильника

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

Таблиця Б.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 довизначення на надлишкових вхідних наборах (рис. Б.2).

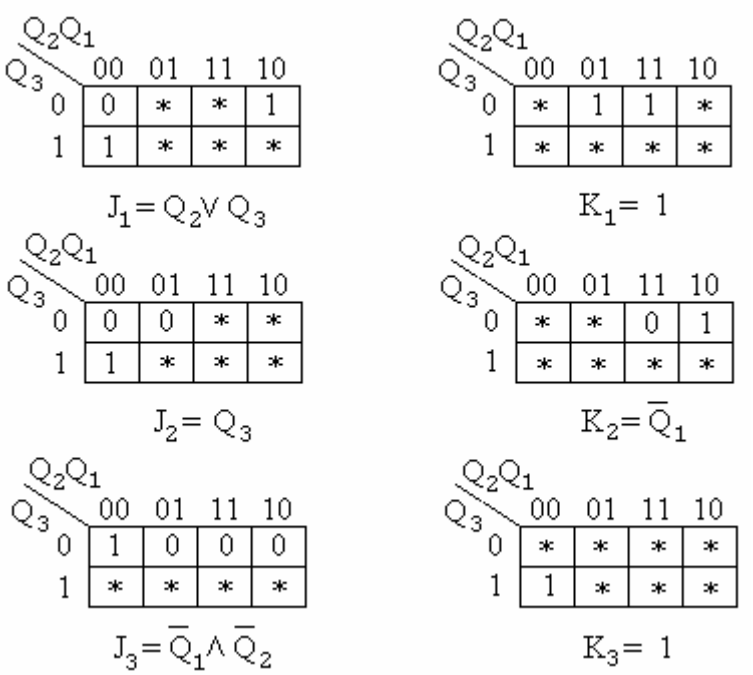



Рисунок Б.2 – Мінімізація функцій лічильника за модулем 5

Логічна схема проектованого лічильника зображена на рисунку Б.3.

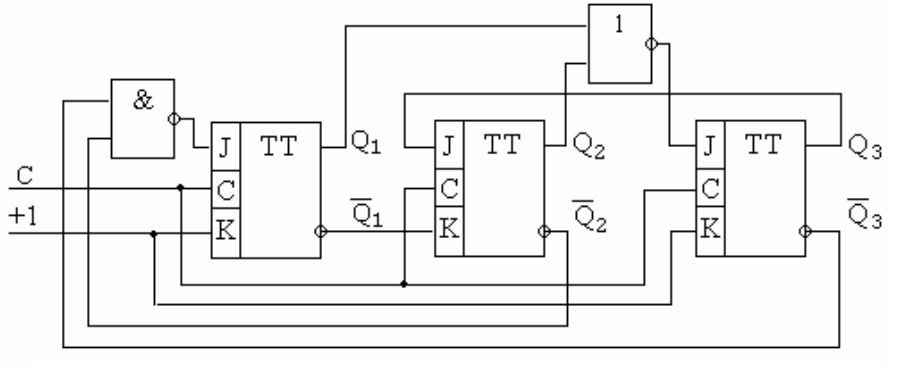
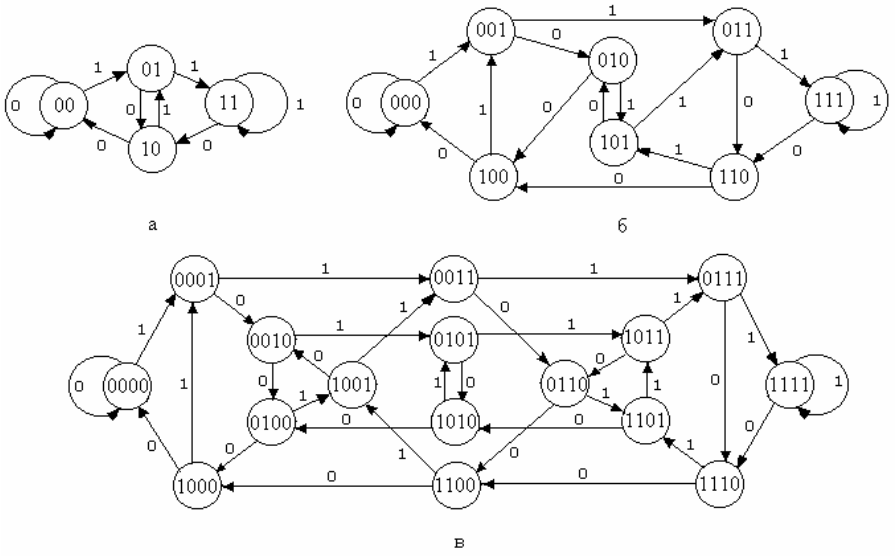



