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

ЛАБОРАТОРНИЙ ПРАКТИКУМ З ЦИФРОВИХ ПРИСТРОЇВ НА ОСНОВІ САПР QUARTUS II

Міністерство освіти і науки України
Вінницький національний технічний університет

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

ЛАБОРАТОРНИЙ ПРАКТИКУМ З ЦИФРОВИХ ПРИСТРОЇВ НА ОСНОВІ САПР QUARTUS II

Затверджено Вченою радою Вінницького національного технічного університету як навчальний посібник для студентів напряму підготовки 0907 – Радіотехніка всіх спеціальностей. Протокол № ___ від "___" _____ 2007 р.

Вінниця ВНТУ 2007

УДК 621.374
К 74

Рецензенти:

О.Д. Азаров, доктор технічних наук професор
С.М. Злепко, доктор технічних наук професор
С.М. Маєвський, доктор технічних наук професор

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

Кофанов В. Л., Осадчук О. В., Гаврілов Д. В.
К 74 **Лабораторний практикум з цифрових пристроїв на основі САПР Quartus II.** Навчальний посібник. – Вінниця: УНІВЕРСУМ-Вінниця, 2007. – 189? с.

Практикум поєднує виконання практичних завдань і лабораторних досліджень цифрових пристроїв (ЦП) на сучасних програмованих ІС (ПЛІС). Одночасно з вивченням ЦП на ПЛІС і традиційній елементній базі передбачено опанування основами повноциклової САПР Quartus II. З метою активізації навчального процесу завдання включає створення діючого макету для експериментального дослідження певного типу ЦП згідно з варіантом індивідуального завдання.

Змістом посібника є типові ЦП, що охоплюють відповідні розділи дисциплін „Цифрові пристрої та мікропроцесори”, „Проектування цифрових пристроїв”, „Елементна база цифрових систем зв’язку”. Матеріал є базою також для курсового і дипломного проектування з можливістю виготовлення експериментального макету.

Для студентів бакалаврського ступеня підготовки радіоелектронних спеціальностей вищих закладів освіти.

УДК 621.374

ЗМІСТ

ВСТУП

Посібник охоплює матеріал для виконання практичних завдань і лабораторних досліджень. Засвоєння основ цифрових пристроїв (ЦП) здійснюється на базі надвеликих інтегрованих мікросхем програмованої структури (для стислості позначатимемо їх ПЛІС – програмовані логічні ІС) відповідно до останніх досягнень мікроелектроніки. Проте одночасно вивчаються також типові ЦП традиційної елементної бази, які зручно запроваджувати до ПЛІС у вигляді стандартних структурних компонентів. Крім того, метою практикуму є набуття студентами навичок сучасної інженерної праці, коли левова частка розробки і налагодження нової радіоелектронної апаратури (РЕА) та діагностики під час експлуатації діючої РЕА має припадати на системи автоматизованого проектування (САПР).

Через складність ПЛІС (в одному кристалі міститься від тисяч до мільйонів еквівалентних вентилів типу двовходових логічних елементів) застосовуються нові автоматизовані методи синтезу цифрових структур і комплексні засоби проектування РЕА на персональних комп'ютерах. Зокрема, для ПЛІС фірми Altera (США), які часто розглядаються як стандарти нової елементної бази, впроваджено декілька пакетів САПР, версії яких постійно оновлюються. Хоч САПР різних фірм мають відмінності, методика проектування на їх основі є багато в чому спільною. У цьому посібнику користуватимемося САПР Quartus II, яка підтримує університетську освіту і пристосована для нових та розроблених раніше ПЛІС фірми Altera і деяких інших фірм. На відміну від попередніх САПР (типу CAD – computer aided design) така система є *повноцикловою* (типу EDA – electronic design automation). Це означає, що вона являє собою закінчене проектне середовище для реалізації виробу типу «система на програмованому кристалі» (SOPC – system-on-a-programmable-chip), тобто забезпечує всі етапи проектування – від введення проекту до моделювання і фізичного програмування ПЛІС та тестування результатів.

У САПР *проектом* прийнято називати комплект файлів, створених користувачем і програмним забезпеченням для досягнення поставленої мети. Етапи проектування за допомогою САПР можна узагальнити як у спрощеному вигляді подано на рис. В1.

Введення проекту здійснюється інструментальними засобами у традиційній для інженера графічній формі (принципова електрична схема) або в текстовій формі (програма апаратною мовою високого рівня) з використанням бібліотеки бази даних – стандартних компонентів САПР або створених користувачем і згорнутих до символу блоків. Для побудови РЕА найдоцільнішою є ієрархічна структура, за якою проект вводиться на рівні блок-схеми в графічному редакторі (проект верхнього рівня), а окремі блоки подаються у текстовому або графічному редакторі (проекти нижніх рівнів). Складні проекти, компонентами яких можуть бути

мікропроцесори, блоки пам'яті і т. ін., створюються спеціальними засобами САПР на рівні системи. Результатом введення проекту є низка сформованих проектних файлів.

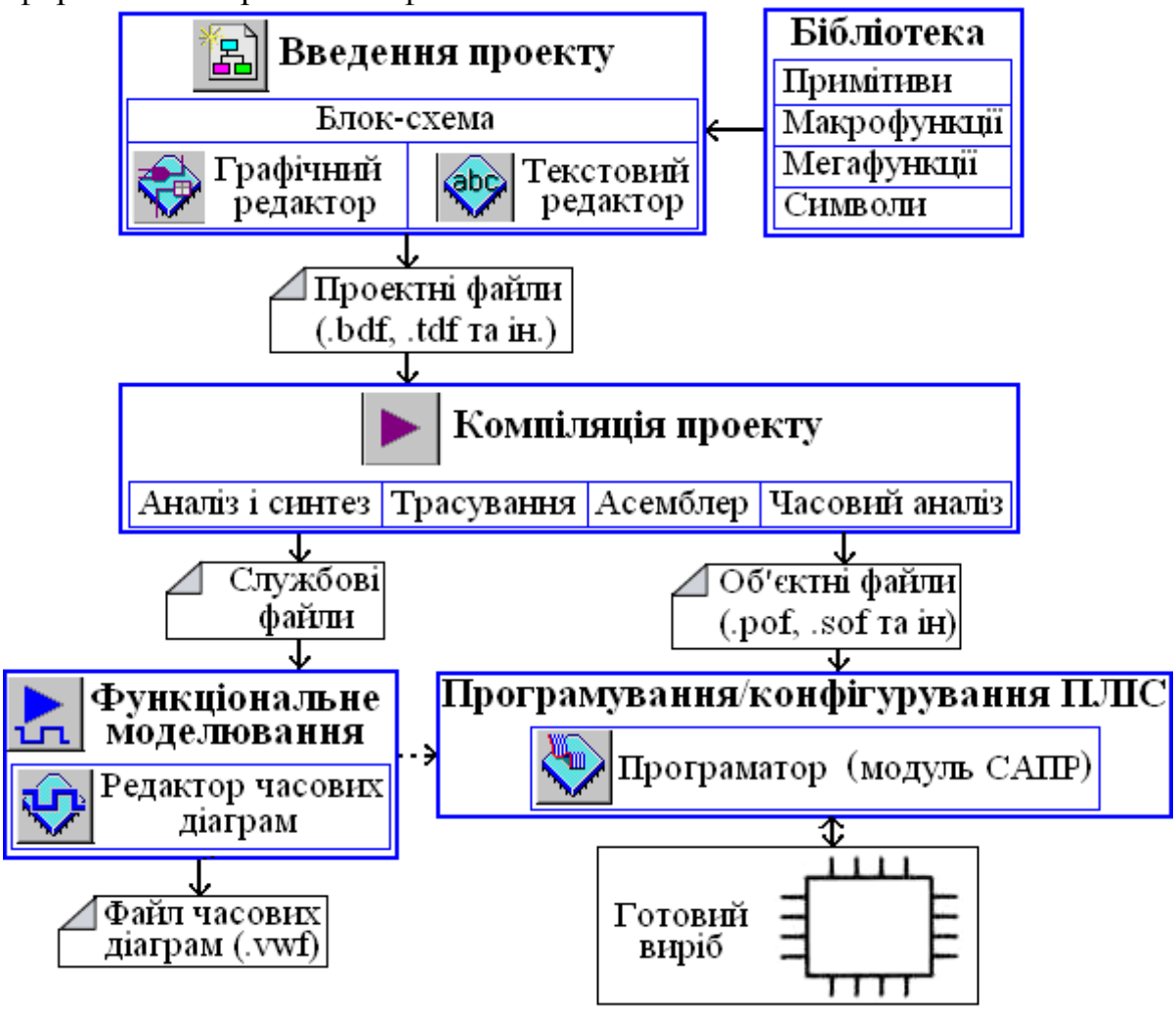


Рисунок В.1 – Етапи проектування за допомогою САПР

Другим і найважливішим етапом створення проекту є його *компіляція*, яка виконується автоматично, в декілька кроків. Після завантаження проектних файлів до компілятора здійснюється їх *аналіз*, щоб сформувати рівняння залежності вихідних функцій від вхідних змінних, і *синтез*, щоб реалізувати проект за використання мінімуму з приступного ресурсу ПЛІС залежно від її технології (наприклад, на мінімальній кількості логічних елементів). При цьому виявляються помилки, наприклад, коли відсутнє джерело сигналу на деякому вході, і застереження, наприклад, про невикористовуваний деякий компонент (до усунення помилок компіляція зупиняється). Наступним кроком є *трасування*, яке полягає у виборі місцеположення комірки ПЛІС для реалізації певної функції і маршруту її зв'язків із виводами мікросхеми таким чином, аби забезпечити максимальну швидкодію. За результатами трасування модуль компілятора *асемблер* генерує об'єктні файли для виготовлення реально працюючого пристрою. Останнім кроком компі-

ляції є *часовий аналіз*, який полягає у вимірюванні затримки сигналів різними шляхами їх поширення, визначенні критичних шляхів і, отже, найгіршої швидкодії, тобто придатності проекту для реалізації заданої мети.

Під час *функціонального моделювання* з переліку сигналів, що міститься в службових файлах компілятора, формується файл часових діаграм. При цьому вхідні сигнали задаються користувачем, а вихідні – генеруються імітатором (програмним модулем Simulator). Залежно від вибраного режиму імітатор може моделювати або лише виконувану пристроєм функцію, або функцію і часові затримки. Завдяки цьому функціональне моделювання дозволяє випробувати і налагодити пристрій перед втіленням його в кристалі, що значно скорочує термін проектування.

Завершальним етапом є фізичне *програмування/конфігурування* мікросхеми, яке здійснюється з об'єктних файлів програматором, що є модулем САПР. Крім того, залежно від використовуваних апаратних засобів програматор може виконувати перевірку функціонування готового виробу шляхом подачі на його входи випробувальних сигналів і порівняння вихідних сигналів зі сформованими часовими діаграмами.

***

З метою активізації навчання практикум має варіантний характер. Для зарахування кожної роботи необхідно: а) засвоїти теоретичний матеріал з даної теми, б) виконати практичне завдання згідно зі своїм варіантом, в) виконати загальну частину лабораторного завдання ознайомчого характеру, г) створити проекти за індивідуальним завданням з моделюванням результатів, д) створити лабораторний макет згідно із заданим варіантом та виконати експериментальні дослідження (відомості про лабораторний стенд містяться в [2, 8]). Щоб виконати завдання, потрібно засвоїти також процедури і елементи програмного забезпечення САПР Quartus II (у зв'язку з надзвичайно розгалуженою цією системою рекомендується обмежитися мінімумом відомостей, достатніх для даної роботи). Методично практикум побудовано таким чином, аби практичні і лабораторні завдання можна було виконати самостійно, у дистанційному режимі (експериментальні дослідження за п. *д* виконуються в лабораторії за наявності заздалегідь підготовлених файлів на магнітному носії).

Зміст звіту з виконання завдання

Назва і мета роботи; результати виконання домашнього завдання; зміст кожного пункту лабораторного завдання: бібліотечні файли (частинами або в скороченому вигляді, щоб помітно було деталі), файли зі схемами, часовими діаграмами (у разі потреби, застосувати альбомну орієнтацію сторінки) та іншими результатами зі стислими поясненнями і висновками; результати експериментальних досліджень, які можна подавати скорочено у формі таблиць і стислих висновків.

1 ЛАБОРАТОРНА РОБОТА №1. ОСНОВНІ ЛОГІЧНІ ФУНКЦІЇ

Мета роботи: дослідження типових логічних елементів; ознайомлення з основами пакету Quartus II та керуванням його файлами.

ДОМАШНЄ ЗАВДАННЯ

1) Засвоїти теоретичні відомості щодо основних логічних функцій, співвідношень алгебри логіки, а також арифметичних основ цифрової техніки, необхідних для розуміння логічних схем.

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

СТИСЛІ ТЕОРЕТИЧНІ ВІДОМОСТІ

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 табл. 1.1. Пояснення щодо основ алгебри логіки, а також арифметичні основи цифрової техніки, необхідні для розуміння логічних схем, викладено в [1].

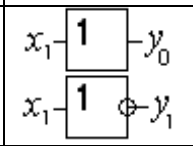
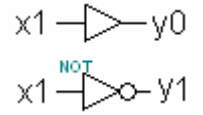
Основні співвідношення алгебри логіки (табл. 1.2) розглядаються в булевій алгебрі, здебільшого, відносно функцій АБО та І, а справедливість тих чи тих формул відносно інших функцій з'ясовується окремо (у табл. 2 для прикладу наведено також співвідношення для функції Виключне АБО).

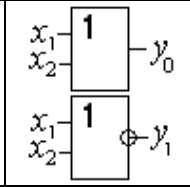
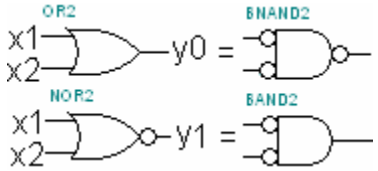
Будь-яка логічна функція зображається в досконалій диз'юнктивній нормальній формі (ДДНФ) або в досконалій кон'юнктивній нормальній формі (ДКНФ) єдиним способом, тому ці форми називають стандартними, що є вихідними для подальшого аналізу й синтезу. Природно, стандартні форми можна зобразити і для інверсного значення функції (табл. 1.3)

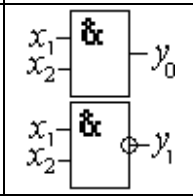
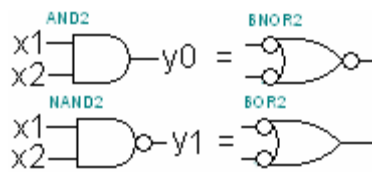
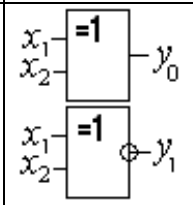
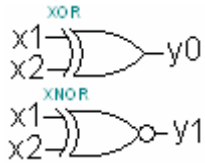
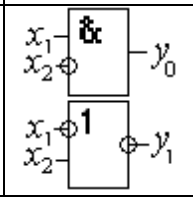
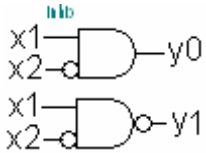
Примітиви є функціональними модулями в САПР, які репрезентують найпростіші типові елементи ЦП, а також допоміжні компоненти в графічних і текстових файлах проекту (термін Primitive охоплює базові логічні елементи, тригери, порти та ін.).

