

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування
імені адмірала Макарова

Г. В. ПАВЛОВ, М. В. ПОКРОВСЬКИЙ, І. Л. ВІННИЧЕНКО

ПРОЕКТУВАННЯ ПРИСТРОЮ НА ОСНОВІ ПЛІС

**Методичні вказівки до виконання
курсowego проекту з дисципліни
„Комп’ютеризоване проектування цифрових електронних систем”**

Рекомендовано Методичною радою НУК

Електронне видання
Комбінованого використання на DVD-ROM

Миколаїв • НУК • 2018

УДК 044.438(076)

П 12

Автори: Г. В. Павлов, д-р техн. наук, професор;

М. В. Покровський, канд. техн. наук;

І. Л. Вінниченко, викладач

Рецензент О. К. Жук, канд. техн. наук, доцент

Павлов Г. В.

П 12 Проектування пристрою на основі ПЛІС: методичні вказівки до виконання курсового проекту з дисципліни "Комп'ютеризоване проектування цифрових електронних систем" / Г. В. Павлов, М. В. Покровський, І. Л. Вінниченко. – Миколаїв: НУК, 2018. – 32 с.

Методичні вказівки містять теоретичні відомості, необхідні для розробки пристроїв на базі програмованих логічних інтегральних схем. Наведено приклади розробки апаратної та програмної частини.

Методичні вказівки призначені для допомоги студентам, що навчаються за спеціальністю 151 "Автоматизація та комп'ютерно-інтегровані технології" всіх форм навчання під час виконання курсового проекту з дисципліни "Комп'ютеризоване проектування цифрових електронних систем".

УДК 044.438(076)

Навчальне видання

ПАВЛОВ Геннадій Вікторович
ПОКРОВСЬКИЙ Михайло Володимирович
ВІННИЧЕНКО Ірина Леонідівна

ПРОЕКТУВАННЯ ПРИСТРОЮ НА ОСНОВІ ПЛІС

Методичні вказівки до виконання

курсowego проекту з дисципліни

„Комп'ютеризоване проектування цифрових електронних систем”

ВСТУП

Метою курсового проекту є оволодіння студентами методикою і навичками проектування пристроїв на базі програмованих логічних інтегральних схем (ПЛІС). Курсовий проект виконується студентами поряд з вивченням ними теоретичних питань у курсі лекцій, проходженням лабораторного практикуму та опануванням мов програмування *Verilog* та *VHDL*. Курсовий проект є важливою формою професійної підготовки фахівця зі спеціальності 151 "Автоматизація та комп'ютерно-інтегровані технології".

Курсовий проект має на меті виконання таких завдань: а) систематизація, закріплення та розширення теоретичних знань при розв'язанні конкретних технічних задач; б) розвиток навичок самостійної роботи з технічною літературою; в) оволодіння творчими навичками при самостійному вирішенні конкретних технічних задач; г) підготовка до дипломного проектування. За своєю сутністю курсовий проект є невеликою самостійною інженерною розробкою одного з актуальних питань теорії та практики конструювання систем управління на базі ПЛІС і характеризує опанування теоретичних знань, набуття практичних навичок у цій галузі.

Для виконання проекту необхідно розв'язати 3 задачі: побудувати модель пристрою, розробити програму на мові опису обладнання та апаратного забезпечення, а також розробити випробувальний стенд та перевірити адекватність роботи пристрою. Кінцевим продуктом в курсовому проекті є лістинг програми. Програма повинна бути відтрансльована, а повідомлення транслятора повинно вказувати на відсутність помилок.

1. ЗАВДАННЯ НА КУРСОВЕ ПРОЕКТУВАННЯ

Завдання та конкретні параметри визначаються викладачем. Студенту надається право вибору завдання. Студент може також запропонувати своє завдання з обґрунтуванням доцільності його розробки.

Приклади завдань:

1. Пристрій для множення 32-розрядних дійсних чисел з фіксованою комою.
2. Пристрій для додавання/віднімання 48-розрядних дійсних чисел з плаваючою комою (40 розрядів – мантиса, 8 розрядів – порядок).
3. Контролер послідовного порту (інтерфейс RS-232).
4. Пристрій для ділення 32-розрядних дійсних чисел з фіксованою комою.
5. Контролер миші (інтерфейс PS/2)
6. Пристрій для множення 48-розрядних дійсних чисел з плаваючою комою (40 розрядів – мантиса, 8 розрядів – порядок).
7. Контролер паралельного порту.
8. Пристрій для додавання/віднімання 32-розрядних дійсних чисел з фіксованою комою.
9. Ядро 16-розрядного RISC-процесора.
10. Контролер VGA-монітора з використанням SDRAM в якості відеопам'яті.
11. Пристрій для ділення 16-розрядних дійсних чисел з фіксованою комою.
12. Арифметико-логічний пристрій для 16-розрядних знакових цілих чисел, що виконує операції додавання, віднімання, множення, зсуву та побітові логічні операції AND, OR, XOR та NOT.
13. Керований генератор синусоїдальних коливань на базі широтно-імпульсної модуляції.
14. Пристрій для множення 24-розрядних дійсних чисел з фіксованою комою.
15. Контролер клавіатури (інтерфейс PS/2).

2. ВИМОГИ ДО ОФОРМЛЕННЯ КУРСОВОГО ПРОЕКТУ

Технічна документація курсового проекту розглядається як різновид звіту про виконану інженерно-конструкторську або науково-дослідну роботу. Матеріали курсового проекту повинні відповідати діючим стандартам у сфері оформлення звітів такого виду.

Технічна документація до курсового проекту включає розрахунково-пояснювальну записку (РПЗ) і графічний матеріал. РПЗ до курсового проекту оформлюється на одній стороні аркушів формату А4. Слід використовувати шрифт *Times New Roman* розміром 14 з міжрядковим інтервалом 1,15-1,5 в межах всього текстового документа. Таблиці та рисунки обов'язково повинні мати назву, номер і посилання в тексті РПЗ.

