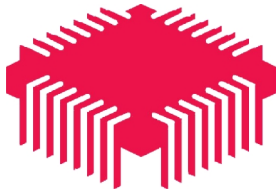
В. Л. Кофанов, О. В. Осадчук, Д. В. Гаврілов

ЛАБОРАТОРНИЙ ПРАКТИКУМ З ДОСЛІДЖЕННЯ ЦИФРОВИХ ПРИСТРОЇВ НА ОСНОВІ САПР MAX+PLUS II



Міністерство освіти і науки України
Вінницький національний технічний університет

В. Л. Кофанов, О. В. Осадчук, Д. В. Гаврілов

ЛАБОРАТОРНИЙ ПРАКТИКУМ З ДОСЛІДЖЕННЯ ЦИФРОВИХ ПРИСТРОЇВ НА ОСНОВІ САПР MAX+PLUS II

Затверджено Вченою радою Вінницького національного технічного університету як лабораторний практикум для студентів бакалаврського ступеня підготовки радіоелектронних спеціальностей вищих навчальних закладів освіти. Протокол № 5 від 28 грудня 2006 р.

Вінниця ВНТУ 2008

УДК 621.374
К 74

Рецензенти:

О. Д. Азаров, доктор технічних наук, професор
С. М. Злепко, доктор технічних наук, професор
В. П. Манойлов, доктор технічних наук, професор

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти та науки України

Кофанов В. Л., Осадчук О. В., Гаврілов Д. В.
К 74 **Лабораторний практикум з дослідження цифрових пристроїв на основі САПР MAX+PLUS II.** Лабораторний практикум. – Вінниця: УНІВЕРСУМ-Вінниця, 2006. – 200 с.

Лабораторний цикл поєднує виконання практичних завдань, моделювання і дослідження цифрових пристроїв (ЦП) на засадах сучасної технології інженерної праці з використанням повноциклової САПР MAX+PLUS II. Особливість циклу в тому, що ЦП засвоюються паралельно на традиційній елементній базі і на сучасних. З метою активізації навчання передбачено розробку, моделювання, складання та дослідження ЦП згідно з варіантом індивідуального завдання.

Змістом посібника є типові структурні одиниці комбінаційних і послідовнісних ЦП на рівні примітивів, макрофункцій і мегафункцій, що охоплюють відповідні розділи дисциплін „Цифрові пристрої та мікропроцесори”, „Проектування цифрових пристроїв”, „Елементна база цифрових систем зв’язку”. Крім того, матеріал є основою для курсового і дипломного проектування.

УДК 621.374

ЗМІСТ

ВСТУП

Змістом лабораторного циклу є дослідження і проектування цифрових пристроїв (ЦП) на засадах сучасної технології інженерної праці з використанням систем автоматизованого проектування (САПР). Особливість методики навчання полягає в тому, що елементи САПР засвоюються паралельно з основами цифрових пристроїв.

З огляду на те, що інтегральні схеми (ІС) жорсткої структури можна розглядати як компоненти сучасних ІС програмованої структури, під час лабораторних досліджень поєднано засвоєння основ реалізації ЦП на традиційній і новій елементній базі. Такий підхід зумовлений тим, що універсальність складних великих і надвеликих інтегральних мікросхем програмованої структури (надалі позначатимемо їх терміном ПЛІС – програмовані логічні ІС) і пов'язане з цим підвищення їх серійноспроможності та зниження вартості радіоелектронної апаратури (РЕА) має спричинити перехід цифрової радіоелектроніки, в основному, на нову елементну базу.

Через складність ПЛІС (в одному кристалі міститься від тисяч до мільйонів еквівалентних вентилів типу двовходових логічних елементів) було розроблено нові автоматизовані методи синтезу цифрових структур і комплексні засоби проектування апаратури на персональних комп'ютерах. Зокрема, для популярних ПЛІС фірми Altera впроваджено системи автоматизованого проектування MAX+plus і Quartus кількох версій. Хоча САПР різних фірм мають відмінності, методика проектування на їх основі є багато в чому спільною. Для визначеності в цьому циклі користуватимемося версією пакета MAX+plus II Baseline, яка підтримується фірмою Altera для університетської освіти. Система є *повноцикловою* (типу EDA – electronic design automation). На відміну від попередніх систем (типу CAD – computer aided design) вона забезпечує всі етапи проекту – від моделювання до фізичного програмування великих інтегральніх схем (ВІС) та тестування результатів.

В1 Початкові поняття програмного забезпечення MAX+plus II

Проект (**Design)** – комплект файлів, створених користувачем і програмним забезпеченням для досягнення поставленої мети. Проектним файлом (**Design File**) з розширенням, що закінчується на **df**, називають такий, на основі якого створюється проект. Низка інших файлів, частина яких утворюється автоматично, є службовими або допоміжними, причому назва всіх файлів (але кожна зі своїм розширенням) має збігатися з ім'ям проекту (без розширення). **Subdesign** (підроект) – проект найнижчого рівня, що є або остаточним, або складником проекту вищого рівня, який у свою чергу може правити за підроект деякого суперпроекту і так далі за ієрархією.

Device (пристрій, мікросхема) – надвелика інтегрована мікросхема програмованої структури. **Pin** (штирок) – реально існуючий вивід мікрос-

хеми для зовнішніх з'єднань. **Port** (у даному сенсі – вивід, контакт) – вивід мікросхеми для обміну із зовнішніми пристроями, тобто штирок, на якому діє інформаційний сигнал.

Symbol (символ) – файл проекту, що відображає або стандартні логічні функції (серед них є складні), вбудовані в програмне забезпечення, або функції, створені користувачем, у тому числі для всього проекту чи його частини. Графічно символ зображається спрощеним умовним позначенням зі всіма входами та виходами. **Pinstub** (місце підімкнення, затискач) – умовно позначений вивід символу для зв'язку з внутрішніми вузлами або портами. **Node** (вузол) – у програмних файлах відображає лише внутрішній сигнал (на відміну від порту, що репрезентує зовнішній сигнал).

В2 Вікно MAX+plus II Manager

САПР MAX+plus II складається з програмних модулів, керування якими здійснюється з головного вікна Manager, що з'являється одразу після запуску програми (рис. В1).

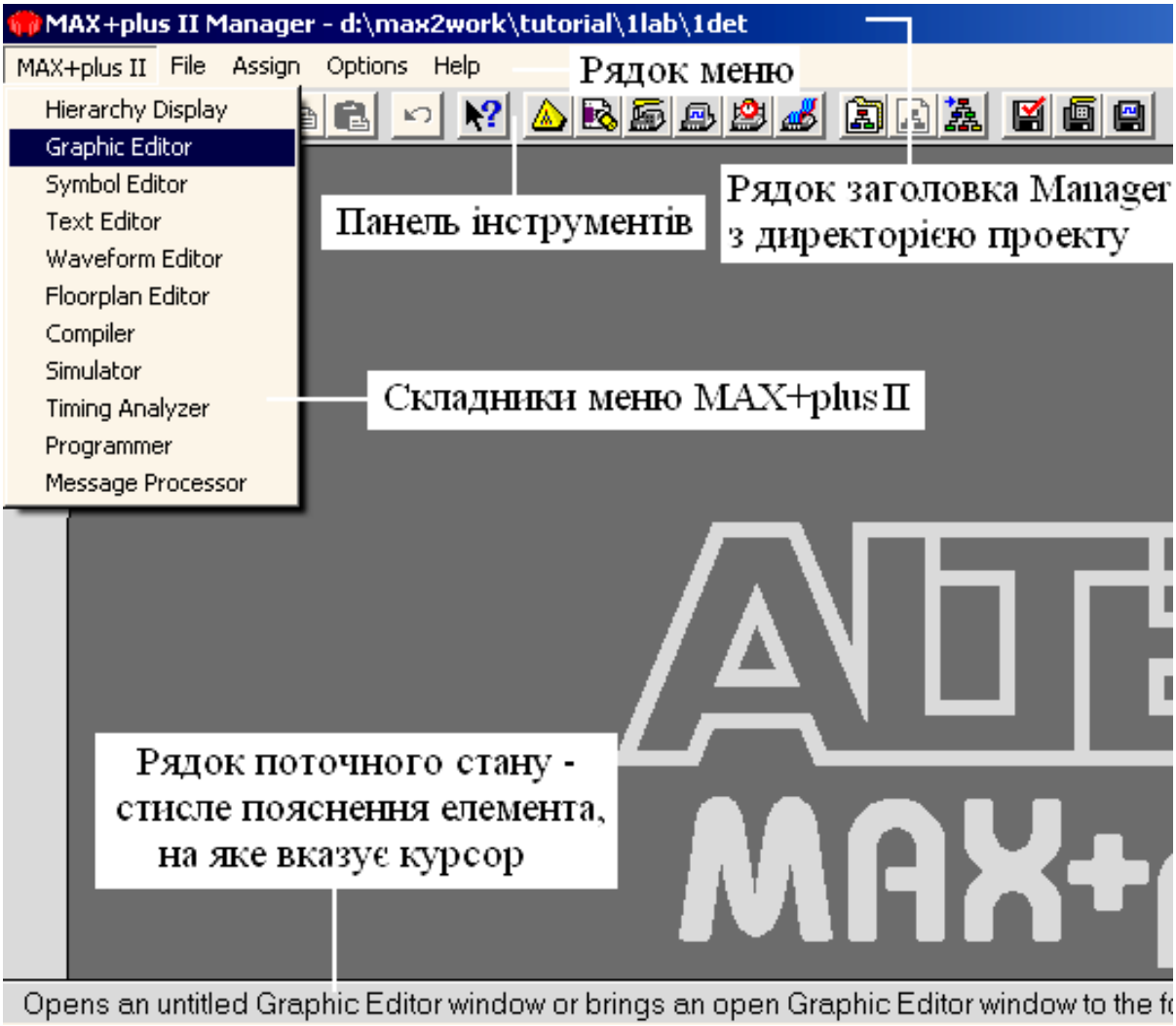


Рисунок В1

Рядок заголовка (**Title Bar**) містить назву пакета (MAX+plus II), повний шлях до папки з поточним проектом (директорію) та ім'я останнього. *Оброблятися можуть файли лише того проекту, ім'я якого зазначене в цьому рядку*, тому для переходу до потрібного проекту необхідно встановити його ім'я в рядку заголовка, що легко зробити, користуючись назвою будь-якого файла цього проекту .

Рядок меню (**Menu**) складається з елементів, набір яких не є сталим, а *змінюється залежно від виконуваного модуля програми* (у кожному модулі до 10 складників меню). Але завжди відображаються 5 складників:

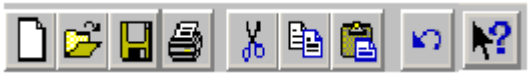
– *MAX+plus II* – викликає потрібний програмний модуль;

– *File* – крім звичайних операцій з файлами (створення нового файла – New, відкриття – Open, видалення – Delete File та виходу з програми – Exit MAX+plus II) містить низку команд керування проектом, частина з яких розкривається списком у рядку Project;

– *Assign* – складається з меню призначень проекту, серед яких вибір мікросхеми – Device, її виводів – Pin/Location/Chip, параметрів логічного синтезатора і часового аналізатора та ін.;

– *Options* – відкриває меню опцій менеджера, серед них User Libraries (бібліотека користувача) з файлами сформованих символів і Preferences (надання переваги) – діалогове вікно з низкою корисних налаштувань: Ask Before Closing MAX+plus II (запитувати перед закриттям програми), Ask Before Deleting Files (запитувати перед видаленням файлів), Iconize Compiler at Start of Processing та Iconize Simulator at Start of Processing (згортати до значків вікна компілятора та імітатора під час їх старту), Show Toolbar та Show Status Bar (показувати рядок інструментів та рядок стану), Automatically Load Last Project at Start-Up (автоматично завантажувати останній проект під час запуску MAX+plus II); доцільно, принаймні, до набуття досвіду встановити прапорці всіх опцій, крім двох Iconize... (вікна компілятора та імітатора під час активізації цих модулів доцільно залишити на екрані аби переконатися, що обробляється файл потрібного проекту, та в правильності деяких налаштувань);

– *Help* – відкриває меню довідки.



Інструментальна панель (**Toolbar)** відображає піктограми (деякі з них відкривають діалогові вікна), що дозволяють швидко викликати команди, які містяться в меню. Перші дев'ять з них – звичайні команди Windows (New – новий файл, Open – відкрити, Save – зберегти, Print – друкувати, Cut – вирізати, Copy – копіювати, Paste – вставити, Undo – скасувати останню дію, Context-Sensitive Help – контекстно-залежна довідка), а 12 наступних – дублюють команди модулів з меню MAX+plus II. Крім того, з

десяток спеціальних піктограм з’являються залежно від активного модуля. Ще одна, вертикальна, *палітра інструментів* (ліворуч у вікні) активізується в окремих модулях для формування файлів (графічних, символьних і часових діаграм).

Рядок поточного стану (**Status Bar**) відображає стислий опис елемента панелі або палітри інструментів, на який вказує курсор маніпулятора (миші).

В3 Меню MAX+plus II

Компонент меню „MAX+plus II” є найголовнішим, бо з нього запускаються всі програмні модулі (програми-додатки), призначені для повного циклу обробки проекту. Розгорнутий список складників меню MAX+plus II у вікні Manager показано на рисунку В1. Необхідний мінімум відомостей щодо використання програмних модулів (деякі з них називаються редакторами) наочно викладається на конкретних прикладах безпосередньо в лабораторному циклі, а тут стисло охарактеризуємо лише їх призначення.

Графічний редактор (**Graphic Editor)** призначений для введення проекту у вигляді схеми з'єднань символів компонентів, що перебувають у стандартних бібліотеках пакета або в бібліотеках користувача. Результатом є сформований проектний файл з розширенням **.gdf (**Graphic Design File).

Редактор часових діаграм (**Waveform Editor)** виконує функцію введення діаграм (епюр) вхідних сигналів та формування на основі компіляції і функціонального моделювання логіко-часових вихідних діаграм з метою наочної імітації функціонування пристрою. Результатом є файл з розширенням **.scf (**Simulator Channel File). За допомогою цього редактора можна також створити проект шляхом введення потрібних часових діаграм. Сформований таким чином проектний файл має розширення **.wdf (**Waveform Design File).

Текстовий редактор (**Text Editor**) призначений для створення і редагування проектних файлів, в яких проект вводиться мовою опису пристроїв AHDL (Altera Hardware Description Language) або близькими до неї мовами типу VHDL чи Verilog HDL. Результатом є файл з розширенням **.tdf**. (Text Design File). Цим редактором можна також задати в текстовому вигляді часові діаграми (вектори) сигналів у файлі з розширенням **.vec**.

Символьний редактор (**Symbol Editor**) дозволяє редагувати існуючі символи і створювати нові; будь-який відкомпільований проект може бути згорнутий до символу, поміщений до бібліотеки символів і використаний як елемент іншого проекту. Символи зручно застосовувати разом з іншими схемними компонентами в графічному і текстовому

редакторах. За ієрархічної побудови проекту його основний файл за допомогою символів набуває вигляду структурної схеми, що спрощує створення складних пристроїв. Файл цього редактора має розширення **.sym** (Symbol File).

Редактор фізичного розміщення (**Floorplan Editor**), виконує автоматичне розміщення елементів мікросхеми і трасування з'єднань. Також є можливість ручного редагування зв'язків на плані розташування комірок мікросхеми. Результатом є конфігураційний файл, необхідний для фізичного програмування структури кристала; створюється також файл звітності з інформацією, зокрема, про використання ресурсу ІС.

Компілятор (**Compiler)** є головним програмним модулем, що виконує функції перевірки коректності введення проекту і локалізації помилок, математичного моделювання щодо оптимізації логічних функцій та часового аналізу, розподілу елементів по комірках мікросхеми тощо. Результатом виконання підпрограм компілятора є формування низки службових файлів, які використовуються іншими редакторами, у тому числі для програмування та конфігурування ІС.

Імітатор (**Simulator)** призначений для функціонального моделювання проекту з метою перевірки правильності логіки його функціонування. Результатом моделювання є формування часових діаграм вихідних сигналів, які є реакцією пристрою на вхідні сигнали, задані за допомогою редактора часових діаграм або текстового редактора. Вихідні сигнали відображаються у файлах з розширенням **.gdf** і **.vec**.

Часовий аналізатор (**Timing Analyser**) виконує розрахунок часових інтервалів поширення сигналів між різними вузлами схеми скомпільованого проекту. Результатами аналізу є матриця затримок або дисплей швидкодії, які можна зберегти у вигляді таблиці файла з розширенням **.tao** (Timing Analyzer Output File – вихідний файл часового аналізатора).

Дисплей ієрархії (**Hierarchy Display**) забезпечує створення проекту ієрархічної структури, який може складатися з множини проектів різних рівнів, причому число рівнів не обмежується. Основний проект (найвищого рівня), створений у графічному редакторі, відображається у файлі дисплея ієрархії зв'язками між складниками з гіперпосиланнями.

Програматор (**Programmer**) виконує функції фізичного програмування і верифікації мікросхем, тобто створення протягом хвилин реально працюючих пристроїв. Крім того, перед фізичним програмуванням можна здійснити перевірку що мікросхема є порожньою, а після нього виконати верифікацію (перевірку на відповідність пристрою даним у файлі програмування) та функціональне тестування (перевірку на відповідність сигналів потрібним часовим діаграмам).

Процесор повідомлень (**Message Processor**) забезпечує формування, відображення і локалізацію (зазначення місця в проекті, якого стосується певна інформація) повідомлень трьох типів: про помилки (Error), попередження (Warning) і інформаційні (Info). Причину появи того чи іншого повідомлення можна з'ясувати через опцію Help on Message менеджера. Компіляція проекту неможлива до повного усунення помилок, а за наявності попереджень вона успішно завершується, проте попередження мають бути ретельно проаналізовані з наступним усуненням їх причин або ігноруванням залежно від наслідків. Інформаційні повідомлення несуть додаткову інформацію, яку можна взяти до уваги на етапі конфігурування мікросхеми.

В4 Методика виконання лабораторних робіт

Лабораторний цикл охоплює: а) теоретичний матеріал з основних розділів цифрових пристроїв, б) виконання практичних завдань і в) лабораторні дослідження. Основна увага приділяється засвоєнню основ цифрових пристроїв, а комп’ютерна техніка є інструментарієм для відпрацювання навичок сучасної інженерної праці, коли левова частка діагностики під час експлуатації діючої апаратури та проектування і налагодження нової апаратури має припадати на програмне моделювання за допомогою САПР. З метою активізації самостійного навчання лабораторні роботи мають варіантний характер – крім спільної частини дослідження типових пристроїв передбачено також індивідуальне завдання. Методично лабораторний цикл побудовано таким чином, аби роботи можна було виконати самостійно, у дистанційному режимі.

З огляду на це перед виконанням лабораторної роботи необхідно: а) засвоїти теоретичний матеріал з її теми, б) виконати практичне завдання згідно зі своїм варіантом, в) ознайомитися з лабораторним завданням з метою з’ясувати, які процедури САПР і елементи програмного забезпечення потрібно відпрацювати під час дослідження. У зв’язку з надзвичайно розгалуженою системою MAX+Plus II рекомендується обмежитися потрібним мінімумом відомостей щодо САПР, достатніх для виконання даної роботи.

В5 Зміст звіту з виконання лабораторної роботи

Назва і мета роботи; результати виконання домашнього завдання; зміст кожного пункту лабораторного завдання: файли зі схемами, часовими діаграмами та іншими результатами зі стислими поясненнями і висновками.

В6 Універсальний лабораторний стенд

Лабораторний стенд призначений виконувати такі функції: 1) моделювати ЦП і їх характеристики на комп’ютері; 2) реалізовувати реальні

пристрої на ПЛІС, що еквівалентно складанню і дослідженню ЦП на будь-якій елементній базі; 3) одночасно з досліджуваним ЦП створювати вимірювальні пристрої, зокрема, осцилограф; 4) відпрацьовувати (макетувати) прототипи реальних розробок, у тому числі під час курсового і дипломного проектування та науково-дослідної роботи (НДР).

Основу стенда становлять персональний комп'ютер PC, лабораторна плата UP2 (UP – University Program) фірми Altera (США) та з'єднувальний пристрій ByteBlaster (рис. В2), описаний у лабораторній роботі № 9. Крім того, є можливість приєднувати джерела сигналів, радіовимірювальні та інші засоби, а також створювати систему з кількох плат. Ресурс плати UP2 поділено між двома мікросхемами – родини MAX7000S (тип EPM7128S) та FLEX10K (тип EPF10K70). На функціональній схемі (рис. В2) подано елементи, що стосуються першої ПЛІС, позначення яких відповідає схемі розташування (рис. В3). За функціональним призначенням ці елементи можна поділити на групи, умовно відокремлені секціями (рис. В2).

1 ***Мікросхема з функціональними рознімачами.*** ПЛІС EPM7128S встановлена у 84-контактний рознімач з пластиковим корпусом і виводами J-форми типу PLCC (Plastic J-lead Chip Carrier) – по 21 контакту з кожного боку. По периметру мікросхеми розташовано чотири 22-контактні дублювальні рознімачі P1...P4, з'єднані з виводами відповідного боку ПЛІС. Номери виводів мікросхеми позначено на самій платі, а їх з'єднання з контактами рознімачів P1...P4 наведено в таблиці Б1 (тут і далі – див. додатки), де символом „x" позначено невикористовувані контакти. Живлення, „земля" та програмувальні сигнали JTAG до цих рознімачів не підведено. Контактні гнізда PS призначені для зовнішніх з'єднань сигнальних виводів користувача I/O. Додатковий рознімач розширення MAX_EXPANSION призначений для під'єднання зовнішніх пристроїв. Його контакти з'єднано зі входами/виходами користувача I/O, глобальними входами, а також контактами живлення та „землі" мікросхеми [9].

2 ***Живлення.*** Через коаксіальний кабель DC_IN до плати подається напруга 7...9 *В* від зовнішнього нестабілізованого джерела живлення, яке має забезпечувати струм не менше 350 мА. Наявність зовнішньої напруги індикується зеленим світлодіодом POWER, а її величину можна контролювати з контактних площин RAW. Усі компоненти плати живляться напругою VCC = 5 В від вбудованого інтегрального стабілізатора напруги Stab типу LM340T. Стабілізовану напругу можна контролювати з контактних площин DC.

3 ***Тактовий генератор.*** На платі є вбудований тактовий генератор G з кварцовою стабілізацією частоти 25,175 МГц. Вихід генератора з'єднано з глобальним синхровходом мікросхеми GCLK (вивід 83).

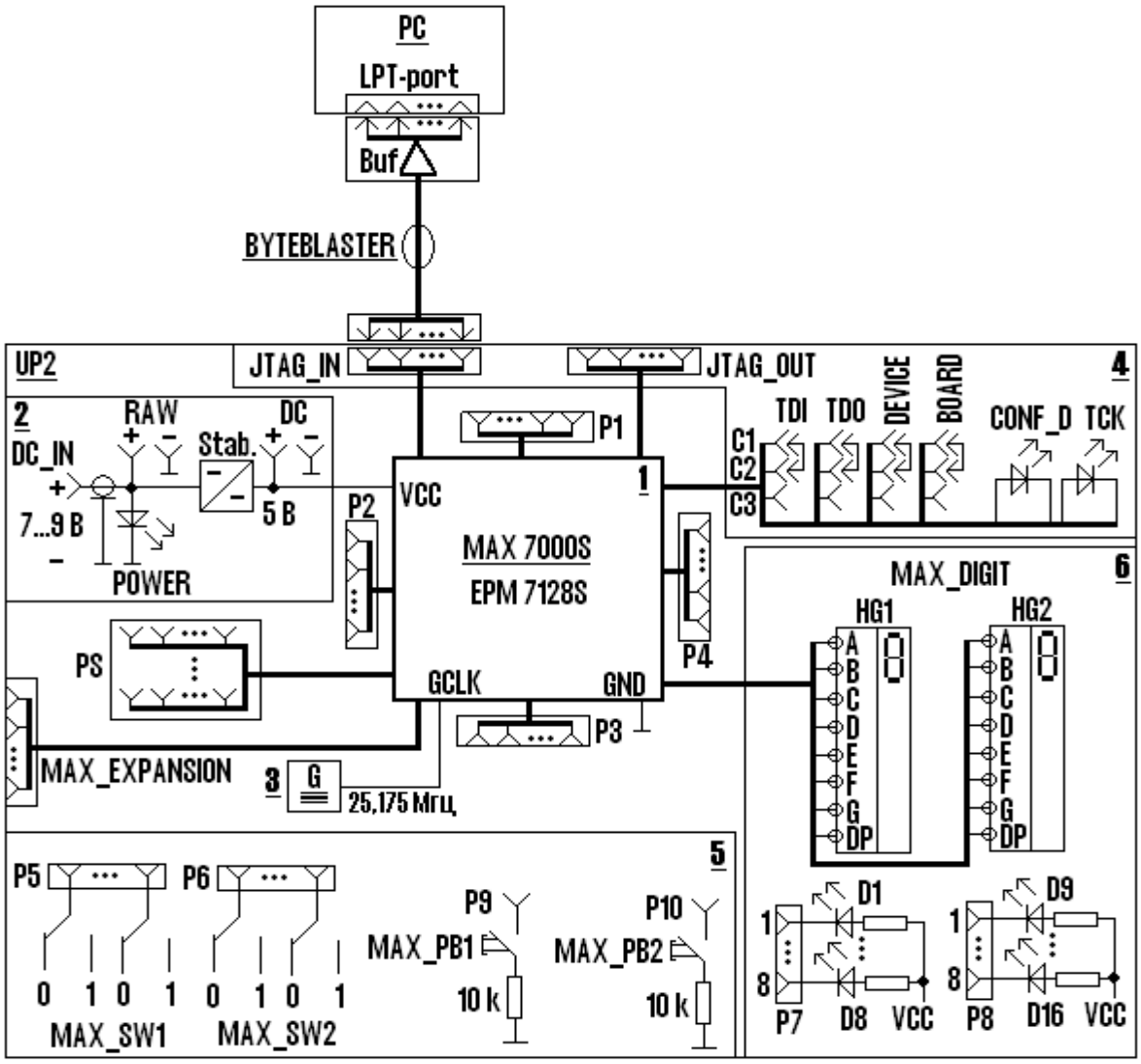


Рисунок В2

4 ***Інтерфейс JTAG.*** Фізичне програмування мікросхеми здійснюється через інтерфейс JTAG, докладно описаний в лабораторній роботі №9. Вхідний рознімач цього інтерфейсу JTAG_IN призначений для з'єднання мікросхеми з комп'ютером через завантажувальний пристрій ByteBlaster II. Як видно з таблиці Б2, до цього рознімача крім програмувальних сигналів підводиться також напруга VCC для живлення буферів Buf, вмонтованих у рознімач пристрою ByteBlaster II. Вихідний рознімач JTAG_OUT інтерфейсу використовується для програмування ланцюжка мікросхем. Режими програмування і конфігурування (таблиця Б3) задаються за допомогою чотирьох штирьових триконтактних перемикачів С1, С2, С3 (на рис. В2 наведено їх положення в режимі програмування однієї мікросхеми), що замикаються перемичками (джамперами). Миготінням світлодіода TCK індикується передача даних під час програмування, а світінням світлодіода CONF_D – завершення цього процесу.

5 ***Перемикачі для задання вхідних сигналів.*** Два задавальні DIP-перемикачі MAX_SW1 і MAX_SW2 (по вісім ключів у кожному), дозво-

ляють сформувати 16 статичних рівнів логічного 0 та 1 на з'єднаних з ними групах контактів P5 і P6 (розімкнений стан ключа відповідає рівневі логічної 1, а замкнений – рівневі логічного 0). За допомогою двох нефіксовних кнопкових перемикачів MAX_PB1 і MAX_PB2, які під час натискання замикають контакти P9 та P10 на землю через резистори 10 кОм, можна сформувати одноразові перепади рівнів, наприклад, на входах скидання або синхронізації. Для подачі сигналів на входи мікросхеми обидва зазначені види перемикачів з'єднують з контактними гніздами дротами.

6 ***Індикатори вихідних сигналів.*** Цифровий індикатор MAX_DIGIT на два знакомісця виконано на ІС семисегментних індикаторів зі спільним катодом, отже, сегменти засвічуються рівнем логічного 0. Позначення сегментів пояснюється в 3 розділі, а їх зв'язок з виводами мікросхеми наведено в табл. В4. Крім того, для індикації статичних станів вихідних сигналів плата містить 16 світлодіодів, запалюваних рівнем логічного 0 (струм через них задається резисторами 330 Ом). Світлодіоди D1…D8 з'єднано з контрольними гніздами 1...8 групи P7; так само з гніздами 1...8 групи P8 з'єднано діоди D9…D16, відповідно.

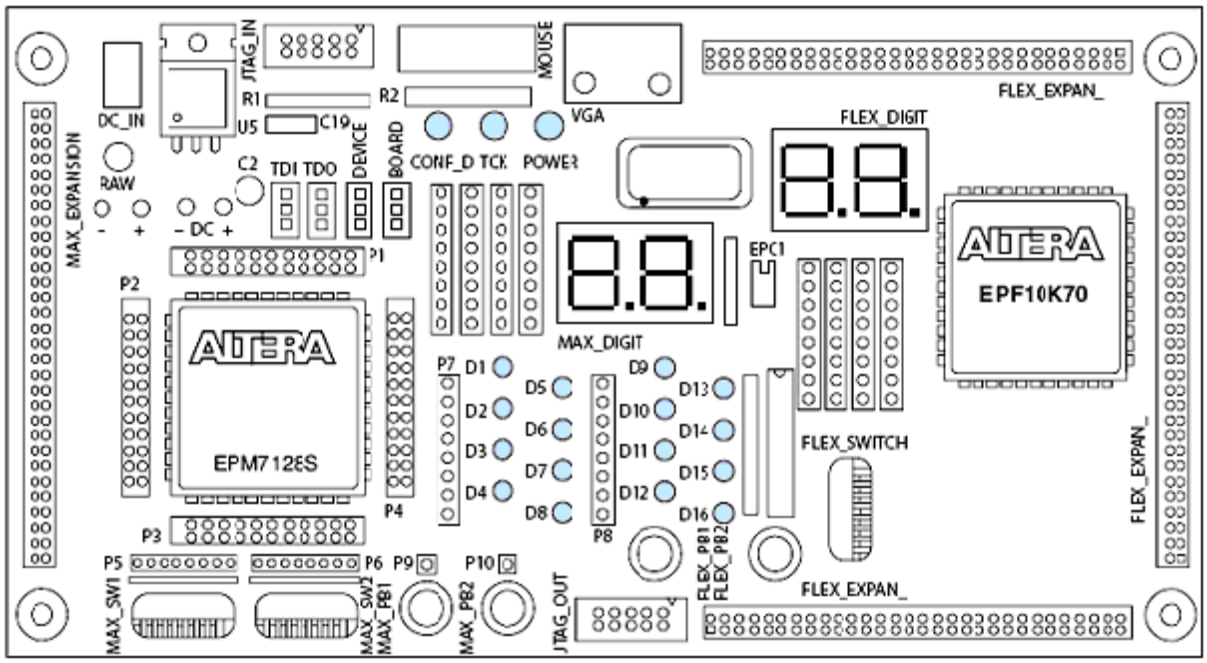


Рисунок В3

1 ОСНОВНІ ЛОГІЧНІ ФУНКЦІЇ

Мета роботи: дослідження типових логічних елементів; засвоєння основ керування файлами у MAX+plus II; засвоєння основ часового аналізу.

Домашнє завдання

1) Засвоїти теоретичні відомості щодо основних логічних функцій, співвідношень алгебри логіки а також арифметичних основ цифрової техніки, необхідних для розуміння логічних схем. Ознайомитися із загальними відомостями щодо пакета MAX+plus II (див. вступ).

2) Визначити, яку логічну функцію виконують схеми з елементом, заданим згідно з варіантом (див. додаткок А, варіанти завдання 1).

1.1 Стислі теоретичні відомості

1.1.1 Основи алгебри логіки

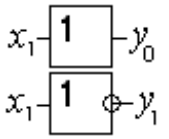
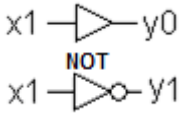
Цифровий пристрій є перетворювачем інформації в тому сенсі, що сигнал на кожному з його виходів залежить від сукупності вхідних сигналів за чітко визначеними правилами. Ці правила описуються мовою алгебри логіки, яка оперує з логічними змінними й функціями (вони називаються також двійковими, бо можуть набувати двох значень: логічного 0 або 1). Отже, алгебра логіки є основою не тільки проектування ЦП, але й розуміння принципу їх дії. Вона використовується також за текстового опису проектів мовами програмування високого рівня HDL (Hardware Description Language – мови опису апаратних засобів), зокрема, мовою AHDL, яка пристосована до ІС фірми Altera. Основні логічні функції та зображення їх мовою AHDL наведено в таблиці 1.1.

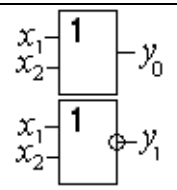
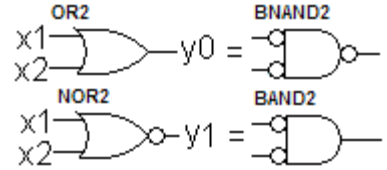
Функціонально повну систему або *базис* становить набір логічних функцій, за допомогою якого можна утворити логічну функцію будь-якої складності, наприклад, система функцій НЕ, АБО, І утворює *булів базис*. *Мінімально повною* є система, з якої не можна вилучити жодної функції без утрати її повноти; обов'язково в них має бути інверсія. Прикладом є системи з одного елемента І-НЕ чи АБО-НЕ (завдяки функціональній повноті елементи І-НЕ та АБО-НЕ називають *універсальними*). Пояснення щодо основ алгебри логіки а також арифметичні основи цифрової техніки, необхідні для розуміння логічних схем, викладено в [1].

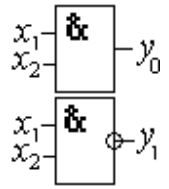
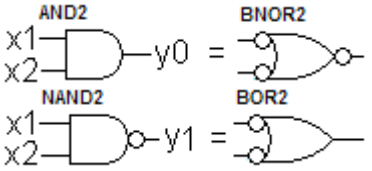
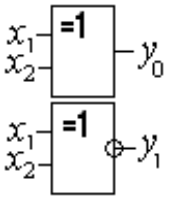
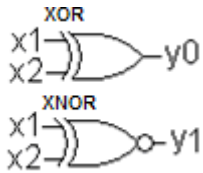
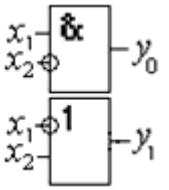
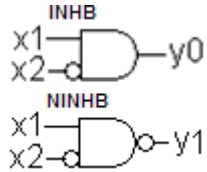
Основні співвідношення (табл. 1.2) розглядаються в булевій алгебрі, здебільшого, відносно функцій АБО та І, а справедливість тих чи інших формул відносно інших функцій з'ясовується окремо (у табл. 1.2 для прикладу наведено також співвідношення для функції Виключне АБО). Важливими серед них є закони двоїстості або де Моргана, узагальнення яких свідчить, що взаємна заміна в логічному виразі будь-якої функції y прямих і інверсних змінних та знаків логічного додавання і множення спричиняє її заперечення.

$$y = x_1\overline{x}_2 + x_3\overline{x}_4; \quad \overline{y} = (\overline{x}_1 + x_2)(\overline{x}_3 + x_4) \qquad (1.1)$$

Таблиця 1.1 – Основні логічні функції

Назва логічної функції (логічного елемента)		Таблиця відповід-ності		Позначення функції		Умовне графічне позначення	
Звичайна	САПР	$x_2\ x_1$	$y_0\ y_1$	Булеве	AHDL	ДЕСТУ	САПР
Повторення (повторювач, буфер) НЕ (елемент НЕ, інвертор)	Buffer NOT	0 1	0 1 1 0	$y_0 = x_1$ $y_1 = \overline{x_1}$	y0 = x1 y1 = !x1		

АБО (елемент АБО, диз'юнктор)
АБО-НЕ (елемент АБО-НЕ / Пірса)
	OR
	NOR
		0 0
		0 1
		1 0
		1 1
			0 1
			1 0
			1 0
			1 0
				$y_0 = x_1 + x_2$
				$y_1 = \overline{x_1 + x_2}$
					y0 = x1 # x2
					y1 = x1 !# x2
							

І (елемент І / збігу, кон'юнктор) І-НЕ (елемент І-НЕ / Шеффера)	AND NAND	0 0 0 1 1 0 1 1	0 1 0 1 0 1 1 0	$y_0 = x_1 x_2$ $y_1 = \overline{x_1 x_2}$	y0 = x1 & x2 y1 = x1 !& x2		
Виключне АБО (елемент нерівнозначності) Виключне АБО-НЕ (елемент рівнозначності)	XOR XNOR	0 0 0 1 1 0 1 1	0 1 1 0 1 0 0 1	$y_0 = x_1 \oplus x_2$ $y_1 = \overline{x_1 \oplus x_2}$	y0 = x1 $ x2 y1 = x1 !$ x2		
Заборона (елемент НІ) Імплікація (імплікатор)	(Inhb) (Ninhb)	0 0 0 1 1 0 1 1	0 1 1 0 0 1 0 1	$y_0 = x_1 \setminus x_2$ $y_1 = x_1 \rightarrow x_2$	y0 = x1 & !x2 y1 = !x1 # x2		

Таблиця 1.2 – Основи алгебри логіки

№	Співвідношення	АБО (OR) а)	І (AND) б)	Виключне АБО (XOR) в)
Аксіоми:				
1	Подвійне заперечення	$\overline{\overline{x}} = x$		–
2	Дозвіл	$x + 0 = x$	$x \cdot 1 = x$	$x \oplus 0 = x$
3	Блокування (інвертування)	$x + 1 = 1$	$x \cdot 0 = 0$	$(x \oplus 1 = \overline{x})$
4	Повторення	$x + x = x$	$x \cdot x = x$	$x \oplus x = 0$
5	Доповнення	$x + \overline{x} = 1$	$x \cdot \overline{x} = 0$	$x \oplus \overline{x} = 1$
Закони:				
1	Переставний	$x_1 + x_2 = x_2 + x_1$	$x_1 x_2 = x_2 x_1$	$x_1 \oplus x_2 = x_2 \oplus x_1$
2	Сполучний	$x_1 + x_2 + x_3 = = x_1 + (x_2 + x_3)$	$x_1 x_2 x_3 = = x_1 (x_2 x_3)$	$x_1 \oplus x_2 \oplus x_3 = = x_1 \oplus (x_2 \oplus x_3)$
3	Розподільчий	$x_1 (x_2 + x_3) = = x_1 x_2 + x_1 x_3$	$x_1 + x_2 x_3 = = (x_1 + x_2) + (x_1 + x_3)$	$x_1 (x_2 \oplus x_3) = = x_1 x_2 \oplus x_1 x_3$
4	Двоїстості (де Морґана)	$\overline{x_1 + x_2} = \overline{x_1}\,\overline{x_2}$ $\overline{x_1 + x_2} = \overline{x_1}\,\overline{x_2}$	$\overline{x_1 x_2} = \overline{x_1} + \overline{x_2}$ $\overline{x_1 x_2} = \overline{x_1} + \overline{x_2}$	–
Наслідки:				
1	Склеювання	$x_1 x_2 + x_1 \overline{x_2} = x_1$	$(x_1 + x_2)(x_1 + \overline{x_2}) = x_1$	$x_1 x_2 \oplus x_1 \overline{x_2} = x_1$
2	Поглинання	$x_1 + x_1 x_2 = x_1$	$x_1 (x_1 + x_2) = x_1$	$x_1 \oplus x_1 x_2 = x_1 \overline{x_2}$
3	Заступлення	$x_1 + \overline{x_1} x_2 = x_1 + x_2$	$x_1 (\overline{x_1} + x_2) = x_1 x_2$	$x_1 \oplus \overline{x_1} x_2 = x_1 + x_2$

На відміну від звичайної, у булевій алгебрі всі співвідношення симетричні відносно функцій логічного додавання та множення. Аби запобігти помилок, операції виконують у послідовності за пріоритетом: найстарший пріоритет має функція логічного заперечення (НЕ), за нею йде логічне множення (І, І-НЕ), нарешті, найнижчими є операції логічного додавання (АБО, АБО-НЕ) та додавання за модулем два (Виключне АБО, Виключне АБО-НЕ). У САПР на основі мов HDL запрограмовано виконання операцій за пріоритетом: 1) НЕ, 2) І, І-НЕ, 3) Виключне АБО, Виключне АБО-НЕ, 4) АБО, АБО-НЕ. Послідовність виконання дій можна змінити, як завжди, за допомогою дужок.

Добутки всіх літералів (змінних чи їх інверсій), при яких функція набуває значення логічної 1 (табл. 1.3), називають *мінтермами* (конституєнтами одиниці). Сума мінтермів зображає функцію в *досконалій диз'юнктивній нормальній* формі (ДДНФ). Диз'юнкція літералів, де функція перетворюється на нуль, називається *макстермом* (конституєнтою нуля), а добутком усіх макстермів функція зображається в *досконалій кон'юнктивній нормальній формі* (ДКНФ).

Будь-яка логічна функція зображається в ДДНФ або ДКНФ єдиним способом, тому ці форми є вихідними для подальшого аналізу й синтезу. Природно, досконалі форми можна зобразити і для інверсного значення функції (див. табл. 1.3.).

Таблиця 1.3 – Досконалі форми зображення функції

x_1	x_2	y	$\bar{y}$	Мінтерми M_i		Макстерми M_i'	
				y=1 за умови:	$\bar{y}$=1 за умови:	y=0 за умови:	$\bar{y}$=0 за умови:
0	0	0	1		$M_0=\bar{x}_1\bar{x}_2$	$M_0'=x_1+x_2$	
0	1	1	0	$M_1=\bar{x}_1x_2$			$M_1'=x_1+\bar{x}_2$
1	0	1	0	$M_2=x_1\bar{x}_2$			$M_2'=\bar{x}_1+x_2$
1	1	0	1		$M_3=x_1x_2$	$M_3'=\bar{x}_1+\bar{x}_2$	
Стандартні форми				ДДНФ		ДКНФ	
				$y=x_1\bar{x}_2+\bar{x}_1x_2$	$\bar{y}=x_1x_2+\bar{x}_1\bar{x}_2$	$y=(x_1+x_2)\times$ $\times(\bar{x}_1+\bar{x}_2)$	$\bar{y}=(x_1+\bar{x}_2)\times$ $\times(\bar{x}_1+x_2)$

1.1.2 Примітиви (Primitives)

Примітиви є функціональними модулями в САПР MAX+plus II, які репрезентують найпростіші типові елементи ЦП, а також допоміжні компоненти в графічних і текстових файлах проекту (термін Primitive охоплює базові логічні елементи, тригери, порти, константи та ін.).

Бібліотека всіх функціональних модулів САПР розташована в каталозі [ім’я диска]:\maxplus2\max2lib, а примітиви зосереджено в підкаталозі \prim. У довідці меню Help > Primitives репрезентовано всі п’ять груп, на які поділяються примітиви за призначенням: Input & Output Primitives/Ports (порти), Buffer Primitives (буфери), Logic Primitives (логічні елементи), Flipflop & Latch Primitives (тригери) та Other Primitives (інші примітиви, серед яких допоміжний Title Block – блок заголовка для оформлення документів.).

Метою цієї лабораторної роботи є розгляд примітивів перших трьох груп, основні елементи яких подано у файлі з директорією \max2work\tutorial\1libr.gdf:

1) **порти** (рис. 1.1, *а*): односпрямовані **INPUT** (введення), **OUTPUT** (виведення) і двоспрямований **BIDIR** (введення-виведення);

2) **буфери** (рис. 1.1, *б*): **GLOBAL** – для підвищення навантажівної здатності в розгалужених колах синхронізації, скидання, дозволу тощо; **TRI Primitive** – буфер із трьома станами виходу з високим рівнем дозволу Output Enable; **LCELL Primitive** (від logic cell – логічна комірка) – буфер для розподілення ресурсу мікросхеми між окремими її комірками (логічний синтезатор може автоматично запровадити буфер LCELL у схему, але

ніколи не вилучає його, якщо він вставлений користувачем); **EXP Primitive** (від expander – розширювач) – буфер з інверсним виходом для використання логічного терму сусідніми комірками того ж самого блоку логічних матриць (LAB) або комірками інших LAB (при використанні буферів LCELL та EXP як елементів затримки слід зважати вплив на її величину дестабілізівних чинників, насамперед температури середовища та нестабільності джерела живлення); **CARRY** – для організації швидких перенесень у логічному ланцюжку схем типу суматорів і лічильників; **SOFT** – програмний „м'який" примітив: логічний синтезатор сам з'ясовує, залишити буфер чи не застосовувати його в схемі; **WIRE Primitive** (від wire – дріт) – буфер фізично не існує і запроваджується до схеми штучно з метою змінити назву сигналу в різних її частинах;

3**) логічні примітиви** (рис. 1.1, *в*): зображення констант для графічних файлів (**GND** – логічний 0 і **VCC** – логічна 1); стандартні елементи (**NOT, OR, NOR, AND, NAND, XOR, XNOR, inhb**), а також деякі їх паралельні еквіваленти (**BOR, BNOR, BAND, BNAND**).

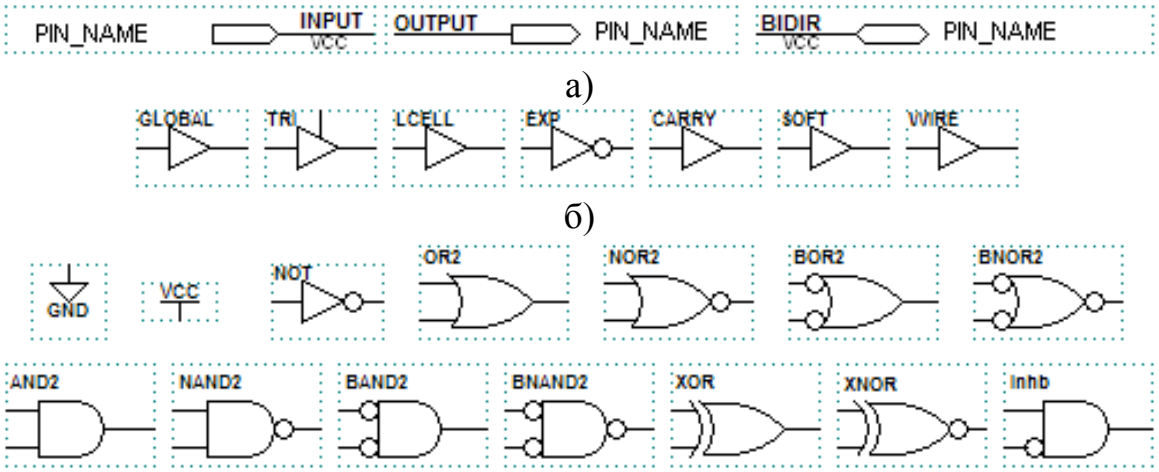
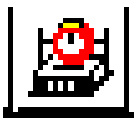


Рисунок 1.1 – Примітиви: а) порти; б) буфери; в) логічні примітиви

1.1.3 Часовий аналізатор



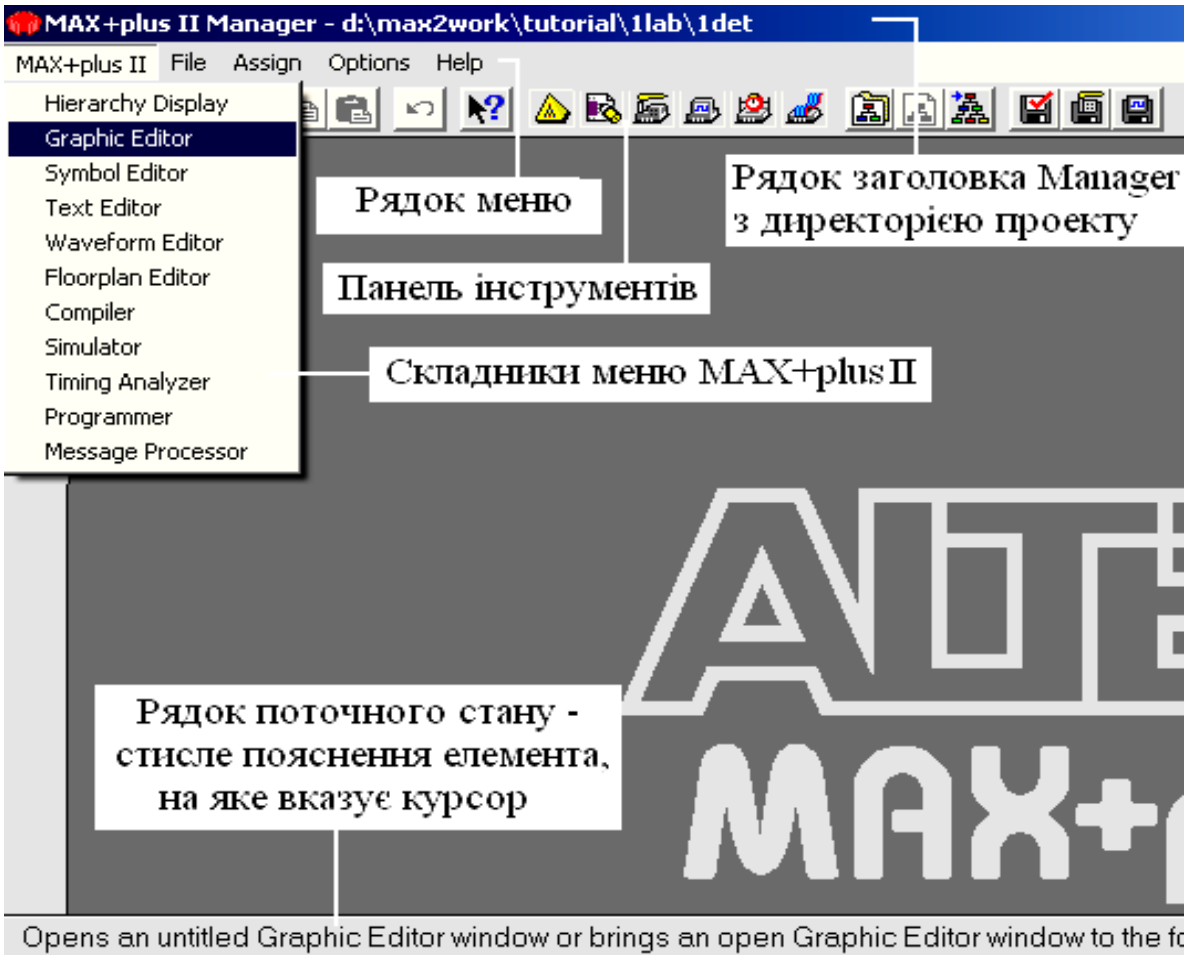
Timing Analyser – часовий аналізатор, призначений простежувати шляхи поширення сигналу, у тому числі критичні, що визначають швидкодію пристрою в проекті, синтезованому компілятором. Залежно від вибраного режиму результати аналізу можуть бути відображені трьома видами. У цій лабораторній роботі розглядається методика часового аналізування в режимі використання матриці затримок (Delay Matrix), в якій наводиться час поширення сигналу від множини вузлів, що вважаються джерелами, до вузлів-адресатів.

1.2 Лабораторне завдання

1.2.1 Основи керування файлами пакету MAX+plus II та підготовки матеріалів звіту

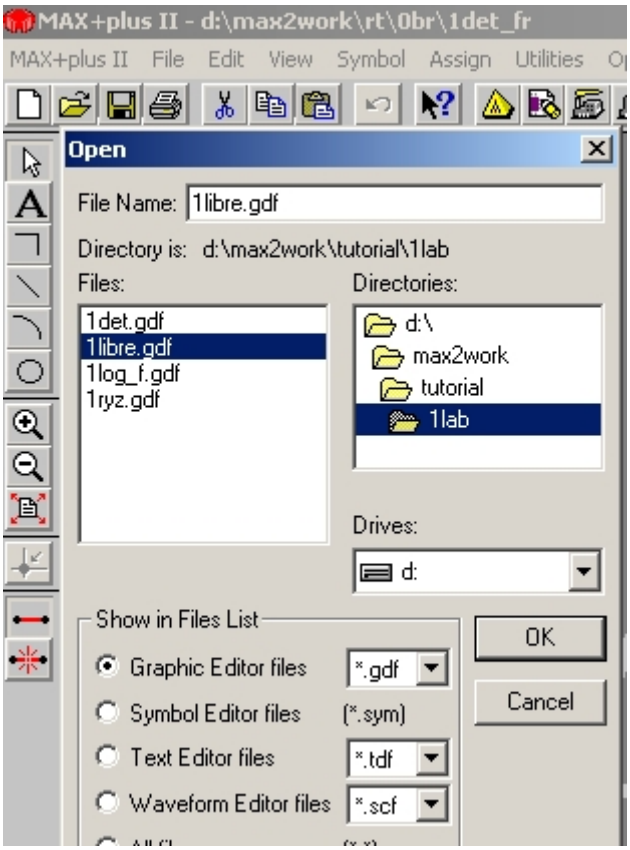
✍ ***Примітка.*** Послідовність дій, виконуваних (в основному вікні, діалогових вікнах, їх частинах, у списках, що розкриваються) натисканням лівої кнопки миші (Button 1), далі найчастіше позначатимемо для стислості знаком „>”, подвійне клацання цією кнопкою – знаком „B**”, а натискання правої кнопки миші (Button 2) – знаком „B2”.

1.2.1.1 Запустити програму MAX+plus II (ярликом на робочому столі або з меню Windows: Пуск > Все программы > MAX+plus II BASELINE > MAX+plus II BASELINE), ознайомитися з органами керування вікна Manager, складниками меню MAX+plus II і списками типових команд меню File та Help (див. Вступ).



1.2.1.2 Відкрити потрібний файл, наприклад, 1libr.gdf з директорії (повного каталогу) d:\max2work\tutorial\1lab\1libr.gdf:

а) піктограмою панелі інструментів Opens a file (відкрити файл) або з меню File > Open викликати діалогове вікно Open (відкрити);



б) у віконці Drives (диски) прокруткою ввести потрібне ім’я диска (наприклад, d:);

в) у віконці Directories подвійним клацанням по відповідних папках послідовно розкрити d:\, max2work, tutorial та 1lab;

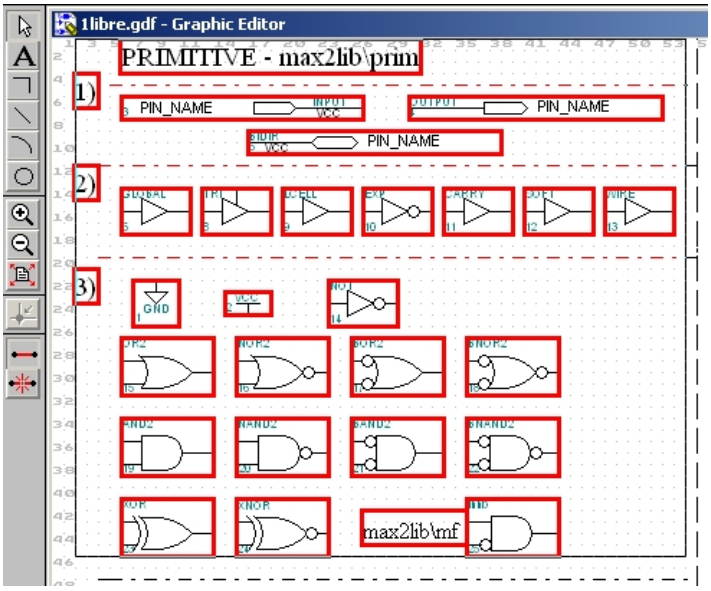
г) на вкладці Show in Files List (показ списку файлів) ввімкнути перемикач потрібного типу файлів, наприклад, Graphic Editor files (файли графічного редактора);

ґ) біля встановленого типу файлів прокруткою вибрати потрібне розширення, наприклад, *.gdf;

д) у віконці Files двічі клацнути на імені потрібного файла (або відмітити потрібний файл та натиснути ОК), який після цього з’явиться у головному вікні.

1.2.1.3 Переглянути і скопіювати графічний файл:

а) інструментом Changes the Display (відобразити на екрані весь файл) вертикальної панелі зліва (у вигляді аркуша файла з червоними стрілками в кутках) або з меню View (вигляд) > Fit in Window (вписати зображення у вікні) стиснути зображення, щоб мати уявлення про його розміри;

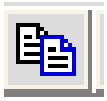


б) з меню View (вигляд) > Normal Size (нормальний розмір) повернутися до звичайного вигляду зображення;

в) інструментом Increases (збільшення – у вигляді лінзи зі знаком плюс) або з меню View (вигляд) > Zoom In збільшити зображення і продивитися потрібні деталі символів елементів, відтак інструментом Reduces (зменшення – у вигляді лінзи

зі знаком мінус) або з меню View (вигляд) > Zoom Out повернутися до нормального розміру;

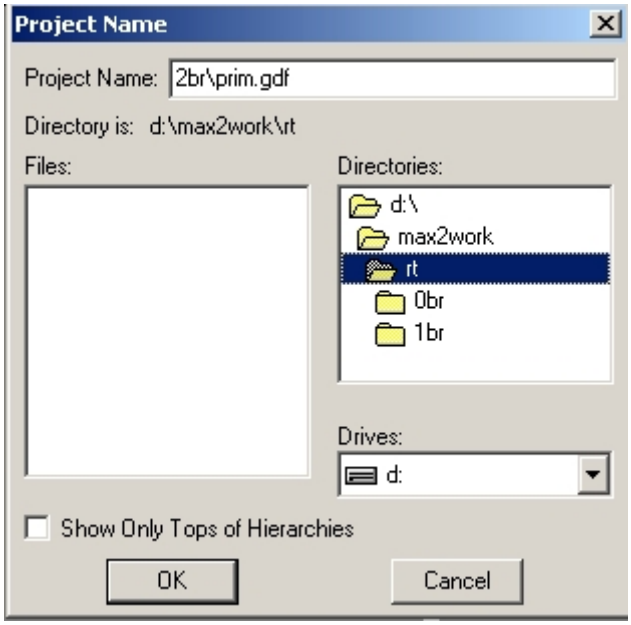
г) виділити потрібну ділянку для копіювання графічного файла: натиснути інструмент Selects (вибрати – перша стрілка зверху) і протягнути ним по діагоналі ділянки, утримуючи натиснутою ліву кнопку миші; після відпускання кнопки ділянка буде облямована чорною рамкою, а всі елементи всередині неї виділяться червоними рамками;



ґ) інструментом горизонтальної панелі Copies (копіювати) або з меню Edit > Copy скопіювати виділену ділянку в буфер.

1.2.1.4 Створити власний файл потрібного типу, наприклад, prim.gdf з директорією d:\max2work\rt\2br\prim.gdf:

а) піктограмою панелі інструментів Specifies the project name (визначити ім'я проекту) або з меню File > Project > Name викликати діалогове вікно Project Name (ім'я проекту);



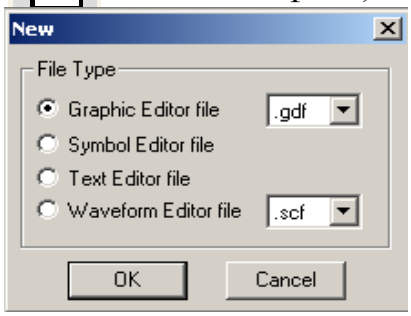
б) у віконці Drives (диски) прокруткою вибрати потрібне ім'я диска (наприклад, d:);

в) у віконці Directories подвійним клацанням по відповідних папках послідовно розкрити d:\, max2work та папку з назвою групи, наприклад, rt;

г) у віконці Project Name ввести номер бригади і через косу риску ім'я проекту та залишити потрібне розширення (все зайве стерти), наприклад, 2br\prim.gdf;

ґ) натиснути OK і, якщо з'явиться повідомлення, що такої директорії не існує, дати згоду на її створення; в результаті буде створена папка з номером бригади і в заголовку вікна Manager відобразиться повна директорія і назва проекту;

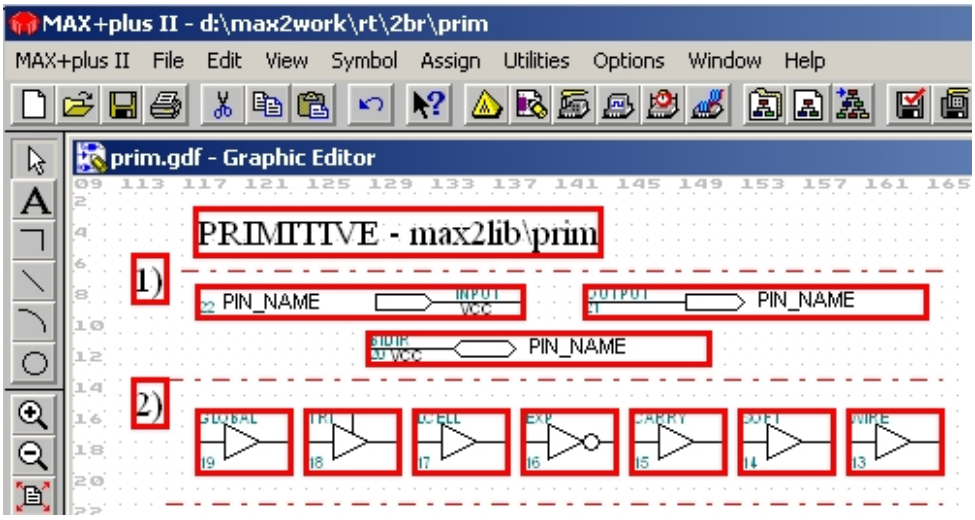
д) піктограмою панелі інструментів Opens a new file (створити новий файл) або з меню File > New викликати діалогове вікно New (новий файл);



е) вибрати тип файла: графічний (Graphic Editor file), символьний (Symbol Editor file), текстовий (Text Editor file) або часових діаграм (Waveform Editor file) та його розширення (прокруткою), наприклад, Graphic Editor file, .gdf > OK, після чого з'явиться вікно безіменного (Untitled) файла;

є) натисненням піктограми панелі інструментів Saves changes to the current file (зберегти зміни в поточному файлі) або з меню File > Save As викликається діалогове вікно Save As, після натиснення в якому ОК файлу автоматично надається ім'я проекту, зазначене в рядку заголовка Manager, із введеним у підпункті *е*) розширенням, наприклад, prim.gdf;

ж) у лівому верхньому кутку файла курсором клацнути на місці вставлення копії і піктограмою панелі інструментів Pastes a copy of the Clipboard (вставити копію з буфера) або з меню Edit > Paste перенести до файла зображення, скопійоване в пунктах 1.2.1.3, *г)*, *ґ)*, яке позиціонується лівим верхнім кутом у місці клацання;



з) повторити підпункти *а) – ж)* з метою скопіювати у власну папку для звіту всі потрібні графічні файли з лабораторної папки d:\max2work\1lab\1libre.gdf, 1log_f.gdf, 1ryz.gdf, 1det.gdf.

Примітки:

1. Копіювати файли для звіту можна під час виконання відповідних пунктів лабораторної роботи. Якщо папка вже існує, немає потреби створювати директорію в пп. 1.2.1.4, *б*), *в*), *г*); достатньо її вибрати та ввести лише ім'я проекту.

2. Тут подано швидке виконання операцій з файлами за допомогою піктограм інструментів або командами складників рядка меню, але деякі з них можна виконати також з контекстного меню, яке викликається кнопкою В2 (виконання операцій сполученням клавіш тут, здебільшого, не розглядається).

3. Наведені операції з файлами дають змогу створювати власний проект, частинами якого можуть бути копії фрагментів інших проектів. Але лише задля документування, як звичайно, достатньо скопіювати файли потрібного типу безпосередньо в папці „Мой компьютер" Windows з лабораторної до своєї папки.

1.2.1.5 Скопіювати у власну папку потрібний ***сигнальний файл*** (файл часових діаграм) або його частину з лабораторної папки:


а) якщо у власній папці не існує проекту з даною назвою (проекту верхнього рівня), наприклад, 1logfun, надати йому ім'я: піктограмою (п. 1.2.1.4, а) викликати діалогове вікно Project Name, у разі потреби виставити потрібну директорію, підняти прапорець Show Only Tops of Hierarchies (показати тільки вершини ієрархій), переконатися у відсутності проекту з даною назвою, ввести його ім'я (за власним вибором) до віконця Project Name та натиснути ОК, після чого це ім'я з'явиться в рядку заголовка Manager (треба бути уважними щодо назв: файли, наприклад, 1logfun.scf та 1log_f.gdf належать до різних проектів);


Якщо ж у власній папці є файл проекту з даною назвою (верхнього рівня), доцільно спочатку відкрити цей файл, наприклад, 1det.gdf, відтак натиснути піктограму Changes the project name to the name of the current file (надати проектові ім'я відповідно до імені поточного файла) або ввести команду File > Project > Set Project to Current File, після чого це ім'я з'явиться в рядку заголовка Manager;


б) піктограмою (п. 1.2.1.4, д), е) створити безіменний (Untitled) файл часових діаграм (Waveform Editor file) з розширенням .scf;


в) піктограмою (п. 1.2.1.4,є) зберегти цей файл з автоматичним наданням йому імені, виставленого в заголовку Manager, наприклад, 1logfun.scf;


г) піктограмою (п. 1.2.1.2) відкрити файл, з якого потрібно зняти копію, наприклад, d:\max2work\1lab\1logfun.scf, інструментом (п. 1.2.1.3, а) розташувати на екрані все зображення, прочитати інтервал часового моделювання (меню File > End Time) та його крок (меню Options > Grid Size);


ґ) натиснути клавішу Shift і клацнути по черзі в полі Name потрібні (всі або вибірково) часові діаграми, які виділяться чорним, та піктограмою скопіювати їх у буфер (п. 1.2.1.3, ґ);


д) повернутися до свого створеного файла, виставити такі самі інтервал часового моделювання (меню File > End Time) та його крок (меню Options > Grid Size), клацнути в поле Name та піктограмою вставити зображення з буфера (п. 1.2.1.4, ж);

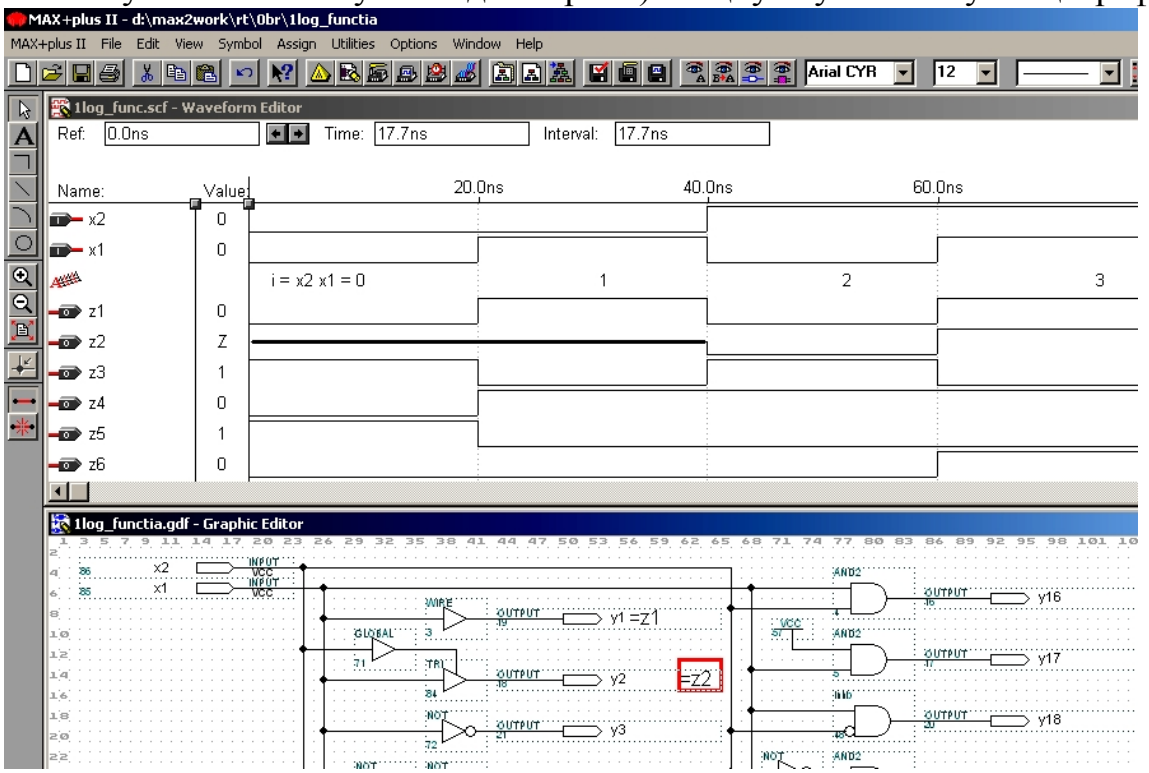
е) повторити підпункти *а) – д)* з метою копіювання у власну папку для звіту всі потрібні сигнальні файли з лабораторної папки d:\max2work\1lab\1logfun.scf, 1ryz.scf, 1det.scf.

☝ ***Примітка.*** Наведені операції з файлами дають змогу пришвидшити введення даних під час моделювання власного проекту шляхом використання фрагментів часових діаграм з інших проектів. Зауваження щодо документування зроблено в примітці 3 до п. 1.2.1.4.

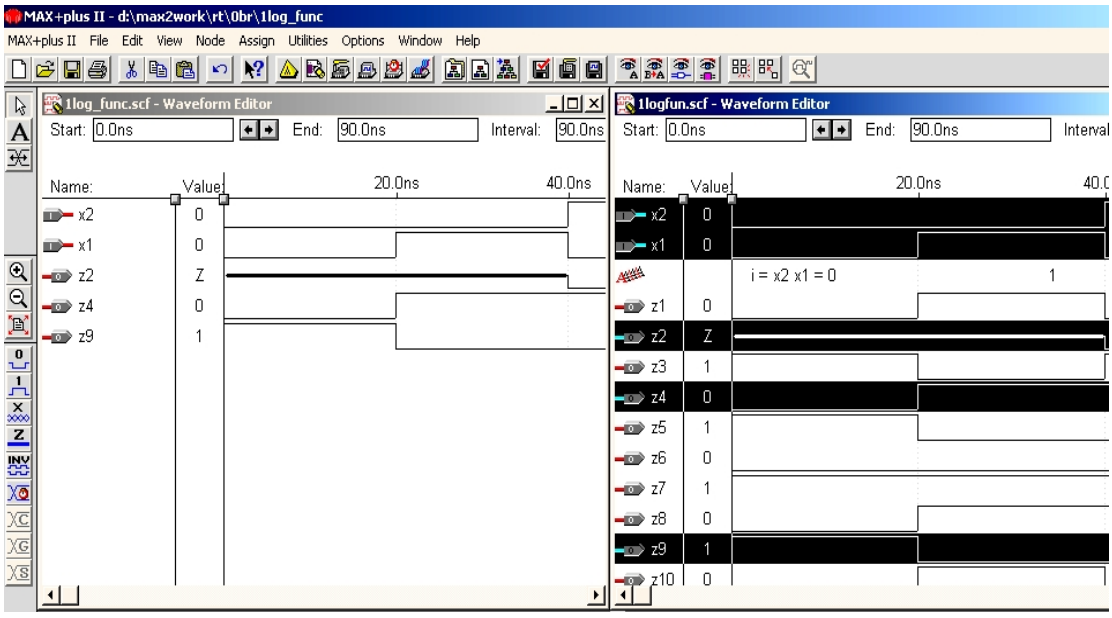
1.2.1.6 Відпрацювати деякі ***прийоми роботи з групою файлів.***

а) Для зіставлення змісту залишити на екрані два файли, наприклад, 1logfun.scf і 1log_f.gdf, відтак розгорнути їх до вікон зі смугами прокручу-

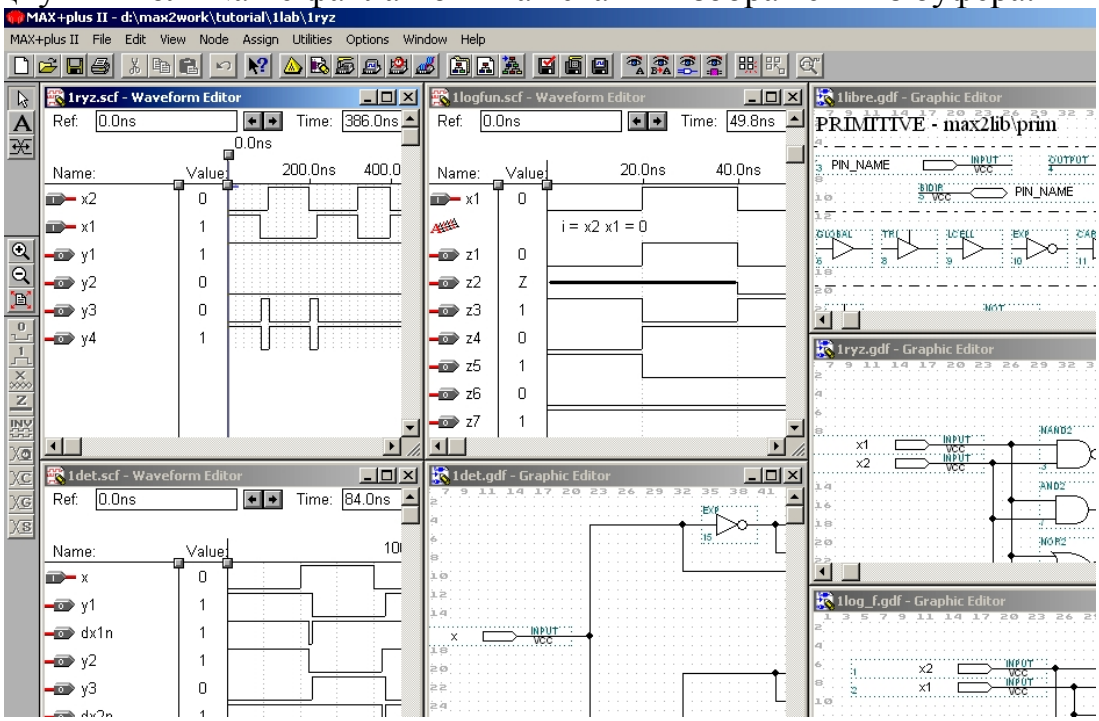
вання в кожному: меню Window > The Horizontally (або клавішами Shift+F4), смугами прокручування на графічному файлі знайти потрібну схему, наприклад, буфер з виходом y_1, прокручуванням часових діаграм на сигнальному файлі дібрати епюру z_i, що відповідає цьому виходу, інструментом вертикальної панелі Enters new text... (вставити новий або редагувати існуючий текст – у вигляді літери А) клацнути у вільному місці графічного файла і надрукувати напис „=z1”, відтак інструментом вибору (стрілка) виділити цей напис червоною рамкою і при утримуваній лівій кнопці миші перетягнути його до виходу схеми для утворення напису „y1=z1”; продовжуючи, зіставити в такий спосіб усі потрібні схеми з епюрами.



б) Для вибіркового копіювання залишити на екрані два файли, на-

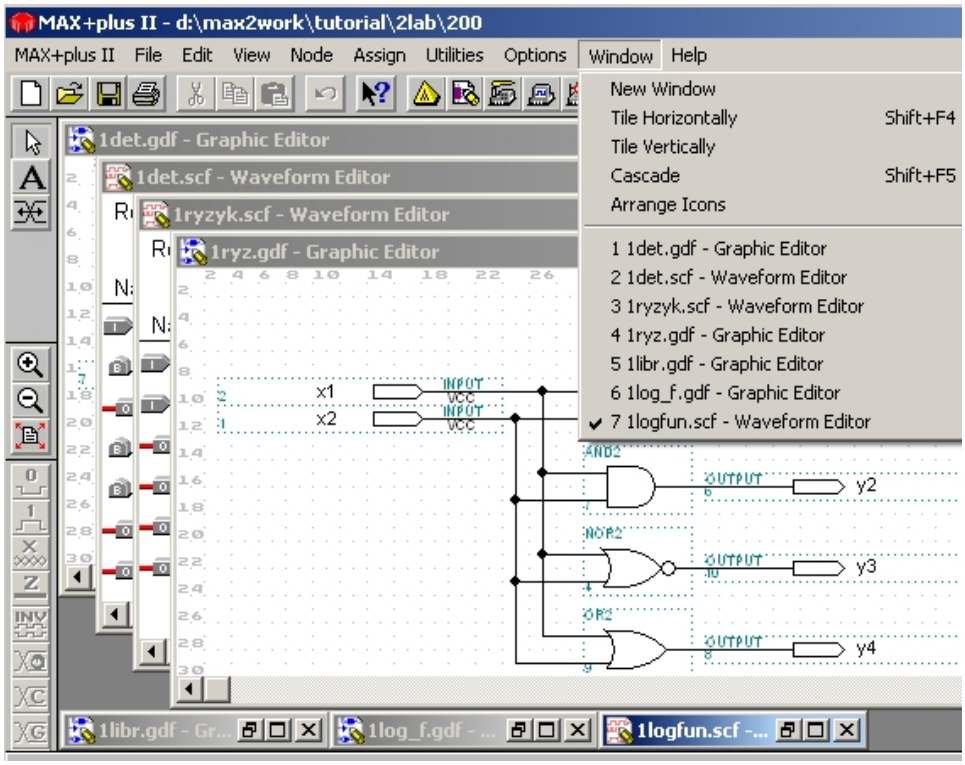


приклад, d:\max2work\rt\0br\1log_func.scf і d:\max2work\1lab\1logfun.scf, відтак розгорнути їх до вікон зі смугами прогону в кожному: меню Window > The Vertically (або клавішами Shift+F5), смугами прогону на файлі-оригіналі знайти потрібні епюри, виділити і скопіювати їх у буфер, відтак клацнути в полі Name файла-копії та вставити зображення з буфера.



в) Коли під час роботи над проектами відкрито багато файлів, до яких доводиться час від часу повертатися, доцільно їх впорядкувати.

Для попереднього перегляду розташувати файли у вигляді неперетинних вікон на екрані (див. рисунок зверху сторінки) командою з меню

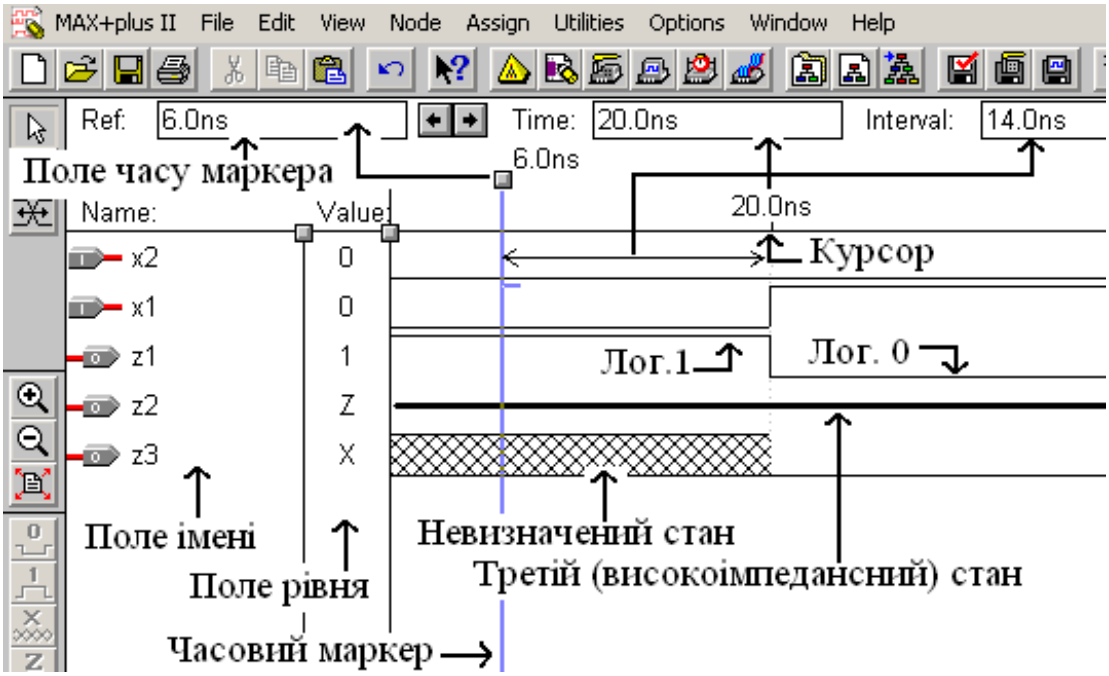


Window > The Horizontally або The Vertically. Відтак файли, не потрібні в першу чергу, згорнути до іконок внизу екрана клацанням символу згортання в рядку заголовку файла, а решту файлів розташувати каскадом з відображенням заголовків командою з меню Window > Cascade (див. рисунок внизу сторінки). Для переведення на передній план відкритого файла достатньо клацнути рядок його заголовка вгорі або нижній символ прокрутки зліва, або рядок з ім'ям 1, 2, 3 ... 9 зі списку в меню Window. Якщо двічі клацнути на іконці згорнутого файла, або один раз клацнути на іконці розгортання чи рядок з ім'ям файла у згаданому списку, файл розгортається до такого вікна (в каскаді чи повного), з якого він був згорнутий.

Крім того, завжди можна повернутися до проектного файла верхнього рівня (тобто до файла, з якого утворено проект, зазначений у рядку заголовка Manager) піктограмою Opens the top-level design file (відкрити проектний файл верхнього рівня). Після натиснення її цей файл або відкриється заново, або пересунеться з каскаду відкритих файлів на передній план.

1.2.2 Дослідити логічні елементи

1.2.2.1 За ідеалізованими часовими діаграмами (без урахування затримок, файл 1logfun.scf) ***побудувати таблиці відповідності*** функцій $z_1 \ldots z_{14}$ і записати вирази $z_1 \ldots z_{14} = f(x_1, x_2)$ у булевому базисі. Під час побудови таблиці зручно користуватися часовим маркером – лінією блакитного кольору. Для цього слід перетягнути вузол (квадратик вгорі) маркера при натиснутій лівій кнопці в довільне положення на першій часовій ділянці діаграм і зчитати в полі рівня (Value) значення сигналів, поданих в полі імені (Name). Відтак перетягнути маркер на другу ділянку і так само зчитати рівні і т. д. до заповнення всієї таблиці відповідності.



✍ ***Примітки:***

1. Переривчастим клацанням по стрілках біля поля часу маркера (Ref) він переводиться до стрибка рівнів на діаграмах.

2. Ширину поля імені і поля рівня можна змінити перетягуванням відповідного вузла (квадратика між полями).

3. Аби коментарі (рядок з червоною лінією А) були правильно розташовані відносно часових епюр, слід спочатку розгорнути файл на весь екран, а відтак, за необхідності, згорнути у вікно (тобто на частину екрану).