Бібліотека всіх функціональних модулів САПР розташована в каталозі c:/quartus/libraries, а примітиви зосереджено в підкаталозі /primitives. Відомості про примітиви (рис. 1.1) подано в [2].

Таблиця 1.1 – Основні логічні функції

Назва логічної функції (логічного елемента)		Таблиця відповідності		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 а)	І б)	Виключне АБО 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 Моргана)	$\overline{\overline{x_1} \overline{x_2}} = x_1 + x_2$ $\overline{\overline{x_1} + \overline{x_2}} = x_1 x_2$	$\overline{\overline{x_1} + \overline{x_2}} = x_1 x_2$ $\overline{\overline{x_1} \overline{x_2}} = x_1 + x_2$	
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	$x_1 + \overline{x_1} x_2 = x_1 + x_2$	$x_1 (\overline{x_1} + x_2) = x_1 x_2$	$x_1 \oplus \overline{x_1} x_2 = x_1 + x_2$

Таблиця 1.3 – Стандартні форми логічних функцій

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

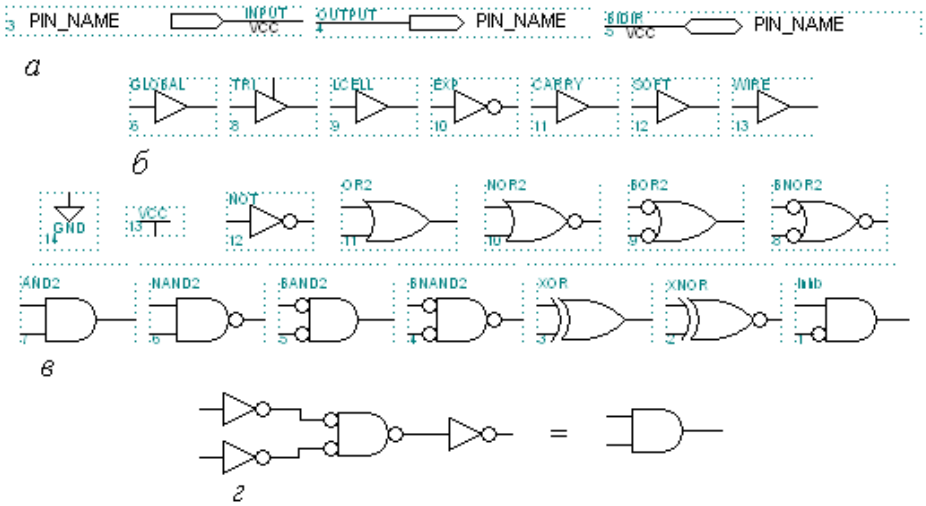


Рис. 1.1 – Основні примітиви

КОНТРОЛЬНІ ЗАПИТАННЯ

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

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

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

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

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) $y = \overline{x_1 + \overline{x_2} + \overline{x_3}\ \overline{x_4 + \overline{x_5}}} - I - АБО - НЕ$;
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) $y = \overline{x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5} - Викл.АБО$.

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

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

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

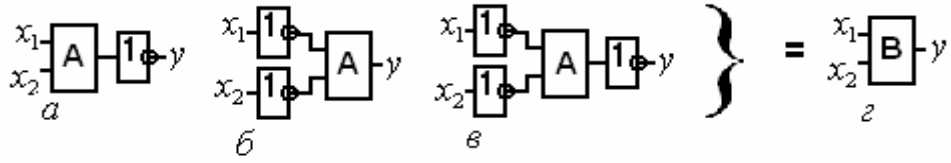


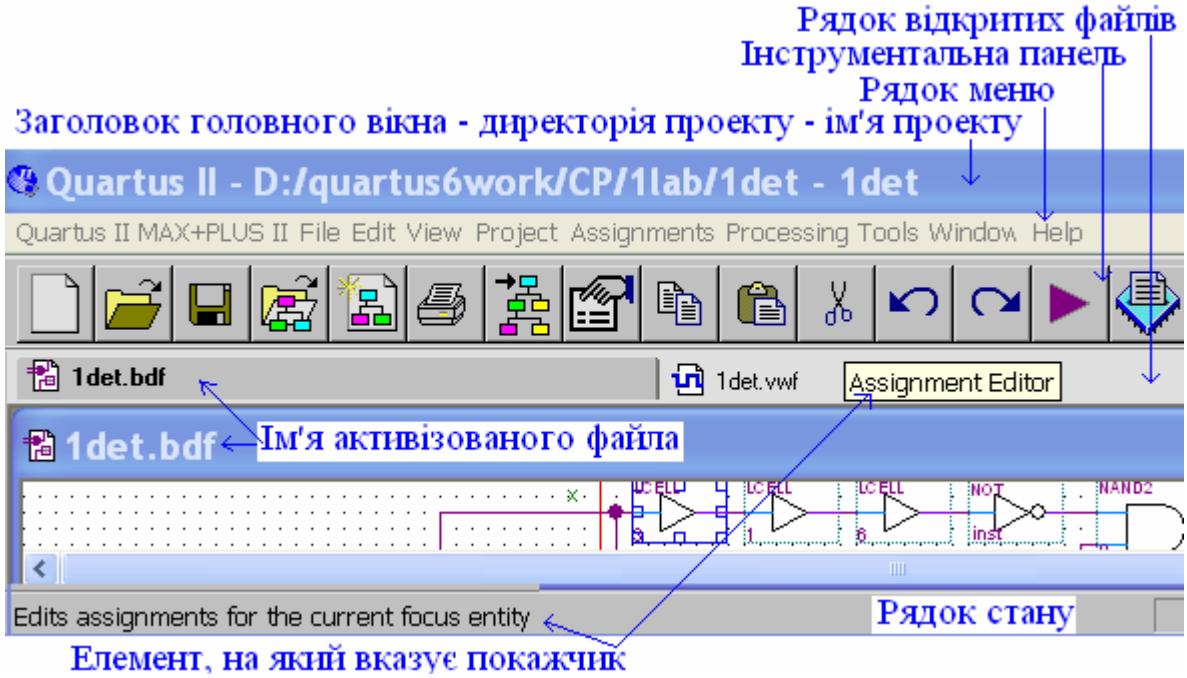
Рисунок 1.2

ЛАБОРАТОРНЕ ЗАВДАННЯ

1 Налаштувати основні органи керування вікнами й файлами програмного пакету САПР Quartus II.

☝ 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 „>”, а натискання правої кнопки миші (Button 2) – інколи знаком „B2”.

1.1 ***Запустити програму*** Quartus II ярликом на робочому столі (або з меню Windows: Пуск > Всі програми > Altera > Quartus II 6.1). Якщо є допоміжні вікна (ліворуч і знизу), зачинити їх та ознайомитися з елементами головного вікна і списками типових команд меню File, Edit (редагування), View (вигляд), Window (вікно) та Help (довідка).



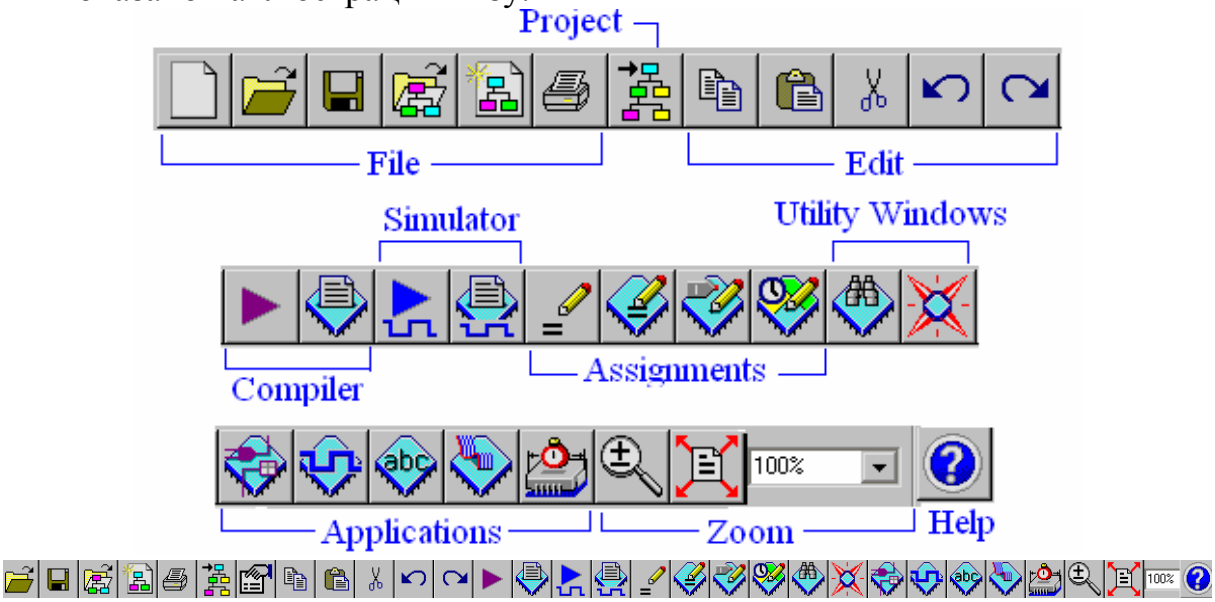
1.2 Налаштувати з мінімуму зручних для роботи піктограм ***інструментальну панель головного вікна.***

а) Налаштувати стиль відображення інструментальної панелі головного вікна. Для цього в меню Tools (інструментальні засоби) > Customize (налаштувати) > на вкладці General (загальні), у розділі Look & Feel (стиль відображення) ввімкнути Quartus II, у розділі Quick menus (швидкі меню), у рядках Quartus II menu та MAX+PLUS II menu прокруткою вибрати Left (розташувати ліворуч).

б) Установити стандартну панель пакету: на вкладці Toolbars (інструментальні панелі) зі списку панелей Toolbars вибрати тільки Standard Quartus II, а всі інші вимкнути. Крім того, встановити облямовані кнопки великого розміру: зняти прапорець Borderless look (кнопки без облямівок), та підняти прапорці Show tooltips (показувати типи інструментів) і Large buttons (великі кнопки).

в) Сформувати інструментальну панель: на вкладці Commands вибрати категорію File, взятися за значок у полі Buttons і перетягнути його до

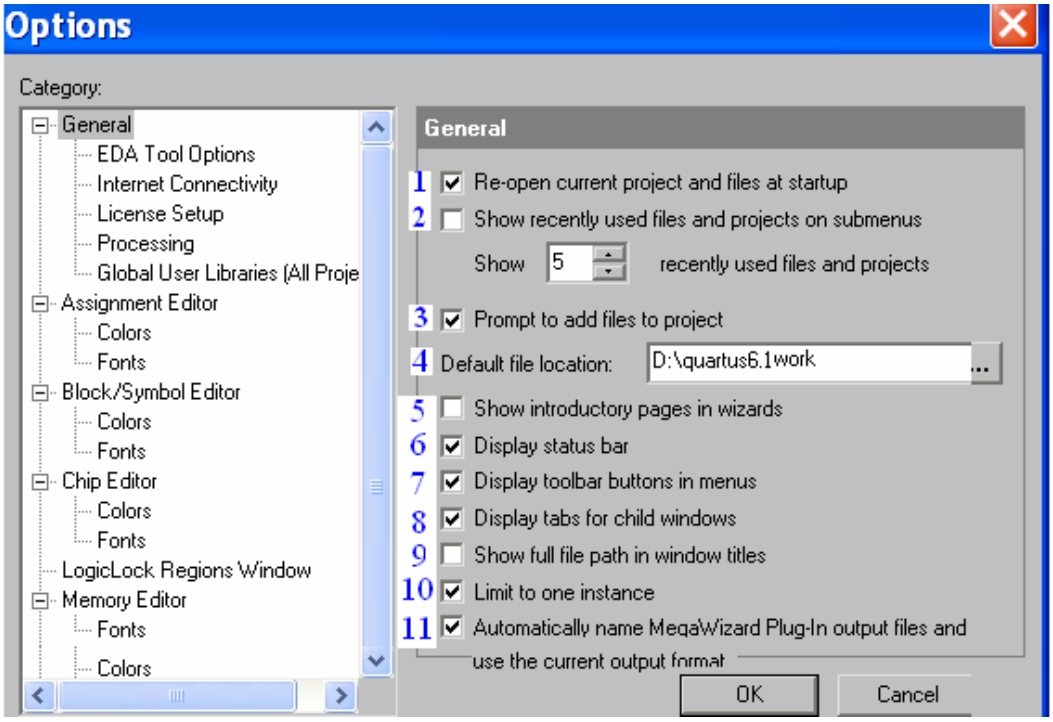
відповідної позиції на панелі та вилучити зайві кнопки перетягуванням їх із панелі до робочого поля вікна. Так само встановити кнопки з інших категорій, зазначених на ілюстрації, та натиснути кнопку ОК у вікні Customize. Налаштовану панель з одного рядка найуживаніших інструментів показано на ілюстрації внизу.



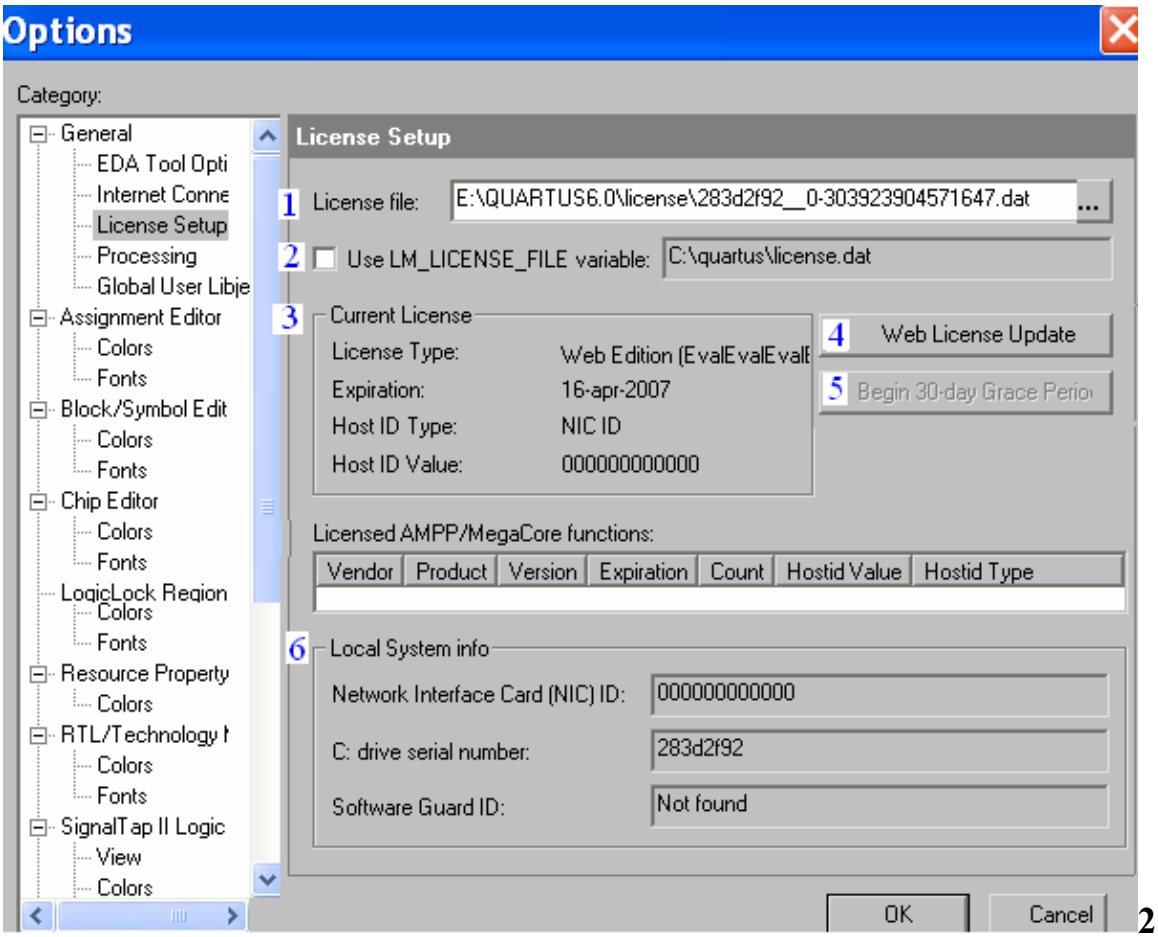
1.3 Налаштувати ***загальні опції*** робочого середовища.