РПЗ повинна містити:

1. Титульний аркуш;
2. Завдання до курсового проекту;
3. Зміст;
4. Перелік умовних позначень символів, одиниць і термінів;
5. Вступ;
6. Опис основних функцій пристрою;
7. Алгоритми роботи пристрою, що проектується;
8. Моделі пристрою;
9. Розроблене програмне забезпечення;
- 10.Список використаних джерел;
- 11.Додатки.

Графічний матеріал курсового проекту містить такі обов'язкові частини:

1. Структурну схему розробленого пристрою;
2. Результати тестування розробленого пристрою.

3. АНАЛІЗ ПРИСТРОЮ, ВИБІР ПЛАТФОРМИ ТА ТЕСТУВАННЯ ПРИСТРОЮ

На етапі опису основних функцій системи управління студент виконує такі дії: визначає функціональний склад модулів пристрою, що проектується; обґрунтовує технічні вимоги до вказаних модулів і встановлює необхідні зв'язки між ними. Виходячи з отриманих вимог до пристрою, проводиться вибір ПЛІС, яка задовольняє цим вимогам. В проекті розробляється цифровий або цифро-аналоговий пристрій на базі 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*, мікросхеми *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 програмується зміною рівня електричного поля 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'ятовуючих пристроїв.

Отримання конкретного проекту на базі *FPGA*, як і на основі інших

ПЛІС, реалізується дією на програмовані з'єднання, внаслідок чого забезпечується замкнений стан одних ділянок і розімкнений стан 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*.

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

Етап 1. Вибір елементної бази і типу системи автоматизованого проектування (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 ПЛІС, а також на основі оцінки можливостей САПР, що підтримує проектування на основі вибраної ПЛІС. У випадку, коли вибір САПР визначений наявним технічним доробком і досвідом проектувальника, уточнюється тільки сімейство ПЛІС. Вибір сімейства повинен задовольняти вимогам ТЗ, наприклад, відповідати певним інтерфейсним стандартам, об'єму вбудованої пам'яті, швидкісним параметрам.

Етап 2. Формалізований опис проектного пристрою. Технічне завдання є сумішшю словесного та технічного опису. Тому необхідно виконати його формалізацію. Формалізований опис представляється у вигляді сукупності основних блоків пристрою (або алгоритму) і їх взаємозв'язків. При цьому реалізується декомпозиція об'єкту проектування з формуванням узагальненої структурної схеми. На цьому етапі САПР використовуються порівняно рідко. Проте сучасні САПР типу САПР

Quartus фірми *Altera* виконують декомпозицію проекту за допомогою спеціальних блокових редакторів. При виконанні курсового проекту декомпозиція виконується проектувальником при наявності необхідності в цьому. Існують безкоштовні версії САПР: *MAX+Plus II Baseline* фірми *Altera*, *WebPack* фірми *Xilinx*, *Actel DeskTOP* фірми *Actel*, які можуть використовуватися в проекті. Зокрема, особливості використання САПР *MAX+Plus II* приведено в додатку А.

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 можна також реалізувати із 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 та ефективністю текстового опису на мові високого рівня. Можна використовувати комбінований підхід, при якому окремі блоки описуються на мові високого рівня, а їх об'єднання виконується в графічному редакторі.

При розробці цифрових пристроїв іноді буває доцільним розбиття їх на операційний блок (ОБ), що здійснює перетворення даних, і блок керування (БК), що забезпечує виконання операційним блоком заданої послідовності операцій. При описі структури операційного блоку зручно користуватися описами типових функціональних вузлів (реєстрів, буферних схем, багатофункціональних логічних блоків), що присутні в бібліотеці САПР. Опис роботи БК зручно виконувати як в графічній, так і в текстовій формах.

Етап 4. Компіляція проекту. Синтез проектованого пристрою для його фізичної реалізації в ПЛІС виконується поетапно декількома компіляторами. На етапі логічного синтезу виробляється представлення проектованого пристрою на вентиляльному рівні, тобто у вигляді логічної схеми пристрою з урахуванням базису реалізації. На цьому етапі виконується й 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 використанні САПР фірм-виробників ПЛІС є завантажувальний файл, що дозволяє виконувати конфігурацію обраної ПЛІС. Створюється також файл звіту з інформацією про процес компіляції та 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:

1. Мінімальних і максимальних затримок між джерелами (вхідними сигналами) і приймачами (вихідними сигналами), інформація про які відображується у вигляді матриці затримок;
2. Максимально допустимої частоти синхронізації елементів пам'яті, використаних в проекті;
3. Часів передустановки й 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.

4. РОЗРОБКА МОДЕЛЕЙ ПРИСТРОЮ

При описі широкого класу цифрових систем доцільним є їх представлення у вигляді сукупності операційного блоку (ОБ) і блоку керування (БК). ОБ здійснює перетворення даних (наприклад, виконує операції множення, додавання), а БК забезпечує виконання операційним блоком заданої у часі послідовності операцій, для чого БК передає на входи ОБ послідовність керуючих сигналів, формування яких в БК залежить від зовнішніх сигналів і результатів операцій в ОБ. При описі алгоритму роботи БК зручно використовувати моделі структурних автоматів різних типів.