1.2.2.2 Визначити еквівалентну логічну функцію, виконувану кожною схемою на примітивах (файл 1log_f.gdf). На копії схеми у власному файлі навести номери часових діаграм (файл 1logfun.scf), що відповідають кожному виходу, наприклад, y1 = z1, y2 = z2 тощо.

1.2.2.3 Спостерігати на виході основних логічних елементів (файл 1ryz.gdf) ***виникнення ризиків*** у вигляді паразитних імпульсів внаслідок небезпечних змагань сигналів (файл 1ryzyk.scf); дані подати фрагментами ідеалізованих (без урахування затримок) осцилограм вхідних і вихідних сигналів у місцях виникнення ризиків зі стислим поясненням.

1.2.2.4 Ознайомитися з дібраними елементами бібліотеки примітивів (файл 1libr.gdf): 1) портами, 2) буферами, 3) логічними елементами, а також прикладами їх мікросхемного виконання на ІС серії 74. Дати тлумачення сенсу кожного позначеного символами примітива. Отримати інформацію з довідки для кожного примітива.

1.2.3 Засвоїти основи *часового аналізу* на прикладі дослідження формувального кола на логічних елементах

1.2.3.1 За схемою і часовими діаграмами (файли 1det.gdf, .scf) ***розглянути принцип побудови різницевого елемента*** (РЕ) на логічному елементі І-НЕ і елементі затримки з інверсією (буфер EXP) та застосування його у колах формування коротких імпульсів – детекторах фронтів вхідних імпульсів *x*: за позитивним (dx1n), негативним (dx2n) їх перепадом та за обома перепадами (dx3).

✍ ***Примітки*:**

1. У класичному детекторі фронтів на ІС жорсткої структури для затримки та інверсії застосовують ланцюжок з непарної кількості логічних елементів з інверсними виходами. Щодо алгебри логіки це не має сенсу, тому компілятор за аксіомою подвійного заперечення усуває зі схеми „зайві” ланцюжки елементів, отже, і затримку. Аби запобігти цьому, в ІС програмованої структури застосовано буфер EXP.

2. На часових діаграмах у полі імені наведено три типи вузлів: вхідні порти з позначкою I (Input), вихідні O (Output) та внутрішні вузли B

(Buried – внутрішній, прихований), не сполучені безпосередньо із зовнішніми контактами („ніжками”) ІС. Внутрішні вузли подано задля наочності з’ясування виникнення коротких імпульсів.

1.2.3.2 Виміряти затримки поширення сигналів та тривалості вихідних імпульсів ручним способом. Для цього переривчастим клацанням по стрілках біля поля часу маркера (див. рисунок до п. 1.2.2.1) переводимо останній до потрібного перепаду сигналу і відлічуємо час у згаданому полі.

***Примітки*:**

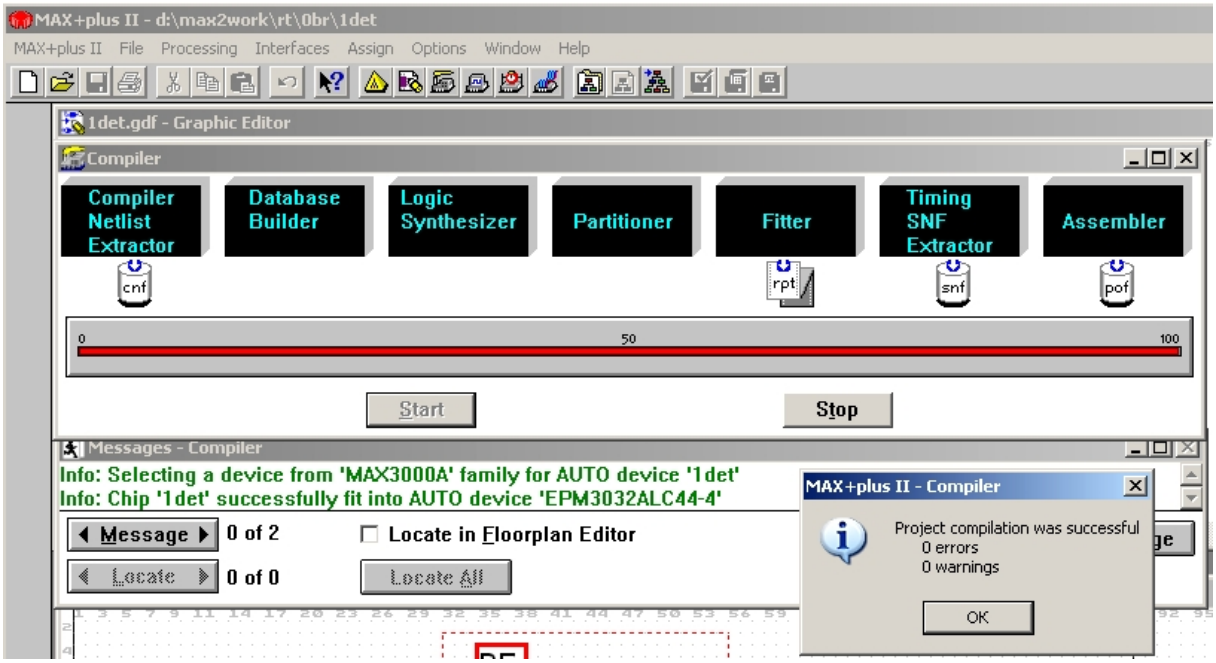
1. Для вимірювань можна користуватися також курсором (інструментом вибору у вигляді стрілки на вертикальній палітрі): індикатори Time та Interval відзначають часове положення стрілки і віддаль між нею і маркером.

2. Подвійним клацанням по індикаторному полю Ref викликається віконце, яке дозволяє перевести маркер в будь-яку позицію на часовій сітці. Для переведення маркера в довільну позицію незалежно від кроку сітки слід зняти прапорець Snap to Grid (прив’язка до сітки) в меню Options.

1.2.3.3 Виміряти затримки поширення сигналів від вхідних до вихідних виводів мікросхеми часовим аналізатором у режимі матриці затримок.

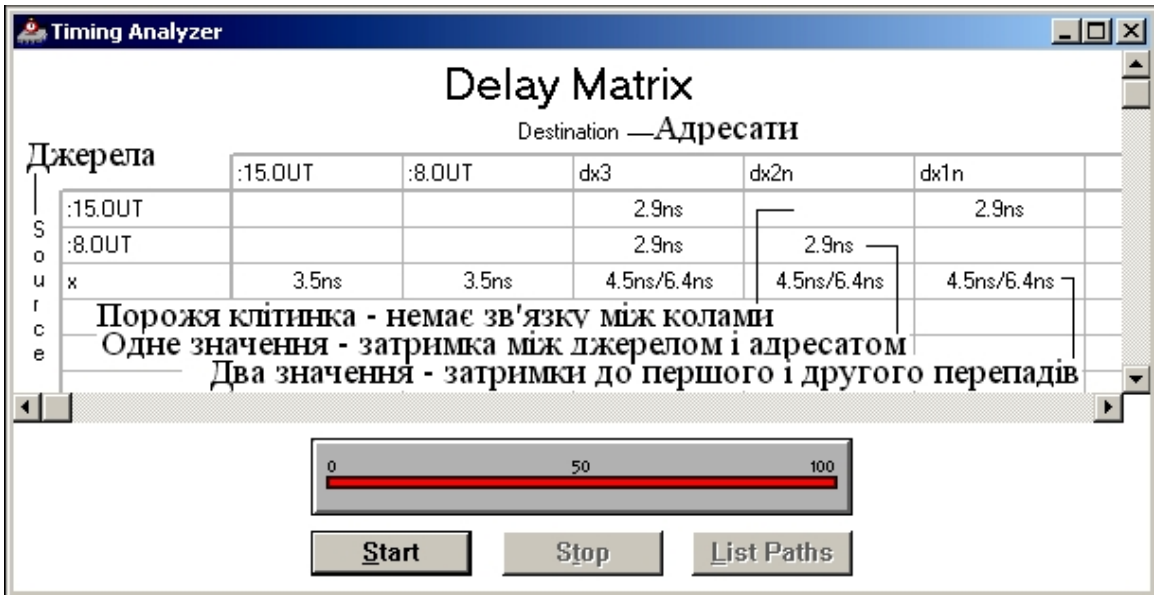
а) Відкрити графічний файл (наочніше) або файл часових діаграм власної копії проекту 1det, директорія та ім’я якого позначено в рядку заголовка Manager, інакше це ім’я треба ввести піктограмою або командою File > Project > Set Project to Current File.

б) Піктограмою Saves … and starts the Compiler (зберегти зміни у файлі і запустити компілятор) або командою File > Project > Save & Compile скомпілювати проект, натиснути ОК в інформаційному віконці про успішну (successful) компіляцію та закрити вікно компілятора.

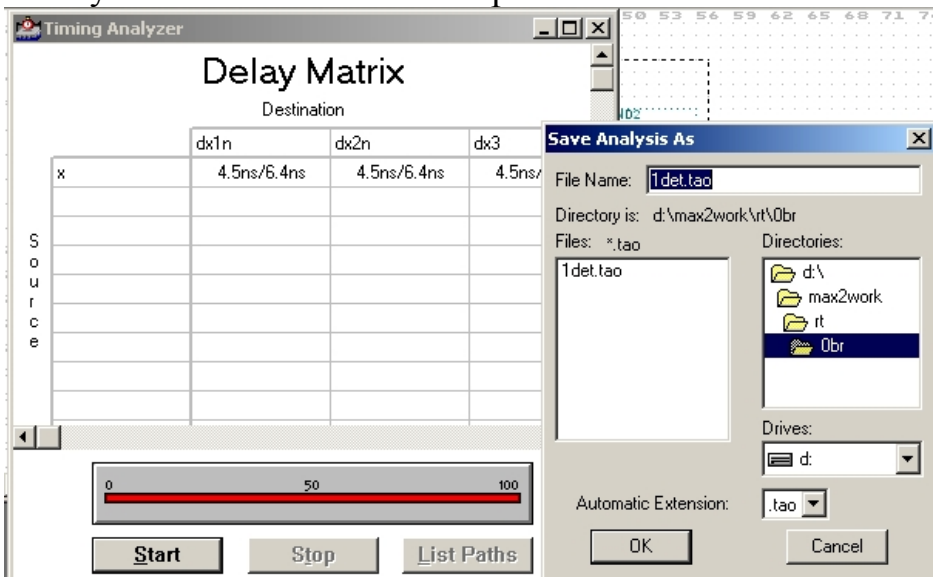


в) Викликати часовий аналізатор піктограмою Opens the Timing Analyzer (відкрити часовий аналізатор) або з меню MAX+plus II > Timing Analyzer та, якщо раніше не було вибрано тип часового аналізу за допомогою матриці затримок, установити його з меню Analysis > Delay Matrix.

г) запустити часовий аналізатор натисненням кнопки Start у вікні аналізатора, відтак натиснути OK у віконці про закінчення (completed) аналізу, після чого матриця виявляється заповненою. Відтак нею можна скористатися для визначення шляхів поширення сигналів, тривалості сформованих імпульсів тощо.



✍ ***Примітка.*** З огляду на те, що матриця затримок (у первісному вигляді) не зберігається, її легко відновити: ввести ім'я проекту активного файла та запустити часовий аналізатор.



ґ) Зберегти результати аналізу у формі таблиці текстового файла з розширенням **.tao** (Timing Analyzer Output File – вихідний файл часового аналізатора): при активному вікні Timing Analyzer із заповненою таблицею

(інакше відновити її як зазначено в примітці) натиснути піктограму збереження файла (або з меню File > Save Analysis As) та натиснути OK у вікнонці Save Analysis As.

д) Відкрити текстовий файл із таблицею затримок: натиснути піктограму відкриття файла Opens a file (або з меню File > Open) > Text Editor files > .tao > [ім'я проекту]. tao > OK (або B** по назві файла у віконці Files). Таблиця затримок (1det.tao) набуде вигляду на зразок наведеного фрагменту.

DELAY MATRIX *Матриця затримок*

SOURCE *Джерело*	DESTINATION *Адресат*	MIN_PATH	MAX_PATH *Шлях*
x	dx1n	4.5ns	6.4ns
x	dx2n	4.5ns	6.4ns
x	dx3	4.5ns	6.4ns

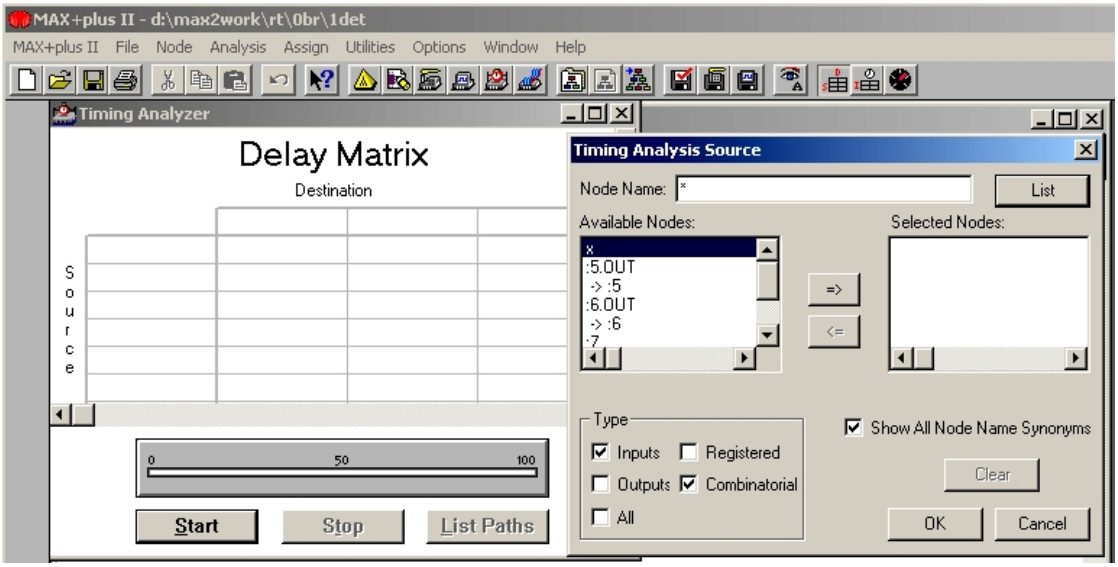
Примітки:

1. Якщо файл з таким ім'ям вже існує, надійде запит: перезаписати файл (наприклад, з метою виправити його) чи додати до існуючого.

2. Файл можна записати і за іншою директорією (наприклад, з метою документувати у власну папку).

1.2.3.4 Виміряти затримки поширення сигналів часовим аналізатором з урахуванням як зовнішніх виводів мікросхеми, так і внутрішніх вузлів.

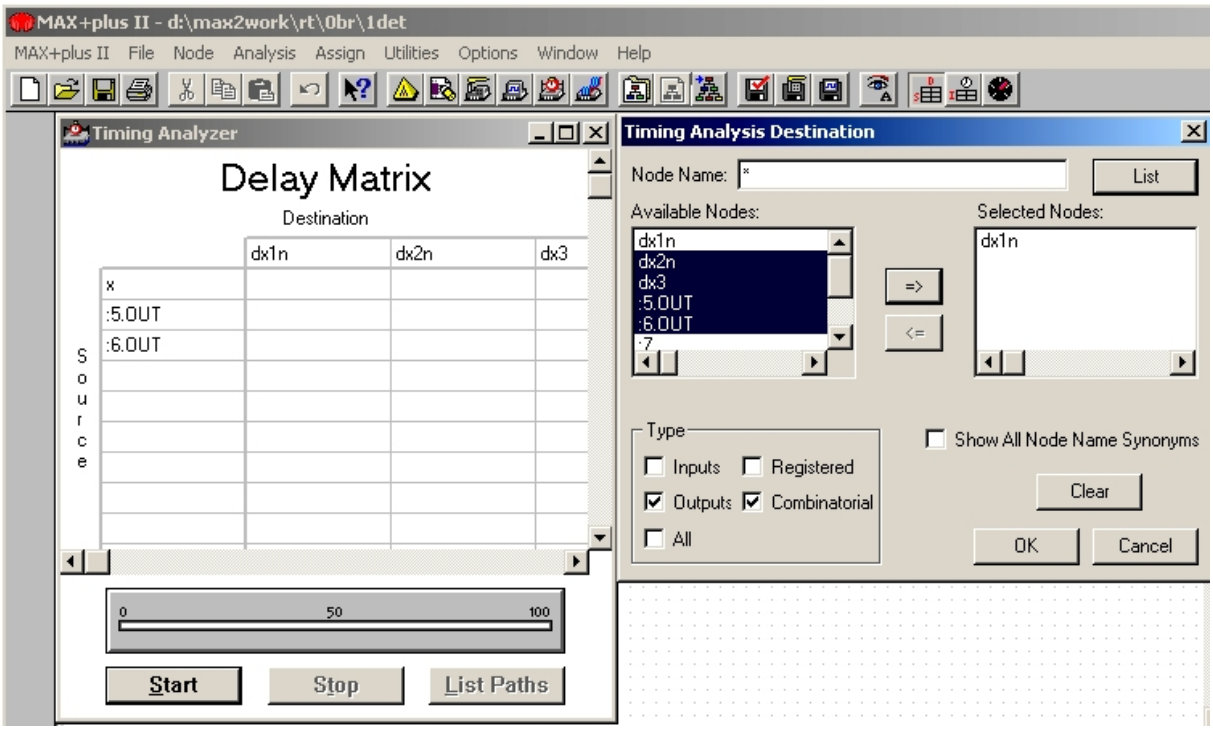
а) Після виконання п. 1.2.3.3, а), б), в) внести до матриці Delay Matrix джерела (Source) – вузли, від яких потрібно почати відлічувати затримку поширення сигналів: меню Node > Timing Analysis Source > Type (у цій закладці діалогового вікна Timing Analysis Source вибрати тип вузлів, наприклад, Inputs – входи і Combinational – комбінаційні кола) > установити прапорець Show All Node Name Synonyms, якщо потрібно продивитися всі доступні з вибраних джерел > Clear для очищення закладки Selected Nodes



> List для запису доступних вузлів до закладки Available Nodes > вибрати вузол у віконці Available Nodes > натиснути стрілку => для перенесення цього вузла до віконця Selected Nodes > вибрати всі потрібні джерела повторенням двох останніх дій > OK (або Cancel, якщо треба скасувати зроблений вибір вузлів і вийти з діалогового вікна).

✍ ***Примітка.*** Аби надмірно не ускладнювати матрицю, доцільно обмежитися мінімумом вузлів, достатнім для досягнення мети аналізу. Наприклад, для аналізу детектора фронтів варто вибрати такі джерела: вхід *x* та виходи буферів EXP (:5.OUT, :6.OUT) з ідентифікаційними номерами 5 і 6.

б) Внести до матриці Delay Matrix адресати (Destination) – вузли, на яких закінчується відлік затримки поширення сигналів відносно вибраних джерел: меню Node > Timing Analysis Destination, відтак вибрати тип вузлів, наприклад, Outputs (виходи) і Combinational та виконати такі самі дії, як у п. 1.2.3.4, а). Наприклад, можна обмежитися виходами трьох детекторів фронтів і двома внутрішніми вузлами – виходами буферів EXP.



✍ ***Примітка.*** Внести до матриці Delay Matrix джерела Source та адресати Destination можна й іншими шляхами:

а) з контекстного меню у графічному файлі: натиснути В2 по порту чи іншому елементу (виділиться червоним кольором) > вибрати в контекстному меню Timing Analysis > Source – для джерела або Destination – для адресата, або Cutoff (зупинення) – для вузла, який не треба враховувати в матриці; продовжуючи, визначити таким чином всі джерела та адресати **або:**

б) у графічному файлі: ввести ім’я проекту > виділити порт чи інший елемент > меню Utilities > Timing Analysis Source – для джерела, Tim-

ing Analysis Destination – для адресата або Timing Analysis Cutoff – для вузла, який не треба враховувати в матриці; продовжуючи, визначити таким чином всі джерела та адресати.

в) Виконати п. 1.2.3.3, г), ґ), д), але матрицю у формі таблиці (з розширенням **.tao**) зберегти під іншим іменем, наприклад, 1det1.tao або 1det.tao1.

Контрольні питання та завдання

1. Які системи логічних функцій є функціонально повними? Які з них є мінімально повними? Доведіть, що мінімально повну систему утворюють: 1) операція заборони і константа одиниці, 2) операція імплікації і константа нуля.

2. Логічну функцію *y* здійснити на заданому елементі з використанням інверторів (далі наводиться варіант: функція – заданий елемент):

1) $y = \overline{x_1 x_2 \overline{x_3} + x_4 + x_5 x_6 x_7 x_8 x_9} - I$;
2) $y = \overline{\overline{x_1 x_2 x_3} + x_4 + x_5} - АБО$;

3) $y = x_1 \oplus \overline{x_2} \oplus \overline{x_3} \oplus x_4 \oplus \overline{x_5} - Викл.АБО$;
4) $y = \overline{\overline{x_1} x_2} - Заборона$;

5) $y = \overline{\overline{x_1 + \overline{x_2}}\ \overline{x_3 + \overline{x_4}}} - I - АБО$;
6) $y = \overline{x} - I - НЕ$;

7) $y = x_1 + \overline{x_2} + \overline{x_3} + x_4 + \overline{x_5} + x_6 - I - НЕ$;
8) $y = x_1 \overline{x_2}\ \overline{x_3}\ x_4 \overline{x_5} - АБО - НЕ$;

9) $y = x_1 \oplus \overline{x_2} \oplus \overline{x_3} \oplus x_4 - Викл.АБО - НЕ$;
10) $y = x_1 + x_2 - Заборона$;

11) $y = \overline{x_1 + \overline{x_2} + \overline{x_3}\ x_4 + \overline{x_5}} - I - АБО - НЕ$;
12) $y = \overline{\overline{x_1 x_2}\ x_3\ x_4\ x_5} - I - НЕ$;

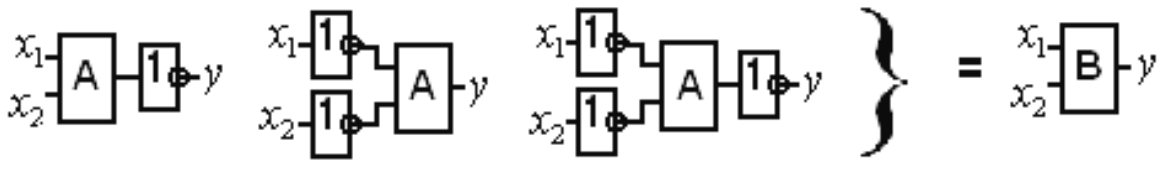
13) $y = \overline{\overline{x_1 + x_2} + x_3 + x_4 + x_5} - АБО - НЕ$;

14) $y = \overline{\overline{x_1 \oplus x_2} \oplus x_3 \oplus x_4 \oplus x_5} - Викл.АБО$.

15) $y = (x_1 + \overline{x_2})(x_3 + \overline{x_4}) x_5 \overline{x_6} - I - АБО - НЕ$;

16) $y = \overline{x_1 + \overline{x_2} + \overline{x_3} + \overline{x_4}}\ \overline{x_5 + \overline{x_6} + x_7}\ \overline{x_8 + \overline{x_9}}\ \overline{x_{10} + \overline{x_{11}}} - I - АБО - НЕ$.

3. Який логічний елемент *B* (рис. 1.2, г) утворюється додаванням інверторів відповідно до рис. 1.2, а), б), в), якщо *A* є елемент: 1) АБО, 2) І, 3) Виключне АБО, 4) Заборона, 5) АБО-НЕ, 6) І-НЕ, 7) Виключне АБО-НЕ, 8) Імплікатор?



а) б) в) г)

Рисунок 1.2

2 РЕАЛІЗАЦІЯ ЛОГІЧНИХ ФУНКЦІЙ

Мета роботи: створення проекту за його графічного введення; засвоєння методики користування електронним довідником для добору потрібної ІС серії 74 і методики настроювання макрофункцій.

Домашнє завдання

1) Засвоїти теоретичні відомості щодо мінімізації логічних функцій, реалізації їх у поширених базисах та способів спрощення логічних схем.

2) Для заданої згідно з варіантом (див. додаток А, варіанти завдання 2) логічної функції *y*: *а*) побудувати таблицю відповідності і діаграму термів (Вайча-Карно); *б*) навести досконалі (ДДНФ або ДКНФ) та мінімальні (МДНФ і МКНФ) форми; *в*) перетворити мінімальні форми до базисів І-НЕ, АБО-НЕ та І-АБО-НЕ; *г*) мінімізувати схему в базисі І-НЕ чи АБО-НЕ, який забезпечує меншу складність; *ґ*) навести схеми за підпунктами ***в), г)***.

2.1 Стислі теоретичні відомості

2.1.1 Мінімізація логічних функцій

Мінімізація логічних функцій полягає в перетворенні ДДНФ до мінімальної ДНФ (МДНФ) або ДКНФ до мінімальної КНФ (МКНФ) функції. Серед формалізованих методів мінімізації найбільшого поширення набув метод за допомогою *діаграм термів* (Вайча – Карно). Для функції *y* чотирьох змінних (рис. 2.1, *а*) клітинки діаграми термів (рис. 2.1, *б*) нумеруються десятковими кодами *i* вхідних кортежів. Стандартна нумерація клітинок полегшує заповнення власне діаграми функції *y* (рис. 2.1, *в*): до діаграми переносять лише одиниці з колонки *y* таблиці відповідності згідно з кодами *i* вхідних кортежів, а до порожніх клітинок вважаються записаними нулі, або, навпаки, переносять лише нулі (рис. 2.1, *г*), тоді порожні клітинки відповідають значенням $y = 1$.

Підсумовуванням мінтермів (сполучень одиниць) дістаємо шукану МДНФ у вигляді

$$y = x_1 x_3 + \overline{x_2}\,\overline{x_3}\,\overline{x_4} + \overline{x_1}\,\overline{x_3}\,x_4.$$

Цю ж саму діаграму термів можна використати й для здобуття інверсної функції КНФ

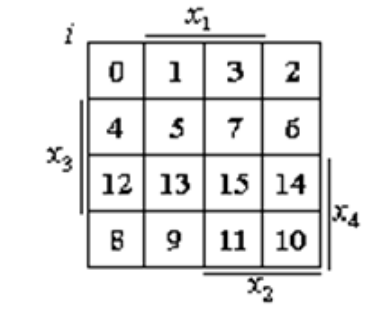
$$\overline{y} = \left(\overline{x_1} + \overline{x_3}\right)\cdot\left(x_2 + x_3 + x_4\right)\cdot\left(x_1 + x_3 + \overline{x_4}\right) .$$

Підсумовуванням макстермів (сполучень нулів) дістаємо шукану МДНФ у вигляді

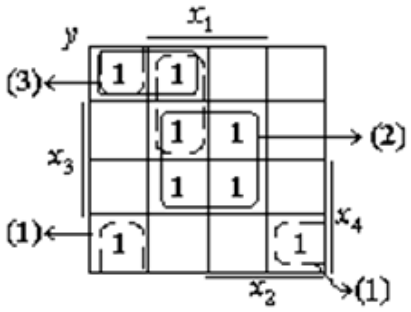
$$y = \left(x_1 + \overline{x_3}\right)\left(\overline{x_2} + x_3 + x_4\right)\left(\overline{x_1} + x_3 + \overline{x_4}\right) .$$

i	x_4	x_3	x_2	x_1	y
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

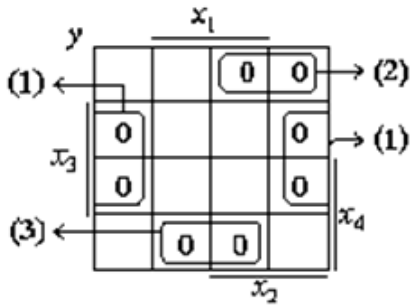
а)



б)



в)



г)

Рисунок 2.1 – Мінімізація логічної функції y чотирьох змінних

Інверсну функцію МДНФ також можна отримати з цієї самої діаграми, зчитуючи сполучення нулів так само, як сполучення одиниць:

$$\overline{y} = \overline{x_1}x_3 + x_2\overline{x_3}\,\overline{x_4} + x_1\overline{x_3}x_4.$$

2.1.2 Реалізація в поширених базисах

Булів базис – функції реалізуються сполученням входів і виходів логічних елементів у послідовності: НЕ, І, АБО відповідно до формул у МДНФ або в послідовності: НЕ, АБО, І за формулами в МКНФ.

Базис І-НЕ – після мінімізації логічної функції в МДНФ (об'єднанням одиниць) реалізацію звичайно виконують за формулою:

$$y = A + B = \overline{\overline{A + B}} = \overline{\overline{A} \cdot \overline{B}}, \qquad (2.1)$$

де A, B – кон'юнкції літералів (змінних або їх заперечень).

Базис АБО-НЕ –після мінімізації логічної функції в МДНФ (об'єднанням нулів) реалізацію звичайно виконують за формулою:

$$y = A \cdot B = \overline{\overline{A \cdot B}} = \overline{\overline{A} + \overline{B}}\,, \qquad (2.2)$$

де A, B – диз'юнкції літералів.

Базис І-АБО-НЕ – після мінімізації інверсної логічної функції в МДНФ (об'єднанням нулів) перетворення схем виконують алгебрично за формулами:

$$\overline{y} = AB + CE; \quad y = \overline{AB + CE}\,, \qquad (2.3)$$

де A, B – кон'юнкції літералів.

2.1.3 Способи спрощення логічних схем

Серії сучасних ІС містять різні комплекти стандартних логічних елементів, у тому числі багатовходових, використання яких дозволяє *мінімізувати схеми*.

Редукцію в диз'юнктивній формі *в загальному вигляді* під час мінімізації в ДНФ можна виконати доповненням заданої функції y неперетинною з нею коригувальною функцією y_k таким чином, аби об'єднання D цих функцій та сама допоміжна функція y_k були якомога простішими. Тоді шукану функцію дістаємо з виразу

$$y = D \setminus y_k = D\overline{y_k}\,. \qquad (2.4)$$

Редукція в кон'юнктивній формі – до нульових клітинок функції y з метою спрощення їх сполучень на діаграмі долучаємо ще деякі одиничні клітинки, утворюючи допоміжну функцію y_k, якщо доповнення об'єднання $\overline{D}$ цих функцій (тобто сполучення нульових і ще приєднаних одиничних клітинок) і нова функція y_k виявляються простими. Для здобуття шуканої функції необхідно в утвореному об'єднанні відновити одиничні клітинки, тобто виконати операцію імплікації

$$y = \overline{D} \leftarrow y_k = \overline{D} + \overline{y_k}. \qquad (2.5)$$

Алгебричне виконання редукції у простих випадках можна здійснити за формулами:

$$A\overline{B} = A\,\overline{AB}\,; \quad A + \overline{B} = A + \overline{A + B}\,. \qquad (2.6)$$

Приклади та інші відомості щодо мінімізації логічних функцій, реалізації їх у поширених базисах і схемної мінімізації наведено в [1, 2].

2.1.4 Макрофункції

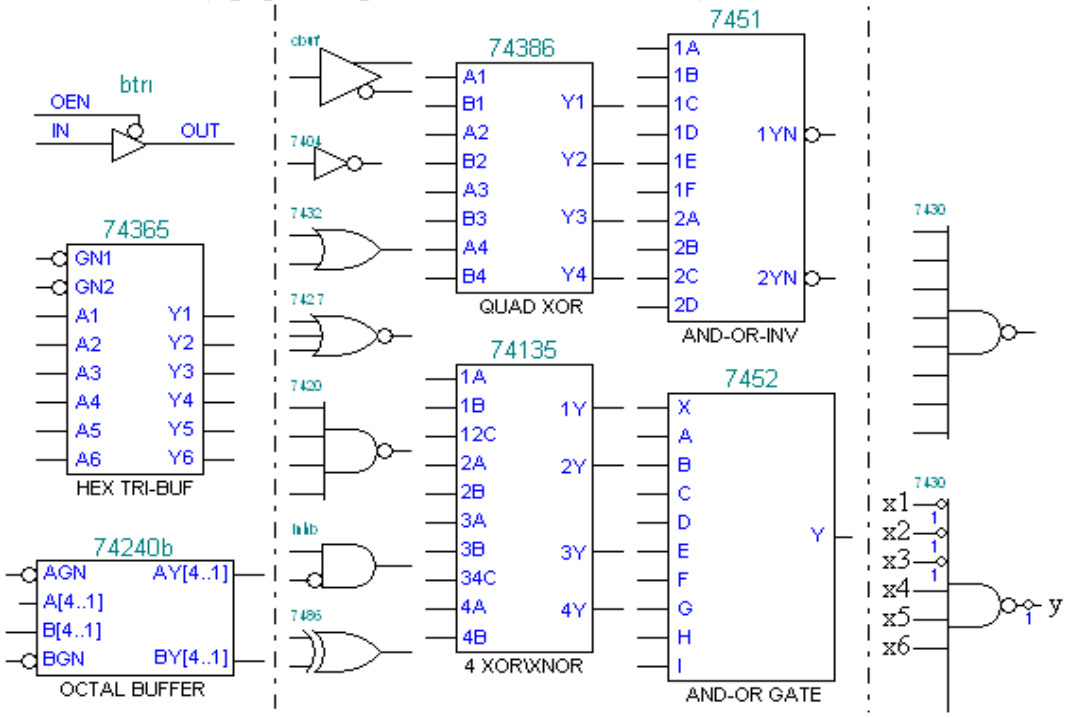
Макрофункція (Macrofunction) – стандартний блок, здебільшого високого рівня, з оптимізованими різноманітними логічними операціями. У системі MAX... запроваджено понад 300 макрофункцій на базі 74-ої серії (з цифровою нумерацією 74...), а також специфічних для цієї програми (позначаються маленькими літерами). Макрофункції можуть використовуватися в графічних і текстових файлах разом із примітивами та іншими структурними компонентами. З використанням макрофункцій зручно відпрацьо-

вувати проекти, що будуються на ІС жорсткої структури.

Макрофункції зосереджено в підкаталозі \maxplus2\max2lib\mf бібліотеки функціональних модулів САПР. Для добору потрібної макрофункції найзручніше з меню Help > Old-Style Macrofunctions вийти на довідкову сторінку Old-Style Macrofunctions (by Function), де репрезентовано всі їх категорії за функціональним призначенням (21 категорія). Якщо ж потрібно продивитися дані щодо відомої макрофункції, то зі сторінки Old-Style Macrofunctions (by Function) можна перейти на сторінку Old-Style Macrofunctions (by Number).

У цій лабораторній роботі розглядаються макрофункції двох категорій:

а) буфери (Buffers, рис. 2.2, а), зокрема, **btri** – еквівалентний примітиву TRI Primitive з інверсним входом дозволу OE, а також ІС, що містять по шість і вісім буферів з трьома станами виходу;



а) б) в)

Рисунок 2.2 – Макрофункції: а) порти; б) буфери; в) логічні примітиви

б) логічні функції (SSI Functions, рис. 2.2, б), серед яких буфер **cbuf** (Complementary Buffer), еквівалентний буферу GLOBAL з прямим та інверсним виходами, логічний елемент заборони **inhb** (Inhibit Gate) та всі стандартні логічні елементи, у тому числі комбіновані: І-АБО-НЕ (AND-OR-INV) та І-АБО (AND-OR).

Макрофункції відрізняються від примітивів не тільки підвищеною складністю, але й тим, що, по-перше, вони виконані *з портами*, приєднаними до всіх їх зовнішніх виводів, та, по-друге, їх можна *перестроювати* щодо кількості входів і виходів (у бік зменшення), а також виконати довільні входи і виходи прямими чи інверсними. Завдяки цьому макрофункції набувають пе-

вної універсальності: на їх основі легко утворити не тільки стандартні, але й довільні функції.

🖫 ***Приклад.*** З використанням макрофункції виконати логічну операцію

$$y = \overline{x_1 + x_2 + x_3 x_4 x_5 x_6}.$$

На рівні примітивів таку функцію можна здійснити на двох логічних елементах – тривходовому АБО-НЕ та чотиривходовому І:

$$a = \overline{x_1 + x_2 + x_3}; \quad y = a x_4 x_5 x_6.$$

Проте якщо задану функцію перетворити до вигляду

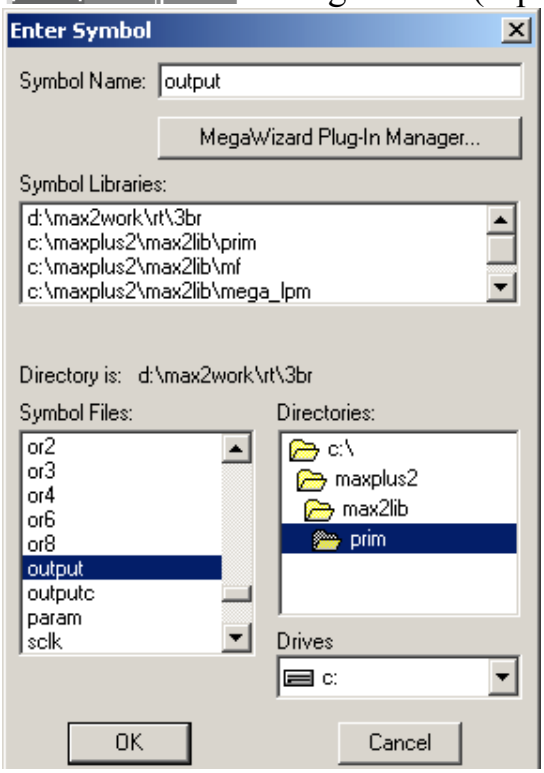
$$y = \bar{x}_1 \bar{x}_2 \bar{x}_3 x_4 x_5 x_6,$$

то потрібен один нестандартний елемент І (із шістьма входами, три з яких є інверсні), який неважко виконати за допомогою макрофункції (рис. 2.2, в).

2.2 Лабораторне завдання

2.2.1 Засвоїти основи *графічного* введення проекту на рівні примітивів для заданого варіанту XX логічної функції

1.1 Створити власний графічний файл, наприклад, 200.gdf (варіант XX=00) з директорією \max2work\rt\0br\200.gdf (послідовністю наведених піктограм або командами з меню): визначити ім'я проекту (1.2.1.4,а)...г), відкрити новий графічний файл (там само, підпункти д), е) та надати йому ім'я проекту (там само, підпункт є).



☝ ***Примітка.*** Ім'я проекту можна ввести (до верхнього рядка головного вікна Manager) піктограмою або командою з меню File > Project > Set Project to Current File і після формування будь-якого файла та надання йому імені.

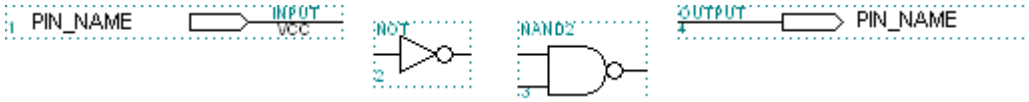
2.2.1.2 Для зручності розташування у вікні елементів схеми доцільно ***нанести сітку***: меню Options > Show guidelines, крок якої визначається командою: меню Options > Guideline Spacing.

2.2.1.3 Вставити символи елементів схеми *в базисі І-НЕ* з бібліотеки примітивів \maxplus2\max2lib\prim до створеного графічного файла:

а) подвійним клацанням (В**) у полі файла (або одиничним кла-

цанням і командою: меню Symbol > Enter Symbol) викликати діалогове вікно Enter Symbol (ввести символ);

б) B** у віконці Symbol Libraries по рядку \maxplus2\max2lib\prim перейти до бібліотеки примітивів > у віконці Symbol Files прокрчуванням знайти потрібний елемент (наприклад, input) і клацанням B** по ньому (або виділити його й натиснути ОК) повернутися до вікна графічного редактора, в якому з'явиться символ цього елемента, що розташується лівим верхнім кутком у місці клацання в підпункті 2.2.1.3, а);



в) повторюючи цю процедуру, ввести всі потрібні елементи, достатньо по одному їх різновиду, наприклад, порт input для утворення входів, по одному елементу not, nand2 для побудови схеми та порт output для організації виходів.

2.2.1.4 Розташувати елементи у вікні графічного редактора (приблизно як на схемі), користуючись інструментом вибору (натиснути значок стрілки на вертикальній палітрі інструментів):

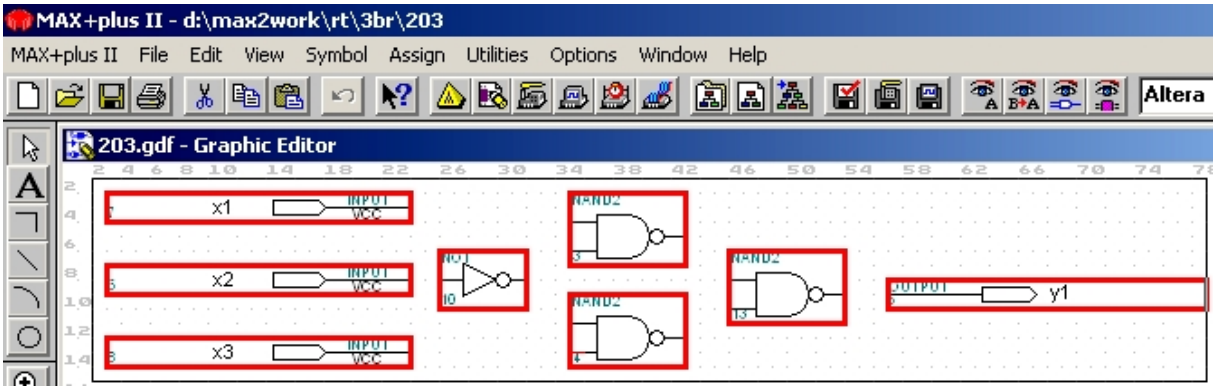
а) клацання всередині елемента *виділяє* його (червоним кольором), а поза елементом – скасовує виділення;

б) якщо поставити курсор миші на елемент, то при натиснутій лівій кнопці можна *пересунути* елемент по екрану (звичайний метод "Drang & Drop");

в) *скопіювати і пересунути* копію елемента на нове місце можна методом "Drang & Drop" при натиснутій і утримуваній клавіші Ctrl (її треба відпускати останньою);

г) за допомогою горизонтальної панелі інструментів попередньо виділений елемент можна, як звичайно, *вирізати*, *скопіювати*, *вставити* потрібну кількість разів у місцях клацання (у цьому або іншому файлі) та *скасувати останню дію* (зігнена стрілка), а також *видалити* клавішею Delete;

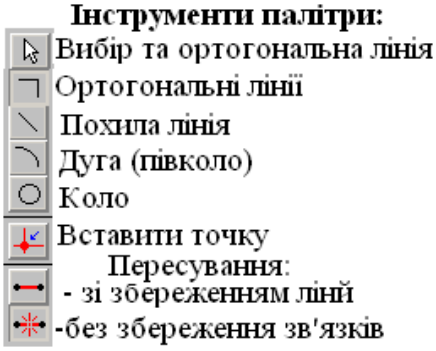
ґ) піктограмами горизонтальної панелі виділений елемент можна також розвернути по горизонталі (за ДСТУ в схемах цього робити не можна) або по вертикалі (змінюється позиціонування написів).



✍ ***Примітка.*** З *контекстного меню* (викликається кнопкою В2) виділений елемент можна так само вирізати зі збереженням у буфері (Cut), скопіювати (Copy), вставити (Paste), вилучити без збереження у буфері (Delete), а також розвернути по горизонталі (Flip Horizontal) або по вертикалі (Flip Vertical) та повернути на певний кут (Rotate).

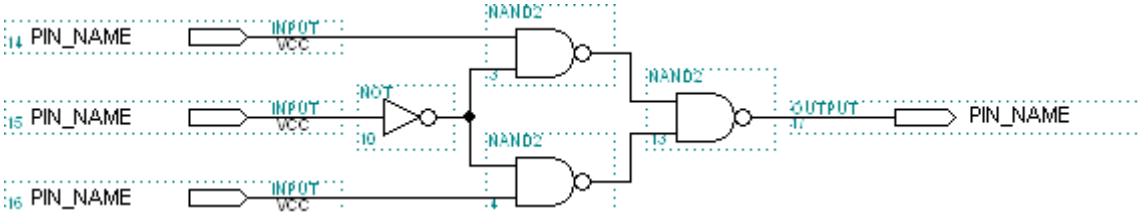
2.2.1.5 З'єднати логічні елементи між собою та з портами введенням з'єднувальних ліній інструментами палітри:

а) підвести курсор інструмента вибору (або ортогональних ліній) до виводу елемента чи порту і, коли він набуде форми хреста, натиснути ліву кнопку і провести лінію у вигляді горизонтальних та вертикальних відрізків, які закінчуються при відпусканні кнопки (інструментами похилої лінії та дуги можна ввести й інші різновиди відрізків);



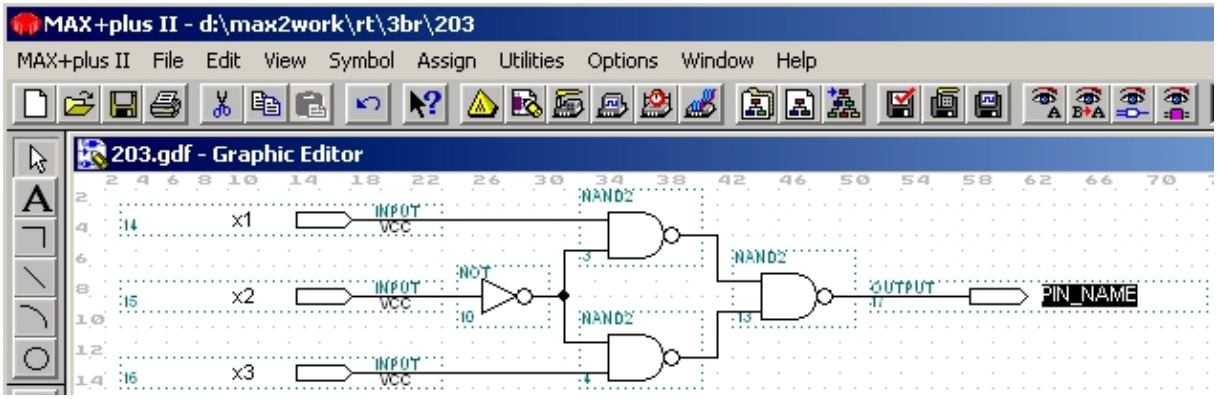
б) для вилучення зайвого відрізка його слід виділити (червоним кольором) підведенням стрілки і клацанням лівою кнопкою та видалити клавішею Delete (або інструментом „ножиці" на горизонтальній панелі інструментів, або командою Delete контекстного меню);

в) точки з'єднань утворюються автоматично після закінчення проведення відрізка (відпусканням клавіші) на перетині ліній; точку можна ввести або вилучити подвійним клацанням інструментами вибору або ортогональних ліній на перетині ліній; крім того, точку можна ввести, клацнувши на перетині ліній та натиснути інструмент вертикальної панелі „вставити точку" (повторне клацання інструмента вилучає точку);



г) якщо під час редагування схеми потрібно пересунути елемент зі збереженням або без збереження з'єднувальних ліній, достатньо скористатися двома інструментами палітри Rubberbanding (метод гумової нитки) – з кнопками on та of відповідно.

2.2.1.6 Позначити назви входів і виходів: Клацнути В** *на словах* PIN_NAME порту і коли ці слова виділяться чорним, ввести з клавіатури (не звертаючи уваги на курсор) вхідні змінні (x1, x2,...) та вихідні функції (y1, ...). Переводити курсор з одного порту на інший того ж типу можна клавішею Enter.



✍ ***Примітки:***

1. Скоригувати імена портів можна звичайним чином: виділити напис інструментом вибору або Enters new text (ввести новий текст – інструмент у вигляді літери А) та ввести новий.

2. Крім бібліотечних компонентів (\maxplus2\max2lib), елементи схеми можна імпортувати і з інших проектів (\max2work\tutorial), а також із системи OrCAD (графічні файли з розширенням **.sch**).

3. Фрагменти схеми легко розмножувати: виділити область, як звичайно, прямокутником (протягнути стрілкою по діагоналі при натиснутій лівій кнопці), відтак за допомогою інструмента Copies або з меню Edit скопіювати і вставити в потрібне місце цього або іншого файла та, у разі потреби, замінити елементи чи внести інші корективи. Вставити виділений фрагмент графічного файла в документ Microsoft Word зручно через графічний редактор Paint.

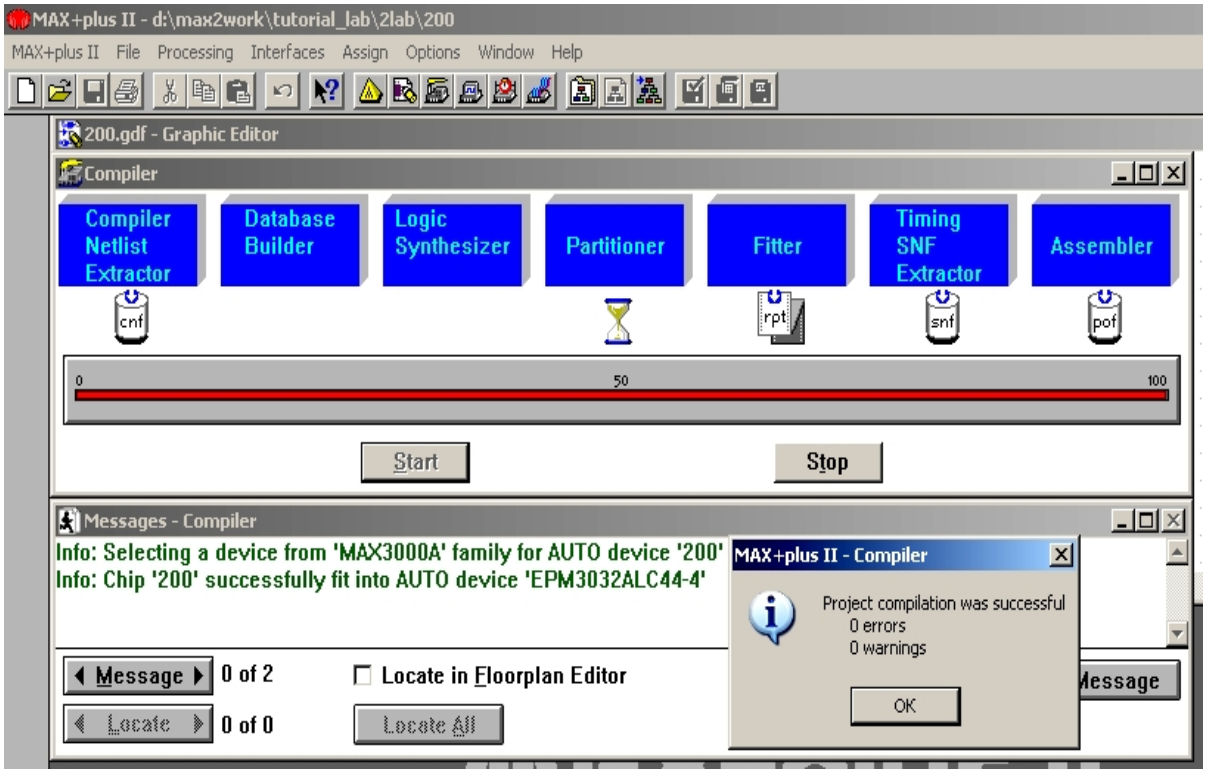
🖫 ***Приклад***: d:\max2work\tutorial\2lab\200.gdf (схема 1).

2.2.2 Виконати компіляцію проектованого пристрою

2.2.2.1 Вибрати тип мікросхеми (за змогою найнижчої складності), наприклад, серії HBIC MAX 3000: меню Assign > Devise > Devise Family: MAX3000A > Devises: EPM3032ALC44-4 (абоAUTO) > OK.

✍ ***Примітка.*** Якщо не виконувати п. 2.2.2.1, тип мікросхеми призначається автоматично, про що висвічується повідомлення під час компіляції (п. 2.2.2.2).

2.2.2.2 Викликати редактор компілятора піктограмою Saves … and starts the Compiler (зберегти зміни у файлі і запустити компілятор), після чого у віконці Messages – Compiler висвічується інформація і пові- домлення про помилки та застереження, а у вікні Compiler відобразяться назви комплекту сформованих файлів (зокрема, файл симулятора з розширенням .snf); відтак слід натиснути кнопку OK у віконці про успішну (successful) компіляцію та закрити вікно компілятора.



Примітки:

1. Помилки (повідомлення червоним кольором) треба усунути для успішної компіляції, застереження (синім кольором) – проаналізувати і врахувати чи проігнорувати, а інформацію (зеленим кольором) – взяти до відома на стадії конфігурування ВІСПС.

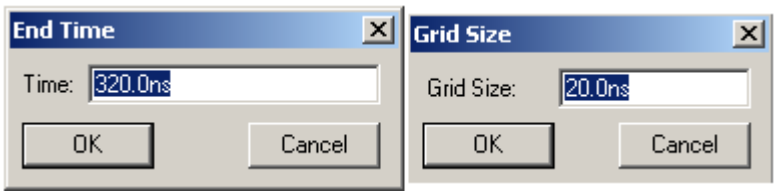
2. Запустити компілятор можна також піктограмою Compiler, зображеною ліворуч, (або з меню MAX+plus II > Compiler) і натисненням кнопки Start у вікні Compiler та, якщо надійде запит про збереження змін, дати ствердну відповідь. Крім того, піктограмою Saves ... and checks (зберегти файл і перевірити його на помилки), що зображена праворуч, першим натисненням кнопки Start у вікні Compiler запускається утиліта перевірки на синтаксичні помилки і на відповідність уведеної схеми правилам реалізації на ВІС, а закриттям (ОК) віконця з повідомленням про відсутність помилок і другим натисненням кнопки Start завершується компіляція.

3. Компілятор опрацьовує той проект, ім'я якого зазначене в рядку заголовка Manager (але не той, файл якого відкрито в даний час, можливо, для перегляду). Якщо потрібно скомпілювати інший проект, слід спочатку відкрити будь-який його файл і натиснути піктограму або ввести команду з меню File > Project > Set Project to Current File (установити проект за назвою поточного файла).

2.2.3 Виконати функціональне моделювання (Simulation) проектованого пристрою, яке полягає в імітації його функціонування шляхом формування часових діаграм вихідних функцій на основі логічних рівнянь, утворених на стадії компіляції.

2.2.3.1 Створити файл часових діаграм і надати йому ім'я з розширенням **.scf** за активного потрібного проекту (в рядку заголовка Manager) послідовністю піктограм або командами з меню (підпункт 1.2.1.4, д), е), є), після чого у вікні цього файла автоматично відобразиться назва 2XX.scf.

2.2.3.2 Задати параметри часових діаграм згідно з таблицею відповідності. Сітку моделювання (якщо не задано частоту сигналів) доцільно вибрати кратною десяти, наприклад, Δt = 20 *ns*, а весь інтервал моделювання – виходячи з кількості вхідних кортежів (наборів змінних), наприклад, при чотирьох змінних $T = 16\Delta t = 320$ *ns*. Ці параметри можна ввести в довільній послідовності, наприклад, меню File > End Time > 320 ns (виправити us = *мкс* на ns = *нс*) > OK, відтак меню Options > Grid Size > 20 ns > OK, після чого інструментом палітри Changes … розташувати все зображення у вікні.



2.2.3.3 Ввести тестові вектори (на часових діаграмах позначити координати сигналів) зі створеного компілятором SNF-файла: меню Node > Enter Nodes from SNF > кнопка List > кнопкою => скопіювати сигнали з лівого віконця (доступні вузли) до правого (вибрані вузли) > OK. У вікні редактора Waveform відобразяться (після натиснення OK) порожні смуги для вхідних сигналів та заштриховані (ознака невизначеності) – для вихідних.

Примітки:

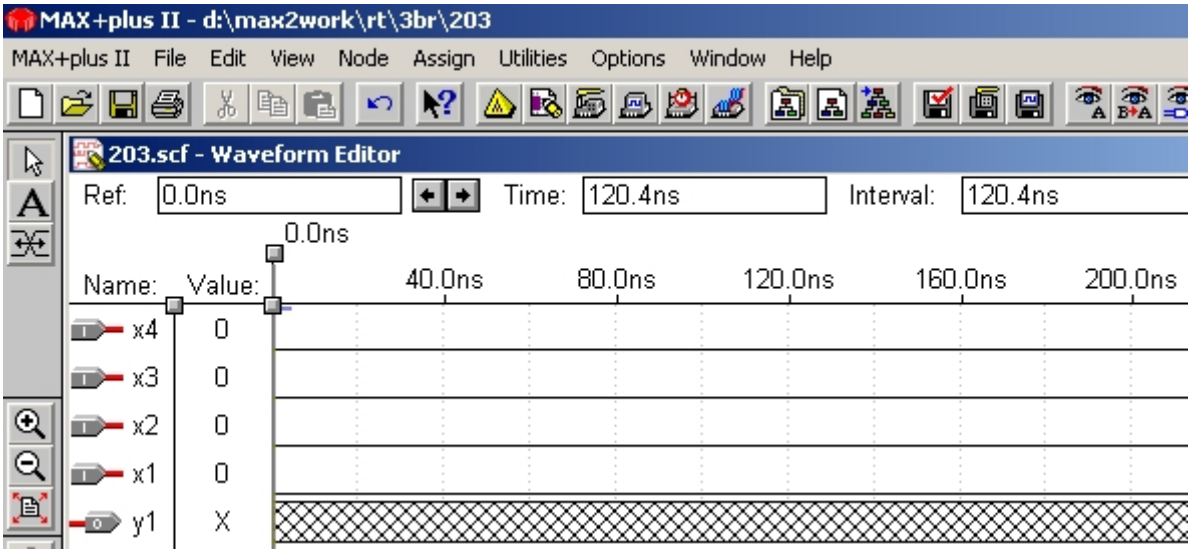
1. У разі потреби, у вікні Enter Nodes from SNF можна перенести не весь список (List) доступних вузлів-сигналів (ліве віконце) до списку вибраних вузлів (праве віконце), а вибірково – клацанням по назві сигналу мишею і натисканням кнопки => (для списку – протягуванням у лівому віконці натиснутою лівою клавішею).

2. За замовчуванням до тестових векторів входять сигнали всіх вхідних і вихідних портів. Якщо потрібно з метою дослідження переглянути також сигнали в проміжних точках схеми, їх можна вибрати в діалоговому вікні Enter Nodes from SNF ввімкненням попередньо прапорця Combinational або All та натисненням List.

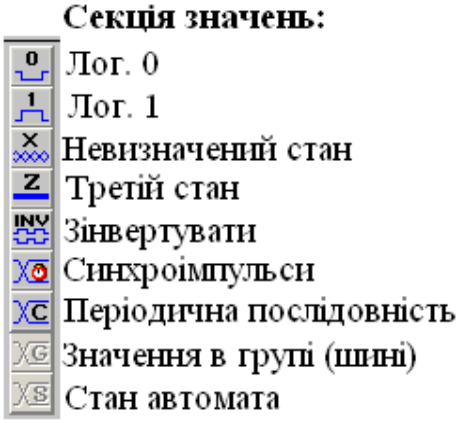
3. Розташування сигналів на часових діаграмах легко змінити: досить взятися за позначку порту в полі Name та (коли виділиться рядок і з'явиться прямокутничок) перетягнути його в потрібну позицію, яка висвічується блакитною лінією.

2.2.3.4 Ввести значення змінних за таблицею відповідності. Для цього за допомогою інструмента палітри вибору (стрілка) або інструмента виділення значень (подвійна горизонтальна стрілка) слід ввести рівні вхідних сигналів із секції значень на цій самій палітрі.

Починати зручно з молодшої змінної x_1, яка на кожному кроці сітки, що відповідає одному із 16 вхідних кортежів, послідовно набуває значень логічних 0 та 1.



Найшвидше такий сигнал ввести за допомогою кнопки періодичної послідовності таким чином:



а) клацнути на змінній x_1 в полі значень (Value), рядок якої виділиться чорним;

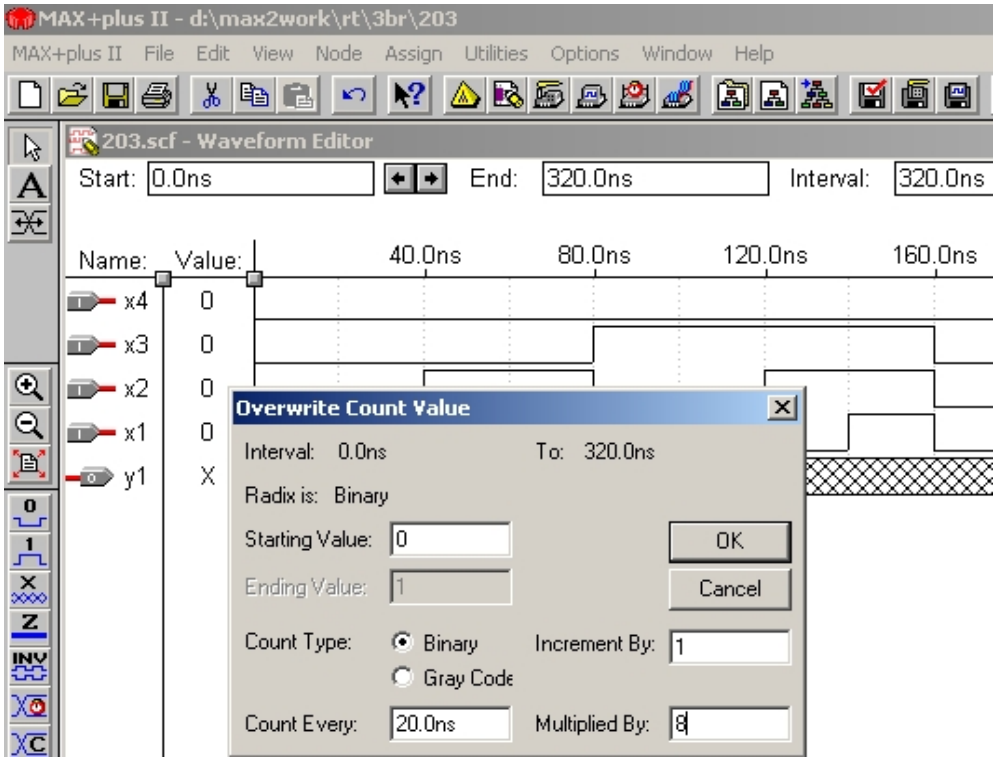
б) натисненням кнопки періодичної послідовності викликати діалогове вікно значень накладання записів (Overwrite Count Value), в якому ввести потрібні параметри послідовності, зокрема, початковий рівень сигналу (Starting Value) – залишити 0 (у разі потреби ввести 1); півперіод послідовності імпульсів (Count Every) – залишити таким, що за замовчуванням дорівнює крокові сітки (у разі потреби ввести інший); помножити на (Multiplied By) – залишити 1;

в) натисненням у цьому вікні ОК вводимо часову діаграму змінної x_1 у формі меандра;

г) повторюємо кроки *а), б), в)* для всіх інших вхідних сигналів, збільшуючи вдвічі період послідовності в міру зростання номерів розрядів змінних введенням коефіцієнтів множення (Multiplied By) на 2, 4, 8 і т. д.

Примітки:

1. Ввести значення змінних можна також в інший спосіб. Для цього протягуємо інструментом виділення значень або інструментом вибору над віссю змінної на одному кроці або кількох кроках часової сітки при натиснутій лівій кнопці – виділений інтервал заливається чорним. Відтак фіксуємо потрібне значення натисненням кнопки секції значень на вертикальній палітрі, наприклад, логічна 1. У такий самий спосіб вводимо потрібні рівні

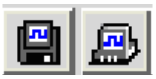


на всіх інших ділянках часових діаграм. Для виправлення помилково введеного рівня слід так само виділити чорним хибний інтервал і зафіксувати кнопкою потрібний рівень, наприклад, замінити логічну 1 на логічний 0.

2. Якщо в меню Options виставлено прапорець Snap to Grid (прив'язка до часової сітки), то ділянки часових діаграм виділяються (отже і фіксуються на них рівні) на інтервалах, кратних кроку часової сітки. Для введення значень на довільних інтервалах незалежно від кроку сітки слід зняти зазначений прапорець.

3. Крок сітки можна змінити (для зручності введення або перегляду) у будь-який час, це не впливає на моделювання. Відображення сітки на екрані, як і в графічному файлі, включається та виключається прапорцем Show Grid в меню Options.

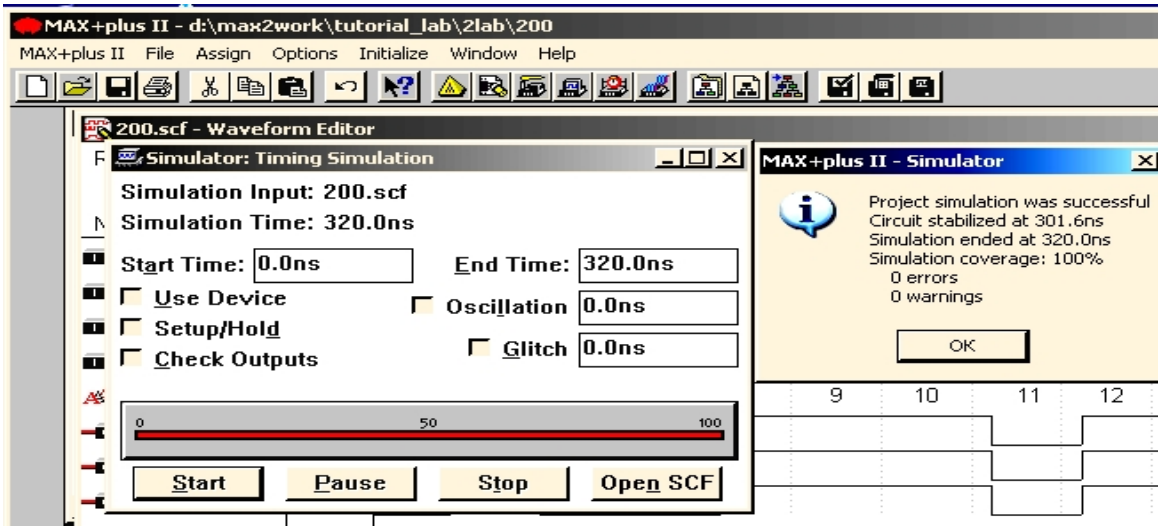
2.2.3.5 Зберегти зміни, внесені до файла часових діаграм, і ***запустити*** редактор функціонального моделювання (***Simulator***):



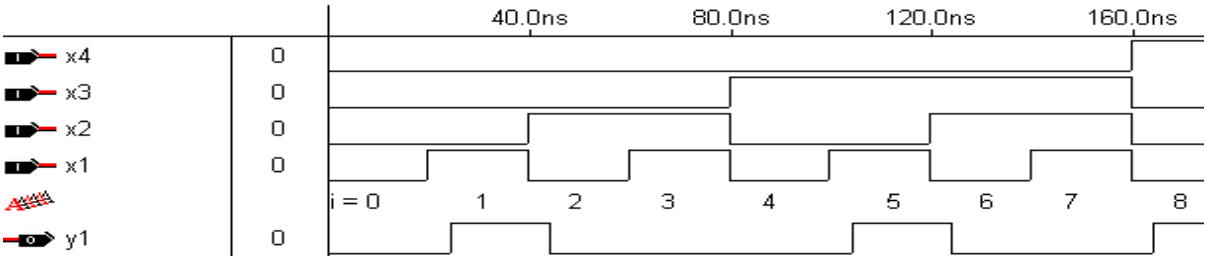
а) піктограмою (зображена зліва) Saves ... and starts the Simulator (зберегти відкриті файли часових діаграм і запустити імітатор – редактор функціонального моделювання) викликати діалогове вікно Simulator: Timing Simulation, в якому переконатися, що моделюється файл потрібного проекту, перевірити значення початку (Start Time) і кінця (End Time) процесу моделювання;

б) в інформаційному віконці MAX+plus II – Simulator з повідомленням про відсутність помилок (errors) та застережень (warnings) натиснути ОК (усунувши, у разі потреби, хиби, що було виявлено) та закрити діалогове вікно Simulator: Timing Simulation.

Примітка. Запустити редактор функціонального моделювання можна також піктограмою (зображена в підпункті 2.3.3.5, а) друга зліва) Opens the Simulator – відкрити імітатор (або з меню File > Save і меню MAX+plus II > Simulator) та в діалоговому вікні Simulator: Timing Simulation натиснути кнопку Start.



Приклад: \max2work\tutorial\2lab\200.scf.



Примітка. Вставити рядок коментарів до файла часових діаграм або відредагувати існуючий текст можна інструментом Enters new text: натиснути зазначений інструмент, клацнути ним рядок у вільному місці робочого поля і ввести текст.

2.2.4 Проаналізувати результати проектування пристрою в різних елементних базисах примітивів

2.2.4.1 Доповнити той самий графічний файл **схемою в *базисі АБО-НЕ*** зі спільними вхідними портами (див. приклад: проект d:\max2work\tutorial\2lab\200), скомпілювати та виконати функціональне моделювання, доповнивши часові діаграми вихідним сигналом нової схеми.

2.2.4.2 Перевірити, чи узгоджуються вихідні функції реалізованих схем з таблицею відповідності на всіх наборах змінних, користуючись часовим маркером (п. 1.2.2.1).

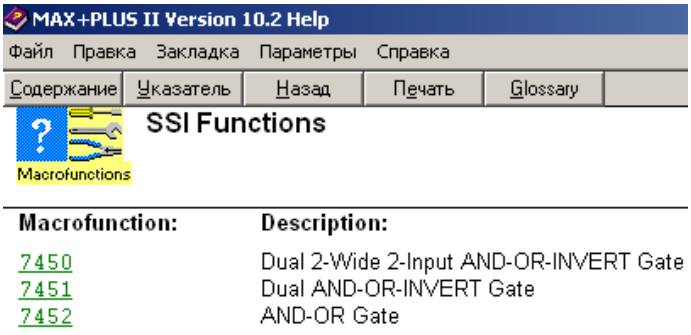
2.2.4.3 Виміряти швидкодію за часовими діаграмами автоматично (часовим аналізатором) та зберегти результати часового аналізу в текстовому файлі (п. 1.2.3).

🖫 ***Приклад*:** d:\max2work\tutorial\2lab\\200.gdf (схема 2), .scf, .tao.

2.2.5 Засвоїти основи створення проекту за його графічного введення на рівні *макрофункцій*

2.2.5.1 Ознайомитися з різновидами логічних елементів бібліотеки бази даних (файл 2libr.gdf): макрофункціями буферів (Buffers) і логічних функцій (SSI Functions). Розглянути принципові електричні схеми: натиснувши B** по символу, який виділяється червоним кольором, і таблицею відповідності: натиснути піктограму після чого клацнути по символу. (У звіті навести типові макрофункції з двох груп та дати тлумачення їх змісту).

2.2.5.2 Засвоїти методику користування електронним довідником системи MAX+... для добору потрібної стандартної IC 74-ої серії (або елемента, специфічного для цієї системи). Розгляд вестимемо на прикладі реалізації логічної функції (10) у базисі І-АБО-НЕ, яка належить до категорії SSI Functions (логічні функції).



а) Дібрати придатну для нашої мети IC: вікно Manager > меню Help > Old-Style Macrofunctions > зі списку Macrofunction Categories вибрати потрібну категорію (у прикладі – SSI Functions) > користуючись колонкою опису (Description), дібрати потрібну IC 74-ої серії або іншу макрофункцію (наприклад, 7451 Dual AND-OR-INVERT Gate – два елементи І-АБО-НЕ) та курсором натиснути її ім'я (7451) > ознайомитися з таблицею відповідності, нумерацією виводів та іншою текстовою інформацією.

б) Вставити символ дібраної IC з бібліотеки макрофункцій \maxplus2\ \max2lib\mf до свого варіанта графічного файла: відкрити вікно цього файла натиснувши > B** у місці розташування символу в лівому верхньому кутку (або одиничним клацанням у цьому місці і командою: меню Symbol > Enter Symbol) викликати діалогове вікно Enter Symbol > ввести у віконці Symbol Name (або вставити кнопкою B2 попередньо скопійоване в довідці) ім'я макрофункції > ОК (або інакше: натиснувши B** у віконці Symbol Libraries на рядку \maxplus2\max2lib\mf перейти до бібліотеки макрофункцій > у віконці Symbol Files знайти потрібний елемент і клацнути B** на ньому) і тим самим повернутися до вікна графічного редактора, в якому з'явиться символ макрофункції.

в) Дістати довідку щодо макрофункції, зображеної символом:

– B** на символі (який виділяється червоним кольором) викликати

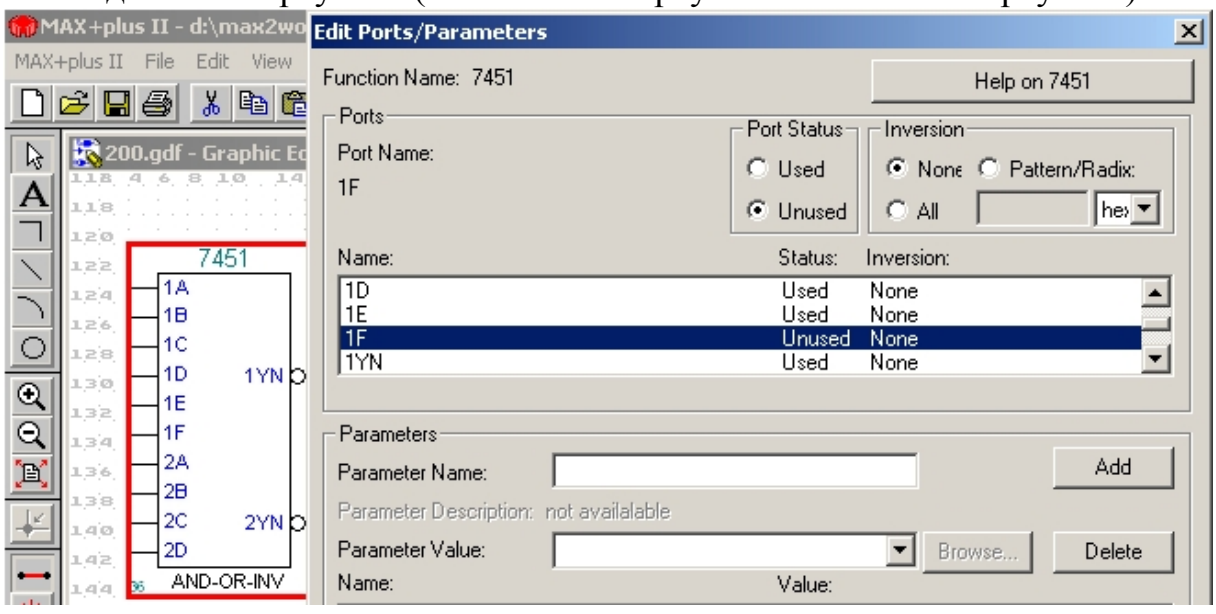
графічний файл з принциповою електричною схемою макрофункції, ознайомитися з нею, зокрема, з нумерацією портів (схемою можна скористатися і для побудови власного графічного файла: скопіювавши, вилучити або перейменувати порти, взяти лише якусь її частину тощо, відтак виділити, скопіювати та вставити до свого файла) та, закривши файл схеми, повернутися до свого файла;

– викликати довідковий файл: натиснути піктограму зі знаком запитання і потім клацнути на символі (або B2 на символі > діалогове вікно Edit Ports/Parameters > натиснути кнопку Help on 7451) > у довідковому вікні (тому ж самому, що викликається з меню Help) розглянути таблицю відповідності та інші текстові дані > закривши довідкове вікно, повернутися до свого файла.

2.2.5.3 Настроїти макрофункцію:

1) клацанням B2 на вставленому символі (див. п. 2.2.5.2, б) та натисненням Edit Ports/Parameters викликати діалогове вікно редагування Edit Ports/Parameters, в якому для макрофункції доступною є лише верхня половина (Ports);

2) користуючись іменами або нумерацією виводів IC (у разі потреби скористатися довідкою – п. 2.2.5.2, в), редагуємо порти: виділяємо порт (віконце Name), у закладці Port Status призначаємо його стан (Used – використовуваний або Unused – невикористовуваний), а в закладці Inversion – необхідність інвертувати (None – не інвертувати або All – інвертувати).



Приклад. До файла 200.gdf (схема 3) вставляємо символ макрофункції 7451; у діалоговому вікні входи 1A, 1B не редагуємо, а вхід 1C вилучаємо: виділяємо його і вмикаємо перемикач Unused; так само залишаємо незмінними входи 1D, 1E та 2A … 2D і вихід 1YN, а вхід 1F вилучаємо; і, нарешті, вихід 2YN інвертуємо: виділяємо його рядок і натискаємо All,

щоб подвійним запереченням скасувати інверсію на виході (з'явиться синя одиничка біля кола інверсії, яка означає в даному випадку, що вона стосується одного виводу символу). У підсумку після закриття (ОК) вікна Edit Ports/Parameters отримуємо настроєну макрофункцію (показана у файлі d:\max2work\tutorial\2lab\200.gdf, схема 3).

☝ ***Примітка.*** У випадку редагування шин для інвертування окремих їх розрядів користуємося кодами в одній із систем числення (останнє віконце, закрите під час виділення одиничних виводів ІС).

2.2.5.4 Вставити до графічного файла інші потрібні елементи, з'єднати їх згідно з логічною формулою, виконати компіляцію і моделювання так само, як і в проекті з примітивами (п. 2.2.1–2.2.3).

🖫 ***Приклад***: d:\max2work\tutorial\2lab\200.gdf, .scf (схема 3).

2.2.5.5 Доповнити той самий графічний файл мінімізованою схемою в базисі одного елемента (АБО-НЕ чи І-НЕ, що забезпечує меншу складність) на макрофункціях (п. 2.2.5.2, б); 2.2.5.3; 2.2.5.4) та проаналізувати результати проектування на різних макрофункціях аналогічно п. 2.2.4.

🖫 ***Приклад***: d:\max2work\tutorial\2lab\200.gdf, .scf (схема 4).

Контрольні питання та завдання

1 Які є відмінні риси зображення логічних функцій, притаманні таким формам: мішаній, ДФ, ДНФ, ДДНФ, МДНФ, КФ, КНФ, ДКНФ, МКНФ?

2 Спростіть вирази:

1) $(\overline{x_1}+\overline{x_2})(x_1+x_3)(\overline{x_2}+x_3)$;
2) $x_1\overline{x_2}+x_1\overline{x_3}+\overline{x_1}x_2+x_2\overline{x_3}$;
3) $(\overline{x_1}+x_2)(x_1+\overline{x_3})(x_2+\overline{x_3})$;
4) $(x_1+\overline{x_2})(\overline{x_1}+x_2)(\overline{x_1}+\overline{x_3})(x_2+\overline{x_3})$;
5) $(x_1+\overline{x_2})(\overline{x_1}+x_2)(\overline{x_1}+\overline{x_3})(x_2+\overline{x_3})$;
6) $x_1\overline{x_2}+x_1\overline{x_3}+x_1\overline{x_4}+x_2\overline{x_3}+x_2\overline{x_4}$;
7) $\overline{x_1x_2+x_3}+x_2\overline{x_1+x_3}+x_3\overline{x_1+x_2}+\overline{x_1+x_2+x_3}$;
8) $(x_1+\overline{x_2})(x_1+\overline{x_3})(x_1+\overline{x_4})(x_1+\overline{x_5})(\overline{x_2}+x_5)(\overline{x_3}+x_5)(\overline{x_4}+x_5)$.

Вказівка. За необхідності, скористайтеся діаграмами термів.

3 Виконайте спільну мінімізацію чотирьох вихідних функцій, що здійснюють перетворення ДДК 8421 у такі коди: *а*) 2421, *б*) 7421, *в*) 8421+3 (з надлишком три), *г*) код Ґрея (перші десять цифр X_{10}=0...9 чотирирозрядного коду). *Вказівки.* На заборонених кортежах $X_{10} > 9$ вважати функції невизначеними. Схему мінімальної складності реалізуйте на довільних логічних елементах.

3 ДЕШИФРАТОРИ І ШИФРАТОРИ

Мета роботи: дослідження типових дешифраторів семисегментного коду, двійкових дешифраторів, демультиплексорів, шифраторів; побудова ЦКП на дешифраторах; створення проекту за його текстового введення, засвоєння основних операторів логічної секції та запровадження прототипів і символів у текстовому редакторі.

Домашнє завдання

Для заданого варіанта (див. додаток А, варіант завдання 3) спроектувати: 1) перетворювач двійкового коду в код семисегментного індикатора для відображення вказаних знаків на дешифраторах різної розрядності та вибрати оптимальний варіант; 2) двійковий дешифратор та дешифратор-демультиплексор; 3) первісний вираз функції зі свого варіанта завдання 2 перетворити мовою опису апаратури AHDL.

3.1 Стислі теоретичні відомості

3.1.1 Перетворювачі кодів

Перетворювачі кодів є ЦКП, що здійснюють перетворення цифрової інформації з однієї форми відображення до іншої. У загальному випадку вхідний *m*-розрядний і вихідний *n*-розрядний коди можуть бути довільними – як числовими, так і комбінаторними.

Прикладом нечислових комбінаторних кодів є коди цифрових індикаторів відображення інформації. Символи на індикаторній панелі формуються на основі рідких кристалів або світлодіодів шляхом 7-сегментного, 14-сегментного чи мозаїчного розкладу зображення. Сегменти поширеного в малогабаритних пристроях 7-сегментного індикатора позначають літерами a, b, c, d, e, f, g (рис. 3.1, а). Під керуванням перетворювача кодів окремі частини панелі активізуються, наприклад, рідкі кристали, виконані у формі сегментів, темнішають на сріблястому тлі, утворюючи зображення потрібного символу. Якщо не активізовано сегменти f та c, індикується цифра 2, а якщо b та e – цифра 5 і т. ін.

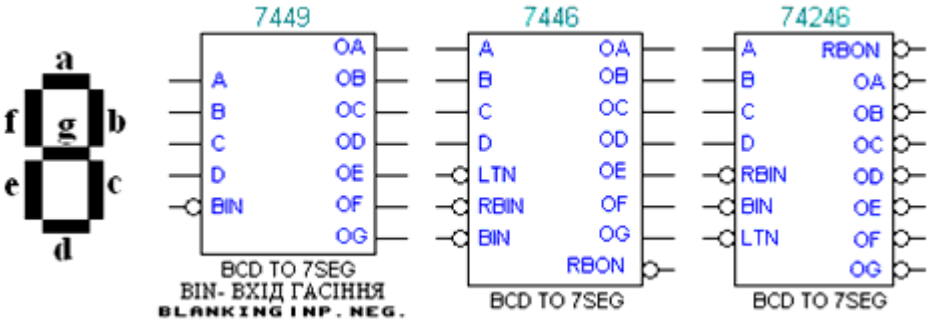


Рисунок 3.1

Для відтворення стандартних знаків, зокрема, цифр 0...9 двійково-десяткового коду (ДДК (BCD) налагоджено випуск серійних ІС – перетворювачів кодів, які в довідковій літературі називають дешифраторами 7-сегментного коду (рис. 3.1, б). Крім інформаційних входів тетради ДДК a[3..0] = D, C, B, A та сегментних виходів $a...g$ = OA…OG такі дешифратори мають інверсний вхід гасіння BIN, яким всі сегменти обнуляються (індикатор гасне).