а) Визначити опції зручності роботи з пакетом. Для цього з меню Tools > Options розкрити сторінку General (загальні) діалогового вікна опцій (ілюстрацію див. нижче), в якому встановити прапорці таких опцій: 1) ввімкнути: повторно відкривати поточні проект і файли під час запуску – останній проект і відкриті файли запам'ятовуються і автоматично перезавантажуються по ввімкненні програми; 2) вимкнути: показувати недавно використовувані файли і проекти в підменю, у віконці Show залишити максимальну їх кількість 5 – останні використовувані файли і проекти відображатимуться безпосередньо в меню File (якщо опцію ввімкнути, до десяти об'єктів відображатимуться в кожному з підменю File > Recent Files та File > Recent Projects); 3) ввімкнути: підказ, чи додавати файли до проекту – це запитання з'являтиметься під час збереження файлу, який не був включений до складу проекту; 4) кнопкою огляду (…) вибрати місцеположення файлу за умовчанням – вибраний початковий каталог з'являтиметься під час відкриття і збереження файлів; 5) вимкнути: показ вступних сторінок майстрів – сторінки із загальним описом не показуватимуться під час створення нового проекту та активізації інших майстрів (програмних модулів); 6) ввімкнути: відображати рядок стану – у звичайному режимі цей рядок дає стислий опис функції інструмента панелі або палітри при наведенні на нього покажчика, а по запуску компілятора і імітатора відображає перебіг процесів; 7) ввімкнути: показувати кнопки панелі в меню – поруч з командами в меню відображатимуться позначки кнопок відповідних інструментів панелей і палітр; 8) ввімкнути: показувати таблички дочірніх вікон – відображатиметься рядок відкритих файлів; 9) вимкнути:

показувати повний шлях у вікнах файлів – у рядках заголовків файлів відображатиметься лише частина директорії, відмінна від директорії проекту, що подається в рядку заголовка головного вікна; 10) ввімкнути: обмежити одним зразком – якщо буде відкрите друге головне вікно Quartus II, один і той самий проект може бути відкритим лише в одному з них; 11) ввімкнути: автоматично давати ім'я вихідному файлові мегафункції і використовувати поточний вихідний формат – під час вставлення зразка символу мегафункції автоматично генерується вихідний текстовий файл завжди тією мовою, яка встановлена при виборі мегафункції, а її ім'я складатиметься зі стандартного імені функції з додаванням номеру.



б) Переглянути файл ліцензії програмного забезпечення і, у разі потреби, встановити його. Для цього відкрити сторінку License Setup (установки ліцензії) з такими позначеннями на ілюстрації: 1) файл ліцензії у програмному забезпеченні Quartus II, який можна встановити кнопкою огляду (…); 2) використовувати змінну файлу ліцензії – якщо цю опцію встановлено, поле License file є неприступне для редагування і вибирається файл ліцензії license.dat; 3) параметри поточної ліцензії: її тип, ***термін дії***, тип та номери ідентифікаторів (пов'язані із системою комп'ютера і типом ліцензії); 4) підімкнення до Altera website для поновлення ліцензії; 5) початок 30-денного використання програмного забезпечення (без можливості програмувати/конфігурувати мікросхему) – по кожному запуску в повідомленні зазначається, скільки днів залишилось із цього періоду, після якого компіляція і імітація блокуються; 6) системні відомості ПК, які можуть знадобитися для одержання ліцензії. Відтак натиснути ОК у вікні Options.

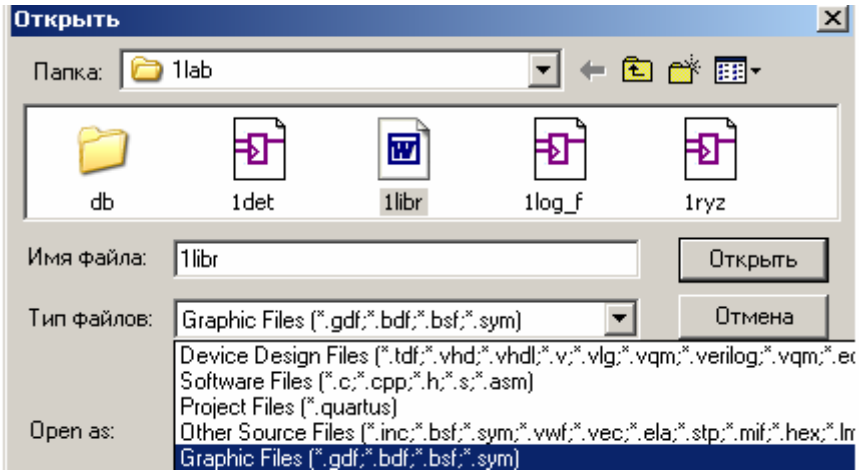


2

Засвоїти основи керування файлами пакету Quartus II.

2.1 ***Відкрити потрібний файл***, наприклад, 1libr.bdf з директорією (повним каталогом) d:\quartus6.1work\1lab\1libr.bdf:

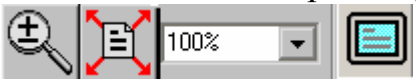
а) піктограмою панелі інструментів Open (або з меню File > Open) викликати діалогове вікно Open (відкрити існуючий файл певного типу);



б) у діалоговому вікні Open прокруткою вибрати потрібний тип файлів – Graphic Files (графічні файли);

в) у цьому ж вікні, у разі потреби, за допомогою стрілки „на один рівень вгору” та подвійним клацанням по відповідних папках послідовно розкрити директорію d:, quartus6.1work, 1lab та відкрити файл 1libr.

2.2 Переглянути і ***скопіювати графічний файл***:



а) інструментом Fit in Window (аркуш файлу з червоними стрілками в кутах) розташу-

вати весь файл у розгорнутій частині вікна, аби мати уявлення про його розміри, відтак інструментами керування масштабом зображення повернутися до нормального розміру (100%) та розгорнути головне вікно на весь екран (інструмент Full Screen у вигляді екрана на вертикальній палітрі);

✍ *Примітка*: Іструмент Zoom Tool (у вигляді лінзи) при натисканні лівою кнопкою миші збільшує зображення, а правою кнопкою – зменшує.

б) виділити потрібну ділянку для копіювання графічного файлу: натиснути інструмент вибору (Selection – у вигляді стрілки) і протягнути ним по діагоналі ділянки, утримуючи натиснутою ліву кнопку миші, по відпусканні якої всі елементи ділянки виділяться синім кольором (виділити весь файл зручно командою меню Edit > Select All);

в) інструментом верхньої панелі Copy – копіювати (або з меню Edit > Copy) скопіювати виділену ділянку в буфер.

2.3 ***Створити власний файл потрібного типу***, наприклад, prim.bdf з директорією d:\RT\2br\prim.bdf:

а) піктограмою горизонтальної панелі New Block Diagram/Schematic File (або з меню File > New > Device Design Files > Block Diagram/Schematic File > OK) створити новий файл у графічному редакторі (Block & Symbol Editors) – з'явиться вікно з безіменною назвою Block N (де N – довільний номер);

б) у лівому верхньому куту цього вікна курсором клацнути місце вставлення копії і натиснути піктограму панелі інструментів Paste (або меню File > Paste) – копія зображення позиціонується лівим верхнім кутом у місці клацання;

в) піктограмою Save (або меню File > Save) викликати діалогове вікно Save As (зберегти як), в якому за допомогою стрілки „на один рівень вгору" та подвійним клацанням по відповідних папках послідовно розкрити директорію d:, у разі потреби, створити вкладені теки RT\2br (піктограмою створення нової теки в цьому ж вікні), розкрити

останню теку 2br, ввести ім'я файлу prim (примітиви) та натиснути кнопку збереження – у рядку заголовка файлу відобразиться директорія і ім'я файлу з розширенням .bdf.

✍ *Примітки:*

1) Копіювати файли для звіту можна під час виконання відповідних пунктів лабораторної роботи. Якщо тека вже існує, немає потреби створювати директорію, достатньо її вибрати та ввести лише ім'я файлу.

2) Під час змін у файлі після його заголовка з'являється зірочка в його вікні і в рядку відкритих файлів, яка зникає по збереженні файлу.

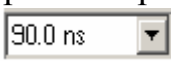
3) Тут подано швидке виконання операцій з файлами за допомогою піктограм інструментів, але їх можна виконати також командами складників рядка меню або з контекстного меню, яке викликається кнопкою B2 (виконання операцій сполученням клавіш тут, здебільшого, не розглядається).

4) 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 теці Мій комп'ютер Windows з лабораторної до своєї теки.

2.4 ***Скопіювати*** у власну теку потрібний ***сигнальний файл*** (файл часових діаграм) або його частину з лабораторної теки:

а) відкрити сигнальний файл (першою піктограмою), наприклад, 1logfun.vwf з директорією d:\quartus6.1work\1lab\1logfun.vwf (виконати п. 2.1, але в п. 2.1,б прокруткою вибрати файли типу Waveform/Vector Files), вписати все зображення у вікні (другою піктограмою), виділити потрібні часові діаграми (командою меню Edit > Select > All Nodes & Buses, або натиснути рядок першого сигналу в полі заголовків і клацнути останній рядок при натиснутій клавіші Shift для виділення всіх діаграм, або клацати рядки при натиснутій клавіші Ctrl для виділення діаграм вибірково) і скопіювати їх піктограмою (або з меню Edit > Copy);



б) зчитати кінець часового інтервалу діаграм та період часової сітки відповідно з масштабного віконця горизонтальної панелі інструментів і верхнього рядка часових діаграм (або з меню Edit > End Time та Edit > Grid Size);

в) піктограмою New Vector Waveform File – створення нового файлу часових діаграм (або з меню File > New > Other Files > Vector Waveform File > OK) відкрити вікно нового файлу і клацнути піктограму вставлення копії з буфера – у цьому вікні з'явиться копія часових діаграм;

г) виставити такі самі часові параметри, як у п. б за допомогою меню Edit > End Time та Edit > Grid Size, відтак для перегляду діаграм піктограмою Fit in Window (вписати все зображення у вікно) розташувати весь часовий інтервал у вікні файлу;

ґ) надати ім'я, наприклад, 1logfun.vwf безіменному Waveform N файлу та зберегти його у власній теці (п. 2.3,в).

2.5 Ознайомитися з керуванням ***групою файлів.***

а) Під час роботи над проектами зручно оперувати з низкою файлів, заголовки яких розташовуються в рядку відкритих файлів. Для активізації потрібного файлу достатньо клацнути його назву в цьому рядку.

б) Файли можна розташувати з відображенням заголовків також каскадом (командою з меню Window > Cascade) або у вигляді неперетинних вікон на екрані (командами з меню Window > The Horizontally або The Vertically), а також згорнути файли, не потрібні в першу чергу, до іконок внизу головного вікна Quartus II клацанням по їх символу згортання.

в) Для відкриття одного з п'яти недавно використовуваних файлів достатньо клацнути його назву з верхнього списку в меню File (нижній список містить п'ять недавно використовуваних проектів).

3 Дослідити логічні елементи.

3.1 Ознайомитися з дібраними елементами бібліотеки примітивів

(файл 1libr.bdf): 1) портами, 2) буферами, 3) логічними елементами, 4) логічними 0 та 1, а також прикладами їх мікросхемного виконання на ІС серії 74. Дати тлумачення сенсу кожного позначеного символами примітива. Отримати інформацію щодо примітивів з довідки: Help > Contents > Primitives > Ім'я примітива. У звіті навести елементи 1 … 3 та деякі приклади ІС.

3.2 Визначити в булевому базисі еквівалентну логічну функцію, виконувану кожною схемою на примітивах (файл 1log_f.bdf).

3.3 За ідеалізованими часовими діаграмами без урахування затримок (файл 1logfun.vwf) 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 маркером часу. 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 у полі імені (Name). Відтак перетягнути маркер на другу ділянку і так само зчитати рівні і т. д. до заповнення всієї таблиці відповідності.

Примітки:

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

2) Часові позиції можна відмітити додатковим маркером часу, який створюється з меню Edit > Time Bar Organizer > ввести в діалоговому вікні Time Bar Organizer, у рядку Time потрібний час положення додаткового маркера > Add > OK. Для анулювання цього маркера досить у діалоговому вікні Time Bar Organizer виділити його час у рядку Existing time bars > натиснути Delete > OK.

3.4 На копії схеми у власному графічному файлі 1log_f.bdf навести номери часових діаграм з файлу 1logfun.vwf, що відповідають кожному виходу. Для цього виділити порт і натиснути інструмент палітри Properties (або клацнути символ порту інструментом палітри Text Tool) та в діалоговому вікні Pin Properties (властивості виводів) > Gene-ral (загальні) до імені виводу в рядку Pin Name додати через знак рівності функцію з часової діаграми, наприклад, y1 = z1 > OK (аби вийти з текстового режиму, слід натиснути інструмент вибору – стрілку).

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

4 Дослідити кола формування коротких імпульсів на логічних елементах.

4.1 За схемою і часовими діаграмами (файли 1det.bdf, .vwf) розглянути принцип побудови різницевого елемента (РЕ) на логічному елементі І-НЕ і елементі затримки (буфери LCELL) з інверсією (NOT) та застосування його у колах формування коротких імпульсів – детекторах фронтів вхідних імпульсів *x*: за позитивним (dx1), негативним (dx2n) їх перепадом та за обома перепадами (dxn). Переглянути також окремі детектори одного фронту (файли 1det1 та 1det2.bdf, .vwf, .bsf).

✍ *Примітки:*

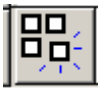
1) У класичному детекторі фронтів на ІС жорсткої структури для затримки та інверсії застосовують ланцюжок з непарної кількості логічних елементів з інверсними виходами. Щодо алгебри логіки це не має сенсу, тому компілятор за аксіомою подвійного заперечення усуває зі схеми „зайві” ланцюжки елементів,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

2) На часових діаграмах 1det.vwf у полі типів контакту наведено три типи вузлів: вхідний порт із позначкою I (Input), вихідні порти O (Output) та деякі внутрішні вузли C (Combinatorial Node – комбінаційний вузол), не сполучені безпосередньо із зовнішніми контактами ІС.

4.2 Виміряти затримки поширення сигналів та тривалості вихідних імпульсів ручним способом. Для цього уривистим клацанням по стрілках біля поля часу маркера (див. малюнок до п. 3.3) 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): індикатори Pointer та Interval відзначають часове положення стрілки і віддаль між нею і маркером.