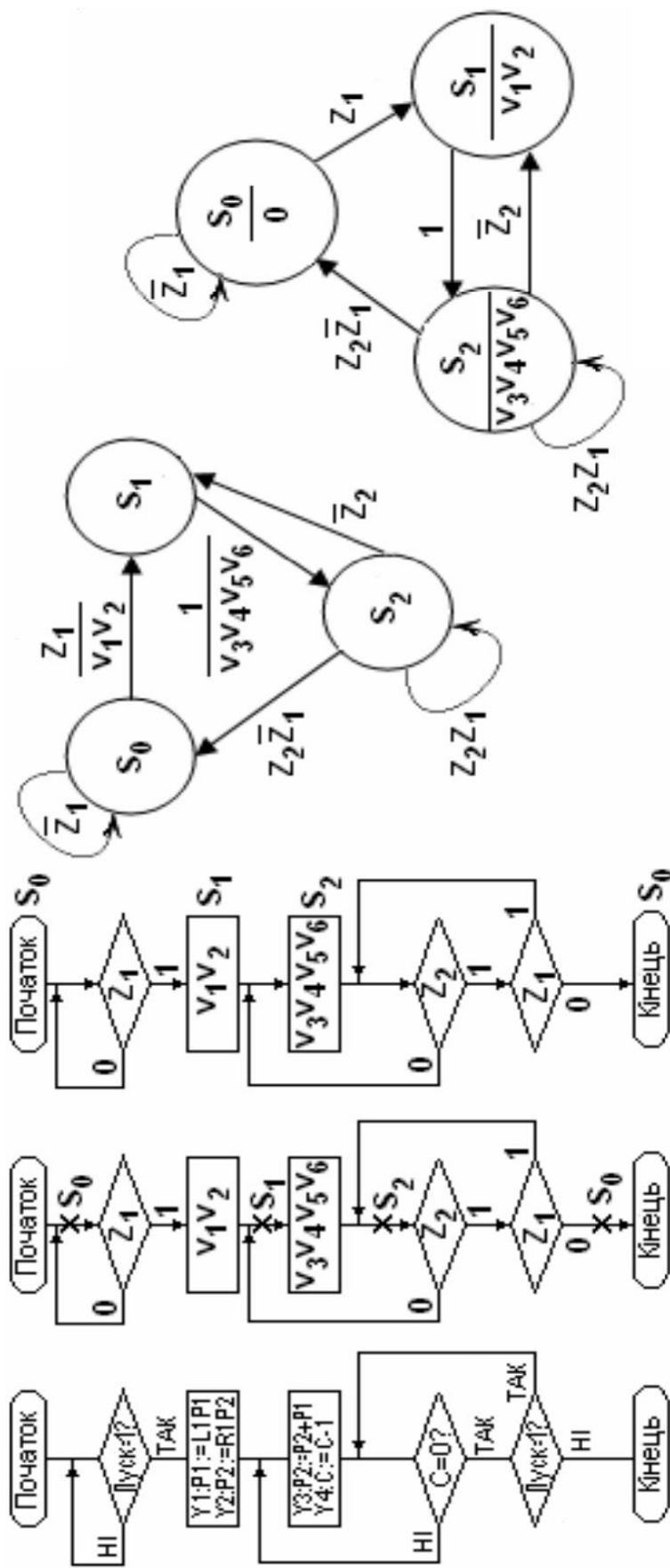
Мікропрограмний автомат (МПА) є однією з форм завдання структурного автомата для випадку, коли вхідні і вихідні сигнали автомата закодовані булевими змінними. МПА – це керуючий автомат, що формує мікрокоманди. Вхідний алфавіт МПА співпадає з множиною сигналів логічних умов, вихідний алфавіт – з множиною сигналів на виконання окремих мікрооперацій. Алфавіт станів і функцій переходів задається графом автомата. Мікрокоманда – це сукупність мікрооперацій, що виконуються одночасно за один такт автоматного часу. Мікрооперація – це елементарний неподільний акт обробки інформації на одному такті. Логічні умови – це змінні для задання порядку проходження мікрокоманд. Перевірка значення логічної умови на кожному такті роботи автомата дозволяє визначити мікрокоманду, яка буде реалізовуватись під час наступного такту. 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. При проектуванні різних пристроїв заздалегідь може бути складена змістовна граф-схема алгоритму (ГСА). Розробка змістовної граф-схеми є достатньо інтуїтивною.

На рис. 1,а представлено змістовну граф-схему алгоритму виконання операції $D = 2A^2 + 0,5B$. У початковому стані операнд A записано в регістрах $P1$ і C , операнд B – в регістрі $P2$. У першому такті виконується подвоєння A в $P1$ і ділення B на 2 в $P2$ шляхом зсуву слів на один розряд ліворуч і праворуч відповідно. Далі до вмісту $P2$ A разів додається слово, записане в $P1$. Після кожного додавання вміст C зменшується на 1. Обчислення закінчуються при виконанні умови $C = 0$. Результат операції формується в $P2$.

При синтезі МПА на основі змістовної ГСА спочатку формується відмічена ГСА. Відмічена ГСА – це орієнтований зв'язний граф, що включає вершини чотирьох типів: початкову, кінцеву, операторну і умовну. Входи і виходи вершин з'єднуються дугами, направленими від виходу однієї вершини до входу іншої.

Відмічена ГСА повинна задовольняти наступним умовам: кожен вихід вершини з'єднується тільки з одним входом іншої вершини; дозволяється в різних умовних вершинах записувати однакові логічні умови; один з виходів умовної вершини може з'єднуватися з її входом, що неприпустимо для операторної вершини; в кожній операторній вершині записується оператор (мікрокоманда). Одна і та ж ГСА може реалізовуватися різними структурними схемами – автоматом Мілі або автоматом Мура. Отримання різних структурних схем пов'язане з різними методиками відмітки ГСА.



д

г

в

б

а

д – граф автомата Мура

Рис. 1. Граф-схеми і графи автоматів

Відмітка ГСА автомата Мілі виконується так:

1. Символом S_0 відмічається вхід вершини, що слідує за початковою вершиною, а також вхід кінцевої вершини;
2. Символами $S_1, \dots, S_m$ відмічаються входи всієї решти вершин, що йдуть за операторними;
3. 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, представленого на рис. 1, а, показані на рис. 1, б і рис. 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 .

Після отримання відміченої ГСА будується граф автомата Мілі або Мура. Число вершин графа рівне числу станів автомата, тобто S_{m+1} . При побудові графа автомата Мілі кожній відмітці ГСА ставиться у відповідність вершина графа. Дві поименовані вершини графа з'єднуються дугою, якщо існує шлях переходу в ГСА від однієї відмітки до іншої з такими ж іменами. Поряд з дугою записується кон'юнкція логічних умов, відповідна цьому шляху, а так само множина мікрооперацій з операторної вершини, через яку проходить цей шлях. Якщо шляхів декілька, то записується диз'юнкція кон'юнкцій. Якщо між операторними вершинами перехід безумовний, то замість кон'юнкції ставиться "1", що означає, що даний перехід відбувається під час надходження синхросигналу. Граф автомата Мілі приведено на рис. 4.1,г, граф автомата Мура – на рис. 4.1,д.