Окремим випадком перетворювачів кодів є пристрої, для яких вхідним або вихідним є так званий унітарний код “1 із К”, в якому активний рівень може існувати тільки в одному розряді. Якщо активним є рівень логічної 1, то код називають прямим, а якщо логічний 0 – інверсним. Прикладом є унітарний десятковий код “1 з 10” відображення натиснутої цифрової клавіші, якщо натиснення більше однієї клавіші заборонено.

3.1.2 Дешифратори

Дешифратором (DC – decoder) називається перетворювач m-розрядного двійкового коду до n-розрядного унітарного; при цьому розрядність коду розширюється, бо $n > m$. Дешифратори найчастіше застосовуються для вибору (селекції) інтегрованої мікросхеми або іншого пристрою з метою обміну інформацією, наприклад, для адресування до окремих комірок пам’яті.

За допомогою вхідного m-розрядного коду дешифратор спроможний керувати $n = 2^m$ вихідними лініями, що є розрядами унітарного коду. Такий дешифратор є повним, а якщо $n < 2^m$ – неповним. Наприклад, дешифратори 1:2, 2:4, перетворювачі двійкового коду в унітарні вісімковий 3:8 і шістнадцятковий 4:16 є повними, а перетворювач тетради ДДК до унітарного десяткового коду 4:10 є неповним.

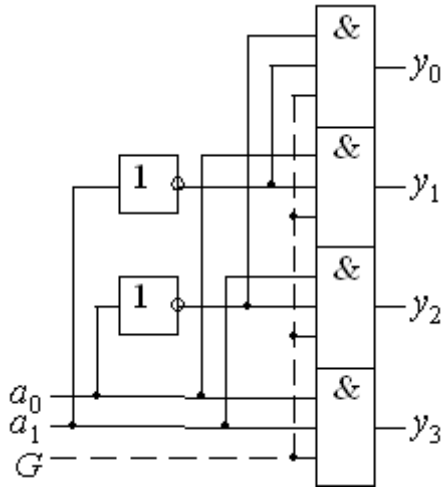
Принцип побудови дешифратора розглянемо на прикладі перетворення двійкового коду $A=a_1a_0$ в унітарний $Y=y_3y_2y_1y_0$ (рис. 3.2, а). З огляду на те, що кожна з вихідних функцій визначається одним мінтермом, вона вже є мінімальною:

$$y_0 = \overline{a_1}\ \overline{a_0};\ y_1 = \overline{a_1}\ a_0;\ y_2 = a_1\ \overline{a_0};\ y_3 = a_1a_0. \qquad (3.1)$$

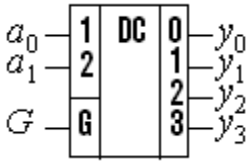
Взагалі, при m змінних функції повного дешифратора реалізуються за допомогою $n = 2^m$ елементів І з m входами кожний (рис. 3.2, б). Активний рівень з’являється тільки на тому виході дешифратора, номер якого відповідає вхідному кодові. Наприклад, при $a_1a_0 = 10_2 = 2_{10}$ дві одиниці прикладено до входів тільки одного елемента І з виходом y_2, тому $y_2 = 1$, а на всіх інших виходах встановлюються рівні логічного 0. Визначити номер активного виходу дуже просто: досить скласти ваги розрядів на вхідному полі умовного графічного позначення дешифратора (рис. 3.2, в), на яких діють рівні логічної 1.

i	a_1	a_0	y_3	y_2	y_1	y_0
0	0	0	0	0	0	1
1	0	1	0	0	1	0
2	1	0	0	1	0	0
3	1	1	1	0	0	0

а)



б)



в)

Рисунок 3.2 – Принцип побудови дешифратора

Якщо додаткові входи елементів І з'єднати зі спільним входом G (від Gate – ворота) як на рис. 3.2, б) позначено пунктиром, здобудемо стробований дешифратор (див. рис. 3.2, в). При $G = 0$ елементи І блоковано і на всіх прямих виходах встановлюються рівні логічного 0, а при $G = 1$ схема функціонує як звичайний дешифратор. За допомогою входів G можна синхронізувати роботу дешифраторів, а також виконувати їх каскадування.

3.1.3 Демультиплексори

Демультиплексором 1 : n ("з 1 в n") називається комутатор сигналів з одного в n каналів. На відміну від механічного перемикача комутування таким пристроєм здійснюється за допомогою цифрових кодів. Функцію демультиплексора виконує дешифратор зі стробовим входом G (див. рис. 3.2, б), в), якщо вхід G використовувати як інформаційний x: адресним, наприклад, дворозрядним кодом $A = a_1a_0$ активізується тільки один елемент І, тому на виході y_i цього елемента з'являється сигнал x. З огляду на це стробований дешифратор називають також дешифратором-демультиплексором. Отже, демультиплексор має один інформаційний, k адресних входів та $n = 2^k$ виходів; при $k = 2, 3, 4$ кількість виходів становить відповідно $n = 4, 8, 16$.

Крім повних дешифраторів серії 74 (рис. 3.3), у тому числі стробованих 2:4, 3:8 та спеціалізованої макрофункції дешифратора 4:16, у бібліотеці є також стандартні неповні дешифратори 4:10 кодів ДДК, Ґрея і ДДК з надлишком три.

3.1.4 Шифратори

Шифратор (CD – Encoder) є перетворювач m-розрядного унітарного коду до n-розрядного двійкового коду. Шифратор $m : n$ здійснює стискання інформації, бо $n < m$. Типовим його застосуванням є зв'язок між блоками, зокрема, уведення інформації з клавіатури: при натисненні однієї з клавіш

на виході шифратора утворюється тетрада ДДК десяткової цифри або двійковий комбінаторний код літери чи іншого символу.

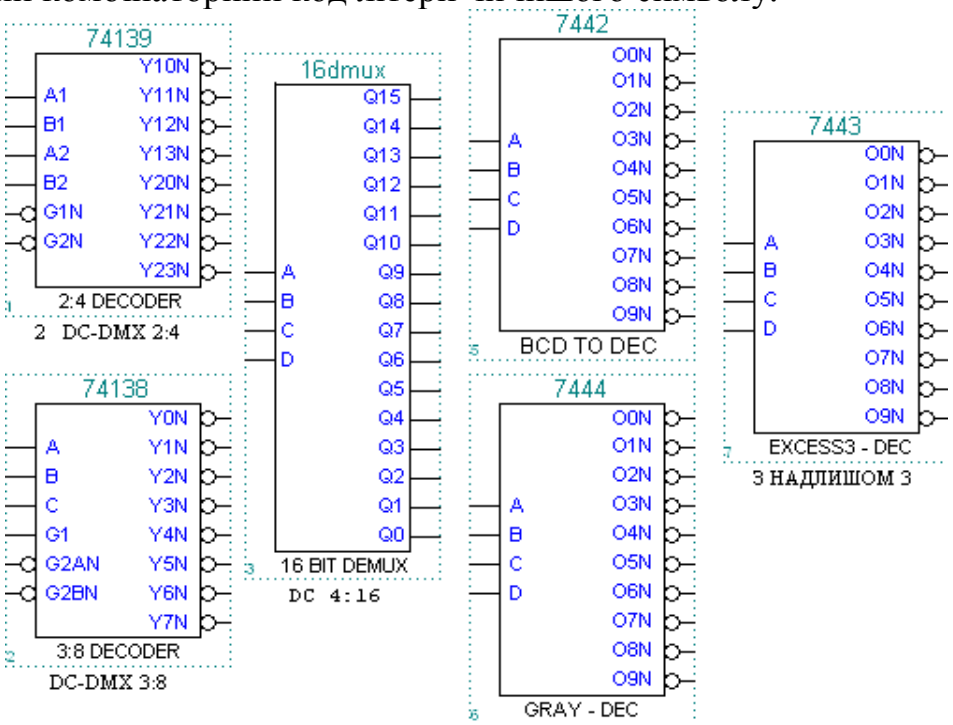
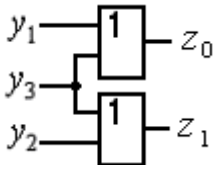


Рисунок 3.3 – Спеціалізовані макрофункції дешифраторів

Вихідний n-розрядний код шифратора спроможний відобразити інформацію, що надходить до нього по $m = 2^n$ лініях, тобто 2^n розрядів унітарного коду. Такий шифратор є повним, а якщо $m < 2^n$ – неповним. Наприклад, повними є шифратори 4:2, 8:3, а перетворювач унітарного десяткового коду в тетради ДДК 10:4 є неповним.

i	y_3	y_2	y_1	y_0	z_1	z_0
-	0	0	0	0	X	X
0	0	0	0	1	0	0
1	0	0	1	0	0	1
2	0	1	0	0	1	0
3	1	0	0	0	1	1

а)



б)

Рисунок 3.4 – Принцип побудови шифратора

Принцип побудови і особливості проектування шифраторів розглянемо на прикладі перетворення 4-розрядного унітарного коду у двійковий: вибираємо кількість розрядів двійкового коду $n = 2$; складаємо скорочену таблицю відповідності (рис. 3.4, а), в якій зазначаємо лише припустимі набори змінних, бо в усіх інших із $2^4 = 16$ рядків таблиці буде міститися більш, ніж одна одиниця у вхідному коді; визначаємо рівняння вихідних функцій

$$z_0 = y_1 + y_3; \quad z_1 = y_2 + y_3$$

та будуємо схему (рис. 3.4, б).

На відміну від унітарного коду “1 із К”, у коді “Х із К” може міститися довільна кількість одиниць, проте вихідним кодом має відображатися лише одна з них, із найбільшим пріоритетом. Код “Х із К” можна також моделювати клавішами, коли можливе натиснення більш ніж однієї з них. Шифратор, що перетворює код “Х із К” у двійковий, називається ***пріоритетним***. Крім вихідного коду *Y* він може формувати також сигнал запиту на обслуговування GS, який стає активним за надходження хоча б одного з вхідних сигналів. Бібліотека містить пріоритетні шифратори 10:4 і 8:3 (рис. 3.5).

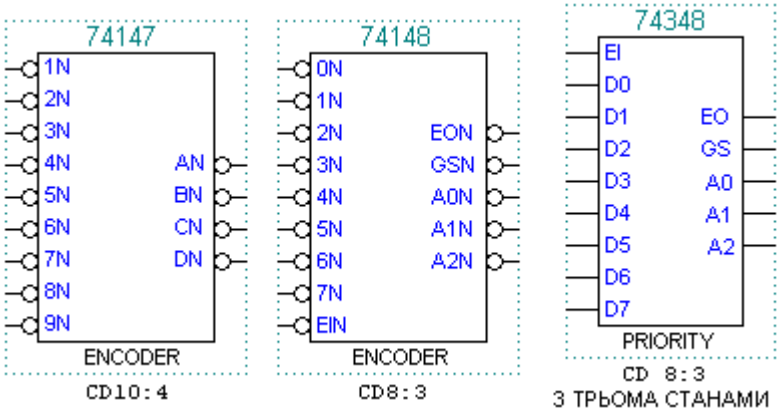


Рисунок 3.5 – Спеціалізовані макрофункції шифраторів

3.1.5 Елементи арифметики

3.1.5.1 Числа

Мовою AHDL застосовуються числа у звичайних системах числення (табл. 3.1) для зображення постійних значень (параметрів, змінних, номерів) в арифметичних і булевих виразах. При цьому значення набору змінних компілятор перетворює в послідовність двійкових цифр. Проте значення однієї змінної в булевих виразах не можна позначати числом, наприклад, замість виразів типу *c* = 0 або *c* = 1 слід вживати *c* = GND або *c*=VCC.

Таблиця 3.1

Основа системи числення	Цифри	Приклад	Примітки
Десяткова	0…9	26	Число не береться в лапки
Двійкова	0, 1, X	B”11010”	X – довільне значення
Шістнадцяткова	0…F	H”1A”	Варіант: X”1A”
Вісімкова	0...7	Q”32”	Варіант: O”32”

3.1.5.2 Групи

Низка однотипних елементів і їх назв для стислості може бути оголошена у вигляді окремого модуля, який позначається символом групи. Наприклад, масив адресних сигналів і відповідних їм портів a3, a2, a1, a0 можна зобразити групою a[3..0].

3.1.5.3 Арифметичні оператори

Оператори використовуються в арифметичних виразах, щоб оцінити значення функцій, констант і параметрів а також виконати операції порівняння (в останньому випадку арифметичні оператори називають також компараторами). Такі вирази не потребують ресурсу ІС і призначені для зручності програмування; компілятор виконує обчислення і використовує лише результати – числа, які не можуть бути від'ємними. Основні арифметичні оператори і компаратори наведено в табл. 3.2.

Таблиця 3.2

Оператор	Пріоритет	Опис	Приклад
+ (unary – одномісний)	1	додатне число	+1 або 1
– (unary – одномісний)	1	від'ємне число	–1
^	1	степінь	c ^ 2
DIV	2	ділення	c DIV 2
*	2	множення	c * 2
LOG2	2	логарифм двійковий	LOG2(c)
+	3	додавання	c + 1
–	3	віднімання	c – 1
== (numeric)	4	числова рівність	a[] == H"1A"
== (string)	4	рядкова рівність	"a" == "b"
!=	4	не дорівнює	c != a + 2
>	4	більше	c > LOG2(a)
>=	4	більше або дорівнює	c >= a DIV 2
<	4	менше	c < a ^ 2
<=	4	менше або дорівнює	c <= a – 2

✐ ***Примітка.*** Для обчислення кількості розрядів пристроїв широко застосовується операція LOG2. Тому якщо її підсумок є нецілe число, підсумок автоматично округляється до більшого цілого числа. Приклади: LOG2(8) = 3; LOG2(9) = 4 – демонструють, зокрема, розрядність повного дешифратора 3:8 і неповного 4:9. Для операцій LOG2 і DIV застосовуються також оператори CEIL (стеля) і FLOOR (підлога). CEIL – те саме, що й зазначене округлення до більшого цілого числа: CEIL(LOG2(9)) = 4, а FLOOR – округлення до меншого цілого числа: FLOOR(LOG2(9)) = 3.

3.1.6 Основні складники проекту текстового редактора

Текстовий опис проекту мовами програмування високого рівня HDL, зокрема, мовою AHDL, містить розгалужену систему засобів, таких як логічні рівняння, таблиці відповідності і т. ін. Пристрій, створений у текстовому редакторі, можна згорнути до символу і ввести до графічного файла. З іншого боку, графічні об'єкти, наприклад, макрофункції можна як готові модулі вводити до текстових файлів.

Структурно проект (Design) складається із секцій (Section), багато з яких в свою чергу поділяються на оператори або підсекції (Statement). Кожний такий структурний блок (їх біля трьох десятків) має суворо відповідати правилам свого синтаксису. Аби пришвидшити складання текстового файла (з розширенням **.tdf** або TDF-файла) і, головним чином, уникнути чи зменшити кількість синтаксичних помилок, введення здійснюється за допомогою шаблонів (Template) для певної мови, зокрема, мови AHDL (AHDL-Template). У шаблоні позначено комірки для введення тексту і, що є важливим, містяться необхідні за синтаксисом розділові знаки. Помилки виявляються під час компіляції і локалізуються процесором повідомлень (Message Processor).

Обов'язковими для кожного TDF-файла є дві секції: секція підпроекту (**Subdesign Section**), призначена для оголошення вхідних та вихідних портів, і логічна секція (**Logic Section**), яка визначає залежність вихідних сигналів від вхідних з використанням інших змінних, оголошених у файлі. Логічна секція складається з низки операторів, найважливішими серед яких є оператор булевих рівнянь **(Boolean Equation),** оператор таблиці відповідності **(Truth Table Statement)**, умовний оператор (**If Then Statement**) та оператор вибору **(Case Statement)**. Оператори If Then і Case подібні і в багатьох випадках може використовуватися будь-який з них. Проте в операторі If Then діють *вирази,* а в операторі Case – *константи*, які порівнюються з єдиним виразом умови.

Інші секції використовуються, здебільшого, залежно від вибраної елементної бази та вибраних засобів проектування. Оператор прототипу функцій **(Function Prototype Statement)** у текстовому файлі відіграє роль, аналогічну символу в графічному файлі з тими ж самими іменами входів і виходів. Наприклад, прототип макрофункції дешифратора 74139 на ІС серії 74 повторює його символ (див. рис. 3.3): FUNCTION 74139 (g1n, b1, a1, g2n, b2, a2) RETURNS (y10n, y11n, y12n, y13n, y20n, y21n, y22n, y23n); на графічних символах макрофункцій входи і виходи позначаються великими літерами, а в прототипах – маленькими, проте компілятор такої різниці не помічає.

Секція змінних **(Variable Section)** складається з низки підсекцій для оголошення змінних різного типу. Змінна мовою програмування високого рівня може означати як просто сигнал, так і деякий внутрішній щодо програмного забезпечення логічний модуль, наприклад, макрофункцію з бази даних або символ, створений користувачем. Підсекція оголошення вузла **(Node Declaration)** дозволяє ввести, зокрема, проміжні вузли (сигнали, особливо якщо вони використовуються неодноразово) для застосування в логічних рівняннях з метою їх спрощення. Підсекція оголошення модуля **(Instance Declaration)** дозволяє розрізнити окремі екземпляри функціональних вузлів, кожному з них надається символічне власне ім'я, наприклад, двом дешифраторам dc1 та dc2.

Булеві вирази компонуються мовою AHDL за допомогою операторів, черговість виконання яких визначається пріоритетом (табл. 3.3).

У найпростішому випадку операнди є однорозрядними змінними, а константи в логічних виразах (на відміну від цифр арифметичних виразів) мають позначатися як GND (логічний 0) та VCC (логічна 1).

Приклад на складний вираз:

$$y_0 = \overline{(x_1 + x_3 + \overline{x_4})\left[\ x_2 + \overline{x_1 x_4 + (x_1 \oplus x_4)}\ \right]} = \overline{(a)\left[x_2 + \overline{(b + c)}\ \right]},$$

де $a = x_1 + x_3 + \overline{x_4}; b = x_1 x_4; c = x_1 \oplus x_4$.

Задля спрощення запису до форми AHDL можна перейти частинами:

y0 = a !& (x2 # (b !# c)) = (x1 # x3 # !x4) !& (x2 # (x1&x4 !# x1$x4)).

Проте з метою уникнення помилок і забезпечення легкості читання вирази можна вводити до логічної секції складниками

a = x1 # x3 # !x4; b = x1 & x4; c = x1 $ x4; y0 = a !& (x2 # (b !# c))

за умови, що проміжні вузли *a, b, c* було попередньо оголошено оператором Node Declaration секції змінних Variable Section.

Змінити порядок виконання операцій можна, як звичайно, за допомогою дужок. Припускається також ліву частину зображати в інверсному вигляді, наприклад,

!y0 = a & (x2 # (b !# c)).

Таблиця 3.3

Операція		Пріоритет	Приклад	
Знак	Назва		Булів вираз	Вираз AHDL
! (оклику)	НЕ (NOT)	1	$\overline{x_1}$	!x1
& (амперсанд)	І (AND)	2	$x_1 x_2$	x1 & x2
!&	І-НЕ (NAND)	2	$\overline{x_1 x_2}$	x1 !& x2
$ (долар)	Виключне АБО (XOR)	3	$x_1 \oplus x_2$	x1 $ x2
!$	Виключне АБО-НЕ (XNOR)	3	$\overline{x_1 \oplus x_2}$	x1 !$ x2
# (дієз)	АБО (OR)	4	$x_1 + x_2$	x1 # x2
!#	АБО-НЕ (NOR)	4	$\overline{x_1 + x_2}$	x1 !# x2

3.2 Лабораторне завдання

3.2.1 Дослідити типові дешифратори і шифратори.

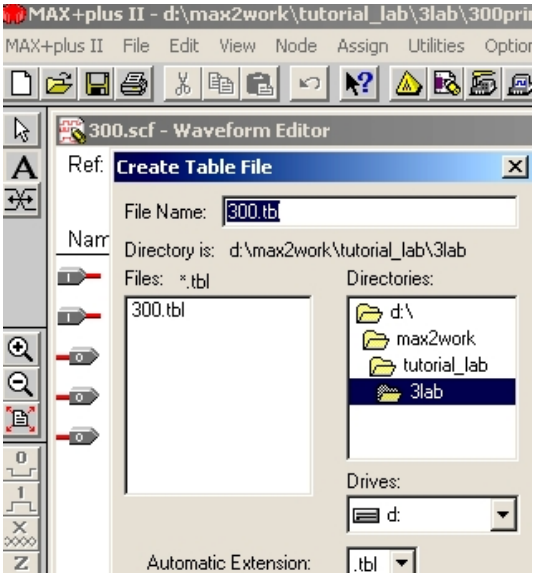
3.2.1.1 За принциповими електричними схемами *на логічних елементах* (файл d:\max2work\tutorial\3lab\3prim.gdf) та осцилограмами сигналів (d:\max2work\tutorial\3lab\3prim.scf) ***стробованого дешифратора*** (дешифратора-демультиплексора, селектора) ***та шифратора*** скласти таблиці відповідності і рівняння вихідних функцій, виміряти затримку вихідних імпульсів, навести умовне графічне позначення за ДСТУ та пояснити принцип дії таких ЦКП.

3.2.1.2 Ознайомитися зі стандартними дешифраторами та шифраторами серії 74 (макрофункціями): 1) повними дешифраторами 2:4, 3:8, 4:16 (файл d:\max2work\tutorial\3lab\3libre.gdf, рис. 3.1); 2) неповними дешифраторами 4:10 з трьома типами вхідних кодів: ДДК, Ґрея, з надлишком 3 (рис. 3.2); 3) дешифраторами (перетворювачами до) 7-сегментного коду (рис. 3.3); 4) шифраторами 8:3 і 10:4 (рис. 3.4). Розглянути принципові електричні схеми (B** на символі) та таблиці відповідності різновидів дешифраторів і шифраторів (достатньо по одному з кожної групи), пояснити призначення та особливості входів і виходів.

3.2.2 Застосувати дешифратор для реалізації ЦКП за *графічного опису* проекту на прикладі створення заданого варіанта перетворювача до 7-сегментного коду

3.2.2.1 У проекті 3XX ***зібрати на логічних елементах схему дешифратора***, потрібного для реалізації перетворювача, виконати компіляцію і функціональне моделювання.

🖫 ***Приклад:*** неповний дешифратор 2:3, файли d:\max2work\tutorial\3lab\300.gdf, .scf.

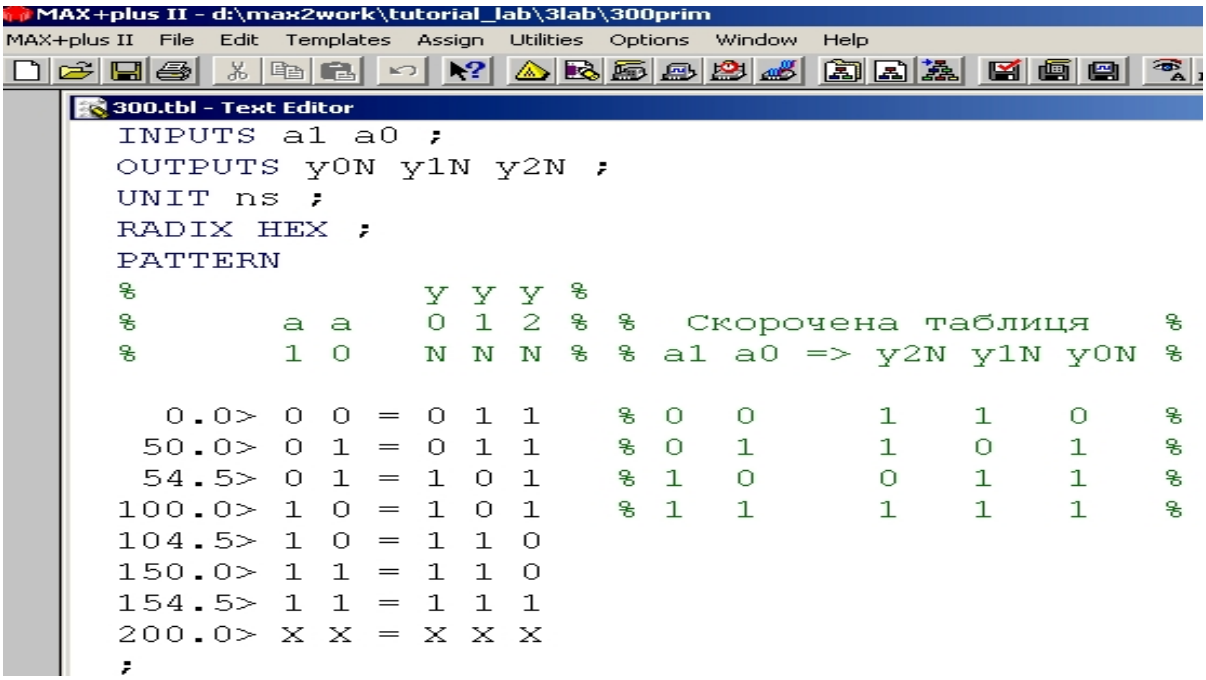


🖉 ***Примітка.*** Аби спростити збирання дешифратора, дозволяється користуватися як шаблонами *копіями* принципових електричних схем з доступних файлів, серед них принципових електричних схем макрофункцій.

3.2.2.2 Створити файл таблиці відповідності: відкрити файл часових діаграм .scf (або графічний файл .gdf і натиснути піктограму Simulator) > меню File > Create Table File > OK у віконці Create Table File > OK у віконці повідомлення про успішне генерування файла.

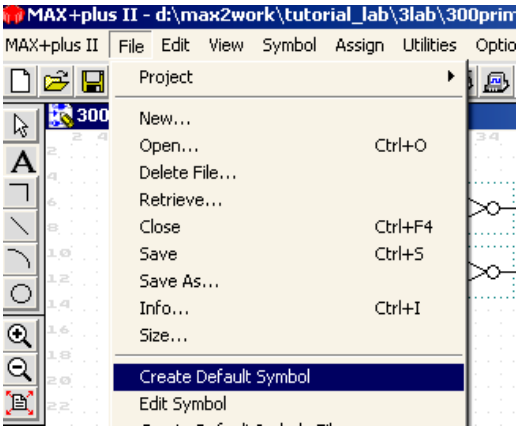
Відтак ***відкрити файл таблиці відповідності*** з розширенням **.tbl**, створений автоматично текстовим редактором: піктограма відкриття файла (або меню File > Open) > Text Editor files > *.tbl > B** на імені файла у списку Files. Порівняти цю таблицю з рівнями сигналів на відповідних інтервалах часових діаграм і скласти скорочену таблицю.

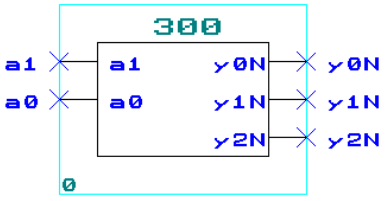
🖫 ***Приклад*** (файл d:\max2work\tutorial\3lab\300.tbl):



🖐 ***Примітка.*** За замовчуванням у таблиці відображаються всі вектори (вхідні змінні і вихідні функції) у тому ж порядку, що й у списку під час введення до імітатора Simulator, отже, їх розташування можна передбачити заздалегідь.

3.2.2.3 Створити символьний файл шляхом згортання схеми дешифратора до символу: відкрити файл, який треба згорнути до символу > ввести ім'я проекту (піктограмою або з меню File > Project > Set Project to Current File), якщо в рядку заголовка Manager зазначено інший проект > меню File > Create Default Symbol. Відтак ***відкрити символьний файл*** з розширенням **.sym**, створений автоматично символьним редактором: піктограма відкриття файла > Symbol Editor files > B** на імені файла у списку Files. Порівняти цей символ з принциповою електричною схемою.





🖐 ***Примітка.*** За замовчуванням на символі відображаються всі входи і виходи

незмінними відносно схеми, але їх можна відредагувати або створити символ вручну з меню Symbol.

3.2.2.4 Дібрати і настроїти потрібну макрофункцію дешифратора, зібрати на її основі схему перетворювача до 7-сегментного коду в графічному файлі нового проекту 3XX_7seg (п. 1.2.1.4), виконати компіляцію і функціональне моделювання та перевірити правильність функціонування пристрою за часовими діаграмами.

🖫 ***Приклад:*** файли d:\max2work\tutorial\3lab\300_7seg.gdf, .scf (схема 1).

☝ ***Примітка.*** Аби позиціонування написів у коментарях SCF-файла залишилося правильним, його вікно потрібно спочатку розгорнути, а відтак, за необхідності, згорнути.

3.2.2.5 Засвоїти використання символу в графічному редакторі: зібрати схему перетворювача до 7-сегментного коду в тому самому графічному файлі, користуючись створеним символом дешифратора (вставляється в графічний файл, як звичайно, але зі списку Symbol Files діалогового вікна Enter Symbol), виконати компіляцію і функціональне моделювання та перевірити правильність функціонування пристрою за часовими діаграмами.

🖫 ***Приклад:*** файли d:\max2work\tutorial\3lab\300_7seg.gdf, .scf (схема 2).

3.2.3 Виконати аналіз функціонування пристрою (з метою перевірки правильності проектування) методом автоматичної компіляції за *текстового* опису проекту апаратною мовою AHDL

3.2.3.1 Надати нове ім'я проекту: 2XXnode (див. п. 1.2.1.4).

3.2.3.2 Створити текстовий файл: викликати редактор Text Editor з меню MAX+plus II > Text Editor (або ярликом відкриття нового файла з панелі інструментів > Text Editor file > OK) і надати ім'я з розширенням .tdf безіменному (Untitled) файлу: File > Save As > OK (або ярлик збереження файла > OK у віконці Save As). Вікно текстового файла автоматично відобразить назву 2XXnode.tdf.

3.2.3.3 Рекомендується (але не обов'язково) ***ввести заголовок*** логічного блока за допомогою діалогового вікна шаблонів: B2 в полі файла (або меню Templates) > AHDL Template (далі ці дії стисло позначатимемо: B2 > AT) > Title Statement > OK. Поставити курсор на початку після лапок, ввести довільну назву (до 255 будь-яких символів крім крапок), в кінці залишити лапки і крапку з комою, а все інше стерти, наприклад:

TITLE "Логічна функція";

відтак ***курсор слід перевести*** на початок нового рядка.

3.2.3.4 Ввести обов'язкову частину Subdesign (підпроект) для оголошення вхідних змінних і вихідних функцій (відповідають портам – зовнішнім виводам мікросхеми): B2 > AT > Subdesign Section > OK та заповнити цей розділ:

а) після ключового слова Subdesign ввести назву проекту (до 32 символів) - ***ту саму***, що в заголовку файла без розширення (2XXnode), а інші символи видалити, наприклад:

SUBDESIGN 2XXnode

б) до блока в дужках ввести *через кому* перелік вхідних змінних і вихідних функцій, залишити двокрапку, ключове слово типу виводів та крапку з комою (після знаку рівності для входів зазначають, у разі потреби, рівні, які слід подати на незадіяні виводи мікросхеми, що становлять за замовчуванням GND чи VCC для функцій АБО та І відповідно), наприклад:

```
(
    x4, x3, x2, x1        : INPUT ;
    y0, y1, y2, y3        : OUTPUT;
)
```

всі зайві символи слід видалити, а ***курсор перевести*** на початок нового рядка після дужки.

☝ ***Примітка.*** Підпроект Subdesign може правити або за самостійний проект, або за складову частину деякого проекту, який у свою чергу може бути підпроектом складнішого суперпроекту і так далі за ієрархією.

3.2.3.5 У разі необхідності оголосити внутрішні вузли (відповідають проміжним змінним, не пов'язаним із зовнішніми виводами мікросхеми) вводять необов'язкову ***секцію змінних:*** B2 > AT > Variable Section > OK (відтак курсор слід перевести на наступний рядок, рекомендується з відступом Tab) та ***підсекцію оголошення вузлів:*** B2 > AT > Node Declaration > OK, в яку вставляють *через кому* імена зазначених змінних, залишають двокрапку, ключове слово та крапку з комою (відтак всі зайві символи видалити, а ***курсор перевести*** на початок нового рядка):

```
VARIABLE
    z       : NODE;
```

3.2.3.6 Ввести обов'язковий логічний блок (у даному випадку – мінімальної конфігурації) з метою пов'язати вихідні функції з вхідними (та, якщо є, проміжними) змінними: B2 > AT > Logic Section > OK. Відтак між ключовими словами (BEGIN та END;) розташувати курсор (краще з відступом Tab), ввести ***підсекцію Булевих рівнянь*:** B2 > AT > Boolean Equation та вставити в утворений таким чином шаблон мовою AHDL *через крапку з комою* логічні рівняння, можна в декілька рядків, наприклад:

```
BEGIN
    z = x1 !& x2;
    y0 = (x1 # x3 # !x4) !& (x2 # (x1&x4 !# x1$x4));
    !y1 = (x1 !& !x2) & (!x2 !& x4) & (!(!x1 & !x3 & x4));
    y2 = !((!x1 !# !x2) # (x1 !# x4) # (!x2 !# !x3));
    y3 = (x1 !& z) !& !(z & (x2 !& x3) & x4);
END;
```

✍ ***Примітка.*** Необов'язкові пропуски між складниками у форматуванні формул покращують легкість для читання та перевірки.

Після цього текстовий файл набуває остаточного вигляду, наприклад, (d:\max2work\tutorial\2lab\200node.tdf):

```
200node.tdf - Text Editor
TITLE "Логічна функція";      % B2 > AHDL Template > Title
SUBDESIGN 200node             % B2 > AHDL Template > Subde
(
    x4, x3, x2, x1       : INPUT;
    y0, y1, y2, y3       : OUTPUT;
)
VARIABLE                          % B2 > AHDL Template > Va
    z                    : NODE;  % B2 > AHDL Template > No
BEGIN                             % B2 > AHDL Template > Lo
    z = x1 !& x2;                 % B2 > AHDL Template > Bo
    y0 = (x1 # x3 # !x4)!&(x2 # (x1&x4 !# x1$x4));
    !y1=(x1 !& !x2) & (!x2 !& x4) & (!(!x1 & !x3 & x4));
    y2=!((!x1 !# !x2) # (x1 !# x4) # (!x2 !# !x3));
    y3=(x1 !& z) !& !(z & (x2 !& x3) & x4);
END;
```

✍ ***Примітка.*** У TDF-файлі ключові слова виділяються синім кольором, імена стандартних макро- і мегафункцій – брунатним, дані для компілятора (у тому числі розділові знаки) – чорним, інформація, взята в лапки, і довільна інформація (у тому числі коментарі, оточені з обох боків знаками „%", які компілятор не сприймає) – зеленим. Якщо знаки набувають іншого забарвлення, це свідчить про те, що файл не включений до проекту або про синтаксичні (у сенсі мови AHDL) помилки, наприклад, пропущений розділовий знак (крапка з комою) між операторами. Змінити для певного файла призначені за замовчуванням кольори можна з меню Options > Color Palette.

3.2.3.7 Зберегти файл, перевірити його на синтаксичні помилки та виконати автоматичну ***компіляцію*** проекту: File > Project > Save & Check > OK в інформаційному віконці за відсутності помилок (інакше виправити) > Start у вікні компілятора > OK в інформаційному віконці за успішної трансляції > Закрити вікно компілятора.

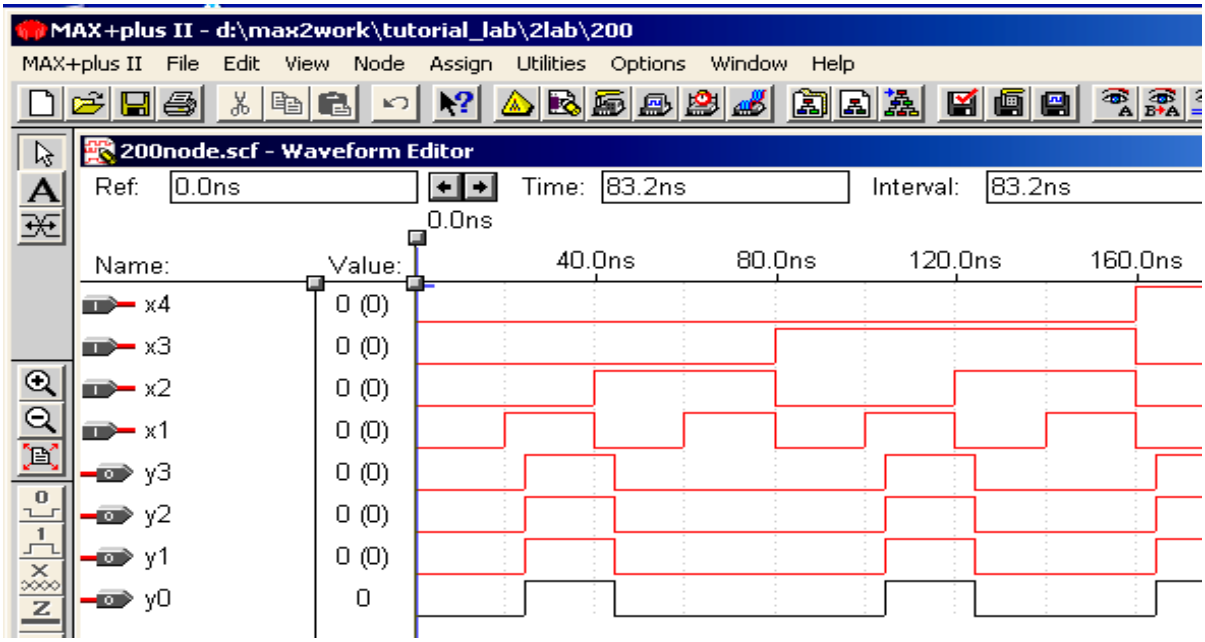
✍ ***Примітка.*** Компіляцію можна виконати так само, як у п. 2.2.2.

3.2.3.8 Виконати функціональне моделювання за допомогою редактора часових діаграм Waveform Editor (п. 2.2.3).

🖫 ***Приклад:*** MAX+plus II - d:\max2work\tutorial\2lab\200node.scf.

3.2.3.9 Виконати автоматичне порівняння часових діаграм за п. 2.2.3 з метою проаналізувати результати проектування: відкрити вікно одного з порівнюваних файлів (2XXnode.scf, діаграми якого вважатимуться початковими) > меню File > Compare... > у діалоговому віконці вибрати другий SCF-файл (2XX.scf) > OK.

Відтак діаграми початкового файла залишаються незмінними, а другого – накладаються на діаграми початкового файла і набувають червоного кольору; на ділянках, де вони не збігаються, діаграми початкового файла залишаються чорними. При цьому в полі Value рівні сигналів другого файла зазначаються в дужках.



☝ ***Примітка.*** Порівнюються вузли (сигнали) з однаковими іменами, тому діаграми додаткових вузлів початкового файла (якщо є) залишаються чорними, а другого файла – не відображаються.

3.2.4 Засвоїти синтез із використанням основних операторів логічного блоку на прикладі заданого варіанта дешифратора

3.2.4.1 Здійснити синтез дешифратора (додаток А, варіанти завдання 3,б) з використанням ***оператора таблиці відповідності Truth Table Statement:*** надати нове ім’я проекту 3XXtab і створити текстовий файл 3XXtab.tdf; відтак:

1) ввести заголовок (Title Statement);

2) ввести вхідні і вихідні вектори (Subdesign Section);

3) ввести логічний блок Logic Section (ці дії докладно описано в п. 2.2.5; приклад див. нижче);

4) між ключовими словами (BEGIN та END;) розташувати курсор (краще з відступом Tab) і ввести ***підсекцію таблиці відповідності*:** B2 > AT > Truth Table Statement та вставити в утворений таким чином шаблон мовою AHDL: а) заголовок таблиці (змінні у вигляді масиву або їх перелік через кому, потім, після відокремлювальної стрілки шаблона, так само функції у вигляді масиву або їх перелік через кому і в кінці залишити крапку з комою; б) тіло таблиці, кожний рядок якого відображає або значення змінних і функцій у вигляді кодів певної системи числення, або перелік логічних значень через кому (нагадує звичайну таблицю) і завершується крапкою з комою; у підсумку текстовий файл набуває вигляду як у наведеному нижче прикладі;

5) виконати компіляцію, функціональне моделювання та автоматичне порівняння утворених таким чином часових діаграм з п.3.2.3.9 з метою проаналізувати результати проектування.

🖫 ***Приклади:*** d:\max2work\tutorial\3lab\300tab і 300tabl – .gdf, .scf.

```
                                        %  Коментарі:                          %
TITLE "Дешифратор 2:4 (таблиця)";  % 1) Title Statement                  %
SUBDESIGN 300tab                    % 2) Subdesign Section               %
(                                       %  Варіант:                            %
    a[1..0]          : INPUT;       % (a1, a0          : INPUT;           %
    y[3..0]      : OUTPUT;          % y3, y2, y1, y0   : OUTPUT);         %
)
BEGIN                                   %     3) Logic Section              %
    TABLE                               %     4) Truth Table Statement      %
     a[] => y[];    % Варіант заголовку таблиці:   |a1,a0=>y3,y2,y1,y0; %
                    %  Варіанти тіла таблиці:       |                        %
     0   =>  1;     %  B"00"=>B"0001";|0 => H"1";|0,  0 =>0,  0,  0,  1;   %
     1   =>  2;     %  B"01"=>B"0010";|1 => H"2";|0,  1 =>0,  0,  1,  0;   %
     2   =>  4;     %  B"10"=>B"0100";|2 => H"4";|1,  0 =>0,  1,  0,  0;   %
     3   =>  8;     %  B"11"=>B"1000";|3 => H"8";|1,  1 =>1,  0,  0,  0;   %
    END TABLE;
END;
```

☝ ***Примітки:***

1. Ім’я в секції Subdesign обов’язково має збігатися з ім’ям проекту (див. п. 3.2.3.2).

2. Компілятор використовує лише дані, помічені в текстовому файлі чорним (у тому числі розділові знаки), тому ця його частина має точно відповідати операторному синтаксису (інакше замість компіляції видається повідомлення про помилки та їх локалізацію – зазначаються номери рядків тексту і що в них є помилкове). Ключові слова шаблону (синього кольору) відокремлюють підрозділи. Все інше не має значення: прогалини між символами, відступи і табуляції, порожні рядки, зокрема, після заголовку таблиці

визначають лише стиль оформлення, легкий для читання і розуміння. Проте кожний рядок коментарів (довільні написи зеленого кольору, а червоні заголовки тут виділено нами) має заключатися з обох боків знаками „%”.

3.2.4.2 Здійснити синтез дешифратора-демультиплексора (див. додаток А, варіанти завдання 3,б) з використанням ***умовного оператора If Then*** (якщо, тоді) у проекті 3XXif_then шляхом створення текстового файла (.tdf) і аналізу результатів так само, як у п.3.2.3.2 за винятком п.3.2.3.6, в якому ввести ***умовний оператор If Then*****:** B2 > AT > If Then Statement та вставити в утворений таким чином шаблон мовою AHDL:

а) між ключовими словами IF та THEN – умову (вираз або змінну), за істинності якої (тобто коли вона набуває значення логічної 1) є дійсними інструкції (функції у вигляді логічних значень або рівнянь через крапку з комою чи таблиці відповідності), що йдуть після слова THEN;

б) після ключового слова ELSE – інструкції, що є дійсними за невиконання зазначеної умови (тобто коли вона набуває значення логічного 0), а все інше слід видалити;

в) виконати так само компіляцію, функціональне моделювання та автоматичне порівняння утворених часових діаграм.

🖫 ***Приклад:*** d:\max2work\tutorial\3lab\300if_then.tdf, .scf.

```
TITLE "Дешифратор-демультиплексор 2:4 (умовний оператор)";
SUBDESIGN 300if_then
(
        a[1..0], G          : INPUT = VCC;
        y[3..0]             : OUTPUT;
)
BEGIN
  IF G THEN                          %          Якщо G=1, тоді              %
     y0=!a1&!a0; y1=!a1&a0; y2=a1&!a0; y3=a1&a0; % дійсні значення      %
  ELSE                                                % інакше (за умови G=0) %
     y[] = GND;                      %          y3=y2=y1=y0=0               %
  END IF;
END;
```

🖒 ***Примітки:***

1. Необов’язкова частина оператора ELSIF THEN (якщо ще, тоді) заповнюється так само, як і основна: між цими ключовими словами вставляється додаткова умова, а після – інструкції, які є дійсні за її виконання.

2. Текстові файли легко модернізувати: скопіювати (все або частину), непотрібне вилучити, а додаткове вставити, як звичайно, за допомогою шаблонів AHDL, не забуваючи привести у відповідність назву підпроекту.

3.2.4.3 Здійснити синтез дешифратора (див. додаток А, варіант завдання 3,а) з використанням ***оператора вибору Case*** (випадок) у проекті 3XXcase шляхом створення текстового файла (.tdf) і аналізу результатів так само, як у п. 3.2.3.2 за винятком п. 3.2.3.6, в якому ввести ***оператора вибору Case*****:** B2 > AT > Case Statement та вставити в утворений таким чином шаблон мовою AHDL:

а) між ключовими словами CASE та IS – вираз (зокрема, код чи набір змінних), від значення якого залежать вихідні функції;

б) після ключового слова WHEN перед стрілкою „=>" – конкретне значення виразу, а після неї – значення функцій (як у таблиці відповідності), відтак залишити крапку з комою; цей ланцюжок може повторюватися довільну кількість разів;

в) у необов'язковій частині WHEN OTHERS (коли інше) після стрілки – значення функцій за інших значень виразу (не перелічених у підпункті б);

г) виконати так само компіляцію, функціональне моделювання та автоматичне порівняння утворених часових діаграм;

д) створити символьний файл (так само, як у п. 3.2.2.3).

🖫 ***Приклад:*** d:\max2work\tutorial\3lab\300case.tdf, .scf, .sym.

```
TITLE "Неповний дешифратор з інверсними виходами 2:3 (оператор вибору)";
SUBDESIGN 300case
(
        a[1..0]                 : INPUT;
        yN[2..0]        : OUTPUT;
)
BEGIN
   CASE a[] IS                  % при наступних значеннях коду a1a0  маємо: %
    WHEN 0 => yN[]=H"6";  % коли 00, то y2y1y0=110 (тобто лише y0=0) %
    WHEN 1 => yN[]=H"5";   %         01             101            y1              %
    WHEN 2 => yN[]=H"3";   %         10             011            y2              %
    WHEN OTHERS => yN[]=X; %    коли (a1a0=3), то довільні значення  %
  END CASE;
END;
```

3.2.5 Засвоїти запровадження макрофункцій і символів у *текстовий* редактор на прикладі реалізації ЦКП виконанням завдання п. 2

3.2.5.1 Засвоїти включення макрофункції: у новому проекті 3XX_7macro створити текстовий файл (.tdf), відтак виконати такі дії (для наочності див. файл 3XX_7seg.gdf).

1) Ввести заголовок Title Statement (нижче пунктів подається приклад).

TITLE "7-сегментний індикатор на макрофункції дешифратора";

2) У секції Function Prototype Statement ввести прототип макрофункції дешифратора (можна скопіювати з довідки Help).

```
FUNCTION 74139 (g1n, b1, a1, g2n, b2, a2)
    RETURNS (y10n, y11n, y12n, y13n, y20n, y21n, y22n, y23n);
```

3) У секції Subdesign Section повторити ім’я проекту 3XX_7macro та оголосити вхідні і вихідні порти.

```
SUBDESIGN 300_7macro
(
    a1, a0              : INPUT;
    a, b, c, d, e, f, g : OUTPUT;
)
```

4) Ввести секцію змінних Variable Section.

```
VARIABLE
```

5) Вставити підсекцію оголошення зразка Instance Declaration і надати ім’я зразку (для наочності у прикладі d:\max2work\tutorial\3lab\300_7seg.gdf позначено: dc – ім’я зразка дешифратора) та через двокрапку ввести ім’я макрофункції (74139).

```
    dc  :  74139;
```

6) Ввести логічний блок Logic Section, між ключовими словами (BEGIN та END;) розташувати курсор (краще з відступом Tab), вставити підсекцію Boolean Equation та з'єднати рівняннями зразок функції з оголошеними портами INPUT і OUTPUT.

```
BEGIN
    dc.a1 = a0; dc.b1 = a1;
    a = dc.y11n; b = !dc.y10n; c = dc.y12n; d = dc.y10n;
    e = VCC;  f = VCC;  g = dc.y12n;
END;
```

☝ ***Примітки:***

1. Логічні зв’язки між виходами та входами макрофункції містяться в її прототипі (п. 2.1.4), тому в булевих рівняннях достатньо лише з'єднати входи і виходи зразка, що репрезентує макрофункцію в проектованому пристрої, з вхідними і вихідними портами пристрою. Входи і виходи зразка мають такі самі назви, як у макрофункції, але позначаються через крап-

ку після імені зразка. У лівій частині рівнянь ставляться невідомі величини, а в правій – відомі, задані вхідними портами INPUT або визначені у прототипі (у нас – y_i). У разі потреби, додаткові з'єднання виконуються через зовнішні відносно макрофункції логічні операції.

2. Імена макрофункції та її параметрів мають суворо відповідати формату, про який можна дізнатися з прототипу функції у довідці (Help > Old-Style Macrofunctions > Decoders > Ім'я макрофункції > AHDL Function Prototype) або з графічного файла (B2 по символу > Edit Ports/Parameters > Help on "Ім'я макрофункції" > AHDL Function Prototype).

7) Виконати компіляцію, функціональне моделювання і переконатися в правильності проектування.

🖫 ***Приклад:*** файли d:\max2work\tutorial\3lab\300_7macro.tdf, .scf.

3.2.5.2 Засвоїти включення символу: у новому проекті 3XX_7symbol створити текстовий файл (.tdf), відтак виконати такі самі дії, як і в п. 3.2.3.2 (для наочності див. файл d:\max2work\tutorial\3lab\ 3XX_7seg1.gdf) із такими відмінностями:

а) у п. 3.2.5.1 до шаблону секції Function Prototype Statement ввести із заздалегідь створеного текстового файла символу 3XXcase.tdf (п. 3.2.4.3) назву і параметри із секції підпроекту; наприклад, із файла d:\max2work\tutorial\3lab\300case.tdf

```
SUBDESIGN 300case
(
    a[1..0]              : INPUT;
    yN[2..0]     : OUTPUT;
)
```

вводимо до нашого файла, дотримуючись шаблона, дані:

```
FUNCTION 300case(a[1..0])
    RETURNS (yN[2..0]);
```

б) у секції змінних так само оголошуємо зразок з іменем символу, наприклад:

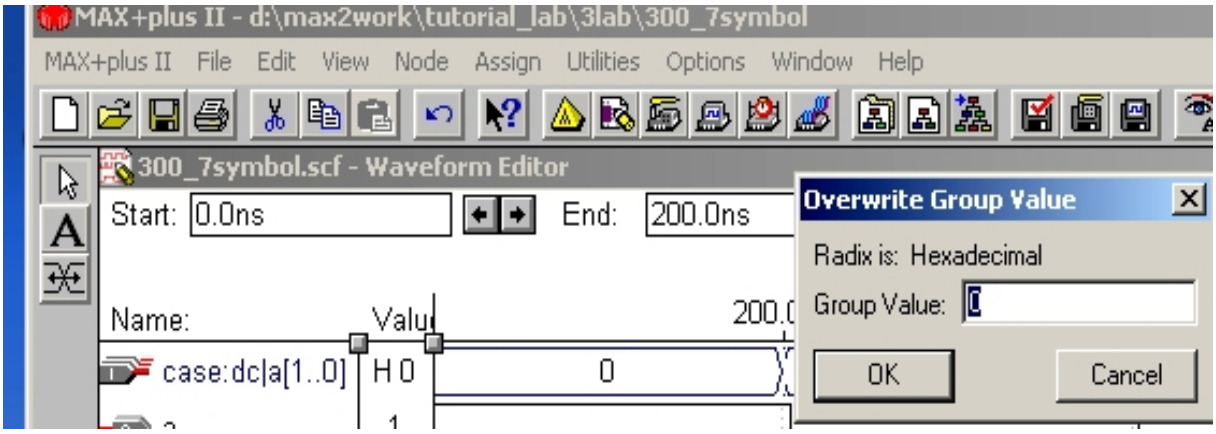
```
VARIABLE
    dc     : 300case;
```

в) до логічної секції вставляємо рівняння, дотримуючись імен входів і виходів символу, наприклад:

```
BEGIN
  dc.a[1..0]=a[1..0];
  a=dc.yN1; b=!dc.yN0; c=dc.yN2; d=dc.yN0; e=VCC; f=VCC; g=dc.yN2;
END;
```

🖫 ***Приклад:*** файли d:\max2work\tutorial\3lab\300_7symbol.tdf, .scf.

🖉 ***Примітка.*** У SCF-файлі дані зручно вводити у вигляді групи (шини): як звичайно виділити ділянку курсором, на палітрі натиснути кнопку групи G і в діалоговому віконці ввести значення коду в шістнадцятковій системі числення.



Контрольні питання та завдання

1. Як слід каскадувати дешифратори з метою збільшення розрядності?

2. Як пов’язані між собою кількість входів і виходів у повних шифраторі та дешифраторі?

3. Запишіть вирази для вихідних логічних функцій 1) шифратора 8×3, 2) дешифратора 3×8, 3) стробованого дешифратора 3×8 та побудуйте їх схеми.

4. Спроектуйте на дешифраторах перетворювач коду, варіант якого задано числом у табл. 3.4, де в лівій колонці наведено вхідний код, а у верхньому рядку – вихідний. Наприклад, варіанту 8 відповідає перетворювач коду 8421 → 8421доп. У табл. 3.4 подано: ДДК 8421, $8421_{доп}$, 8421+3, 2421, 7421, ХҐ – цифри 0...9 чотирирозрядного коду Ґрея, а також коди для відображення на семисегментному індикаторі знаків: Z_1 – цифр 0...9, Z_2 – літер латиниці A, b, C, d, E, F, H, U, O та Z_3 – літер кирилиці А, Г, Е, Н, О, П, Р, С, У.

Таблиця 3.4

Коди	Z_1	Z_2	Z_3	$X_Г$	742	242	8421 + 3	$8421_{доп}$
8421	1	2	3	4	5	6	7	8
$8421_{доп}$	9	10	11	12	13	14	15	–
8421+3	17	18	19	20	21	22	–	23
2421	25	26	27	28	29	–	30	31
7421	33	34	35	36	–	37	38	39
$X_Г$	41	42	43	–	44	45	460	47

4 МУЛЬТИПЛЕКСОРИ

Мета роботи: дослідження типових мультиплексорів, проектування ЦКП на мультиплексорах; групи і шини; мегафункції, основи їх настроювання, менеджер автоматичного створення мегафункцій, запровадження мегафункцій до графічного і текстового файлів та застосування їх для побудови цифрових пристроїв.

Домашнє завдання

Спроектувати ЦКП для реалізації логічної функції, заданої згідно з варіантом завдання (див. додаток А, варіанти завдання 4), на мультиплексорах різної розрядності та вибрати оптимальний варіант.

4.1 Стислі теоретичні відомості

4.1.1 Принцип побудови мультиплексорів

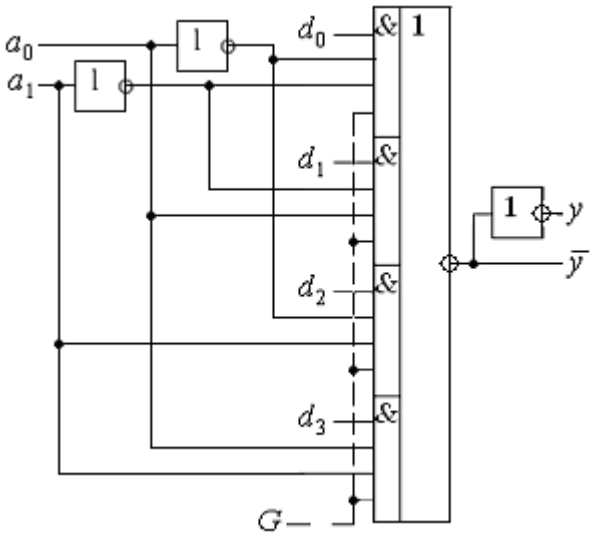
Мультиплексором (multiplexer, MUX) m : 1 (“з m в 1”) називається комутатор сигналів з m каналів в один, а стробований мультиплексор називається також мультиплексором-селектором (multiplexer-selector, MS).

Залежно від адреси $A = a_1a_0$ (рис. 4.1,а) до виходу y мультиплексора має надходити один із вхідних сигналів $d_0 \ldots d_3$, тобто

$$y = \overline{a_1}\,\overline{a_0}d_0 + \overline{a_1}a_0d_1 + a_1\overline{a_0}d_2 + a_1a_0d_3 = y_0d_0 + y_1d_1 + y_2d_2 + y_3d_3, \qquad (4.1)$$

де y_i - вихідні функції дешифратора.

i	a_1	a_0	y
0	0	0	d_0
1	0	1	d_1
2	1	0	d_2
3	1	1	d_3



а) б) в)

Рисунок 4.1 – Принцип побудови мультиплексора

Схему за виразом (4.1) можна реалізувати на елементі І-АБО-НЕ (рис. 4.1,б): за адресою a_1a_0 рівень логічної одиниці з’являється на всіх входах тільки одного з елементів І, що дозволяє проходити відповідному

вхідному сигналові d_i до виходу. Стробовий вхід G мультиплексора-селектора (пунктир на рис. 4.1, б) при застосуванні елементів з трьома станами (рис. 4.1, в) розширює функціональні можливості схеми.

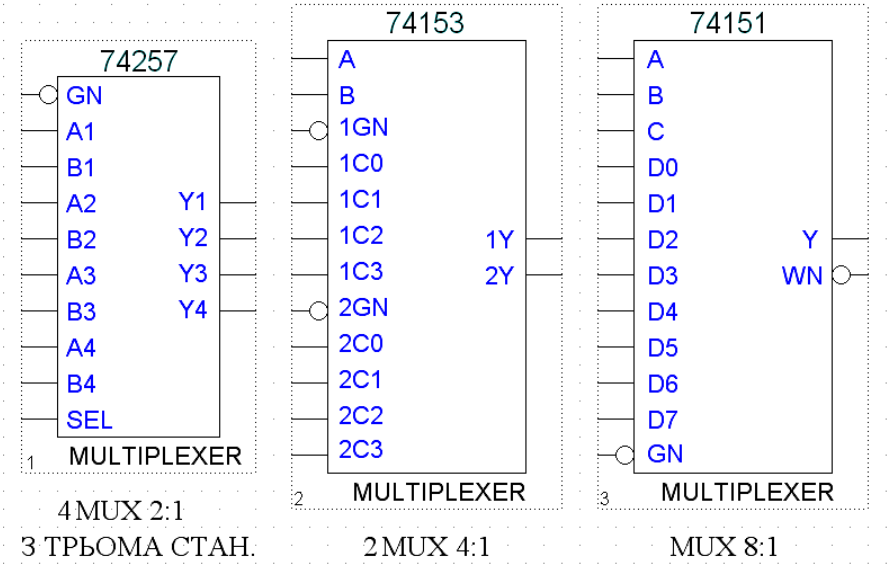


Рисунок 4.2

Таким чином, мультиплексор має k адресних, $m = 2^k$ інформаційних входів та один вихід (або два взаємоінверсні виходи). Серед ІС мультиплексорів, що випускаються серійно (рис. 4.2), найпоширенішими є з розрядністю адреси k = 1, 2, 3. Вони забезпечують перемикання m = 2, 4, 8 входів до одного виходу.

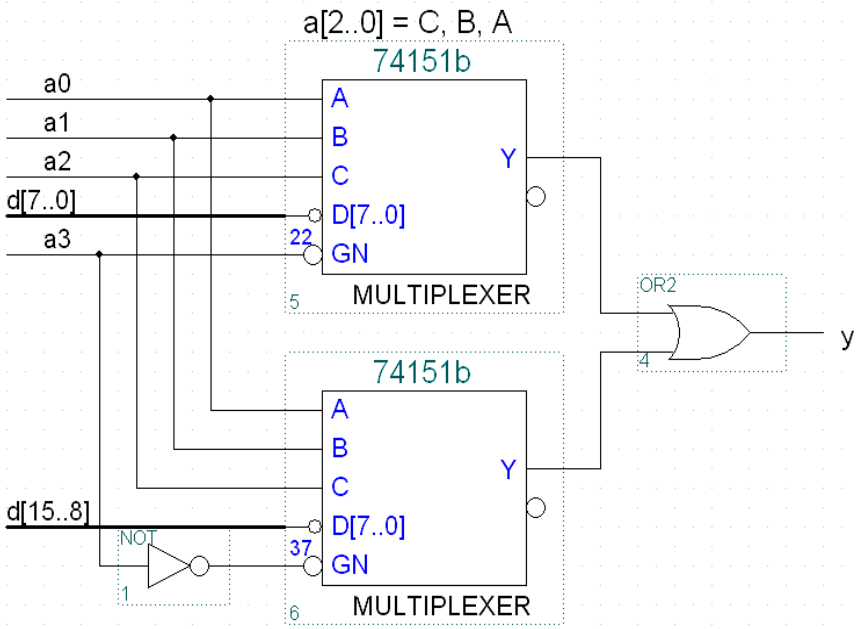


Рисунок 4.3

Розширення розрядності здійснюють шляхом каскадування мультиплексорів. З цією метою старший розряд адреси (рис. 4.3) з'єднують зі стробовим входом одного мультиплексора безпосередньо, а другого – через інвертор і виходи об'єднують через елемент АБО. Коли зазначений розряд $a_3 = 0$, до виходу комутуються молодші біти $d_0 ... d_7$, а при $a_3 = 1$ – старші $d_8 ... d_{15}$. Є і інші способи каскадування.

4.1.2 Групи і шини

4.1.2.1 Одновимірні групи

Низка однотипних елементів і їх назв для стислості може бути оголошена у вигляді окремого модуля, який позначається символом групи. Наприклад, масив адресних сигналів і відповідних їм портів a3, a2, a1, a0 можна зобразити групою a[3..0]. Згідно із синтаксисом мови AHDL групу становить символічне ім'я (до 32 символів) і діапазон групи (до 256 складників, тобто бітів), який подається у квадратних дужках: старший біт, дві крапки, молодший біт. Ім'я біта може містити до трьох цифр. Після оголошення діапазону певної групи її можна позначати в наступних виразах стенографічно, без зазначення цього діапазону: a[].

Прирівнювання груп означає, що складники з'єднуються відповідно до позицій у групах, наприклад, рівність груп a[3..0] = sel[7..4] відповідає з'єднанню елементів a3 = sel7, a2 = sel6, a1 = sel5, a0 = sel4. Отже, кількість бітів у лівій і правій частинах рівняння має бути однаковою. Проте всю групу можна приєднати до одного вузла, тоді всі її складники з'єднуються з цим вузлом. Якщо, наприклад, d[6..4] = GND, то d6 = GND, d5 = GND, d4 = GND. Але якщо в позначенні група прирівняна до десяткової одиниці: d[6..4] = 1, то компілятор розширює число у двійковій системі до розміру групи B"001", отже молодший розряд з'єднується з джерелом живлення, а інші – із землею: d[6..4] = 1 = B"001", тому d6 = GND, d5 = GND, d4 = VCC.

☝ ***Примітки:***

1. Часто використовувану змінну ***x*** у *текстовому* редакторі не можна застосовувати без індексу, у тому числі для позначення групи типу x[], бо ця літера є зарезервованою (невизначений стан).

2. Своєрідний узагальнений запис типу data[WIDTH-1..0] означає групу з кількості елементів WIDTH (ширина), розташованих у порядку від старшого біта до молодшого. Такий запис є легітимним, якщо параметр WIDTH визначений попередньо в секції **Parameters Statement.** Наприклад, якщо WIDTH = 8, то групу data[WIDTH-1..0] можна зобразити у вигляді data[7..0].

3. Діапазон з одного складника, наприклад, a[2] є еквівалентний запису a2.

4. Звичайний порядок бітів у діапазоні (від старшого до молодшого) можна змінити в секції **Options Statement** шаблону AHDL Template.

5. Діапазон можна позначати числами в будь-якій системі числення або арифметичними виразами, наприклад, a[B"11"..B"00"] або a[log2(8)..0].

4.1.2.2 Послідовні групи

Застосовується також послідовне позначення групи з переліку елементів, відокремлених комами і взятих у круглі дужки (у текстовому реда-

кторі), наприклад, масив адресних (селекторних) входів: (d, c, b, a) = (a3, a2, a1, a0) = a[3..0]. При комбінованому запису групи в межах круглих дужок можуть бути елементи, що є, у свою чергу, також групами. Таке зображення є зручним для позначення портів зразка функції, наприклад, вхідні порти мультиплексора-селектора 16:1 зручно позначити групою мх.(g, sel[3..0], d[15..0]). Крім того, діапазон групи можна поділити на піддіапазони, наприклад, діапазон d[15..0] еквівалентний формі (d[15..9]), d8, d[7..0]).

4.1.2.3 Двовимірні групи

Дводіапазонні або двовимірні групи складаються із символічного імені і двох діапазонів у квадратних дужках. Наприклад, група a[3..1][1..0] аналогічно двовимірній матриці складається із шістьох елементів і є еквівалентною послідовному зображенню у вигляді множини складників (ai_j):

$$a[3..1][1..0] = \begin{array}{c} \\ 3 \\ 2 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 1 & 0 \end{array} \\ \left\| \begin{array}{cc} a3\_1 & a3\_0 \\ a2\_1 & a2\_0 \\ a1\_1 & a1\_0 \end{array} \right\| \end{array} = (a3\_1, a2\_1, a1\_1, a3\_0, a2\_0, a1\_0).$$

Окремий вузол такої групи може позначатися як a2_1 або a[2][1]. Після оголошення діапазонів групу можна позначати стисло: a[][].

4.1.2.4 Шини

Шиною передається множина сигналів від 2 до 256 бітів між компонентами проекту. У графічному редакторі шина позначається грубою лінією (рис. 4.4, а), а в текстовому редакторі (рис. 4.4, б) і редакторі часових діаграм (рис. 4.4, в) – групою. Назви шин можуть бути одно- та дводіапазонні або послідовні і підпорядковуватися правилам, викладеним у п. 4.1.2. Зазначимо тут особливості подання шин у графічному редакторі.

Приєднання шини до примітива утворює масив примітивів, наприклад, масив з чотирьох портів a3, a2, a1 та a0 на рис. 4.4, а). Проте порт можна позначати лише однодіапазонною групою, не припускаються дводіапазонні та послідовні групи в назвах портів.

З'єднувати шинами можна групи лише однакового діапазону, наприклад, з'єднання на рис. 4.4, а) означає, що мультиплексор має 4 адресні входи в групі sel[]. Нерозгалужену шину, як у цьому прикладі, можна не іменувати: розряди масивів a[] і sel[] з'єднуються відповідно до їх позицій у групах. Проте шина, що живить множину примітивів (рис. 4.4, г), має іменуватися, бо діапазон групи в її назві визначає їх кількість. У цьому прикладі перші входи 16-ти елементів I з'єднано із сигналами d15…d0, а на другі входи паралельно подано сигнал C.

У випадку розгалужених шин мають іменуватися як шини, так і окремі лінії до і після місця розгалуження (рис. 4.4, г). При цьому імена шин і ліній мають збігатися з назвами з'єднуваних ними портів і вузлів (входів та виходів), тому, у разі потреби, імена узгоджують буферами Wire (рис. 4.4, д). Так, шина, що виходить з масиву портів x[4..1], має ту саму

назву x[4..1], а шина, що з'єднує 8 інформаційних входів мультиплексора data[] з джерелами, також називається data[7..0]. Тому імена з'єднуваних ліній x3 і data6 узгоджуються буфером Wire.

Примітки:

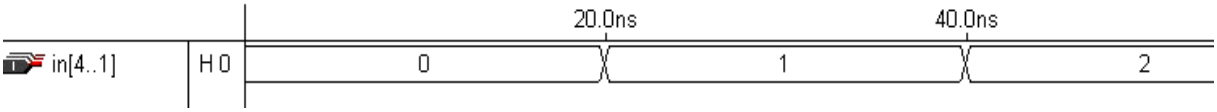
1. У графічному редакторі послідовні групи позначаються без дужок, наприклад, група у вигляді: data7, data[5..0] на рис. 4.4, д).

2. Стенографічне позначення типу sel[] є припустимим тільки в текстовому редакторі, а в графічному редакторі для шин завжди застосовується повне позначення групи типу sel[2..0].

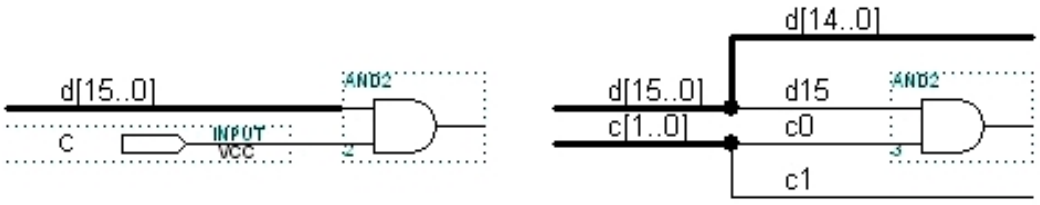
а)

in[4..1] : INPUT;

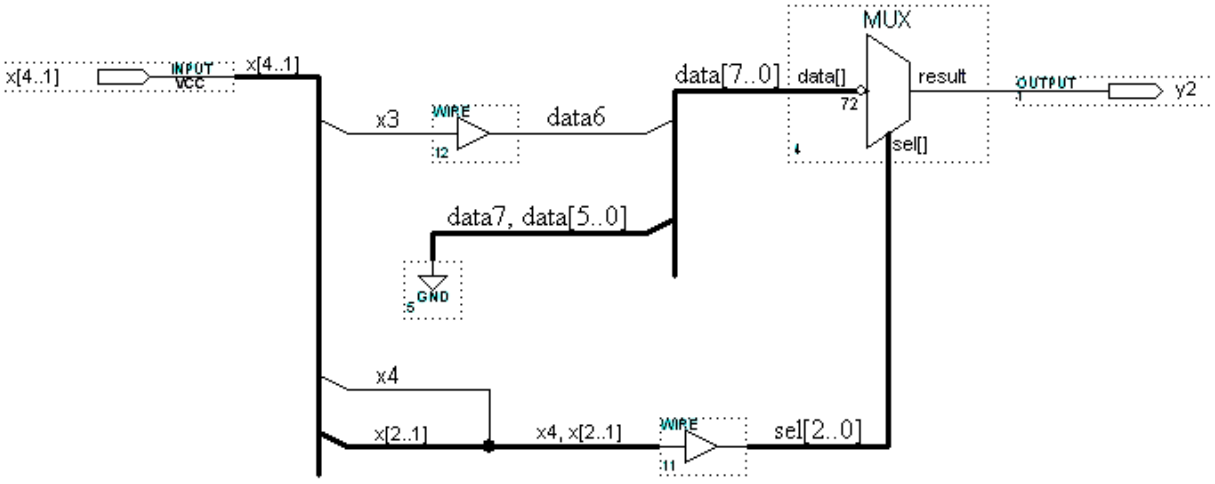
б)



в)



г) д)



е)

Рисунок 4.4

3. Шини пов’язані з вузлами (компонентами) тільки іменами, тому з’єднання іменованих шин і ліній можна не зображати взагалі, їх зображають лише для наочності схеми. Отже, і точки з’єднань показувати необов’язково. Виняток становлять шини, на яких без точок з’єднань утворюються різні імена, як, наприклад, на нижній лінії рис. 4.4, е) без точки було б дві назви: “x[2..1]” і “x4, x[2..1]”.

4.1.3 Мегафункції

4.1.3.1 Означення мегафункції

Мегафункцією називається структурна модель у вигляді стандартного блоку високого рівня, що реально не існує, але після попереднього настроювання може використовуватися як компонент схеми поряд із примітивами і макрофункціями. Від макрофункції мегафункція відрізняється гнучкішими можливостями для настроювання параметрів і тим, що сигнали з’єднуються з нею, як правило, групами й шинами (вузли деяких різновидів макрофункцій також зображено шинами). Мегафункції містяться в директорії \maxplus2\max2lib\mega_lpm бібліотеки параметризованих модулів (LPM – Library of Parameterized Modules). До графічного файла мегафункція вставляється як звичайний елемент, але із зазначеної директорії, а до текстового редактора – за допомогою файла включення (.inc) оператором Include Statement.

Мегафункції поділяються на три основні і одну додаткову групи. Відомості про потрібну мегафункцію можна дістати з довідки: меню Help > Megafunctions/LPM > вибрати з групи і натиснути ім’я мегафункції. Схарактеризуємо стисло деякі різновиди мегафункцій, наведених у файлі 4libre_mega.gdf.

4.1.3.2 Логічні функції

Мегафункції логічних елементів подано в категорії Gates (ворота).

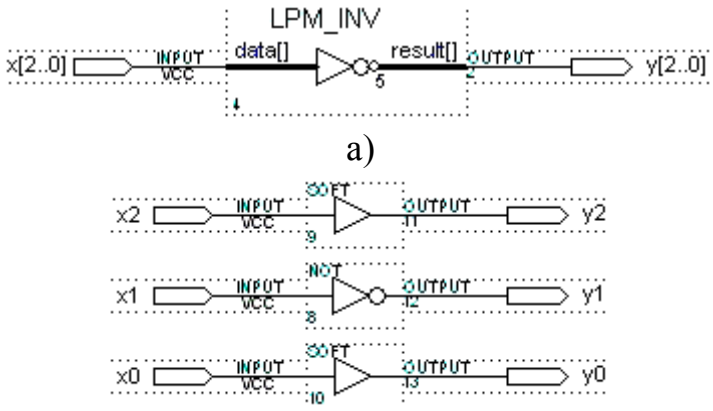
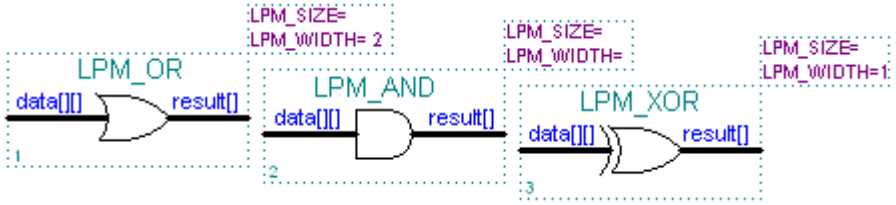


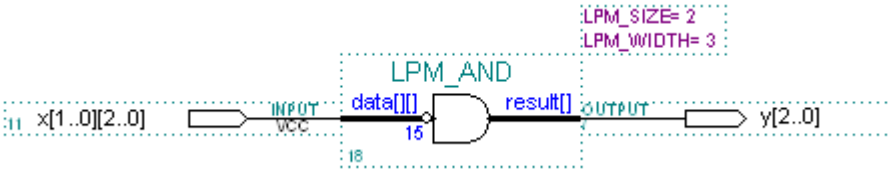
Рисунок 4.5

Мегафункція lpm_inv (рис. 4.5, а) є одновимірним масивом інверторів з параметром LPM_WIDTH, який визначає кількість входів і виходів. Якщо задати діапазон групи LPM_WIDTH= 3, еквівалентна схема набуває вигляду як на рис. 4.5, б). Інвертування входів і виходів здійснюється так само, як і в макрофункцій. Від-

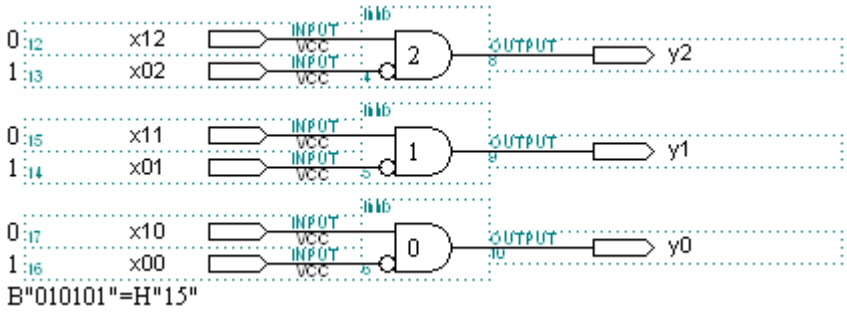
мінність полягає тільки в тому, що інверсії задаються для всієї групи кодом (на символі відображається шістнадцятковий код, а вводити можна також двійковий або десятковий код). У цьому коді розряд, що інвертується, позначається одиницею, а який залишається без інверсії – нулем. У прикладі (див. рис. 4.5) подвійним запереченням (відображено кружком з шістнадцятковим числом 5 на виході символу мегафункції) усунуто інверсію розрядів x_2 та x_0 кодом $101_2 = 5_{16}$.



а)



б)



в)

Рисунок 4.6

Усі інші логічні елементи, що є багатовходовими, репрезентовано мегафункціями lpm_or, lpm_and та lpm_xor (рис. 4.6, а). Такі мегафункції характеризуються параметрами: LPM_WIDTH – кількість елементів масиву, отже, кількість вхідних шин і розрядність вихідної шини, наприклад, для мегафункції lpm_and при LPM_WIDTH = 3 (рис. 4.6, б) маємо 3 елементи з виходами y_2, y_1, y_0 (рис. 4.6, в); LPM_SIZE – кількість входів кожного елемента, отже і розрядність вхідних шин, наприклад, при LPM_SIZE = 2 маємо двовходові елементи зі входами x_{1j}, x_{0j}, де j = 2, 1, 0 (див. рис. 4.6, б), в). Шляхом інвертування входів шістнадцятковим кодом H”15” отримуємо три елементи заборони.

4.1.3.3 Дешифратор

Мегафункція дешифратора lpm_decode (рис. 4.7) характеризується двома основними параметрами (відображаються на символах мегафункцій брунатним кольором у правому верхньому куту за ввімкненої опції: меню

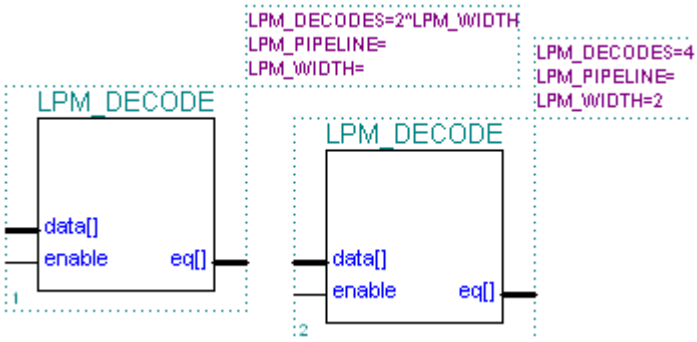


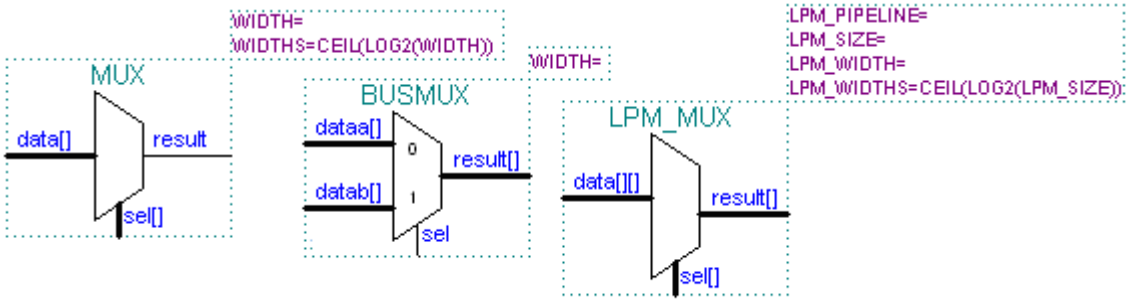
Рисунок 4.7

Options > Show Parameters або Show All): LPM_WIDTH – розрядність двійкового вхідного коду та LPM_DECODES – розрядність унітарного вихідного коду 2^ LPM_WIDTH (для повного дешифратора) або <2^ LPM_WIDTH (для неповного дешифратора).

Для прикладу на рис. 4.7 зображено настроєний дешифратор 2:4.

4.1.3.4 Мультиплексори

Мегафункція звичайного мультиплексора mux (рис. 4.8, а) параметром WIDTHS – кількістю адресних входів sel[] – визначає припустиму розрядність WIDTH ≤ 2^WIDTHS вхідного коду data[]. Так, настроюванням WIDTH = 16 та WIDTHS = CEIL(LOG2(WIDTH)) = 4 дістанемо мультиплексор 16:1, а зміною величини WIDTH – мультиплексор з довільною кількістю входів.



а) б) в)

Рисунок 4.8

Мегафункція мультиплексора – перемикача шин busmux (рис. 4.8, б) з одним адресним входом при sel = 0 з'єднує з виходом result[] вхідну шину dataa[], а при sel = 1 – шину datab[]. Розрядність вхідних і вихідної шин визначається параметром WIDTH.

Проте базовою є мегафункція lpm_mux (рис. 4.8, в), що поєднує в собі властивості двох вищезгаданих, але з розрядністю адресних входів LPM_WIDTHS, яка визначає кількість вхідних шин LPM_SIZE ≤ 2^ LPM_WIDTHS розрядністю LPM_WIDTH кожна. Двовимірний масив вхідних даних має розмір data[LPM_SIZE-1..0][LPM_WIDTH-1..0], а ширина вихідної шини result[LPM_WIDTH-1..0] збігається з розрядністю вхідних шин LPM_WIDTH.

Якщо призначити LPM_WIDTH = 1, то така мегафункція перетво-

рюється у звичайний мультиплексор (див. рис. 4.8, а), який перемикає одну з LPM_SIZE ліній до однодротового виходу. Але згідно з форматом її прототипу позначати вхідні лінії все одно потрібно двовимірним кодом data[][], в якому другий складник є 0. Настроювання значно спрощується з використанням спеціальної підпрограми – менеджера створювання мегафункцій. Ця методика докладно викладається в лабораторному завданні.

Примітка. Інші параметри мегафункцій дешифратора і мультиплексора, що застосовуються в послідовнісних пристроях, зокрема, LPM_PIPELINE (конвеєрний режим) не є обов'язковими і під час настроювання їх можна не визначати.