2) Для переведення маркера в довільну позицію незалежно від кроку сітки слід відтиснути інструмент палітри або зняти прапорець Snap to Grid (прив’язка до сітки) в меню View.

2 ЛАБОРАТОРНА РОБОТА №2.
РЕАЛІЗАЦІЯ ЛОГІЧНИХ ФУНКЦІЙ

Мета роботи: реалізація логічних функцій у поширених базисах; 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. Додатки, 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 п. ***в, г.***

СТИСЛІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Мінімізацію логічних функцій, реалізацію їх у поширених базисах, а також питання схемної мінімізації розглянуто в [1]. Відомості про макрофункції логічних елементів (рис. 2.1,а,б) подано в [2].

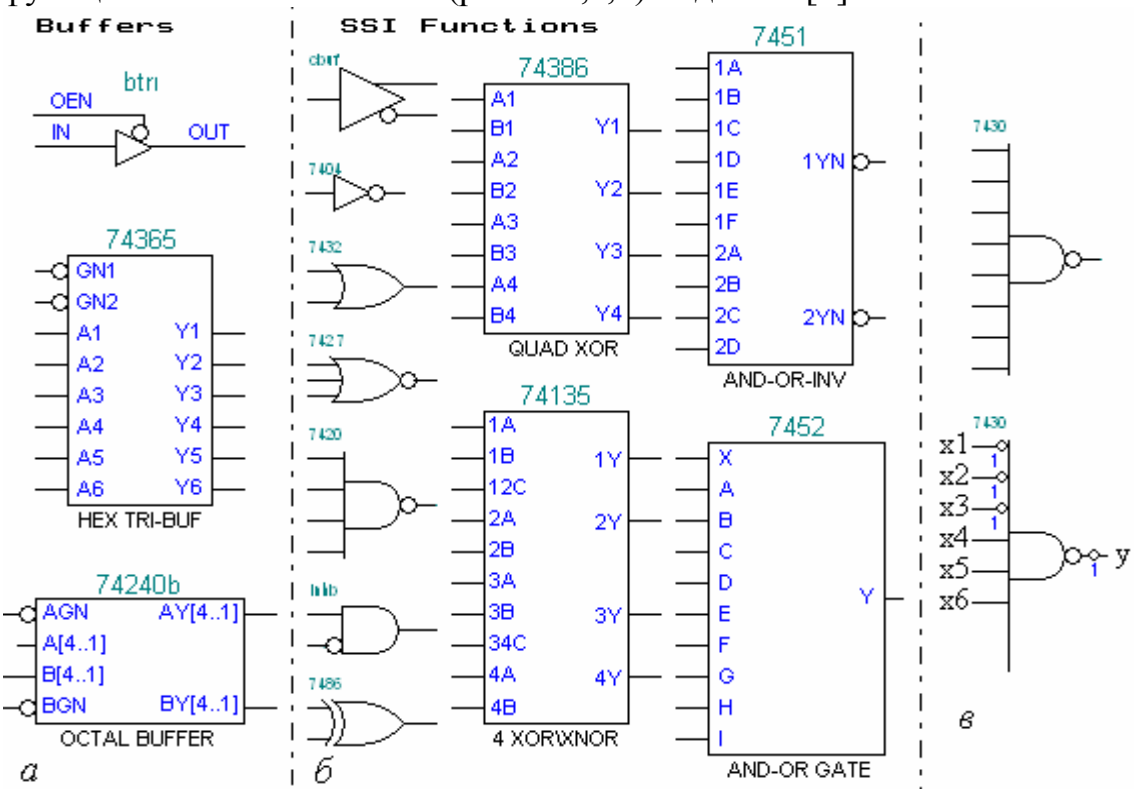


Рисунок 2.1

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

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

то потрібен один нестандартний елемент І (із шістьма входами, три з яких є інверсні), який неважко виконати шляхом настроювання макрофункції (рис. 2.1,в).

КОНТРОЛЬНІ ЗАПИТАННЯ

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

2.2.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) $x_1\overline{x_2 + x_3} + x_2\overline{x_1 + x_3} + x_3\overline{x_1 + x_2} + \overline{x_1 + x_2 + x_3}$;

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 необхідністю, скористайтеся діаграмами термів.

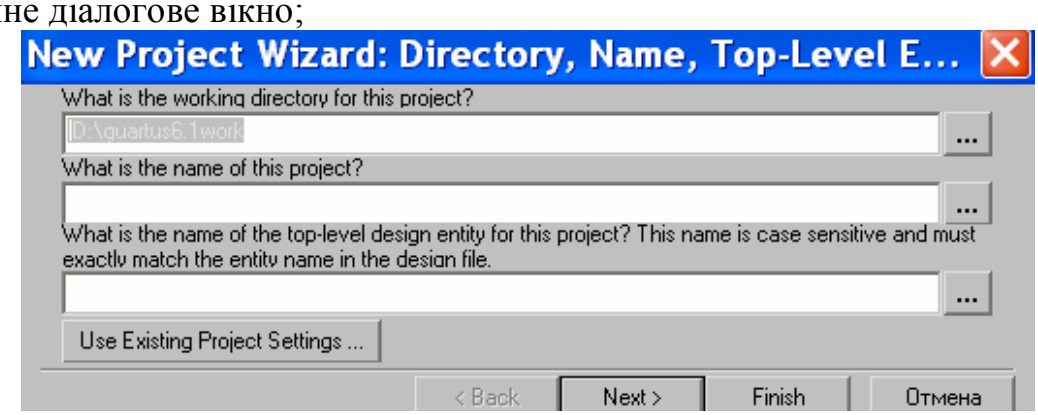
2.3. Виконайте спільну мінімізацію чотирьох вихідних функцій, що здійснюють перетворення ДДК 8421 у такі коди: *а*) 2421, *б*) 7421, *в*) 8421+3 (з надлишком три), *г*) код Ґрея (перші десять цифр $X_{10} = 0...9$ чотирирозрядного коду). *Вказівки.* На заборонених кортежах $X_{10} > 9$ вважати функції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

ЛАБОРАТОРНЕ ЗАВДАННЯ

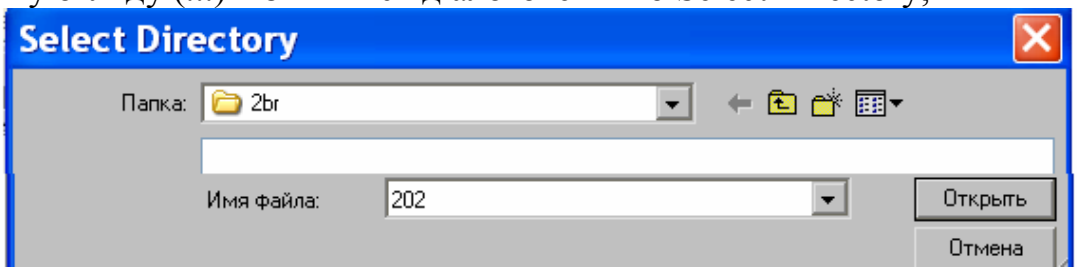
1 Засвоїти основи *графічного* введення проекту на рівні примітивів для заданого варіанту XX логічної функції (для прикладу див. файл ..\quartus6.1work\2lab\200.bdf, схема 1).

1.1 ***Дати ім'я новому проекту*** 2XX (XX – варіант) з директорією власної теки, наприклад, ..\RT\2br\202:

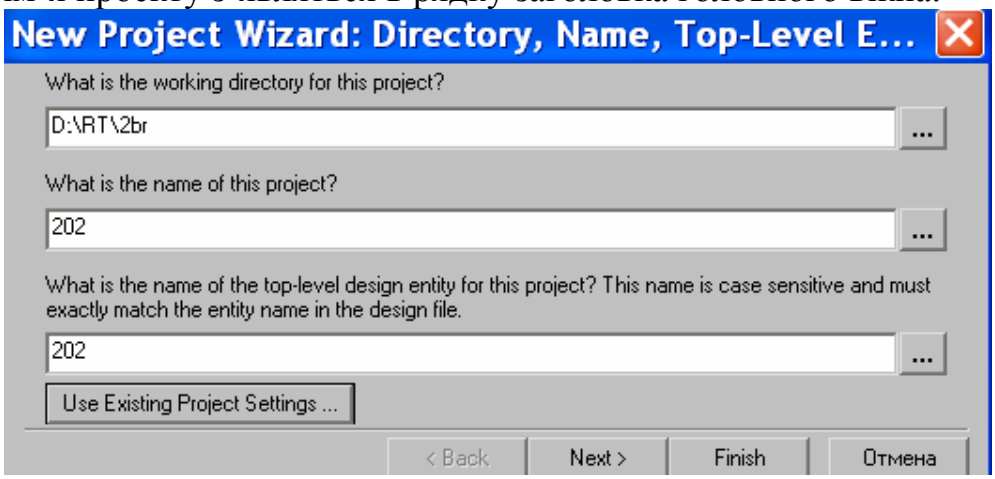
а) піктограмою New Project Wizard (або з меню File > New Project Wizard) запустити майстра нового проекту – з'явиться однойменне діалогове вікно;



б) у верхньому рядку, якщо директорія зазначена інша, натиснути кнопку огляду (...) – з'явиться діалогове вікно Select Directory;



в) у цьому вікні вибрати потрібну теку (\RT\2br), ввести ім'я файлу (202) і натиснути кнопку відкриття – заповниться вся сторінка діалогового вікна New Project Wizard: у верхньому рядку з'явиться директорія, у другому – ім'я проекту (202), яке продублюється в третьому рядку як ім'я об'єкту верхнього рівня проекту. Відтак натиснути кнопку Finish – директорія і ім'я проекту з'являться в рядку заголовка головного вікна.

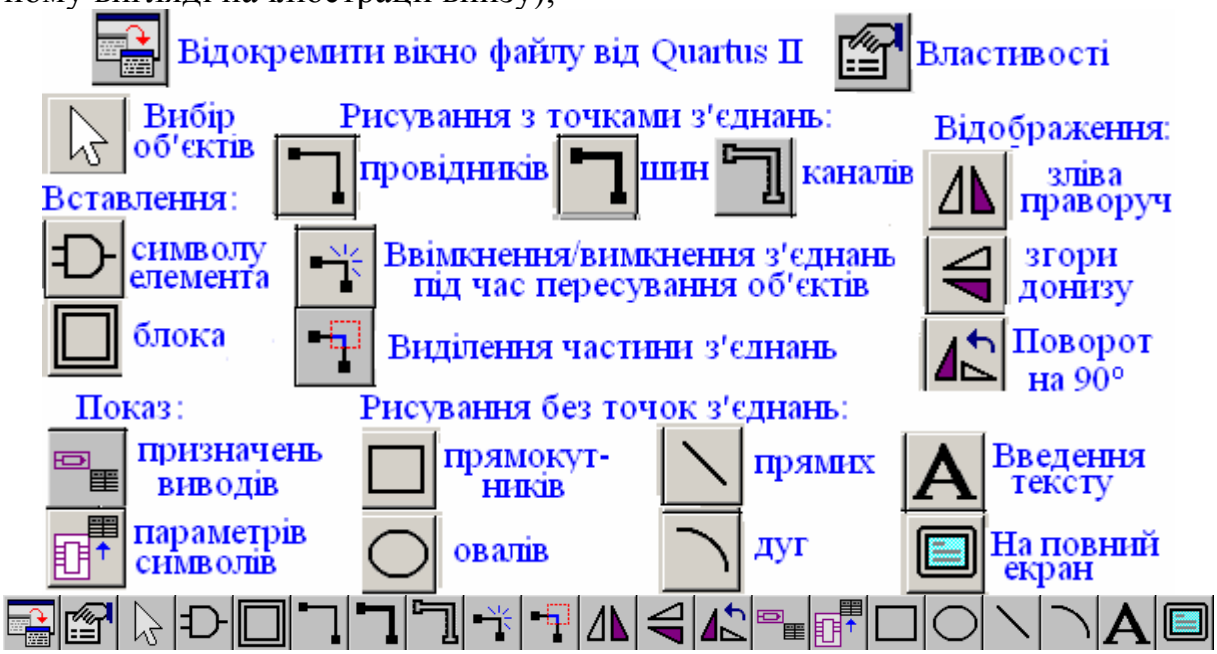


✍ *Примітка:* Рядки діалогового вікна New Project Wizard можна заповнити або скоригувати також вручну.

1.2 ***Створити графічний файл*** 2XX.bdf нового проекту:

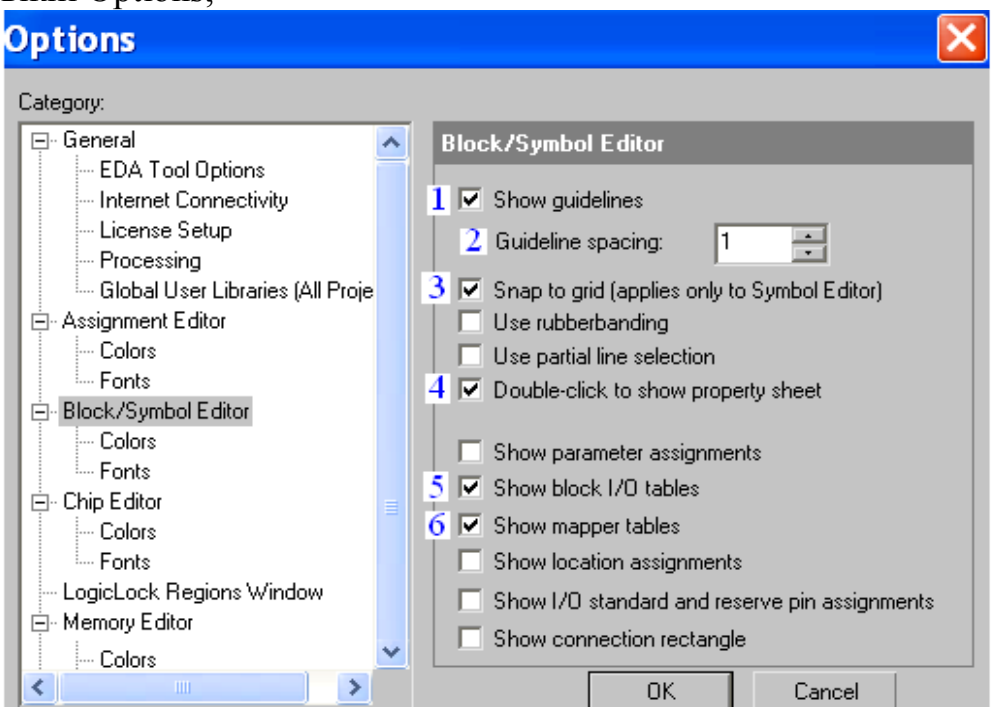
а) піктограмою горизонтальної панелі New Block Diagram / Schematic File (або з меню File > New > сторінка Device Design Files > Block Diagram/Schematic File) *відкрити новий графічний файл* – з'явиться вікно з безіменною назвою BlockN.bdf (де N – номер);