При описі МПА з великим числом станів і переходів наочність графічного уявлення втрачається, тому використовується структурна таблиця МПА, на підставі якої формуються вирази функцій збудження і функцій переходів.

5. СТРУКТУРНИЙ СИНТЕЗ ЦИФРОВОГО АВТОМАТА

Для подальшого синтезу автомата використовується метод структурного синтезу, в якому можна умовно виділити наступні етапи:

1. Кодування;
2. Вибір типів елементів пам'яті автомата;
3. Вибір структурно-повної системи елементів;
4. Формування логічних функцій виходів автомата і функцій збудження тригерів;
5. Побудова логічної структурної схеми автомата.

Кодування – це процес заміни символів вхідного та вихідного алфавітів і алфавіту внутрішніх станів автомата двійковими кодами. Елементами пам'яті послідовнісних автоматів найчастіше є синхронізовані фронтом синхроімпульсів *JK*-, *RS*-, *D*- і *T*-тригери. Для тригерів характерні чотири типи переходів: $0 \rightarrow 0$, $0 \rightarrow 1$, $1 \rightarrow 0$, $1 \rightarrow 1$. Реалізація цих переходів виконується під впливом відповідних сигналів збудження (табл. 1).

Таблиця 1. Сигнали збудження тригерів

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

У табл. 1 знак "*" означає, що відповідний перехід здійснюється як при нульовому, так і при одиничному значенні сигналу. Функції збудження тригерів є булевими функціями. Для їх реалізації у вигляді

комбінаційних схем використовується яка-небудь функціонально повна система логічних елементів. Після опису сигналів збудження і вихідних сигналів логічними функціями та їх мінімізації здійснюється представлення вказаних функцій у вибраному базисі, наприклад, "І-НІ" чи "АБО-НІ". На основі отриманих мінімізованих булевих функцій розробляється логічна структура автомата, яка складається з комбінаційних схем функцій збудження, комбінаційних схем формування вихідних сигналів і елементів пам'яті.

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:

1. v_1 – установка суматора в 0;
2. v_2 – прийом і порозрядне додавання вмісту суматора до числа A ;
3. v_3 – прийом і порозрядне додавання вмісту суматора до числа B ;
4. v_4 – здійснення переносу та додавання їх до вмісту суматора;
5. v_5 – видача результату.

Операції заданого суматора можна описати у вигляді мікропрограми, представленої в табл. 2.

Таблиця 2. Операції суматора і мікропрограма його роботи

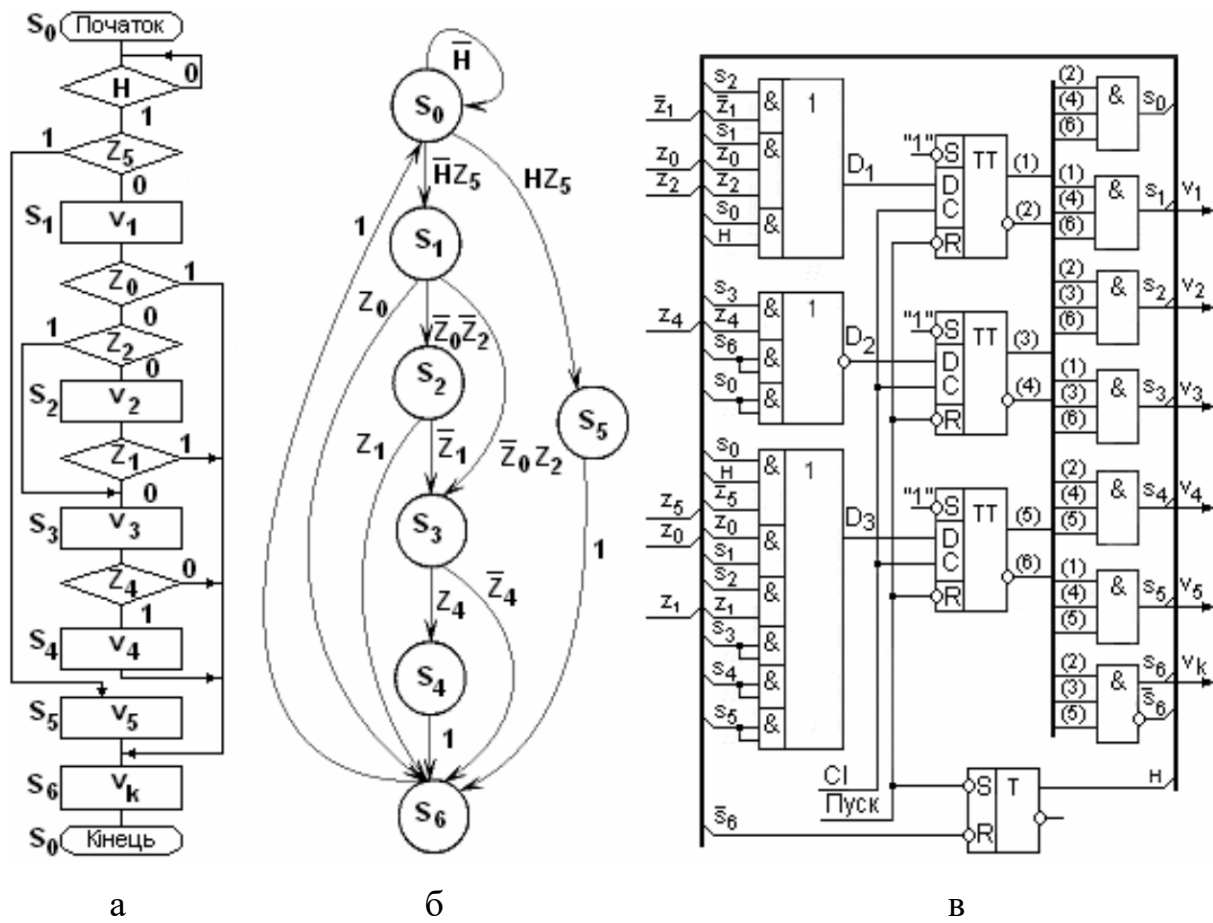
Операція	Умова	Мікропрограма
Скидання суматора	Z_0	$МП0 = Hv_1vk$
Прийом числа A	Z_1	$МП1 = Hv_1v_2vk$
Прийом числа B	Z_2	$МП2 = Hv_1v_3vk$
$C = A \oplus B$	Z_3	$МП3 = Hv_1v_2v_3vk$
$C = A + B$	Z_4	$МП4 = Hv_1v_2v_3v_4vk$
Видача результату	Z_5	$МП5 = Hv_5vk$