4.2 Лабораторне завдання

4.2.1 Дослідити типові мультиплексори

4.2.1.1 За принциповою електричною схемою та осцилограмами сигналів (4prim.gdf, .scf) ***мультиплексора з трьома станами виходу на логічних елементах*** скласти таблицю відповідності і рівняння вихідної функції, виміряти затримку вихідних імпульсів. (У звіті навести схему, умовне графічне позначення за ДСТУ, таблицю відповідності, рівняння вихідної функції, часові діаграми, затримку, стислі пояснення принципу дії такого ЦКП з трьома станами виходу).

4.2.1.2 Ознайомитися з макрофункціями стандартних мультиплексорів серії 74, у тому числі з трьома станами виходу (файл d:\max2work\tutorial\4lab\4libr_macro.gdf, рис. 4.1...4.4). Розглянути принципові електричні схеми (B** по символу) та таблиці відповідності (клацнути піктограмою по символу) різновидів мультиплексорів. (У звіті навести два-три символи з різних груп, пояснити призначення та особливості входів і виходів).

4.2.1.3 Ознайомитися з дібраними елементами бібліотеки мегафункцій (файл d:\max2work\tutorial\4lab\4libr_mega.gdf): 1) буфер, 2) логічні функції, 3) дешифратор, 4) мультиплексори та пояснювальними прикладами 1...10. Розглянути виконувані функції в текстовому (B** по символу) та довідковому (клацнути піктограмою по символу) файлах, у довідці звернути увагу на приклади (Example) і розділ функції (Function) та дати тлумачення сенсу кожної з чотирьох груп мегафункцій і їхніх параметрів. (У звіті навести принаймні по одному символу з кожної групи, пояснити призначення та особливості входів і виходів та дати стисле тлумачення основних параметрів).

4.2.2 Застосувати *макрофункції* мультиплексорів для реалізації заданого варіанту (XX=01, 02, …) логічної функції *y*

4.2.2.1 У графічному файлі проекту 4XXgr_macro ***дібрати макрофункції,*** потрібні для реалізації логічної функції на мультиплексорах різної

розрядності, настроїти їх, доповнити схеми іншими елементами, виконати компіляцію і функціональне моделювання, переконатися в правильності проектування та вибрати оптимальний варіант схеми на ІС жорсткої логіки.

🖫 ***Приклад:*** d:\max2work\tutorial\4lab\400gr_macro.gdf, .scf (схеми 1...4).

☝ ***Примітки:***

1. Відмінність застосування груп полягає в тому, що масиви примітивів вводимо у вигляді одного елемента, ім'я якого позначається діапазоном групи, а з'єднувальні шини зображаємо грубими лініями (тип лінії вибирається з верхньої панелі інструментів).

2. У випадку розгалужених шин треба вводити їх імена, а також імена з'єднаних з ними ліній: курсором (інструменти "стрілка" або літера "A") на шині (лінії) позначити миготливу точку і ввести текст (або ввести його у вільному місці і перетягнути інструментом "стрілка" в потрібну позицію). При цьому назви розгалужених шин (ліній) мають збігатися з назвами з'єднуваних ними вузлів (входів, виходів), інакше узгоджуємо імена за допомогою буферів Wire, які ресурсу не потребують, бо реально не існують.

3. Компіляцію і моделювання доцільно виконувати кроками: спочатку змоделювати схему 1, відтак доповнити її схемою 2 і повторити моделювання і т. д.

4.2.2.2 У текстовому файлі проекту 4XXtx_macro ***реалізувати один з варіантів***, еквівалентний схемам за п. 4.2.2.1.

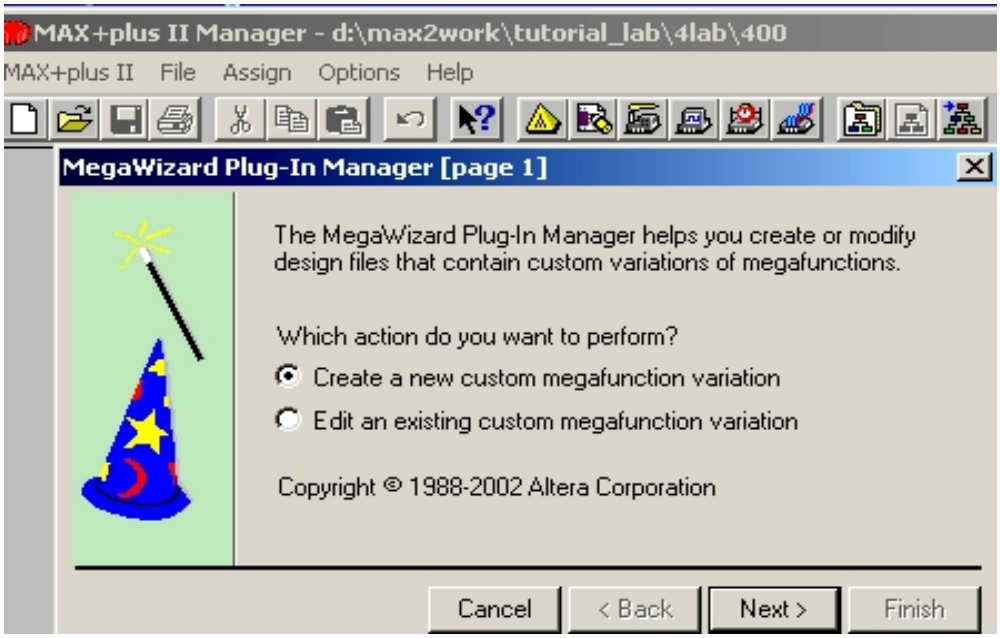
🖫 ***Приклад:*** d:\max2work\tutorial\4lab\400tx_macro.tdf, .scf.

☝ ***Примітка.*** У текстових файлах літера „x" зарезервована (для невизначеного стану), тому без індексу на кшталт x[4..1] її не можна використовувати; отже, для позначення груп слід вживати імена типу xin[], xi[], d[] тощо.

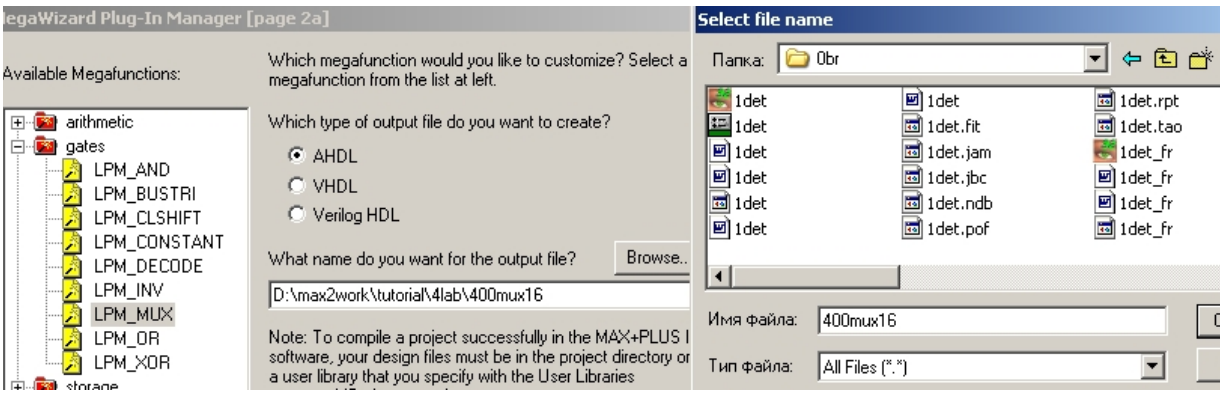
4.2.3 Засвоїти основи автоматичного створення різновидів мегафункцій за допомогою менеджера MegaWizard Plug-In Manager на прикладі синтезу мультиплексора найбільшої розрядності N, необхідного для реалізації заданого варіанту функції (див. d:\max2work\tutorial\4lab\ 400gr_macro.gdf, схема 1).

4.2.3.1 Запустити менеджер мегафункцій (незалежно від наявності відкритого якогось файла): меню File > MegaWizard Plug-In Manager і відповісти на запитання в діалогових вікнах.

4.2.3.2 На сторінці 1 (page 1) ***вибрати дію, яку треба виконати*** – створити (Create …) новий різновид мегафункції або редагувати (Edit …) існуючий різновид > Next.



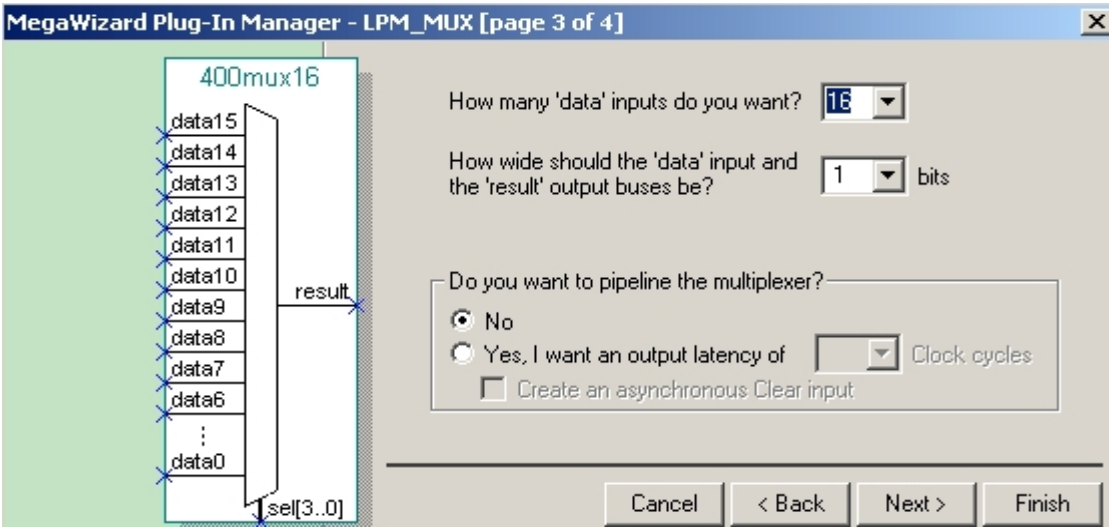
4.2.3.3 На сторінці 2а зліва зі списку доступних мегафункцій ***вибрати*** (натиснути +) ***категорію*** (gates) ***і функцію*** (LPM_MUX), а справа – тип мови для створюваного файла (AHDL) і до віконця What name … ввести ім'я файла 4XXmuxN, але якщо директорія не відповідає потрібній, треба її змінити: натиснути кнопку Browse (огляд) і в діалоговому вікні Select file name прокруткою по черзі встановити потрібні папки, ввести ім'я файла і зберегти (у віконці відобразиться директорія з ім'ям) > Next.



Примітка. У діалоговому вікні Select file name, у разі потреби, можна створити нову папку за допомогою піктограми.

4.2.3.4 На сторінці 3 в графі How many 'data' inputs ...? ***ввести кількість інформаційних входів*** (входів даних), у графі How wide should 'data' input and 'result' output buses be? – ширину в бітах кожного з них і виходу (залишити 1, але для перемикача шин ввести потрібну їх ширину) > Finish.

4.2.3.5 Відкрити утворені файли нового різновиду мегафункції із заданими конкретними параметрами, зокрема, 4XXmuxN.tdf, .inc, .sym (тут N – кількість інформаційних входів мультиплексора).



🖫 ***Приклад:*** файли d:\max2work\tutorial\4lab\400mux16.tdf, .inc, .sym, 400mux16_inst.tdf.

☝ ***Примітки:***

1. З метою відредагувати існуючий різновид мегафункції відповідаємо на запитання: на сторінці 2b вибираємо тип файла і імена файла та мегафункції, на сторінці 3 коригуємо параметри і або завершуємо редагування (Finish), або ще змінюємо директорію на сторінці 4.

2. Різновид мегафункції можна створити не тільки автоматично, але й ручним способом як подано в останньому необов'язковому пункті роботи.

4.2.3.6 Відредагувати символьний файл для зручності користування ним у графічних проектах виконанням таких дій.

а) Створити новий графічний файл (.gdf) проекту 4XXmN (послідовністю наведених піктограм або командами з меню): визначити ім'я проекту, відкрити новий графічний файл та присвоїти йому ім'я.

б) Ввести до графічного файла символ створеного різновиду мегафункції, вхідні та вихідний порти та згрупувати вхідні сигнали в шину.

в) Зберегти файл, перевірити його на помилки і скомпілювати проект піктограмою та керуванням вікна компілятора, яке з'являється автоматично (або командами з меню).

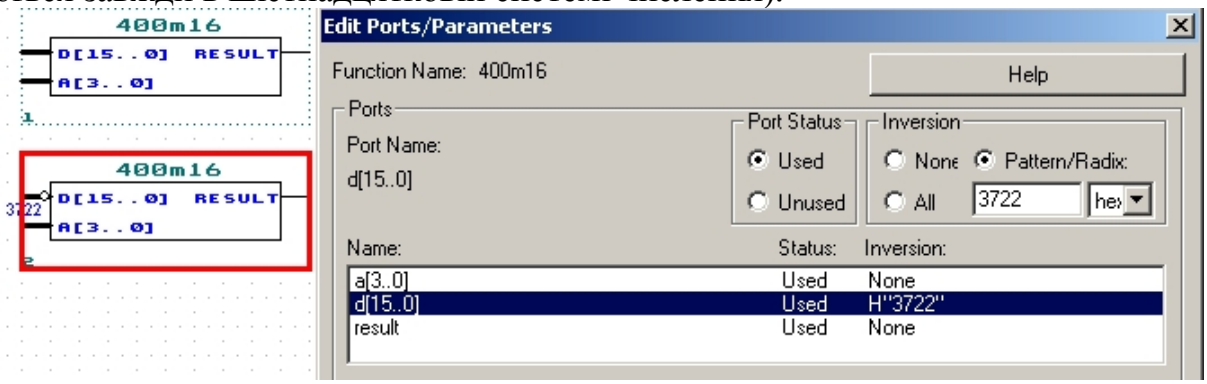
г) Командою з меню File > Create Default Symbol створити символ мегафункції, в якому виводи зображено компактно, у вигляді шин, та відкрити його для перегляду.

🖫 ***Приклад:*** файл d:\max2work\tutorial\4lab\400m16.gdf, .sym.

4.2.4 Засвоїти основи настроювання мегафункцій і їх застосування у *графічному* редакторі (на прикладі варіантів п. 4.2.2.1).

4.2.4.1 До графічного файла (.gdf) нового проекту 4XXgr_mega ***вставити символ створеного за допомогою менеджера MegaWizard Plug-In Manager*** і відредагованого різновиду мегафункції 4XXmN.sym (зі списку Symbol Files діалогового вікна Enter Symbol).

4.2.4.2 Настроїти символ: викликати діалогове вікно редагування (B2 по символу > Edit Ports/Parameters) і призначити порти у закладці Ports так само, як і для макрофункцій; відмінності полягають у тому, що тепер користуємося не нумерацією виводів IC, а іменами портів (віконце Name) та для інвертування окремих розрядів шин застосовуємо числа в одній із систем числення (натиснути Patten/Radix у закладці Inversion та ввести до віконця код інвертування). Користуючись, наприклад, еквівалентною схемою на кшталт 400rys.gdf, з'ясовуємо, що всі інформаційні входи IN[15..0] мультиплексора за схемою 1 з метою спрощення можна заземлити, якщо зінвертувати ті з них, на які слід подати логічну 1. У вікні редагування бітам, що інвертуються, надається значення 1, а тим, що залишаються неінвертованими, – значення 0, тому код інвертування становитиме у цьому прикладі IN[15..0] = B"0011 0111 0010 0010" = H"3722", який і вводимо до віконця Patten/Radix в одній з чотирьох систем числення, основу якої вибираємо прокруткою. Після закриття (OK) діалогового вікна результати редагування (див. виділений символ) відобразяться кружечком інверсії з кодом інвертування (на символі код подається завжди в шістнадцятковій системі числення).



4.2.4.3 Увести інші елементи схеми, потрібні для реалізації заданої функції, виконати компіляцію та функціональне моделювання, переконатися в правильності проектування.

🖫 ***Приклад:*** d:\max2work\tutorial\4lab\400gr_mega.gdf (схема 1), .scf.

4.2.4.4 У тому ж графічному файлі виконати п. 4.2.3, 4.2.4.1...4.2.4.3 на мультиплексорах меншої розрядності.

🖫 ***Приклад:*** d:\max2work\tutorial\4lab\400gr_mega.gdf, .scf (схеми 2, 3).

☝ ***Примітки:***

1. У графічному файлі з'єднувати можна шини тільки однакового розміру: наприклад, на схемі 1 чотири порти з'єднано з чотирма адресними входами чотирирозрядною шиною. У випадку нерозгалужених шин їх можна не йменувати, бо за угодою розряди з'єднуються за старшинством, тому номери розрядів не обов'язково будуть однаковими: a3=x4, a2=x3, a1=x2, a0=x1.

2. У п. 4.2.4.4 достатньо виконати один-два варіанти схеми на мультиплексорах різної розрядності; п. 4.2.4.5...4.2.4.7 є факультативні, необов'язкові.

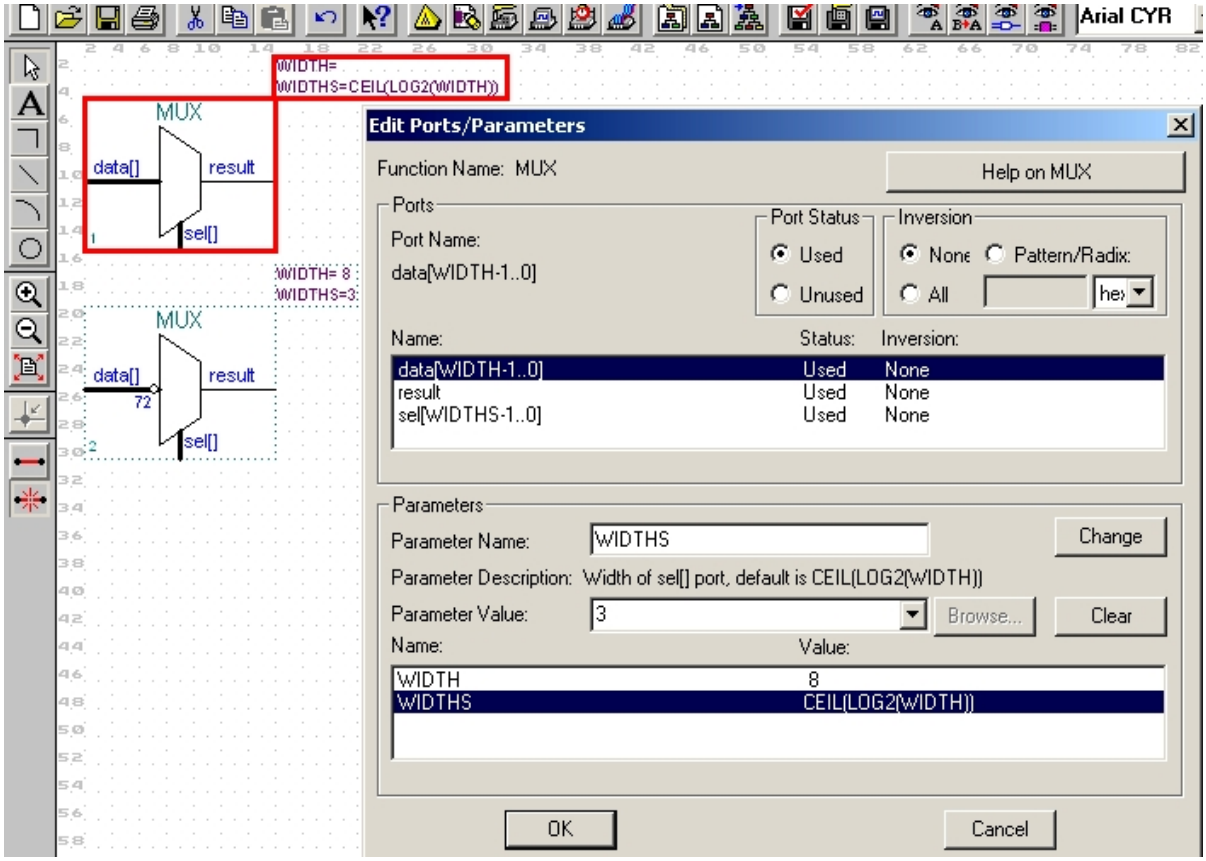
4.2.4.5 У тому ж графічному файлі виконати п. 4.2.4.1...4.2.4.3 на мультиплексорі 2:1, користуючись його символом, створеним безпосередньо менеджером MegaWizard Plug-In Manager (без спрощення за п. 3.2.3.6).

🖫 ***Приклад:*** d:\max2work\tutorial\4lab\400gr_mega.gdf (схема 4), .scf.

4.2.4.6 У тому ж графічному файлі виконати завдання п. 4.2.2.1 на мегафункції мультиплексора **mux** за ручного її настроювання (достатньо один варіант мультиплексора).

а) Вставити символ мегафункції до файла звичайним чином, але з бібліотеки мегафункцій d:\maxplus2\max2lib\mega_lpm і викликати діалогове вікно редагування (B2 по символу > Edit Ports/Parameters), в якому для мегафункції доступними є обидві закладки (Ports і Parameters).

б) Зручніше спочатку призначити параметри в закладці Parameters: виділяємо параметр у віконці Name/Value, ім'я якого з'являється у віконці Parameter Name, вводимо його значення Parameter Value (прокруткою або з клавіатури) і фіксуємо в графі Value кнопкою Change (кнопкою Clear можна видалити значення, а відтак знов його ввести); повторюючи цю процедуру, призначаємо всі параметри, доступні в графі Name. Порти призначаємо в закладці Ports так само, як і в п. 4.2.4.2 (результат показано на невиділеному символі наведеного нижче рисунку).



в) Увести інші елементи схеми (за необхідності, розщепити шини і перейменувати порти), виконати компіляцію та функціональне моделювання, переконатися в правильності проектування.

🖫 ***Приклад***. Для реалізації заданої логічної функції, наприклад, на мультиплексорі 8:1 вставляємо до файла мегафункцію mux (див. файл d:\max2work\tutorial\4lab\400gr_mega.gdf, схема 5). Викликаємо діалогове вікно редагування і в закладці Parameters, у віконці Name/Value, виділяємо параметр WIDTH – ширину інформаційної шини, вводимо значення 8 у віконці Parameter Value і фіксуємо його в графі Value кнопкою Change. Відтак переходимо до наступного параметра WIDTHS – ширини адресної (селекторної – S) шини, яка для мультиплексора 8:1 дорівнює трьом, тому у віконці Parameter Value можна або залишити CEIL(LOG2(WIDTH)), або для наочності ввести значення 3 і зафіксувати кнопкою Change.

Відтак редагуємо порти в закладці Ports (див. d:\max2work\tutorial\4lab\400rys.gdf, схема 2): вихід result і адресні входи sel[WIDTHS-1..0] залишаємо незмінними (Used – використовується, None – без інверсії), а розряди інформаційної шини data[WIDTH-1..0] інвертуємо введенням Patten/Radix > 72 hex (або код в іншій системі числення).

Після закриття діалогового вікна на символі мегафункції відобразяться результати редагування. Інші елементи схеми вводимо як у звичайному графічному файлі. Аби схема не потребувала надлишкового ресурсу (7 зайвих виводів pin і кілька комірок логічної матриці, адже немає сенсу деякі виводи ззовні закорочувати на землю), шину даних da[7..0] з метою зменшення на 7 вхідних портів розщеплюємо: лінію da6 відокремлюємо у вхідний порт, а інші її лінії з'єднуємо із землею. Крім того, імена ліній і шин узгоджуємо з назвами портів і виводів.

Після компіляції та функціонального моделювання (файл d:\max2work\tutorial\4lab\400gr_mega.scf) переконуємося в правильності проектування: функція *y2a* узгоджується з таблицею відповідності.

☝ ***Примітки:***

1. В узагальнених назвах параметра, наприклад, data[WIDTH-1..0] або sel[WIDTHS-1..0] цифри 1..0 означають не його величину, а те, що діапазон зазначається стандартно – від старшого розряду до молодшого, наприклад, d[7..0], a[2..0].

2. У мультиплексорах ширина адресної шини WIDTHS за замовчуванням становить CEIL(LOG2(WIDTH)), де CEIL (ceiling – стеля) означає найближче ціле число, що не менше виразу в дужках, наприклад, CEIL(LOG2(7)) = 3, CEIL(LOG2(9)) = 4.

3. На символі мегафункції параметри позначаються (вгорі в правому кутку бузковим кольором) за ввімкненої опції Show Parameters меню Options.

4.2.4.7 У тому ж графічному файлі ***виконати завдання*** п. 4.2.2.1 ***на мегафункції мультиплексора*** **lpm_mux** за ручного її настроювання (достатньо один варіант мультиплексора) аналогічно п. 4.6.

🖫 ***Приклад:*** d:\max2work\tutorial\4lab\400gr_mega.gdf (схема 6), .scf.

4.2.5 Засвоїти основи застосування мегафункцій у *текстовому* редакторі (на прикладі варіантів п. 4.2.2.1).

4.2.5.1 У текстовому файлі (.tdf) нового проекту 4XXtx_mega ***виконати завдання*** п. 4.2.2.1 ***на мегафункції зразка мультиплексора найбільшої розрядності*** 4XXmuxN.inc, .sym, створеного менеджером MegaWizard Plug-In Mana-ger (п. 4.2.3), для чого виконати такі дії (задля наочності див. файл d:\max2work\tutorial\4lab\400gr_mega.gdf, схема 1 і приклад – фрагменти файла 4XXtx_mega.tdf, що наводяться під кожним пунктом):

1) ввести заголовок (**Title Statement**);
TITLE "*y* на основі мегафункцій ";

2) включити вибрану мегафункцію до складу проекту: вставити шаблон оператора включення **Include Statement** і в лапках ввести ім'я різновиду мегафункції 4XXmuxN, надане під час його створення (п. 4.2.3.3), з розширенням файла включення **.inc**;
INCLUDE "400mux16.inc";

3) ввести вхідні і вихідні порти в секції підпроекту (**Subdesign Section**), використовуючи замість x[] інші імена змінних, наприклад, in[];

```
SUBDESIGN 400tx_mega
    (
        in[4..1]    : INPUT;
        y1          : OUTPUT;
    )
```

4) вставити секцію змінних **Variable Section**;
VARIABLE

5) викликати підсекцію оголошення параметризованого зразка **Instance Declaration (parametererized)** і ввести в її шаблон ім'я зразка функції (яке ми надаємо модулю настроєної мегафункції в уявній схемі), через двокрапку – ім'я мегафункції та після ключового слова WITH – її параметри (кількість входів даних WIDTH і розрядність адресної шини WIDTHS);

```
mx1         : 400mux16
      WITH (WIDTH = 16, WIDTHS = 4);
```

6) ввести логічний блок **Logic Section**;

```
BEGIN
END;
```

7) між ключовими словами (BEGIN та END;) ввести оператор булевих рівнянь **Boolean Equation** та вставити в утворений таким чином шаблон через крапку з комою рівняння, якими з'єднати входи мегафункції з

вхідними портами та вихідний порт – з виходом мегафункції.

```
BEGIN
    mx1.sel[3..0] = in[4..1]; mx1.data[15..0] = H"3722"; y1 = mx1.result;
END;
```

8) виконати компіляцію та функціональне моделювання і переконатися в правильності проектування (задля наочності див. файли d:\max2work\tutorial\4lab\400gr_mega.gdf, схема 1, 400gr_mega.scf).

Приклад: d:\max2work\tutorial\4lab\400tx _mega.tdf, .scf.

Примітка. Логічні зв'язки між виходами та входами мегафункції містяться у файлі її включення (п. 4.2.5.1, підпункт 2), тому в булевих рівняннях достатньо лише з'єднати входи і виходи зразка модуля, що репрезентує настроєну мегафункцію в проектованому пристрої, з вхідними і вихідними портами пристрою. Входи і виходи зразка мають такі самі назви, як у мегафункції, але позначаються через крапку після імені зразка. У лівій частині рівнянь ставляться невідомі величини, а в правій – відомі, задані вхідними портами INPUT або визначені у файлі включення (у нас – result). У разі потреби, додаткові з'єднання виконуються через зовнішні відносно мегафункції логічні операції, наприклад, входи даних з'єднано з константами згідно з таблицею відповідності.

4.2.5.2 У тому ж текстовому файлі ***виконати*** п. 4.2.5.1 ***на мегафункціях мультиплексорів меншої розрядності***, створених менеджером MegaWizard Plug-In Manager, шляхом доповнення розділів 2, 3, 5, 7 цього пункту потрібними даними (задля наочності див. файл d:\max2work\tutorial\4lab\400gr_mega.gdf, схеми 2, 3, 4).

Приклад: d:\max2work\tutorial\4lab\400tx _mega.tdf, .scf.

Примітка. У п. 4.2.5.2 достатньо виконати один-два варіанти схеми на мультиплексорах різної розрядності; п. 4.2.5.3, 4.2.5.4 є факультативними, необов'язковими.

4.2.5.3 У тому ж текстовому файлі ***виконати*** п. 4.2.5.1 ***на мегафункції мультиплексора*** **mux** шляхом доповнення підпунктів 2, 3, 5, 7 цього пункту потрібними даними (задля наочності див. файл d:\max2work\tutorial\4lab\400gr_mega.gdf, схема 5).

Приклад: d:\max2work\tutorial\4lab\400tx _mega.tdf, .scf.

Примітка. Ім'я мегафункції та її параметрів мають суворо відповідати формату, які для функції **mux** можна зчитати безпосередньо з її символу в графічному файлі.

5.2.5.4 У тому ж текстовому файлі ***виконати*** п. 4.2.5.1 ***на мегафункції мультиплексора*** **lpm_mux** шляхом доповнення підпунктів 2, 3, 5, 7 цього пункту потрібними даними (задля наочності див. файл d:\max2work\tutorial\4lab\400gr_mega.gdf, схема 6).

Приклад: d:\max2work\tutorial\4lab\400tx _mega.tdf, .scf.

Примітка. Ім’я та параметри мегафункції **lpm_mux** можна зчитати з її прототипу (AHDL Function Prototype) у довідці клацанням по імені в текстовому файлі або по символу в графічному файлі (або з меню Help > Mega-functions/LPM > Ім'я мегафункції > AHDL Function Prototype, чи з контекстного меню клацанням В2 по графічному символу > Edit Ports/Parameters > Help on “Ім'я мегафункції” > AHDL Function Prototype).

Контрольні питання та завдання

1. Самостійно синтезувати мультиплексор за його ***текстовим описом:***

а) з використанням ***оператора таблиці відповідності Truth Table Statement*** (надати нове ім’я проекту 4XXtab, створити текстовий файл 4XXtab.tdf, до якого ввести заголовок (Title Statement), вхідні і вихідні вектори (Subdesign Section), логічний блок (Logic Section), підсекцію таблиці відповідності (Truth Table Statement), відтак виконати компіляцію, функціональне моделювання та автоматичне порівняння утворених таким чином часових діаграм;

б) з використанням ***умовного оператора If Then*** у проекті 4XXif_then шляхом створення текстового файла 4XXif_then.tdf і аналізу результатів так само, як у підпункті а), в якому замість таблиці відповідності ввести оператор If Then Statement;

в) з використанням ***оператора вибору Case*** у проекті 4XXcase шляхом створення текстового файла 4XXcase.tdf і аналізу результатів так само, як у підпункті а), в якому замість таблиці відповідності ввести оператор Case Statement.

2. Як слід каскадувати мультиплексори з метою збільшення розрядності?

3. Як пов’язані між собою кількість адресних і інформаційних входів у мультиплексорі?

4. Запишіть вираз для вихідної логічної функції мультиплексора 8:1 та побудуйте його схему.

5. Спроектуйте на мультиплексорах: 1) ЦКП для реалізації логічної функції, що задана варіантом завдання 2; 2) шифратор 4:2; 3) пріоритетний шифратор 4:2.

5 АРИФМЕТИЧНІ ПРИСТРОЇ

Мета роботи: дослідження типових арифметичних пристроїв – суматорів, арифметико-логічних пристроїв, помножувачів і компараторів; побудова ЦКП на суматорах і компараторах; засвоєння основ створення ієрархічних проектів.

Домашнє завдання

Спроектувати заданий згідно з варіантом (див. додаток А, варіанти завдання 5) пороговий елемент на суматорах і компараторах.

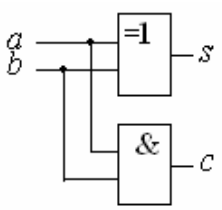
5.1 Стислі теоретичні відомості

5.1.1 Двійкові суматори

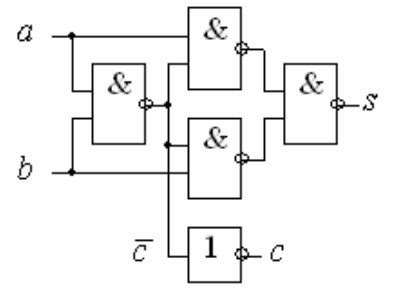
Пристрій, що виконує арифметичну операцію додавання двох розрядів *a, b* двійкових чисел *A, B* є ***півсуматором*** (HS - Halfsum - півсума). Його вихідні логічні функції суми *s* та перенесення (Carry) *c* до старшого розряду (рис. 5.1, а)

i	*a*	*b*	*s*	*c*
0	0	0	0	0
1	0	1	1	0
2	1	0	1	0
3	1	1	0	1

а)



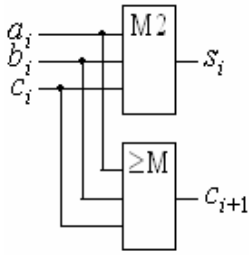
б)



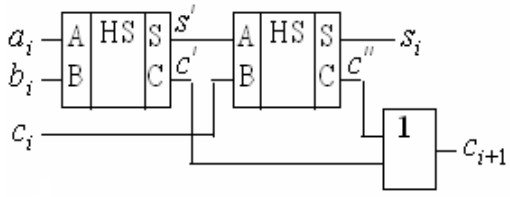
в)

i	a_i	b_i	c_i	s_i	c_{i+1}
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

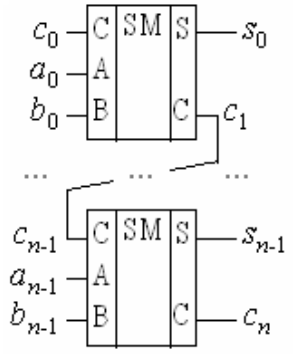
г)



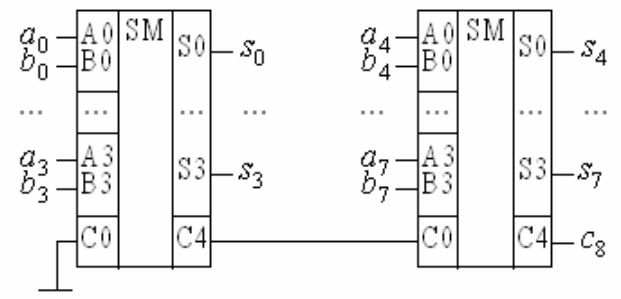
д)



е)



ж)



и)

Рисунок 5.1

$$s = a\bar{b} + \bar{a}b = a \oplus b; \quad c = ab \tag{5.1}$$

можна реалізувати на елементах виключне АБО та І (рис. 5.1, б) або на елементах І-НЕ (рис. 5.1, в).

Комбінаційний однорозрядний повний суматор або просто ***суматор*** (SM - Sum - сума) виконує операцію додавання двох розрядів a_i, b_i та перенесення c_i до цього розряду з попереднього (рис. 5.1, г). Згідно з таблицею функції суми s_i у цьому розряді і перенесення до старшого розряду c_{i+1}

$$s_i = a_i \oplus b_i \oplus c_i; \quad c_{i+1} = a_i b_i + a_i c_i + b_i c_i, \tag{5.2}$$

виконуються суматором за модулем 2 та мажоритарним елементом (рис. 5.1, д). Перетворенням (5.2) з урахуванням (5.1) дістанемо реалізацію на двох півсуматорах та елементі АБО (рис. 5.1, е).

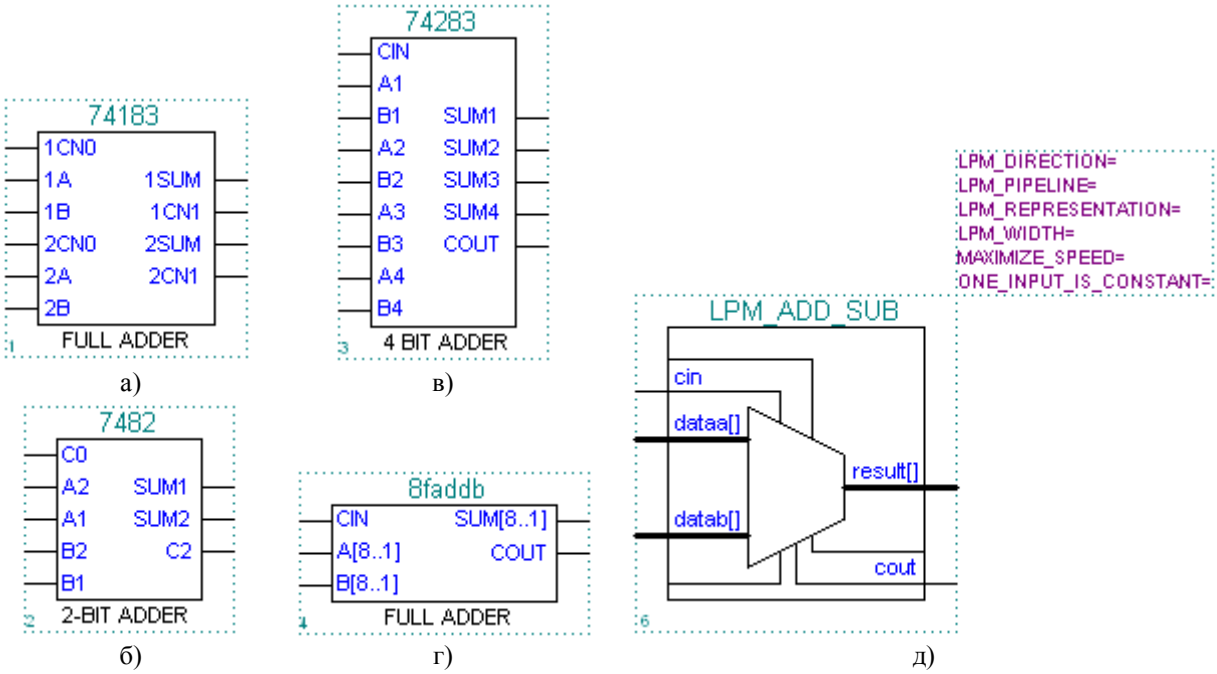


Рисунок 5.2

Багаторозрядний суматор утворюється шляхом послідовного ввімкнення n суматорів, де n - кількість розрядів доданків (рис. 5.1, ж). Вихід старшого перенесення можна використати як старший розряд зчитуваної суми, для індикації переповнення розрядної сітки, а також для каскадування багаторозрядних суматорів з метою збільшення розрядності доданків (рис. 5.1, и). З метою підвищення швидкодії групи розрядів можна з'єднувати за допомогою спеціальних схем пришвидшеного перенесення.

Суматори використовуються як для додавання, так і для віднімання, якщо двійкові числа (операнди) зображати в оберненому або доповняльному коді. Серед стандартних ІС суматорів найпоширенішими є однорозрядні повні суматори (по кілька в корпусі ІС) та дво- і чотирирозрядні (рис. 5.2, а), б), в). Бібліотека САПР містить також спеціальну макрофункцію восьмирозрядного суматора (рис. 5.2, г) і мегафункцію суматора-віднімача (рис. 5.2, д), яку легко настроїти на додавання або віднімання, а також на репрезентацію даних зі знаком або без знаку. Вхідний і вихідний

Таблиця 5.1

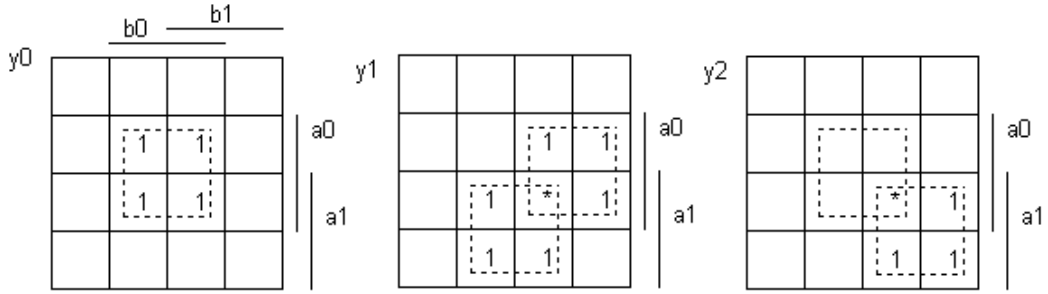
i	$a_1a_0b_1b_0$	$y_3y_2y_1y_0$
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 0
2	0 0 1 0	0 0 0 0
3	0 0 1 1	0 0 0 0
4	0 1 0 0	0 0 0 0
5	0 1 0 1	0 0 0 1
6	0 1 1 0	0 0 1 0
7	0 1 1 1	0 0 1 1
8	1 0 0 0	0 0 0 0
9	1 0 0 1	0 0 1 0
10	1 0 1 0	0 1 0 0
11	1 0 1 1	0 1 1 0
12	1 1 0 0	0 0 0 0
13	1 1 0 1	0 0 1 1
14	1 1 1 0	0 1 1 0
15	1 1 1 1	1 0 0 1

переноси можуть позначатися на символах або з номером розряду, або зручніше назвою відповідно cin (carry input) та cout (carry output).

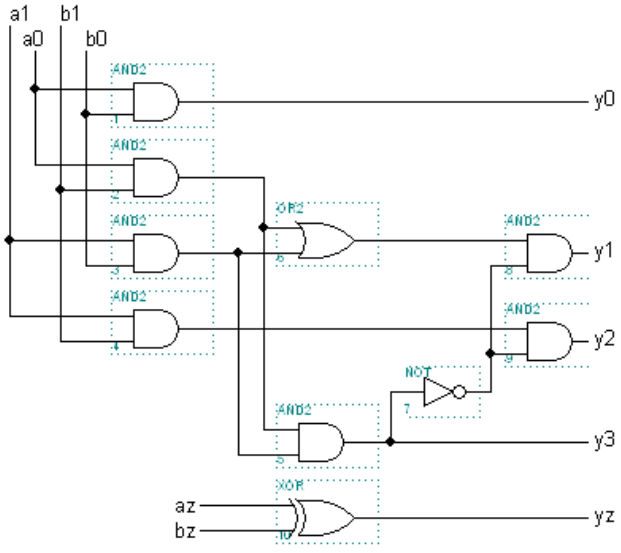
Каскадування суматорів із застосування схем пришвидшеного перенесення тут не розглядається, бо за наявності сучасних програмованих ІС це питання практично втрачає сенс.

5.1.2 Комбінаційні помножувачі

Програмно в мікропроцесорах та мікроЕОМ множення двійкових чисел здійснюють шляхом зсуву за допомогою ЦПП та додавання часткових добутків у сумато-рах згідно з відомими алгоритмами. Процедура множення виконується протягом багатьох тактів, що може спричинити зниження швидкодії в неприйнятних межах, зокрема, радіотехнічних пристроїв та систем з обробкою інформації в реальному масштабі часу.



а)



б)

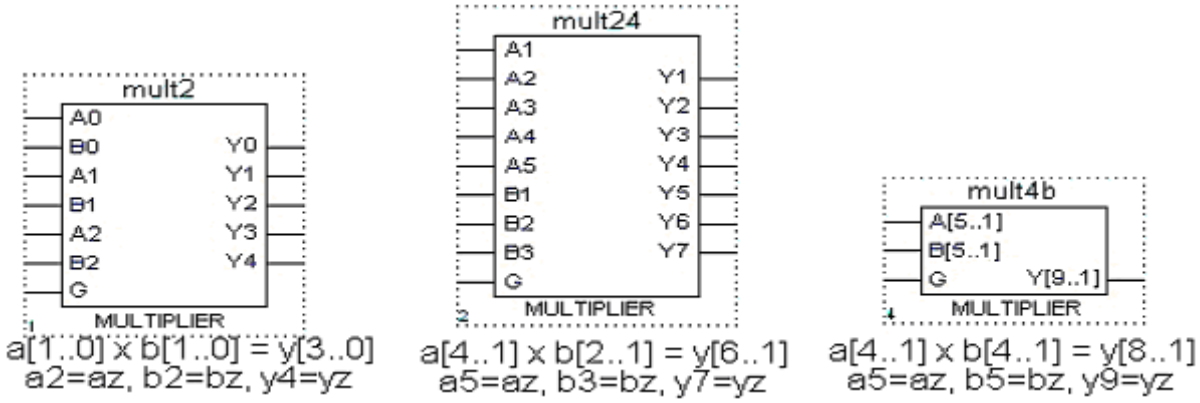
Рисунок 5.3

Підвищення швидкодії було досягнуто апаратно в ІС матрицевих помножувачів з ланцюжком суматорів часткових добутків. При цьому усталення всього добутку розтягується в часі перемиканням кількості суматорів у найдовшій діагоналі матриці суматорів.

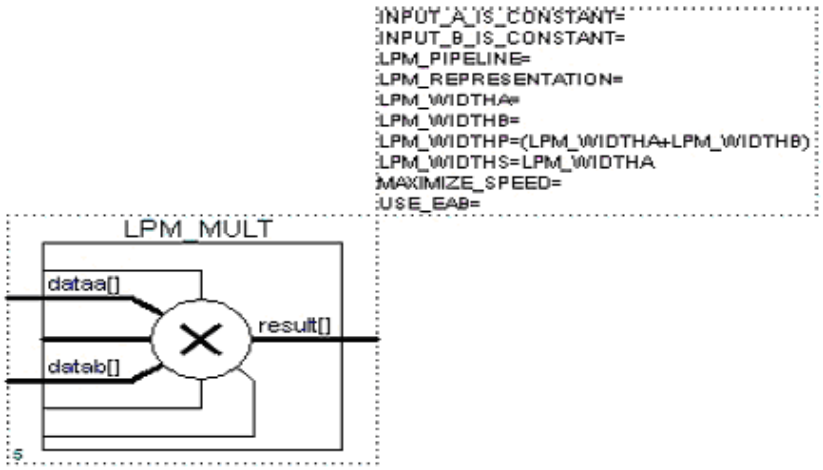
Найвища швидкодія досягається в комбінаційних помножувачах з безпосереднім формуванням розрядів остаточного добутку. Розглянемо для прикладу множення дворозрядних чисел $A=a_1a_0$ та $B=b_1b_0$ (табл. 5.1). Розрядність добутку $Y= y_3y_2y_1y_0$ вибирається як сума розрядів співмножників. Функцію для старшого розряду записуємо безпосередньо з таблиці, а інших розрядів мінімізуємо за допомогою діаграм термів (рис. 5.3, а) із застосуванням редукції:

$$\alpha = a_0b_1;\ \beta = a_1b_0;\ y_3 = \alpha\beta;\ y_2 = a_1b_1\overline{y_3};\ y_1 = (\alpha+\beta)\overline{y_3};\ y_0 = a_0b_0.$$

Мінімізована схема (рис. 5.3, б) практично не поступається за швидкодією реалізованій за первісними термами.



а) б) в)



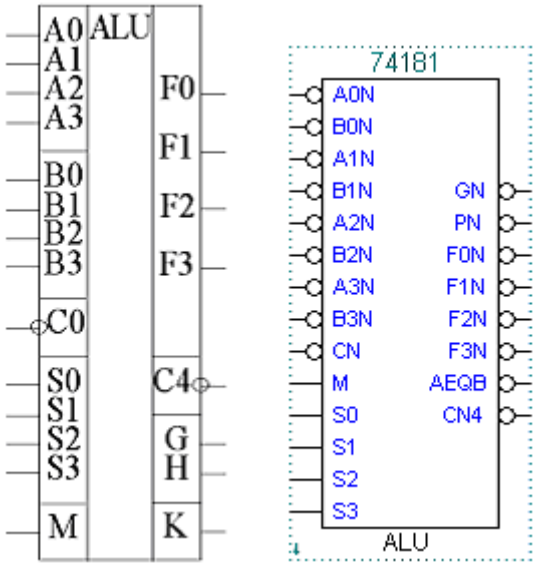
г)

Рисунок 5.4

Для чисел зі старшим знаковим розрядом $A=a_za_1a_0$, $B=b_zb_1b_0$ знак добутку $Y=y_zy_3y_2y_1y_0$ формується окремо, за допомогою елемента Виключне АБО: $y_z = a_z \oplus b_z$.

Серед стандартних ІС комбінаційних помножувачів (макрофункцій) поширеними є з розрядністю співмножників 2×2, 2×4 та 4×4

(рис. 5.4, а), б), в). Старші розряди співмножників $a[]$, $b[]$ та добутку $y[]$ репрезентують знак числа, а вхід G є стробовим. Бібліотека САПР містить також мегафункцію помножувача lpm_mult (рис. 5.4, г), основними параметрами якої є LPM_WIDTHA, LPM_WIDTHB, LPM_WIDTHP – розрядність відповідно співмножників $a[]$, $b[]$ та добутку $y[]$.



а) б)

S				F	
s_3	s_2	s_1	s_0	M=0	M=1
0	0	0	0	A	$\overline{A}$
0	0	0	1	$A \vee B$	$\overline{A \vee B}$
0	0	1	0	$A \vee \overline{B}$	$\overline{A}B$
0	1	1	0	A-B-1	$\overline{A \oplus B}$
0	1	1	1	$A\overline{B}$-1	$A\overline{B}$
1	0	0	1	A+B	$A \oplus B$
1	0	1	1	AB-1	AB
1	1	0	0	A+A	1
1	1	1	1	A-1	A

в)

Рисунок 5.5

5.1.3 Арифметико-логічні пристрої

З метою уніфікації обладнання та спрощення операційних пристроїв, призначених для виконання різних арифметичних та логічних операцій, у цифровій техніці застосовують багатофункціональні блоки – арифметико-логічні пристрої АЛП (arithmetic-logic Unit, ALU).

Подібно до суматора АЛП має входи двох операндів A, B, вхід і вихід перенесення c_0, c_n та виходи результату F (для прикладу на рис. 5, а) наведено ІС чотирирозрядного АЛП, а на рис. 5, б) – типову макрофункцію АЛП). Крім функцій суматора АЛП виконує ще низку логічних функцій, а також функцію компаратора з виходом K.

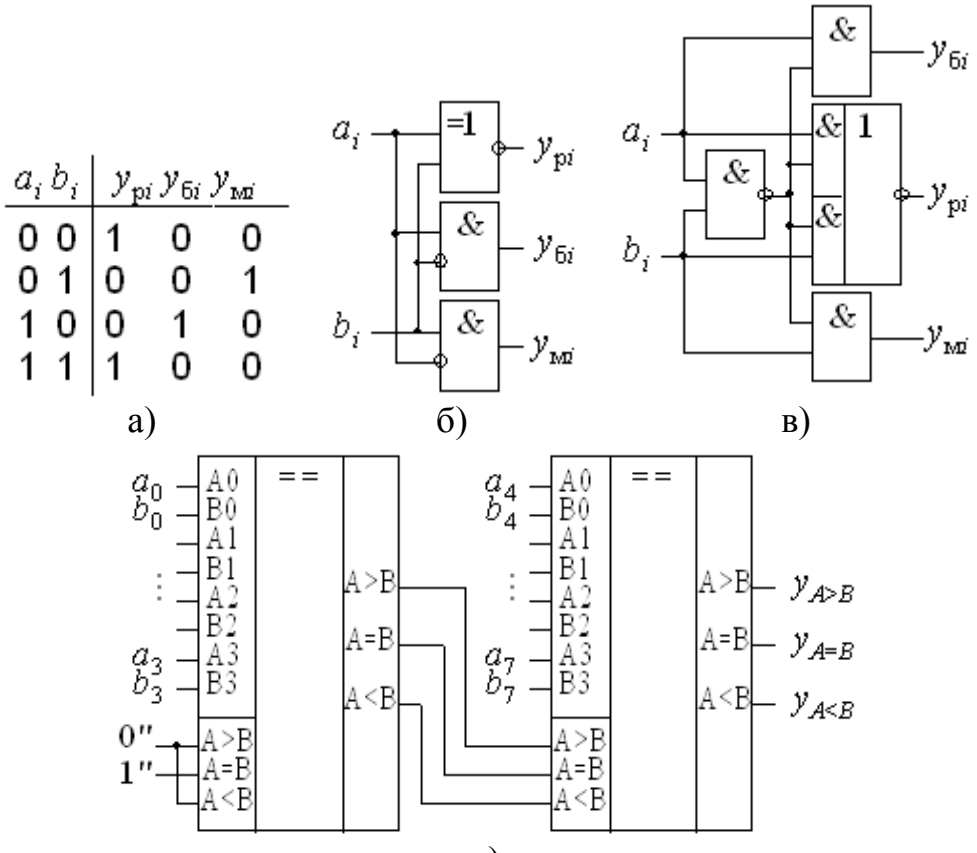
З метою збільшення розрядності операндів АЛП каскадують так само, як і суматори: послідовним з'єднанням перенесень на кшталт поданого на рис. 5.1, и) або для підвищення швидкодії через блоки пришвидшеного перенесення за допомогою спеціально для цього призначених виходів АЛП G, P генерації та передачі перенесення. Використовувати АЛП доцільно на старих ІС жорсткої структури, але бібліотека САПР програмованих ІС також містить декілька різновидів макрофункцій.

5.1.4 Цифрові компаратори

Цифровими компараторами (схемами порівняння) називаються пристрої, що виконують кількісне порівняння двійкових чисел. Вони часто

використовуються для формування сигналу закінчення певної операції, зокрема, ознаки галуження програм.

Співвідношення між двома двійковими числами $A = a_{n-1}...a_1a_0$ та $B = b_{n-1}...b_1b_0$ встановлюють за допомогою функцій компаратора: *A* equal to *B* (*aeb* = 1, якщо $A = B$ та *aeb* = 0, якщо $A \neq B$), *A* less than *B* (*alb* = 1, якщо $A < B$ та *alb* = 0, якщо $A \geq B$), *A* greater than *B* (*agb* = 1, якщо $A > B$ та *agb*= = 0, якщо $A \leq B$), *A* less or equal *B* (*aleb* = 1, якщо $A \leq B$ та *aleb* = 0, якщо $A > B$), *A* greater or equal *B* (*ageb* = 1, якщо $A \geq B$ та *ageb* = 0, якщо $A < B$), *A* not equal to *B* (*aneb* = 1, якщо $A \neq B$ та *aneb* = 0, якщо $A = B$). З усіх функцій порівняння тільки дві є незалежними, бо через них можна виразити інші. Якщо за основні правитимуть, наприклад, функції *aeb* і *agb*, то $aneb = \overline{aeb}$; $ageb = aeb + agb$; $alb = \overline{ageb}$; $aleb = aeb + alb$.



а) б) в)

г)

Рисунок 5.6

Зазначені функції порівняння чисел *A, B* базуються на функціях порівняння двох їхніх розрядів a_i, b_i однакової ваги типу рівності y_{pi}, більше $y_{бi}$ або менше $y_{мi}$ (рис. 5.6, а):

$$y_{pi} = \overline{a_i \oplus b_i}; \; y_{бi} = a_i \overline{b_i}; \; y_{мi} = \overline{a_i} b_i,$$

які можна реалізувати на елементах виключне АБО-НЕ та заборона (рис. 5.6, б) або на поширеніших елементах (рис. 5.6, в).

Багаторозрядні компаратори найвищої швидкодії можна синтезувати також шляхом одночасного порівняння всіх розрядів, проте з усклад-

ненням схеми, яка потребує багатовходових елементів. Цього недоліку можна уникнути, але з погіршенням швидкодії багатоступінчастим сполученням розрядних функцій. Вихідні функції групи розрядів є одночасно входами перенесень наступної групи, що зручно для каскадування компараторів на ІС жорсткої структури з метою нарощування їх розрядності (рис. 5.6, г). Слід звернути увагу на те, що входи перенесень наймолодшої групи з'єднуються з константами як подано на схемі.

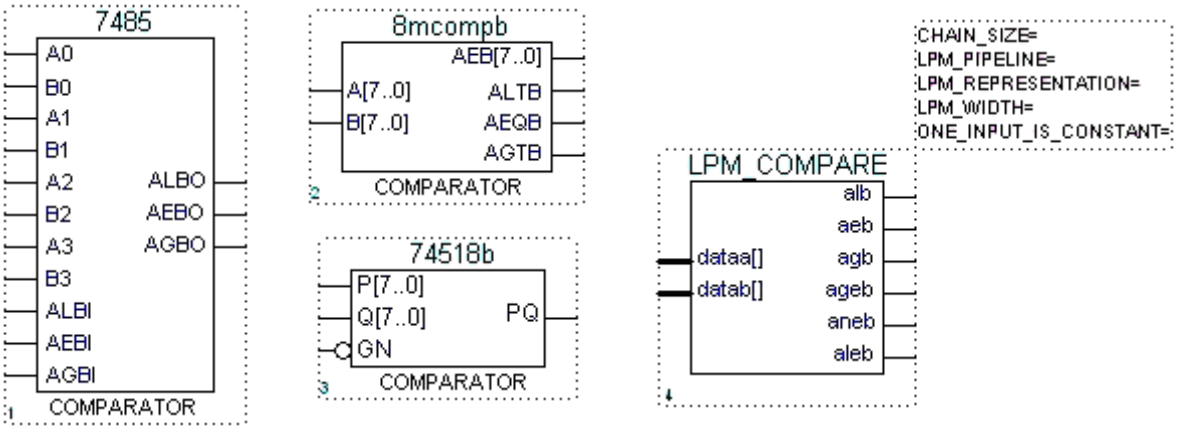


Рисунок 5.7

Серед стандартних ІС поширеними є компаратори (макрофункції) з розрядністю 4 та 8 (рис. 5.7). Бібліотека САПР містить також мегафункцію компаратора lpm_compare (див. рис. 7), основним параметром якої є LPM_WIDTH – розрядність порівнюваних чисел dataa[] та datab[].

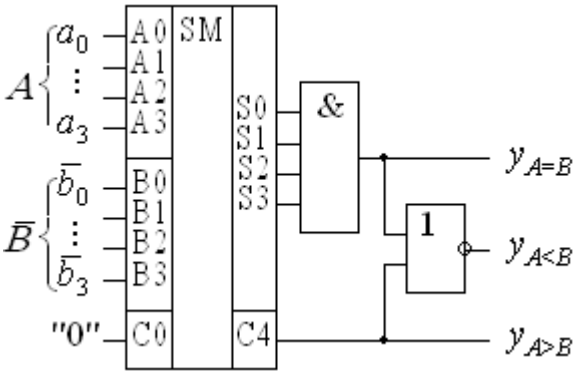


Рисунок 5.8

Крім спеціалізованих ІС компараторів для порівняння чисел можуть використовуватися й інші ЦКП. Типовою є ідея використання для порівняння кодів суматорів: відніманням одного числа від іншого за різницею легко з'ясувати співвідношення між ними. Якщо віднімання виконується шляхом додавання в оберненому коді (рис. 5.8), то всі розряди суми $S = A + \overline{B} + (2^n - 1 - B)$ при $A = B$ перетворюються на одиниці $S = 2^n - 1 = 11...1$, тобто функцію $y_{A=B} = aeb$ можна сформувати за допомогою елемента збігу. Якщо ж $A > B$, то сума $S = A - B + 2^n - 1 \geq 2^n$, тобто виникає перенесення c_n до старшого розряду, яке і є ознакою функції $y_{A>B} = c_n = agb$. Функцію $y_{A<B} = alb$, як і інші, можна сформувати логічними елементами. Аналогічно і в АЛП (див. рис. 5.5, а) з'єднання виходів $f_0...f_4$ на елементі І, виконане всередині ІС, утворює функцію компаратора $K=y_{A=B}=aeb$, шляхом комбінації якої з виходом перенесення за допомогою зовнішніх елементів можна утворити потрібні функції порівняння на кшталт зображеного на рис. 5.8.

5.1.5 Дисплей ієрархічного проекту

З метою полегшення проектування відносно складних ЦП, забезпечення ефективного контролю синтезу складників і їх взаємодії проект доцільно поділити на частини – підпроекти, які можна синтезувати окремо в графічному або текстовому редакторі, але весь ієрархічний проект опрацьовується тільки в графічному редакторі. При цьому пристрій зображається у вигляді, що нагадує структурну або функціональну схему, елементами якої є примітиви, макро- і мегафункції, а також згорнуті до символів файли, створені розробником. Ієрархічний проект має містити всі зовнішні сигнали і внутрішні сигнали з'єднань між складниками, подані шинами і лініями, як у звичайному проекті.

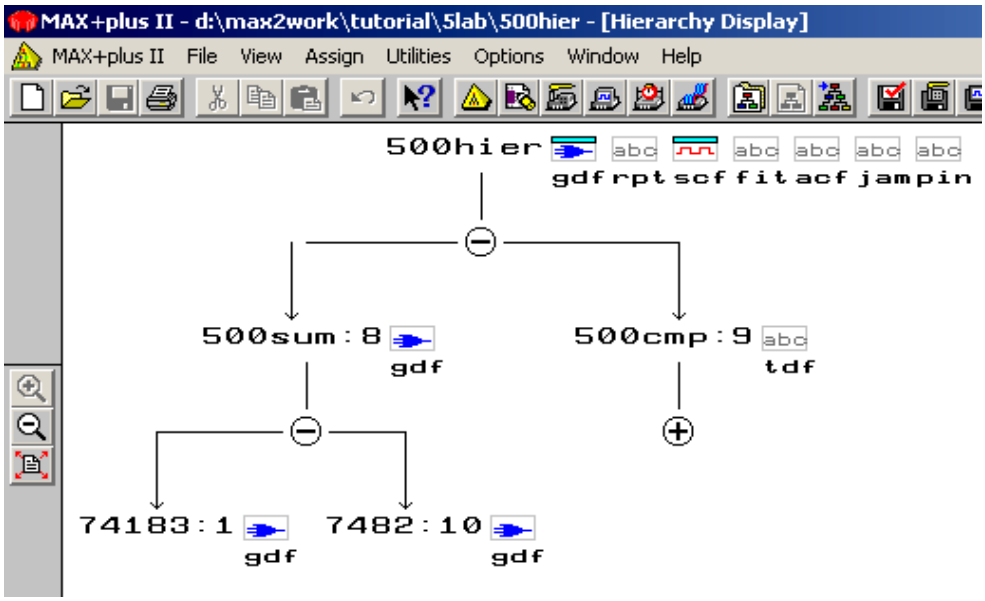


Рисунок 5.9

Дисплей ієрархічного проекту (рис. 5.9) відображає його структуру у вигляді дерева, гілками якого є складові підпроекти. Кожна гілка позначена ім'ям файла і значком відповідного підпроекту, а біля вершини ієрархії розташовано також значки допоміжних файлів, зокрема, звітного (rpt), часових діаграм (scf), цоколівки ІС (pin) та ін. Подвійним клацанням по значку можна відкрити ці файли, при цьому на ієрархічному дереві відкриті файли помічаються. Натисненням вершин дерево можна розгортати і згортати, а з меню Options змінити його орієнтацію – горизонтальну чи вертикальну.

5.2 Лабораторне завдання

5.2.1 Дослідити типові арифметичні пристрої

5.2.1.1 Дослідити суматори: за принциповими електричними схемами (файл d:\max2work\tutorial\5lab\5sum.gdf) та осцилограмами сигналів (d:\max2work\tutorial\5lab\5sum.scf) ***півсуматора*** (схема 1) і однорозрядного повного ***суматора*** (схема 2) на логічних елементах І-АБО-НЕ скласти табли-

ці відповідності і рівняння вихідних функцій, виміряти затримку вихідних імпульсів. У звіті навести також умовне графічне позначення за ДСТУ таких ЦКП зі стислим поясненням принципу їх дії. Розглянути особливості побудови і перемикання типового ***багаторозрядного суматора*** (макрофункції за схемою 3) та суматора на мегафункції (схема 4).

5.2.1.2 Дослідити типовий арифметико-логічний пристрій (АЛП) на макрофункції серії 74 (схема 5) у режимах виконання арифметичної операції додавання (при M=0) та логічної операції кон'юнкції (при M=1).

5.2.1.3 Дослідити компаратори: за принциповими електричними схемами (файл d:\max2work\tutorial\5lab\5comp.gdf) та осцилограмами сигналів (d:\max2work\tutorial\5lab\5comp.scf) ***однорозрядного компаратора*** (схема 1) на логічних елементах та типового ***багаторозрядного компаратора*** (макрофункції за схемою 2) скласти таблиці відповідності і рівняння вихідних функцій, виміряти затримку вихідних імпульсів. Розглянути особливості побудови і перемикання компаратора на мегафункції (схема 3) та виконання функцій компаратора на основі багаторозрядного суматора (схема 4).

5.2.1.4 Ознайомитися зі стандартними арифметичними пристроями серії 74 (макрофункціями) та параметризованими мегафункціями (файл d:\max2work\tutorial\5lab\5libr.gdf): суматорів і суматора-віднімача (Adder Macrofunctions, Parame-terized Adder/Subtractor Megafunctions); АЛП (Arithmetic Logic Unit Macro-functions); помножувачів (Multiplier Macrofunctions, Parameterized Multiplier Megafunctions) та компараторів (Comparator Macrofunctions, Parameterized Comparator Megafunction). Розглянути функціональні прототипи, параметри, таблиці виконуваних функцій різновидів арифметичних пристроїв (достатньо по одному з кожної групи), пояснити призначення та особливості входів і виходів.

5.2.2 Застосувати арифметичні пристрої в *графічному* редакторі для побудови порогового елемента, заданого згідно з варіантом XX

5.2.2.1 Скласти пристрій для підсумовування кількості N одиничних рівнів вхідних сигналів на макрофункціях суматорів вибраного типу в графічному файлі d:\max2work\tutorial\5lab\5XXsum.gdf проекту **5**XXsum, виконати компіляцію і моделювання, переконатися в правильності функціонування пристрою за часовими діаграмами d:\max2work\tutorial\5lab\5XXsum.scf та створити символьний файл 5XXsum.sym (див. п. 3.2).

✐ ***Примітка.*** Входи суматорів можна з'єднувати з розрядами чисел лише відповідної ваги (яка не завжди відображається на символах у назвах входів і виходів).

🖫 ***Приклад:*** d:\max2work\tutorial\5lab\500sum.gdf, .scf, .sym.

5.2.2.2 Скласти пороговий елемент у новому графічному файлі 5XXp_el.gdf проекту **5**XXp_el, для чого скопіювати і вставити до цього файла суматор 5XXsum.gdf (п. 5.2.2.1) і доповнити файл компаратором: 1) на логічних елементах, 2) на налаштованій макрофункції компаратора дібраного типу, 3) на налаштованій мегафункції компаратора, 4) на автоматично створеному за допомогою менеджера MegaWizard Plug-In Manager різновиді мегафункції компаратора 5XXcmp.sym (п. 4.2). Відтак виконати компіляцію і моделювання та переконатися в правильності функціонування пристрою за часовими діаграмами 5XXp_el.scf.

🖫 ***Приклад:*** d:\max2work\tutorial\5lab\500p_el.gdf, .scf, 500cmp.sym.

5.2.3 Засвоїти основи сумісного застосування макро- і мегафункцій у *текстовому* редакторі на прикладі виконання завдання п. 5.2.2 (на зразок схеми 5XXp_el.gdf (п. 5.2.2.2) з четвертим варіантом компаратора)**.**

1) Створити новий текстовий файл 5XXpe.tdf, включити його до проекту 5XXpe і ввести заголовок Title Statement (під кожним пунктом наводиться приклад варіанта 00).

```
TITLE "Пороговий елемент на SM і CMP";
```

2) У секції Include Statement включити до проекту автоматично створений різновид мегафункції компаратора 5XXcomp.inc (п. 5.2.2.2).

```
INCLUDE "500cmp.inc";
```

3) У секції Function Prototype Statement ввести прототипи макрофункцій суматорів (можна скопіювати з довідки Help).

```
FUNCTION 74183 (1cn0, 1b, 1a, 2cn0, 2a, 2b)
    RETURNS (1sum, 1cn1, 2sum, 2cn1);
FUNCTION 7482 (a[2..1], b[2..1], c0)
    RETURNS (sum[2..1], c2);
```

4) У секції Subdesign Section повторити ім’я проекту 5XXpe та оголосити вхідні і вихідні порти.

```
SUBDESIGN 500pe
(
    x1, x2, x3, x4, x5, x6, x7        :  INPUT;
    p, N[2..0]                        :  OUTPUT;
)
```

5) Ввести секцію змінних Variable Section.

```
VARIABLE
```

6) Вставити підсекцію Instance Declaration і подати імена зразків (через двокрапку) оголошеним у п. 2), 3) макро- і мегафункціям (для наочності у прикладі d:\max2work\tutorial\5lab\500p_el.gdf імена зразків позначено для суматорів і четвертого варіанта компаратора).

```
sm1   : 74183;  sm2 : 7482;   cmp : 500cmp;
```

7) Викликати логічну секцію Logic Section;

```
BEGIN

END;
```

8) Вставити до неї підсекцію Boolean Equation і рівняннями з'єднати зразки функцій між собою та з оголошеними портами INPUT і OUTPUT.

```
BEGIN
    sm1.1cn0=x1; sm1.1a=x2; sm1.1b=x3;
    sm1.2cn0=x4; sm1.2a=x5; sm1.2b=x6;
    sm2.b1=sm1.1sum; sm2.b2=sm1.1cn1;
    sm2.a1=sm1.2sum; sm2.a2=sm1.2cn1; sm2.c0=x7;
    cmp.dataa[]=(sm2.c2, sm2.sum2, sm2.sum1);
    p=cmp.ageb; N[]=(sm2.c2, sm2.sum2, sm2.sum1);
END;
```

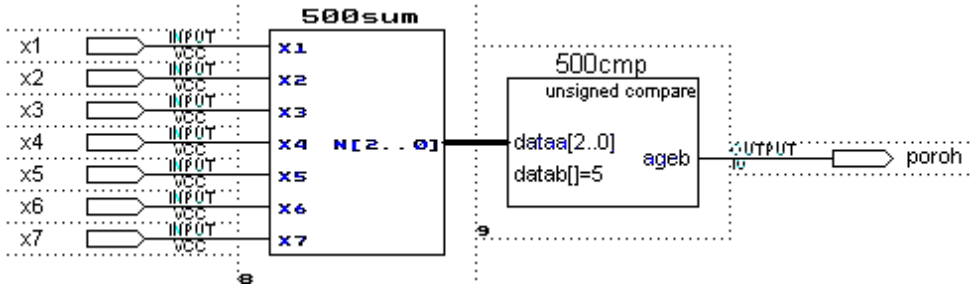
8) Відтак виконати компіляцію і моделювання та переконатися в правильності функціонування пристрою за часовими діаграмами 5XXpe.scf.

🖫 ***Приклад:*** d:\max2work\tutorial\5lab\500pe.tdf, .scf.

5.2.4 Засвоїти основи створення *ієрархічного* проекту виконанням завдання п. 5.2.2 (на зразок схеми 5XXp_el.gdf з четвертим варіантом компаратора)

5.2.4.1 Скласти пороговий елемент у новому графічному файлі 5XXhier.gdf проекту **5**XXhier (на кшталт структурної схеми), для чого вставити до цього файла символи пристроїв, створені в *графічних* і/або *текстових* файлах проектів нижчого рівня, та, у разі потреби, додаткові будь-які компоненти (примітиви, макро- чи мегафункції) і з'єднати їх між собою належним чином. Для прикладу варіанта 00 потрібно вставити створений у *графічному* файлі символ суматора d:\max2work\tutorial\5lab\500sum.sym (п. 5.2.2.1) та створений у *текстовому* файлі за допомогою менеджера MegaWizard Plug-In Manager символ компаратора d:\max2work\tutorial\5lab\500cmp.sym (п. 5.2.2.2), а також вхідні і вихідний порти.

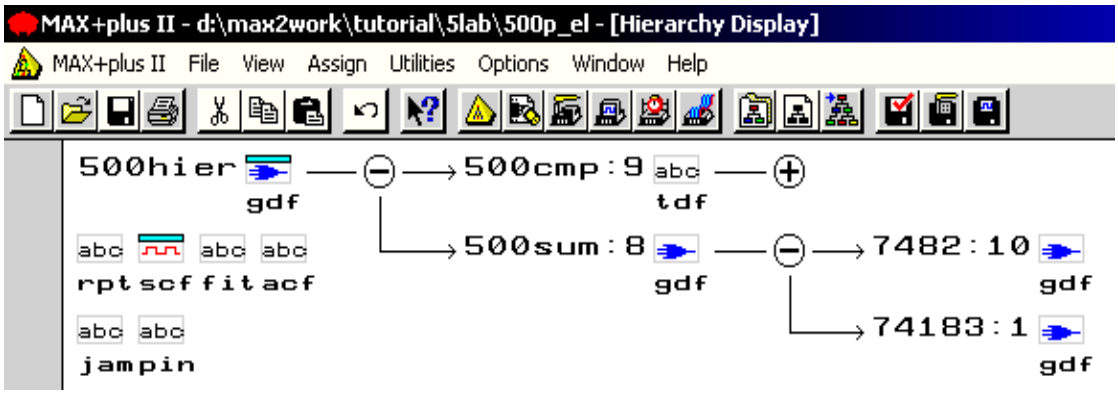
5.2.4.2 Виконати компіляцію і моделювання та переконатися в правиль-ності функціонування пристрою за часовими діаграмами **5**XXhier.scf.



5.2.4.3 Відкрити файл ієрархічного дисплея піктограмою з інструментальної панелі (або з меню MAX+plus II > Hierarchy Display), в якому спостерігати ієрархічне дерево розкриттям (+) або згортанням (-) його гілок із значками основних файлів, що утворюють проект, а також допоміжні файли, значки яких розташовані поруч із деревом. Зорієнтувати дерево для зручного його перегляду та копіювання: меню Option > Orientation > Vertical / Horizontal. Ознайомитися в загальних рисах зі змістом основних і допоміжних файлів, зокрема, зі звітного файла **.rpt** скопіювати блок-схему, нумерацію та призначення виводів ВІС.

🖫 ***Приклад:*** d:\max2work\tutorial\5lab\500hier.gdf, .scf, .rpt.

☝ ***Примітка.*** Відкрити головний проектний файл (файл вершини дерева ієрархії) або пересунути його на передній план вікна можна піктограмою з інструментальної панелі. Для відкриття будь-якого файла досить двічі клацнути по його значку на ієрархічному дереві (або клацнути B2 > Open Editor), при цьому вгорі значків відкритих файлів висвітлюються бузкові смужки. Одноразове клацання виділяє файл чорним, а смужка відкритого файла стає червоною; клацанням за натиснутої клавіші Shift виділяється декілька файлів. Відкрити всі виділені файли одночасно можна командою з меню File > Open Editor, а закрити – командою File > Close Editor. Наведені операції особливо корисні під час опрацювання складних проектів.



Контрольні питання та завдання

1. Як слід каскадувати з метою збільшення розрядності 1) суматори, 2) АЛП, 3) компаратори?

2. Реалізуйте на основі заданих в дужках пристроїв і, за необхідності, додаткових логічних елементів таку арифметичну операцію та підрахуйте швидкодію її виконання:

1) $a_0 + b_0 + c_0$ (І-НЕ),

2) $a_0 + b_0 + c_0$ (АБО-НЕ),

3) $a_1a_0 + b_1b_0$ (виключне АБО),

4) $a_1a_0 + b_1b_0$ (дешифратор),

5) $a_2a_1a_0 + b_2b_1b_0$ (півсуматори),

6) $a_2a_1a_0 + b_2b_1b_0$ (повні однорозрядні суматори),

7) $a_2a_1a_0 - b_2b_1b_0$ (повні однорозрядні суматори),

8) $a_2a_1a_0 - b_2b_1b_0$ (чотирирозрядний суматор).

3. Спроектуйте на основі суматорів:

а) мажоритарні елементи:

1) на 5 входів;

2) на 7 входів;

3) на 9 входів;

4) на 11 входів;

б) порогові елементи:

1) 2 з 5;

2) 3 з 6;

3) 3 з 7;

4) 4 з 5;

5) 4 з 6;

6) 4 з 8;

7) 4 з 9;

8) 5 з 7;

9) 5 з 8;

10) 6 з 9.

Підрахуйте, скільки потрібно було б елементів І-НЕ для побудови такої схеми.

4. Спроектуйте на основі мультиплексорів:

1) мажоритарний елемент на 3 входи,

2) пороговий елемент 2 з 5,

3) компаратор рівності трирозрядних кодів,

4) півсуматор,

5) суматор,

6) компаратор дворозрядних чисел з функціями “рівно” та “більше”.

5. Реалізуйте цифровий компаратор:

а) 4-розрядних чисел на:

1) логічних елементах для двох основних функцій порівняння;

2) дешифраторі та мультиплексорі для функції рівності;

3) мультиплексорах для двох функцій порівняння;

4) АЛП для всіх функцій порівняння;

5) суматорах для всіх функцій порівняння;,

б) 12-розрядних чисел для трьох функцій порівняння на основі:

1) 4-розрядних компараторів;

2) 4-розрядних суматорів;

3) 4-розрядних АЛП.

6 ТРИГЕРИ

Мета роботи: дослідження типових тригерів; побудова ЦПП на основі тригерів; засвоєння основ застосування тригерів у проектах, використання шаблону оголошення регістрів/тригерів (Register Declaration).

Домашнє завдання

За дóзвільного рівня *Enable* синхроімпульсом *Clock* записати до тригерів дані і сформувати на їхніх виходах логічну функцію, задану варіантом завдання 1, але за власним вибором базису (див. додаток А, варіанти завдання 6).

6.1 Стислі теоретичні відомості

6.1.1 Класифікація тригерів

Тригером (trigger – спусковий гачок вогнепальної зброї) називається пристрій з двома стійкими станами, що змінює їх стрибкоподібно під дією керувальних сигналів. Один зі стійких станів відповідає інформаційному значенню логічного 0, а другий – логічної 1, отже, такий ЦПП здатний запам’ятовувати і зберігати 1 біт інформації, тому найпростіші тригери називають також бістабільними комірками або елементами пам’яті.

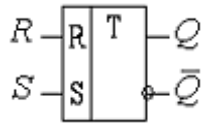
Рівні сигналів, за яких тригер залишається в режимі збереження, називають *пасивними*, а які спричиняють його перемикання – *активними*. Відповідно і входи тригера називають *прямими*, якщо активними є рівні логічної 1, та *інверсними*, якщо активними є рівні логічного 0.

За способом запису інформації тригери поділяють на *асинхронні* та *синхронні*. В асинхронних тригерах перемикання відбувається в той самий момент часу, коли змінюється комбінація інформаційних сигналів, а в синхронних – лише з надходженням синхроімпульсу. За способом синхронізації тригери поділяють на два типи. Тригери зі *статичним керуванням* (керовані рівнем) перемикаються протягом дії активного рівня синхросигналу: логічної 1 (прямий статичний синхровхід) або логічного 0 (інверсний статичний синхровхід). Тригери з *динамічним керуванням* (керовані фронтом) перемикаються лише під час дії активного перепаду напруги синхроімпульсу: від логічного 0 до логічної 1, тобто за його позитивним фронтом (прямий динамічний синхровхід), або від логічної 1 до логічного 0, тобто за його негативним фронтом чи спадом (інверсний динамічний синхровхід). За *логікою функціонування* практичного значення набули лише декілька типів тригерів. Основними можна вважати тригери типів RS, D, T та JK, а всі інші є їх різновидами або комбінованими тригерами, в яких суміщаються функції тригерів кількох типів.

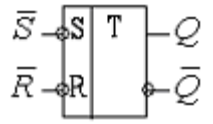
6.1.2 Асинхронні тригери

6.1.2.1 RS-тригери

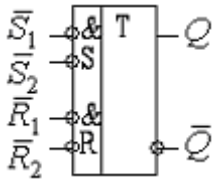
RS-тригерами називають тригери з роздільним установленням (запуском) за допомогою двох входів: S (Set – установлення) – вхід установлення до одиничного стану $Q = 1$ та R (Reset – повернення, скидання) – вхід скидання до нульового стану $Q = 0$. Часто застосовують скорочені назви асинхронних тригерів: типу RS для тригерів з прямими входами (рис. 6.1, а) та типу $\overline{\mathrm{R}}\,\overline{\mathrm{S}}$ для тригерів з інверсними входами (рис. 6.1, б), в). Правила функціонування тригера зручно задавати таблицею відповідності, яку називають також *перемикальною* таблицею або таблицею *станів*. Для RS тригера з *прямими* входами таку таблицю наведено на рис. 6.1, г). Вона визначає стан Q^+, до якого має перейти тригер з попереднього стану Q, якщо на його входи R, S подати зазначену комбінацію сигналів. За пасивних рівнів $R = S = 0$ (вхідні кортежі $i = 0, 1$) тригер перебуває в режимі схову $Q^+ = Q$; установлення до одиничного стану ($i = 2, 3$) та скидання до нульового ($i = 4, 5$) здійснюється активними рівнями відповідно $S = 1$ та $R = 1$ однойменних сигналів, причому цей стан або підтверджується ($i = 3, 4$), або тригер до нього перемикається ($i = 2, 5$); комбінація $R \cdot S = 1$ ($i = 6, 7$) є забороненою, бо надходження активних рівнів на обидва входи може призвести після одночасного припинення їх дії до невизначеного тану $Q^+ =$ X. У кінцевому підсумку правила функціонування тригера можна описати за допомогою вкороченої перемикальної таблиці (рис. 6.1, д).



а)



б)



в)

i	R	S	Q	Q^+	Режим
0	0	0	0	0	Збереження
1	0	0	1	1	
2	0	1	0	1	Установлення
3	0	1	1	1	
4	1	0	0	0	Скидання
5	1	0	1	0	
6	1	1	0	X	Заборонено
7	1	1	1	X	

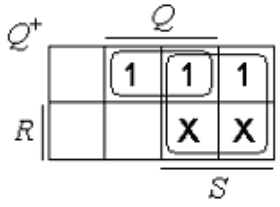
г)

R	S	Q^+
0	0	Q
0	1	1
1	0	0
1	1	X

д)

$\overline{R}$	$\overline{S}$	Q^+
0	0	X
0	1	0
1	0	1
1	1	Q

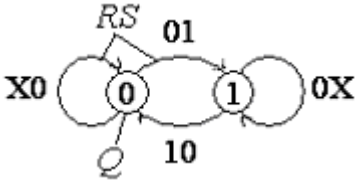
е)



ж)

Q	Q^+	R	S	$\overline{R}$	$\overline{S}$
0	0	X	0	X	1
0	1	0	1	1	0
1	0	1	0	0	1
1	1	0	X	1	X

и)



к)

Рисунок 6.1

Із перемикальної таблиці для $\overline{R}\,\overline{S}$-тригера з *інверсними* входами (рис. 6.1, е) випливає, що рівні логічної 1 для нього є пасивними: при $\overline{R} = \overline{S} = 1$ тригер перебуває в режимі схову $Q^+ = Q$, активними рівнями $\overline{R} = 0$ та $\overline{S} = 0$ він відповідно скидається та встановлюється, а комбінація $\overline{R} + \overline{S} = 0$ є забороненою. Тригери з прямими і інверсними входами є дуальні, бо інвертуванням змінних їх перемикальні таблиці перетворюються одна в одну.