б) налаштувати *інструментальну палітру* графічного редактора (розташована вертикально ліворуч): меню Tools (інструментальні засоби) > Customize Block Editor (налаштувати графічний редактор) > на сторінці Toolbars (інструментальні панелі) встановити облямовані кнопки великого розміру: зняти прапорець Borderless look (кнопки без облямівок), та підняти прапорці Show tooltips (показ типів інструментів) і Large buttons (великі кнопки) > на сторінці Commands вибрати категорію Block Editor, взятися за значок у полі Buttons і перетягнути його до відповідної позиції на палітрі та вилучити зайві кнопки перетягуванням їх із палітри до робочого поля файлу; відтак натиснути кнопку OK у вікні Customize (налаштовану палітру з одного стовпця найуживаніших інструментів показано в горизонтальному вигляді на ілюстрації внизу);

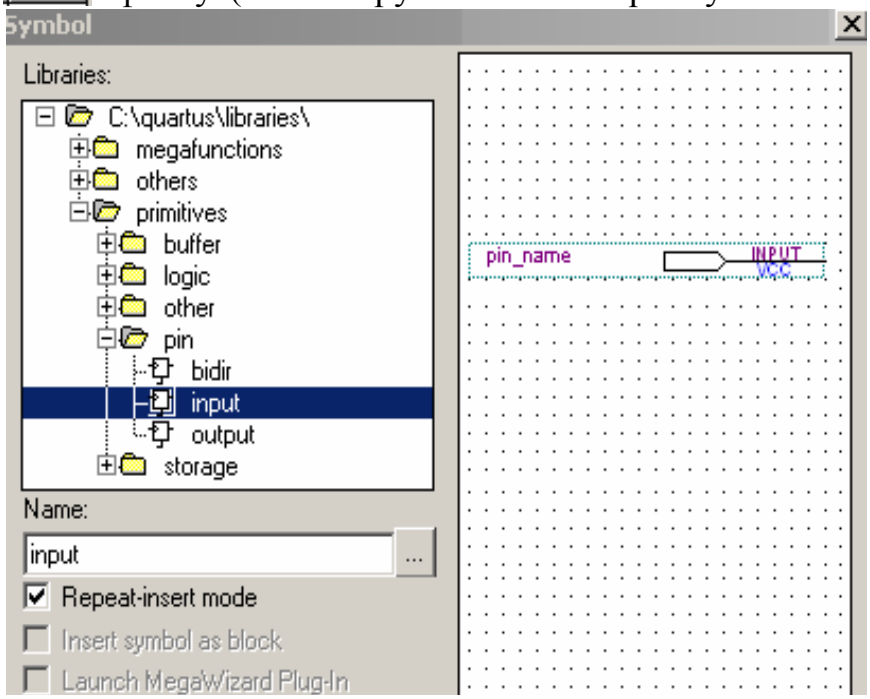


в) налаштувати *опції* графічного редактора, для чого з меню Tools > Options розкрити сторінку Block/Symbol Editor діалогового вікна опцій (ілюстрацію див. нижче), в якому ввімкнути прапорці таких опцій: 1) показ сітки – для зручності розташування у вікні елементів схеми; 2) в рядку Guideline spacing прокруткою встановити крок сітки 1; 3) прив'язка до сітки (застосовується тільки до символьного редактора, а в графічному редакторі об'єкти завжди прив'язано до координатної сітки) – дозволяє позиціонувати елементи символу під час його графічного редагування; 4) подвійним клацанням об'єкту викликáти вікно його властивостей (ця опція діє незалежно від прапорця в окремих версіях пакету); 5) показ таблиць вхо-

дів/виходів блоку – у проектах на рівні блок-схеми відображає всередині блоку таблиці його входів і виходів; 6) показ трасувальних таблиць – у проектах на рівні блок-схеми відображає зв'язки виводів блоку із сигналами зовнішніх з'єднань (функції інших опцій, не відмічених на ілюстрації цифрами, або оперативно керуються інструментами палітри, або відображають дані, що містяться в довідці на мікросхему); відтак натиснути кнопку OK у вікні Options;



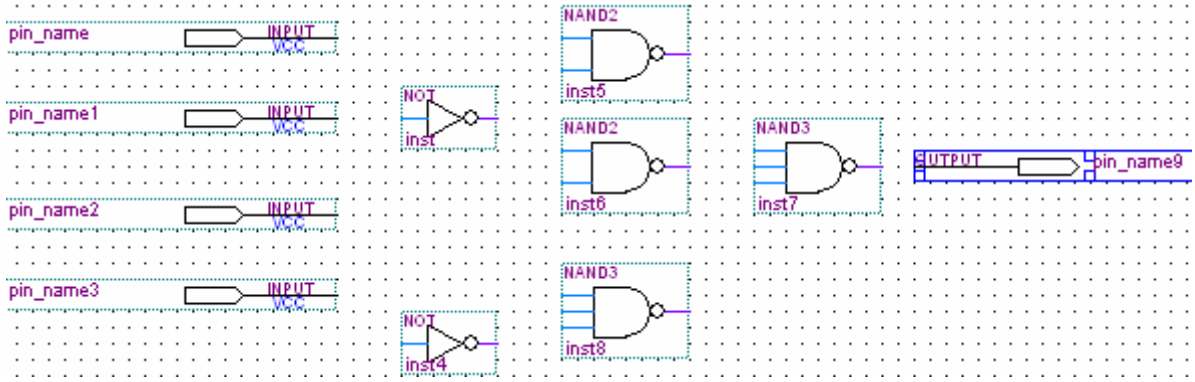
г) подвійним клацанням у вільному місці поля графічного файлу (або інструментом палітри Symbol Tool, або з меню Edit > Insert Symbol) викликати діалогове вікно Symbol;



ґ) розгортанням бібліотеки у віконці Libraries (натисканням знаку +) відчинити primitives > pin > input – у правому вікні з'явиться зразок символу вхідного порту (виводу IC); відтак підняти прапорець Repeat-insert mode (режим вставлення з повторенням) та натиснути OK – символ буде прив'язаний до покажчика миші;

д) клацанням лівою кнопкою миші вставити до графічного файлу необхідну кількість вхідних портів, які позиціонуються верхнім лівим ку-

том у місцях клацання (аби скасувати прив'язку символу до покажчика, достатньо натиснути інструмент вибору – стрілку);



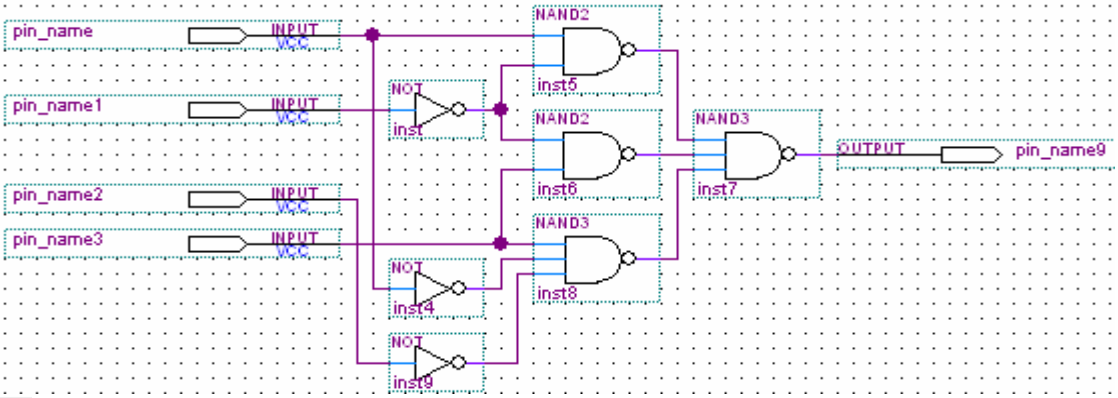
е) повторюючи п. г, ґ, д, вставити всі потрібні елементи, наприклад, not, nand2, nand3 (категорія logic) та вихідний порт output (категорія pin);

є) користуючись інструментами палітри, розташувати елементи у вікні графічного файлу (приблизно як на схемі): клацання інструментом вибору всередині елемента *виділяє* його, а поза елементом – скасовує виділення (виділити всі елементи можна з меню Edit > Select All); якщо поставити цей інструмент на елемент, то при натиснутій лівій кнопці його можна *пересунути* по екрану (звичайний метод "Drang & Drop"); *скопіювати і пересунути* копію елемента на нове місце можна методом "Drang & Drop" при натиснутій і утримуваній клавіші Ctrl (її треба відпускати останньою); виділений елемент можна також *відобразити* по горизонталі (за ДЕСТУ в логічних схемах цього робити не можна) або по вертикалі (змінюється позиціонування написів), повернути на кут 90° тощо; крім того, за допомогою горизонтальної панелі інструментів

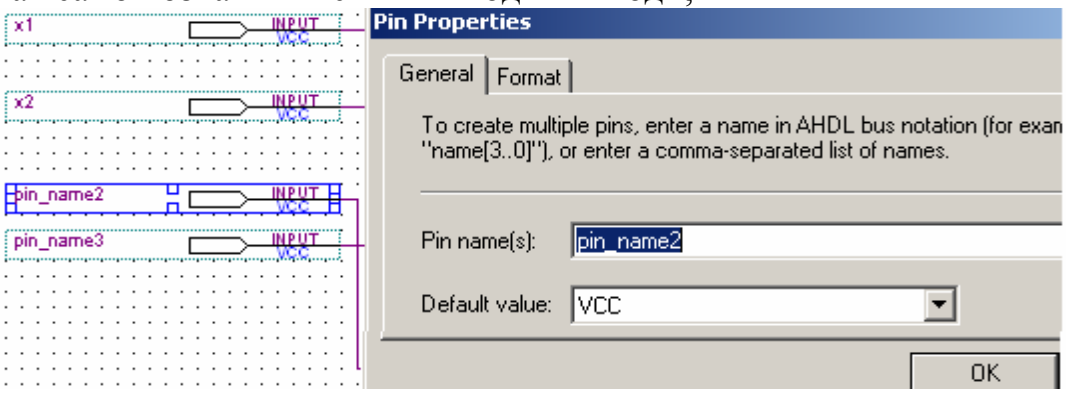
попередньо виділений елемент можна, як звичайно, *вирізати* (у буфер), *скопіювати*, *вставити* потрібну кількість разів у місцях клацання (у цьому або іншому файлі), *скасувати* останні дії або *відмінити* скасування (зігнені стрілки) та *видалити* без збереження в буфері (клавішею Delete);

✍ *Примітка:* З *контекстного меню* (викликається кнопкою В2) з виділеним елементом можна виконати такі самі операції.

ж) з'єднати логічні елементи між собою та з портами введенням з'єднувальних ліній за допомогою інструментів палітри: підвести курсор інструмента ортогональних ліній (або вибору) до виводу елемента чи порту і, коли він набуде форми хреста, натиснути ліву кнопку та протягнути лінію у вигляді горизонтальних і вертикальних відрізків, які закінчуються по відпусканні кнопки (точки з'єднань утворюються автоматично); для вилучення зайвого відрізка його слід виділити підведенням стрілки і клацанням лівою кнопкою та видалити; якщо під час редагування схеми потрібно пересунути елемент зі збереженням або без збереження з'єднувальних ліній, достатньо скористатися інструментом палітри Rubberbanding (метод гумової нитки); за необхідністю, вставити або вилучити точку з'єднань можна подвійним клацанням в місці з'єднання будь-яким інструментом з'єднувальних ліній (зображені з точками);



з) позначити назви входів і виходів: інструментом введення тексту клацнути символ (або виділити символ та натиснути інструмент Properties – властивості, або B2 > вибрати Properties у контекстному меню) і в діалоговому вікні Pin Properties (властивості штирка), на вкладці General (загальні) ввести ім'я, наприклад, x1 та натиснути OK; так само позначити всі інші входи і виходи;



✍ *Примітки:*

1) Таким самим чином можна скоригувати імена портів та інших компонентів, змінити їх ідентифікаційні номери (у лівому нижньому куту символу), а також ввести коментарі (клацнути інструментом введення тексту у вільному місці). При цьому на вкладці Format можна вибрати колір тексту, а для коментарів – також параметри шрифту на вкладці Font.

2) 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 файлу та, у разі потреби, замінити елементи чи внести інші корективи. Копію потрібного фрагменту графічного файлу можна вставити в документ Microsoft Word безпосередньо або за допомогою клавіші Print Screen через графічний редактор Paint (в останньому випадку ілюстрацію можна скоригувати).

и) дати ім'я файлу і зберегти його: піктограмою Save (або з меню File > Save) викликати діалогове вікно Save As, в якому підняти прапорець Add file to current project (додати файл до поточного проекту) та натиснути кнопку збереження – файлу автоматично надається ім'я проекту з розширенням .bdf.

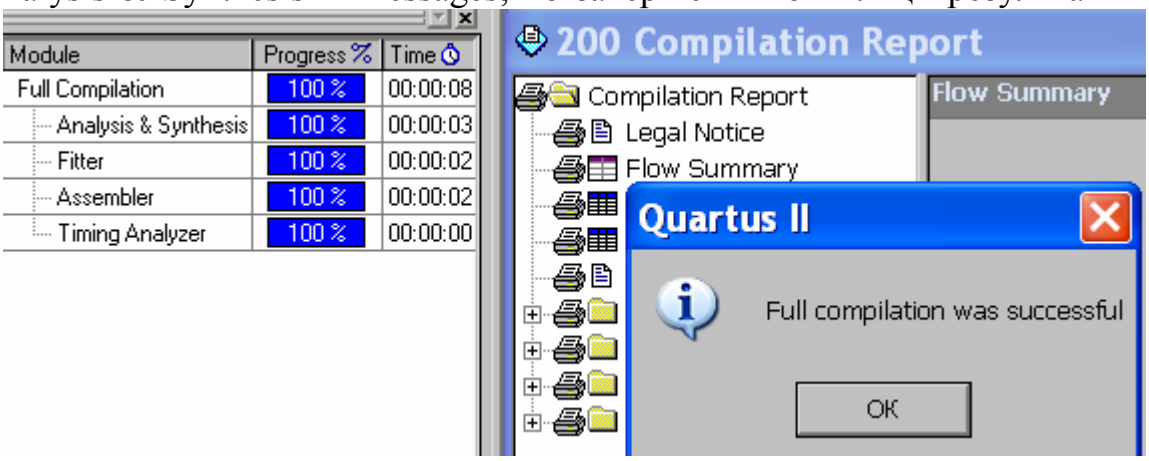
Приклад: \quartus6.1work\2lab\200.bdf (схема 1).

2 Виконати *компіляцію* проектованого пристрою:

а) якщо це не зроблено, відкрити проект: піктограмою Open Project (або з меню File > Open Project) викликати діалогове вікно Open Project, вибрати в ньому потрібні директорію і ім'я проекту (із синьою емблемою Quartus) та двічі клацнути ім'я або емблему – директорія з ім'ям проекту з'являться в рядку заголовка головного вікна Quartus II;

б) переконатися, що графічний файл включено до проекту: піктограмою Settings (або з меню Project > Add/Remove Files in Project – додати файли до проекту чи вилучити з проекту) викликати діалогове вікно Settings (налаштування) і на сторінці Files, у разі потреби, вибрати потрібні файли кнопкою огляду (…), додати їх кнопкою Add, виділити і вилучити зайві кнопкою Remove та натиснути кнопку OK;

в) піктограмою Start Compilation (або з меню Processing > Start Compilation) запустити компілятор – процес і час обробки програмних модулів відображаються в лівому вікні стану та в рядку стану головного вікна, в правому вікні з'являється частинами звіт про компіляцію Compilation Report, а в нижньому – повідомлення з розділу цього звіту Analysis & Synthesis > Messages; по завершенні компіляції результати по-



даються в інформаційному віконці, яке слід зачинити натисненням OK та значком закриття зачинити також ліве вікно стану (його можна будь-коли відчинити з меню View > Utility Windows > Status);

✍ *Примітки:*

1) По натисненні кнопки OK у зазначеному інформаційному віконці ліве вікно стану зачинятиметься автоматично, якщо ввімкнути опцію Tools > Options > Processing > Automatically close the Status window when processing is complete (автоматично зачиняти вікно стану по завершенні обробки).