У табл. 2 символом H позначається початкова логічна умова, що ініціює виконання заданої операції при $H = 1$; символом vk позначається порожня мікрооперація, за допомогою якої керуючий автомат сигналізує зовнішньому пристрою про завершення операції. На підставі табл. 2 складається граф-схема мікропрограми (рис. 2, а) і виконується її відмітка. На підставі відміченої ГСА складається граф переходів (рис. 2, б) і структурна таблиця МПА Мура (табл. 3), в якій вказуються всі можливі переходи (безумовні переходи помічено знаком "*").

Таблиця 3. Структурна таблиця МПА Мура

Поточний стан	Код поточного стану	Наступний стан	Код наступного стану	Умови переходу	Функції збудження тригерів
	$Q_1^n Q_2^n Q_3^n$		$Q_1^{n+1} Q_2^{n+1} Q_3^{n+1}$		$D3 D2 D1$
S_0	0 0 0	S_0	0 0 0	$\bar{H}$	0 0 0
S_0	0 0 0	S_1	0 0 1	$H\bar{z}_5$	0 0 1
S_0	0 0 0	S_5	1 0 1	$H\bar{z}_5$	1 0 1
S_1	0 0 1	S_2	0 1 0	$\bar{z}_0\bar{z}_2$	0 1 0
S_1	0 0 1	S_3	0 1 1	z_0z_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	$\bar{z}_1$	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	$\bar{z}_4$	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

Логічну схему керуючого автомата Мура приведено на рис. 2, в.



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

в – логічна схема автомата

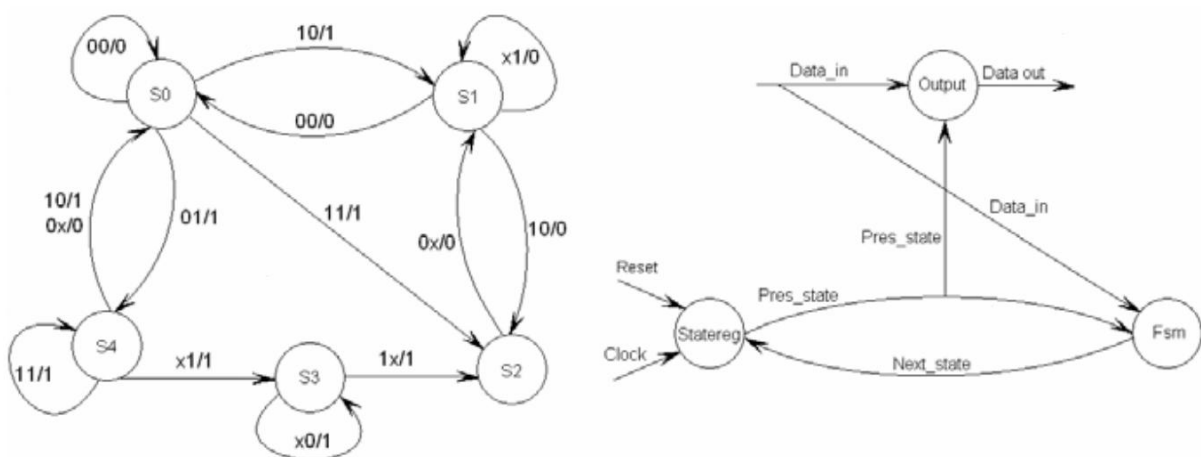
Рис. 2. Автомат Мура

Таким чином, метод структурного синтезу дозволяє отримати логічну схему автомата для подальшої реалізації у вигляді типових логічних блоків.

6. ОПИС ПОСЛІДОВНІСНИХ ПРИСТРОЇВ АВТОМАТНИМИ МОДЕЛЯМИ У VHDL-КОДАХ

ПЛІС, виконані по архітектурі *FPGA*, мають достатнє число тригерів, тому використання автоматних моделей дозволяє одержати достатньо швидкодіючу і в той же час наочну реалізацію пристрою при прийнятних витратах ресурсів. Нижче розглядається приклад проектування послідовнісного пристрою на базі автомата Мілі.

VHDL-код (додаток Б) реалізує поведінку кінцевого автомата, заданого графом (рис. 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 під час наступного такту. 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, б).



а – граф автомата; б – опис функціонування кінцевого автомата

Рис. 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*.

Кільцеві лічильники використовуються в системах управління для формування послідовностей імпульсів. Синтезуються вони на основі регістрів зсуву, для чого останній вихід регістра зсуву з'єднується з входом першого тригера регістра. На діаграмі станів (рис. 4) вхідні сигнали відсутні, оскільки єдиним вхідним сигналом для лічильника є синхронізуючий сигнал. Послідовно на кожному виході кільцевого лічильника (*Q0-Q7*) формується одиничний імпульс тривалістю в один період тактового сигналу. Ці імпульси можуть використовуватися для управління функціональними блоками цифрової системи.