Внаслідок цього RS- та $\overline{R}\,\overline{S}$-тригери аналітично описуються однаковими логічними функціями Q^+ (*R, S, Q*), що відображають залежність вихідного сигналу Q^+ від зовнішніх змінних – сигналів *R, S* та внутрішньої змінної – попереднього стану *Q*. Такі функції називаються *характеристичними рівняннями*. Для їх мінімізації за допомогою таблиці станів (див. рис. 6.1, г) або безпосередньо за правилами функціонування тригера будуємо діаграму термів (рис. 6.1, ж). Довизначаючи факультативні значення функції до X = 1 та об'єднуючи одиниці, отримуємо характеристичне рівняння тригера

$$Q^+ = S + Q\,\overline{R}, \text{ якщо } RS = 0. \qquad (6.1)$$

Підставляючи можливі комбінації змінних, крім забороненої, яка подана обмежувальною умовою в (6.1), неважко переконатись, що це рівняння описує аналітично правила функціонування тригера.

Під час синтезу послідовнісних пристроїв з використанням тригерів як елементної бази доводиться розв'язувати завдання, обернене аналізу, коли вихідні сигнали стають аргументами, а вхідні – функціями, які називаються *функціями збудження*. Отримання їх полегшується за допомогою *таблиці переходів* (рис. 6.1, и), що визначає, яку комбінацію сигналів необхідно подати на входи тригера, аби він перейшов зі стану *Q* до стану Q^+. Так, із таблиці станів RS-тригера (див. рис. 6.1, г) визначаємо, що переходові $Q \to Q^+ = 0 \to 0$ відповідають комбінації *RS* = 00 (кортеж *i* = 0) або *RS* = 10 (*i* = 4), тому в першому рядку таблиці занотовуємо код *RS* = X0, де, як завжди, X = 0 або 1. Так само заповнюємо й інші рядки таблиці переходів RS-тригера, а для $\overline{R}\,\overline{S}$-тригера досить зінвертувати значення змінних. Довизначаючи в цій таблиці X = 1, можна виразити мінімальні функції збудження *RS*-тригера через стан Q^+, до якого він має перейти за довільного початкового стану:

$$R = \overline{Q}^+;\ S = Q^+ \text{ при } Q = \text{X}. \qquad (6.2)$$

Наочніше переходи від одного стану до іншого, подані таблицею переходів, можна відобразити графічно – за допомогою спрямованого сигнального *графа* (або просто графа) тригера (рис. 6.1, к). У його *вершинах*, зображених колами, зазначають стани тригера *Q* = 0 та 1, а біля *гілок* (ребер, дуг) – коди сигналів *RS*, що відповідають даному переходові. Гілки,

які замикаються біля однієї вершини, утворюють *петлі*, що відображають стійкі стани тригера. Так, петлі RS = X0 відповідає режим збереження $Q = 0$, гілці $RS = 01$ – перехід тригера до стійкого стану $Q = 1$, в якому він перебуває, і при повторенні цієї комбінації сигналів, що відображається петлею 0X. Відсутність на графі комбінації $RS = 11$ означає, що вона є заборонена.

6.1.2.2 Схемна реалізація асинхронних RS-тригерів. Побудова тригерів базується на створенні позитивного зворотного зв'язку (ПЗЗ) шляхом перехресного з'єднання виходів і входів логічних елементів (рис. 6.2, а), б). Коли під час перемикання тригера обидва елементи опиняються в активному режимі, тобто стають підсилювачами, внаслідок дії ПЗЗ виконуються умови самозбудження. Завдяки цьому стан тригера виявляється нестійким, тимчасово пристрій стає генератором (релаксатором – джерелом розривних, негармонічних коливань) і в ньому виникає регенеративний процес лавиноподібного перемикання (перекидання) до одного зі стійких станів. Регенеративний процес припиняється, коли один з елементів опиняється в режимі відсічки або інший – у режимі насичення, бо при цьому порушується баланс амплітуд (як відомо, в обох зазначених режимах коефіцієнт підсилення за змінним струмом дорівнює нулю).

Для $\overline{\mathrm{R}}\,\overline{\mathrm{S}}$-тригера на елементах І-НЕ (див. рис. 6.2, а) згідно з аксіомою алгебри логіки $x \cdot 1 = x$ пасивними є рівні логічної 1, тому він функціонує за перемикальною таблицею на рис. 6.1, е). Якщо при $\overline{R} = \overline{S} = 1$ асинхронний $\overline{\mathrm{R}}\,\overline{\mathrm{S}}$-тригер перебуває в режимі збереження, наприклад, у стані $Q = 0$, як показано на схемі рис. 6.2, а) кодами в першій позиції станів та рівнями сигналів на часових діаграмах рис. 6.2, в) при $t < t_1$, то з надходженням у момент t_1 активного рівня логічного 0 до входу $\overline{S}$ (у другій позиції станів верхнім індексом нуль $\overline{S} = 0^0$ позначено, що від цього моменту відлічується час перемикання елементів схеми) спочатку перемикається плече Q з однією затримкою $t_{\text{з.п}}$ (верхній індекс 1), а відтак плече $\overline{Q}$ з двома затримками $t_{\text{з.п}}$ (індекс 2) відносно появи рівня $\overline{S} = 0$. Кількість затримок $t_{\text{з.п}}$ на ідеалізованих часових діаграмах для спрощення показано цифрами 1, 2. Аналогічно відбувається й скидання тригера до стану $Q = 0$ рівнем $\overline{R} = 0$, як показано в третій позиції станів на схемі та на часових діаграмах, починаючи з моменту t_2.

Отже, час затримки поширення сигналу від входів до виходів тригера, тобто тривалість його перемикання до повного усталення вихідного коду при переході від стану 0 до стану 1 та навпаки, становить $t^{01} = t^{10} = t_{\text{Т}} = 2t_{\text{з.п.}}$ Тригер надійно спрацює, тобто не зможе повернутися до попереднього стану за зникнення активного рівня вхідного сигналу, якщо цей сигнал триватиме до закінчення процесу регенеративного перемикання: $t_{\text{і}} \geq 2t_{\text{з.п}}$. Проте, якщо активний сигнал на другому вході з'явиться одразу ж після переми-

кання тригера, тривалість одного з вихідних сигналів (на виході $\overline{Q}$ діаграми) буде меншою за $2t_{з.п}$, що недостатньо для надійного спрацьовування аналогічного навантажувального каскаду. Через це роздільний час тригера – мінімальний інтервал між надходженням вхідних імпульсів – має становити $T \geq 3t_{з.п}$, а робоча частота чергування імпульсів $f \leq 1/3t_{з.п}$.

Для RS-тригера на *елементах АБО-НЕ* (див. рис. 6.2, б) згідно з аксіомою $x + 0 = x$ пасивними є рівні логічного 0 і він функціонує за таблицею на рис. 6.1, д). Швидкодія RS-тригера, як видно з позицій станів на схемі та часових діаграм (рис. 6.2, г), визначається так само, як для $\overline{\mathrm{R}}\,\overline{\mathrm{S}}$ тригера.

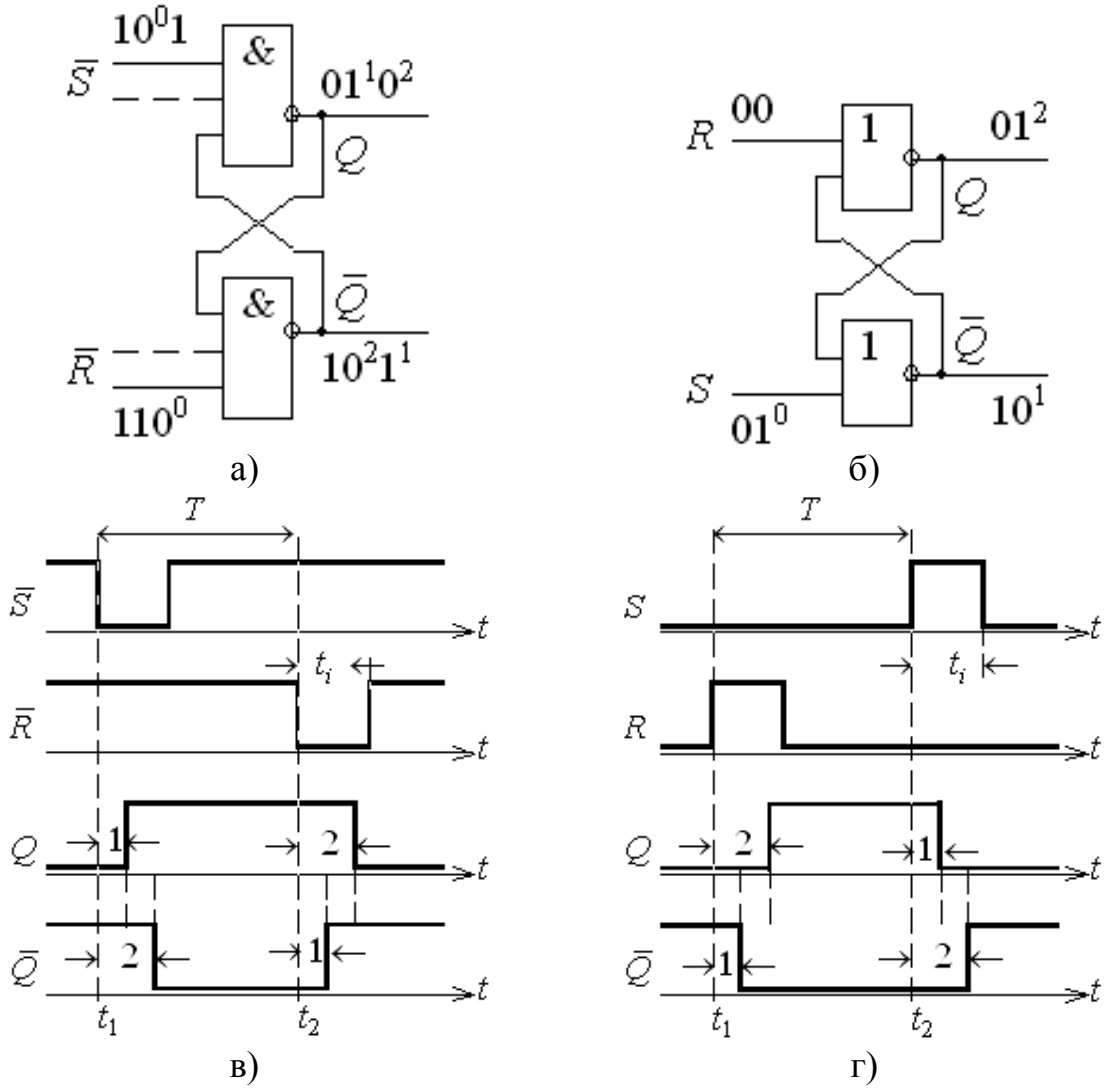


Рисунок 6.2

Тригери $\overline{\mathrm{R}}\,\overline{\mathrm{S}}$ та RS можна перетворити один до одного за допомогою інверторів на їх входах. Кілька інших різновидів асинхронних тригерів відрізняються від RS-тригера переходом під час дії забороненої для нього комбінації відповідно до станів: тригера типу S – до $Q^+ = 1$, типу R – до

$Q^+ = 0$ та типу E (Exclusive – винятковий, особливий) – перебуванням у режимі схову $Q^+ = Q$.

6.1.2.3 Макрофункції. Специфічні для програми макрофункції асинхронних RS-тригерів (елементи з позиційними номерами 1, 2 на рис. 6.3) і ІС жорсткої структури серії 74 (елемент 3) цілком відповідають тригерам з інверсними і прямими входами в базисах І-НЕ та АБО-НЕ (див. рис. 6.2, а), б).

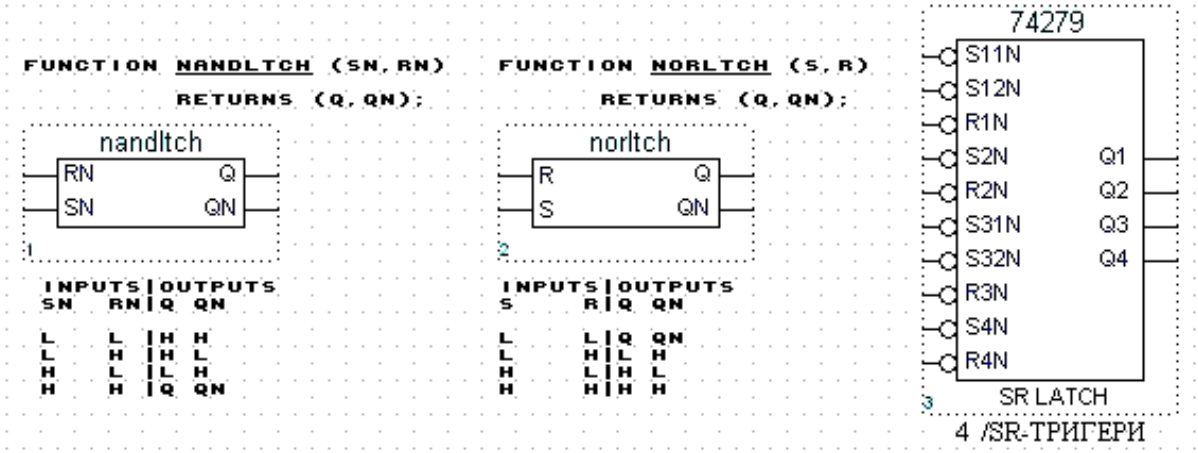
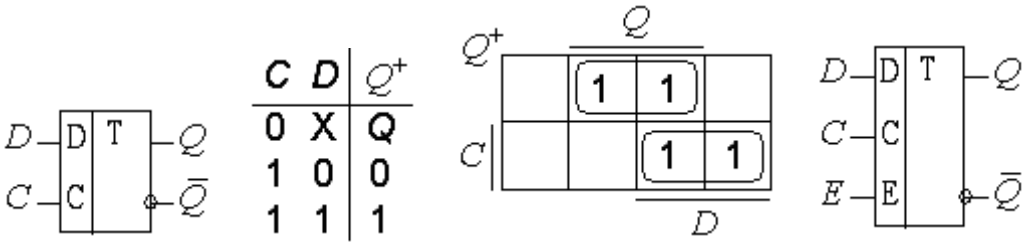


Рисунок 6.3

6.1.3 Синхронні тригери зі статичним керуванням

6.1.3.1 D-тригери. Ознакою синхронних тригерів є наявність входу синхронізації *C* (Clock). D-тригерами називаються синхронні тригери з *однофазним* записом інформації (Data – дані) за входом *D*. Інформація до D-три-гера надходить одним дротом (рис. 6.4, а), що зручно для міжкаскадного сполучення, тому D-тригери набули поширення в інтегральній схемотехніці.



а) б) в) г)

Рисунок 6.4

За відсутності синхроімпульсу ($C = 0$) тригер перебуває в режимі схову $Q^+ = Q$ (рис. 6.4, б), а під час дії активної частини синхроімпульсу ($C = 1$) до нього записується біт інформації $Q^+ = D$. Характеристичне рівняння, отримане з діаграми термів (рис. 6.4, *в*)

$$Q^+ = CD + \bar{C}Q, \qquad (6.3)$$

як і перемикальна таблиця, свідчить, що режим схову забезпечується тільки завдяки синхровходові, отже, асинхронний D-тригер не має сенсу: при

C = 1 за виходом Q він еквівалентний повторювачу, а за виходом $\overline{Q}$ – інвертору. Не має сенсу також тригер з інверсним входом D – досить взаємно замінити позначення виходів Q та $\overline{Q}$ для перетворення тригерів з прямим і інверсним входом D.

Згідно з (6.3) при C = 1 та довільному вихідному стані функція збудження набуває вигляду:

$$D = Q^{+}, \text{ якщо } C = 1,\ Q = \text{X}. \qquad (6.4)$$

Для зручності керування застосовуються тригери різного типу з додатковим входом E (Enable – дозвіл). Так, DE-тригер (рис. 6.4, г) при E = 0 перебуває в початковому стані незалежно від інших сигналів, а при E = 1 перемикається як звичайний D-тригер.

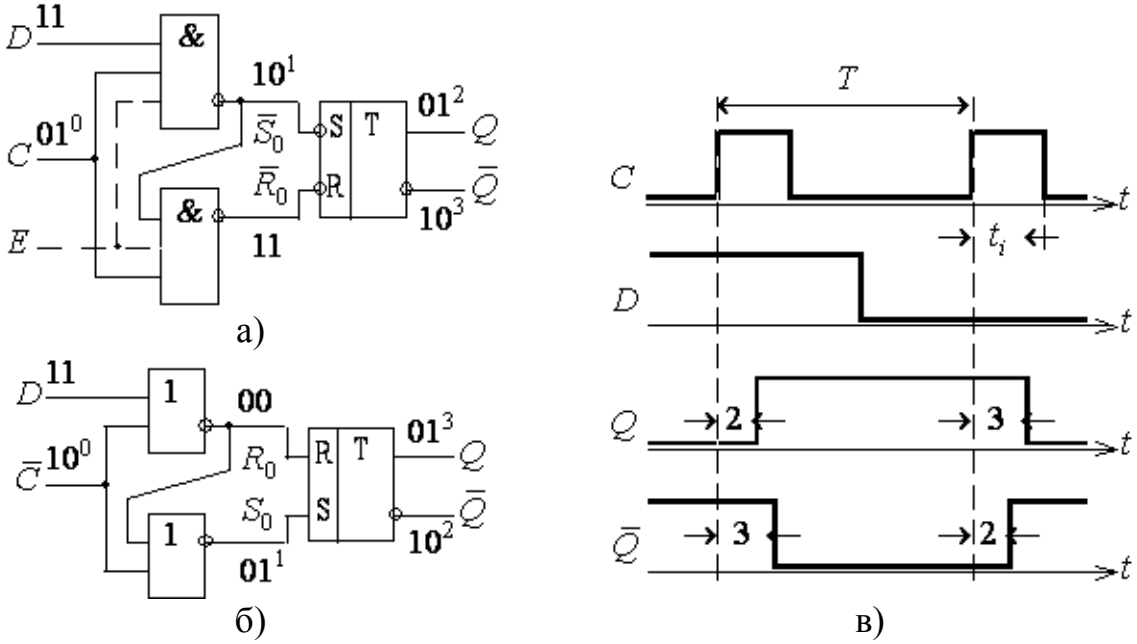


Рисунок 6.5

6.1.3.2 Схемна реалізація D-тригерів. Для схемної реалізації синхронних тригерів зі статичним керуванням (рис. 6.5, а), б) за допомогою додаткових логічних елементів з вхідних інформаційних сигналів формують на входах комірки пам’яті (асинхронного RS-тригера) активні рівні сигналів $\overline{R_0}, \overline{S_0}$ або R_0, S_0 в інтервалах часу, що задаються рівнем синхроімпульсів C, $\overline{C}$, а в проміжках між синхроімпульсами ці рівні мають бути пасивними.

У D-тригері з прямим синхровходом на елементах І-НЕ (див. рис. 6.5, а) пасивним рівнем C = 0 вхідні елементи І-НЕ блокують проходження інформації до комірки пам’яті: незалежно від значення D на обох входах $\overline{R_0}$, $\overline{S_0}$ асинхронного $\overline{\text{R}}\,\overline{\text{S}}$-тригера встановлюються рівні логічної 1 і підтримують його в режимі схову, а з надходженням активного рівня C = 1 вхідні елементи виконують функції інверторів інформаційного

сигналу і тригер функціонує за перемикальною таблицею (рис. 6.4, б). За допомогою додаткових входів (пунктир на рис. 6.5, а) утворюється тригер зі входом дозволу E, який схемно є паралельним входові C і відрізняється від нього лише призначенням: за постійного рівня $E = 0$ тригер блокується, а при $E = 1$ функціонує як звичайний D-тригер.

Внаслідок додаткової затримки сигналів у вхідних елементах І-НЕ, як зазначено позиціями кодів на схемі рис. 6.5, а) та кількістю затримок $t_{з.п}$ на часових діаграмах рис. 6.5, в), тривалість перемикання синхронного тригера збільшується на одну затримку порівняно з асинхронним і становить $t_T = 3t_{з.п}$, тривалість синхроімпульсів для надійного перемикання має бути не меншою за $t_і \geq 3t_{з.п}$, а тому роздільна здатність тригера, тобто період синхроімпульсів $T \geq 4t_{з.п}$ і робоча частота $f \leq 1/4\ t_{з.п}$.

6.1.3.3 Примітив, макро- і мегафункція D-тригерів. Слід звернути увагу на те, що, як зазначено на часових діаграмах, перемикання відбуваються з надходженням імпульсів C, відносно них і відлічуються затримки в елементах. Крім того, протягом $C = 1$ інформаційний сигнал D має бути незмінним, інакше пристрій функціонуватиме як асинхронний. З огляду на це асинхронні тригери (див. рис. 6.3) і синхронні зі статичним керуванням (рис. 6.6) позначаються в програмному забезпеченні терміном Latch („замок”), а їх синхровхід – терміном ENA (Enable – дозвіл) або gate (ворота).

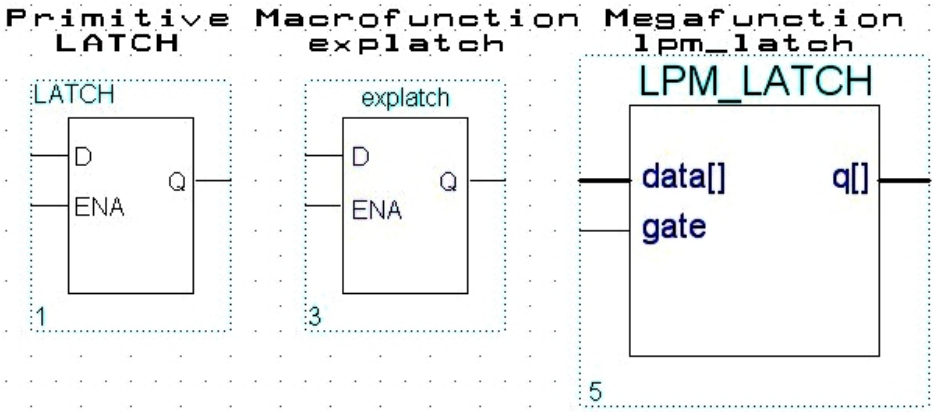


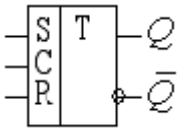
Рисунок 6.6

6.1.3.4 RSC-тригери. Синхронні RS-тригери з прямими (RSC тригер) та інверсними ($\overline{R}\overline{S}\overline{C}$-тригер) входами є тригерами з *двофазним* записом інформації (рис. 6.7, а), б). При $C = 0$ тригер типу RSC (див. рис. 6.7, а) перебуває в режимі схову незалежно від значення інформаційних сигналів R, S (рис. 6.7, в), а при $C = 1$ він функціонує за правилами асинхронного тригера (див. рис. 6.1, д). Так само $R\overline{S}\overline{C}$-тригер (див. рис. 6.7, б) пасивним рівнем $\overline{C} = 1$ підтримується в початковому стані (рис. 6.7, г), а з надходженням активного рівня $\overline{C} = 0$ функціонує як асинхронний $R\overline{S}$-тригер (див. рис. 6.1, е).

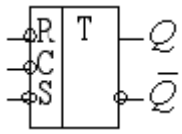
За діаграмою термів RSC-тригера (рис. 6.7, д), складеною, як і раніше, з урахуванням попереднього стану Q, мінімізуємо характеристичне рівняння

$$Q^+ = CS + \overline{CR}\,Q, \text{ якщо } CRS = 0, \qquad (6.5)$$

з обмежувальною умовою, що виключає заборонену комбінацію вхідних сигналів $CRS = 1$. З рівняння виходить, що при $C = 0$ у тригері забезпечується режим схову $Q^+ = Q$, а при $C = 1$ вираз (6.5) перетворюється до (6.1).



а)



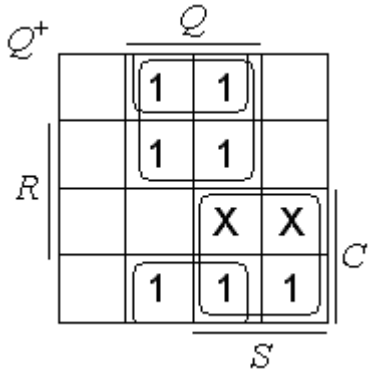
б)

C	R	S	Q^+
0	X	X	Q
1	0	0	Q
1	0	1	1
1	1	0	0
1	1	1	X

в)

$\overline{C}$	$\overline{R}$	$\overline{S}$	Q^+
0	0	0	X
0	0	1	0
0	1	0	1
0	1	1	Q
1	X	X	Q

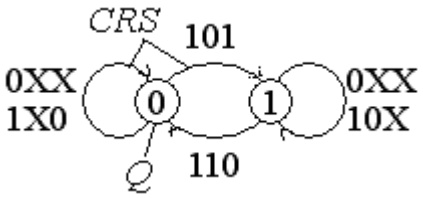
г)



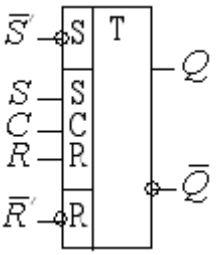
д)

Q	Q^+	C	R	S
0	0	0	X	X
0	0	1	X	0
0	1	1	0	1
1	0	1	1	0
1	1	1	0	X
1	1	0	X	X

е)



ж)



и)

Рисунок 6.7

Таблиця переходів і граф RSC-тригера (рис. 6.7, е), ж) відрізняються лише доповненням стійких станів при $C = 0$ та сигналу $C = 1$ при інших переходах. Часто користуються вкороченою таблицею переходів, яка вказує переходи $Q \to Q^+$ тільки при C=1; тоді вона збігається з таблицею RS-тригера (див. рис. 6.1, и) і функції збудження при $C = 1$ збігаються з (6.2).

За допомогою паралельних входів кінцевого асинхронного тригера (див. рис. 6.1, в), 6.2, а) синхронні тригери різних типів можуть виконуватися з одним, наприклад, R' або двома R' /, S' / установлювальними входами (рис. 6.7, и), призначеними для асинхронного (незалежно від рівнів усіх інших сигналів) установлення до визначеного стану, найчастіше для скидання до $Q = 0$ перед виконанням певної операції. Синхронні тригери типів RC, SC, EC при $C = 0$ зберігають початковий стан, а при $C = 1$ функціонують за правилами відповідних асинхронних тригерів (п. 6.1.2.2).

6.1.3.5 Схемна реалізація RSC-тригерів. Двофазний RSC-тригер зі статичним керуванням аналогічно однофазному D-тригеру будується за узагальненою схемою рис. 6.8, а). Логічними елементами пристрою керування ПК з вхідних сигналів X формуються сигнали R_0, S_0 на входах комірки пам'яті. У RSC-тригері (рис. 6.8, б) вхідні елементи з надходженням активного рівня $C = 1$ виконують функції інверторів інформаційних сигналів, тому однойменні входи всього пристрою S, R стають прямими і тригер функціонує за перемикальною таблицею на рис. 6.7, *в)*. За допомогою додаткових входів асинхронного тригера (пунктир) виконують тригери з установлювальними входами.

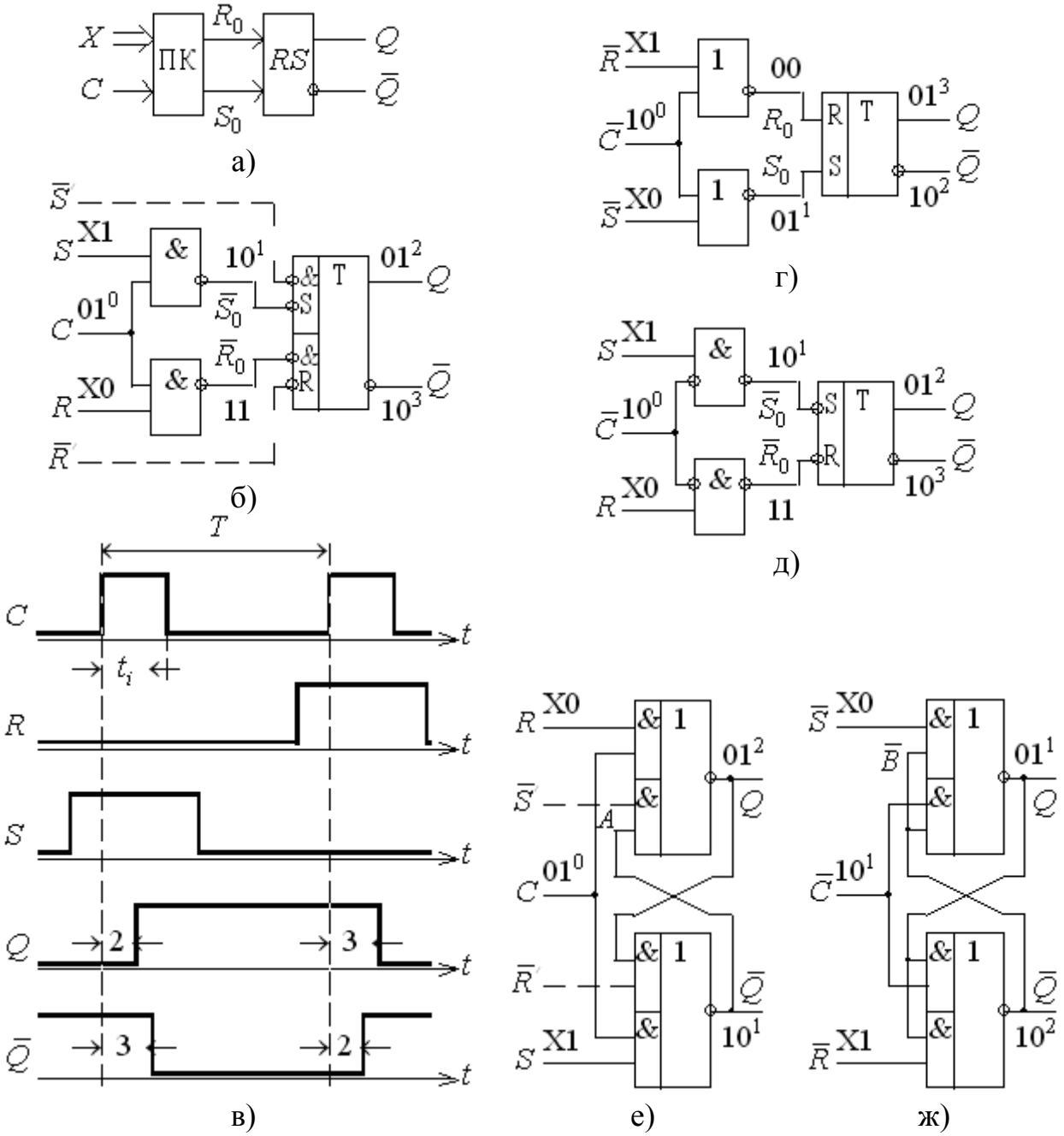


Рисунок 6.8

Внаслідок додаткової затримки сигналів у вхідних елементах І-НЕ, як зазначено позиціями кодів на схемі рис. 6.8, б) та кількістю затримок *t*з.п на часових діаграмах рис. 6.8, в) тривалість перемикання синхронного тригера збільшується на одну затримку порівняно з асинхронним і становить $t_T = 3t_{з.п}$, тому потрібна тривалість синхроімпульсів і робоча частота такі самі, як у D-тригера.

Дуальний $\overline{R}\overline{S}C$-тригер виконується на елементах АБО-НЕ (рис. 6.8, г), а використання інверсного входу елемента І-НЕ (елемент заборони з інверсією або імплікатор у схемотехніці Т-ТТЛ) дозволяє досить просто побудувати RSC-тригер з інверсним синхровходом (рис. 6.8, д) на елементах того самого типу технології (ТТЛ), що й тригер з прямими входами. Шляхом суміщення функцій вхідних елементів і комірки пам'яті можна побудувати синхронні RS-тригери на елементах І-АБО-НЕ з прямими та інверсними входами (рис. 6.8, е), ж). Завдяки різнополярному керуванню парами тригерів на рис. 6.8, б), г) та рис. 6.8, е), ж) спрощується виконання на них двоступінчастих тригерів в єдиному технологічному циклі. У сучасних серіях ІС статичні RSC-тригери окремо майже не випускаються, вони широко застосовуються у складі тригерів з динамічним керуванням.

6.1.4 Тригери з динамічним керуванням

6.1.4.1 Т-тригери. Т-тригером (від Togle – перемикач станів) називається тригер з лічильним входом *C* (рис. 6.9, а), який при *C* = 0 перебуває в режимі схову $Q^+ = Q$ (рис. 6.9, б), а з надходженням активної частини чергового імпульсу (позначеної як *C* = 1) змінює свій стан на протилежний $Q^+ = \overline{Q}$. Т-тригер здійснює лічбу за модулем 2, бо з кожним вхідним імпульсом перемикається між двома станами 0 та 1, тому називається також *лічильним тригером*.

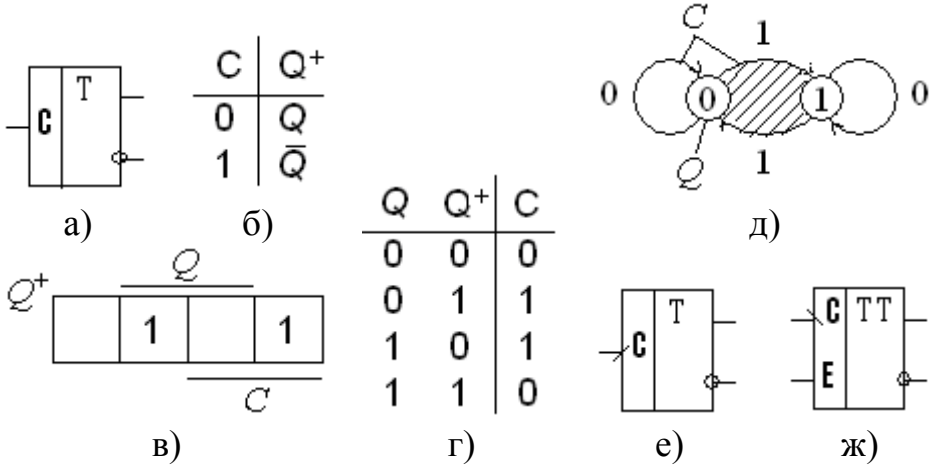


Рисунок 6.9

Згідно з діаграмою термів (рис. 6.9, *в*) його характеристичне рівняння можна виразити через операцію додавання за модулем 2:

$$Q^+ = C\,\overline{Q} + \overline{C}\,Q = C \oplus Q, \qquad (6.6)$$

що відповідає таблиці переходів (рис. 6.9, *г*). Підставляючи до (6.6) $Q = 0$, отримаємо $C = Q^+$, а при $Q = 1$ маємо $C = \overline{Q^+}$, що можна коротко відобразити у вигляді функції збудження:

$$C = Q^+, \text{ якщо } Q = 0;\ C = \overline{Q^+}, \text{ якщо } Q = 1. \qquad (6.7)$$

Граф тригера (рис. 6.9, д), побудованого на одному елементарному потенціальному тригері з двома можливими станами 0 та 1, при $C = 1$ утворює замкнене коло (заштриховане), тобто тригер переходитиме безперервно з одного стану до іншого, поки $C = 1$, як автоколивальний генератор. Отже, тригер буде *нестійким*, якщо його граф утворює *замкнене коло з однаковими сигналами* біля гілок (крім петель). Тому не можна побудувати стійкий тригер на одному елементарному потенціальному тригері, якщо його перемикальна таблиця містить перехід $Q^+ = \overline{Q}$.

Усунути цей недолік і, отже, реалізувати стійкий лічильний тригер можна, якщо активна частина імпульсу C закінчиться після його переходу з одного стану до іншого, тобто за допомогою динамічного керування (рис. 6.9, е). Інший шлях полягає в доповненні графа більшою кількістю станів (завдяки чому замкнене коло розривається), що можна здійснити в тригерному пристрої, складеному, наприклад, з двох потенціальних тригерів.

ТЕ-тригер (рис. 6.9, ж) з додатковим входом E (від Enable –дозвіл) дозволяє гнучкіше будувати послідовнісні пристрої. Якщо вхід E використовувати як керувальний, під час $E = 0$ тригер не перемикається, а при $E = 1$ функціонує як лічильний тригер. За використання входу E як синхровходу утворюється синхронний лічильний тригер, моменти перемикання якого задаються не зміною інформації на лічильному вході, а надходженням синхроімпульсів.

6.1.4.2 JK-тригери. Аналогічно RSC-тригерам JK-тригери (рис. 6.10, а) мають крім синхровходу C два інформаційні входи: J (Jerk – раптове вмикання) – вхід установлення до стану логічної 1 та K (Kill – раптове вимикання) – вхід скидання до стану логічного 0, тобто вхід J відповідає входові S тригера типу RSC, а вхід K – входові R. Поширений різновид JKE-тригера (рис. 6.10, б) має, як і ТЕ-тригер, вхід дозволу E. JK тригер функціонує як RSC-тригер (на рис. 6.10, в) наявність активного перепаду синхроімпульсу позначено стрілкою, а його відсутність – рискою), але на відміну від нього не має забороненої комбінації вхідних сигналів: при $CJK = 1$ (у рівняннях наявність активного перепаду синхроімпульсу позначатимемо як $C = 1$, а його відсутність – як $C = 0$) він перемикається до протилежного стану $Q^+ = \overline{Q}$, тобто як Т-тригер. Внаслідок цього для забезпечення стійкості JK-тригери виконуються з динамічним керуванням, наприклад, за двоступінчастою схемою, як зазначено на рис. 6.10, б).

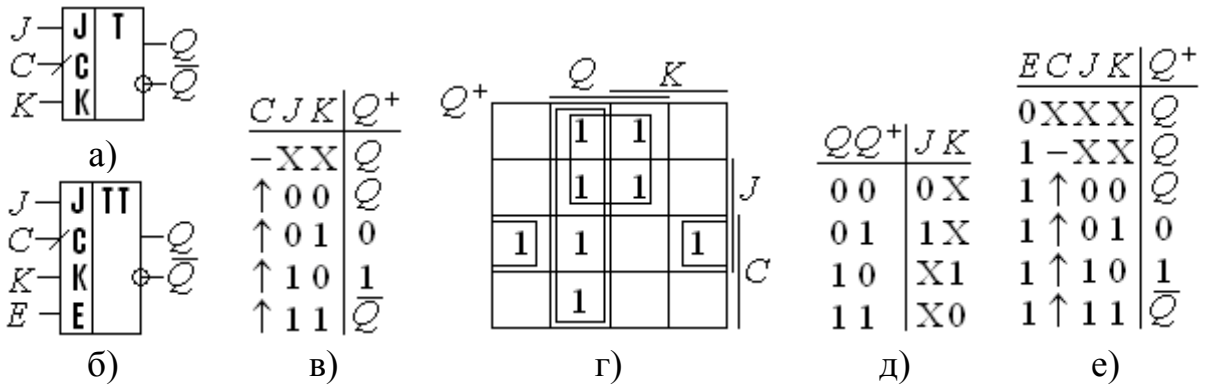


Рисунок 6.10

Із характеристичного рівняння JK-тригера, мінімізованого за допомогою діаграми термів (рис. 6.10, г)

$$Q^+ = CJ\overline{Q} + \overline{C}Q + \overline{K}Q = CJ\overline{Q} + \overline{CK}Q, \qquad (6.8)$$

та перетвореного при $C = 1$ до рівняння асинхронного JK-тригера

$$Q^+ = J\overline{Q} + \overline{K}Q, \qquad (6.9)$$

можна визначити, які сигнали необхідно подати на його інформаційні входи аби з надходженням синхроімпульсу він перемикався до потрібного стану Q^+. Так, якщо $Q = 0$, з (6.9) дістанемо $J = Q^+$ при довільному значенні K = X, а якщо $Q = 1$, то $K = \overline{Q^+}$ при J = X. Лаконічно це можна відобразити у вигляді функцій збудження

$$J = Q^+,\ K = \text{X},\ \text{якщо } Q = 0;\ \ J = \text{X},\ K = \overline{Q^+},\ \text{якщо } Q = 1, \qquad (6.10)$$

або вкороченої таблиці переходів (рис. 6.10, д), дійсної при $C = 1$. З таблиці видно, що переходи $Q \to Q^+$ до протилежного стану можуть відбуватись у режимі як RS-тригера, так і T-тригера при $J = K = 1$.

Для JKE-тригера (див. рис. 6.10, б) перемикальна таблиця (рис. 6.10, е) при $E = 1$ відповідає таблиці JK-тригера, а при $E = 0$ зберігається початковий стан $Q^+ = Q$.

6.1.4.3 Реалізація динамічних тригерів за двоступеневою схемою MS. У мікросхемотехніці набули поширення універсальні потенціальні тригери за двоступеневою схемою MS (Master-Slave – господар-невільник, основний-допоміжний). Ідея побудови такої схеми ґрунтується на *двотактному* принципі дії пристрою, складеному з двох синхронних тригерів M та S зі статичним керуванням (рис. 6.11, а).

У тригері за схемою MS з інвертором синхроімпульси C та $\overline{C}$ на входах тригерів M, S рознесено в часі, тому вони перемикатимуться по черзі. У першому такті при $C = 1$, $\overline{C} = 0$ вхідна інформація записується до тригера M, а тригер S залишається в початковому стані; у другому – при $C = 0$, $\overline{C} = 1$, навпаки, тригер M перебуває в режимі схову, а інформація з його виходів переписується до тригера S. Уведенням перехресних зворотних зв’язків з вихо-

дів на входи (пунктир) можна реалізувати режим Т-тригера: за кожні два такти пристрій перемикатиметься до протилежного стану. Головна особливість схеми полягає в тому, що під час перемикання кожного тригера інформація на його входах залишатиметься сталою, бо коли перемикається один з них, другий перебуває в режимі схову. Це зумовлює стійку роботу пристрою, який, на відміну від одноступінчастого потенціального тригера зі зворотними зв'язками, не зможе перемкнутись протягом такту більше одного разу.

Розглянемо докладніше функціонування схеми в режимі RSC-тригера (без пунктирних зворотних зв'язків на рис. 6.11, а).

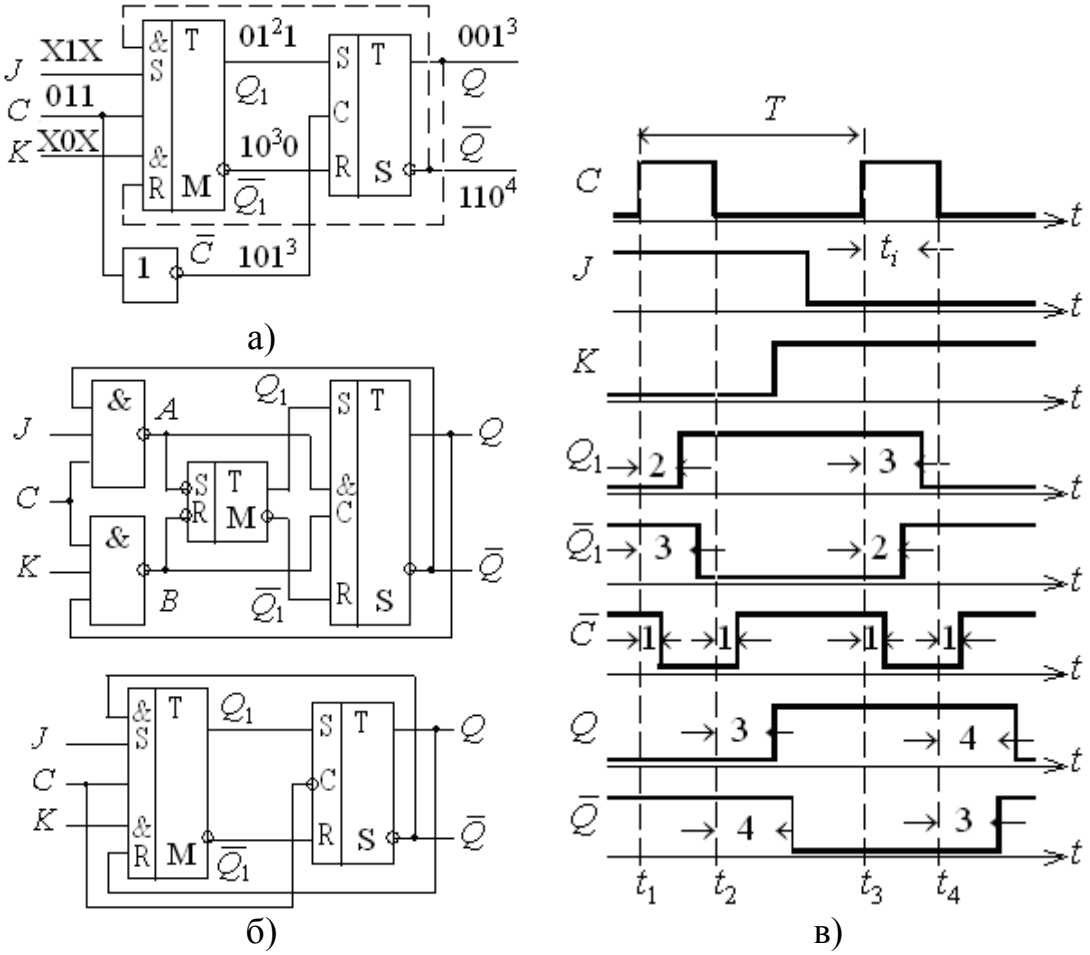


Рисунок 6.11

Хай *у початковому стані*, позначеному першою позицією кодів на схемі та рівнями напруг при $t < t1$ на часових діаграмах (рис. 6.11, б), за відсутності синхроімпульсу тригер М перебуває в режимі схову, наприклад, $Q_1 = 0$, а рівнем $\overline{C} = 0$ тригер S підтримується в стані тригера М: $Q = Q_1 = 0$.

На етапі підготовки (друга позиція кодів, момент $t \geq t_1$) з надходженням синхроімпульсу C інформація зі входів $J \equiv S$ та $K \equiv R$ записується до тригера М як у звичайному RSC-тригері (див. рис. 8,а,б), тому етап підготовки продовжується протягом $t_{T1} = 3t_{з.п}$ і для надійного перемикання тригера тривалість синхроімпульсів має бути $t_i > 3t_{з.п}$. При цьому рівнем $\overline{C} = 0$ тригер S підтримується в початковому стані.

На етапі перемикання (третя позиція кодів, $t \geq t_2$), після закінчення синхроімпульсу тригер М залишається в режимі схову, а рівнем $\overline{C} = 1$ тригер S перемикається до стану М. Через наявність інвертора в колі синхроімпульсу етап перемикання збільшується на одну затримку і триває $t_{T2} = 4t_{з.п}$. Отже, разом час затримки перемикання тригера становить $t_T = t_{T1} + t = 7t_{з.п}$. Проте, він встигне перемкнутись, якщо пауза між синхроімпульсами буде $3t_{з.п}$, тому мінімальний період синхроімпульсів становить $T = 6t_{з.п}$ і максимальна робоча частота $f = 1/6t_{з.п}$. Перемикання тригера у зворотному напрямку в моменти t_3, t_4 відбувається аналогічно.

Отже, за позитивним фронтом синхроімпульсу вхідна інформація записується до тригера М, а за негативним його фронтом – до тригера S. Якщо синхронізувати тригер короткими позитивними імпульсами C, то з точки зору споживача вихідної інформації Q вона затримується на тривалість імпульсу t_i, тому такі тригери називають тригерами з внутрішньою затримкою або з внутрішньою пам'яттю. Час затримки поширення сигналу в розглянутому тригері відносно негативного фронту синхроімпульсу становить $t_T = t_{T2} = 4t_{з.п}$.

У RSC-тригері, як звичайно, не можна одночасно подавати рівні логічної 1 на обидва інформаційні входи. Якщо ввести перехресні зворотні зв'язки (пунктир), отримаємо JK-тригер, який функціонує аналогічно RSC тригеру. У стані, наприклад, $Q = 0$ при $J = 1$, $K = 0$ з надходженням синхроімпульсу сигналами на входах першого ступеня $S_1 = J\overline{Q} = 1$, $R_1 = KQ = 0$ він так само, у два етапи, перемикається до стану $Q = 1$. Після перемикання тригера М, при $C = 1$, зміна вхідної інформації не сприйматиметься. Дійсно, рівнем $C = 0$ тригер S підтримується в певному стані, наприклад, $Q = 0$, тому кон'юнкція на вході $R_1 = 0$ не залежить від рівня K, а на вході $S_1 = J$ (бо $Q = 1$) сигнал завади не здатний перемкнути тригер M: за обох можливих комбінацій $S_1 = 1$, $R_1 = 0$ та $S_1 = R_1 = 0$ лише підтверджується стан $Q_1 = 1$, тобто схема функціонує як *тригер з динамічним керуванням*.

Перевагою JK-тригера є некритичність до зміни інформації на входах: її заборонено змінювати лише протягом $3t$з.п після надходження позитивного фронту синхроімпульсу, коли перемикається тригер М; якщо імпульси C короткі, можна цю вимогу спростити, дозволивши зміну сигналів J, K в паузах між синхроімпульсами.

Друга відміна від RSC-тригера полягає в тому, що при $J = K = 1$ схема зі зворотними зв'язками функціонує як Т-тригер. Дійсно, рівні логічної 1 на інформаційних входах не змінюють кон'юнкцію, тому інформація зчитуватиметься зі зворотних зв'язків. У стані $Q = 0$ сигналами на входах $S_1 = J\overline{Q} = 1$, $R_1 = KQ = 0$ з надходженням синхроімпульсу в момент t_1 тригер перемкнеться до протилежного стану $Q = 1$ так само, як показано на діаграмах (тільки рівні $J = K = 1$ залишаються сталими), а в момент $t_з$ – до стану $Q = 0$.

Під час функціонування схеми на входах асинхронного тригера у складі ступеня М – у точках *A,B* ($\overline{R_0}$, $\overline{S_0}$ на рис. 6.8, а) формуються імпульси, які можна використати для синхронізації другого ступеня, скоротивши інвертор (рис. 6.11, в). Утворений таким чином різновид *схеми MS із забороненими зв'язками* функціонує аналогічно схемі MS з інвертором: під час перемикання тригера М хоча б один із сигналів $\overline{R_0}$, $\overline{S_0}$ дорівнює логічному 0, який подібно до сигналу $C = 0$ забороняє перемикання тригера S, а в проміжках між синхроімпульсами встановлюються рівні $\overline{R_0} = \overline{S_0} = 1$ і тригер S переходить до стану М.

У *схемі MS з різнополярним керуванням* (рис. 6.11, г) використовуються тригери М та S із протилежним активним рівнем *C* (див. рис. 6.8, а), д), тому при безпосередньому з'єднанні синхровходів до одного з цих тригерів дозволено запис інформації, а другий перебуває в режимі схову.

З огляду на відсутність інвертора в схемах на рис. 6.11, в), г) час затримки тригера становить $t_T = t_{T1} = t_{T2} = 3t_{з.п}$, проте мінімальний період і робоча частота синхроімпульсів такі ж самі: $t_T = 6t_{з.п}$, $f = 1/6t_{з.п}$.

6.1.4.4 Перетворення тригерів з динамічним керуванням. В інтегральній схемотехніці з-поміж тригерів з динамічним керуванням найбільшого поширення набули тригери типів JK та D, які є *універсальними*, бо з'єднанням їх входів безпосередньо або за допомогою зовнішніх елементів можна побудувати тригери потрібного типу.

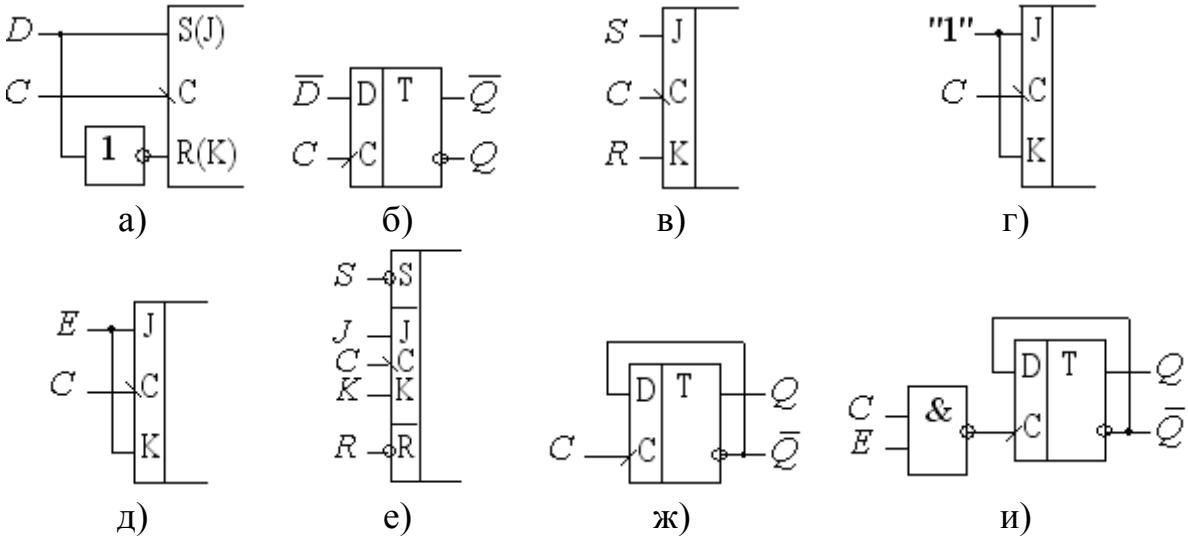


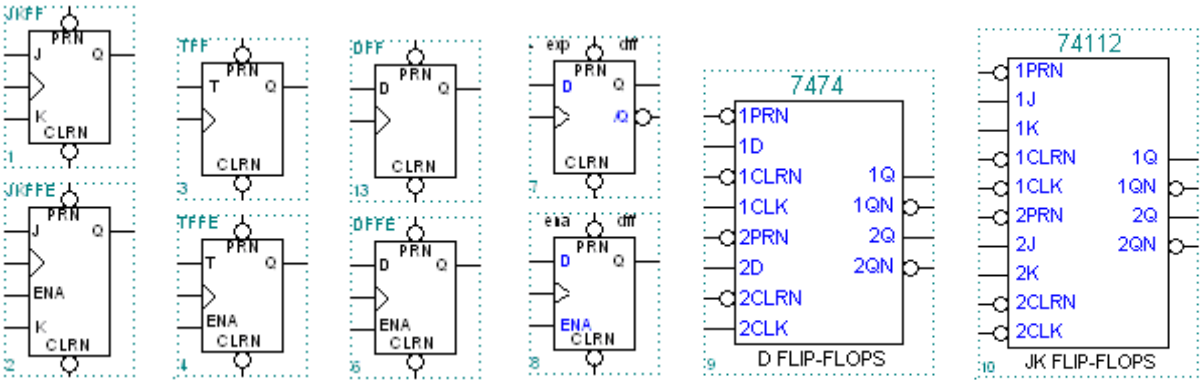
Рисунок 6.12

Тригери типу RSC або JK перетворюються до однофазного D-тригера з використанням зовнішнього інвертора (рис. 6.12, *а*). Якщо вхід *D* має бути інверсним, досить інвертор ввімкнути до протилежного входу двофазного тригера відносно рис. 6.12, *а)* або просто перейменувати виходи тригера (рис. 6.12, б). Природно, такі перетворення придатні для тригерів як зі статичним, так і з динамічним керуванням.

Від RSC-тригера з динамічним входом C легко перейти до JK-тригера за допомогою дубльованих входів (пунктир на рис. 6.11, а) або зовнішніх елементів І. JK-тригер безпосередньо виконує функції *RSC*-тригера з динамічним керуванням (фрагмент умовного позначення на рис. 6.12, в), якщо заборонити комбінацію $CJK = 1$. Реалізація Т-тригера на JK-тригері (рис. 6.12, г) очевидна: при $J = K = 1$ характеристичне рівняння (6.8) JK-тригера перетворюється до характеристичного рівняння (6.6) Т-тригера. На підставі того, що при $J = K = 0$ тригер залишається в режимі схову, а при $J = K = 1$ перемикається до протилежного стану, реалізується ТЕ-тригер (рис. 6.12, д). Аналогічно рис. 6.8, а) за допомогою асинхронних входів, наприклад, R, S (рис. 6.12, е) утворюються тригери з установлювальними входами типів JKRS, TRS, DRS тощо.

Якщо в D-тригері з *динамічним* керуванням вхід D з'єднати з виходом $\overline{Q}$, то при $D = \overline{Q}$ його рівняння (6.3) перетворюється в (6.6), тобто схема стає Т-тригером (рис. 6.12, ж). За допомогою зовнішнього елемента І-НЕ організується ТЕ-тригер (рис. 6.12, и), який перемикатиметься за негативним фронтом синхроімпульсів, бо при $E = 1$ вони потрапляють до прямого динамічного входу C через інвертор.

6.1.4.5 Примітиви і макрофункції тригерів з динамічним керуванням. Бібліотека програмного комплекту містить тригери з динамічним керуванням типу JK, JKE (рис. 6.13, а), T, TE (рис. 6.13, б), D, DE (рис. 6.13, в), до назви яких долучається абревіатура FF (Flipflop). Прямий динамічний синхровхід позначається на символі трикутничком, а для інверсного додається ще маленьке коло. Тригери виконано з асинхронними установлювальними входами CLRN (Clear – скидання, очищення) та PRN (Preset – передустановлення); до назви інверсних входів додається літера N (Not). Спеціалізовані для пакета макрофункції (рис. 6.13, г) відтворюють, по суті, тригери типу D і DE, а типові макрофункції серії 74 (рис. 6.13, д), е) містять, в основному, тригери типу D і JK (по декілька тригерів у корпусі ІС).



а) б) в) г) д) е)

Рисунок 6.13

6.1.5 Застосування тригерів у текстовому редакторі

У графічному редакторі примітиви, макро- і мегафункції тригерів і інших елементів бібліотеки ЦПП застосовують так само, як і елементи бібліотеки ЦКП. Проте в текстовому редакторі використовують спеціальні засоби програмного забезпечення.

Примітиви тригерів запроваджують до проекту за допомогою підсекції оголошення регістрів/тригерів ***Register Declaration*** секції змінних Variable Section і далі застосовують в булевих рівняннях Boolean Equation логічної секції Logic Section так само, як і зразки інших елементів. При цьому якщо зразок має лише один вхід або вихід, ім'я цього виводу в булевих рівняннях можна не зазначати: у лівій частині рівнянь ім'я зразка означатиме вхід, а в правій частині – його вихід. *Макрофункції* тригерів попередньо, зазвичай, вносять до складу проекту (Include Statement) із зазначенням їх імен і параметрів в підсекції Instance Declaration (parametererized) секції змінних (Variable Section) і далі застосовують в булевих рівняннях.

6.2 Лабораторне завдання

6.2.1 Дослідити основні типи тригерів

6.2.1.1 Дослідити асинхронний RS-тригер з інверсними входами (RS/-тригер) на елементах I-НЕ: за принциповою електричною схемою та осцилограмами сигналів (файли 6asyn.gdf, .scf) скласти таблиці відповідності (перемикальну) і переходів, мінімізовані рівняння вихідних функцій, виміряти затримку вихідних імпульсів; розглянути особливості побудови і перемикання макрофункції цього тригера та принципової схеми і макрофункції асинхронного RS-тригера з прямими входами (RS-тригер). У звіті навести також умовне графічне позначення за ДСТУ таких тригерів зі стислим поясненням принципу їх дії та осцилограм сигналів.

6.2.1.2 Дослідити синхронний D-тригер зі статичним керуванням (Latch – защіпка) на елементах I-НЕ за п. 6.2.1.1 (файли d:\max2work\tutorial\6lab\6stat.gdf, .scf) та розглянути особливості побудови і перемикання макрофункції і мегафункції цього тригера та схеми синхронного RS-тригера (RSC-тригер).

6.2.1.3 Дослідити синхронний JK-тригер з динамічним керуванням структури MS (схема 1) за п. 6.2.1.1 (файли d:\max2work\tutorial\6lab\6dyn.gdf, .scf); розглянути особливості побудови і перемикання базових примітивів JKFF, TFF і DFF (схеми 2, 3, 4), T-тригера на основі D-тригера (схема 5), макрофункції D-тригера expdff (схема 6), T-тригера на основі типової макрофункції JK-тригера (схема 7), мегафункції D-тригера lpm_dff (схема 8) та універсальної мегафункції lpm_ff (схе-

ма 9), налаштованої на виконання функції D-тригера з інверсним динамічним керуванням.

6.2.1.4 Ознайомитися з різновидами тригерів бібліотеки бази даних (файл d:\max2work\tutorial\6lab\6libre.gdf): примітивами, макрофункціями (у тому числі вибраними ІС серії 74) та мегафункціями. Розглянути функціональні прототипи (FUNCTION), таблиці відповідності та інші довідкові дані (див. п. 1.1.2, в), пояснити призначення та особливості входів (основних) і виходів.

6.2.2 Застосувати тригери в *графічному* редакторі для зберігання вхідних даних з подальшою реалізацією на їх основі логічної функції, що задана згідно з варіантом XX: за дóзвільного рівня *Enable* синхроімпульсом *Clock* записати до тригерів дані і сформувати на їх виходах потрібну логічну функцію

6.2.2.1 Скласти пристрій на примітивах тригерів вибраного типу і логічних елементів у графічному файлі 6XXy.gdf проекту 6XXy за структурною схемою 6str.gdf, виконати компіляцію і моделювання та переконатися в правильності його функціонування за часовими діаграмами 6XXy.scf.

🖫 ***Приклад:*** $y = d_2\overline{d}_1 d_0 \rightarrow$ d:\max2work\tutorial\6lab\600y.gdf (схема 1), d:\max2work\tutorial\6lab\600y.scf (вихід *y*).

6.2.2.2 Виконати п. 2.1 на ***мегафункціях тригерів*** вибраного типу і логічних елементів (див. п. 4.2.2).

🖫 ***Приклад:*** d:\max2work\tutorial\6lab\600y.gdf (схема 2), d:\max2work\tutorial\6lab\600y.scf (вихід *y*1).

6.2.2.3 Виконати п. 6.2.2.1 на автоматично створених за допомогою ***менеджера MegaWizard Plug-In Manager*** різновидах мегафункцій тригерів і логічних елементів (див. п. 4.2.5.1).

☝ ***Примітки:***

1. Для створення тригера на сторінці 2a діалогового вікна менеджера слід вибрати категорію storage і мегафункцію LPM_FF, на сторінці 3 – кількість тригерів, їх тип та встановити прапорець входу дозволу, а сторінку 4 щодо необов'язкових (optional) установчих входів – пропустити. Під час створення логічної функції входи доцільно згорнути в шину: на сторінці 3 діалогового вікна встановити прапорець Show input as bus.

2. У графічному файлі d:\max2work\tutorial\6lab\600y.gdf (схема 1) для прикладу наведено перетворення RS- і JK-тригерів у D-тригер та інвертування біта даних на виході тригера (схема 1), або безпосередньо на вході мегафункції (схема 2), або за допомогою вхідного інвертора (схема 3). При цьому назви ліній і шин мають відрізнятися (у прикладі data і dat), адже сигнали в них з урахуванням інверсії є різні.

🖫 ***Приклад:*** d:\max2work\tutorial\6lab\600wiz_3d.sym, d:\max2work\tutorial\6lab\600wiz_3and.sym, d:\max2work\tutorial\6lab\600y.gdf (схема 3), d:\max2work\tutorial\6lab\600y.scf (вихід *y*2).

6.2.3 Застосувати тригери в *текстовому* редакторі виконанням завдання за п. 6.2.2

6.2.3.1 Створити пристрій на основі примітивів тригерів вибраного типу у тестовому файлі 6XXreg_decl.tdf проекту 6XXreg_decl на кшталт схеми 1 файла 6XXy.gdf, для чого в секцію змінних (Variable Section) вставити шаблони підсекції оголошення регістрів/тригерів (**Register Declaration**), в яких ввести умовне ім'я зразка (на схемі d:\max2work\tutorial\6lab\600y.gdf зазначено в дужках) та через двокрапку назву вибраного типу тригера; відтак у логічну секцію (Logic Section) вставити шаблон підсекції булевих рівнянь (Boolean Equation), за допомогою яких утворити з'єднання на зразок графічного файла. Перевірити правильність функціонування пристрою за часовими діаграмами 6XXreg_decl.scf.

🖫 ***Приклад:*** d:\max2work\tutorial\6lab\600reg_decl.tdf, .scf.

```
SUBDESIGN 600reg_decl
(
    E, C, d[2..0]   : INPUT ;
    y               : OUTPUT;
)
VARIABLE
    de      : DFFE;
    sre     : SRFFE;
    jke     : JKFFE;
BEGIN
    de.D=d2; de.CLK=C; de.ENA=E; sre.S=d1; sre.R=!d1;
    sre.CLK=C;  sre.ENA=E; jke.J=d0; jke.K=!d0; jke.CLK=C;
    jke.ENA=E; y=de.Q & !sre.Q & jke.Q;
END;
```

☝ ***Примітка.*** Якщо зразок має лише один вхід або вихід, ім'я цього виводу в булевих рівняннях можна не зазначати: у лівій частині рівнянь ім'я зразка означатиме вхід, а в правій частині – його вихід.

🖫 ***Приклад:*** d:\max2work\tutorial\6lab\600reg_decl1.tdf**,** .scf.

```
VARIABLE
    t1      : DFFE;
    t2      : SRFFE;
    t3      : JKFFE;
BEGIN
    t1.D=d2; t1.CLK=C; t1.ENA=E;   t2.S=d1; t2.R=!d1;
    t2.CLK=C; t2.ENA=E; t3.J=d0; t3.K=!d0; t3.CLK=C;
    t3.ENA=E;    y=t1 & !t2 & t3;
END;
```

6.2.3.2 Виконати п. 6.2.3.1 на ***мегафункціях*** тригерів вибраного типу і логічних елементів (див. п. 4.1.3) у тестовому файлі 6XXmega_y.tdf проекту 6XXmega_y на кшталт схеми 2 файла 6XXy.gdf, для чого включити до складу проекту вибрані мегафункції тригера і логічного елемента (Include Statement) із зазначенням їх імен і параметрів в підсекції Instance Declaration (parametererized) секції змінних (Variable Section) та в логічну секцію (Logic Section) вставити шаблон підсекції булевих рівнянь (Boolean Equation), за допомогою яких утворити з'єднання на зразок графічного файла. Перевірити правильність функціонування пристрою за часовими діаграмами 6XXmega_y.scf.

🖫 ***Приклад:*** d:\max2work\tutorial\6lab\600mega_y.tdf, .scf.

6.2.3.3 Виконати п. 6.2.3.1 ***на автоматично створених за допомогою менеджера MegaWizard Plug-In Manager*** різновидах мегафункцій тригерів і логічних елементів (див. п. 4.2.4.1) у тестовому файлі 6XXwiz_y.tdf проекту 6XXwiz_y на кшталт схеми 3 файла 6XXy.gdf, для чого включити до складу проекту створені в п. 2.3 різновиди мегафункцій тригера і логічного елемента (Include Statement) без зазначення їх параметрів в підсекції Instance Declaration (non-parametererized) секції змінних (Variable Section) та в логічну секцію (Logic Section) вставити шаблон підсекції булевих рівнянь (Boolean Equation), за допомогою яких утворити з'єднання на зразок графічного файла. Перевірити правильність функціонування пристрою за часовими діаграмами 6XXwiz_y.scf.

🖫 ***Приклад:*** d:\max2work\tutorial\6lab\600wiz_y.tdf, 600wiz_y.scf.

Контрольні питання та завдання

1. Складіть принципову електричну схему комірки пам'яті з двовходових базових елементів типу а) ТТЛ, б) ЕСЛ, в) МОН ТЛ (елемент АБО-НЕ), г) КМОН ТЛ (елемент І-НЕ) і покажіть: 1) які величини напруг діють на входах і виходах тригера в обох його статичних станах та чому тригер може перебувати в них необмежено довго; 2) в якому стані опиниться цей тригер, якщо один з його входів перебуває під рівнем а) логічного 0, б) логічної 1, а другий вхід підімкнено таким чином: а) закорочено на землю, б) залишено вільним, в) підімкнено (через резистор) до джерела живлення, г) з'єднано з першим входом; 3) в якому стані опиниться цей тригер з надходженням забороненої для нього комбінації вхідних сигналів та при якій комбінації, що настає після забороненої, він перейде до невизначеного стану, а при якій – до визначеного; 4) у чому полягають умови, за яких можливе стрибкоподібне перемикання цього тригера, та як воно відбувається в даній схемі; 5) при якій мінімальній тривалості вхідних імпульсів надійно спрацьовуватиме зображений тригер та з якою максимальною частотою він зможе перемикатись (цифрові дані наведіть на прикладі конкретної серії ІС зазначеної елементної бази).

2. Наведіть а) перемикальну таблицю, б) таблицю переходів, в) характеристичні рівняння в МДНФ та МКНФ відносно прямого та інверсного виходів, г) логічні функції збудження для тригерів типу: 1) R, 2) $\overline{S}$, 3) E, 4) RSC, 5) RC, 6) SC, 7) EC, 8) DC, 9) DCE, 10) TE, 11) $\overline{T}$E, 12) JKC, 13) JK$\overline{C}$, 14) JKE, 15) асинхронний JK.

3. Побудуйте граф перемикань та за допомогою позицій станів на схемі і часових діаграм охарактеризуйте швидкодію і вимоги щодо тривалості вхідних імпульсів і інтервалів часу, на яких заборонено змінювати вхідну інформацію для тригерів такого типу (через риску зазначено елементи або схема, за якими складено тригер):

а) асинхронних: 1) RS – І-НЕ, 2) RS – АБО-НЕ;

б) синхронних, керованих рівнем: 1) RSC – І-НЕ, 2) RSC – АБО-НЕ, 3) $\overline{R}\,\overline{S}\,\overline{C}$ – І-АБО-НЕ, 4) RSC – І-АБО-НЕ, 5) DC – АБО-НЕ, 5) D$\overline{C}$ – І-АБО-НЕ та НЕ, 7) DC – І-АБО-НЕ, 8) DC – І-НЕ, 9) D$\overline{C}$ – АБО НЕ, 10) $\overline{D}\,\overline{C}$ – АБО-НЕ;

в) за схемою MS: 1) RSC – з інвертором, 2) JKC – із забороненими зв'язками, 3) T – з різнополярним керуванням на синхронних RS-тригерах, 4) D – з різнополярним керуванням на однофазних D-тригерах;

г) за схемою трьох тригерів: 1) DC, 2) JK, 3) T, 4) TE;

д) з імпульсно-потенціальним керуванням: 1) JKC, 2) DC, 3) TE.

7 РЕГІСТРИ

Мета роботи: дослідження типових регістрів; побудова ЦПП на основі регістрів; засвоєння основ застосування регістрів у проектах, використання шаблону оголошення регістрів (Register Declaration); визначення швидкодії за допомогою регістрового дисплея; засвоєння основ побудови послідовнісного пристрою як скінченного автомата.

Домашнє завдання

Спроектувати на основі регістра зсуву ЦПП згідно з варіантом завдання 7 (див. додаток А, варіанти завдання 7).

7.1 Стислі теоретичні відомості

Регістрами називаються ЦПП, що виконують операції запису (приймання, введення), збереження, перетворення (наприклад, шляхом зсуву) та зчитування (передавання, виведення) двійкових багатоцифрових чисел (слів). Особливістю регістрів є регулярність їх структури: кожний розряд складається з однакових тригера і додаткових елементів керування розрядом. Через це розрядність регістрів легко нарощується застосуванням програмованих ІС або з'єднанням кількох ІС жорсткої структури.

Регістри належать до однієї з найпоширеніших груп ЦПП. Вони застосовуються як оперативна пам'ять для тимчасового зберігання даних, для сполучення ЕОМ і радіоелектронних приладів із зовнішніми пристроями, зокрема, з метою індикації і введення-виведення інформації, перетворення її для передавання на далекі відстані, для виконання арифметичних операцій тощо.

Залежно від способу записування і зчитування інформації регістри поділяються на два основні типи: *паралельні* і *зсуву*. Інші різновиди регістрів (буферні, багаторежимні буферні, комбіновані, універсальні тощо) базуються на цих двох зазначених типах.

7.1.1 Паралельні регістри

7.1.1.1 Принцип побудови. Паралельними (регістрами пам'яті, схову) є регістри, в яких записування і зчитування всіх розрядів слова відбувається одночасно і розряди слова займають фіксоване положення відносно розрядів регістра.

Паралельний регістр – це набір тригерів з окремими інформаційними входами і виходами в кожному розряді та зі спільним для всіх розрядів синхровходом. Найпоширенішими є регістри з *однофазним записом* інформації на D-тригерах (рис. 7.1, а). З надходженням синхроімпульсу ($G = 1$) відбувається *записування* до регістра в паралельному коді вхідного слова D, наприклад, чотирирозрядного $Q_3 \ldots Q_0 = d_3 \ldots d_0$, а за його відсутності ($G = 0$) регістр перебуває в *режимі зберігання* записаної інформації, коли її можна *зчитувати* (також у паралельному коді) з виходів $Q_3 \ldots Q_0$.

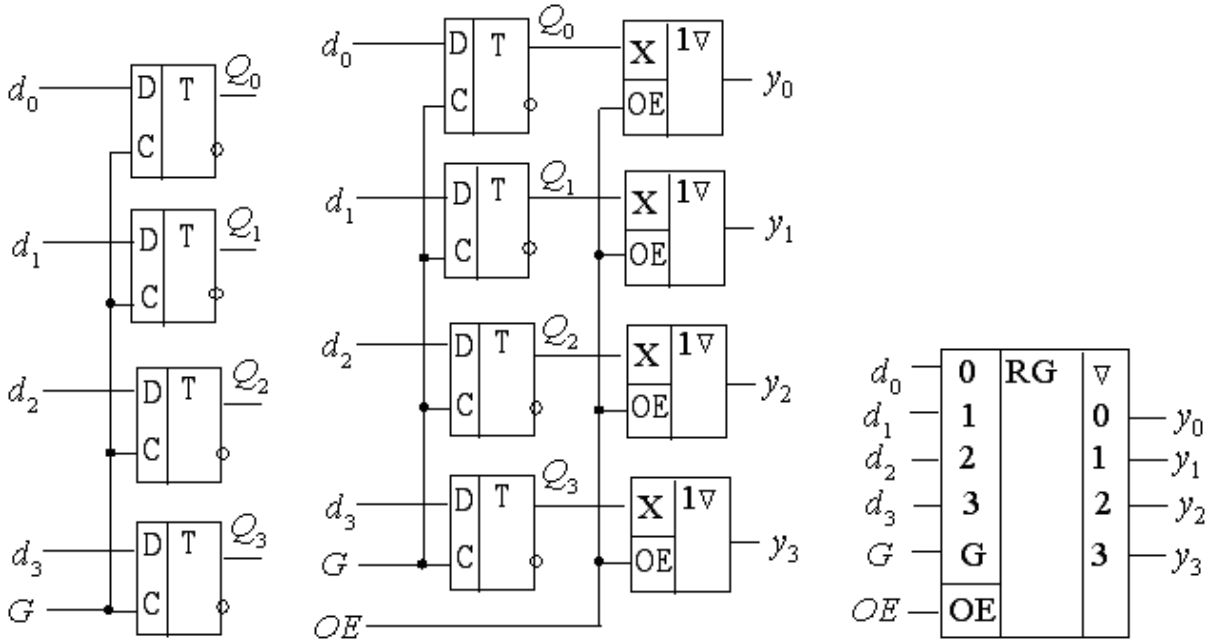


Рисунок 7.1

Регістри з *трьома станами* виходу (рис. 7.1, б) дозволяють здійснювати гнучкіший зв'язок з вихідною шиною. У такому регістрі так само активним рівнем синхроімпульсу $G = 1$ вхідне слово записується до тригерів, які за пасивного рівня $G = 0$ перебувають у режимі зберігання. Елементи з трьома станами виходу за відсутності сигналу дозволу ($OE = 0$) перебувають у високоімпедансному стані $y_i = Z$ і від'єднують регістр від шини, а під час *зчитування*, коли тригери перебувають у режимі зберігання, сигналом $OE = 1$ інформація передається в шину: $y_i = Q_i$. Такі регістри з потужними вихідними елементами називають *буферними* (рис. 7.1, в), бо забезпечують підвищену навантажівну здатність і часову буферизацію даних.

7.1.1.2 Різновиди паралельних регістрів, їх макро- і мегафункції. Основні різновиди стандартних ІС серії 74 (макрофункції з ідентифікаційними номерами 1...3) і дві мегафункції (з номерами 4, 5) наведено на рис. 7.2.

Паралельні регістри *зі статичним керуванням* (символи 1, 4), як і тригери, на яких вони виконуються, відрізняються позначенням синхровходу (зазвичай *G, gate,* інколи *C*) і складником LATCH у назві символу. У таких регістрах заборонено змінювати вхідну інформацію під час дії активного *рівня* синхроімпульсу, бо її зміна, у тому числі й випадкова, тобто завада, як в асинхронному пристрої через ворота Gate потраплятиме до виходів. У паралельних регістрів *з динамічним керуванням* (символи 2, 3, 5) синхровходи позначаються літерами CLK або у вигляді трикутника стрілки і складником FF (Flip-Flop) у назві символу. Такі регістри мають підвищену завадостійкість, бо в них заборонено змінювати вхідну інформацію лише під час дії активного *перепаду* синхроімпульсу.

Найпоширенішими є регістри з *однофазним записом* інформації на D-тригерах (символи 1, 2, 4, 5): регістр 1 відповідає схемі на рис. 1,б, а регістр 2 – схемі на рис. 7.1, а), в якій використовуються D-тригери з динамічним керуванням. В однофазних регістрах розряди вхідного слова (позначаються D_i, *data*[], а також *A, C, B, D*…, причому зазвичай вага розряду зростає від початкових літер абетки до наступних, отже, правильною є відповідність груп *data*[3..0] = *D, C, B, A*) передаються однодротовою лінією, що зручно для обміну інформацією з шинами. Проте більші функціональні можливості мають регістри з *парафазним записом* інформації на JK-тригерах (символ 3); їх можна використати як набір тригерів зі спільним синхровходом для побудови, наприклад, лічильників або регістрів зсуву.