2) Компіляція відбувається у фоновому режимі, що дозволяє під час її виконання працювати з іншими файлами. Проте обробка простих проектів займає небагато часу, тому в нашому випадку немає сенсу переходити до інших файлів.

г) у нижньому вікні повідомлень натиснути закладку Error і за наявності помилок виправити їх та знов виконати компіляцію; відтак натис-

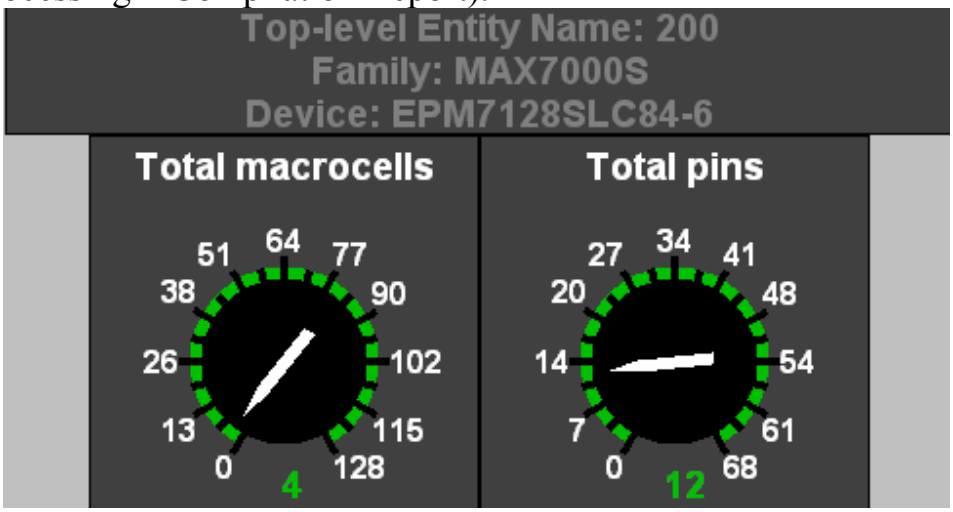
нути закладку Warning і за наявності застережень проаналізувати їх та, залежно від наслідків, або проігнорувати застереження, або скоригувати проект (в останньому випадку знов виконати компіляцію) і зачинити нижнє вікно повідомлень;

✍ *Примітки:*

1) Для локалізації місця помилки в проекті слід двічі клацнути відповідне повідомлення – відкриється файл, в якому виділиться це місце.

2) Аналіз повідомлення про помилку або застереження полегшується за допомогою формалізованої довідки. Для її виклику слід клацнути відповідне повідомлення і натиснути клавішу F1 – з'явиться стандартне вікно, в якому подаються розділи CAUSE (причина) та ACTION (дія, якщо вона потрібна для усунення цієї причини).

г) ознайомитися в загальних рисах зі звітом про компіляцію Compilation Report. У розділі Flow Summary (резюме) ознайомитися з використаним ресурсом мікросхеми: макрокомірок (Total macrocells) і виводів (Total pins) та значком закриття в рядку меню зачинити вікно звіту (його можна будь-коли відчинити піктограмою Compilation Report або з меню Processing > Compilation Report).



✍ *Примітка.* Для наочного відображення ресурсу у вигляді діаграм слід клацнути кнопкою B2 у полі розділу звіту Flow Summary і вибрати в контекстному меню Gauge Summary. Тепер, якщо клацнути кнопкою B2 всередині відповідної діаграми, можна вибрати відображення у величині або відсотках.

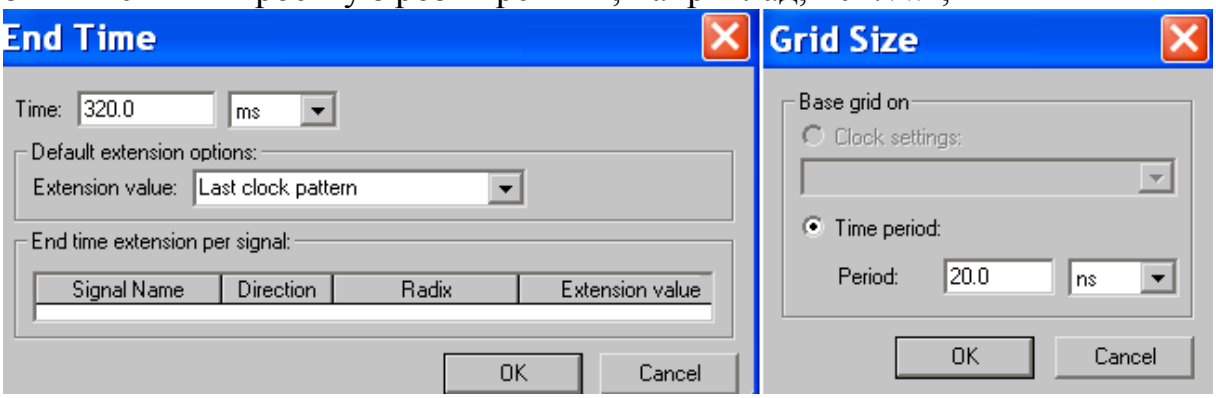
3 Виконати *функціональне моделювання* скомпільованого проекту.

3.1 ***Створити файл часових діаграм*** сигналів 2XX.vwf:

а) переконатися, що активізовано потрібний проект (інакше піктограмою відкриття проекту або з меню File ввести його директорію і ім'я до рядка заголовка головного вікна);

б) піктограмою горизонтальної панелі New Vector Waveform File (або з меню File > New > сторінка Other Files > Vector Waveform File) відкрити новий файл часових діаграм – з'явиться вікно з безіменною назвою WaveformN.vwf (де N – номер);

в) дати ім’я файлу і зберегти його: клацнути піктограму збереження, у діалоговому вікні Save As (зберегти як) підняти прапорець Add file to current project (долучити файл до поточного проекту) і натиснути кнопку збереження – у рядку заголовка файлу автоматично з’явиться ім’я проекту з розширенням, наприклад, 202.vwf;



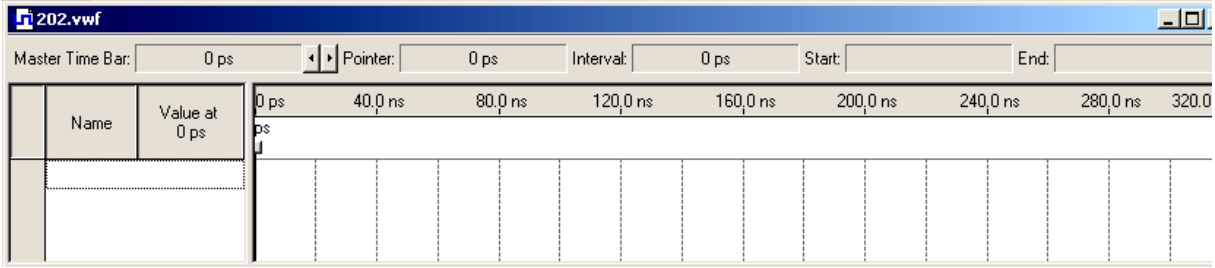
г) задати параметри часових діаграм згідно з таблицею відповідності (сітку моделювання доцільно вибрати кратною десяти, наприклад, Δt = = 20 *ns,* а весь інтервал моделювання – виходячи з кількості наборів вхідних змінних, наприклад, при чотирьох змінних $T = 2^4\Delta t = 16\Delta t = 320$ *ns*): меню Edit > End Time > ввести 320 ns (прокруткою вибрати одиницю вимірювання) > OK та меню Edit > Grid Size > Period 20 ns > OK, після чого піктограмою Fit in Window вписати все зображення у вікні – у порожньому файлі відобразиться весь інтервал з часовою сіткою.

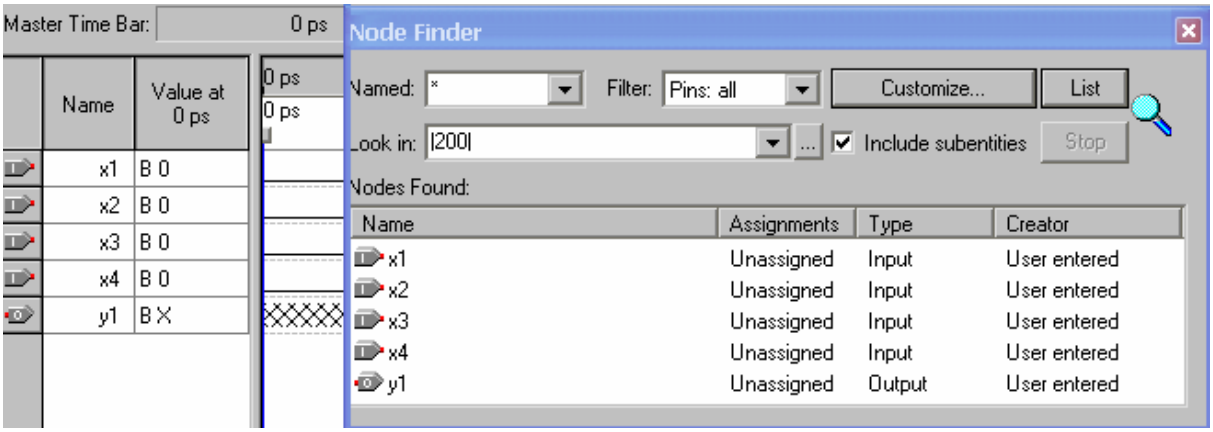
3.2 ***Ввести тестові вектори*** (на часових діаграмах позначити координати сигналів), створені компілятором:

а) піктограмою Node Finder (або з меню View > Utility Windows > Node Finder) викликати вікно засобу пошуку вузлів, тобто сигналів (ілюстрацію див. нижче);

б) у віконці Filter (вибір типу сигналів) прокруткою встановити Pins: all (всі зовнішні виводи IC) і натиснути кнопку List – у нижньому вікні Nodes Found (знайдені вузли) відобразиться перелік вхідних і вихідних сигналів;

в) у колонці Name виділити потрібні сигнали (клацнути позначки вузлів при натиснутій клавіші Shift для виділення всіх сигналів або при натиснутій клавіші Ctrl для вибіркового виділення), взятися за будь-який вузол і перетягнути до колонки Name свого файлу, в якому після цього відобразяться діаграми сигналів (на входах нульові рівні, а на виходах невизначені стани); відтак зачинити вікно Node Finder (повторно натиснути піктограму Node Finder або позначку закриття у вікні).



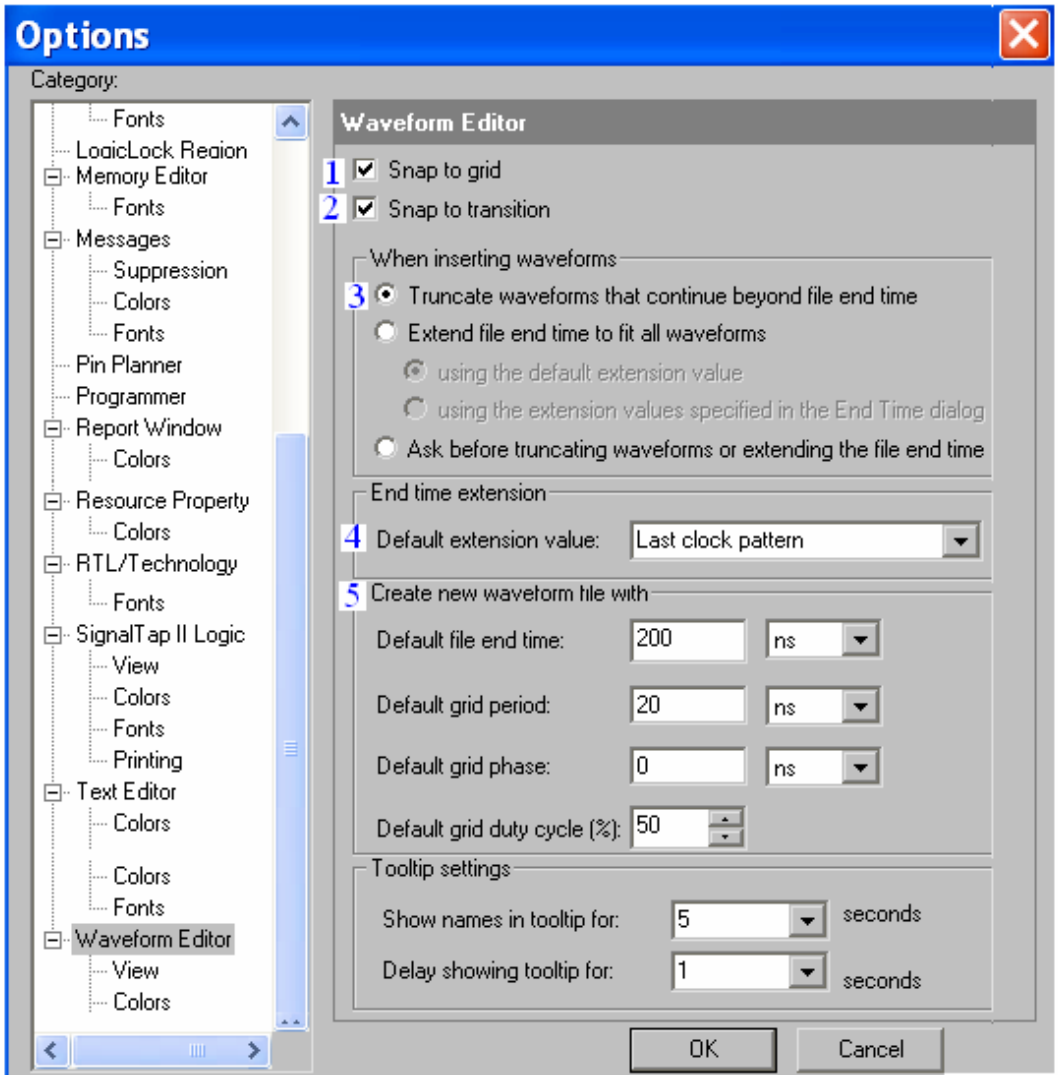


Примітка: Розташування сигналів на часових діаграмах легко змінити: досить виділити рядок (клацнути його в полі заголовків і відпустити кнопку миші), взятися за нього та перетягнути (з'явиться прямокутник біля покажчика) в потрібну позицію, яка висвічується кольоровою лінією.