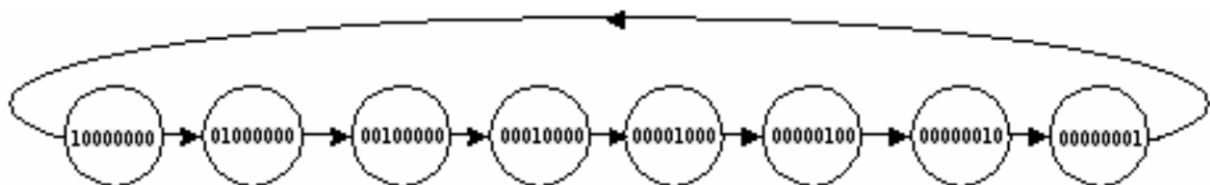


Рис. 4. Діаграма станів кільцевого лічильника

Початковий стан лічильника встановлюється за допомогою асинхронних установчих сигналів. Схема лічильника, що відповідає приведеній діаграмі станів, показана на рис. 5, а часова діаграма його роботи – на рис. 6.

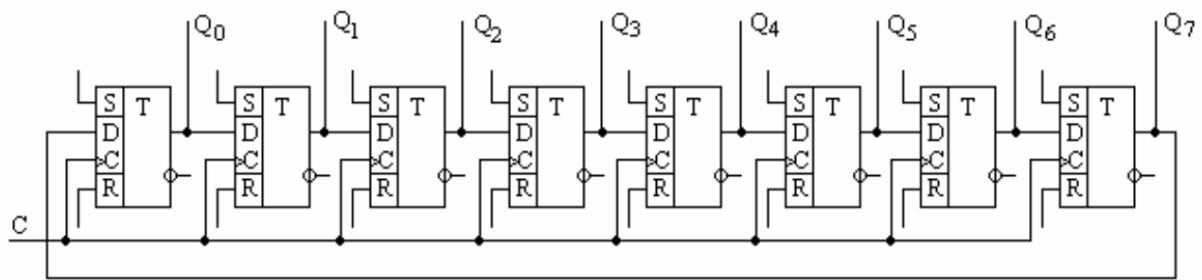


Рис. 5. Кільцевий лічильник з входами скидання і установки, що забезпечують установку лічильника в початковий стан

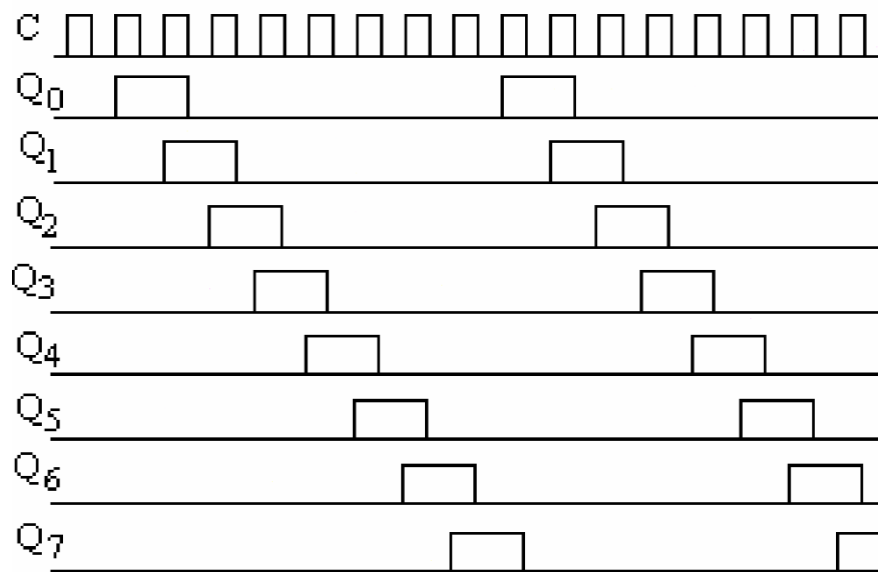


Рис. 6. Часова діаграма роботи кільцевого лічильника

Якщо сигнал з інверсного виходу останнього тригера подати на вхід першого тригера, то утворюється перехресний кільцевий лічильник, що називається лічильником Джонсона (рис. 7).

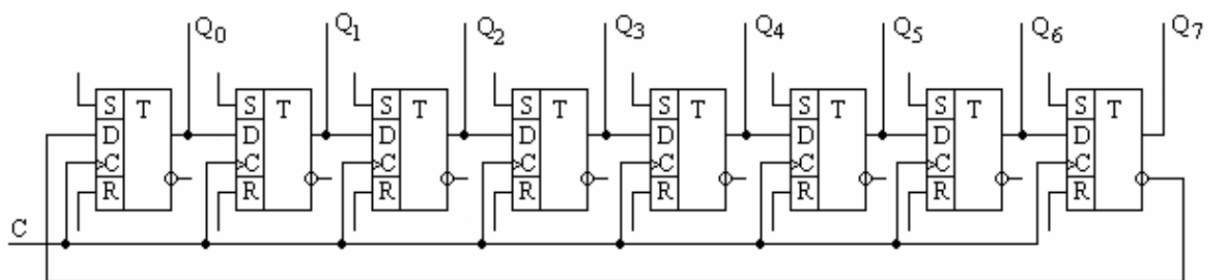


Рис. 7. Перехресний кільцевий лічильник

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

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 сформована 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.

Проблему початкових умов можна вирішити 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. Другий спосіб передбачає використання входів скидання/установки для примусової установки потрібних початкових умов. Лічильник, який спроектовано відповідно до першого способу, має на виході повторення послідовності 000, 001, 011, 110, 000, Діаграму станів такого лічильника приведено на рис. 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 зображено на рис. 8, б, а схема – на рис. 8, в.