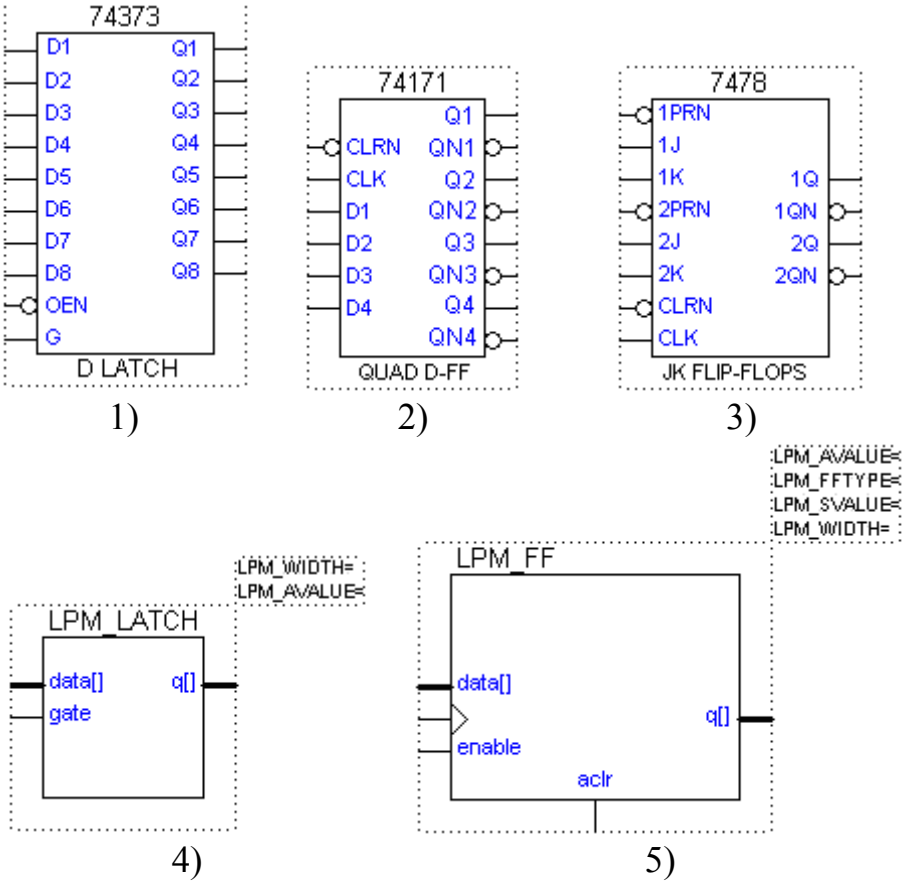
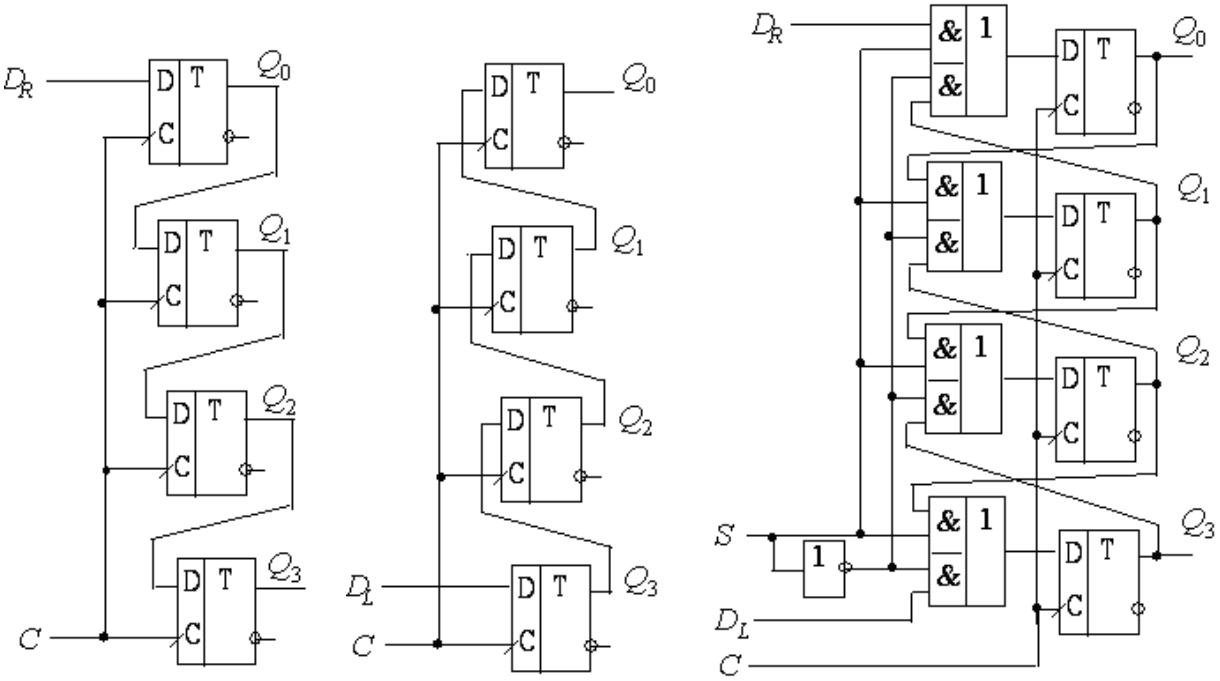


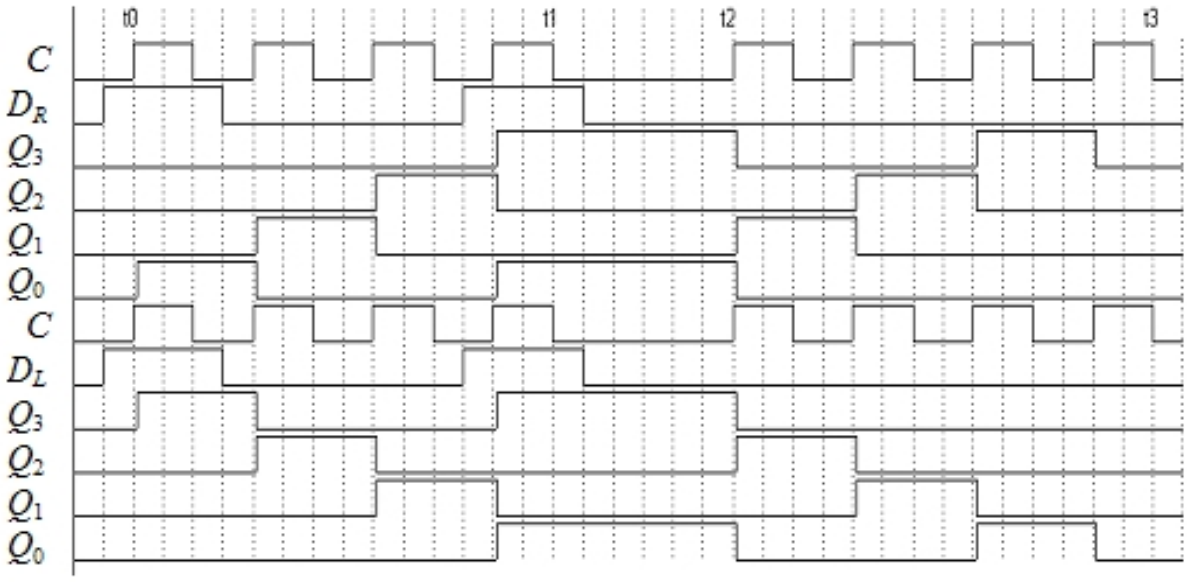
Рисунок 7.2

Регістри виконуються, здебільшого, з асинхронними *установлювальними входами* (символи 2, 3, 5). Спільним для всіх розрядів входом *CLR* (*CLRN* – інверсний вхід, *aclr* – асинхронний) регістр обнуляється, а по індивідуальному для кожного розряду входу передустановлення *PR* (*PRN* – інверсний вхід) до регістра можна асинхронно записати довільне слово. Під час функціонування регістра на установлювальних входах рівні мають бути пасивними. За допомогою *мегафункцій* (символи 4, 5) можна побудувати паралельні регістри в широкому діапазоні параметрів, основним і обов'язковим з яких є LPM_WIDTH – розрядність вхідного і вихідного слова.

7.1.2 Регістри зсуву

7.1.2.1 Принцип побудови. Регістрами зсуву (послідовними) є регістри, в яких записування і зчитування розрядів слова відбувається послідовно в часі (у послідовному коді) через один вхід та один вихід, відповідно, отже, розряди слова з кожним тактом синхроімпульсів просуваються з одного розряду регістра до іншого. Регістр зсуву є набором тригерів з послідовно з'єднаними входами і виходами та зі спільним для всіх розрядів синхровходом. Напрямок з'єднань визначає і напрямок просування розрядів слова, залежно від якого розрізняють регістри прямого зсуву, зворотного зсуву та реверсивні (рис. 7.3, а), б), в).





г)

Рисунок 7.3

У регістрі *прямого зсуву* (зсуву праворуч), наприклад, чотирирозрядного (див. рис. 7.3, а), розряди вхідного слова $D = d_3d_2d_1d_0$, починаючи від старшого d_3, надходять до входу послідовного введення D_R, що є входом молодшого розряду регістра, а зчитуються з виходу послідовного виведення Q_3, що є виходом старшого розряду регістра.

Розглянемо для наочності *запис* до цього регістра чотирирозрядного слова $D = d_3d_2d_1d_0 = 1001_2$ (на рис. 7.3, г) верхні шість часових діаграм). Припустімо, що попередньо регістр заповнено нулями: $Q_3 = Q_2 = Q_1 = Q_0 = 0$. Тоді, з надходженням в момент t_0 *позитивного перепаду* першого синхроімпульсу C до тригерів одночасно (із затримкою на час перемикання одного тригера) записується інформація, присутня на їх входах D, а саме: $Q_0 = D_R = d_3 = 1$, $Q_1 = Q_0 = 0$, $Q_2 = Q_1 = 0$, $Q_3 = Q_2 = 0$. З надходженням позитивного перепаду другого синхроімпульсу C до входу послідовного введення прикладено наступний розряд слова $D_R = d_2 = 0$, а до входів інших тригерів – інформація з виходів попередніх, отже, відбувається запис: $Q_0 = D_R = d_2 = 0$, $Q_1 = Q_0 = d_3 = 1$, $Q_2 = Q_1 = 0$, $Q_3 = Q_2 = 0$. Таким чином, з кожним синхроімпульсом біти вхідного слова просуваються в бік старших розрядів і після закінчення серії з чотирьох синхроімпульсів (момент t_1) все слово виявляється записаним до регістра: $Q_0 = d_0 = 1$, $Q_1 = d_1 = 0$, $Q_2 = d_2 = 0$, $Q_3 = d_3 = 1$. Щоб протягом одного такту біти слова не могли просунутися більше ніж на один розряд, у регістрах зсуву застосовують тригери тільки з *динамічним керуванням*.

Для *зчитування* слова з виходу Q_3 у послідовному коді необхідно подати ще одну серію з чотирьох синхроімпульсів на інтервалі (t_2, t_3): перед позитивним перепадом кожного синхроімпульсу на виході Q_3 з'являється черговий розряд. Якщо при цьому до входу послідовного введення прикладено $D_R = 0$, регістр буде *заповнено нулями*, а якщо надходитимуть біти нового слова, воно буде записано під час зчитування попереднього.

Внаслідок того, що всі тригери регістра перемикаються одночасно, *швидкодія* регістра зсуву, як і паралельного, визначається часом перемикання одного тригера.

У регістрі *зворотного зсуву* (зсуву ліворуч), наприклад, чотирирозрядного (див. рис. 7.3, б), розряди вхідного слова $D = d_3d_2d_1d_0$, починаючи від молодшого d_0, надходять до входу послідовного введення D_L, що є входом старшого розряду регістра, а зчитуються з виходу послідовного виведення Q_0, що є виходом молодшого розряду. Процеси записування і зчитування (на рис. 7.3, г) нижні шість часових діаграм) відбуваються так само, як і в регістрі прямого зсуву, але в протилежному напрямку. Проте напрямок зсуву є відносним: досить перенумерувати розряди для взаємного перетворення регістрів прямого і зворотного зсуву. Про це свідчать і часові діаграми (див. рис. 7.3, г): при зміні напрямку нумерації розрядів одного з регістрів (від старшого розряду до молодшого чи навпаки) діаграми обох регістрів збігаються. Тому й випускають регістри зсуву без поділу на два види.

Принципового значення напрямок зсуву набуває в *реверсивних* регістрах (див. рис. 7.3, в), в яких шляхом перемикання міжрозрядних зв'язків слово просувають у прямому або зворотному напрямку. Дійсно, при керувальному сигналі $S = 1$ пристрій перетворюється на регістр прямого зсуву (див. рис. 7.3, а), а при $S = 0$ – на регістр зворотного зсуву (див. рис. 7.3, б).

7.1.2.2 Різновиди регістрів зсуву, їх макрофункції і мегафункція. Основні різновиди стандартних ІС серії 74 (макрофункції з ідентифікаційними номерами 1...5), спеціалізована макрофункція програмного комплекту (символ 6) та мегафункція (символ 7) наведено на рис. 7.4.

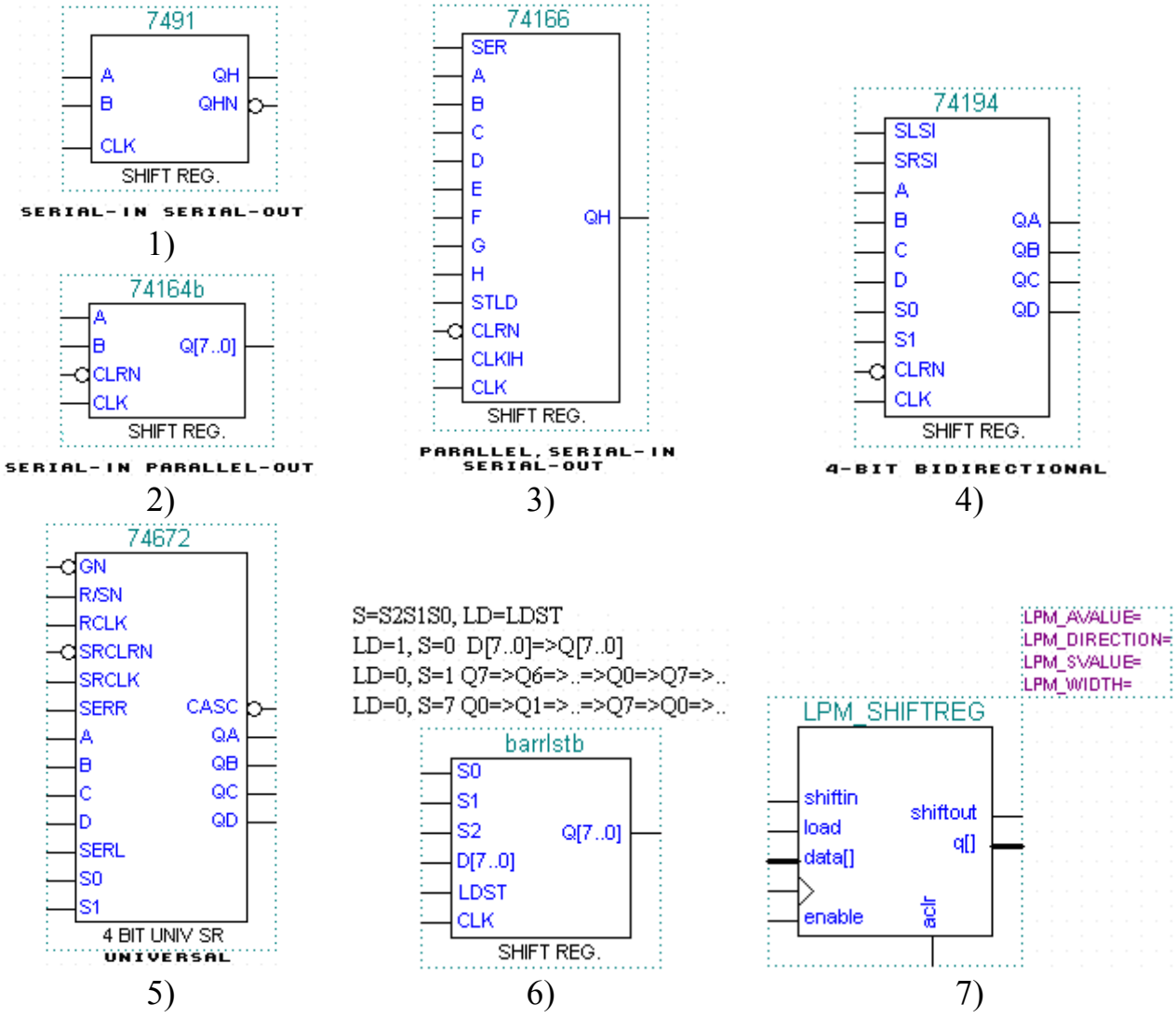


Рисунок 7.4

У регістрах зсуву обов'язковими є входи послідовного (serial) введення *SER* і синхровхід *CLK*, часто застосовують також входи передустановлення: *CLR* – скидання та *LD* (*STLD, LDST, load*) – попереднього завантаження даних у паралельному коді. Запровадженням різних входів та виходів з поєднанням властивостей паралельних і послідовних регістрів утворюють комбіновані і багатофункціональні регістри.

Класичний (з послідовним введенням і послідовним виведенням)

регістр зсуву (символ 1 зі входом *SER=A&B*) стає *послідовно-паралельним* (символ 2), якщо використовуються виходи всіх розрядних тригерів ($Q_{0...}Q_3$ на рис. 7.3, а). Після послідовного введення серією з n синхроімпульсів (n – кількість розрядів слова і регістра) слово можна зчитувати в паралельному коді (на інтервалі $t_1 \ldots t_2$ рис. 7.3, г), тобто перетворити послідовний код у паралельний. Таке перетворення застосовується для передавання інформації на значну відстань у послідовному коді однодротовою лінією зв'язку (яка є не тільки ощадливою, але й більш завадостійкою) з подальшою обробкою в паралельному коді.

Для передачі інформації в однодротову лінію виконують зворотне перетворення (паралельного коду в послідовний) за допомогою *паралельно-послідовного* регістра (символ 3). Однорозрядним керувальним сигналом *STLD* перемикаються міжрозрядні зв'язки в регістрі: при *STLD* = 1 схема є звичайним регістром зсуву зі входом послідовного введення *SER* (див. рис. 7.3, а), а при *STLD* = 1 набуває вигляду паралельного регістра з розрядними входами *A ...H* (див. рис. 7.1, а). Рівнем сигналу *CLCIH* = 0 дозволяється проходження синхроімпульсів *CLC* на входи тригерів.

Комбінований регістр (символ 4) дворозрядним керувальним кодом S_1S_0 можна перетворити у паралельний з розрядними входами *A ...D* та в *реверсивний* зі входом прямого *SRSI* або зворотного *SLSI* зсуву.

Універсальний регістр (символ 5) з дворозрядним керувальним кодом S_1S_0 виконує подібні операції, але з розширеними функціональними можливостями. По розрядних входах *A ...D* і входу *SRCLRN* здійснюється відповідно паралельне завантаження або скидання, а прямий і зворотний зсув – по входах *SERR* та *SERL*. З основного регістра зсуву із синхровходом *SRCLC* дані можна перезаписати до вихідного паралельного регістра синхроімпульсами *RCLC*. Сигналом *R/SN* виходи регістра зсуву безпосередньо або виходи паралельного регістра підмикаються до вихідних буферів з трьома станами виходу, які сигналом *GN* можна переводити до третього стану або з'єднувати з шиною. На додатковому виході *CASC* відображається біт *A* або *D* (залежно від напрямку зсуву).

Під час просування через регістр зсуву інформація потрапляє до виходу послідовного виведення і втрачається. Якщо цей вихід з'єднати зі входом послідовного введення (наприклад, *SERR* = *QD* або *SERL* = *QA* на символі 5), утворюється *кільцевий* (циклічний) регістр, в якому попередньо записане слово циркулюватиме по колу: з надходженням кожного чергового синхроімпульсу біти зсуваються на один розряд і на виходах з'являється певна їх комбінація. У спеціалізованій макрофункції (символ 6) такі з'єднання можна здійснити трирозрядним керувальним кодом $S = S_2S_1S_0$, який дозволяє налаштувати регістр на виконання різноманітних операцій. Наприклад, при $S = 7$ утворюється кільцевий регістр прямого зсуву, при $S = 1$ – зворотного зсуву, а при $S = 0$ – паралельний регістр.

За допомогою *мегафункції* (символ 7) можна побудувати всі різно-

види регістрів у широкому діапазоні параметрів, основним і обов'язковим з яких є LPM_WIDTH – розрядність вхідного і вихідного слова.

7.1.3 Генератори цифрових кодів і розподільники

Крім виконання операцій, зазначених на початку теоретичних відомостей, регістри зсуву застосовуються в цифрових системах радіотехніки та зв'язку, обчислювальної техніки, автоматики тощо для діагностування і корекції похибок цифрових пристроїв, у колах їх керування і синхронізації, мікропрограмного керування вузлами шляхом розподілення перемикальних сигналів у певну кількість каналів і т. ін.

Для розв'язання подібних задач використовуються *генератори кодової послідовності* (ГКП) та *розподільники імпульсів і рівнів* (РІР). Серед різного типу ГКП набули поширення генератори псевдовипадкових чисел і генератори зі сталими кодами, в яких сполучення нулів та одиниць у розрядах регістра залишається незмінним. На основі останнього типу ГКП будуються і РІР, які за видом сигналів поділяють на розподільники рівнів (РР), в яких активна величина потенціалу (логічна 1 або 0) діє протягом такту синхроімпульсів, та розподільники імпульсів (РІ), в яких активний сигнал триває протягом синхроімпульсу. Проте з точки зору схемотехніки розподільники є комбінованими, бо РІ утворюють з РР.

Згадані пристрої циклічної дії, що ґрунтуються на кодах зі сталим сполученням нулів і одиниць, можна побудувати, зокрема, на регістрах зсуву. Достатньо попередньо записане певне слово зсувати в кільцевому регістрі для отримання на його виходах періодично повторюваної послідовності чисел, яка і правитиме за вихідні сигнали ГКП та РІР. Просування через розряди регістра періодично повторюваної послідовності символів, наприклад, 000111000111... для стислості зручно позначити періодом у дужках: (000111). При цьому значення кодів N на виходах кільцевого регістра залежать від кількості його розрядів n. Так, просуванням вправо наведеної послідовності символів через дворозрядний регістр (n = 2) на його виходах утворюватимуться коди $N = Q_1Q_0$ = 00, 00, 01, 11, 11, 10, які далі повторюються. Стисло це можна позначити в десятковій системі кодів як N = (000111) = 0, 0, 1, 3, 3, 2. Детерміновані коди послідовності мають бути *різними*, тому дворозрядний регістр є неприйнятний для її формування, адже серед кодів є однакові. Розрядність регістра збільшують, поки не впевнюються, що серед вихідних кодів немає однакових, таку величину n і беруть за *мінімальну*. Так, просуванням цієї ж послідовності через трирозрядний регістр утворюються коди N = (000111) = $Q_2Q_1Q_0$ = 0, 1, 3, 7, 6, 4, які є різними, отже, мінімальна кількість розрядів становить n = 3. Кількість різних кодів, що періодично повторюються на виходах ЦПП, називається його *модулем*. Зі збільшенням розрядності змінюються значення кодів, але їх кількість залишається незмінною, модуль визначається лише *кількістю символів* послідовності.

7.2 Лабораторне завдання

7.2.1 Дослідити основні типи регістрів

7.2.1.1 Дослідити паралельний регістр з трьома станами виходів на основі D-тригерів зі статичним керуванням: за принциповою електричною схемою та осцилограмами сигналів (файли d:\max2work\tutorial\7lab\ 7par.gdf, .scf) визначити умови режимів записування, зберігання та зчитування інформації, виміряти затримку перемикання. У звіті навести також умовне графічне позначення за ДСТУ такого регістра зі стислим поясненням принципу його дії та осцилограм сигналів.

7.2.1.2 Дослідити регістр прямого зсуву (зсуву праворуч) на D-тригерах з динамічним керуванням (схема 1) у режимах послідовного запису, паралельного та послідовного зчитування (файли d:\max2work\tutorial\7lab\7zsuw.gdf, .scf) та розглянути особливості побудови і перемикання регістра зсуву вліво (зворотного зсуву) і реверсивного регістра (схеми 2, 3).

7.2.1.3 Ознайомитися з ***різновидами регістрів*** бібліотеки бази даних (файл d:\max2work\tutorial\7lab\7libre.gdf): макрофункціями (вибраними ІС серії 74) паралельних регістрів і регістрів зсуву та мегафункціями. Розглянути функціональні прототипи (FUNCTION), таблиці відповідності та інші довідкові дані (див. п. 1.1.1.2), пояснити призначення та особливості входів і виходів.

7.2.2 Застосувати регістр зсуву в *графічному* редакторі для побудови заданого варіанту XX генератора кодової послідовності (ГКП) або розподільника імпульсів/рівнів (РІР)

7.2.2.1 Зібрати найпростіший (несамовідновлювальний) пристрій на макрофункції регістра зсуву вибраного типу в графічному файлі d:\max2work\tutorial\7lab\7XXgkp(ri)1.gdf проекту d:\max2work\tutorial\7lab\ **7**XXgkp(ri)1, виконати компіляцію і моделювання та випробувати його на самовідновлюваність за часовими діаграмами d:\max2work\ tutorial\7lab\ 7XXgkp(ri)1.scf.

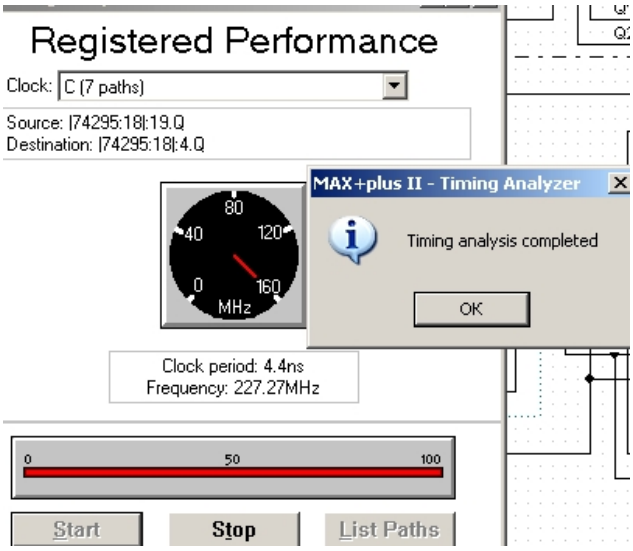
🖫 ***Приклади****:* d:\max2work\tutorial\7lab\700gkp1.gdf, scf, 700ri1.gdf, scf (схема 1).

☝ ***Примітка.*** У наведених прикладах пристрій переводиться до станів поза робочим циклом шляхом паралельного завантаження коду зі входів d_i сигналом *MODE*=1. Додатковим буфером LCELL у схемі РІР (d:\max2work\tutorial\7lab\700ri1.gdf) синхроімпульси затримуються, аби уникнути завад на краях імпульсів Q_i.

7.2.2.2 Скоригувати схему для утворення самовідновлювального пристрою, виконати п. 7.2.2.1 (можна в тому самому файлі) та скласти по-

вний перемикальний граф.

🖫 ***Приклади:*** d:\max2work\tutorial\7lab\700gkp1.gdf, scf (схема 2), d:\max2work\tutorial\7lab\700ri1.gdf, scf (схема 2); інверсні пристрої: d:\max2work\tutorial\7lab\700gkp0.gdf, scf, 700ri2.gdf, scf, 700ri3.gdf, scf.



7.2.2.3 Визначити швидкодію за допомогою *утиліти Registered Performance*: натиснути значок часового аналізатора інструментальної панелі (або меню MAX+plus II > Timing Analyzer) > меню Analysis > Registered Performance > Start у віконці часового аналізатора, де зчитати інформацію щодо швидкодії > з меню File або інструментальної панелі. Зберегти файл у формі таблиці 7XXgkp(ri)1.tao, після чого зчитати *текстовий* файл 7XXgkp(ri)1.tao > List Paths у віконці часового аналізатора > OK в інформаційному віконці і зчитати повідомлення про шляхи та величину затримок > зберегти файл 7XXgkp(ri)1.mtf > відтак зчитати *текстовий* файл 7XXgkp(ri)1.mtf.

🖫 ***Приклад:*** d:\max2work\tutorial\7lab\700gkp1.tao, .mtf.

7.2.2.4 Скласти та дослідити самовідновлювальний пристрій на мегафункції регістра зсуву (можна в тому самому графічному файлі).

🖫 ***Приклад:*** d:\max2work\tutorial\7lab\700gkp1.gdf, scf (схема 3).

7.2.2.5 Скласти та дослідити самовідновлювальний пристрій на автоматично створеному за допомогою *менеджера MegaWizard Plug-In Manager* ***різновиді мегафункції регістра зсуву*** 7XXwiz_reg.sym (можна в тому самому графічному файлі).

🖫 ***Приклад:*** d:\max2work\tutorial\7lab\700wiz_reg.sym, 700gkp1.gdf, scf (схема 3).

7.2.3 Застосувати регістр зсуву в *текстовому* редакторі виконанням завдання за п. 7.2.2.

7.2.3.1 Створити та дослідити пристрій на основі макрофункції регістра вибраного типу в тестовому файлі (.tdf) проекту 7XXreg_decl на кшталт самовідновлювальної схеми на макрофункції з графічного файла, для чого:

а) після заголовку TITLE вставити шаблон Function Prototype

Statement (non-parameterized), в який ввести параметри прототипу регістра безпосередньо з довідки Help (можна скопіювати і вставити копію):

FUNCTION **74295** (oe, ld/shn, clk, ser, d[3..0])

RETURNS (q[3..0]);

б) після оголошення портів у секції підпроекту вставити в секцію змінних (Variable Section) шаблон оголошення регістрів (***Register Declaration***), де надати ім'я зразку регістра:

```
VARIABLE                    % Variable Section      %
    rg     : 74295;         % Register Declaration  %
```

в) у підсекції булевих рівнянь логічної секції виконати потрібні з'єднання (докладніше див. п. 6.2.3.1).

🖫 ***Приклад:*** d:\max2work\tutorial\7lab\700reg_decl.tdf**,** .scf (на кшталт схеми 2 з файла 700gkp1.gdf).

7.2.3.2 Виконати п. 7.2.3.1 на ***мегафункції*** регістра (див. п. 6.2.3.2) у тестовому файлі (.tdf) проекту **7**XXmega_gkp(ri) на кшталт схеми на мегафункції з графічного файла.

🖫 ***Приклад:*** d:\max2work\tutorial\7lab\700mega_gkp.tdf, .scf (на кшталт схеми 3 файла 700gkp1.gdf).

7.2.3.3 Виконати п. 7.2.3.1 ***на автоматично створеному*** за допомогою *менеджера MegaWizard Plug-In Manager* ***різновиді мегафункції регістра*** (див. п. 6.2.3.3) у тестовому файлі (.tdf) проекту **7**XXwiz_gkp(ri) на кшталт аналогічної схеми з графічного файла.

🖫 ***Приклад:*** d:\max2work\tutorial\7lab\700wiz_reg.tdf, 700wiz_gkp.tdf, .scf (на кшталт схеми 4 файла 700gkp1.gdf).

7.2.4 Засвоїти основи побудови послідовнісного пристрою як *скінченного автомата* (State Machine) виконанням завдання за п. 7.2.2.

7.2.4.1 За перемикальним графом 7XXgkp(ri)1.gdf ***створити пристрій з використанням оператора вибору*** Case Statement у тестовому файлі (.tdf) проекту **7**XXsmcase_gkp(ri), для чого після оголошення портів:

а) до секції змінних Variable Section вставити шаблон підсекції оголошення скінченного автомата State Machine Declaration, в який ввести символьну назву скінченного автомата, наприклад, gkp або ri, після якої залишити двокрапку і ключове слово Machine; відтак після ключових слів OF BITS у круглих дужках ввести через кому або у вигляді масиву символьну назву розрядів автомата, наприклад, (q2, q1, q0) або (q[2..0]); нарешті, після ключових слів WITH STATES у круглих дужках ввести через кому символьні назви станів автомата і через знак рівності їх значення в одній із систем числення, наприклад, (s0 = 0, s1 = H"1", s2 = B"011", …); у кінці за-

лишити крапку з комою:

```
VARIABLE
    gkp   : MACHINE                 %  State Machine Declaration  %
    OF BITS (q[2..0])
    WITH STATES    (s0 = 0, s1 = 1, s2 = 3, s3 = 7, s4 = 6,
                    s5 = 4, il1 = 2, il2 = 5);
```

✍ ***Примітки:***

1. У підсекції State Machine Declaration речення OF BITS залежно від розв'язуваного завдання є необов'язковим, тобто компілятор сам може призначити кількість розрядів *n* пристрою за критерієм мінімальної схеми (інколи схема є простішою за більшої величини *n* через спрощення міжрозрядних зв'язків).

2. У реченні WITH STATES можна подати список символічних назв станів без зазначення їх кодів, якщо важливою є лише кількість станів, наприклад, (s0, s1, s2, ...).

3. З усіх 2^n можливих станів дозволеними є ті, що подані в списку символічними назвами. Для усунення недозволених станів їх потрібно також перелічити в списку, наприклад, il1, il2 (від illegal).

```
BEGIN
        gkp.clk = C;
        Q[] = gkp.q[];
END;
```

б) до логічної секції Logic Section вставити шаблон булевих рівнянь Boolean equations, до якого ввести *булеві рівняння керування*, що пов'язують порти (входи та виходи) автомата із зовнішніми портами пристрою.

✍ ***Примітка.*** Скінченний автомат State Machine має лише три вхідні порти: CLK (синхровхід з динамічним керуванням), RESET (скидання) і ENA (дозвіл на проходження синхроімпульсів, тобто на перемикання) та вихідні порти, оголошені перед ключовим словом OUTPUT секції SUBDESIGN. *Булеві рівняння керування* Boolean equations логічної секції складаються згідно із синтаксисом: <symbolic name>.clk, <symbolic name>.reset, <symbolic name>.ena, <symbolic name>.q_i, де <symbolic name> - назва, оголошена в підсекції State Machine Declaration. У прикладі рівняння

gkp.clk = C приєднує зовнішній порт „C” до синхровходу автомата, а рівняння Q[] = gkp.q[] приєднує його виходи до зовнішніх портів „Q1”, „Q2”, „Q3”. Отже, інші зовнішні сигнали не можна вважати портами автомата, їх треба пов’язувати з переходами автомата іншими логічними рівняннями або гілками умовних операторів чи таблиці, як, наприклад, у проекті d:\max2work\tutorial\7lab\700smtabl_gkp.

в) після рівнянь у логічну секцію вставити шаблон умовного оператора Case Statement (див. п. 3.2.4.3), до якого ввести переходи між станами автомата, так само, як у перемикальному графі:

```
BEGIN
    gkp.clk = C;
    Q[] = gkp.q[];
    CASE gkp IS                          %  Case Statement        %
        WHEN s0 => gkp = s1;             %  0 => 1                 %
        WHEN s1 => gkp = s2;             %  1 => 3                 %
        WHEN s2 => gkp = s3;             %  3 => 7                 %
        WHEN s3 => gkp = s4;             %  7 => 6                 %
        WHEN s4 => gkp = s5;             %  6 => 4                 %
        WHEN OTHERS => gkp = s0;  %  4=>0,  2=>0,  5=>0   %
    END CASE;
END;
```

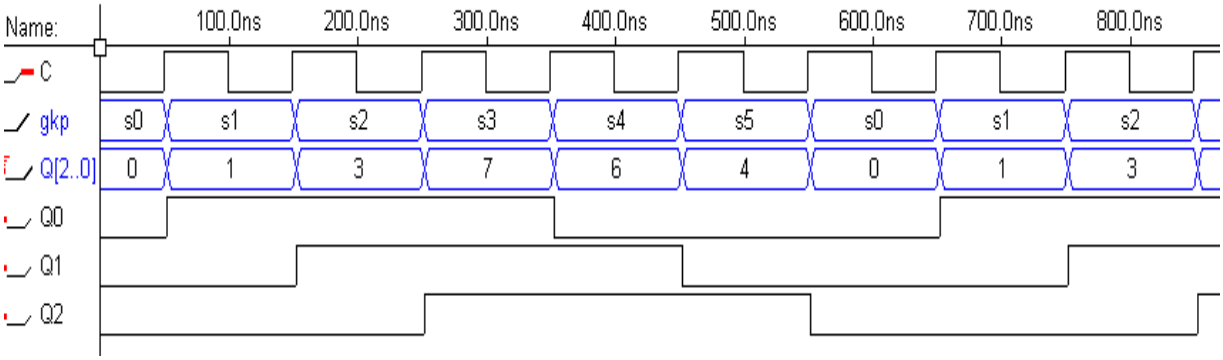
г) після компіляції і функціонального моделювання перевірити результати проектування за часовими діаграмами 7XXsmcase_gkp(ri).scf.

🖫 ***Приклад:*** d:\max2work\tutorial\7lab\700smcase_gkp.tdf, scf; перемикальний граф 5, 700gkp1.gdf.

✍ ***Примітка.*** Замість оператора вибору Case також можна використовувати умовний оператор IF THEN.

7.2.4.2 За перемикальним графом 7XXgkp(ri)1.gdf ***створити пристрій з використанням таблиці відповідності*** в тестовому файлі (.tdf) проекту **7**XXsmtabl_gkp(ri), для чого виконати п. 7.2.4.1, де у п. 7.2.4.1, в) до логічної секції вставити шаблон таблиці відповідності Truth Table Statement (див. п. 3.2.4.1), до якого ввести переходи між станами автомата,

так само, як у перемикальному графі.




```
TITLE "ГКП - State Machine - перемикальна таблиця";
        SUBDESIGN 700smtabl_gkp
        (
            C, R, LD      : INPUT;
            Q[2..0]       : OUTPUT;
        )
        VARIABLE                              % Variable Section                %
            gkp    : MACHINE                  % State Machine Declaration   %
            OF BITS (q[2..0])
            WITH STATES (
                    s0 = 0, s1 = 1, s2 = 3, s3 = 7, s4 = 6, s5 = 4,
                    il1 = 2, il2 = 5);
        BEGIN                                 %  Logic Section                 %
            gkp.clk = C; gkp.reset = R;       %  Boolean equations             %
            Q[] = gkp.q[];
          TABLE                               %  Truth Table Statement        %
            LD, gkp         => gkp;
            ------------------------
            0,  s0    => s1;   1,  s0    => il1;   0,  s1    => s2;   1,  s1    => s1;
            0,  s2    => s3;   1,  s2    => s2;   0,  s3    => s4;    1,  s3    => s3;
            0,  s4    => s5;   1,  s4    => s4;   0,  s5    => s0;    1,  s5    => s5;
            0,  il1   => il2;   1,  il1   => il1;   0,  il2   => s2;    1,  il2   => il2;
          END TABLE;
```

END;

🖫 ***Приклад:*** d:\max2work\tutorial\7lab\700smtabl_gkp.tdf, scf; перемикальний граф 6, 700gkp1.gdf.

Контрольні питання та завдання

1. Тригери якого типу керування можна застосовувати в паралельних регістрах і регістрах зсуву?

2. Побудуйте на тригерах: а) JK-, б) RS-, в) D-типу такий трирозрядний регістр: 1) паралельний, 2) зсуву (послідовно-послідовний), 3) паралельно-послідовний, 4) послідовно-паралельний, 5) реверсивний, 6) кільцевий, 7) універсальний, 8) буферний з трьома станами виходу.

3. Користуючись умовними графічними позначеннями, складіть з ІС 4-розрядних регістрів схему 12-розрядного регістра: а) паралельного, б) зсуву.

4 Перетворіть трирозрядний паралельний регістр у регістр зсуву.

8 ЛІЧИЛЬНИКИ

Мета роботи: дослідження типових лічильників; застосування лічильників у проектах; засвоєння основ створення проекту за *сигнального* його введення (часовими діаграмами).

Домашнє завдання

Спроектувати згідно з варіантом (див. додаток А, варіанти завдання 8): а) недвійковий лічильник шляхом перетворення двійкового лічильника, б) недвійковий лічильник на тригерах зі зворотними зв'язками, *в*) безвентильний подільник частоти на JK-тригерах.

8.1 Стислі теоретичні відомості

8.1.1 Вступ

Лічильником називається ЦПП, що перетворює послідовність вхідних імпульсів у цифровий код, значення якого залежить від кількості цих імпульсів. Сформований код може зберігатися в лічильнику, як і в будь-якому послідовнісному пристрої, зокрема, у регістрі. Лічильники застосовуються для кількісного визначення часових інтервалів, до яких попередньо перетворюють вимірювану величину в різного типу вимірювачах, для поділу частоти в синтезаторах частот і годинниках, для формування команд галуження програм під час цифрової обробки інформації тощо.

З надходженням вхідних імпульсів лічильник переходить з одного стійкого стану до іншого і код на його виходах змінюється циклічно. З огляду на це кількість різних стійких станів лічильника характеризується *модулем лічби M.* Цей параметр відображає, яку максимальну кількість імпульсів здатний однозначно підрахувати певний лічильник. Якщо, наприклад, вихідний код змінюється в межах $N = 0, 1, \ldots, 9, 0, 1, \ldots, 9, \ldots$, модуль лічби становить $M = 10$.

Швидкодія лічильника характеризується *часом усталення* $t_л$ – найбільшим часовим інтервалом, протягом якого після надходження кожного з вхідних імпульсів встигають встановитися всі розряди вихідного коду, а також *максимальною частотою* лічби $f_л$ – частотою вхідних імпульсів, за якої лічильник здатний перемкнутися від одного стійкого стану до іншого.

За модулем лічби лічильники поділяють на *двійкові*, якщо модуль становить $M = 2^n$ (де n – ціле число), та *з довільним модулем* лічби (недвійкові), в яких модуль $M \neq 2^n$. За напрямком лічби розрізняють такі лічильники: *підсумовувальні*, код яких з надходженням чергового імпульсу збільшується на одиницю (інкрементується), *віднімальні*, коли код зменшується на одиницю (декрементується), та *реверсивні*, в яких напрямок лічби є керованим. Крім того, залежно від порядку зміни коду можна виокреми-

ти лічильники з *природним* порядком лічби, якщо код з надходженням чергового імпульсу збільшується або зменшується на одиницю, та зі *штучним* порядком лічби, коли код змінюється в довільному порядку. Щодо організації міжрозрядних зв'язків будують лічильники, в основному, за схемами з *послідовним*, *паралельним* і *груповим* перенесенням.

8.1.2 Двійкові лічильники

8.1.2.1 Лічильники з послідовним перенесенням. Модуль лічби двійкового лічильника $M = 2^n$ визначається кількістю його розрядів n, а вихідний код змінюється в межах 0 ... M – 1. Наприклад, при n = 3 модуль становить M = 8 і вихідний код $N = Q_2Q_1Q_0$ у випадку *підсумовувального* лічильника (табл. 1) з надходженням вхідних імпульсів зростає від 0 до 7 і далі цикл лічби повторюється. Позначаючи для наочності новий стан після перемикання кодом $Q_2^+Q_1^+Q_0^+$ і напрямок перепадів вихідних сигналів dQ_i від логічного нуля до одиниці та навпаки знаками плюс і мінус відповідно, переконуємось, що молодший розряд перемикається до протилежного стану кожним лічильним імпульсом, а наступні розряди – *негативним* перепадом напруги з виходу попереднього розряду.

Таблиця 8.1

N	$Q_2Q_1Q_0$	$Q_2^+Q_1^+Q_0^+$	$dQ_2\ dQ_1\ dQ_0$
0	0 0 0	0 0 1	+
1	0 0 1	0 1 0	+ -
2	0 1 0	0 1 1	+
3	0 1 1	1 0 0	+ - -
4	1 0 0	1 0 1	+
5	1 0 1	1 1 0	+ -
6	1 1 0	1 1 1	+
7	1 1 1	0 0 0	- - -

Отже, лічильник можна побудувати послідовним увімкненням ланцюжка Т-тригерів. З огляду на те, що негативному перепаду напруги на прямому виході тригера відповідає позитивний перепад на його інверсному виході, при використанні Т-тригерів з прямим динамічним керуванням необхідно з'єднувати їх синхровходи з інверсними виходами попередніх розрядів $C_i = \overline{Q_{i-1}}$ (рис. 8.1, а). І, навпаки, синхровходи Т-тригерів з інверсним динамічним керуванням слід з'єднувати з прямими виходами попередніх розрядів $C_i = Q_{i\text{-}1}$.

У таких лічильниках з послідовним перенесенням (послідовних лічильниках) кожний наступний тригер перемикається із запізненням відносно фронту вихідного сигналу попереднього розряду, а тригер молодшого розряду – відносно лічильних імпульсів C на час перемикання тригера t_T. Через це час усталення вихідного коду лічильника, який визначається за перемикання всіх розрядів, залежить від їх кількості і становить $t_л = nt_T$ (на рис. 1,б кількість затримок t_T позначено цифрами). З огляду на відсутність спільного для всіх розрядів синхровходу і внаслідок цього неодночасність їх перемикання послідовні лічильники є *асинхронними.*

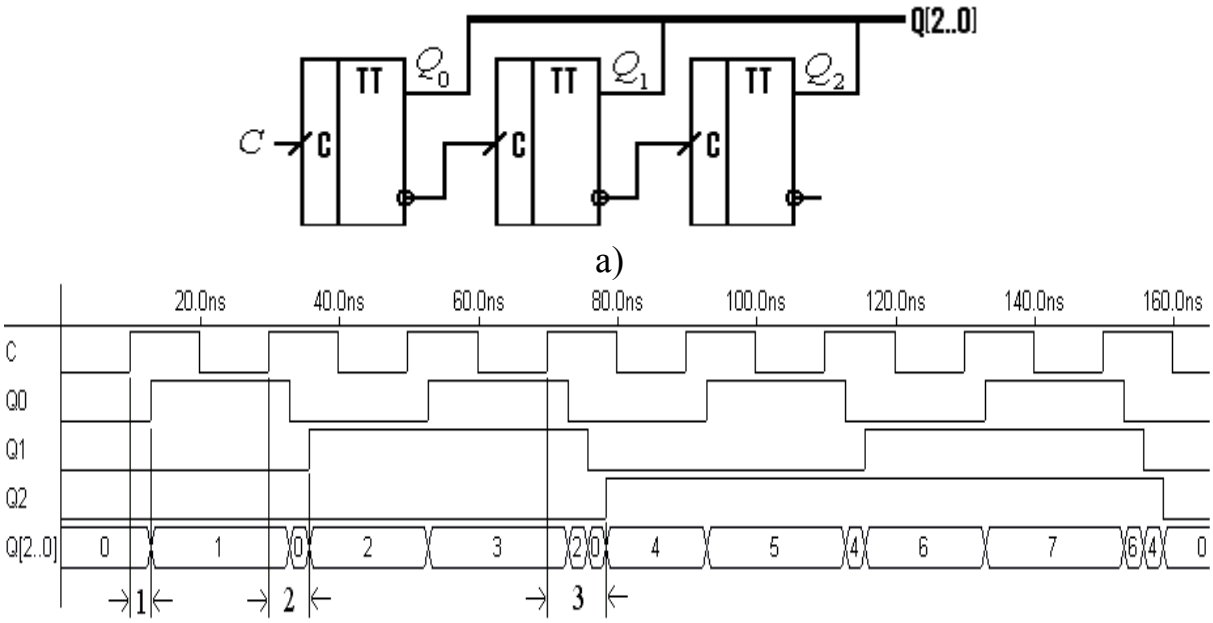


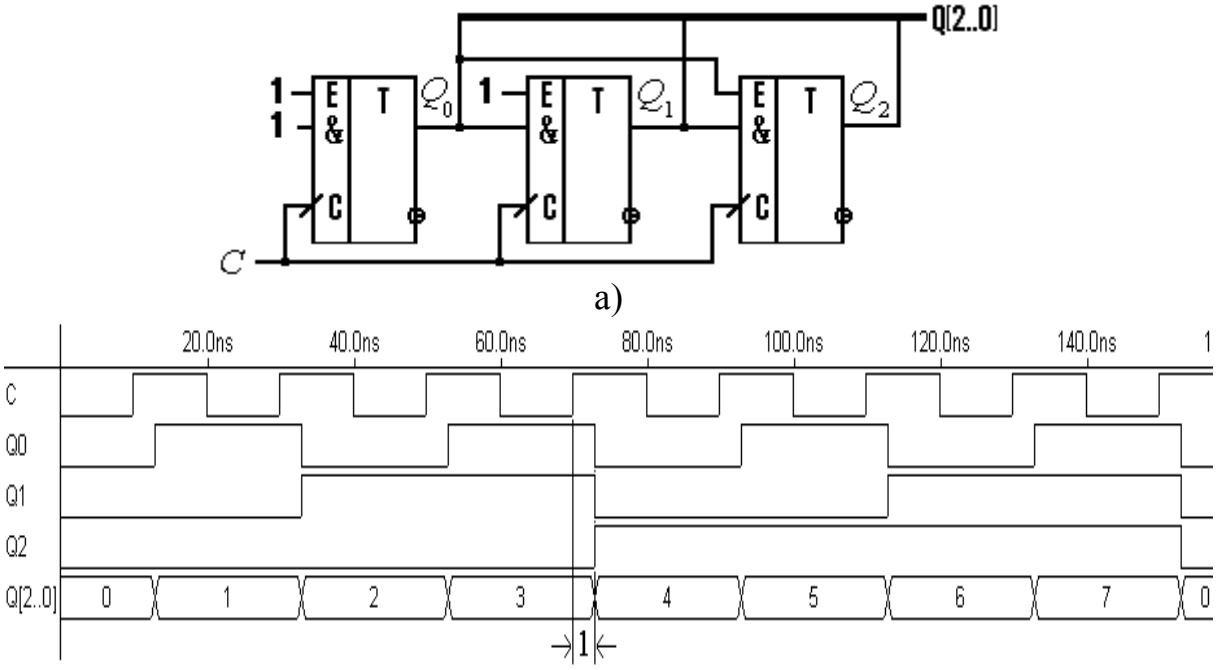
Рисунок 8.1

Перевагою лічильників з послідовним перенесенням є їхня простота через регулярність структури і мінімум міжз'єднань, що сприяє також підвищенню завадостійкості, а недоліком – низька швидкодія та залежність її від розрядності *n:* зі збільшенням модуля лічби *M* зростає і час усталення $t_л$. Крім того, під час перемикання виникають хибні стани, наприклад, перехід вихідного коду від 3 до 4 супроводжується проміжними станами 2 і 0 (див. рис. 8.1, б), що може бути небезпечним для роботи пристроїв, сполучених з виходами лічильника. Отже, вихідний код зчитують після закінчення перехідного процесу його усталення, тобто через час $t_л$.

Проте у вимірювальній техніці, коли з метою підвищення вірогідності результатів підраховують пачки імпульсів і зчитування коду виконують після закінчення низки таких пачок, ці недоліки не є суттєвими. Не впливають вони також на поділ частоти: як видно з діаграм на рис. 8.1, б), частота сигналу на виході кожного наступного розряду зменшується вдвічі. Для подібних застосувань максимальна частота вхідного сигналу обмежується лише часом перемикання тригера молодшого розряду $f_{макс} = 1/t_Т$ (бо кожний наступний розряд перемикається з удвічі меншою частотою, ніж попередній), тобто в цьому разі послідовні лічильники мають найвищу швидкодію і потребують мінімум ресурсу мікросхем.

8.1.2.2 Лічильники з паралельним перенесенням. Як видно з перемикальної таблиці двійкового лічильника (див. табл. 8.1), кожний наступний його розряд має перемикатися до протилежного стану з надходженням чергового лічильного імпульсу лише за умови перебування всіх попередніх розрядів у стані логічної 1. Так, розряд з виходом Q_1 перемикається зі станів N = 1, 3, 5, 7, коли Q_0 = 1, а з виходом Q_2 – зі станів N = 3, 7, коли

$Q_0 = Q_1 = 1$. Отже, синхровходи тригерів всіх розряди можна з'єднати паралельно з лічильним входом, якщо дозвіл на перемикання i-го розряду здобути шляхом логічного множення сигналів з виходів усіх попередніх розрядів $E_i = Q_0Q_1 \ldots Q_{i-1}$ і побудувати лічильник на ТЕ-тригерах (рис. 8.2, а).



а)

б)

Рисунок 8.2

У таких лічильниках з паралельним перенесенням (паралельних лічильниках) усі розряди, на входах дозволу яких діють рівні логічної 1, перемикаються одночасно (рис. 8.2, б), тому час усталення вихідного коду визначається затримкою перемикання лише одного тригера і становить $t_л = t_Т$. З огляду на те, що синхровхід для всіх розрядів є спільним і внаслідок цього вони перемикаються одночасно, паралельні лічильники є *синхронними.*

ТЕ-тригери з кількома входами дозволу E утворюються на основі JK-тригерів шляхом об'єднання паралельних входів J та K, а за браком таких паралельних входів – за допомогою додаткових елементів І. Проте базовою є схема паралельного лічильника на економічних D-тригерах, яку можна синтезувати з урахуванням функції збудження D-тригера $D_i = Q_i^+$. Безпосередньо з перемикальної таблиці (див. табл. 8.1) вносимо до діаграм термів значення Q_i^+ (рис. 8.3, а) і мінімізуємо функції збудження лічильника:

$$D_0 = \overline{Q_0};\ D_1 = Q_0\overline{Q_1} + \overline{Q_0}Q_1 = Q_0 \oplus Q_1;$$

$$D_2 = Q_0Q_1\overline{Q_2} + \overline{Q_0Q_1}Q_2 = Q_0Q_1 \oplus Q_2. \qquad (8.1)$$

Узагальнюючи (8.1), дістанемо функцію збудження довільного розряду:

$$D_i = Q_0 Q_1 \ldots Q_{i-1} \oplus Q_i, \qquad (8.2)$$

на підставі якої можна побудувати схему багаторозрядного лічильника, зокрема, трирозрядного (рис. 8.3, б). Якщо лічильні імпульси подавати на входи тригерів з прямим динамічним керуванням через інвертор, перемикання відбуватиметься за негативними перепадами цих імпульсів (рис. 8.3, в), як за використання тригерів з інверсним динамічним керуванням.

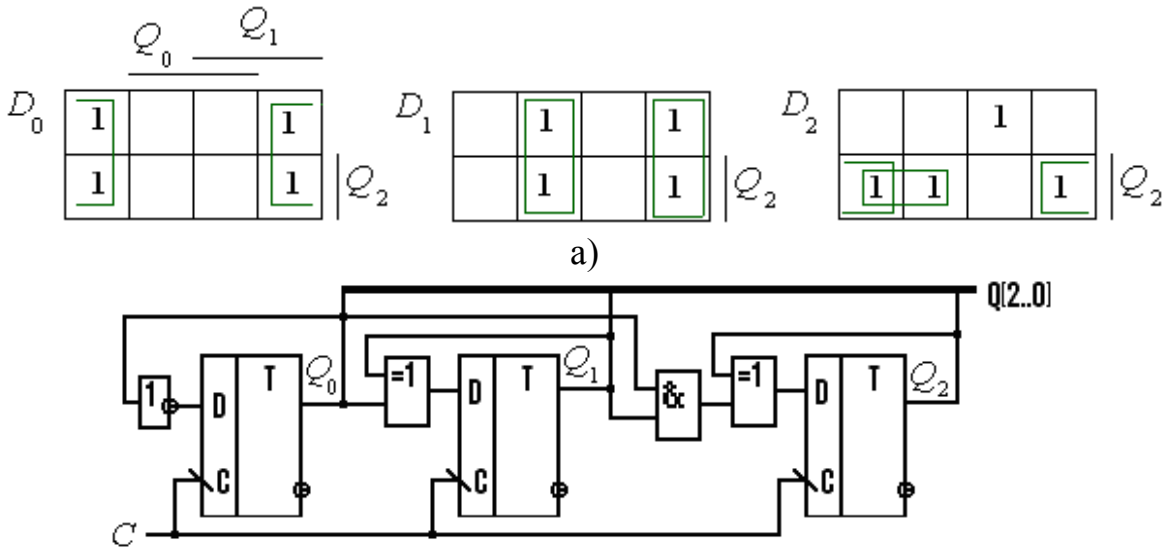


Рисунок 8.3

Таким чином, перевагою лічильників з паралельним перенесенням є їхня максимально можлива швидкодія, а недоліком – нерегулярність структури і ускладнення міжз'єднань зі збільшенням розрядності *n*, бо на вході кожного наступного розряду потрібно утворювати кон'юнкцію сигналів з виходів усіх попередніх розрядів. Крім того, зростає навантаження на молодші розряди та через одночасність перемикання в окремі моменти всіх тригерів виникають значні сплески струму в колах живлення. З огляду на те, що такі струмові імпульси можуть спричинити збої в роботі ЦП, є об-

меження на максимальну розрядність паралельних лічильників у деяких ПЛІС, а в ІС жорсткої структури таке обмеження пов'язане також з конструктивними міркуваннями.

8.1.2.3 Лічильники з груповим перенесенням. Аби подолати протиріччя між швидкодією і складністю, лічильники розбивають на групи з певним видом переносу всередині них, а між групами окремо застосовують *груповий перенос* того чи іншого типу. Найбільшого поширення набули *паралельно-послідовні* лічильники – з паралельним перенесенням у групах і послідовним перенесенням між ними (рис. 8.4), які в цілому є *асинхронними*.

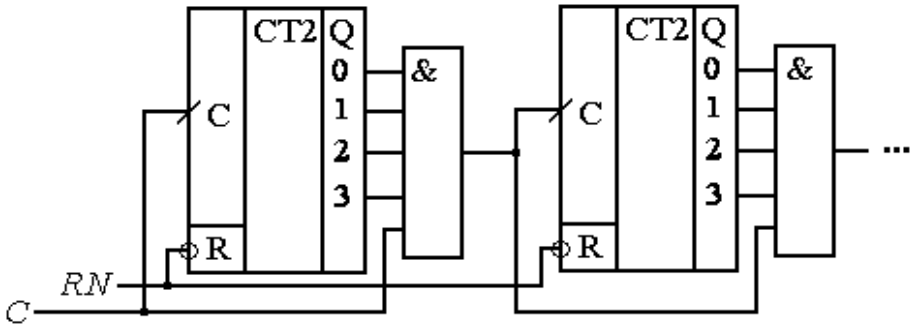


Рисунок 8.4

Коли всі (наприклад, чотири) розряди групи після чергового лічильного імпульсу переходять до одиничного стану, формується груповий перенос: елемент збігу дозволяє проходження наступному імпульсу до входу старшої групи розрядів. З урахуванням середнього часу затримки поширення $t_{з.п}$ в елементі І та часу усталення групи з паралельним перенесенням $t_{гр} = t_Т$ внаслідок послідовного поширення групових переносів загальний час усталення коду на виході лічильника з K груп становитиме $t_Л=(K–1)\ t_{з.п} + Kt_Т$. Отже, за помірного погіршення швидкодії істотно спрощуються міжрозрядні зв'язки та зменшується потрібний ресурс мікросхем.

Є також інші різновиди переносів у лічильниках. Наприклад, уведенням у міжгрупових зв'язках на рис. 8.4 додаткових елементів І утворюється *синхронний паралельно-паралельний лічильник*. Якщо в міжрозрядних зв'язках схеми на рис. 8.2, а), 8.3, б) з метою спрощення замість багатовходових запровадити ланцюжок двовходових елементів І, отримаємо *синхронний лічильник з послідовним перенесенням*. Запровадження з метою поліпшення швидкодії такого ланцюжка в міжрозрядних зв'язках схеми на рис. 8.1, а) утворює *асинхронний лічильник з наскрізним перенесенням*.

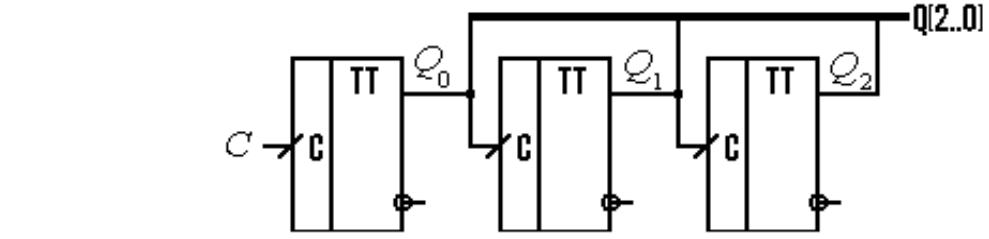
8.1.2.4 Реверсивні лічильники. За напрямком зміни коду розглянуті лічильники є підсумовувальними. *Віднімальний* лічильник (табл. 8.2) відрізняється від підсумовувального (див. п. 8.1.2.1) тим, що наступний розряд перемикається за *позитивного* перепаду сигналу на виході попереднього розряду. Отже, у міжрозрядних зв'язках віднімального лічильника слід використовувати виходи тригерів, протилежні відносно підсумовувального: входи тригерів з прямим динамічним керуванням з'єднують з прямими виходa-

ми попередніх розрядів, а тригерів з інверсним динамічним керуванням – з інверсними виходами попередніх розрядів. Для прикладу на рис. 8.5 наведено схему і часові діаграми для послідовного віднімального лічильника.

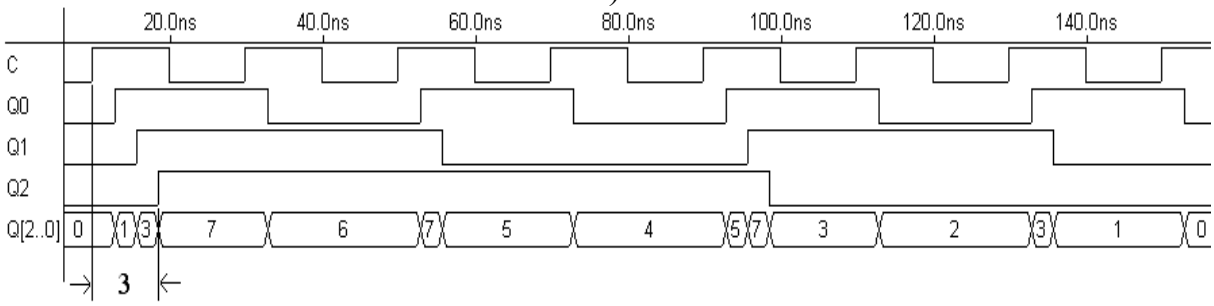
Таблиця 8.2

N	Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+	dQ_2	dQ_1	dQ_0
7	1	1	1	1	1	0			–
6	1	1	0	1	0	1		–	+
5	1	0	1	1	0	0			–
4	1	0	0	0	1	1	–	+	+
3	0	1	1	0	1	0			–
2	0	1	0	0	0	1		–	+
1	0	0	1	0	0	0			–
0	0	0	0	1	1	1	+	+	+

Підсумовувальний і віднімальний лічильники є дуальними, бо перетворюються один в одного інвертуванням їх виходів (див. табл. 8.1, 8.2). Через це випускають один тип ІС односпрямованих лічильників, які за структурою є лише підсумовувальними, якщо вихідний код знімати з прямих виходів, а для отримання віднімального лічильника досить знімати вихідний код з інверсних виходів. Два зазначені різновиди структури набувають сенсу в *реверсивних* лічильниках, в яких напрямок лічби є керованим: на окремих інтервалах часу, наприклад, у системах стеження, лічильник має працювати в режимі підсумовування, а на інших інтервалах – у режимі віднімання.



а)



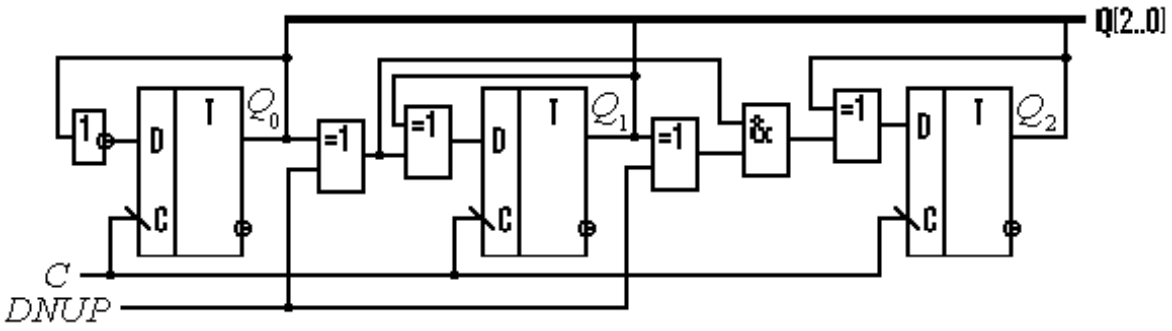
б)

Рисунок 8.5

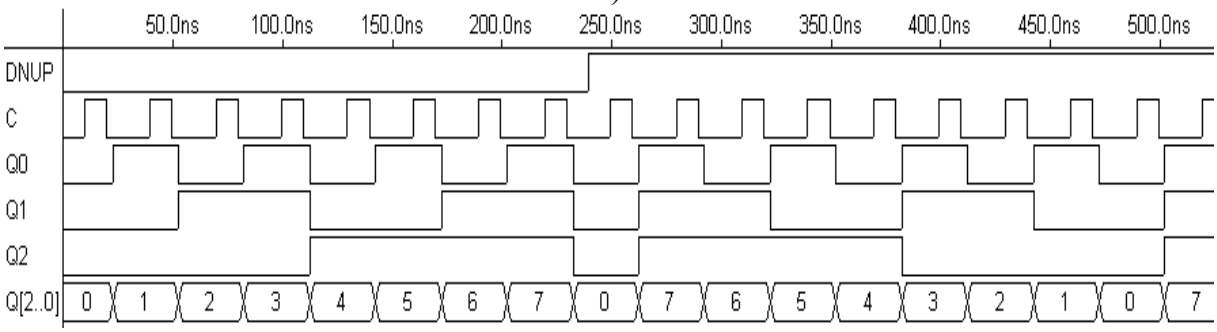
У реверсивних лічильниках напрямок лічби змінюють шляхом перемикання міжрозрядних зв’язків: за допомогою додаткових логічних елементів вхід наступного розряду з’єднують з прямими або інверсними виходами попередніх розрядів. У базовій схемі паралельного лічильника на D-тригерах (див. рис. 8.3, б) таке перемикання здійснюють елементами виключне АБО на виходах Q_i (рис. 8.6, а) під дією керувального сигналу *DNUP* (Up – прямо, Down – зворотно). Рівнем *DNUP* = 0 в міжрозрядних

зв'язках вмикаються прямі виходи $\left(Q_i \oplus 0 = Q_i\right)$ і лічильник переводиться в режим підсумовування (прямо), а рівнем *DNUP* = 1 вмикаються інверсні виходи $\left(Q_i \oplus 1 = \overline{Q}_i\right)$, тому лічильник переводиться в режим віднімання (зворотно). Отже, в режимі підсумовування для розрядів дійсними є функції збудження (8.2), а в режимі віднімання вони перетворюються на функції

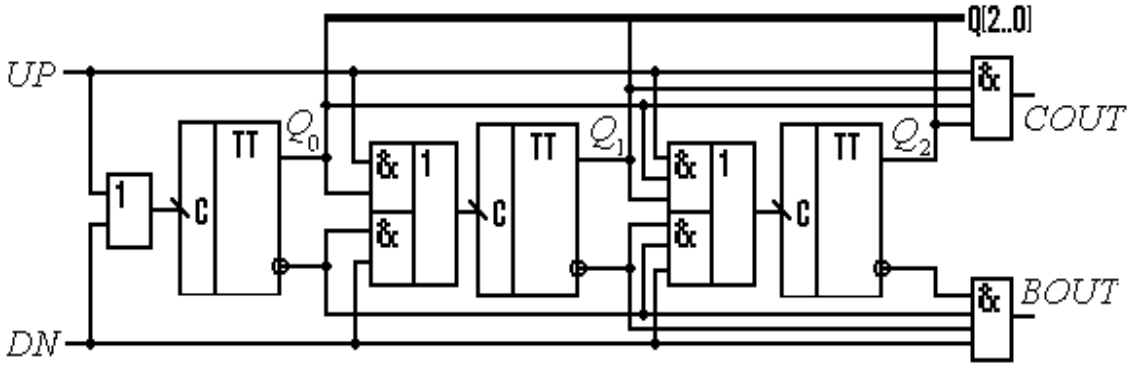
$$D_i = \overline{Q_0}\,\overline{Q_1}\ldots\overline{Q_{i-1}} \oplus Q_i\,. \tag{8.3}$$



а)



б)



в)

Рисунок 8.6

Функціонування реверсивного лічильника в двох режимах залежно від сигналу *DNUP* ілюструється часовими діаграмами на рис. 8.6, б).

У наведеній базовій схемі на *D*-тригерах в обох режимах лічильні імпульси *C* подаються в одну точку. В іншому варіанті реверсивного лічильника на Т-тригерах (рис. 8.6, в) керування напрямком лічби здійснюється шляхом зміни точки подавання лічильних імпульсів. При цьому на невикористо-

вуваному вході має діяти пасивний рівень (у прикладі логічний 0). У режимі підсумовування імпульси подаються на вхід *UP* і вмикаються міжрозрядні зв'язки з прямих виходів тригерів, а рівнем *DN* = 0 вимикаються зв'язки з інверсних виходів. У режимі віднімання, навпаки, лічильні імпульси надходять на вхід *DN*, вмикаються зв'язки з інверсних виходів, а рівнем *UP* = 0 вимикаються зв'язки з прямих виходів. Вихідні сигнали переносу *COUT* (Carry output) і позики *BOUT* (Borrow output) призначені для каскадування лічильників.

8.1.2.5 Макрофункції. Лічильники належать до найпоширенішої групи ЦПП: засвоєно випуск чотирьох десятків їх типономіналів у складі серії 74, що є стандартними макрофункціями, крім того, для програмування ВІСПС застосовуються кілька спеціалізованих макрофункцій та універсальна мегафункція лічильника. Типові різновиди макрофункцій двійкових лічильників наведено на рис. 8.7.

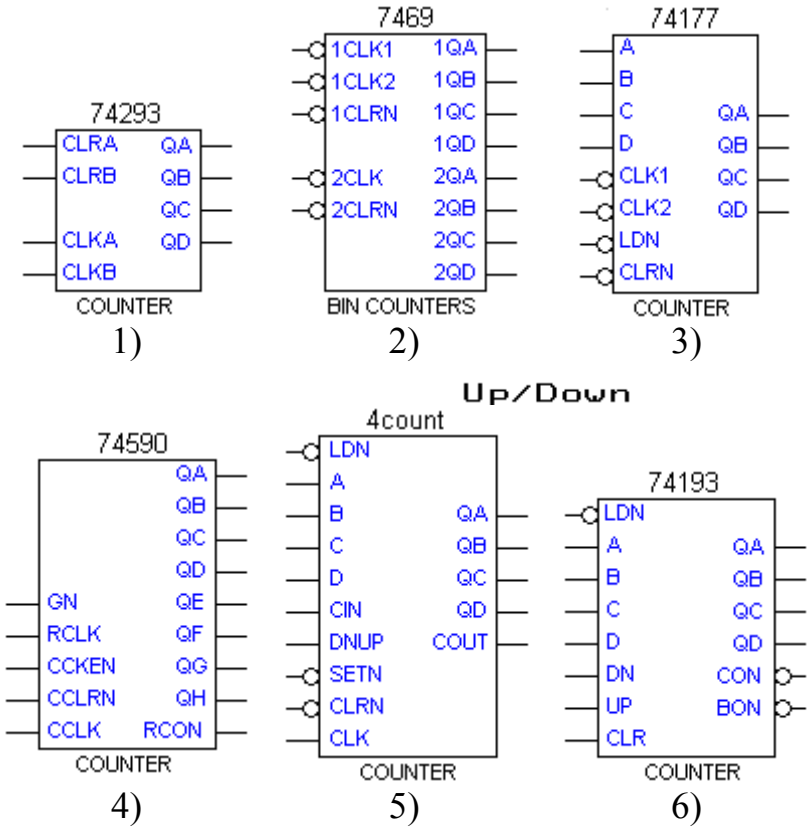


Рисунок 8.7

У таких макрофункціях обов'язковими є лічильний вхід *CLK* (або два лічильні входи *UP* і *DN*) та вхід скидання *CLR* (символи 1 ... 6), часто є також вхід *LD* асинхронного передустановлення до потрібного стану зі входів *A, B, C,* ... у паралельному коді (символи 3, 5, 6). Класичні лічильники є паралельними за схемою на рис. 8.3, б); при цьому в корпусі ІС можуть міститися кілька груп паралельних лічильників з окремими лічильними входами. Послідовним з'єднанням груп з одного і трьох розрядів (символи 1, 2, 3) можна утворити лічильники на 1...4 розряди, а додаванням ще чотирирозрядної гру-

пи (символ 2) – лічильники на 1...8 розрядів. Проте за браком спільного синхровходу утворені таким чином лічильники є асинхронними, з проміжними станами під час зміни вихідного коду. Цього недоліку позбавлена низка ІС 8-розрядних паралельних лічильників, у тому числі з додатковими функціями. Прикладом є лічильник (символ 4) з вихідним регістром (його синхровхід *RCLK*, по виходах *QA* ...*QH* формується вихідний перенос *RCON*), дозволом *CCKEN* на проходження лічильних імпульсів *CCLK*, асинхронним скиданням *CCLRN* та трьома станами виходів (керуються сигналом *GN*).

Реверсивні двійкові лічильники виконано, як правило, за схемою на рис. 8.6, а) (символ 5), інколи за схемою на рис. 8.6, в (символ 6).

8.1.3 Лічильники з довільним модулем лічби

8.1.3.1 Перетворення двійкових лічильників у недвійкові

Якщо в *n*-розрядному двійковому лічильнику з модулем 2^n виключити $M_н = 2^n - M$ надлишкових станів, то він перетвориться в недвійковий з модулем $M < 2^n$. Тому потрібна кількість розрядів *n* двійкового лічильника для утворення недвійкового з модулем *M* обчислюється в САПР як

$$n = Ceil\ (log_2 M), \tag{8.4}$$

де *Ceil* (ceiling – стеля) – найближче ціле число, що не менше виразу в дужках, наприклад, *Ceil* ($log_2 5$) = 3. Або простіше вручну розрядність можна визначити з умови

$$2^{n-1} < M < 2^n, \tag{8.5}$$

наприклад, недвійковий лічильник з модулем $M = 5$, виходячи з $2^2 < 5 < 2^3$, можна утворити з трирозрядного двійкового лічильника, якщо усунути $M_н = 2^n - M = 2^3 - 5 = 3$ надлишкові стани.

Є декілька методів перетворення двійкових лічильників у недвійкові залежно від способу усунення надлишкових станів. Лічильник з *природним* порядком лічби утворюється виключенням *старших* надлишкових станів $N = M, \ldots, 2^n - 1$ шляхом ***примусового скидання*** двійкового лічильника. Принцип такого перетворення розглянемо з самого початку. На ІС жорсткої структури приступними є лише її зовнішні виводи, зокрема, вхід скидання двійкового лічильника *R* і виходи, з яких знімається код *N*. З надходженням лічильних імпульсів *C* вихідний код зростає в межах $N = 0, 1, ..., M$ і перетворюється в дешифраторі в унітарний код. Коли на виході, номер якого збігається з модулем лічби *M*, з'являється активний рівень R_M, лічильник скидається до нуля і далі цикл лічби повторюється. Стан $N = M$, в якому лічильник перебуває короткочасно, не використовується, тому кількість фіксованих станів $N = 0, 1, ..., M - 1$ становить потрібний модуль лічби *M*. Програмований лічильник за методом примусового скидання двійкового лічильника можна побудувати заміною дешифратора на цифровий компаратор (рис. 8.9, а). Коли, під час лічби вихідний код *N*

сягає потрібного модуля M, сигналом R_M лічильник обнуляється, а з надходженням наступних імпульсів на вхід C цикл лічби повторюється.

8.1.3.2 Лічильники зі зворотними зв'язками

Двійкові лічильники утворюються за допомогою *прямих* міжрозрядних зв'язків – з виходів попередніх на входи наступних розрядів. Запровадженням *зворотних* зв'язків – з виходів наступних розрядів на входи попередніх – блокуванням перенесень можна усунути будь-які надлишкові стани, отже, отримати лічильники з довільним модулем і порядком лічби. Якщо будувати в такий спосіб лічильники з послідовним або комбінованим перенесенням між окремими розрядами, потрібно визначати функції збудження як інформаційних, так і синхровходів тригерів, а в паралельних лічильників синхровхід всіх розрядів є спільним, тому достатньо знайти функції збудження лише інформаційних входів.

Методику проектування розглянемо на прикладі побудови паралельного лічильника з модулем $M = 5$ і природним порядком лічби, який має перемикатися за часовими діаграмами (рис. 8.8).

1) За потрібною кількістю розрядів $n = 3$ аналогічно табл. 8.1, 8.2 у перемикальній таблиці (рис. 8.9, а) заповнюємо колонки початкового Q_i та наступного по надходженні лічильного імпульсу Q_i^+ станів тригерів з урахуванням того, що коди $N = 5 \ldots 7$ є надлишковими.

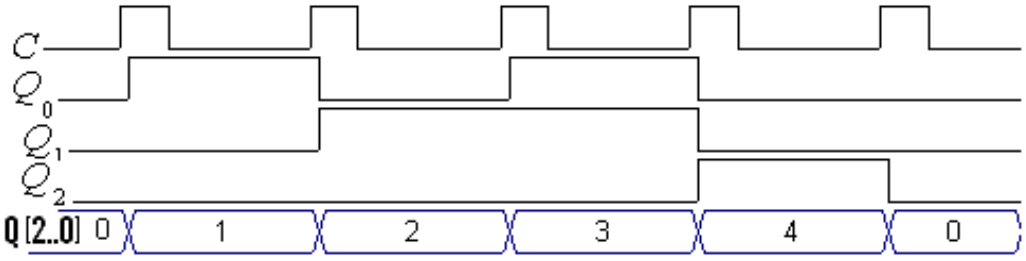


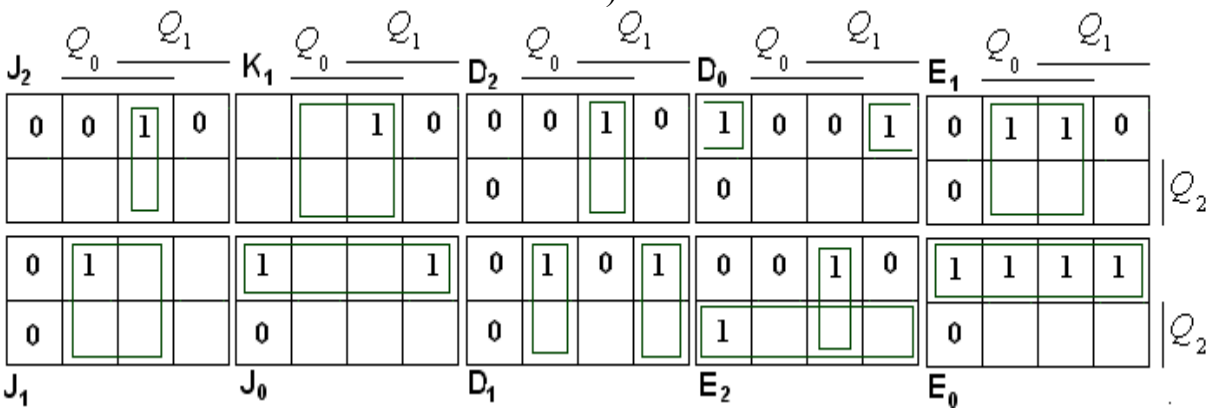
Рисунок 8.8

2) Вибираємо тип тригерів (для прикладу розглянемо варіанти лічильника на JK-, D- і TE-тригерах) та заповнюємо стовпці таблиці для функцій збудження на їхніх інформаційних входах. Так, у JK-тригеру старшого розряду колонкам $Q_2Q_2^+$ лівої частини таблиці відповідають колонки J_2K_2 її правої частини. Тому для переходів $Q_2Q_2^+ = 00, 01, 10$ вносимо значення $J_2K_2 = 0X, 1X, X1$, відтак аналогічно заповнюємо колонки для всіх інших розрядів згідно з таблицею переходів JK-тригера. Для розрядів на D-тригерах вносимо значення $D_i = Q_i^+$, а на TE-тригерах записуємо $E_i = 0$, якщо тригер не перемикається та $E_i = 1$, якщо перемикається.

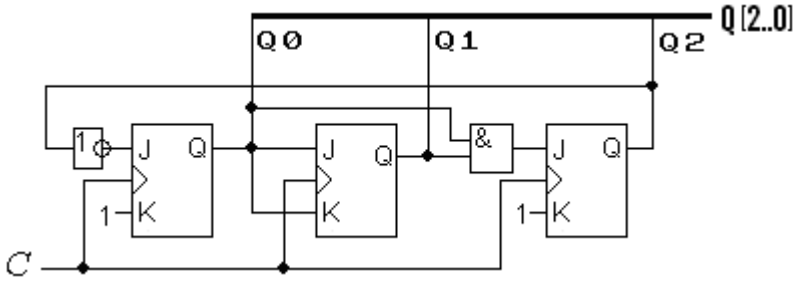
3) Безпосередньо з таблиці довизначенням X = 1 маємо $K_2 = K_0 = 1$, а інші функції збудження мінімізуємо за діаграмами термів (на рис. 8.11, б) порожнім клітинкам відповідають факультативні значення X):

N	$Q_2Q_1Q_0$	$Q_2^+Q_1^+Q_0^+$	$J_2K_2\ J_1K_1\ J_0K_0$	$D_2\ D_1\ D_0$	$E_2\ E_1\ E_0$
0	0 0 0	0 0 1	0X 0X 1X	0 0 1	0 0 1
1	0 0 1	0 1 0	0X 1X X1	0 1 0	0 1 1
2	0 1 0	0 1 1	0X X0 1X	0 1 1	0 0 1
3	0 1 1	1 0 0	1X X1 X1	1 0 0	1 1 1
4	1 0 0	0 0 0	X1 0X 0X	0 0 0	1 0 0
5...7	- - -	- - -	XX XX XX	X X X	X X X

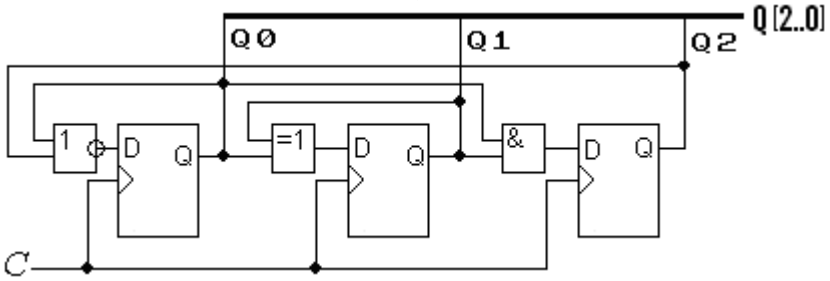
а)



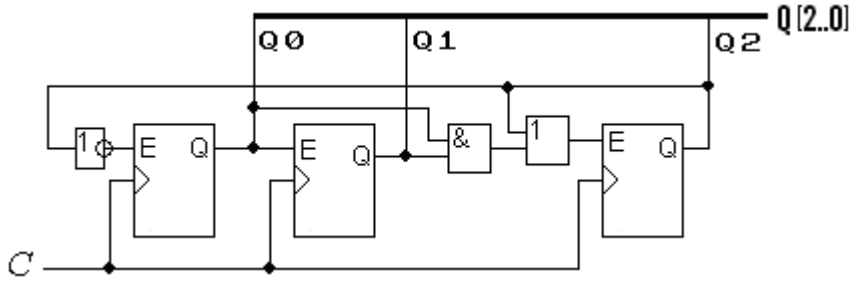
б)



в)



г)



д)

Рисунок 8.9

$$J_2 = D_2 = Q_0 Q_1; J_1 = K_1 = E_1 = Q_0; J_0 = E_0 = \overline{Q_2};$$
$$D_1 = Q_0 \oplus Q_1; D_0 = \overline{Q_0 + Q_2}; E_2 = Q_0 Q_1 + Q_2.$$

4) З'єднуючи синхровходи тригерів зі спільним лічильним входом *C*, а інші входи – згідно з функціями збудження, отримуємо варіанти реалізації лічильника на JK-, D- і TE-тригерах (рис. 8.9, в), г), д) відповідно). Такі лічильники є синхронними, отже, перемикаються без проміжних станів вихідного коду (див. рис. 8.10).

Найпростіший щодо реалізації варіант визначається елементною базою. Так, лічильники на JK-тригерах з дубльованими входами потребують мінімуму додаткових елементів у міжрозрядних зв'язках (при використанні інверсного виходу і подвійних входів у старшому розряді схема на рис. 8.9, в) не містить таких елементів) і можуть виявитися зручними для побудови на ІС жорсткої структури.

На програмованих ІС економічнішими є схеми на D- і RSC-тригерах з динамічним керуванням. Крім того, для низки застосувань доцільною є побудова лічильників з довільним модулем і порядком лічби за схемами з послідовним і комбінованим перенесенням.

8.1.3.3 Макрофункції і універсальна мегафункція

Серед ІС лічильників приблизно половина є декадними, з модулем лічби $M = 10$ (символи 1...7 на рис. 8.10), послідовним з'єднанням яких можна отримати лічильник з модулем $M = 10^n$. Такі лічильники є паралельними за схемою зі зворотними зв'язками і задля гнучкості використання можуть містити в корпусі ІС окремі лічильники з модулем $M = 2$ і з модулем $M = 5$ (див. рис. 8.9, г), послідовне з'єднання яких утворює асинхронний лічильник з модулем $M = 10$ (символи 1, 3, 4). Наявність в корпусі ІС ще й декадного лічильника (символ 1) дає змогу отримати також модулі $M = 20, 50, 100$.

Користуючись входами скидання до нуля *CLR* (символи 1...4), передустановлення до дев'ятого стану *SET* 9 (символи 2, 3) та завантаження сигналом *LDN* зі входів *A, B, C, D* (символ 4), декадні лічильники, як і двійкові, можна перетворити у недвійкові з довільним модулем шляхом примусового скидання або нараховування.

Реверсивні декадні лічильники будуються за таким самим принципом, як і двійкові: з одним лічильним входом *U/DN* (символи 5, 7) або з двома – додавання *UP* та віднімання *DN* (символ 6). Засвоєно випуск низки лічильників з додатковими функціями, наприклад, з вихідним регістром і трьома станами виходу (символ 7) або з дешифратором семисегментного коду (символ 9). Є також спеціалізована щодо програмного пакету макрофункція для лічби в коді Ґрея (символ 8).