3.3 ***Відредагувати часові діаграми*** вхідних сигналів за допомогою основних інструментів палітри:

а) налаштувати *інструментальну палітру* редактора часових діаграм (розташована вертикально ліворуч): меню Tools (інструментальні засоби) > Customize Waveform Editor (налаштувати редактор часових діаграм) > на сторінці Toolbars (інструментальні панелі) встановити облямовані кнопки великого розміру: зняти прапорець Borderless look (кнопки без облямівок), та підняти прапорці Show tooltips (показ типів інструментів) і Large buttons (великі кнопки) > на сторінці Commands вибрати категорію Waveform Editor, взятися за значок у полі Buttons і перетягнути його до відповідної позиції на палітрі та вилучити зайві кнопки перетягуванням їх із палітри до робочого поля файлу; відтак натиснути кнопку ОК у вікні Customize (налаштовану палітру з одного стовпця найуживаніших інструментів показано в горизонтальному вигляді на ілюстрації вверху);

б) налаштувати *опції* редактора часових діаграм, для чого з меню Tools > Options розкрити сторінку Waveform Editor діалогового вікна опцій, в якому ввімкнути прапорці таких опцій: 1) прив'язка до сітки – під час введення діаграми чітко розташовуються відносно координатної сітки і

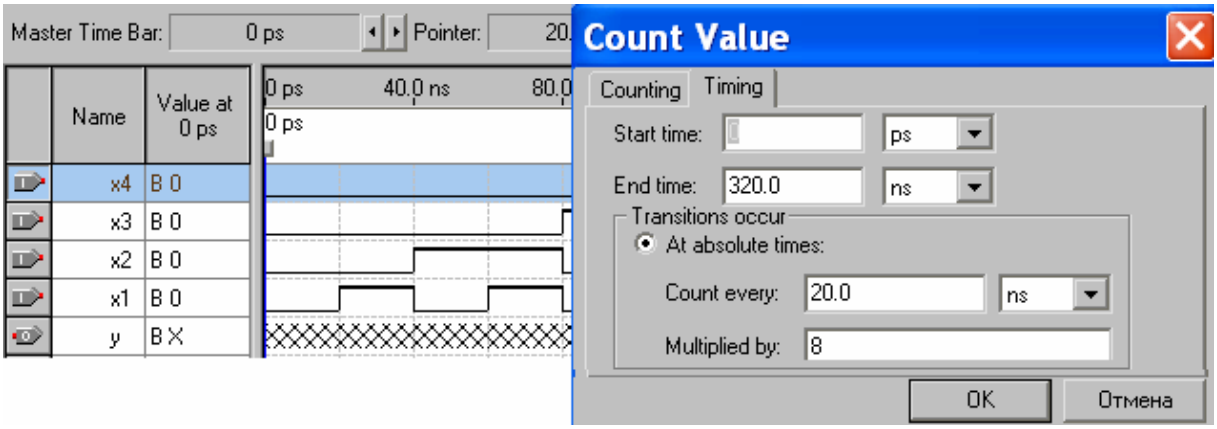


маркер часу під час переміщення зупиняється на лініях сітки (опцію слід вимкнути, якщо діаграми або маркер потрібно розташувати в довільних місцях – оперативно це можна зробити з меню View); 2) прив'язка до переходів (до стрибків на часових діаграмах) – дозволяє фіксувати між переходами часові інтервали під час їх виділення і фіксувати маркер на переходах; 3) продовжити часові діаграми, що утинаються кінцем часового інтервалу, поза межами файлу – після розширення часового інтервалу (з меню Edit > End Time) діаграми буде автоматично продовжено; 4) з чого продовжити часові діаграми після розширення – залишити за умовчанням значення з попереднього часового інтервалу; 5) створювати новий файл часових діаграм за умовчанням з таких налаштувань: кінцевий час моделювання встановити, наприклад, 200 *нс*, крок сітки 20 *нс*, сітку починати з 0 *нс*, коефіцієнт заповнення сигналу 50% (тобто шпаруватість імпульсів дорівнює 2, що відповідає формі меандру); відтак натиснути кнопку OK у вікні Options;

в) інструментом вибору виділити діаграму x1 (клацнути в полі заголовків) і натиснути інструмент циклічного повторення рівнів;

г) у діалоговому вікні значень параметрів Count Value, на сторінці вибору часу Timing, в рядку Count every ввести крок сітки 20 *ns* (одиниці вимірювання встановлюються прокруткою), залишити коефіцієнт множення Multiplied by 1 та натиснути кнопку OK – у файлі заповниться діаграма x1;

ґ) повторити п. *в*, *г* для всіх інших вхідних сигналів, збільшуючи кожний раз удвічі коефіцієнт множення Multiplied by та зберегти результати редагування.



Примітки:

1) Ввести значення змінних, зокрема, нециклічних, можна також в інший спосіб. Для цього слід протягнути інструментом виділення інтервалів або інструментом вибору над віссю змінної на одному кроці або кількох кроках часової сітки при натиснутій лівій кнопці – виділений інтервал заливається кольором. Відтак потрібне значення треба зафіксувати натисненням відповідної кнопки значень палітри (під час протягування інструментом виділення рівень автоматично змінюється на протилежний). У такий самий спосіб можна зінвертувати сигнал на потрібних ділянках часових діаграм. Для виправлення помилково введеного рівня слід так само виділити хибний інтервал і зафіксувати кнопкою потрібний рівень, наприклад, замінити лог. 1 на лог. 0.

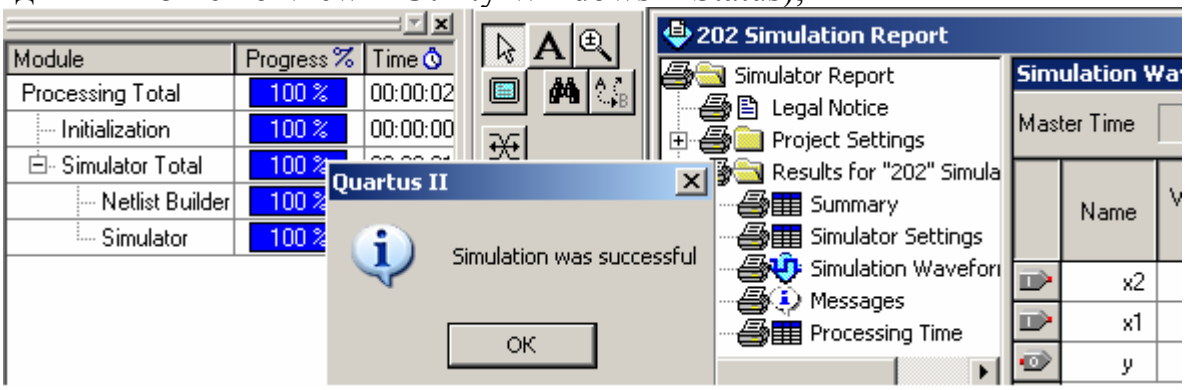
2) Командою з меню View > Snap to Grid (прив'язка до часової сітки) ділянки часових діаграм виділяються (отже і фіксуються на них рівні) на інтервалах, кратних кроку часової сітки. Для введення значень на довільних інтервалах незалежно від кроку сітки слід вимкнути зазначену команду.

3) Крок сітки можна змінити (для зручності введення або перегляду) у будь-який час, це не впливає на моделювання. Зображення сітки на екрані вмикається та вимикається опцією показу сітки з меню Tools > Options > Waveform Editor > View > Show time grid.

3.4 ***Змоделювати часові діаграми вихідних сигналів***:

а) піктограмою Start Simulation (або з меню Processing > Start Simulation) запустити імітатор – процес і час обробки програмних модулів відображаються в лівому вікні стану та в нижньому рядку стану головного вікна, в правому вікні з'являється частинами звіт про мо-

делювання Simulation Report, а в нижньому вікні – повідомлення з розділу цього звіту Simulator > Messages; по завершенні компіляції результати подаються в інформаційному віконці, яке слід зачинити натисненням OK та, у разі потреби (якщо не ввімкнено опцію згідно з приміткою 1 до п. 2,в), значком закриття зачинити також ліве вікно стану (його можна будь-коли відчинити з меню View > Utility Windows > Status);

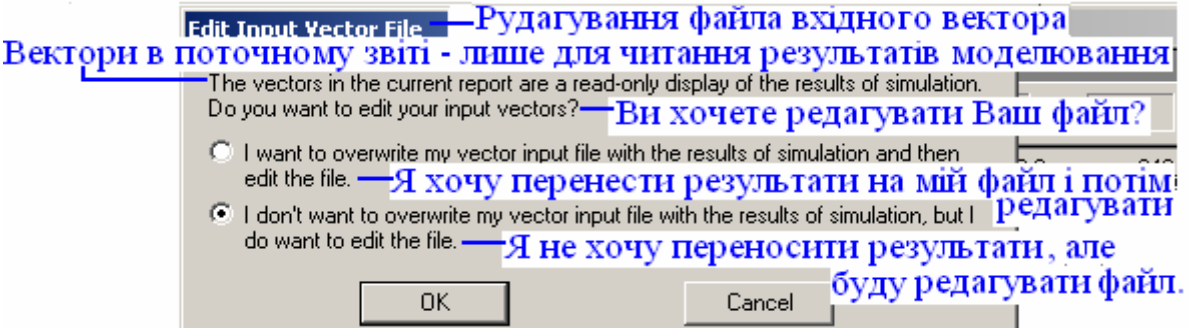


Примітка: Моделювання відбувається у фоновому режимі, що дозволяє під час його виконання працювати з іншими файлами. Проте обробка простих проектів займає небагато часу, тому в нашому випадку немає сенсу переходити до інших файлів.

б) у нижньому вікні повідомлень натиснути закладку Error і за наявності помилок виправити їх та знов виконати моделювання; відтак натиснути закладку Warning і за наявності застережень проаналізувати їх та, залежно від наслідків, або проігнорувати застереження, або скоригувати часові діаграми (в останньому випадку знов виконати моделювання) і зачинити нижнє вікно повідомлень;

в) розташувати часові діаграми у звітному вікні Simulation Waveforms піктограмою Fit in Window (вписати у вікно) для зручного їх перегляду (ознайомитися в загальних рисах з іншими розділами звіту про моделювання можна шляхом їх розгортання: клацнути по складнику, який з'явиться в правому вікні);

г) за відкритого вікна Simulation Waveforms клацнути в цьому вікні інструмент вибору інтервалу (або виділити у звітному файлі окремі потрібні сигнали клацанням у полі типу контактів за натиснутої клавіші Ctrl, взятися за один з них і почати перетягувати на свій файл) – одразу ж з'явиться діалогове вікно його редагування;



ґ) у цьому вікні ввімкнути перший перемикач (дати згоду перенести результати моделювання на свій векторний файл), натиснути OK та дати

згоду на запит про внесення змін до файлу – на WVF-файлі заповняться діаграми вихідних сигналів;

✍ *Примітка:* Результати моделювання автоматично переносяться на файл часових діаграм проекту, якщо з меню Tools > Options > General > Processing на сторінці Proces-sing ввімкнути прапорець Overwrite simulation input file with simulation result > OK. Проте така операція не завжди є зручна: втрачається можливість порівняння часових діаграм до і після внесення корекцій до проекту.

д) зачинити вікно звітного файлу; відчинити його для перегляду компонентів звіту можна в будь-який час піктограмою Simulation Report (або з меню Processing > Simulation Report).

✍ *Примітка:* Вставити коментарі до файлу часових діаграм або відредагувати існуючий текст можна інструментом Text Tool: натиснути зазначений інструмент, підвести його до робочого поля і, коли покажчик набуде форми хреста з літерою А, клацнути ним діаграму. Відтак клацнути утворену рамку і, коли вона виділиться темним, ввести до неї текст. Цим же інструментом стрілку і текст можна пересунути.

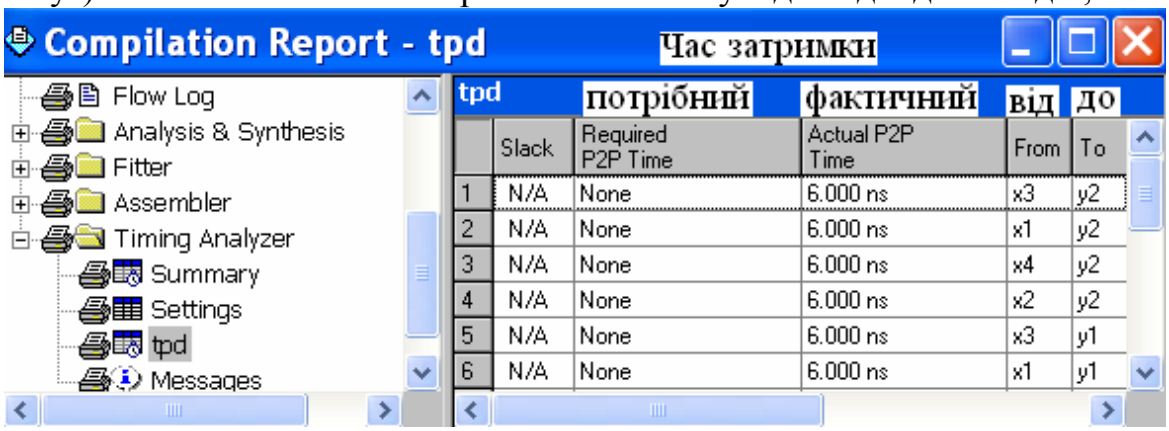
4. Проаналізувати результати проектування пристрою в різних елементних базисах примітивів.

4.1 Перевірити, чи узгоджується вихідна функція реалізованої схеми з таблицею відповідності на всіх наборах змінних, та вручну виміряти затримки поширення сигналів (див. Лаб. робота №1, п. 4.2).

4.2 ***В автоматичному режимі визначити швидкодію*** спроектованого пристрою:

а) у разі потреби, послідовністю наведених піктограм відкрити аналізований проект (переконатися, що його директорію і назву відображено в рядку заголовка головного вікна) та задля наочності відкрити файл верхнього рівня проекту (у нашому випадку це є графічний файл);

б) піктограмою Compilation Report (або з меню Processing > Compilation Report) відкрити вікно звіту про компіляцію і розгортанням складників звіту Timing Analyzer > tpd розчинити вкладку з часом затримки поширення сигналу між контактами мікросхеми tpd (Pin to Pin Delays) та ознайомитися із затримками сигналу від входів до виходів;



✍ *Примітка:* П. 4.2,б можна виконати запуском часового аналізатора піктограмою Timing Analyzer Tool (або з меню Processing > Timing Analyzer Tool), в однойменному вікні перейти на сторінку tpd та, у разі потреби, натиснути кнопку Start.

в) зберегти таблицю результатів часового аналізу в текстовому файлі: клацнути правою кнопкою миші будь-де в полі таблиці звіту (або з меню File > Save Current Report Section As) і викликати діалогове вікно Save Current Report Section As, в якому вибрати тип файлу Timing Analysis Output File (.tao) та натиснути кнопку збереження;

г) продивитися зазначений файл: натиснути піктограму відкриття файлу, прокруткою встановити тип файлів Output Files, вибрати зі списку потрібний файл та натиснути кнопку відкриття.

***Приклад*:** quartus6.1work\2lab\200.bdf (схема 1), 200.wvf, 200 tpd (Pin to Pin Delays).tao.

4.3 Доповнити той самий графічний файл схемами згідно із домашнім завданням, п. *г* (див. приклад: проект quartus6.1work\2lab\200), скомпілювати та виконати функціональне моделювання, доповнивши часові діаграми вихідними сигналами нових схем, переконатися в правильності проектування.

3 ЛАБОРАТОРНА РОБОТА №3. ПЕРЕТВОРЮВАЧІ КОДІВ

Мета роботи: дослідження типових перетворювачів коду – двійкових дешифраторів, дешифраторів 7-сегментного коду, шифраторів, перетворювачів між двійковим кодом і ДДК; побудова ЦКП на дешифраторах; засвоєння основ створення проекту на макрофункціях та настроювання макрофункцій; засвоєння методики розведення виводів ВІС на друкованій платі, її фізичного програмування та дослідження запрограмованого пристрою.