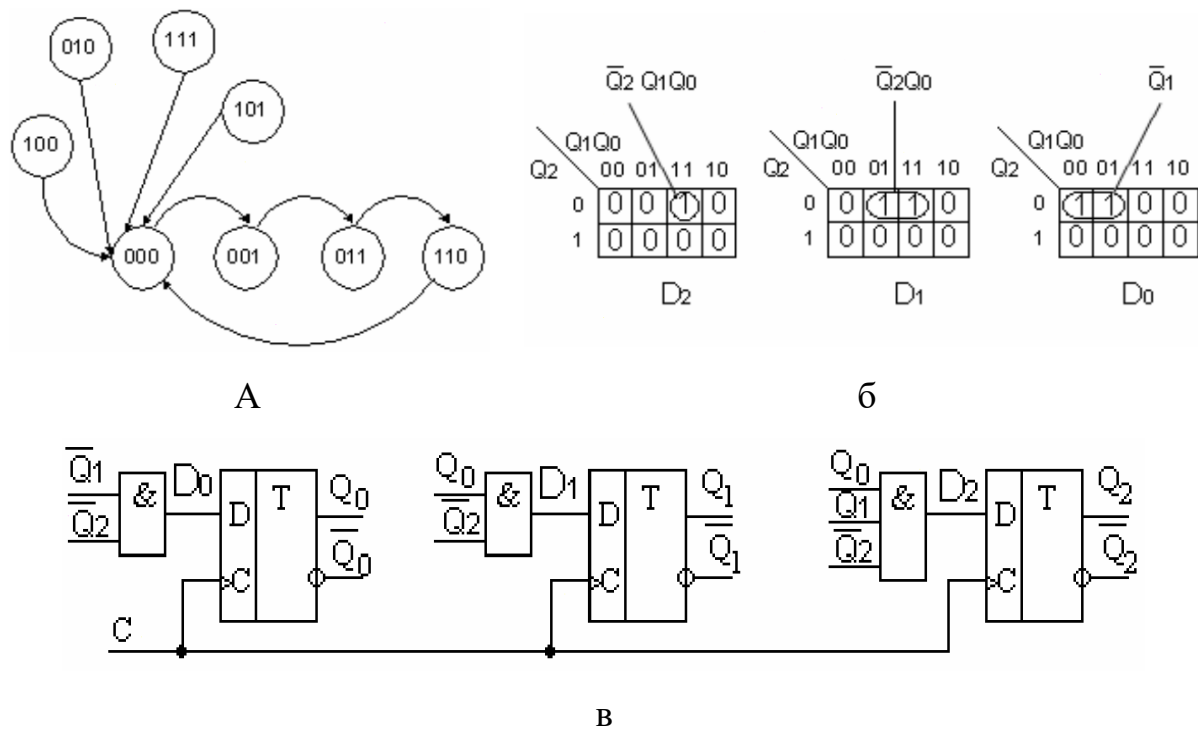


Рис. 8. Лічильник з установкою початкового стану

Наведені принципи проектування дозволяють ефективно створювати послідовнісні пристрої на базі ПЛІС. Також наведено приклад програмного опису пристрою на базі автомата Мілі.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

Основна література

1. Грушвицкий Р.И., Мурсаев А.Х., Угрюмов Е.П. Проектирование систем на микросхемах с программируемой структурой. – СПб.: БХВ-Петербург, 2006. – 736 с.
2. Стешенко В.Б. ПЛИС фирмы "Altera": Элементная база, система проектирования и языки описания аппаратуры. – М.: Издательский дом "Додэка-XXI", 2007. – 576 с.
3. Стешенко В.Б. EDA. Практика автоматизированного проектирования радиоэлектронных устройств. – М.: Издательство "Нолидж", 2002. – 768 с.
4. Соловьев, В. В. Основы языка проектирования цифровой аппаратуры Verilog: учебное пособие для студентов. – Москва : Горячая линия-Телеком, 2016. – 205 с.
5. Бибило П.Н. Основы языка VHDL. – М.: Издательство ЛКИ, 2007. – 328 с.
6. Бибило П.Н. Синтез логических схем с использованием языка VHDL. – М.: Солон-Пресс, 2009. – 385 с.
7. Угрюмов Е.П. Цифровая схемотехника. – СПб.: БХВ-Петербург, 2000. – 528 с.
8. Короткова М.А. Математическая теория автоматов: учебное пособие для вузов. – М.: Изд-во МИФИ, 2008. – 116 с.
9. Синтез цифровых автоматов: Учебное пособие / Н. Г. Захаров, В. Н. Рогов. – Ульяновск: УлГТУ, 2003. – 136 с.

Додаткова література

10. Комолов Д.А., Мьяльк Р.А., Зобенко А.А., Филиппов А.С. Системы автоматизированного проектирования фирмы Altera MAX+plus II и

- Quartus II / Краткое описание и самоучитель. – М.: ИП РадиоСофт, 2002. – 352 с.
- 11.Сергиенко А.М. VHDL для проектирования вычислительных устройств. – К.: ЧП "Корнейчук", ООО "ТИД "ДС", 2003. – 208 с.
 - 12.Стешенко В.Б. ПЛИС фирмы ALTERA: проектирование устройств обработки сигналов. – М.: ДОДЭКА, 2000. – 128 с.
 - 13.Стешенко В. Школа разработки аппаратуры цифровой обработки сигналов на ПЛИС // Chip News, 1999, № 8-10; 2000, № 1-3.
 - 14.Максфилд Клайв. Проектирование на ПЛИС. Курс молодого бойца: учебное пособие: пер. с англ. – М.: ДМК Пресс, 2010. – 407 с.
 - 15.Зотов В.Ю. Проектирование цифровых устройств на основе ПЛИС фирмы Xilinx в САПР WebPACK ISE. – М.: Горячая линия-Телеком, 2003. – 624 с.
 - 16.Бойко В.И., Гуржий А.Н., Жуйков В.Я. и др. Схемотехника электронных систем. Цифровые устройства. СПб.: БХВ-Петербург, 2004. – 512 с.
 - 17.Точчи Р.Дж., Уидмер Н.С. Цифровые системы. Теория и практика: пер. с англ. – М.: Издательский дом "Вильямс", 2004. – 1024 с.
 - 18.Хопкрофт Д., Мотвани Р., Ульман Дж. Введение в теорию автоматов, языков и вычислений: пер. с англ. – М.: «Вильямс», 2002. – 527 с.
 - 19.Карпов Ю.Г. Теория автоматов: Учебник для вузов. – СПб.: Питер, 2002. – 224 с.