Рисунок Б.3 – Логічна схема лічильника зворотного лічення за модулем 5

**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. Б.4.




а – дворозрядний регістр;
б - трирозрядний регістр;
в - чотирирозрядний регістр

Рисунок Б.4 – Графи станів регістрів зсуву

Щоб здобути десять різних станів, потрібний чотирирозрядний регістр зсуву, граф станів якого зображено на рис. Б.4, в. На графі станів слід вибрати по напряму 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. Б.2).

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

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

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

| Попередній стан | | | | Наступний стан | | | | Сигнали збудження | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Q*4 | *Q*3 | *Q*2 | *Q*1 | *Q*4 | *Q*3 | *Q*2 | *Q*1 | *D*4 | *D*3 | *D*2 | *D*1 |
| 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 |

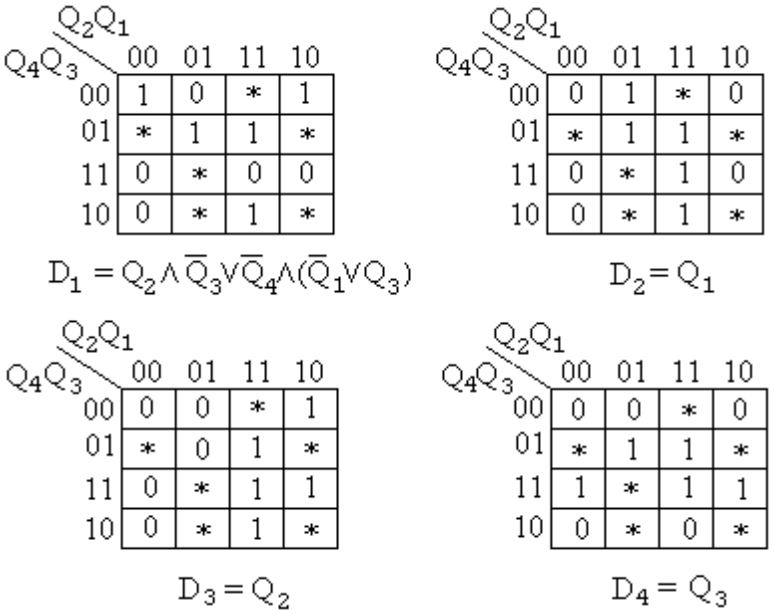



Рисунок Б.5 - Мінімізація функцій лічильника за модулем 10

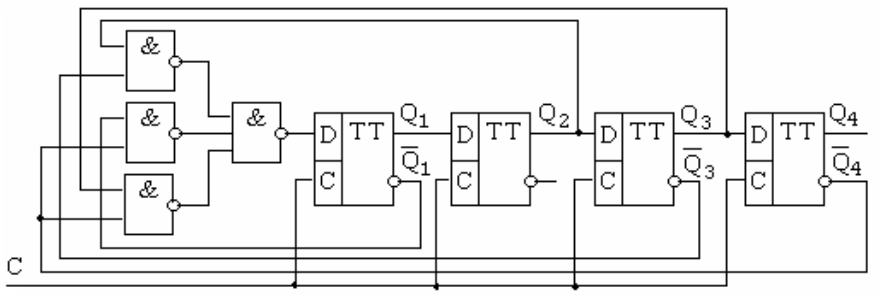



Рисунок Б.6 – Синхронний лічильник за модулем 10 на базі регістра зсуву

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

Функціональна схема генератора двійкової послідовності складається 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 графи станів дво-, три-, і чотирирозрядних регістрів зсуву (рис. Б.4).