Універсальна мегафункція (символ 10) дозволяє реалізувати потрібний тип двійковокодованого лічильника з довільним модулем лічби.

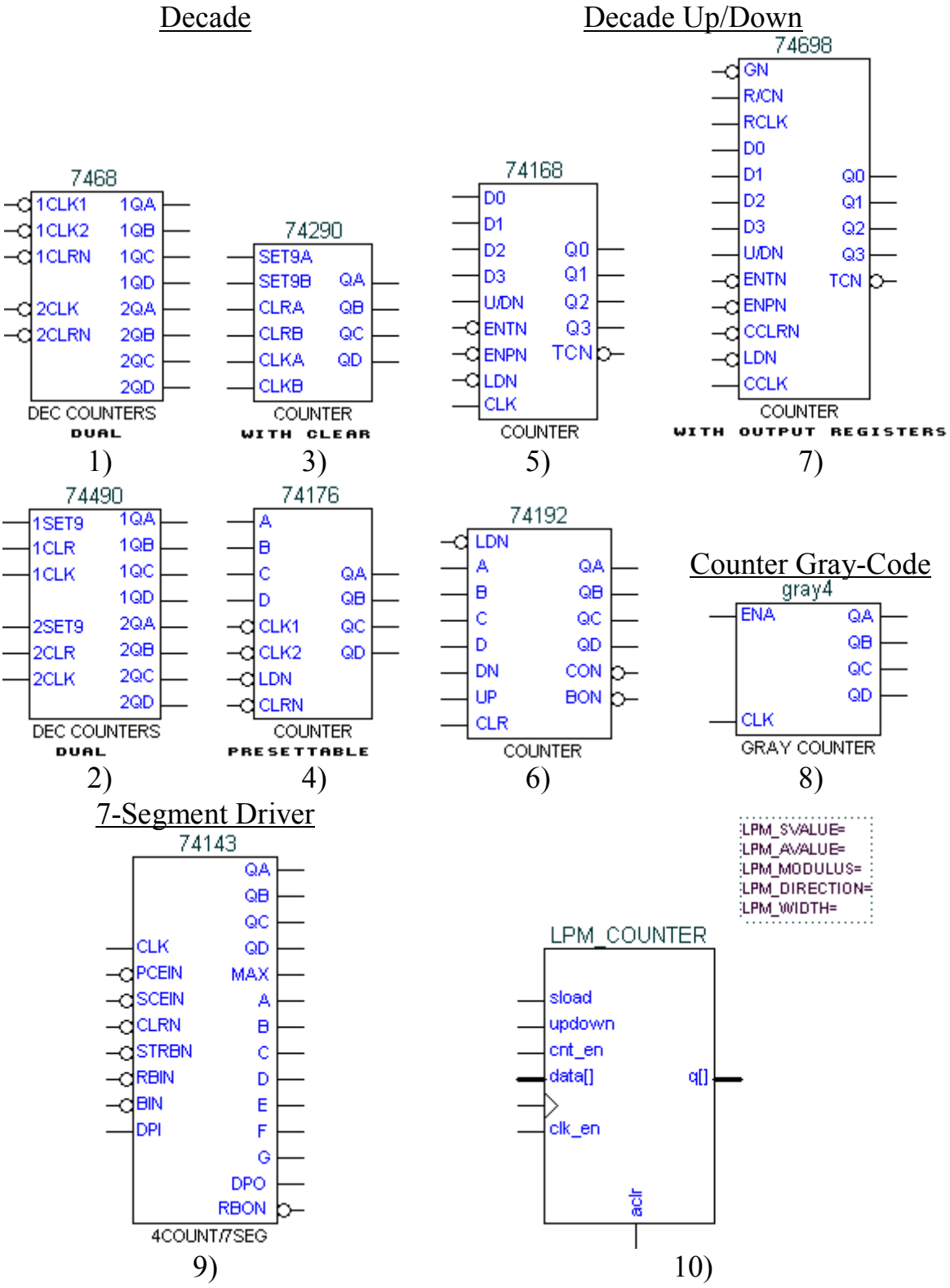


Рисунок 8.10

8.1.4 Подільники частоти

8.1.4.1 Загальна схема безвентильного подільника частоти

Будь-які лічильники можуть використовуватися як подільники частоти, коефіцієнт поділу якої на виході лічильника дорівнює його модулю лічби.

Проте додаткові елементи (вентилі) у міжрозрядних зв'язках або потреба дубльованих входів у тригерів ускладнюють лічильники з довільним модулем і природним порядком лічби. З огляду на те, що в подільниках частоти порядок лічби не має значення, без його дотримання схему гранично спрощують.

Безвентильні лічильники на JK-тригерах *без дубльованих входів* будуються шляхом збільшення модуля лічби на одиницю. Для цього лічильник з *довільним* модулем M_0 охоплюють зворотним зв'язком за допомогою двох тригерів (варіанти схеми на тригерах з інверсним і прямим динамічним керуванням подано на рис. 8.11, а, б). При цьому перший тригер збільшує модуль вдвічі, а останній додає одиницю (через це для стислості його називають „одиничним"), тому в цілому модуль такого лічильника становить $M = 2M_0 + 1$. Наприклад, охоплюючи таким зв'язком чотирирозрядний двійковий лічильник, дістанемо модуль $M = 33$ (рис. 8.11, в).

Так само утворюються лічильники з будь-яким непарним модулем, а при $M_0 = 1$ дістанемо модуль $M = 3$ безпосереднім з'єднанням першого і одиничного тригерів (виокремлена частина на рис. 8. 11, г). Для отримання парного модуля лічби досить послідовно ввімкнути лічильний тригер на вході або на виході. В останньому випадку (див. рис. 8.11, г) вихідні імпульси Q матимуть форму меандра.

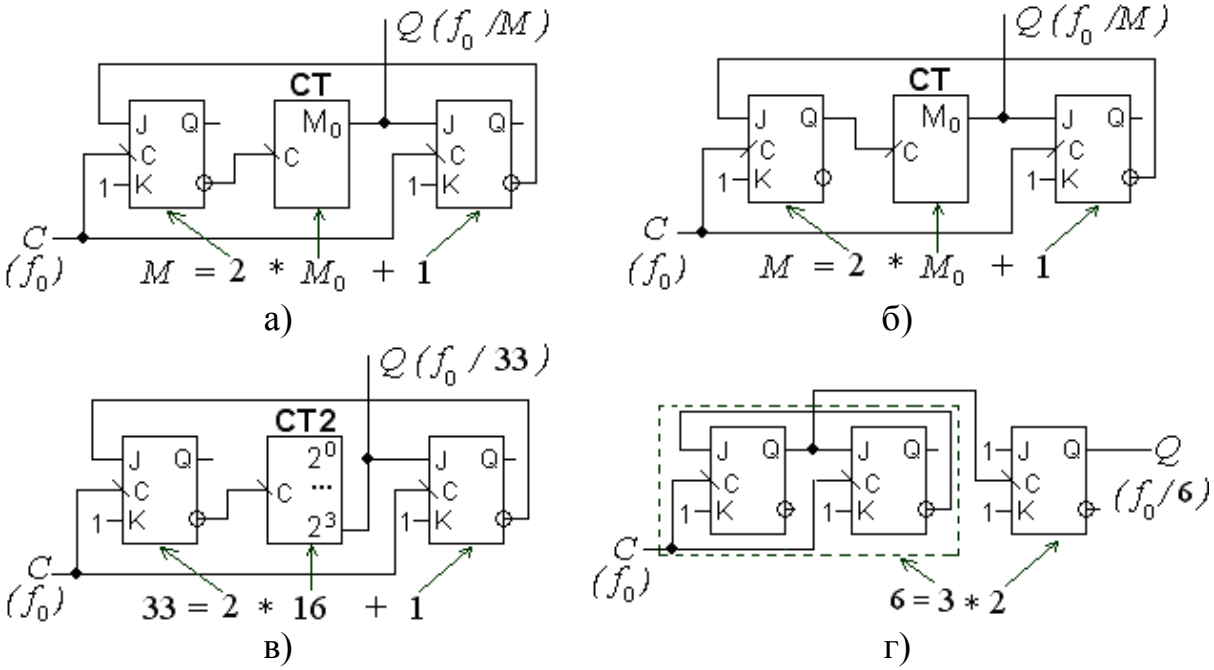


Рисунок 8.11

8.1.4.2 Макрофункції подільників частоти і цифрових таймерів

Випускається обмежена кількість різновидів ІС подільників частоти з фіксованим коефіцієнтом поділу через зниження їх серійноздатності, тому доцільніше перетворювати лічильники на подільники з довільним модулем. Макрофункція подільника частоти програмного пакета freqdiv (символ 1 на рис. 8.10) є паралельним двійковим лічильником з наскрізним

перенесенням. На основі двійкових лічильників будуються подільники частоти з програмованим коефіцієнтом поділу або цифрові таймери (задавачі цифрових інтервалів). Модуль для макрофункцій серії 74 програмується в межах 2^3 ... 2^{15} або 2^3 ... 2^{31}; крім того, за додатковими виходами встановлюються інші модулі, поєднанням яких зручно задавати часові інтервали.

8.2 Лабораторне завдання

8.2.1 Дослідити основні типи *двійкових* лічильників

8.2.1.1 Дослідити двійковий підсумовувальний лічильник з послідовним перенесенням (послідовний лічильник) на основі Т-тригерів: за принциповою електричною схемою і осцилограмами сигналів (схема 1, файли d:\max2work\tutorial\8lab\8dwij.gdf, .scf) з'ясувати принцип побудови і міжрозрядні зв'язки такого лічильника та визначити затримку його перемикання. (У звіті навести схему, умовне графічне позначення лічильника за ДСТУ, перемикальну таблицю, осцилограми сигналів, відомості щодо швидкодії).

8.2.1.2 Дослідити двійковий підсумовувальний лічильник з паралельним перенесенням (паралельний лічильник) на основі ТЕ-тригерів (схему і символ тригера див. у файлах d:\max2work\tutorial\8lab\8te.gdf, .sym) за п. 8.2.1.1 (схема 2, файли d:\max2work\tutorial\8lab\8dwij.gdf, .scf) та розглянути особливості побудови і перемикання базової схеми на D-тригерах (схема 3, файли d:\max2work\tutorial\8lab\8dwij.gdf, .scf). (У звіті навести принаймні одну зі схем, осцилограми сигналів, відомості щодо швидкодії).

8.2.1.3 Розглянути особливості побудови і перемикання двійкового лічильника в режимі віднімання (схема 4, файли d:\max2work\tutorial\ 8lab\8dwij.gdf, .scf) ***та дослідити двійкові реверсивні лічильники*** з подаванням вхідних імпульсів до однієї точки (схема 5, файли d:\max2work\tutorial\8lab\8dwij.gdf, .scf) і двох точок схеми (схема 6, файли d:\max2work\ tutorial\8lab\8dwij.gdf, .scf). (У звіті навести схему реверсивного лічильника на D-тригерах, осцилограми сигналів, стисле пояснення принципу його дії).

8.2.1.4 Дослідити недвійковий лічильник, побудований шляхом ***перетворення двійкового*** (файли d:\max2work\tutorial\8lab\8m16.gdf, .sym) скиданням його надлишкових станів за допомогою дешифратора (файли d:\max2work\tutorial\8lab\8dc.gdf, .sym) на прикладі декадного лічильника (схема 1, файли d:\max2work\tutorial\8lab\8nedw.gdf, .scf); розглянути особливості побудови і перемикання лічильника, в якому за дешифратор править елемент І (схема 2, файли d:\max2work\tutorial\8lab\8nedw.gdf, .scf), ***декадного лічильника*** на стандартній ***макрофункції*** – ІС серії 74 (схема 3, файли d:\max2work\tutorial\8lab\8nedw.gdf, .scf) та реверсивного декадного

лічильника на ***мегафункції*** (схема 4, файли d:\max2work\tutorial\8lab\ 8nedw.gdf, .scf). (У звіті навести схему 2, осцилограми сигналів такого лічильника, стисле пояснення принципу його дії).

8.2.1.5 Ознайомитися з різновидами двійкових лічильників бібліотеки бази даних (файл d:\max2work\tutorial\8lab\8libr.gdf): 1) макрофункціями (дібраними ІС серії 74) двійкових лічильників (Binary) та 2) двійкових реверсивних лічильників (Binary Up/Down), з ***різновидами недвійкових лічильників***: 3) макрофункціями (дібраними ІС серії 74) декадних лічильників (Decade), 4) декадних реверсивних лічильників (Decade Up/Down), з іншими різновидами лічильників і подільників частоти і 5) програмованими цифровими таймерами, а також 6) з універсальною ***мегафункцією*** лічильника. (У звіті навести принаймні по одному символу з груп 1...6, пояснити призначення та особливості входів і виходів, а також параметри мегафункції).

8.2.2 Спроектувати недвійковий лічильник із заданим модулем *M* (згідно з варіантом завдання див. додаток А, варіанти завдання 8, а) і природним порядком лічби шляхом *перетворення* двійкового лічильника.

8.2.2.1 Побудувати лічильник на основі макрофункції вибраного типу за графічного (.gdf) введення проекту 8XXper_gr, виконати компіляцію, моделювання (.scf) та часовий аналіз у формі матриці затримок DELAY MATRIX зі збереженням результату (.tao) і дисплея швидкодії послідовнісних схем REGISTERED PERFORMANCE зі збереженням результату (.tao1).

✍ ***Примітка.*** Тип часового аналізу перемикається з меню Analysis за ввімкненого часового аналізатора Timing Analyzer.

🖫 ***Приклад****:* програмований лічильник з природним порядком лічби і модулем *M* = 5, 7 на макрофункції 74293 у файлах d:\max2work\tutorial\ 8lab\800per_gr.gdf, .scf, .tao, .tao1.

8.2.2.2 Виконати п. 8.2.2.1 ***за текстового*** (.tdf) введення проекту 8XXper_tx.

🖫 ***Приклад****:* програмований лічильник з природним порядком лічби і модулем *M* = 5, 7 на макрофункції 74293 у файлах d:\max2work\tutorial\ 8lab\800per_tx.tdf, .scf.

✍ ***Примітка.*** У текстовому файлі (підсекція Boolean Equation) імена виводів зразка макрофункції мають точно відповідати його функціональному прототипові, який можна скопіювати з довідки Help (клацнути по символу макрофункції в графічному файлі або по її імені в текстовому файлі), наприклад,

AHDL Function Prototype
FUNCTION 74293 (clka, clkb, clra, clrb)

RETURNS (qd, qc, qb, qa);

і далі використовувати в рівняннях (з позначенням імені зразка з крапкою): ct.clra, ct.clrb, ct.clka, ct.clkb, ct.(qd, qc, qb, qa).

8.2.3 Спроектувати паралельний лічильник *зі зворотними зв'язками* із заданими модулем *M* і порядком лічби (згідно з варіантом завдання, див. додаток А, варіанти завдання 8, б) та дослідити його, користуючись часовими діаграмами.

8.2.3.1 Побудувати лічильник на тригерах заданого типу за графічного (.gdf) введення проекту 8XXzz_gr, виконати компіляцію та функціональне моделювання (.scf)

🖫 ***Приклади**:* лічильник зі звичайним порядком лічби і модулем 5 на тригерах типу JK (схема 1), D (схема 2), T (схема 3) та реверсивний лічильник на JK-тригерах (схема 4) у файлах d:\max2work\tutorial\8lab\ 800zz_gr.gdf, .scf.

8.2.3.2 Виконати п. 8.2.3.1 ***на мегафункції лічильника*** в тому самому проекті.

🖫 ***Приклад**:* реверсивний лічильник зі звичайним порядком лічби і модулем 5 на мегафункції (схема 5) у файлах d:\max2work\tutorial\8lab\ 800zz_gr.gdf, .scf.

8.2.3.3 Виконати п. 8.2.3 ***за текстового*** (.tdf) введення проекту 8XXzz_tx на автоматично створеному за допомогою *менеджера MegaWizard Plug-In Manager* різновиді мегафункції лічильника 8XXwiz_rew.sym.

🖫 ***Приклад**:* реверсивний лічильник зі звичайним порядком лічби і модулем 5 на зразку мегафункції d:\max2work\tutorial\8lab\800wiz_rew.sym у файлах d:\max2work\tutorial\8lab\800zz_ tx.tdf, .scf.

☝ ***Примітка.*** Зауваження щодо відповідності імен стосується і мегафункції, наприклад, з автоматично створеного її зразка

FUNCTION 800wiz_rew (clock, updown)
RETURNS (q[2..0]);

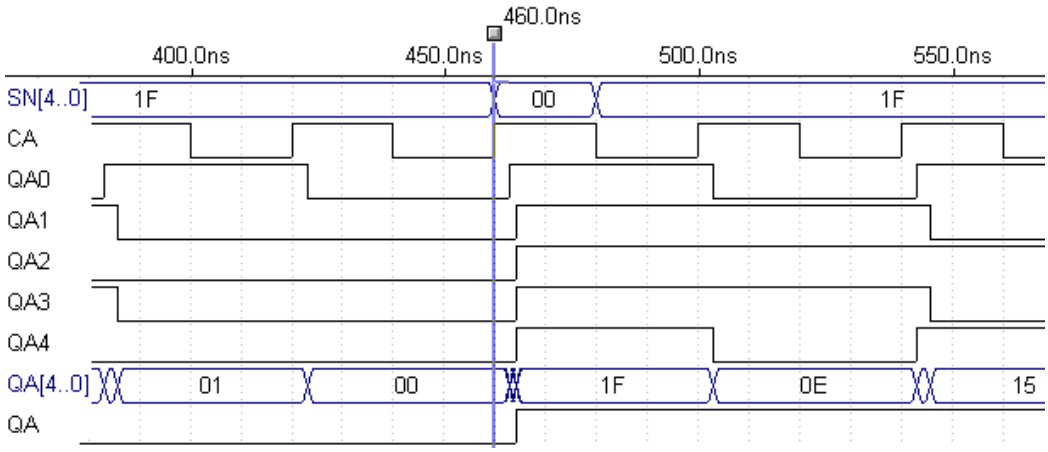
використовуємо в рівняннях ct.clock, ct.updown, ct.q[]. Імена виводів можна взяти з її файлів включення 800wiz_rew.inc або символьного 800wiz_rew.sym.

8.2.4 Спроектувати *подільник частоти* із заданими коефіцієнтом поділу – модулем *M* (згідно з варіантом завдання, див. додаток А, варіант завдання 8, в)

8.2.4.1 Побудувати безвентильний подільник частоти з використанням JK-тригерів за *графічного* (.gdf) введення проекту 8XXpod_gr, виконати компіляцію і моделювання (.scf), визначити його швидкодію та скласти перемикальний граф.

🖫 ***Приклади**:* подільник частоти з коефіцієнтом поділу 5 на основі лічильника 74293 (схема 1) і з коефіцієнтом поділу 11 на JK-тригерах (схема 2) у файлах d:\max2work\tutorial\8lab\800pod_gr.gdf, scf.

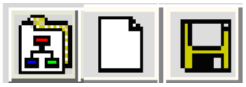
☝ ***Примітка.*** Інверсні асинхронні входи передустановлення SN[4..0] і скидання RN (схема 6) призначені для випробування лічильника на самовідновлюваність і побудови повного перемикального графа. Для цього інверсним кодом, наприклад, SN[4..0] = 00_{16} (інтервал 460...480 нс на часових діаграмах) можна асинхронно записати до всіх розрядів лічильника одиниці і по моделюванні імітатором простежити шлях повернення до робочого циклу: 1F → 0E → 15 → ...; відтак записати з метою наочності в цьому (або іншому) місці інший стан поза робочим циклом, знов змоделювати та простежити шлях і так само повторити цю процедуру для потрібної кількості станів.



8.2.4.2 Створити подільник частоти за текстового (.tdf) введення проекту 8XXpod_sm як *скінченний автомат* (State Machine) за власним вибором оператора логічної секції для позначення його переходів (див. п. 7.2.4) на кшталт файла 8XXpod_gr.gdf та дослідити його. Користуючись часовими діаграмами (.scf), визначити тип пристрою, утворений автоматичним синтезатором як подільник частоти.

🖫 ***Приклади**:* подільник частоти із коефіцієнтом поділу M = 11 – файли d:\max2work\tutorial\8lab\800pod_sm.tdf, scf.

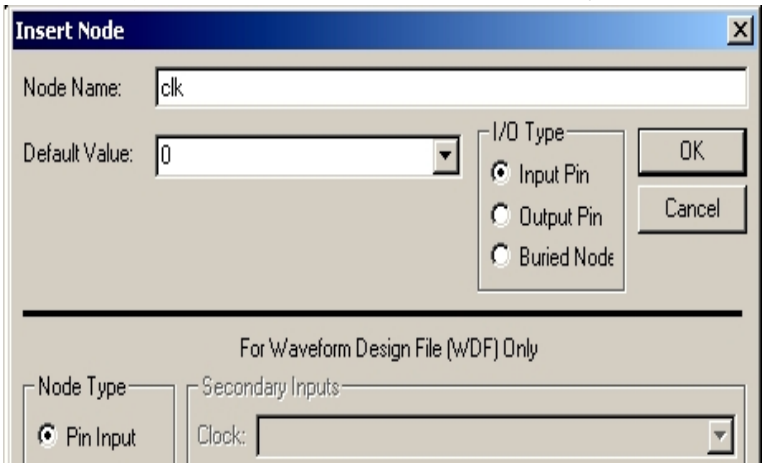
8.2.5 Засвоїти основи створення проекту за *сигнального* його введення (часовими діаграмами) на прикладі автоматичного синтезу подільника частоти із заданими коефіцієнтом поділу – модулем M (згідно з варіантом завдання, див. додаток А, варіанти завдання 8, в).



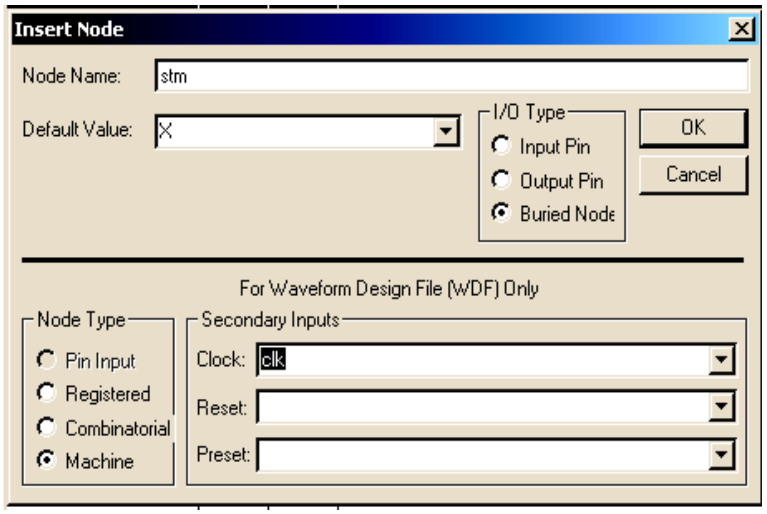
8.2.5.1 Створити новий файл часових діаграм 8XXwav.wdf з розширенням **wdf** (Waveform Design File – проектний файл часових діаграм) за допомогою послідовності наведених піктограм (або командами з меню): визначити ім'я проекту, відкрити новий файл часових діаграм із зазначеним розширенням та надати йому ім'я проекту.

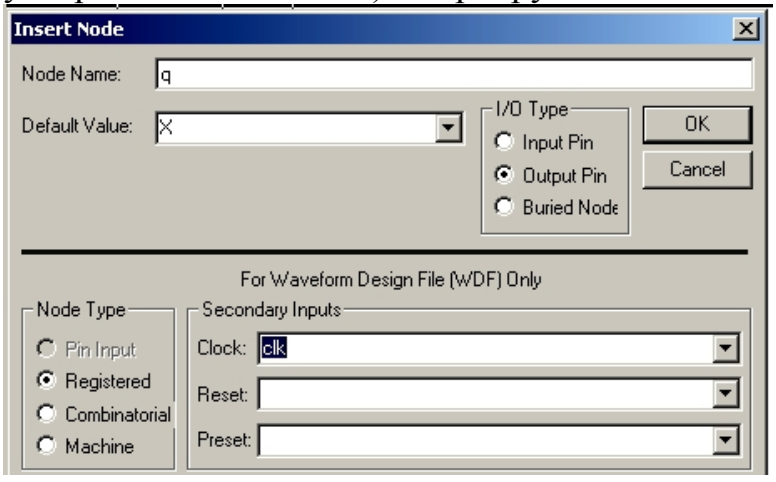
8.2.5.2 Створити такі кола для подільника частоти:

а) вхідне коло лічильних імпульсів: В** по полю у верхній частині діаграм (під заголовками Name, Type, Value) > у діалоговому вікні Insert Node (вставити коло) ввести в графах: Node Name – ім'я вхідного порту, наприклад, clk; Default Value (рівень за замовчуванням) – пасивний рівень, наприклад, 0 (можна прокруткою); I/O Type (вхідний/вихідний тип порту) – прапорець Input Pin > OK (так само, у разі потреби, вводимо інші входи, наприклад, скидання та передустановлення тригерів);



б) внутрішнє коло для логічного зв'язку вхідних і вихідних сигналів: повторюємо п. *а,* але в діалоговому вікні Insert Node вводимо в графах: Node Name – ім'я кінцевого автомата (State Machine), наприклад, stm; I/O Type – прапорець Buried Node (внутрішнє коло); Node Type – ввімкнути Machine; Default Value – залишаємо невизначений стан X (або прокруткою призначаємо третій стан Z чи визначені стани 0 або 1); Secondary Inputs (вторинні входи, тобто керувальні входи, що приєднуються до автомата з утворених вхідних кіл) – прокруткою вводимо Clock (тактові імпульси): clk та, якщо є, так само Reset (скидання) і Preset (передустановлення) > OK;





в) вихідне коло подільника частоти: повторюємо п. а, але в діалоговому вікні Insert Node вводимо в графах: Node Name – ім'я вихідного порту, напри-

клад, q; Default Value – залишаємо довільне значення X (або прокруткою призначаємо потрібний стан); I/O Type – прапорець Output Pin; Node Type – вмикаємо Registered; Secondary Inputs – прокруткою водимо Clock: clk > OK (те саме, у разі потреби, повторюємо для інших виходів, наприклад, можна вивести деякі розряди або їх групи).

Примітки:

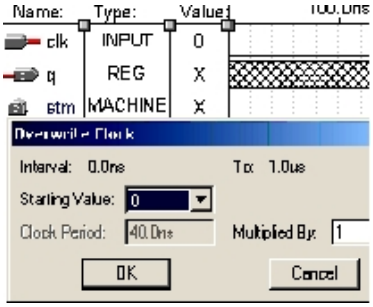
1. Якщо потрібно виправити якесь коло, достатньо подвійним клацанням на ньому в полі заголовків викликати діалогове вікно Edit Node, таке саме, як і Insert Node, та внести необхідні зміни.

2. Діалогове вікно створення кіл Insert Node можна викликати і іншими способами, наприклад, з контекстного меню: натиснути кнопкою B2 в полі Name або Type > Insert Node.

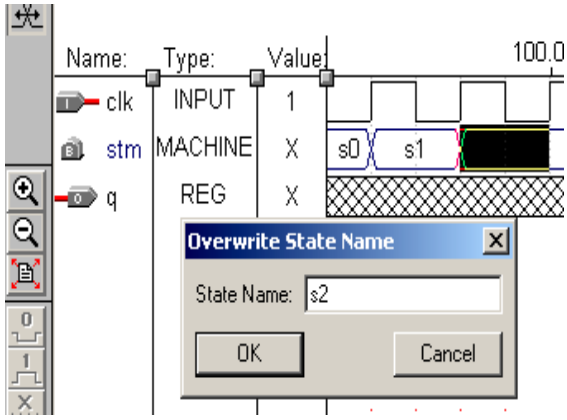
3. Для змінення ширини полів Name, Type, Value достатньо взятися за відповідну прямокутну кнопку вгорі та перетягнути її ліворуч або праворуч.

8.2.5.3 Відредагувати створені часові діаграми:

а) попередньо, як і для SCF-файла, визначити часовий інтервал моделювання: меню File > End Time > ввести потрібну величину, наприклад, залишити 1.0us > OK та позначити часову сітку: з меню Options викликати діалогове віконце Grid Size і ввести потрібний крок сітки, наприклад, 20.0ns > OK; зауважимо, що одиниці вимірювання (us = *мкс*, ns = *нс*) вводяться тут без пропуску після цифри, а для відображення ліній сітки на екрані слід у цьому ж меню встановити прапорець Show Grid;



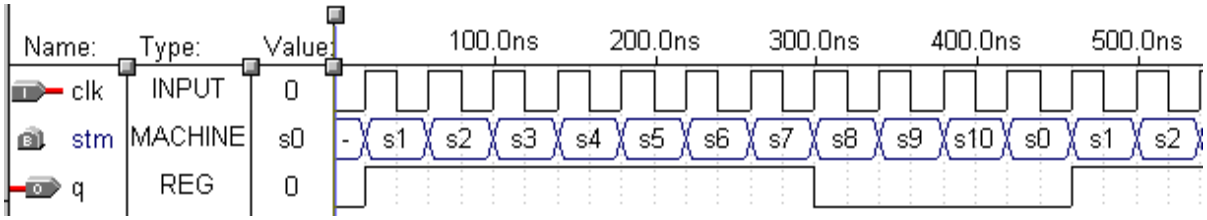
б) ввести вхідні лічильні імпульси у формі меандра: клацнути в полі заголовків (Name, Type або Value) порт clk, рядок якого виділиться чорним, відтак натисненням інструмента Clock (годинник червоного кольору) вертикальної панелі зліва викликати діалогове вікно Overwrite Clock (задати тактові імпульси), в якому або залишити Starting Value (призначений початковий рівень) 0, або прокруткою змінити його на 1 та ввести Multiplied By (коефіцієнт *k* періоду тактових імпульсів), який за замовчуванням при *k* = 1 становить два кроки часової сітки 40.0ns; після натиснення OK відтвориться весь рядок імпульсів;



в) ввести стани скінченного автомата stm: натиснути інструмент редагування (верхня кнопка зі стрілками вертикальної панелі ліворуч), провести курсором редагування (зі стрілками) при утримуваній лівій кнопці ми-

ші над ділянкою рядка stm до першого активного (наприклад, позитивного) перепаду тактових імпульсів, відпустити кнопку, у діалоговому вікні Overwrite State Name (задати ім'я стану), що при цьому з'явиться автоматично, ввести ім'я стану, наприклад, s0, яке по натисненні ОК відобразиться на виділеній ділянці (якщо ім'я не вміщується, відображається риска, але збільшенням розміру лінзою зі знаком плюс ім'я можна відтворити); відтак так само виділити ділянку до наступного активного перепаду тактових імпульсів і ввести ім'я s1; продовжуючи, аналогічно ввести всі стани, кількість яких у даному випадку дорівнює коефіцієнту поділу частоти, наприклад, 11;

г) ввести діаграму імпульсів, яку бажано отримати на виході q подільника частоти: у нашому прикладі їхній період має бути в 11 разів більшим, ніж період вхідних імпульсів clk і складатися з півхвиль різної тривалості (через непарний коефіцієнт поділу), наприклад, із семи і чотирьох періодів імпульсів clk як у проекті 800ctm (оптимальний варіант цього співвідношення можна отримати шляхом порівняння складності під час моделювання). Редагуємо цю часову діаграму звичайним чином (за допомогою курсора виділенням або редагуванням та кнопок „0" і „1").



Примітки:

1. Часові діаграми WDF–файла є ідеалізованими, без урахування затримок в елементах схеми, бо вони лише задають бажане співвідношення між сигналами, тому всі перепади проміжних і вихідних епюр мають точно відповідати моментам зміни вхідних сигналів. В меню Options команда Snap to Grid (сумістити фронти сигналів з вертикальними лініями часової сітки) є завжди ввімкненою, тому в часовій моделі не можна розузгоджувати взаємне положення фронтів, яке може виникнути, наприклад, якщо до WDF-файла вставити скопійовану епюру з деякого SCF-файла (бо в останньому є затримки фронтів).

2. Введення часових діаграм подано найпростішим способом, але є також інші варіанти виходу на діалогові вікна Overwrite... після виділення ділянки – з меню Edit або з контекстного меню (при натисненні B2) розкриваються вкладки, які повторюють у текстовій формі команди інструментів вертикальної панелі.

3. Змінити розмір зображення по горизонталі у WDF-файлі можна так само, як і в SCF-файлі, трьома кнопками вертикальної панелі: збільшення, зменшення масштабу та розміщення всього зображення в межах екрану (кнопка з червоними стрілками).

8.2.5.4 Виконати компіляцію і функціональне моделювання проекту звичайним чином, так само, як за його графічного або текстового введення. Користуючись часовими діаграмами, визначити тип пристрою, утворений синтезатором як подільник частоти, порівняти з результатами проектування за п. 8.2.3, 8.2.4 та переглянути автоматично створений символ подільника.

🖫 ***Приклад****:* подільник частоти із коефіцієнтом поділу M = 11 – файли d:\max2work\tutorial\8lab\800wav.wdf, scf, .sym.

Контрольні питання та завдання

1. Порівняйте лічильники з перенесенням різного типу за критеріями швидкодії і складності. Для яких застосувань послідовні лічильники мають перевагу перед паралельними?

2. Зобразіть раціональну схему з'єднань символів макрофункції 7464 із вхідними і вихідними портами для отримання лічильника з модулем: а) 2^2, б) 2^4, в) 2^5, г) 2^7, д) 2^8, е) 2^{10}, ж) 2^{11}, и) 2^{15}.

3. Спроектуйте лічильник з модулем 2778 і звичайним порядком лічби (для вимірювача фазових зсувів) на основі: а) макрофункцій двійкових лічильників, б) макрофункцій декадних лічильників, в) мегафункції лічильника. *Вказівка: за необхідності скористайтеся виразом* (8.7).

4. Спроектуйте подільник частоти на 60 на основі: а) макрофункцій двійкових лічильників, б) макрофункцій декадних лічильників, в) макрофункцій подільників частоти, г) макрофункцій JK-тригерів (безвентильний), д) макрофункцій цифрових таймерів, е) мегафункції лічильника.

9 ФІЗИЧНЕ ПРОГРАМУВАННЯ ПЛІС

Мета роботи: дослідження типового ЦП на ПЛІС; підготовка матеріалів проекту для фізичного програмування ПЛІС; засвоєння основ програмування і верифікації ПЛІС.

Домашнє завдання

1) Засвоїти теоретичні відомості щодо ПЛІС та інтерфейсу їх програмування; 2) ознайомитися з описом лабораторного стенда на основі плати UP2 (у частині що стосується програмування ПЛІС родини MAX 7000S); 3) підготувати згідно зі своїм варіантом окремий проект за п. 3.2.2.4 для програмування ПЛІС EPM7128S.

9.1 Стислі теоретичні відомості

9.1.1 Загальні відомості

1.1 ***Сучасні ПЛІС***

Інтегровані мікросхеми (ІС) за рівнем інтеграції (який для цифрових ІС найчастіше оцінюється кількістю еквівалентних двовходових логічних елементів в одному кристалі) поділяють на малі (МІС), середні (СІС), великі (ВІС) та надвеликі (НВІС). У вигляді МІС випускаються, в основному, функціонально повні набори логічних елементів, наприклад, І-НЕ, а також допоміжні щодо логіки елементи, наприклад, буфери. З огляду на це МІС *універсальні* – на них можна побудувати ЦП довільної складності, але для цього потрібно багато корпусів ІС, що збільшує габарити і витрати та знижує надійність. Саме тому було налагоджено випуск СІС, які вміщують готові ЦКП або ЦПП, наприклад, дешифратори, мультиплексори, регістри, лічильники невеликої розрядності. При цьому кількість друкованих плат, необхідних для побудови ЦП, зменшується, але внаслідок зниження універсальності СІС є потреба у широкій номенклатурі (розвинуті серії містять до сотень типономіналів ІС), що знижує серійноспроможність мікросхем і, як наслідок, збільшується вартість.

Попри можливості мікроелектроніки, подальший шлях нарощування ступеня інтеграції *стандартних* ІС жорсткої структури виявляється невиправданим – величезні витрати на розробку ВІС і НВІС такого типу не окупаються внаслідок зниження їх універсальності, отже, і серійноспроможності. Через це випускаються *спеціалізовані* ВІС / НВІС на замовлення для масового виробництва, завдяки чому витрати на їх розробку розкладаються на велику кількість виробів і вартість ІС у розрахунку на один виріб стає низькою.

Забезпечення універсальності ІС полягає в їх *програмованості*, коли кожний споживач може налаштовувати стандартну ІС на виконання потрібних функцій. Для здійснення цього є два шляхи. Перший пов’язаний з розробкою мікропроцесорних ІС, структура яких залишається незмінною,

а функції визначаються записаною до пам'яті програмою у вигляді команд, що виконуються одна за одною послідовно в часі. Складність вирішуваних задач є при цьому практично необмеженою (за нарощуваного обсягу пам'яті), але швидкодія через послідовність дій може виявитися недостатньою. Через це в радіотехнічних системах вдаються, наприклад, до застосування апаратних засобів арифметичного множення, яке в мікропроцесорі займає багато часу, бо виконується за алгоритмом з багатьох кроків.

Другий шлях полягає в програмуванні *структури* ІС на виконання потрібних функцій. Властивість програмованості набували вже деякі СІС або їх комбінації (такі як арифметико-логічний пристрій, універсальний логічний модуль на мультиплексорі з вхідним регістром, програмований лічильник на базі двійкового лічильника і компаратора тощо). Проте суттєвий крок в бік програмованості структури було зроблено лише із впровадженням програмованих логічних матриць (ПЛМ) і базових матричних кристалів (БМК) – мікросхем, які стали називати програмованими логічними ІС (ПЛІС). На базі ПЛМ і БМК з'явилися якісно нові складні ВІС і НВІС програмованої структури (випускаються, в основному, фірмами США). За відсутності усталеного слов'янського терміна користуватимемося назвою ПЛІС, розуміючи при цьому їх сучасне покоління.

Крім універсальності і, як наслідок, зниження вартості проектів на їх основі такі НВІС забезпечують високу швидкодію, проте складність виконуваних функцій, на відміну від мікропроцесорів, цілком визначається кількістю елементів на кристалі і системою їх зв'язків (сучасні ІС вміщують до кількох мільйонів еквівалентних вентилів). В міру зростання ступеня інтеграції в ПЛІС стали розміщувати системні модулі, такі як блоки пам'яті, мікроконтролери, мікропроцесори, периферійні пристрої, зокрема, 32-розрядні порти введення-виведення, пристрої звертання до зовнішньої пам'яті, арифметичні помножувачі, помножувачі частоти, системи ФАПЧ, приймачі-передавачі систем передачі даних тощо. Подібні ПЛІС здобули назву SOC (Systems On Chip – системи на кристалі). На цей час освоєно випуск великої кількості типів ПЛІС, які поділяються на родини (родина складається з однотипних ІС різної складності) і цей напрямок радіоелектроніки, що поєднує мікроелектроніку, схемотехніку і САПР, стрімко розвивається.

Сучасні ПЛІС, які крім логічних блоків містять процесори, пам'ять, периферійні пристрої, різні інші модулі, придатні для побудови складних ЦП і радіотехнічних систем обробки інформації, інформаційно-вимірювальних систем і систем зв'язку тощо. Такі ПЛІС здатні замінити громіздке і коштовне обладнання для експериментального відпрацювання прототипів розробок і пришвидшити їх впровадження у виробництво, у тому числі якщо практична реалізація здійснюватиметься на інших засобах. У царині наукових досліджень з'являються можливості швидкого переходу від моделювання в САПР до експерименту на ПЛІС. І, нарешті, усунуто переш-

коди щодо активізації навчального процесу шляхом моделювання і дослідження власних проектів або їх фрагментів, а також є можливість подолати традиційну відсталість експериментальної бази навчальних закладів.

9.1.1.2 Класифікація ПЛІС за способами програмування структури

Під час використання САПР, довідкової інформації, зокрема, про типи мікросхем і їхня програмування доводиться стикатися з масою англомовних термінів (у перекладній літературі їх залишають без перекладу або перекладають по-різному). Крім того, програмовані елементи, в яких зберігається інформація про конфігурацію ІС, за технологією виконання подібні до запам'ятовувальних пристроїв (ЗП), тому в інформації щодо типу ПЛІС зазвичай робляться посилання на тип ЗП (бо історично останні з'явилися раніше). Аби усунути плутанину та послабити труднощі застосування ПЛІС, спричинені термінологією, наведемо стисло класифікацію мікросхем за способами програмування їхньої структури (рис. 1) із зазначенням основних понять (їхні абревіатури виділено в тексті жирним шрифтом), необхідних для розуміння САПР і довідкової інформації.

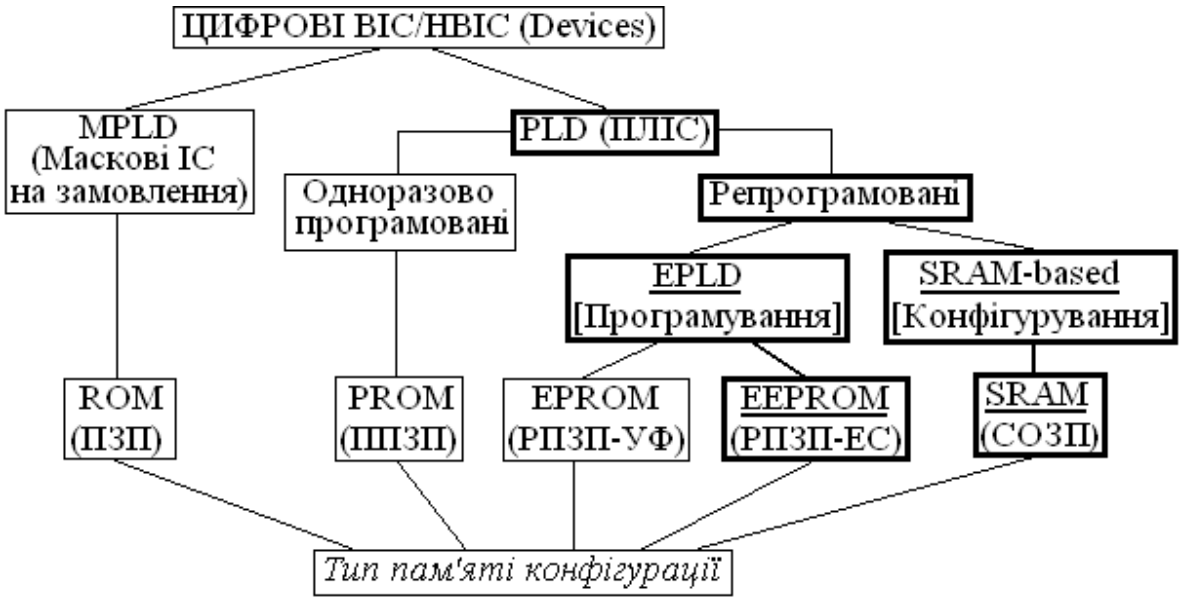


Рисунок 9.1

1) Якщо мікросхеми (в САПР подаються як Devices) *програмуються під час виготовлення* шляхом зміни масок, такий метод базується на інтегральній технології постійних запам'ятовувальних пристроїв (ПЗП), що відповідає англомовному терміну Read-Only Memory (**ROM**) – „пам'ять тільки для читання". Шляхом вилучення комірок пам'яті конфігурації з ПЛІС і оптимізацією їхньої структури для заданого застосування досягаються високі якісні показники мікросхем. Такі ІС (наприклад, фірми Altera) виготовляються на замовлення для великосерійного виробництва,

де вони є рентабельними, або для спеціальної, зокрема, космічної техніки.

2) Для типів ПЛІС, що виконуються за технологією програмованих ПЗП (ППЗП), у САПР використовується термін Programmable ROM (**PROM**). ПЛІС такого типу (наприклад, фірми Actel) є *одноразово програмованими* користувачем, тому їх доцільно застосовувати, як правило, в серійному виробництві, коли проект є ретельно відпрацьованим і виготовлені вироби не підлягають модернізації.

3) Якщо конфігурацію мікросхем можна змінювати шляхом стирання інформації, що в них міститься, і записом нової, за технологією вони відповідають репрограмованим ПЗП (РПЗП). Такі ПЛІС позначаються в САПР як **EPLD –** Erasable Programmable Logic Device (стиранна програмована логічна ІС), а типономінали родини ІС (далі посилання робляться на ІС фірми Altera) містять абревіатуру **EP (**Erasable Programmable), наприклад, ІС типу EPM7032S родини MAX 7000S. Процедура зміни конфігурації мікросхем типу EPLD, яка полягає в стиранні старої і записуванні нової інформації, називається в САПР ***програмуванням***.

4) Різновид мікросхем EPLD, в яких стирання записаної інформації здійснюється ультрафіолетовим промінням, базується на технології РПЗП з ультрафіолетовим стиранням (РПЗП-УФ), що відповідає англійській назві Erasable Programmable Read-Only Memory (**EPROM**), тому вони позначаються як EPROM-based EPLD; прикладом є ІС родин Classic, MAX 5000. ПЛІС, що базуються на технології РПЗП-УФ, компактні, але виконуються в дорогому корпусі з прозорим для ультрафіолетового проміння віконцем. Крім того, для стирання старої інформації потрібно опромінювати кристал (тривалістю біля години) з вилученням ІС із пристрою та кількість циклів стирання обмежена (до 10 ... 100) через деградацію властивостей матеріалів під дією УФ-променів.

5) ПЛІС типу EPLD, в яких пам’ять конфігурації стирається електричними сигналами, за технологією подібні до РПЗП з електричним стиранням (РПЗП-ЕС), що відповідає англомовному терміну Electrically Erasable Programmable Read-Only Memory (**EEPROM** або E^2PROM). До таких ІС, що позначаються як EEPROM-based EPLD, належать родини MAX 7000A, 7000B, 7000E, 7000S, 7000AE (є ще MAX 7000, але ця родина ІС в нових розробках не рекомендована); MAX 3000A; MAX 9000, 9000A (MAX – Multiple Array MatriX – матриця багатьох масивів). Програмування ПЛІС, що базуються на технології РПЗП-ЕС, виконується підімкненням до комп’ютера в звичайних лабораторних умовах або пристрою без вилучення з нього ІС, або апаратного програматора, в який вставляється ІС. Стирання старої і запис нової пам’яті конфігурації здійснюється протягом мілісекунд, а кількість циклів перепрограмування сягає 10^4 ... 10^6. З удосконаленням технології площа програмувальних елементів з електричним стиранням зменшується і такі ПЛІС витісняють схемотехніку з УФ-сти-

ранням. ПЛІС типу EPLD особливо зручно використовувати в нових розробках, що значно скорочує час і витрати.

6) Різновидом РПЗП-ЕС є пам'ять типу **Flash** (спалах, мить), що відрізняється структурою, яка дозволяє виконувати електричне стирання всієї інформації одночасно або великими блоками. Серед EPLD є родина ІС FLASHlogic, яка складається з мікросхем EPX8160 та EPX880.

7) Для типів ПЛІС, пам'ять конфігурації яких виконується за технологією оперативних запам'ятовувальних пристроїв (ОЗП), у САПР використовується термін Random-access memory (**RAM**) – „пам'ять із прямим доступом" (ЗП з довільною вибіркою). Тригерні ЗП належать до статичних ОЗП (СОЗП) – Static RAM (**SRAM**), тому ПЛІС із тригерною пам'яттю конфігурації позначаються як **SRAM-based** devices. До таких ПЛІС належать родини ACEX 1K; FLEX 6000, 6000A; FLEX 8000, 8000A; FLEX 10K, 10KA, 10KE; APEX 20K, 20KC, 20KE; APEX II; ARM-Based Excalibur (мікросхема EPXA10); Mercury та ін. У ПЛІС, що базуються на технології СОЗП, стирання не потрібне, бо в тригерах інформація поновлюється під час її записування, а кількість циклів перепрограмування не обмежена. Процедура зміни конфігурації мікросхем типу SRAM-based, яка полягає в записування нової інформації без стирання старої, називається в САПР ***конфігуруванням***. Проте, на відміну від ПЗП, тригерна пам'ять в ОЗП є енергозалежною: після вимкнення джерела живлення інформація руйнується. Цей недолік подолано шляхом вбудовування в ІС енергонезалежної пам'яті, яка автоматично завантажується до ОЗП конфігурації після вмикання джерела живлення.

ПЛІС фірми Altera типів EPLD і SRAM-based до модифікацій FLEX 10K включно підтримуються САПР MAX+plus II, а подальші розробки, починаючи з APEX 20K (як і всі попередні) підтримуються САПР Quartus II.

9.1.1.3 Програмовані елементи ПЛІС

Програмованість ІС досягається за допомогою програмованих двополюсників *ab*, еквівалентних ключам (рис. 9.2, а). Програмування полягає в зміні провідності двополюсника, тобто в переведенні ключа до замкненого або розімкненого стану. Якщо ключ замкнено, відповідний сигнал s_j надходить до входу логічного елемента ЛЕ, а якщо розімкнено – не надходить, отже вихідний сигнал z_i визначатиметься функцією ЛЕ і запрограмованою ключами сукупністю вхідних сигналів. На спрощеному зображенні замкнені ключі позначаються точками на перетині ліній, а розімкнені – відсутністю зв'язків, тому зазначені двополюсники називаються програмованими точками зв'язку (ПТЗ) або в англомовній документації Programmable Interconnection Point (PIP). Кількість ПТЗ залежно від складності ІС може сягати мільйонів. Зупинимося стисло на основних типах ПТЗ, що застосовуються в сучасних ПЛІС.

Перетинки (рис. 9.2, б) використовуються як ПТЗ в одноразово програмованих ІС (див. рис. 9.1). Під час програмування залишають лише необхідні елементи зв'язку *ab*, а непотрібні усувають перепалюванням тонких перетинок імпульсами струму достатньої величини і тривалості. У сучасних ПЛІС із перетинками новітньої технології у вигляді тришарового діелектрика оксид-нітрид-оксид, навпаки, програмувальним імпульсом напруги перетинка пробивається, завдяки чому створюється провідний канал між точками *a, b*. Перевагою ІС із ПТЗ останнього типу є компактність і низька вартість, а недоліком – доцільність застосування лише в серійному виробництві, коли проект є ретельно відпрацьованим і виготовлені вироби не підлягають модернізації.

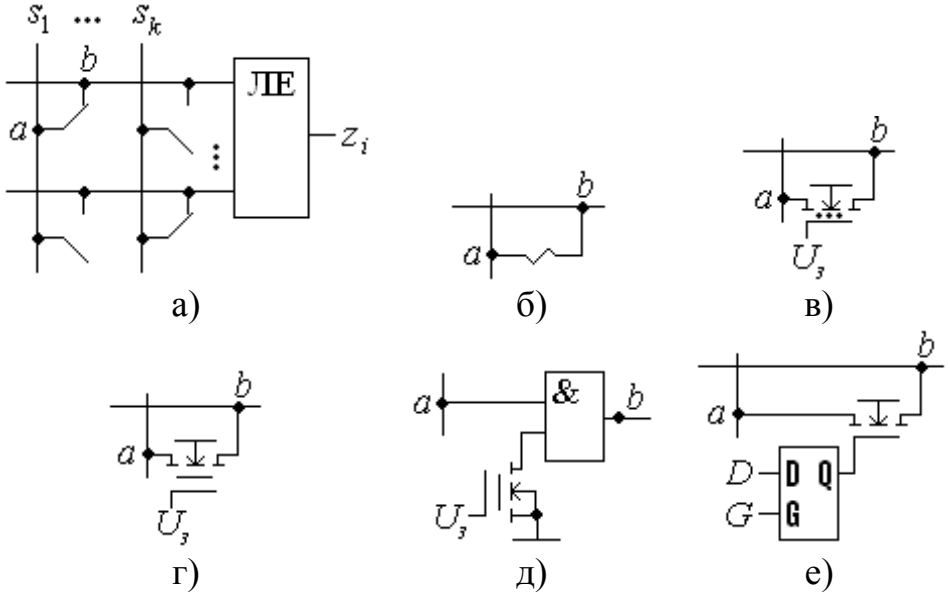


Рисунок 9.2

Елементом пам'яті конфігурації сучасних ПЛІС типу EPLD з УФ-стиранням є транзистори типу *ЛІЗМОН з плаваючим затвором* (ЛІЗМОН – МОН-транзистори з лавинною інжекцією заряду). Плаваючий затвор може бути єдиний або додатковий до звичайного затвору як умовно зображено на рис. 9.2, в). Область між керувальним затвором і каналом, оточена з усіх боків діелектриком, називається плаваючим затвором. Під час програмування напругою на керувальному затворі $U_з$ до цієї області вводиться заряд, що визначає стан транзистора: відкритий транзистор з'єднує точки *a, b*, а запертий – залишає їх роз'єднаними. Після зняття програмувальної напруги цей заряд здатний зберігатися довгий час (протягом багатьох років), тобто запам'ятовувати конфігурацію ІС. Перед перепрограмуванням виконують стирання інформації в елементі пам'яті УФ-промінням через прозоре віконце в корпусі ІС. Це проміння спричиняє фотоструми і теплові струми, які руйнують заряд плаваючого затвору за кілька десятків хвилин.

Аналогічним за принципом дії є елемент пам'яті конфігурації ПЛІС типу EPLD з електричним стиранням на транзисторі типу *ЛІЗМОН з по-*

двійним затвором (рис. 9.2, г). Між керувальним затвором і каналом вміщується оточена з усіх боків діелектриком провідна ділянка з полікремнію або металу, яка утворює другий затвор. Під час програмування до нього як у пастку потрапляє заряд, який залишається після зняття програмувальної напруги $U_з$. Запис здійснюється подачею на затвор високої напруги, при відімкненому затворі інформація зберігається, а подачею низької напруги (заземленням) відбувається швидке стирання пам'яті конфігурації. З метою зменшення затримки поширення через програмувальні ключі їх вилучають з кола передачі сигналу (рис. 9.2, д): якщо до ЛІЗМОН-транзистора записано логічну 1, сигнал передається через елемент І з малою затримкою, а якщо логічний 0 – ділянка між точками *a, b* розімкнена.

У ПЛІС типу SRAM-based програмованою точкою зв'язку є ключовий транзистор (pass-transistor) між точками *a, b,* а елементом пам'яті конфігурації – тригер (рис. 9.2, е). Під час програмування сигналом $G = 1$ тригер активізується і через вхід D до нього записується інформація. У робочому режимі сигналом $G = 0$ тригер перебуває в режимі зберігання конфігурації: якщо $Q = 1$, точки *a, b* відчиненим транзистором з'єднуються, а якщо $Q = 0$ транзистор зачинено і точки *a, b* залишаються роз'єднаними. До тригера не ставиться вимога високої швидкодії, тому його проектують з міркувань максимально можливої компактності і стійкості статичних станів. У такій схемі перед конфігуруванням не потрібна процедура стирання інформації, а енергонезалежність забезпечується елементами пам'яті, які не розподілені по всьому кристалу як ПТЗ. Після ввімкнення джерела живлення автоматично відбувається ініціалізація – процедура конфігурування шляхом перезаписування інформації з енергонезалежної пам'яті до тригерів. Тригерна пам'ять конфігурації широко застосовується в останніх розробка ПЛІС.

9.1.1.4 Принцип побудови програмованих логічних матриць

Для реалізації логічних функцій в диз'юнктивній формі спочатку в елементах І утворюють терми – добутки змінних, а відтак потрібні терми підсумовують в елементах АБО. Регулярні структури з елементів І та АБО, в яких змінні до входів елементів І та терми до входів елементів АБО вибираються за допомогою ПТЗ, називають програмованими логічними матрицями ПЛМ (Programmable Logic Array, PLA).

Проілюструємо утворення таких матриць на прикладі побудови повного суматора розрядів двох чисел a_i, b_i та вхідного переносу до цього розряду c_{in} (рис. 9.3, а). Мінімізуємо функцію вихідного переносу c_{out}, а функція суми s_i не піддається мінімізації (рис. 9.3, б), тому їх ДНФ має вигляд:

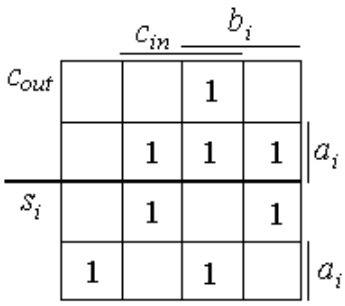
$$c_{out} = a_i b_i + a_i c_{in} + b_i c_{in} = t_1 + t_2 + t_3;$$

$$s_i = a_i \overline{b_i}\,\overline{c_{in}} + \overline{a_i} b_i \overline{c_{in}} + \overline{a_i}\,\overline{b_i} c_{in} + a_i b_i c_{in} = t_4 + t_5 + t_6 + t_7,$$

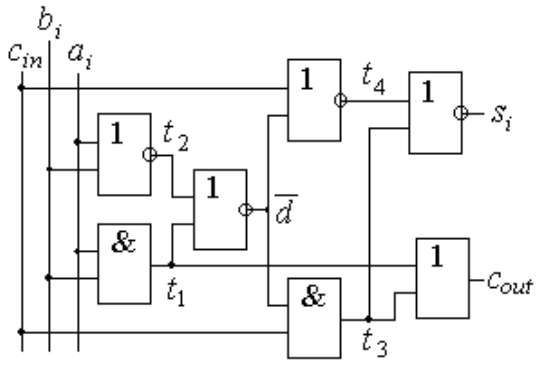
де t_i – відповідні терми.

i	a_i	b_i	c_{in}	c_{out}	s_i
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	1	1

а)



б)

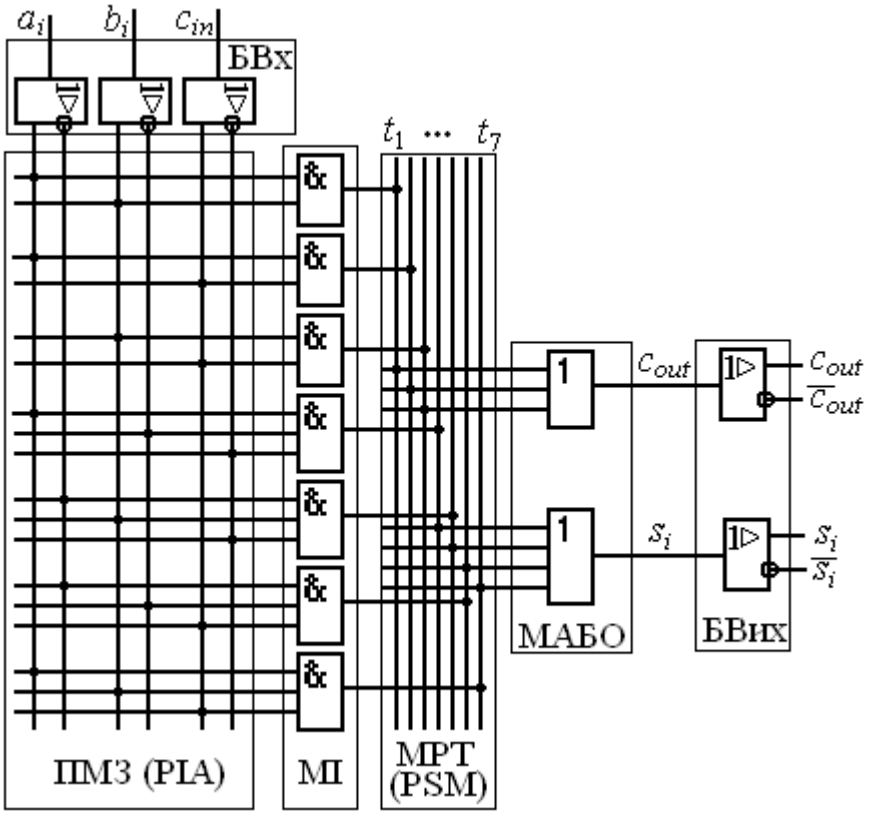


в)

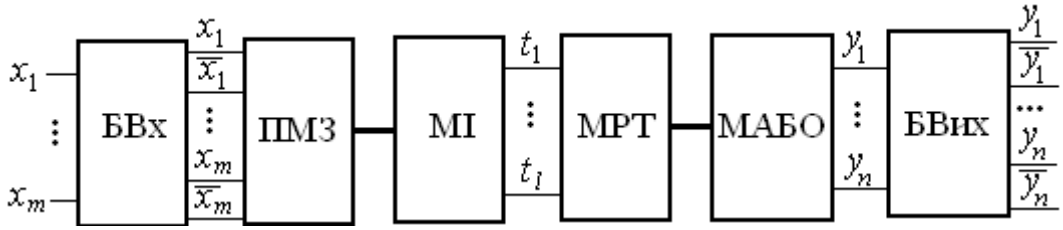
Рисунок 9.3

Реалізацію суматора згідно з цими виразами зображено на рис. 9.4, а. Вхідні однофазні сигнали a_i, b_i, c_{in} у блоці вхідних буферів БВх перетворюються в парафазні і підсилюються за потужністю для забезпечення необхідного коефіцієнта розгалуження. У програмованій матриці з'єднань ПМЗ (Programmable Interconnect Array, PIA) за допомогою ПТЗ (ввімкнені елементи з'єднань позначено точками) виконуються потрібні з'єднання прямих або інверсних сигналів зі входами матриці МІ елементів І для утворення термів t_j. Вихід кожного елемента І живить одну з вертикальних ліній матриці розподілу термів МРТ (Product-Term Select Matrix, PSM), щоб можна було використовувати одні й ті самі терми для утворення різних функцій. Горизонтальними лініями так само за допомогою ПТЗ вибираються потрібні терми, які надходять до входів матриці МАБО елементів АБО, де і реалізуються задані логічні функції. Блок вихідних буферів БВих призначений для забезпечення навантажувальної здатності виходів, формування сигналів потрібної полярності та, у разі потреби, і рівнів (для зменшення споживаної потужності внутрішні елементи логічної матриці можуть живитися від джерела з меншою напругою, ніж зовнішні елементи). Вихідні буфери здатні виконувати й інші функції, зокрема, зберігати інформацію у вихідному регістрі, а за застосування елементів з трьома станами виходу також приєднувати ІС до зовнішньої шини або від'єднувати від неї додатковим сигналом дозволу *OE*.

Таким чином, структурна схема ПЛМ набуває вигляду рис. 9.4, б). З числа m змінних $x_1 \ldots x_m$ у матриці І формуються терми $t_1 \ldots t_l$, отже, кількість l кон'юнкторів дорівнює числу термів. Відтак у матриці АБО утворюються функції $y_1 \ldots y_n$, тому кількість диз'юнкторів n дорівнює числу функцій. Розмірність матриці $m : l : n$ є одним з основних її параметрів. Другим важливим параметром є швидкодія, яка для схеми на рис. 9.4, а) буде найбільшою для базових елементів певного типу через двоступеневу (без урахування буферів) реалізацію функцій.



а)



б)

Рисунок 9.4

Проте задля ощадливості щодо ресурсу і гнучкості побудови вдаються до розширення функціональних можливостей ПЛМ. Так, складність схеми (див. рис. 4,а) q = 9 / 25 (кількість логічних елементів без буферів / загальна кількість їх входів) можна зменшити до q = 7 / 14 (рис. 3,в) за спільної схемної мінімізації (див. п. 2.1.3). Перетворенням схеми (див. рис. 9.3, в) до структури І, АБО

$$t_1 = a_i b_i;\ t_2 = \overline{a_i + b_i} = \overline{a_i}\,\overline{b_i};\ \overline{d} = \overline{t_1 + t_2},\ d = t_1 + t_2;$$

$$t_3 = c_{in}\overline{d};\ t_4 = \overline{c_{in} + \overline{d}} = \overline{c_{in}}d;\ c_{out} = t_1 + t_3;\ s_i = \overline{t_3 + t_4}$$

дістанемо реалізацію суматора на ПЛМ (рис. 9.5). На цій схемі подано загальноприйняте для ВІС і НВІС спрощене зображення елементів: багатовходові кон'юнктури і диз'юнктори замінюються одновходовими. Якщо на горизонтальній лінії, що відображає входи елемента, є точка перетину з вертикальною сигнальною лінією, то ця змінна з'єднується зі входом елемента, а якщо точка відсутня, то не з'єднується. Отже, кількість точок на

горизонтальній лінії дорівнює числу задіяних входів елемента.

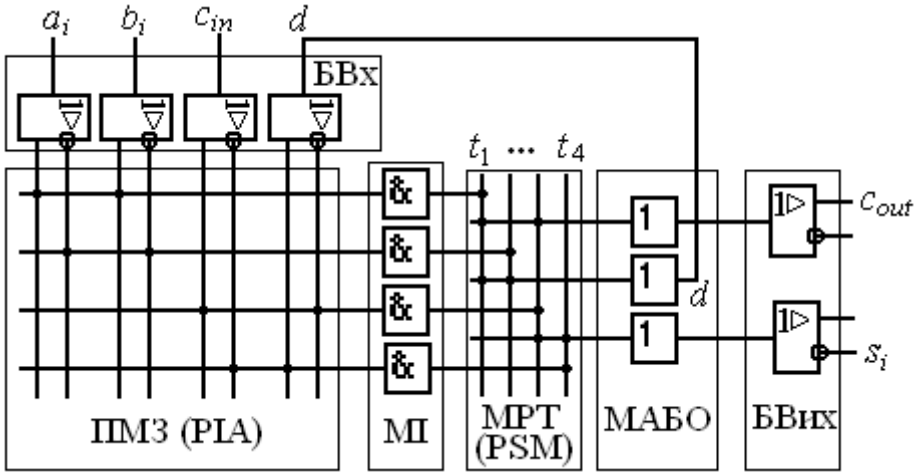


Рисунок 9.5

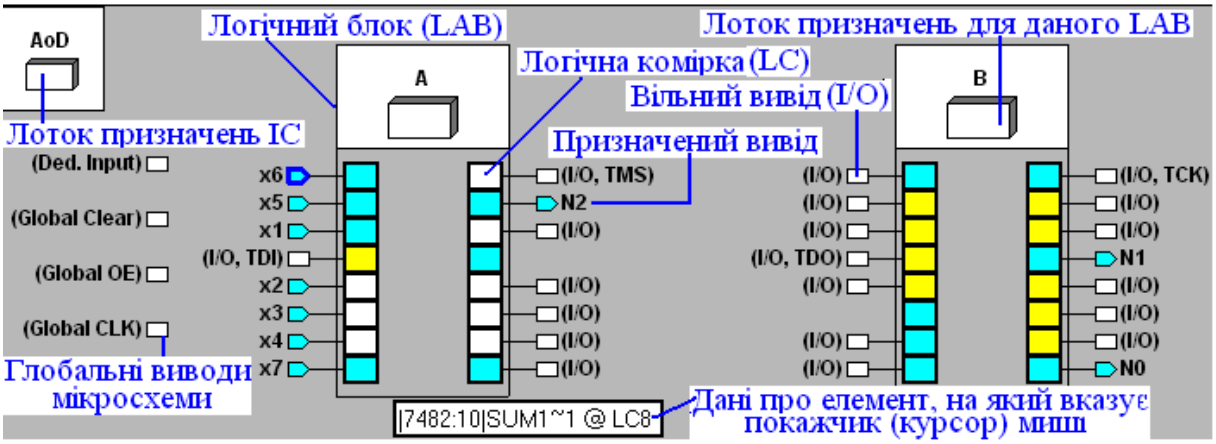
Спрощення ПЛМ відбувається завдяки так званому розділюваному (спільному) розширенню: спільні вертикальні лінії термів МРТ використовуються неодноразово для утворення диз'юнкцій у МАБО та паралельному розширенню: диз'юнкції з виходу окремих елементів АБО (у прикладі диз'юнкція *d*) знов повертаються до ПМЗ для утворення інших логічних функцій. Проте через збільшення глибини реалізації збільшується також і затримка поширення сигналів, так само, як і на рис. 9.3, в).

9.1.2 Архітектура ПЛІС

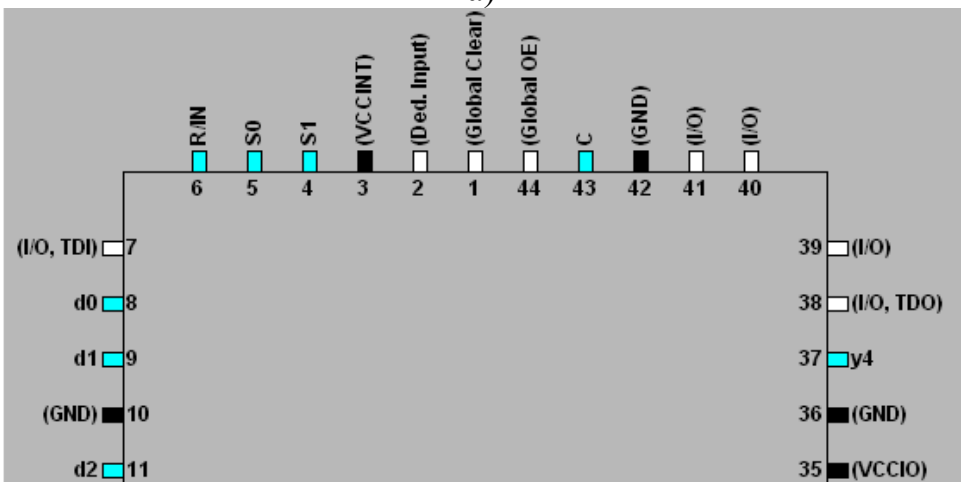
9.1.2.1 План розміщення ПЛІС

ПЛІС, що крім багатьох ПЛМ містить регістри, буфери, розгалужену систему програмованих з'єднань тощо, структурована на окремі частини. У загальному вигляді ці частини зображаються в САПР на плані її розміщення (Floorplan) як подано на рис. 9.6, а). Компілятор автоматично призначає елементи схеми, необхідні для реалізації проекту, виконує потрібні з'єднання і вносить всю цю інформацію до програмувального файла (після цього проектувальник має змогу скоригувати призначення вручну і знов перекомпілювати проект). Призначення (Assignments) для мікросхеми на плані розміщення містяться в лотку AoD (Anywhere on Device) і відображаються при натисканні на нього.

Найбільшою функціональною частиною ПЛІС є логічний блок ЛБ (Logic Array Block, LAB), в якому згруповано набір логічних ресурсів таким чином, що будь-який сигнал є доступний для кожного елемента набору. На плані ЛБ позначено літерами A, B біля лотків, в яких відображаються призначення відповідного блоку. ПЛІС однієї родини однотипні і відрізняються кількістю ЛБ; у прикладі подано ІС родини MAX3000A найменшої складності з двох ЛБ, а ІС цієї родини найбільшої (на цей час) складності містить 32 такі блоки.



а)



б)

Рисунок 9.6

Наступною за ієрархією складовою частиною є логічна комірка ЛК (Logic Cell, LCELL, LC), яка в технічній документації для ПЛІС з електричним стиранням (EEPROM-based) називається макрокоміркою МК (Macrocell, MC). Кожний ЛБ розглядуваного типу ПЛІС містить 16 макрокомірок, а їх загальна кількість у мікросхемі визначає її назву. Так, у родині MAX3000A найпростіша мікросхема MAX3032A містить у двох логічних блоках 32 макрокомірки, а найскладніша MAX3512A – 512 МК у складі 32 ЛБ. Призначені (задіяні) комірки на плані забарвлюються, а вільні залишаються білими.

Так само і виводи (контакти, штирки – Pin) мікросхеми на плані (див. рис. 9.6, а) і на цоколівці (на рис. 9.6, б) подано її фрагмент), що є призначеними, забарвлено, а вільні – білі. Основна їхня частина належить до входів/виходів користувача (User I/O Pins), позначається на вільних виводах у дужках як (I/O), а на призначених виводах – їх іменами відповідно до проектного файла. Інші виводи є спеціалізованими (Dedicated). Перш за все це GND – земля та VCC або VCCINT (5 *В*) і VCCIO (3,3 *В*) – напруги живлення (відповідно до довідника, в дужках подано приклад); ці виводи, як і незадіяні (RESERVED або N.C. – No Connect), якщо вони є, наводяться лише на цоколівці (виводи GND у схемі не можна залишати вільними, навіть якщо їх деякі-

лька). Наступна група спеціалізованих виводів, призначена для фізичного програмування (TDI, TDO, TCK, TMS), є недоступною для користувача (пояснюється нижче). І, нарешті, остання група спеціалізованих виводів для так званих глобальних сигналів, до яких належать спільні для всіх ЛБ сигнали, зображається на плані окремо від них (у нас – ліворуч). Типовими є такі глобальні сигнали: GCLK – Global Clock (спільний синхровхід, у нас показано як призначений для синхросигналу С), GCLRn – Global Clear (спільний сигнал скидання, зазвичай інверсний), Global OE – дозвіл виходу для компонентів з трьома станами, Ded. Input –глобальний вхід за призначенням користувача.

Будь-який вивід ІС, розташований на плані біля комірки, може бути запрограмований як вхід або вихід пристрою (у разі потреби, як двоспрямований вхід/вихід). Якщо певний контакт стає виходом (на плані позначається стрілкою, що виходить з ЛБ), він закріплюється за суміжною з ним коміркою, яка стає кінцевою і остаточно формує для нього сигнал (такі комірки забарвлено блакитним кольором). Якщо ж контакт стає входом (позначається стрілкою, що входить до ЛБ), він не закріплюється за певною МК, а пов’язується з будь-якими комірками через програмовані матриці з’єднань. Так само МК, призначені як проміжні (забарвлені жовтим кольором), що беруть участь у формуванні проміжної логіки, обмінюються вхідними і вихідними даними з виводами і з іншими комірками через програмовані матриці з’єднань. Деякі МК, що не мають суміжних виводів (у нашій ПЛІС дві таких МК) не можуть бути запрограмовані як кінцеві.

9.1.2.2 Структура ПЛІС із електричним стиранням

Зв’язки між компонентами ПЛІС відображаються на її плані графічно (кольоровими лініями) або логічними рівняннями (у спеціальному вікні на частині екрана). Проте наочніше такі зв’язки можна показати на структурній схемі як дещо спрощено подано на рис. 9.7 для типової ПЛІС із електричним стиранням (типу EEPROM-based EPLD) найменшої складності (ширина шин і кількість елементів, позначених цифрами, у ПЛІС різного типу можуть відрізнятися). Структура складнішої ПЛІС така сама: програмована матриця з’єднань ПМЗ є спільною і продовжується вниз, а з обох сторін ярусами розташовуються логічні блоки ЛБ, кожний зі своїм блоком керування введенням/виведенням БКВВ. Показані знизу інтерфейсні кола і сигнали, що діють у режимі програмування, розглядаються в наступному розділі.

Основні зв’язки між структурними складниками ПЛІС здійснюються через ПМЗ, до якої в робочому режимі надходять глобальні сигнали зі спеціалізованих входів безпосередньо (показано вгорі; у деяких ПЛІС може бути два синхровходи для організації двотактових схем), інформаційні сигнали з контактних площинок КП входів/виходів загального користування через БКВВ (показано з обох сторін нижніми шинами розрядністю 6...12 – за кількістю КП, що обслуговуються даним ЛБ) та сигнали з виходів макроко-

мірок ЛБ (показано 16-розрядними шинами – за кількістю МК в кожному ЛБ).

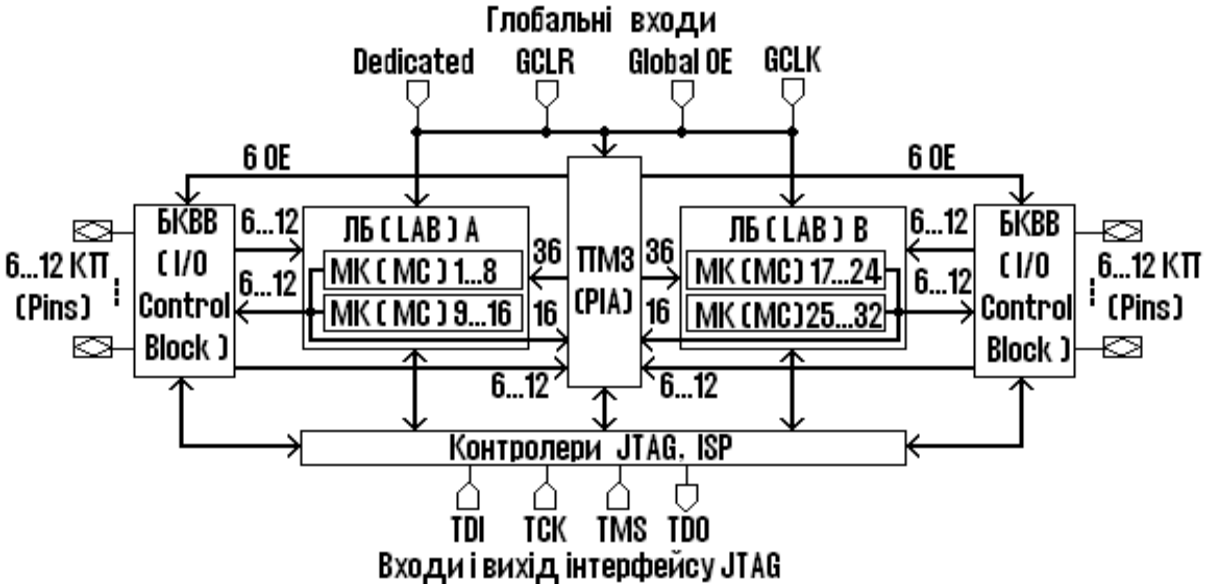


Рисунок 9.7

Усі сигнали ПМЗ стають доступними для макрокомірок кожного ЛБ: потрібні з них програмуванням ПМЗ вводяться до блоків 36-розрядними шинами. Крім того, з метою підвищення швидкодії ЛБ можуть одержувати сигнали колами швидкого введення безпосередньо з глобальних входів і з контактних площинок через БКВВ (шини розрядністю 6...12). У логічних блоках відбувається обробка інформації і її остаточні результати виводяться на КП через БКВВ (шини 6...12), а проміжні результати надходять до ПМЗ (шини 16) для передачі до інших блоків, в яких завершується обробка.

Блоки керування введенням/виведенням БКВВ програмуються на введення інформації (КП стають входами) або на її виведення (КП стають виходами). Під дією шести глобальних сигналів дозволу виходу, що надходять з ПМЗ (шина 6 OE) досягається гнучкість керування блоком.

9.1.2.3 Макрокомірка

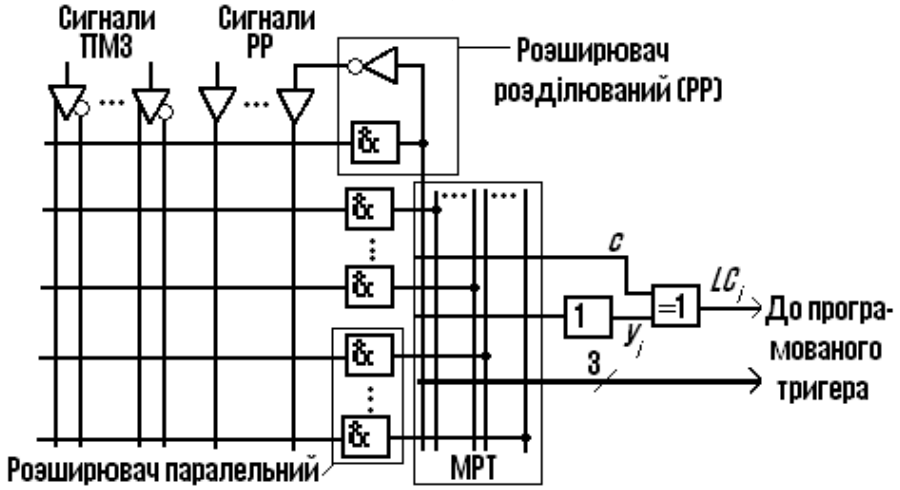
Макрокомірка МК (або логічна комірка ЛК) охоплює ресурс ПЛІС, потрібний для формування одного сигналу, який може бути або вихідним, або проміжним щодо перетворення інформації. Вона складається з двох частин: комбінаційної на основі ПЛМ і послідовнісної у вигляді програмованого тригера (рис. 9.8, а).

До ПЛМ (рис. 9.8, б) із ПМЗ надходять змінні від зовнішніх виводів і з інших комірок. Із цих змінних у локальній матриці І формуються терми, які в МРТ програмуванням з'єднуються зі входами елемента АБО для формування диз'юнкції y_i. Таким чином, ця логічна частина МК не відрізняється від звичайної ПЛМ відносно одного з її виходів (див. рис. 9.4, а).

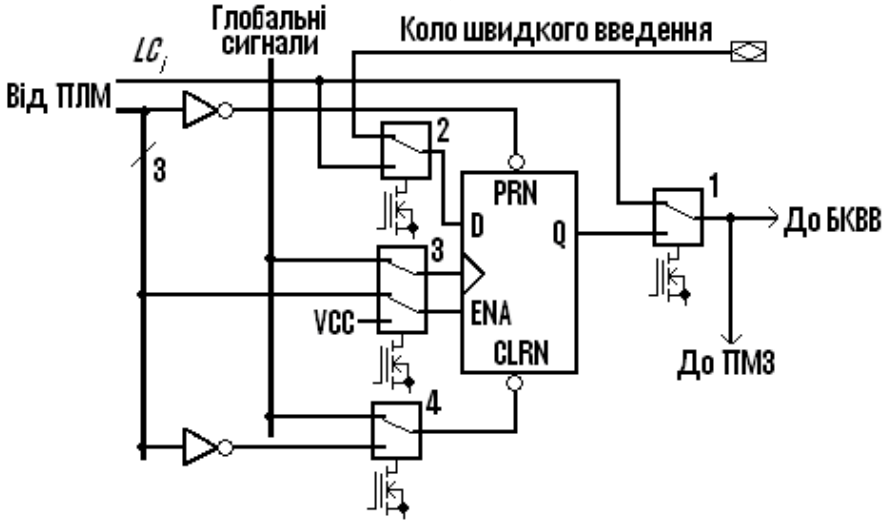
Для забезпечення гнучкості щодо формування функцій комірка має

два розширювача. Так званий розділюваний розширювач (РР) повертає один з термів через інвертор (див. рис. 9.8, б) до сигналів на входах ПЛМ і він стає доступним для всіх МК даного ЛБ. Паралельний розширювач (ПР) дозволяє вибрати диз'юнкцію з одного з виходів інших МК для з'єднання зі входом елемента АБО з метою утворення спрощених форм функцій з дужками (аналогічно як на рис. 9.5 диз'юнкція з виходу одного елемента АБО з'єднується зі входами інших елементів АБО).