ДОДАТОК А

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 обирається створена робоча папка.

Створення текстового файлу. Якщо текстовий файл, створений в будь-якому текстовому редакторі (наприклад, в редакторі MS Word), вже існує, то необхідно його вміст записати в буфер пам'яті за допомогою введення команд *Edit|Select All|Ctrl+C*. Далі слід активізувати текстовий редактор, задавши послідовність команд *Max+plusII|Text Editor*, а також перемістити інформацію з буферу пам'яті за допомогою натиснення поєднання клавіш *Ctrl+V*. Якщо введений файл є програмою, написаною на мові VHDL, то його слід зберегти в директорії проекту з розширенням *.vhd*. При цьому задається послідовність команд *File|Save As* і вводиться ім'я файлу (наприклад, *vlsi_1.vhd*).

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

Створення Include-файлу і символу бібліотечного елементу. Include-файл створюється командами *File|Create Default Include File* і записується в бібліотеку користувача. Прочитати цей файл і уточнити назви входів і виходів можна за допомогою виклику додатку *Hierarchy*

Display. При цьому відображаються всі модулі проекту і їх взаємозв'язки, а також всі типи файлів, сформовані в процесі обробки проекту. *Include*-файл відображається з розширенням *.inc*. Його активізація дозволяє проглянути вміст вказаного файлу.

Створення графічного файлу. Для виклику графічного редактора потрібно в меню *Menager* вибрати *Max+PlusII|Grafic Editor*. Графічному файлу із схемою необхідно командою *File|Save As* привласнити ім'я з розширенням *.gdf* (наприклад, *vlsi_1.gdf*). Після того, як функціональні блоки введені, потрібно ввести символи вхідних і вихідних портів. Їх необхідно імпортувати з бібліотеки примітивів. Для цього необхідно двічі клацнути мишею по порожньому полю графічного редактора. Відкриється діалогове вікно, в якому в меню *Symbol Libraries* вказана бібліотека знаходиться за адресою *c:\program file\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* і *output*. Далі необхідно привласнити імена всім портам.

Симуляція. Симуляція – це процес функціонального моделювання роботи схеми. Перед виконанням моделювання необхідно створити тестові вектори, тобто задати значення вхідних сигналів. Для цієї мети можна використати редактор діаграм, який вибирається послідовністю команд *Max+PlusII|Waveform Editor*. Коли вікно редактора відкриється, створюється файл (наприклад, з назвою *vlsi_1.scf*) послідовністю *File|Save As* і вказівкою назви файлу (наприклад, *vlsi_1.scf*) в рядку *File Name* діалогового вікна, що відкрилося. Далі визначаються вхідні і вихідні сигнальні лінії схеми для процесу симуляції. Для цього використовуються сигнальні лінії, занесені в *SNF-файл (Simulator Netlist File)*, створений на етапі компіляції схеми. Необхідно відкрити список доступних в *SNF-файлі* сигнальних ліній за допомогою введення *Node|Enter Node from SNF*.

Відкриється екран з двома вікнами: *Available Nodes & Groups* і *Selected Nodes & Groups*. Після натиснення *List* в першому вікні з'явиться список вхідних і вихідних ліній з *SNF-файлу*. Необхідно скопіювати список вхідних і вихідних ліній в друге вікно, тобто створити список сигнальних ліній, що обираються. Після введення *OK* у вікні графічного редактора буде відображено вхідні та вихідні лінії. Далі задається кінцевий час симуляції введенням *File|End Time* і інтервал часової сітки *Options|Grid Size*. Для установки значень вхідних сигналів можна скористатися одним із способів: за допомогою вертикального репера встановити клацанням з протяганням тривалість сигналу, а потім ввести його значення за допомогою кнопки символом "1" на лівій інструментальній панелі. Для запуску пакету моделювання потрібно або ввести *File|Simulator*, або клацнути по кнопці симулятора на головній інструментальній панелі. Якщо результати моделювання виявилися успішними, можна за допомогою виклику додатку *File|Timing Analyzer* відобразити таблицю "*Delay Matrix*", в якій записані затримки формування вихідних сигналів щодо вхідних сигналів.

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

ДОДАТОК Б

ЛІСТИНГ VHDL-ПРОГРАМИ

"АВТОМАТ МІЛІ З П'ЯТЬМА СТАНАМИ"

```
library ieee;
use ieee.std_logic_1164.all;
entity mealy is
    port (clock, reset: in std_logic;
          data out: out std_logic;
          data in: in std_logic_vector (1 downto 0));
end mealy;
architecture behave of mealy is
    type state_values is (st0, st1, st2, st3, st4);
    signal pres_state, next_state: state_values;
begin
    statereg: process (clock, reset)
        --FSM perictp
    begin
        if (reset = '0') then
            pres_state <= st0;
        elsif (clock'event and clock = '1') then
            pres_state <= next_state;
        end if;
    end process statereg;
    fsm: process (pres_state, data_in)
        --FSM – процесс
    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 fms;
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;

```

ЗМІСТ

ВСТУП.....	3
1. ЗАВДАННЯ НА КУРСОВЕ ПРОЕКТУВАННЯ	4
2. ВИМОГИ ДО ОФОРМЛЕННЯ КУРСОВОГО ПРОЕКТУ	5
3. АНАЛІЗ ПРИСТРОЮ, ВИБІР ПЛАТФОРМИ ТА ТЕСТУВАННЯ ПРИСТРОЮ.....	6
4. РОЗРОБКА МОДЕЛЕЙ ПРИСТРОЮ.....	11
5. СТРУКТУРНИЙ СИНТЕЗ ЦИФРОВОГО АВТОМАТА.....	15
6. ОПИС ПОСЛІДОВНІСНИХ ПРИСТРОЇВ АВТОМАТНИМИ МОДЕЛЯМИ У VHDL-КОДАХ.....	19
РЕКОМЕНДОВАНА ЛІТЕРАТУРА	24
ДОДАТОК А.....	26
ДОДАТОК Б	29