Щоб отримати двійкову послідовність, яка вміщує 6 біт, потрібний трирозрядний лічильник. Він може бути реалізований на базі регістра зсуву, граф станів якого зображено на рис. Б.4,б. На графі 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 Б.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. Б.7).

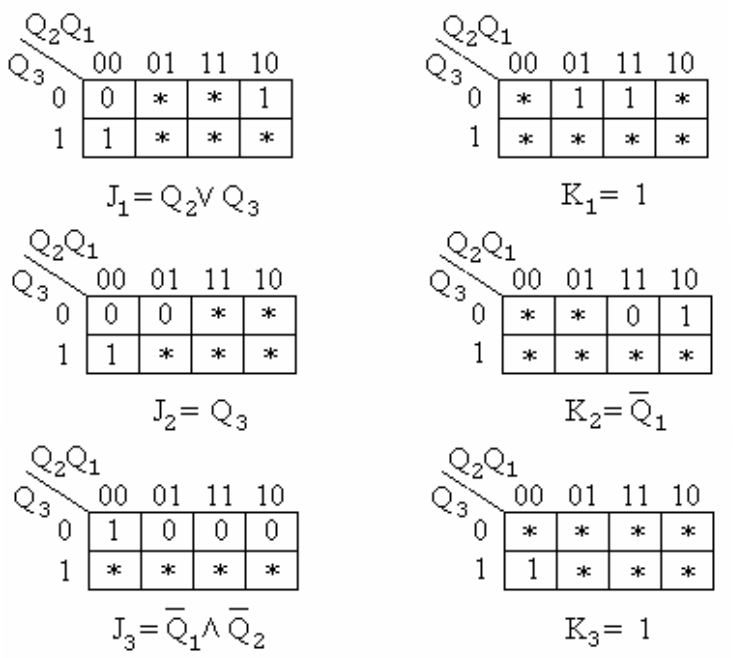



Рисунок Б.7 – Мінімізація функцій генератора двійкової послідовності 0-1-1-1-1-0

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

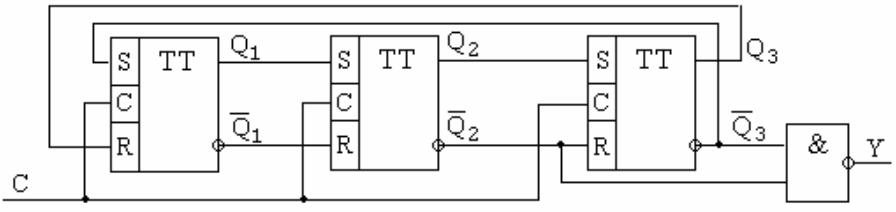



Рисунок Б.8 – Генератор двійкової послідовності 0-1-1-1-1-0

# Додаток В
# 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 регістр
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;
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;
```

# Додаток Д
# Використання шин у графічному файлі

На рис Д.1 показаний приклад використання шинної організації у графічному файлі.

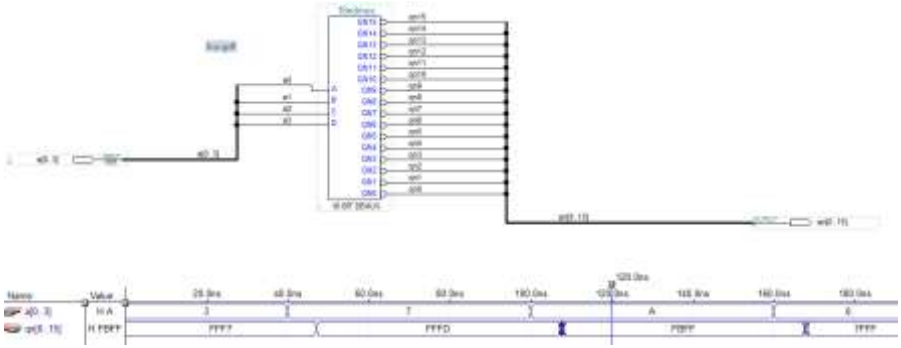


Рисунок Д.1 - Приклад використання шинної організації у графічному файлі