а)



б)



в)

Рисунок 9.8

За допомогою елемента Виключне АБО досягається гнучкість керування вихідним сигналом МК, який у САПР позначається LC_i відповідно до номера i комірки. З огляду на те, що $LC_i = y_i \oplus c$, програмуванням $c = 0$ диз'юнкція y_i передається на вихід без зміни, при $c = 1$ інвертується, а при

з’єднанні *с* з термом, зокрема, вхідною змінною, утворюється сума за модулем два. Додатково з ПЛМ знімаються 3 терми для керування тригером.

Тригер, що є розрядом регістра, утворюваного комірками ЛБ, програмується мультиплексорами 1 ... 4, зображеними на рис. 9.8, в) у вигляді ключів. Записом до ЛІЗМОН-транзистора (див. рис. 9.2, д), пов’язаного з адресним входом мультиплексора, логічний 0 або 1, останній переводиться до потрібного положення. Так, при верхньому по схемі положенні мультиплексора 1 сигнал LC_i від ПЛМ надходить безпосередньо на вихід МК, звідки передається до ПМЗ та, якщо комірка є кінцевою, до блоку керування введенням/виведенням БКВВ. Через мультиплексор 2 цей сигнал може бути переданий на вхід D для записування в регістр; в іншому положенні мультиплексора 2 до регістра записується сигнал безпосередньо з контактної площинки ПЛІС, при цьому утворюється коло швидкого введення. Здвоєним мультиплексором 3 тригер може перемикатися на тактування глобальним сигналом GCLK з мінімальною затримкою від спільного входу ПЛІС або термом з виходу ПЛМ. При цьому під керуванням зазначеного терма по входу дозволу ENA може утворюватися тригер типу DE або з’єднанням ENA = VCC – тригер типу D. Зінвертованими термами від ПЛМ тригер керується по індивідуальних входах передустановлення PRN і скидання CLRN, але мультиплексором 4 можна перемкнути скидання від спільного для всіх розрядів глобального сигналу GCLRN, що характерно для регістрів. У деяких родинах ПЛІС можна керувати також типом тригерів – DE, TE, JKE або RSCE.

9.1.2.4 Блок керування введенням/виведенням

БКВВ складається з однакових керованих буферних каскадів БК1 ... БКN (рис. 9.9), кількість яких визначається числом N контактних площинок, закріплених за даним ЛБ (зазвичай N = 4 ... 16). Напрямок передавання даних програмується мультиплексором.

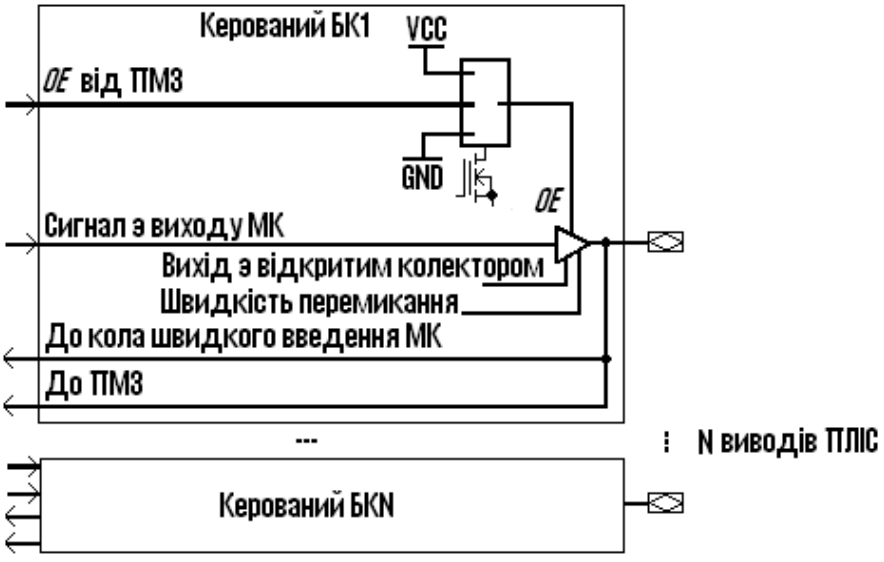


Рисунок 9.9

Якщо вхід дозволу *OE* програмуванням з'єднується із землею GND, буфер переводиться до третього стану і від'єднується від контакту, який стає входом: сигнал від нього спрямовується до буферів ПМЗ і є тепер приступним для всіх ЛБ, а також потрапляє до кола швидкого введення макрокомірки (див. рис. 9.8, в). При з'єднанні входу *OE* з напругою живлення VCC буфер передає (з підвищеною навантажувальною здатністю) сигнал з виходу МК до контакту, який стає тепер виходом ПЛІС. І, нарешті, за допомогою одного із шести глобальних сигналів від ПМЗ здійснюється оперативне керування входом *OE* в режимі входу/виходу, коли напрямок передавання даних може змінюватися (наприклад, спочатку від контактних площинок приймається адреса, а відтак на них передаються результати обробки інформації).

Крім того, програмуванням буфер можна перетворити в схему з відкритим колектором з метою гнучкості використання його виходу. Є також можливість регулювати швидкість перемикання (Slew Rate) буфера шляхом зміни тривалості фронтів вихідного сигналу. З двох швидкостей перемикання висока забезпечує максимальну швидкодію, але круті фронти перемикання спричиняють при цьому високий рівень завад у вихідній лінії. Тому, коли дозволяють міркування швидкодії, використовують режим положистих фронтів, який завжди запроваджується автоматично під час вмикання джерела живлення.

9.1.2.5 Особливості архітектури ПЛІС із тригерною пам'яттю конфігурації

З розвитком технології і зростанням ступеня інтеграції з'явилися ІС комбінованого типу, що поєднують в собі в себе властивості CPLD зі схемотехнікою запам'ятовувальних пристроїв. Прикладом є популярна ПЛІС родини FLEX 10K фірми Altera (FLEX – Flexible Logic Element Matrix – гнучка матриця логічних елементів), план розміщення якої (Floorplan) подано на рис. 9.10. Як і для ПЛІС типу CPLD (див. рис. 6,а), зв'язки між компонентами такої ПЛІС так само можна відображати на її плані графічно або логічними рівняннями.

Призначення (Assignments) для компонентів містяться в лотоках, пояснення яких винесено на плані вгорі. Крім логічних блоків ЛБ (Logic Array Block, LAB) до складу ІС входять також вбудовані блоки пам'яті ВБП (Embedded Array Block, EAB). Логічні блоки розташовано по рядках (Row) і колонках (Col), тому позначаються двокоординатною системою відліку, наприклад, відмічений внизу на плані блок має ім'я „B11", а блок над ним – ім'я „A11". ІС містить лише один стовпець блоків пам'яті (розташований всередині між стовпцями ЛБ), тому вони іменуються назвою рядка, наприклад, позначений внизу на плані блок має ім'я „EAB_B", а блок над ним – ім'я „EAB_A".

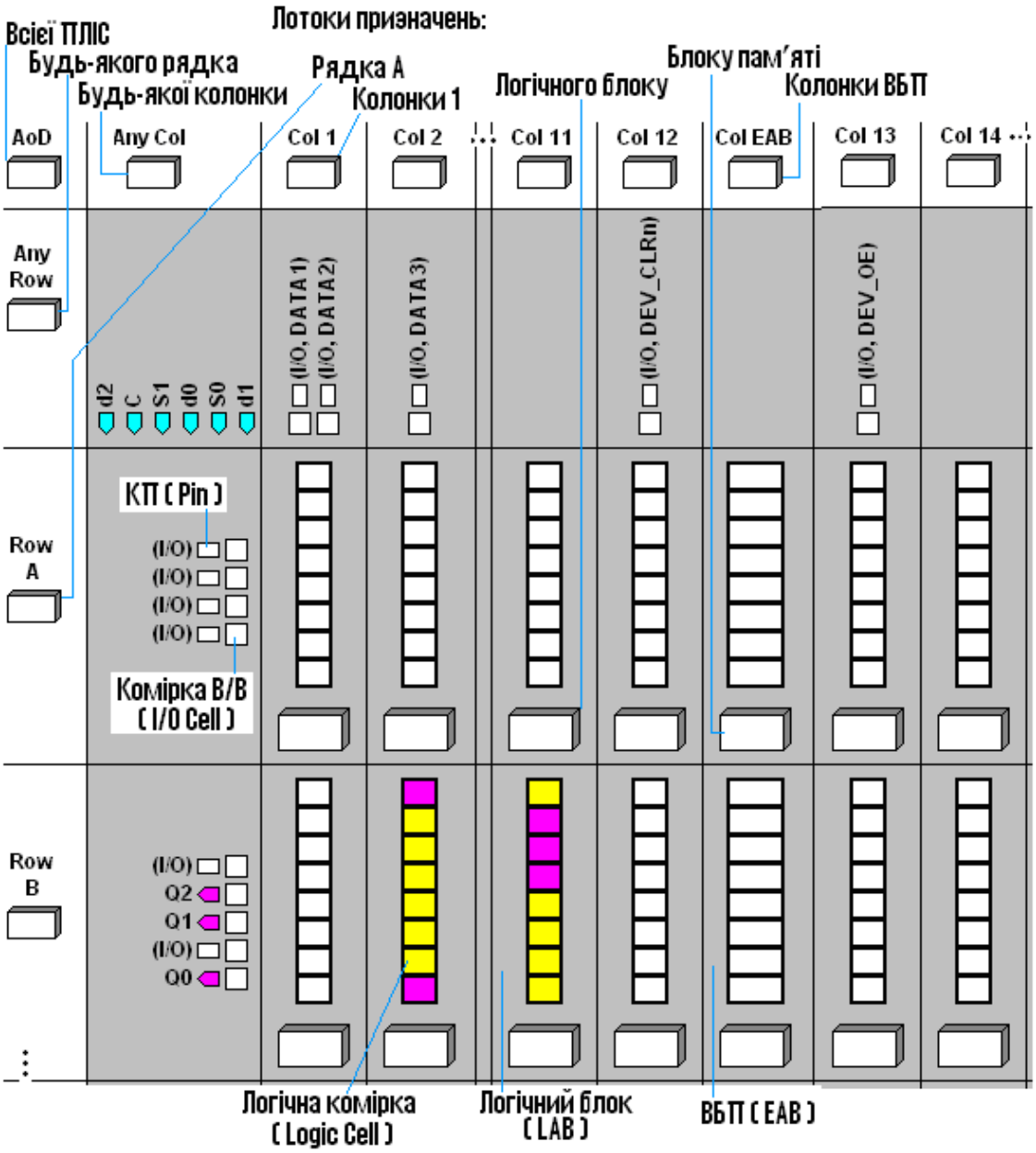


Рисунок 9.10

Наступною за ієрархією структурних одиниць є логічна комірка ЛК (Logic Cell, LC), яка в технічній документації для ПЛІС із тригерною пам'яттю конфігурації (SRAM-based) називається логічним елементом ЛЕ (Logic Element, LE). Такий „елемент" подібно до макрокомірки має логічну частину і програмований тригер, що є розрядом регістра, але відрізняється тим, що логічна функція реалізується програмуванням її таблиці відповідності в запам'ятовувальному пристрої (табличний спосіб реалізації). Крім того, ЛЕ містить коло переносу (для підвищення швидкодії схем типу лічильників і суматорів) та коло каскадування (для гнучкості формування функцій багатьох змінних кількома комірками). Кожний ЛБ розглядуваного типу ПЛІС містить 8 логічних комірок, які позначаються своїм номером у блоці та його координатами, наприклад, відмічена внизу на плані ЛК має ім'я „LC7_B2".

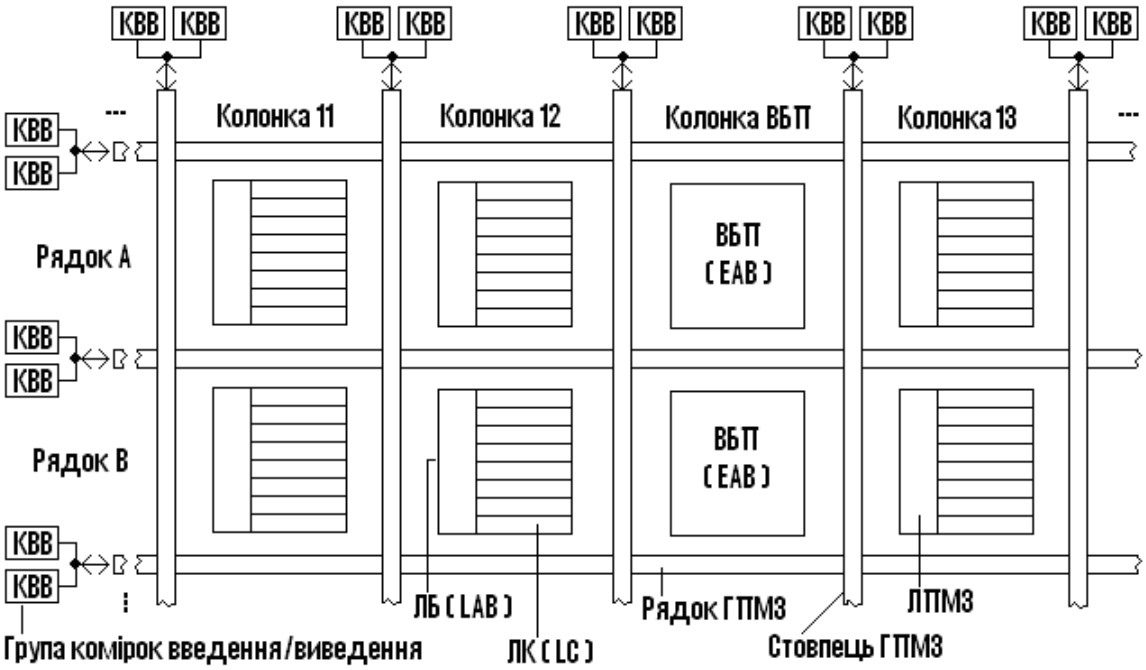


Рисунок 9.11

Виводи (контакти, Pin) мікросхеми, які так само поділяються на спеціалізовані і виводи користувача (глобальні входи, у разі потреби, можуть використовуватися як звичайні), розташовано на плані по периметру з усіх боків. При цьому будь-який вивід користувача (але не глобальні входи) може бути запрограмований як вхід, вихід або двоспрямований вхід/вихід пристрою за допомогою комірок введення/виведення КВВ (I/O Cell) біля кожного з них. Аналогічно блоку керування введенням/виведенням БВВ (див. рис. 9.9) у КВВ здійснюються функції підвищення навантажувальної здатності, перемикання напрямку передачі інформації, регулювання режимів буфера (швидкість перемикання, емуляція схеми з відкритим колектором). Крім того, КВВ містить вхідний і вихідний регістри для тимчасового зберігання даних та складну систему перемикання сигналів між шинами (більше десятка програмованих мультиплексорів та інших елементів).

MAX+PLUS II **Devices**

MAX 7000 EPLDs
Selection Guide

EPLD	Package	fCNT (MHz)	LCell (Reg.)	Ded. Inputs	I/O
EPM7128S	84L,100Q,160Q,100T	147.1	128	4	64,80,96,80
EPM7128S	84L,100Q,160Q,100T	125.0	128	4	64,80,96,80
EPM7128S	84L,100Q,160Q,100T	100.0	128	4	64,80,96,80
EPM7128S	84L,100Q,160Q,100T	76.9	128	4	64,80,96,80

FLEX 10K Devices
Selection Guide

Device	Package	LCells	Flipflops	Memory Bits	Ded. Inputs	I/O
EPF10K20	144T,208R,240R	1,152	1,344	12,288	6	96,141,183
EPF10K20	144T,208R,240R	1,152	1,344	12,288	6	96,141,183

Рисунок 9.12

Двокоординатна система позначень блоків відповідає дворівневій системі зв'язків між ними за допомогою глобальної програмованої матриці з'єднань ГПМЗ, поділеній на рядки і стовпці (рис. 9.11), з якими з'єднуються комірки вводу-виводу КВВ по всьому периметру ПЛІС. Через рядки і стовпці ГПМЗ, між якими розташовані ЛБ і ВБП, здійснюється обмін даними, а обмін між комірками організовано через локальну програмовану матрицю ЛПМЗ.

ПЛІС однієї родини, як і EPLD, однотипні і відрізняються лише складністю, що й визначає назву ІС (основні довідкові відомості для двох типів ПЛІС наведено на рис. 9.12). Так, мікросхема EPF10K10 родини FLEX 10K найменшої складності має логічну ємність 10 тис. еквівалентних вентилів (72 ЛБ, 576 ЛК, 3 ВБП обсягом 2048 бітів кожний), а ІС цієї родини EPF10K250 найбільшої (на цей час) складності має логічну ємність 250 тис. еквівалентних вентилів (1520 ЛБ, 12160 ЛК, 20 ВБП). Такі ПЛІС придатні для побудови складних пристроїв обробки інформації.

9.1.3 Програмування і конфігурування

9.1.3.1 Режими програмування і конфігурування

Як вже зазначалося (див. п. 9.1.1.2), структуру ПЛІС можна налаштовувати для реалізації проектів за програмувальними файлами у двох режимах – у режимі *програмування* (Program) або *конфігурування* (Configure). Режим програмування застосовується для мікросхем типу EPLD, виготовлених за технологією РПЗП-ЕС (EEPROM), у тому числі такому різновиду як флеш-пам'ять, та РПЗП-УФ (EPROM**)**. Режим конфігурування може застосовуватися для мікросхем, виготовлених за технологією СОЗП (SRAM). Перед перепрограмуванням ПЛІС, що базуються на технології РПЗП-УФ, потрібне попереднє стирання даних ультрафіолетовим промінням з вийманням її з пристрою.

Фізичне програмування або конфігурування виконується під дією згенерованих під час компіляції проекту програмувальних файлів, що надходять з комп'ютера. Залежно від установлених опцій програматора MAX+PLUS II в автоматичному режимі реалізуються такі функції: 1) перевірка наявності контакту між мікросхемою і відповідним з'єднувальним пристроєм; 2) перевірка на порожність (Blank-Check), тобто на відсутність перед програмуванням записаної до ІС інформації; 3) програмування (Program) або конфігурування (Configure), яке полягає в завантаженні до ІС даних з програмувального файла і налаштуванні її структури на виконання потрібних функцій; 4) верифікація (Verify), тобто перевірка ІС після програмування на відповідність записаної до неї інформації даним програмувального файла; 5) функціональне тестування запрограмованої ІС (Test) шляхом подачі на її входи комбінацій сигналів згідно із сигнальним файлом та порівняння вихідних сигналів на відповідність їх часовим діаграмам SCF-файла; 6) випробування (Examine), яке полягає в зчитуванні даних від

запрограмованої ІС (якщо вона не захищена спеціальним бітом захисту) з можливістю запису їх у власному файлі і подальшого використання, наприклад, з метою з'ясувати причини незадовільної роботи даної ІС або для програмування інших ІС.

9.1.3.2 Інтерфейс JTAG

Зручно мати можливість програмувати або конфігурувати мікросхему безпосередньо на її робочому місці без вилучення з монтажної плати пристрою, без фізичного доступу до всіх її виводів та без застосування спеціального апаратного програматора. Якщо під час роботи пристрою виявляться недоліки або виникне потреба в модернізації, мікросхему можна перепрограмувати. Така можливість, яка позначається терміном in-system programmability (ISP) – програмованість у системі, є в мікросхем, що містять комірки периферійного сканування BSC (Boundary-Scan Cells). Майже всі сучасні ІС провідних фірм-виробників мають властивості ISP. Об'єднаною групою по тестах Joint Test Action Group (JTAG) схему тестування BST було покладено в основу стандарту IEEE Std 1149.1, який визначає інтерфейс JTAG – сукупність засобів і порядок операцій для тестування ІС без фізичного доступу до кожного її зовнішнього виводу.

Ідея периферійного сканування ілюструється спрощеною схемою на рис. 9.13. Периферійні комірки BSC можуть працювати у двох режимах. У робочому режимі логічні комірки кристалу з'єднано із зовнішніми вхідними x_i та вихідними y_i виводами і BSC не змінюють функціонування пристрою.

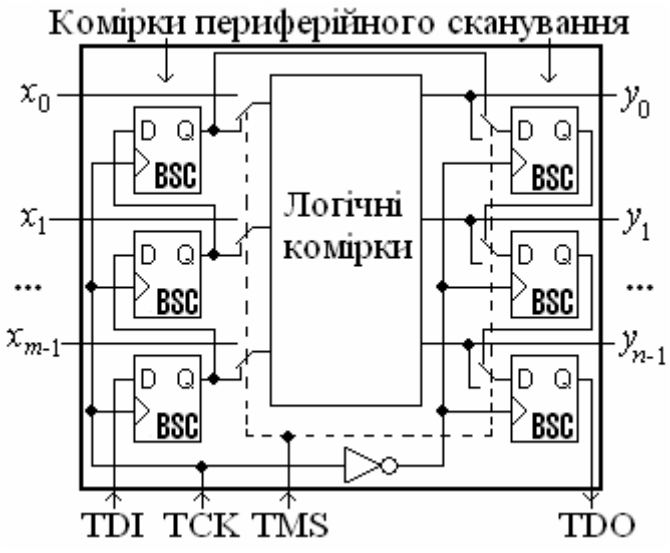


Рисунок 9.13

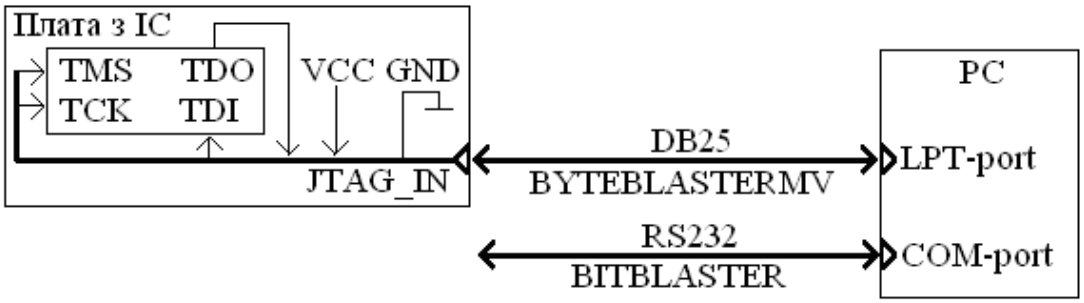
У режимі програмування ПЛІС функціонує послідовним виконанням команд за допомогою вбудованих контролерів JTAG і ISP (див. рис. 9.7). Сигналом керування TMS мультиплексорами (на рис. 9.13 показано перемикачами) зв'язки перемикаються: кристал від'єднується від зовнішніх входів, з комірок BSC утворюються вхідний і вихідний регістри зсуву, за допомогою яких здійснюється периферійне сканування. Вхідні тестові дані в послідовному коді надходять на вхід TDI послідовного введення вхідного регістра і за позитивними перепадами тактових імпульсів TCK відбувається послідовно-паралельне перетворення, після чого інформація з регістра подається в кристал у паралельному коді. Через входи паралельного введення вихідного регістра дані з

кристала записуються до вихідного регістра, після чого за негативними перепадами тактових імпульсів TCK відбувається паралельно-послідовне перетворення – з виходу послідовного виведення TDO знімаються вихідні тестові дані в послідовному коді.

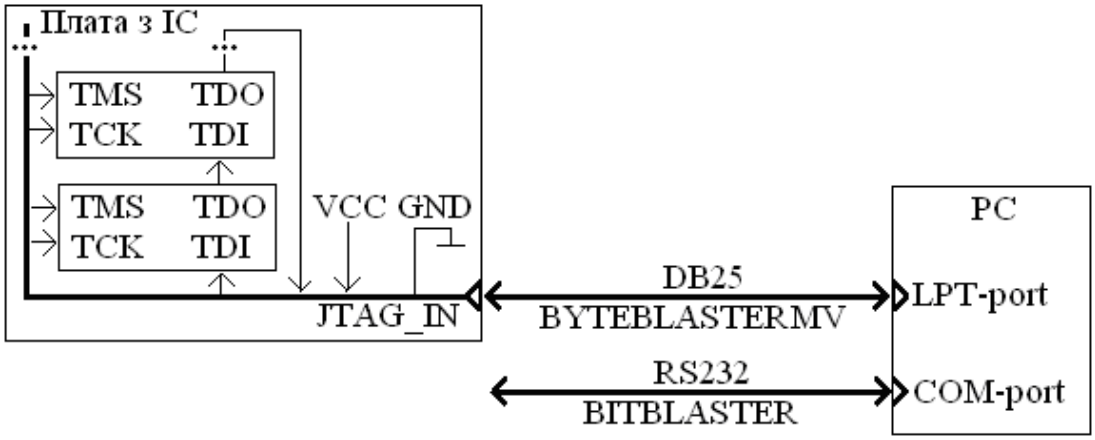
Така схема периферійного тестового сканування, що позначається абревіатурою BST (Boundary-Scan Test), використовується для перевірки наявності контактів на друкованій платі і на порожність ІС, її програмування або конфігурування в системі, верифікацію, функціональне тестування та випробування ІС. Для автоматичного виконання таких операцій програматор (програмний модуль MAX+PLUS II) формує тестовий код, який вводиться до комірок мікросхеми зі входу TDI. Відтак з виходу TDO зчитується результат і порівнюється залежно від операції, наприклад, з програмувальним або сигнальним файлом.

9.1.3.3 Схеми програмування і конфігурування

а) *Програмування однієї мікросхеми в системі.* Виводи ІС, призначені для програмування, мають бути з'єднані на монтажній платі зі стандартним 10-штирковим рознімачем 1DC10M інтерфейсу JTAG (рис. 9.14, а). За допомогою одного з інтерфейсних завантажувальних пристроїв цей рознімач з'єднується з комп'ютером PC, завдяки чому під час програмування або конфігурування відбувається обмін даними між програматором MAX+PLUS II і мікросхемою.



а)



б)

Рисунок 9.14

Поширеним є завантажувальний пристрій ByteBlasterMV, який є 10-жильним стрічковим кабелем DB25 з двома рознімачами на кінцях. Один з них з'єднується з рознімачем JTAG, розташованим на платі з мікросхемою, а другий через шинний формувач (який живиться від джерела VCC на платі з ІС і змонтований в корпусі рознімача), з'єднується з 25-контактним рознімачем паралельного LPT-порту комп'ютера. Програмування з ByteBlasterMV (ByteBlaster MultiVolt) може застосовуватися за напруг живлення VCC = 2,5 В; 3,3 В та 5 В, а різновид ByteBlaster є придатним лише за напруги 5 В. За значного (більше 1,2 *м*) віддалення плати з мікросхемою від комп'ютера використовують завантажувальні пристрої BitBlaster або MasterBlaster, що з'єднують стрічковим кабелем RS-232 рознімач JTAG на платі з послідовним портом комп'ютера RS-232 (порт COM).

Процедуру фізичного програмування однієї мікросхеми в системі з інтерфейсними завантажувальними пристроями BitBlaster, ByteBlasterMV або MasterBlaster, а також типи ІС (із підтримуваних даною версією пакету MAX+PLUS II), які можна програмувати в такий спосіб, наведено в розділі довідки Programming a Single Device with the BitBlaster, ByteBlasterMV, or MasterBlaster;

б) *Програмування ланцюжка мікросхем.* Стандарт IEEE Std 1149.1 інтерфейсу JTAG дозволяє поширити схему периферійного сканування BST на низку мікросхем, якщо послідовно ввімкнути їх тестові входи TDI і виходи TDO даних (рис. 9.14, б). Таке ввімкнення, що утворює ланцюжок JTAG (JTAG chain), застосовується для створення пристроїв та систем з кількох мікросхем. Схема з'єднань для програмування ланцюжка JTAG не відрізняється від схеми програмування однієї ІС (див. рис. 9.14, а).

Процедуру фізичного програмування ланцюжка мікросхем з інтерфейсними завантажувальними пристроями BitBlaster, ByteBlasterMV або MasterBlaster, а також типи ІС (із підтримуваних даною версією пакету MAX+PLUS II), які можна програмувати в такий спосіб, наведено в розділі довідки Programming Multiple Devices in a JTAG Chain with the BitBlaster, ByteBlasterMV, or MasterBlaster;

в) *Конфігурування мікросхем.* Схеми конфігурування однієї ІС в системі і кількох ІС ланцюжка JTAG аналогічні схемам програмування (див. рис. 9.13), відрізняються лише назви і кількість виводів ІС для конфігурування, з'єднаних на платі зі стандартним рознімачем.

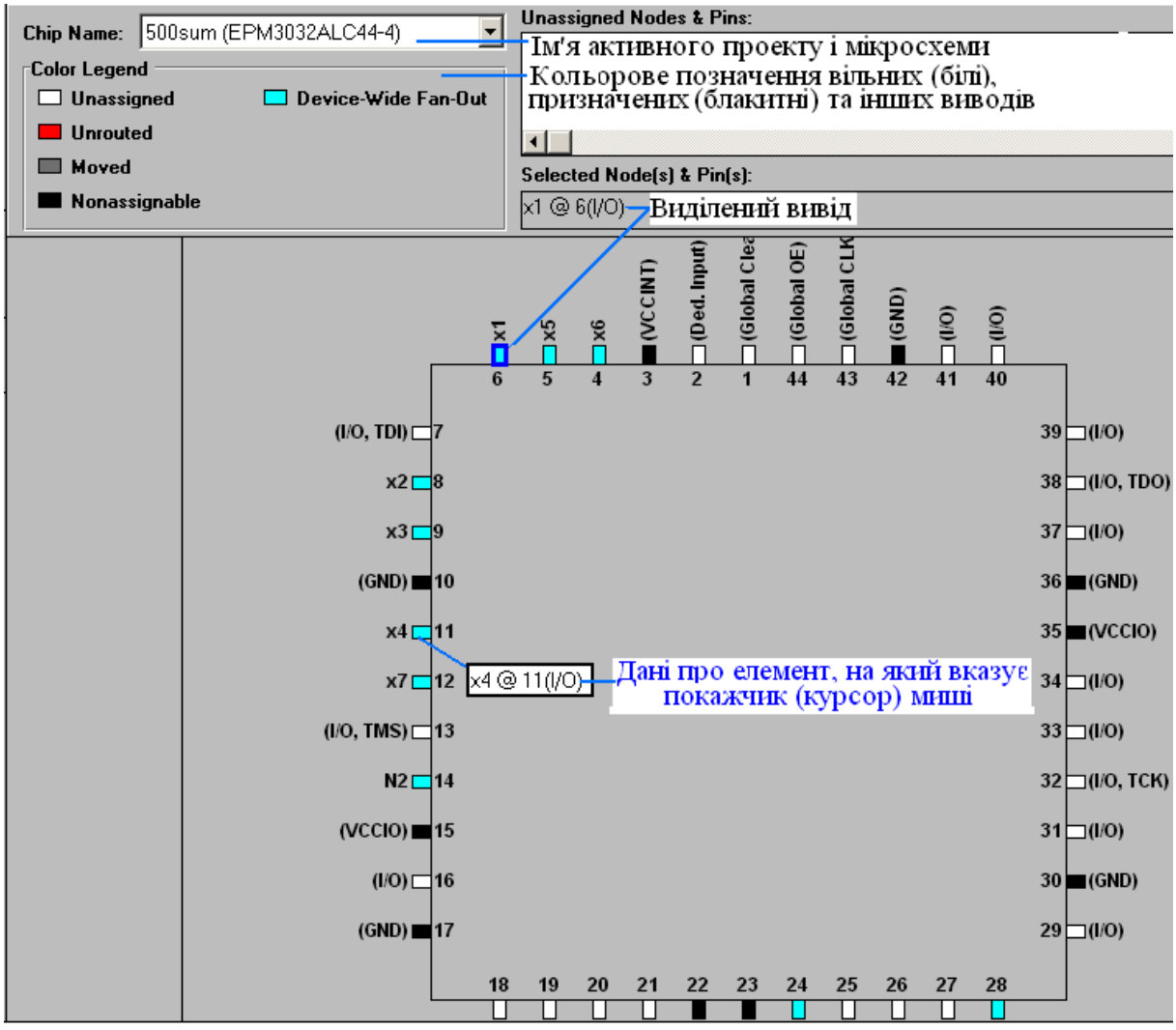
Процедуру конфігурування однієї ІС і ланцюжка мікросхем з інтерфейсними завантажувальними пристроями, а також типи ІС (із підтримуваних даною версією пакету MAX+PLUS II), які можна конфігурувати в такий спосіб, наведено в розділах довідки відповідно Configuring a Single Device with the BitBlaster, ByteBlasterMV, MasterBlaster, or FLEX Download Cable та Configuring Multiple Devices in a JTAG Chain with the BitBlaster, ByteBlasterMV, or MasterBlaster;

г) *Програмування мікросхеми апаратним програматором.* Мікросхему, зокрема, якщо її виконано без інтерфейсу JTAG, можна запрограмувати в окремому вигляді, використовуючи апаратний засіб програмування, а після того встановити на робочу плату пристрою або системи. У такий спосіб програмування здійснюється *базовим модулем програматора* Master Programming Unit (MPU) або Altera Programming Unit (APU – для Windows 98 та Windows 2000). Базовий модуль з'єднується з комп'ютером за допомогою плати програматора, а мікросхема з'єднується з MPU через програмувальний адаптер, завдяки чому забезпечується доступ до всіх зовнішніх виводів IC. Тип адаптера вибирається залежно від родини IC, типу її корпуса і кількості штирків, про що можна дізнатися з довідки Help > Devices & Adapters > Adapters.

9.2 Лабораторне завдання

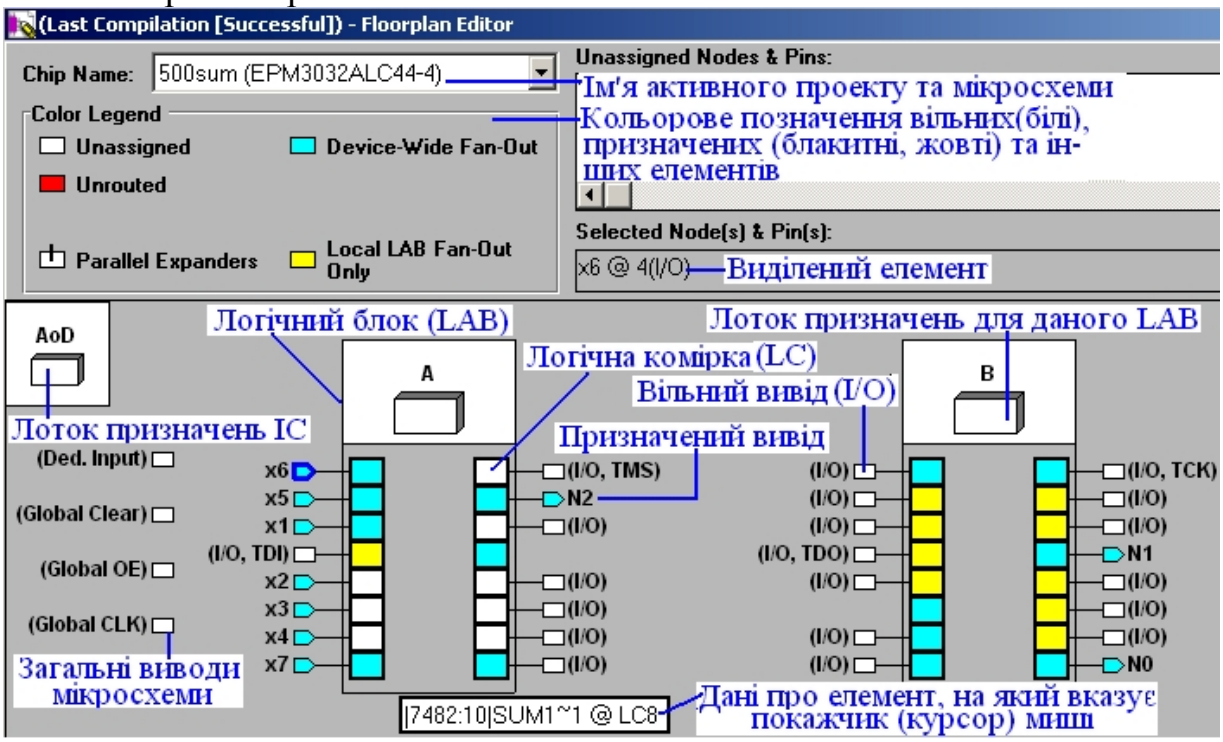
9.2.1 Засвоїти основи використання редактора фізичного розміщення Floorplan Editor

9.2.1.1 Активізувати потрібний проект, наприклад, 5lab\5XXsum (ввести до рядка заголовку Manager його назву), пікто-



грамою (або з меню MAX+plus II > Floorplan Editor) викликати редактор розміщення Floorplan Editor та переглянути результати реалізації проекту компілятором:

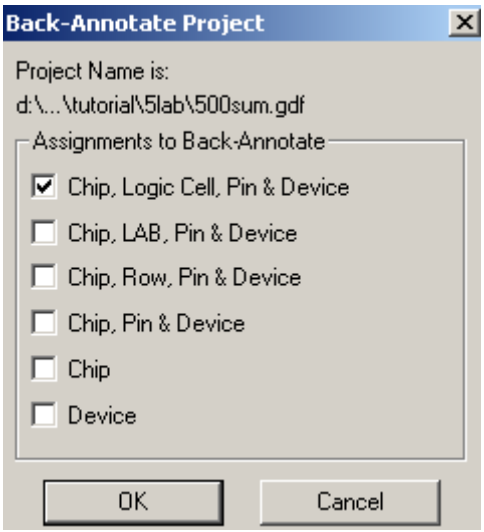
а) У разі потреби подвійним клацанням в робочому полі вікна Floorplan Editor або з меню Layout (розташування) > Device View (вигляд мікросхеми) викликати вікно Device View із загальним виглядом мікросхеми. Відтак натиснути на вертикальній палітрі інструмент відображення інформації про останню компіляцію (або в меню Layout виставити прапорець Last Compilation Floorplan) та ознайомитися з призначеною компілятором мікросхемою.



б) Подвійним клацанням всередині мікросхеми або з меню Layout (розташування) > LAB View (вигляд логічних блоків) викликати вікно з виглядом внутрішньої структури, в якому ознайомитися з логічними блоками (LAB) і їх комірками (LC), призначеними компілятором.

9.2.1.2 Відредагувати розміщення вхідних і вихідних контактів IC з метою зручного їх розташування на друкованій платі та перекомпілювати проект:

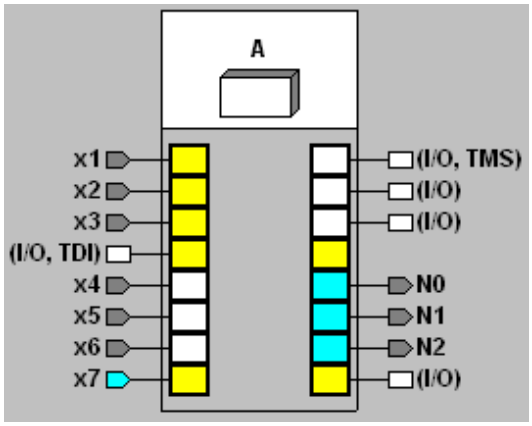
а) Щоб можна було відредагувати розміщення елементів проекту в мікросхемі, необхідно спочатку в меню Assign (призначення) ввімкнути команду Back-Annotate Project (зміна результатів компіляції проекту).



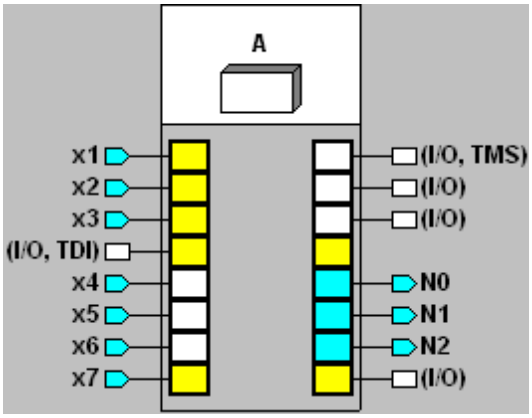
б) У діалоговому вікні Back-Annotate Project на вкладці Assignments to Back-Annotate (призначення для зміни) вибрати тип, що містить Pin (вивід, штирок), наприклад, Chip, Logic Cell, Pin & Device та натиснути ОК;

в) Інструментом вертикальної палітри (або з меню Layout > Current Assignments Floorplan) ввімкнути поточні (тобто нові) призначення заданих у пункті б) елементів розміщення.

Примітка. Якщо в проекті призначено не всі елементи, то редактор розміщення виведе список непризначених кіл і контактів у полі вікна Unassigned Nodes & Pins. Це поле можна використовувати також для тимчасового зберігання елементів під час їх переміщення.

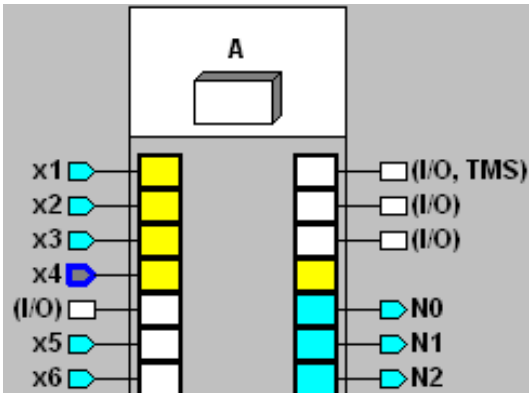


г) Виставити в меню Options прапорець Show Moved Nodes in Gray (показувати сірим переміщені вузли), після чого елемент Moved відобразиться в полі кольорових позначень вікна редактора;



д) Взятися за контакт, наприклад, x6 і пересунути його для тимчасового зберігання до поля Unassigned Nodes & Pins (непризначені кола і контакти) або на вільний вивід (I/O), відтак на його місце перетягнути вхід x1, який відобразиться сірим кольором. У такий самий спосіб впорядкувати розташування всіх входів (x1...x7) і виходів (N0…N2);

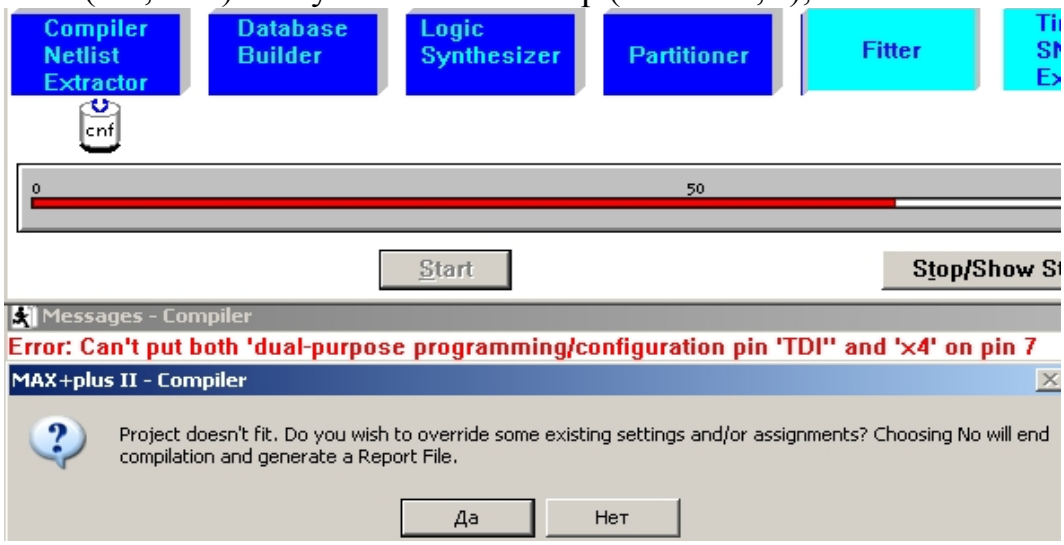
е) Виконати перекомпіляцію проекту: піктограмою (або з меню MAX+plus II > Compiler) відкрити вікно компілятора, натиснути в ньому кнопку Start, кнопку ОК у віконці з повідомленням про успішну компіляцію після її завершення та закрити вікно компілятора. Внаслідок цього перепризначені контакти набудуть блакитного кольору.



9.2.1.3 Виправити результати

неправильного призначення елементів у редакторі фізичного розміщення Floorplan Editor:

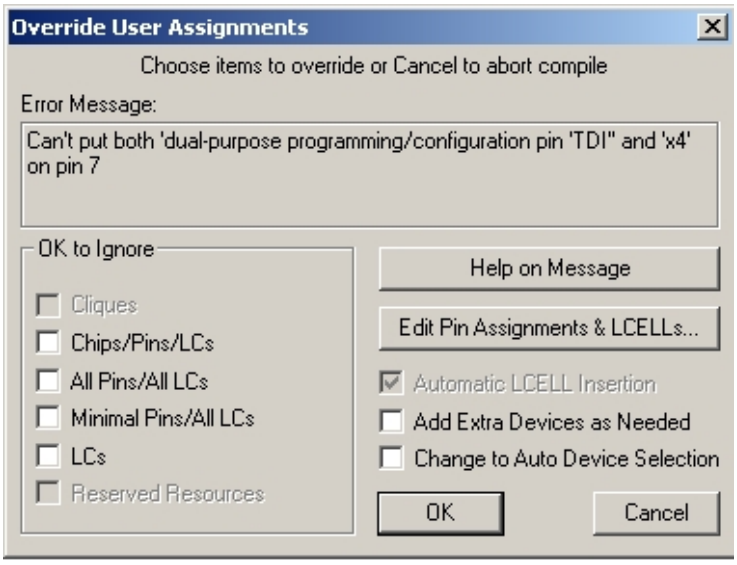
а) Виконати п. 9.2.1.2, а)...д), але перемістити, наприклад, вхід x4 на контакт 7 (I/O, TDI) і запустити компілятор (п. 9.2.1.2, е);



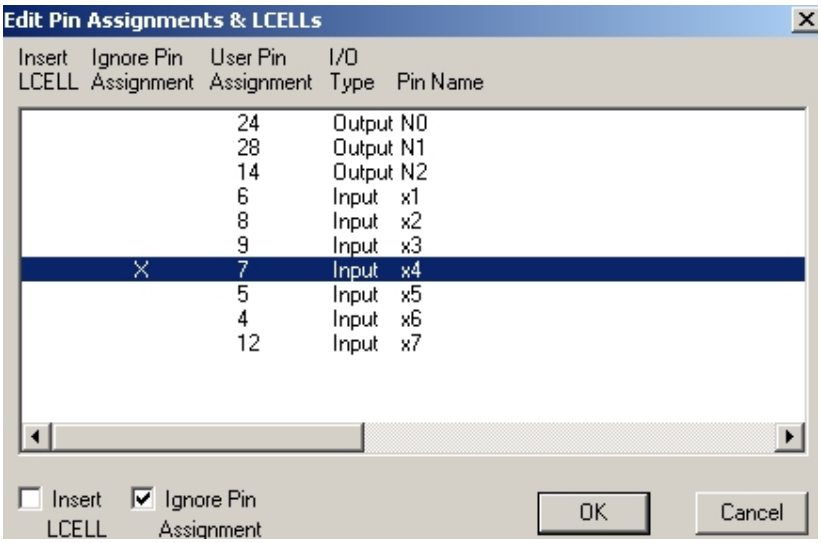
б) Після призупинення процесу компіляції у вікні процесора повідомлень з'явиться інформація про помилку, що не можна на вивід 7, призначений для програмування/ конфігурування, помістити сигнал x4. У віконці із запитом, чи не бажаєте Ви скасувати деякі налаштування і/або призначення слід дати негативну відповідь, закрити віконце запиту і вікно компілятора, пересунути вхід x4 на попередню або іншу припустиму позицію (на вільний контакт I/O) та знов скомпілювати проект.

☝ ***Примітка.*** Виправити неправильне призначення можна і в інший спосіб: виконати п. 9.2.1.3, а), б), але у віконці із запитом дати ствердну відповідь.

в) З'явиться діалогове вікно Override User Assignments (відхилити призначення користувача), в якому повторюється інформація процесора повідомлень про зміст помилки. У цьому вікні слід натиснути кнопку Edit Pin Assignments & LCELLs (редагувати призначення контактів і логічних комірок).



г) З'явиться нове діалогове вікно Edit Pin Assignments & LCELLs (редагувати призначення контактів і логічних комірок), в якому слід виділити рядок з неправильно призначеним



контактом для входу x4, ввімкнути опцію Ignore Pin Assignment (скасувати призначення контакту) і натиснути кнопку OK.

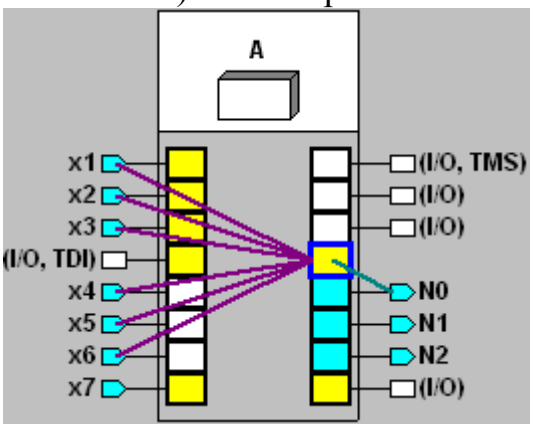
д) Натиснути кнопку OK у попередньому діалоговому вікні Override User Assignments, після чого процес компіляції завершиться. Відтак слід натиснути кнопку OK у віконці з повідомленням про успішну компіляцію і закрити вікно компілятора.

е) Натиснути на вертикальній палітрі інструмент відображення інформації про останню компіляцію (або в меню Layout виставити прапорець Last Compilation Floorplan) – вхід x4 повернеться на своє місце.

9.2.1.4 Переглянути логічні зв'язки між контактами і логічними комірками або лотоками призначень:

а) Виділити одну або декілька логічних комірок, натиснути інструмент палітри Displays the fan-in to the selected item(s) – відображати вхідні зв'язки виділен(ого/их) елемент(а/ів) – (або в меню Options вибрати опцію Show Node Fan-In – показати вхідні зв'язки кіл) і спостерігати логічні зв'язки у вигляді рожево-бузкових ліній;

б) Виділити один або декілька вхідних контактів, вимкнути інструмент Fan-In і натиснути інструмент палітри Displays the fan-out to the selected item(s) – відображати вихідні зв'язки виділен(ого/их) елемент(а/ів) – (або в меню Options вибрати опцію Show Node Fan-Out – показати вихідні зв'язки кіл) і спостерігати логічні зв'язки у вигляді зелених ліній. Так само виділити одну або декілька логічних комірок і спостерігати їх зв'язки з виходами;

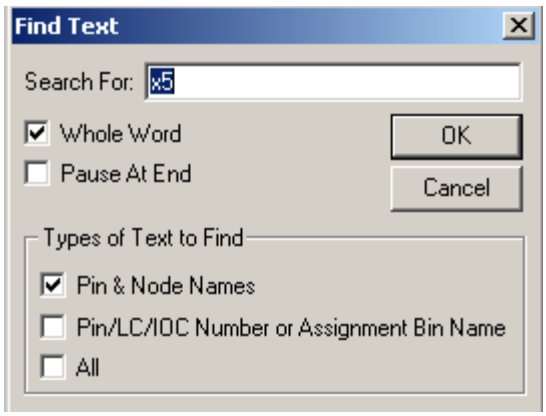


в) Виділити одну або декілька логічних комірок, ввімкнути обидва інструменти (Fan-In та Fan-Out) і спостерігати логічні зв'язки комір(ки/ок) зі входами і виходами (як подано на ілюстрації);

г) Виділити дві або декілька логічних комірок (утримувати клавішу Shift), натиснути інструмент палітри Displays the path between selected items

– відображати шлях між виділеними елементами – (або в меню Options вибрати опцію Show Path – показати шлях) і спостерігати логічні зв'язки у вигляді синіх ліній.

☝ ***Примітка.*** Команда Show Path дозволяє лише з'ясувати наявність або відсутність зв'язків між виділеними комірками; за її ввімкнення автоматично вимикаються дві інші команди (Fan-In та Fan-Out).



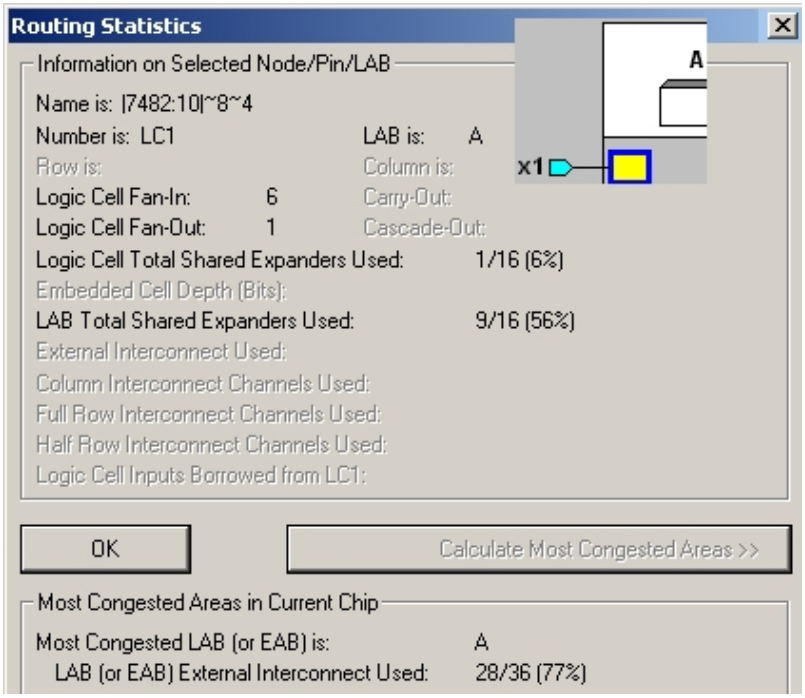
д) Піктограмою панелі інструментів (або з меню Utilities > Find Text) викликати діалогове вікно Find Text (знайти текст), у віконці Search For (пошук) ввести, наприклад, назву входу x5, на вкладці Types of Text to Find вибрати Pin & Node Names (імена контактів і кіл) та натиснути ОК. Після цього шуканий контакт буде виділений та частина плану складної мікросхеми з цим елементом пересунеться в поле зору вікна;

е) Аби з'ясувати зв'язок елемента на плані мікросхеми з проектними файлами слід виділити, наприклад, логічну комірку LC1 і натиснути піктограму панелі інструментів (або меню Utilities > Find Node in Design File > натиснути ОК у діалоговому віконці Find Node in Design File). Після цього з'явиться проектний файл із зазначеним елементом, наприклад, схемою суматора;

☝ ***Примітка.*** Якщо в проектному файлі імена входів або виходів інші, з'явиться повідомлення, що якийсь контакт не знайдено. Деякі компоненти проектного файла можуть бути відсутніми на плані мікросхеми, якщо компілятором було інакше синтезовано проект.

ж) Аби з'ясувати розміщення елемента проектного файла на плані мікросхеми, слід відкрити цей файл, наприклад, 500sum.gdf, виділити елемент і натиснути піктограму панелі інструментів (або меню Utilities > Find Node in Floorplan > натиснути ОК у діалоговому віконці Find Node in Floorplan). Після цього з'явиться план мікросхеми Floorplan з виділеними комірками або контактами, в яких розміщено заданий елемент.

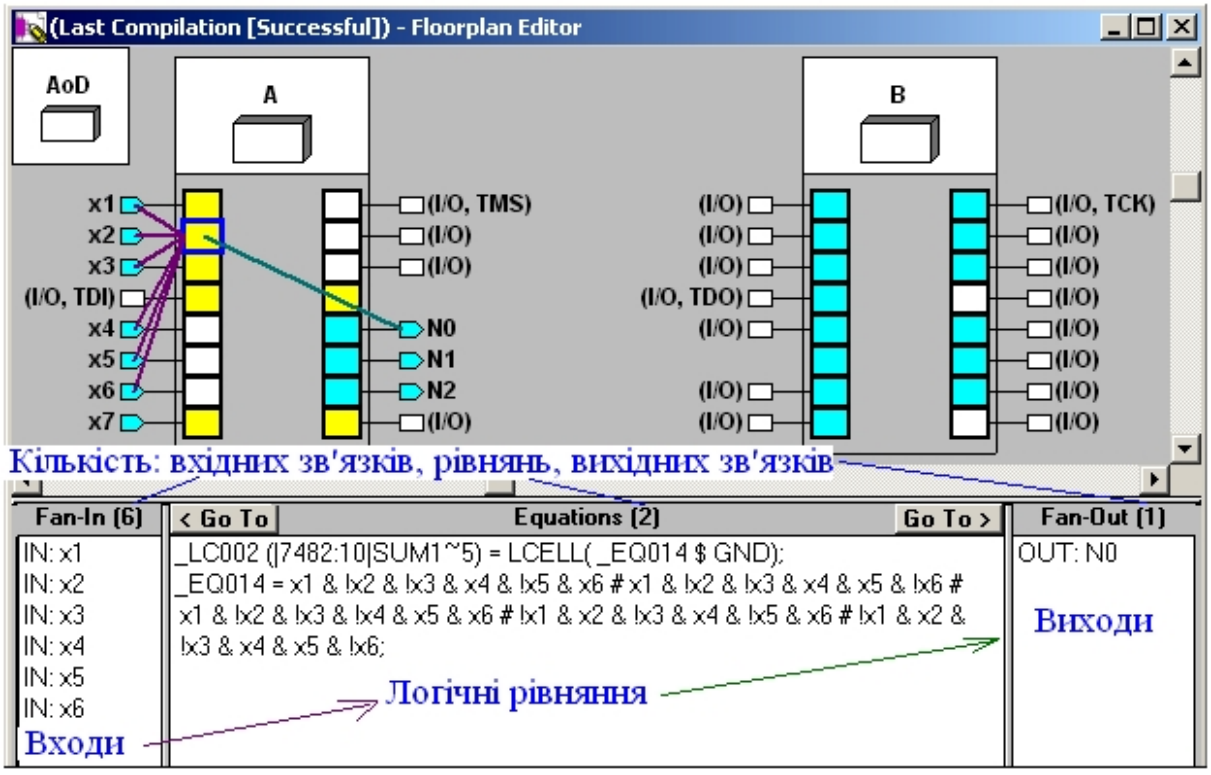
9.2.1.5 Переглянути інформацію про використання ресурсу мікросхеми. Для цього подвійним клацанням на комірці (показано фрагмент на ілюстрації), на контакті або лотоці призначень (або виділити один чи декілька з цих елементів і в меню Options вибрати опцію Routing Statistics – статистика маршрутизації) викликати вікно Routing Statistics з інформацією про використання ресурсу мікросхеми для з'єднання логічних елементів. Відтак натиснути кнопку Calculate Most Congested Areas (обчислити



область мікросхеми, в якій залишилось найменше ресурсу) і зчитати інформацію з вікна статистики.

9.2.1.6 Переглянути логічні рівняння і їх зв'язок з елементами плану мікросхеми:

а) У меню Layout (розташування) вибрати Report File Equation Viewer (вікно перегляду рівнянь звітного файла) та для зручності встановити в цьому ж меню Full Screen (повний екран). Внизу екрана з'явиться вікно для рівнянь;



б) Клацнути на логічній комірці – у вікні відобразяться сформовані в ній рівняння. Вікно Report File Equation Viewer (див. ілюстрацію вище) складається з трьох частин (ширину кожної з них можна змінити перетягуванням перегородки). Власне рівняння розташовано в центральній частині, у лівій – вхідні сигнали, з яких формуються терми рівнянь, а в правій – вихідні сигнали, для формування яких використовуються рівняння;

в) Інструментами палітри Displays the fan-in та Displays the fan-out (відображати вхідні та вихідні зв'язки) або з меню Options вибором опцій Show Node Fan-In та Show Node Fan-Out (показати вхідні та вихідні зв'язки) наочно відобразити кольоровими лініями на плані мікросхеми зв'язок логічних рівнянь з вхідними і вихідними сигналами;

г) Клацнути на плані мікросхеми вхідний контакт, наприклад, x1 – у полі рівнянь з'явиться його ім'я, у полі виходів – комірки і вихідні сигнали, в яких використовується цей вхід, а на плані ці зв'язки буде відображено зеленими лініями. Цю ж саму дію можна виконати, якщо у вікні рівнянь комірки за п. 9.2.1.6, б) двічі клацнути на імені входу в полі Fan-In (або клацнути і натиснути кнопку < Go To). Для повернення до рівнянь комірки слід натиснути кнопку Go To >;

д) Клацнути на плані мікросхеми вихідний контакт, наприклад, N0 – у полі Equations з'являться рівняння, що формують вихідний сигнал, у полі входів – комірки і вхідні сигнали, які використовуються в рівняннях, а на плані ці зв'язки буде відображено рожево-бузковими лініями. Цю ж саму дію можна виконати, якщо у вікні рівнянь комірки за п. 9.2.1.6, б) двічі клацнути ім'я виходу в полі Fan-Out (або клацнути і натиснути кнопку Go To >). Для повернення до рівнянь комірки слід натиснути кнопку < Go To.

9.2.2 Засвоїти основи фізичного програмування мікросхеми

9.2.2.1 Підімкнути до персонального комп'ютера апаратні засоби програмування, переконатися в правильності з'єднань.

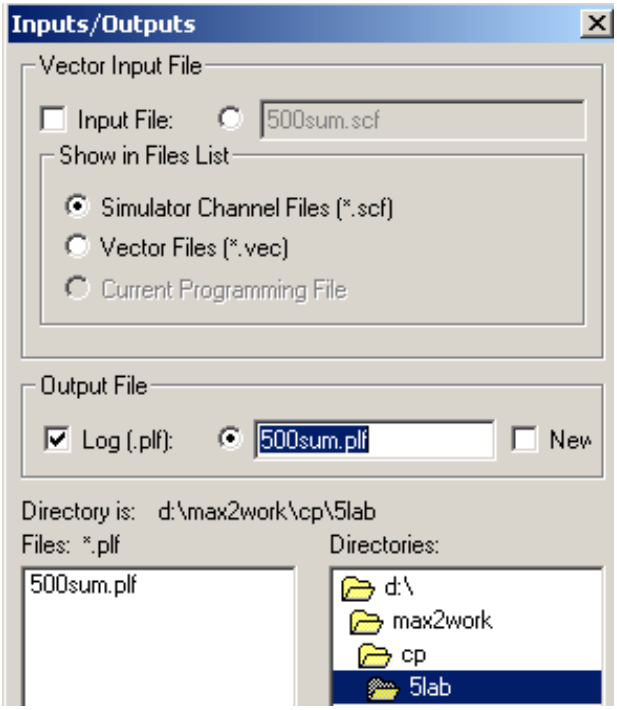
9.2.2.2 Відкрити вікно програматора:

а) Відкрити потрібний згідно із завданням проектний файл і надати проектові ім'я відповідно до імені поточного файла;

б) Піктограмою панелі інструментів (або з меню MAX+plus II > Programmer) викликати вікно програматора (Programmer);

в) Ознайомитися з органами керування та інформаційними полями вікна програматора (Programmer), усвідомити їх призначення.

9.2.2.3 Створити вихідний файл реєстрації програматора Programmer Log File (.plf):



а) З меню File > Inputs/Outputs (входи/виходи) викликати діалогове вікно Inputs/Outputs;

б) На вкладці Output File (вихідний файл) ввімкнути тип Log (.plf) – файл реєстрації – та його ім'я, яке вноситься до списку файлів Files;

в) Натиснути кнопку ОК у цьому вікні.

Примітка. У текстовому файлі звітності (.plf) записуються дії програматора. Потрібні вхідні файли (серед них і SCF-файл часових діаграм), якщо вони існують у проекті, завантажуються програматором автоматично для подальшого функціонального тестування ІС. Якщо PLF-файл існує, то за ввімкненої опції New він записується заново, а за вимкненої опції New він додається до вже існуючого.

9.2.2.4 Виконати фізичне програмування ІС

а) У разі потреби встановити режим програмування одної ІС: в меню JTAG вимкнути Multi-Device JTAG Chain (ланцюжок JTAG з декількох мікросхем) та в меню FLEX вимкнути Multi-Device FLEX Chain (ланцюжок з декількох мікросхем FLEX).

б) Для автоматичного виконання низки дій під час фізичного програмування викликати з меню Options діалогове вікно Programming Options (опції програмування) і встановити в ньому прапорці:

– Blank-Check Before Programming – перевірка ІС на відсутність записаної до неї інформації перед програмуванням (за вимкненої цієї опції

таку перевірку можна виконати вручну кнопкою Blank-Check у діалоговому вікні Programmer);

– Verify After Programming – верифікація (перевірка) ІС після програмування на відповідність записаної до неї інформації даним програмувального файла (за вимкненої цієї опції таку перевірку можна виконати вручну кнопкою Verify в діалоговому вікні Programmer); якщо верифікацію не виконувати, надійність не гарантується;

– Test After Programming – функціональне тестування ІС після програмування, яке полягає в завантаженні SCF-файла і перевірці вхідних і вихідних сигналів ІС на відповідність часовим діаграмам SCF-файла (за вимкненої цієї опції таку перевірку можна виконати вручну кнопкою Test у діалоговому вікні Programmer);

– Initiate Configuration After Programming – використовується під час конфігурування ІС;

в) За необхідності вставити відповідний адаптер в модуль MPU (або APU) та мікросхему в рознімач програмування (у „панельку”).

г) У вікні Programmer натиснути кнопку Program. Після цього програматор за ввімкнених відповідних опцій автоматично перевіряє ІС на відсутність записаної до неї інформації, програмує проект до мікросхеми, зчитує з неї записані дані і порівнює їх з даними програмного файла (.pof) та виконує функціональне тестування.

д) Переглянути текстовий файл звітності програматора (.plf).

Примітка. Для мікросхем з ультрафіолетовим стиранням даних (Classic, MAX 5000, MAX 7000) перед програмуванням (або якщо під час програмування була зупинка) вимагається стирання. Для інших мікросхем стирання не потрібно. Для мікросхеми ACEX 1K, FLEX 6000, FLEX 8000 та FLEX 10K використовується режим конфігурування, але не програмування.

9.2.3 Дослідити пристрій на запрограмованій ІС

9.2.3.1 Розробити методику експериментального дослідження пристрою на запрограмованій ІС.

9.2.3.2 Виконати необхідні експериментальні дослідження.

9.2.3.3 Порівняти результати експериментальних досліджень з даними проектних файлів, зробити висновки.

Контрольні питання та завдання

1. Чим відрізняються сучасні програмовані ПЛІС від мікропроцесорних ІС, робота яких базується на виконанні програми? Порівняйте можливості таких ІС щодо складності виконуваних функцій і швидкодії.

2. Схарактеризуйте відмінності щодо реконфігурування структури ПЛІС типу EPLD з електричним стиранням (EEPROM-based) і з пам'яттю конфігурації, основаній на СОЗУ (SRAM-based). Які елементи пам'яті конфігурації застосовуються для цих типів ПЛІС? Що розуміється в САПР під термінами „програмування" і „конфігурування"?

3. Зобразіть схему ПЛМ для реалізації одного з варіантів функції завдання 2 (див. додаток А, варіанти завдання 2).

4. Поясніть архітектуру ПЛІС типу EPLD: призначення сигналів і складників та їх взаємодію.

5. Поясніть архітектуру ПЛІС типу FLEX: призначення сигналів і складників та їх взаємодію.

6. Дайте поняття інтерфейсу JTAG і поясніть схеми програмування однієї ПЛІС та їх ланцюжка.

ЛІТЕРАТУРА

1. Кофанов В.Л. Математичні та схемотехнічні основи цифрових пристроїв: Навч. посібник. – Вінниця: «УНІВЕРСУМ-Вінниця», 2005. – 165 с.

2. Справочник по цифровой схемотехнике / В.И. Зубчук, В.П. Сигорский, А.Н. Шкуро – К.: Техніка, 1990. – 448 с.

3. Угрюмов Е.П. Цифровая схемотехника: Учеб. пособие. – СПб.: БХВ – Петербург, 2002. – 528 с.

4. Кофанов В.Л. Базовые элементы цифровых интегральных микросхем: Учеб. пособие. – К.: УМК ВО, 1988. – 116 с.

5. Ерофеев Ю.Н. Импульсные устройства: Учеб. пособие для вузов по спец. «Радиотехника». – М.: Высшая школа, 1989. – 527 с.

6. Калабеков Б.А. Цифровые устройства и микропроцессорные системы: Учебник для техникумов связи. – М.: Горячая линия – Телеком, 2002. – 336 с.

7. Радіотехніка: Енциклопедичний навчальний довідник: Навч. посібник / За ред. Ю.Л. Мазора, Є.А. Мачуського, В.І. Правди. – К.: Вища школа, 1999. – 838 с.

8. Рудик А.В. Радіоавтоматика. Частина 3. Дискретні та цифрові системи радіоавтоматики. Навчальний посібник. – Вінниця: ВНТУ, 2003. – 152 с.

9. University Program UP2 Education Kit. – Altera Corporation, v. 3.1, 2004.

Додаток А
Індивідуальні завдання до лабораторних робіт

Варіанти завдання 1

Наведіть для кожної схеми (а...п) заданого рисунку значення константи або умовне позначення еквівалентного логічного елемента, якщо елемент *А* виконує таку логічну функцію (далі наводиться варіант – рисунок – функція): 1) А1 – АБО, 2) А1 – І, 3) А1 – Виключне АБО, 4) А1 – Заборона, 5) А1 – АБО-НЕ, 6) А1 – І-НЕ, 7) А1 – Виключне АБО-НЕ, 8) А1 – Імплікація, 9) А2 – АБО, 10) А2 – І, 11) А2 – Виключне АБО, 12) А2 – Заборона, 13) А2 – АБО-НЕ, 14) А2 – І-НЕ, 15) А2 – Виключне АБО-НЕ, 16) А2 – Імплікація.

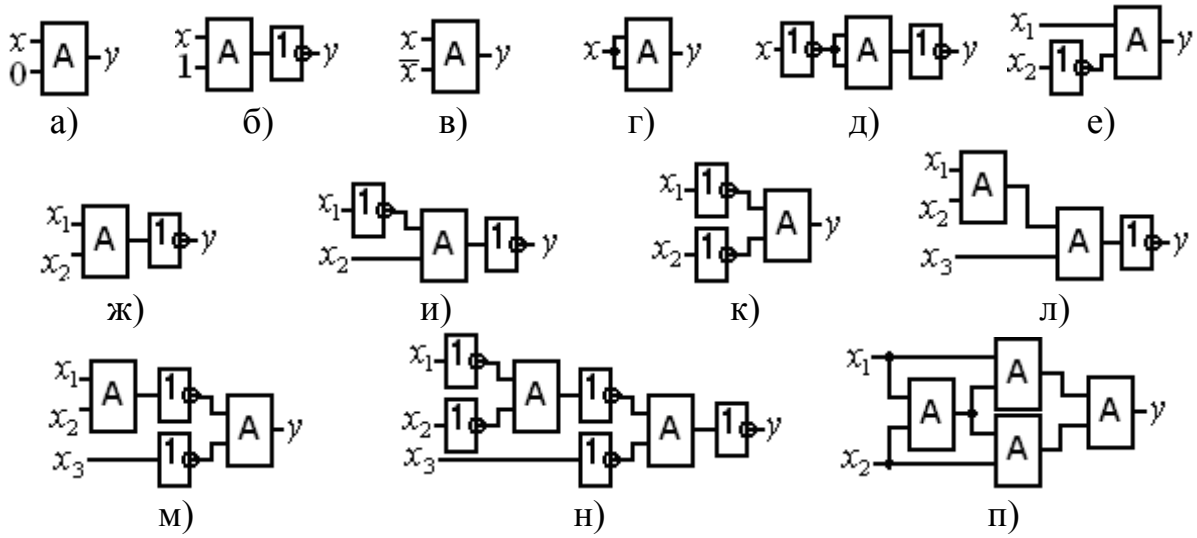


Рисунок А1

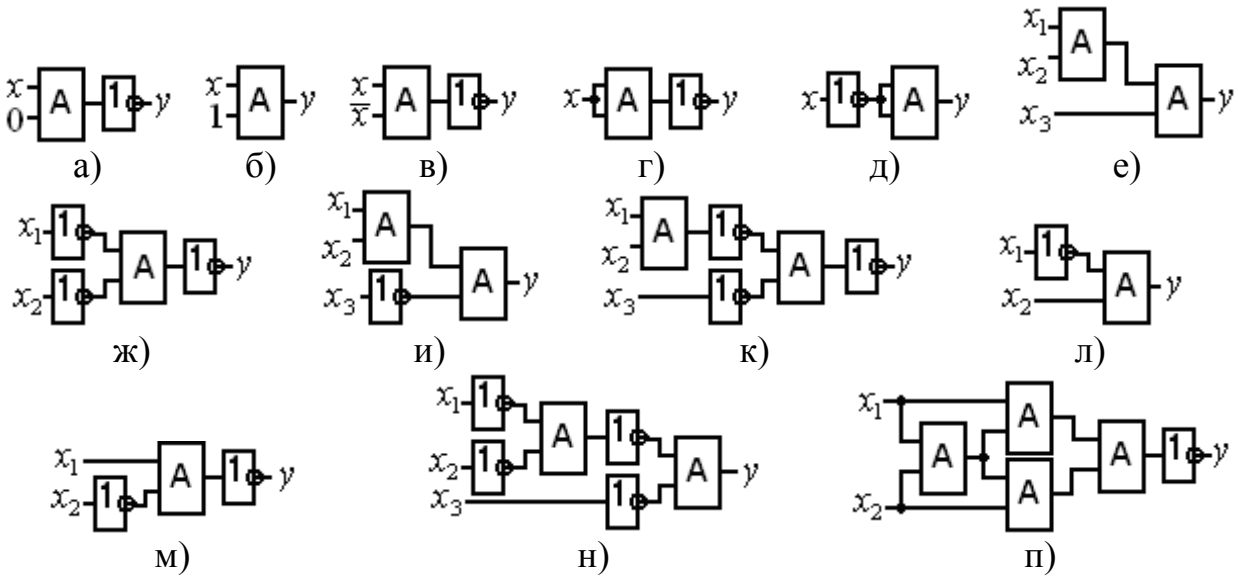


Рисунок А2

Варіанти завдання 2

Для заданої згідно з варіантом логічної функції *y*: *а)* побудувати таблицю відповідності і діаграму термів (Вайча-Карно); *б)* навести досконалі (ДДНФ або ДКНФ) та мінімальні (МДНФ і МКНФ) форми; *в)* перетворити мінімальні форми до базисів І-НЕ, АБО-НЕ, І-АБО-НЕ; *г)* мінімізувати схему в базисі І-НЕ, АБО-НЕ чи мішаному, який забезпечує меншу складність; *д)* навести схеми за п. *в), г)*; *е)* записати задану логічну функцію (без перетворень) та вирази за п. *в), г)* мовою опису апаратури AHDL.

1) $y = (1 \setminus x_3) \oplus (x_1 \to \overline{x_2})$;

2) $y = 1 \setminus [x_2 \oplus (\overline{x_3} \to x_1)]$;

3) $y = \overline{x_3 \setminus x_2} \to (\overline{x_1} \oplus \overline{x_3}) + (x_2 \setminus x_1)$;

4) $y = 1 \setminus [(x_1 \to x_2 x_3) \to (x_2 \oplus x_3)]$;

5) $y = 1 \setminus [(x_2 + x_4) \to x_3 \overline{x_1 \oplus x_3}]$;

6) $y = 1 \setminus [(\overline{x_1} \to x_4) \setminus (x_3 \overline{x_2 \oplus x_3})]$;

7) $y = 1 \setminus [(x_1 + x_2 + \overline{x_3}) \to x_3 \overline{x_2 \oplus x_4} \setminus (\overline{x_1} x_2 x_3)]$;

8) $y = (x_1 + \overline{x_2 x_3}) \to \overline{x_1 + (x_2 \oplus x_4)} \setminus (\overline{x_2} x_3)$;

9) $y = (x_2 + \overline{x_1 x_3 + x_4 x_5}) \to (x_4 x_5 \setminus \overline{x_1 \oplus x_2})$;

10) $y = 1 \setminus [(x_1 + x_2 + \overline{x_3 \oplus x_4}\ \overline{x_3}) \to x_4 \overline{x_3 + x_5}]$.

Варіанти завдання 3

а) Записати апаратною мовою AHDL логічну функцію (первісний вираз без перетворень) свого варіанта завдання 2.

б) Спроектувати на основі двійкового дешифратора перетворювач (дешифратор) до семисегментного коду для індикації знаків:

1) E, r, A; 2) d, o, b, A; 3) r, A, d, I, o; 4) A, b, C, d, E, F; 5) P, O, L, E, H, A, C; 6) b, L, o, c, A, h, r, E; 7) C, U, r, d, O, H, L, E, b; 8) 0…9; 9) 0...7; 10) H, E, r, c.

в) Побудувати дешифратор-демультиплексор (двійковий):

Варіант*	1	2	3	4	5	6	7	8	9	10
Розрядність	4:9	3:5	4:10	3:6	4:11	3:7	4:12	3:8	4:13	4:14

* Варіанти 11...20 – те саме, але з інверсними виходами.

Варіанти завдання 4

Спроектувати ЦКП для реалізації логічної функції, заданої згідно з варіантом завдання 2, на мультиплексорах різної розрядності та вибрати оптимальний варіант.

Варіанти завдання 5

Побудувати пороговий елемент на суматорах і компараторах:

1) 2 з 5, 2) 3 з 6, 3) 3 з 7, 4) 4 з 8, 5) 3 з 5, 6) 4 з 6, 7) 3 з 8, 8) 4 з 9, 9) 4 з 5, 10) 5 з 8, 11) 6 з 9, 12) 7 з 10.

Варіанти завдання 6

За до́звільного рівня *Enable* синхроімпульсом *Clock* записати до тригерів дані і сформувати на їхніх виходах логічну функцію, задану варіантом завдання 2, за власним вибором базису.

Варіанти завдання 7

Спроектувати на основі регістра зсуву ЦПП (далі наводиться тип пристрою – варіант – дані):

ГКП реверсивний – 1) N = (0000101); 5) M = 5; 9) N = (00101);

PIP – 2) N = (01111); 6) M = 7; 10) M = 9;

ГКП– 3) M = 7; 7) N = (01011); 11) N = (0001101);

PIP реверсивний – 4) M = 5; 8) N = (00011); 12) M = 6.

Варіанти завдання 8

Спроектувати згідно з варіантом:

а) недвійковий лічильник із заданим модулем *M* і звичайним порядком лічби шляхом перетворення двійкового лічильника (далі наводиться варіант – модуль): 1) *M* =3, 4, 5 – програмований; 2) *M* = 15; 3) *M* = 6, 7, 8 - програмований; 4) *M* = 14; 5) *M* = 5, 10 – програмований; 6) *M* = 13; 7) *M* = 9, 10 – програмований; 8) *M* = 12; 9) *M* = 10, 15 - програмований; 10) *M* = 11;

б) недвійковий лічильник із заданим модулем *M* і звичайним порядком лічби на тригерах зі зворотними зв’язками (далі наводиться варіант – модуль – тип тригерів): 1) *M* = 11 – D; 2) *M* = 7 – JK – реверсивний; 3) *M* = 5 – RSC; 4) *M* = 7 – D – реверсивний; 5) *M* = 9 – TE; 6) *M* = 6 – RSC – реверсивний; 7) *M* =10 – JK; 8) *M* = 6 – TE – реверсивний; 9) *M* = 9 – JK; 10) *M* = 5 – D – реверсивний;

в) безвентильний подільник частоти на JK-тригерах (далі наводиться варіант – модуль): 1) *M* = 6; 2) *M* = 7; 3) *M* = 9; 4) *M* = 10; 5) *M* = 14; 6) *M* = 12; 7) *M* = 18; 8) *M* = 20; 9) *M* = 15; 10) *M* = 17.

Варіанти завдання 9

Виконати фізичне програмування ПЛІС EPM7128S для реалізації функції згідно з варіантом завдання 3, б).

Додаток Б
Таблиці до опису лабораторного стенда

Таблиця Б1 – Дублювальні роз'єми мікросхеми EPM7128S

P1		P2		P3		P4	
Зовнішній	Внутрішній	Зовнішній	Внутрішній	Зовнішній	Внутрішній	Зовнішній	Внутрішній
75	76	12	13	33	34	54	55
77	78	14	15	35	36	56	57
79	80	16	17	37	38	58	59
81	82	18	19	39	40	60	61
83	84	20	21	41	42	62	63
1	2	22	23	43	44	64	65
3	4	24	25	45	46	66	67
5	6	26	27	47	48	68	69
7	8	28	29	49	50	70	71
9	10	30	31	51	52	72	73
11	X	32	X	53	X	74	X

Таблиця Б2 – Контакти рознімача JTAG_IN

Контакт	Сигнал JTAG	Призначення
1	TCK	Тактові імпульси
2	GND	Земля
3	TDO	Вихід даних
4	VCC	Напруга живлення (+)
5	TMS	Керування автоматом JTAG
6	Не задіяний	–
7	Не задіяний	–
8	Не задіяний	–
9	TDI	Вхід даних
10	GND	Земля

Таблиця Б3 – Режими програмування

Режим	TDI	TDO	DEVICE	BOARD
Програмування мікросхеми	C1&C2	C1&C2	C1&C2	C1&C2
Конфігурування мікросхеми	C2&C3	C2&C3	C1&C2	C1&C2
Сумісне конфігурування двох мікросхем	C2&C3	C1&C2	C2&C3	C1&C2
Об'єднання кількох плат UP2	C2&C3	Відкритий	C2&C3	C2&C3

Таблиця Б4 – Семисегментні індикатори мікросхеми EPM7128S

Сегмент	**Контакти мікросхеми, приєднані до першого знакомісця**	**Контакти мікросхеми, приєднані до другого знакомісця**
a	58	69
b	60	70
c	61	73
d	63	74
e	64	76
f	65	75
g	67	77
DP (знак десяткового дробу)	68	79

Навчальне видання

Віктор Леонідович Кофанов
Олександр Володимирович Осадчук
Дмитро Володимирович Гаврілов

ЛАБОРАТОРНИЙ ПРАКТИКУМ З ДОСЛІДЖЕННЯ ЦИФРОВИХ ПРИСТРОЇВ НА ОСНОВІ САПР MAX+PLUS II

Навчальний посібник

Оригінал-макет підготовлено Гавріловим Д.В.

Редактор Т.О. Старічек

Науково-методичний відділ ВНТУ
Свідоцтво Держкомінформу України
серія ДК № 746 від 25.12.2001
21021, м. Вінниця, Хмельницьке шосе, 95, ВНТУ

Підписано до друку
Формат 29,7х42 1/4
Друк різографічний
Тираж прим.
Зам. №

Гарнітура Times New Roman
Папір офсетний
Ум. друк. арк.

Віддруковано в комп'ютерному інформаційно-видавничому центрі
Вінницького національного технічного університету
Свідоцтво Держкомінформу України
серія ДК № 746 від 25.12.2001
21021, м. Вінниця, Хмельницьке шосе, 95, ВНТУ