ДОМАШНЄ ЗАВДАННЯ

Для заданого варіанту (див. Додатки, завдання 3): а) спроектувати спеціальний перетворювач двійкового коду до коду семисегментного індикатора для відображення заданих знаків на дешифраторах різної розрядності і вибрати оптимальний варіант; б) розробити схему передачі даних заданої розрядності на основі шифратора і дешифратора.

СТИСЛІ ТЕОРЕТИЧНІ ВІДОМОСТІ

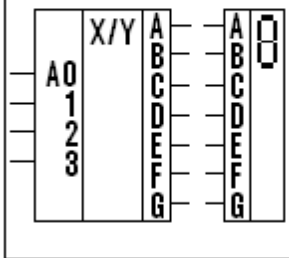
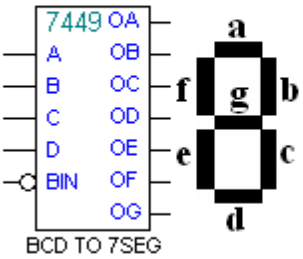
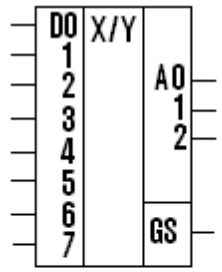
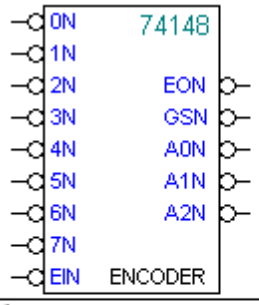
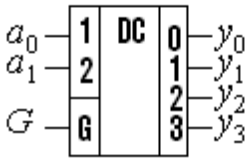
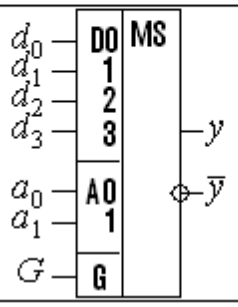
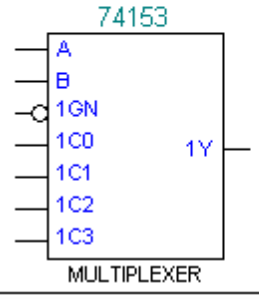
Перетворювачі кодів (ПК) є ЦКП, що здійснюють перетворення цифрової інформації з однієї форми зображення до іншої. Прикладом є стандартні ІС перетворювачів двійкового коду в ДДК та, навпаки, ДДК у двійковий. Окремим випадком ПК є пристрої, для яких вхідним або вихідним є так званий унітарний код “1 із К”, в якому активний рівень може існувати тільки в одному розряді. Якщо активним є рівень лог. 1, то код називають прямим, а якщо лог. 0 – інверсним. Прикладом є унітарний десятковий код “1 з 10” відображення натиснутої цифрової клавіші, якщо натиснення більш однієї клавіші заборонено. Якщо ж активний рівень може існувати в кількох розрядах, код “Х із К” називають пріоритетним, в якому старшим за пріоритетом за умовчанням є розряд з більшим номером. Унітарні коди використовуються в перетворювачах, які називаються дешифраторами і шифраторами. Дешифратори 7-сегментного коду, дешифратори і шифратори, а також елементи бібліотеки САПР розглянуто в [2], основні типи ЦПК наведено в табл. 3.1.

1 Перетворювачі між двійковим кодом і ДДК

Молодший розряд у двійкового коду і ДДК (див. [1]) збігається, тому мікросхеми виконують без його врахування і, отже, таблиці відповідності мають однаковий вигляд для парних і непарних вхідних слів: 0 і 1, 2 і 3, 4 і 5 і т. д. У типовій ІС перетворювача двійкового коду в ДДК (рис. 3.1,а) із шестирозрядного вхідного двійкового коду на виходах утворюються молодша тетрада ДДК $y_3 \ldots y_0$ і неповна старша тетрада $y_6 \ldots y_4$.

З метою збільшення розрядності вдаються до каскадування ІС. Так, з’єднанням трьох стандартних ІС утворюється перетворювач 8 розрядів двійкового коду в ДДК (на рис. 3.1,б наведено його символ, на принципову схему можна вийти з файлу 3libr.bdf), а для перетворення 16-розрядного двійкового коду в ДДК потрібно вже 16 таких ІС.

Таблиця 3.1 – Основні ЦКП

Позначення за ДЕСТУ	Типова макрофункція	Таблиця відповідності
Перетворювач (дешифратор) до семисегментного коду		
		i | a_3 a_2 a_1 a_0 | a b c d e f g 0 | 0 0 0 0 | 1 1 1 1 1 1 0 1 | 0 0 0 1 | 0 1 1 0 0 0 0 2 | 0 0 1 0 | 1 1 0 1 1 0 1 … | … | … 9 | 1 0 0 1 | 1 1 1 1 0 0 1
Шифратор пріоритетний		
		i | d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0 | a_2 a_1 a_0 | GS - | 0 0 0 0 0 0 0 0 | 0 0 0 | 0 0 | 0 0 0 0 0 0 0 1 | 0 0 0 | 1 1 | 0 0 0 0 0 0 1 X | 0 0 1 | 1 2 | 0 0 0 0 0 1 X X | 0 1 0 | 1 3 | 0 0 0 0 1 X X X | 0 1 1 | 1 4 | 0 0 0 1 X X X X | 1 0 0 | 1 5 | 0 0 1 X X X X X | 1 0 1 | 1 6 | 0 1 X X X X X X | 1 1 0 | 1 7 | 1 X X X X X X X | 1 1 1 | 1
Дешифратор-демультиплексор		
		i | G | a_1 a_0 | y_3 y_2 y_1 y_0 - | 0 | X X | 0 0 0 0 0 | 1 | 0 0 | 0 0 0 1 1 | 1 | 0 1 | 0 0 1 0 2 | 1 | 1 0 | 0 1 0 0 3 | 1 | 1 1 | 1 0 0 0
Мультиплексор-селектор		
		i | G | a_1 a_0 | y - | 0 | X X | 0 0 | 1 | 0 0 | d_0 1 | 1 | 0 1 | d_1 2 | 1 | 1 0 | d_2 3 | 1 | 1 1 | d_3

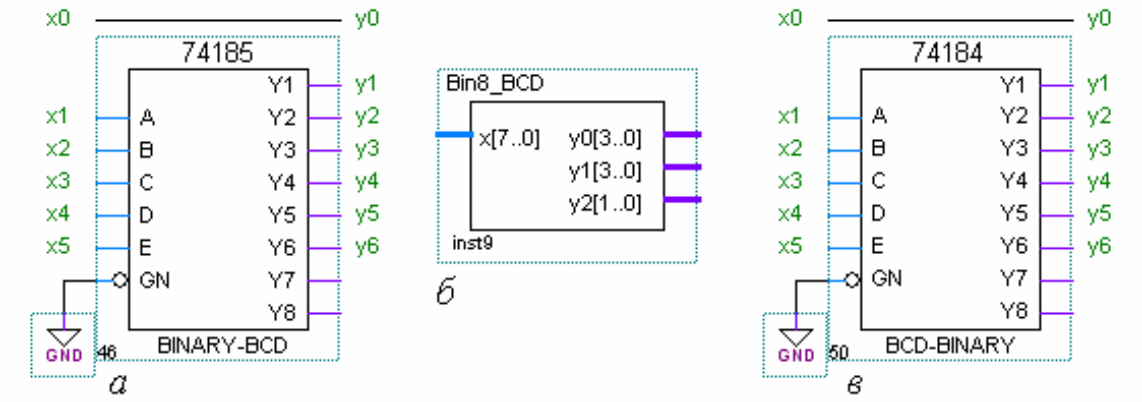


Рисунок 3.1 – Типові ІС

Зворотне перетворення від ДДК до двійкового коду здійснюється аналогічно. У типовій ІС перетворювача ДДК у двійковий код (рис. 3.1,в) із двох тетрад ДДК (молодшої тетради x_3 ... x_0 і неповної старшої x_5x_4) на виходах утворюється семирозрядний двійковий код. Два старші розряди на виходах обох ІС використовуються для перетворення до доповняльних кодів (приклад подано у файлі 3libr.bdf). Докладніше схеми перетворювачів до доповняльних кодів і каскадування різної розрядності наводяться в нормативних документах і довідковій літературі на ІС.

2 Проектування ЦКП на дешифраторах

Вихідні функції повного дешифратора являють собою набір усіх можливих мінтермів, тому сполученням виходів дешифратора можна реалізувати будь-яку логічну функцію від його вхідних змінних. На ПЛІС такий прийом використовувати доцільно, якщо він спрощує проектування, а на ІС жорсткої структури застосовувати дешифратори для реалізації простих функцій немає сенсу, крім випадку, коли один з них залишається в корпусі ІС невикористаним. Застосовування дешифраторів може виявитися доцільним для побудови *ЦКП з кількома виходами*, якщо функція по кожному з них зображається відносно простим сполученням вихідних функцій дешифратора. При цьому оптимальність рішення з'ясовується шляхом порівняння варіантів схеми на дешифраторах різної розрядності.

Приклад. Методику проектування ЦКП на основі дешифратора розглянемо на прикладі побудови перетворювача коду для відображення трьох знаків (Z) латиниці. За кількістю потрібних адрес визначаємо розрядність стандартного дешифратора 2:4 і складаємо таблицю відповідності (рис. 3.2,*а*), в якій зайва адреса $i = 3$ відповідає факультативним значенням вихідних функцій перетворювача *a*...*g*. Відтак виражаємо ці функції через вихідні функції дешифратора:

$$a = \overline{a_1}\,\overline{a_0} = y_1;\ b = \overline{d} = \overline{a_1}\,\overline{a_0} = y_0;\ c = \overline{g} = a_1\,\overline{a_0} = y_2;\ e = f = 1,$$

і згідно з виразами $a = \overline{y_1};\ b = y_0;\ c = g = \overline{y_2};\ d = \overline{y_0};\ e = f = 1$ будуємо схему (рис. 3.2,*б*).

Якщо сегменти 7-сегментного індикатора ввімкнено за схемою зі спільним анодом (на рис. 3.2,в зображено еквівалентну схему, наприклад, сегменту **а**), вони засвічуватимуться рівнем лог. 0, тому вихідні функції перетворювача *a*...*g* мають бути інверсними відносно наведених у таблиці на рис. 3.2,а, тобто $a = y_1;\ b = \overline{y_0};\ c = g = y_2;\ d = y_0;\ e = f = 0.$

Перевагою використання дешифратора з кількістю адресних входів, що дорівнює кількості змінних *m*, є відносна простота вихідних функцій, а недоліком – складність такого дешифратора і, головним чином, обмеженість кількості входів, адже при $m > 3...4$ вже необхідне каскадування дешифраторів на ІС жорсткої структури. З огляду на це може виявитися доцільним варіант побудови схеми на дешифраторі меншої розрядності.

i	a_1	a_0	*a*	*b*	*c*	*d*	*e*	*f*	*g*	Z
0	0	0	1	1	1	0	1	1	1	A
1	0	1	0	0	1	1	1	1	1	b
2	1	0	1	0	0	1	1	1	0	C
3	1	1	x	x	x	x	x	x	x	-

а

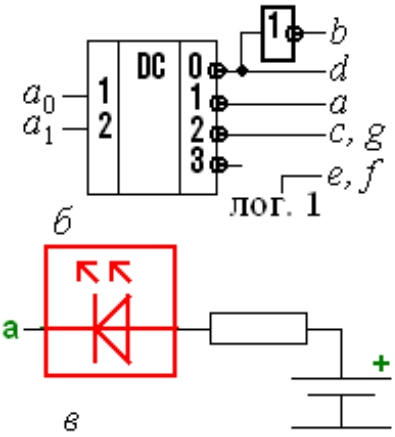


Рисунок 3.2 – Проектування ЦКП на дешифраторі

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Як слід каскадувати дешифратори з метою збільшення розрядності?

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

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

4. Спроектуйте на дешифраторах перетворювач коду, варіант якого задано числом у табл. 3.2, де в лівій колонці наведено вхідний код, а у верхньому рядку - вихідний. Наприклад, варіанту 8 відповідає перетворювач коду 8421 $\rightarrow$ 8421доп. У табл. 2 подано: ДДК 8421, $8421_{доп}$, 8421+3, 2421, 7421, ХГ - 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 3.2

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

ЛАБОРАТОРНЕ ЗАВДАННЯ

1 Ознайомитися з основними типами перетворювачів коду.

1.1 За принциповими електричними схемами *на логічних елементах* (файл 3prim.bdf) та осцилограмами сигналів (3prim.vwf) розглянути принцип побудови ***стробованого дешифратора 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.

1.2 Ознайомитися зі ***стандартними дешифраторами, шифраторами та перетворювачами коду*** серії 74 (макрофункціями): 1) повними дешифраторами 2:4, 3:8, 4:16 (перетворювачами двійкового, вісімкового і шістнадцяткового коду в унітарний) – файл 3libr.bdf, рис.1; 2) неповними дешифраторами 4:10 (перетворювачами кодів ДДК, Ґрея та з надлишком 3 в десятковий унітарний) – рис. 3.2; 3) дешифраторами (перетворювачами до) 7-сег-ментного коду – рис. 3.3; 4) пріоритетними шифраторами 8:3 і 10:4 (перетворювачами унітарного коду у вісімковий і ДДК) – рис. 3.4; 5) перетворювачами коду: двійкового до ДДК і ДДК до двійкового та їх застосуванням і каскадуванням (клацнути символ Bin8_BCD правою кнопкою > Open Design File). У звіті навести, принаймні, по одній макрофункції з кожної категорії і пояснити призначення та особливості входів і виходів.

✍ *Примітка:* Для виклику бібліотечного файлу з принциповою схемою макрофункції слід клацнути її символ правою кнопкою миші і вибрати Open Design File. Інші довідкові дані з таблицею відповідності включно є приступні з програмного пакету MAX+PLUS II.

2 Застосувати дешифратор для реалізації ЦКП за *графічного опису* проекту на прикладі створення заданого варіанту спеціального перетворювача до 7-сегментного коду.

2.1 У проекті 3XX зібрати на *логічних елементах* схему *дешифратора*, потрібного для реалізації перетворювача, виконати компіляцію і функціональне моделювання (див. л. р. № 2, п. 2, 3).

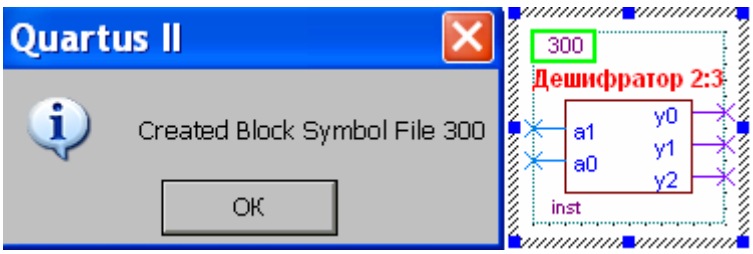
Приклад: неповний дешифратор 2:3, файли 300.bdf, .vwf.

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

2.2 ***Створити символьний файл*** шляхом згортання схеми дешифратора до символу:

а) відкрити файл, який треба згорнути до символу, в меню File вибрати Create / Update > Create Symbol File for Current File (створити символьний файл для поточного файлу), натиснути ОК у повідомленні про створення такого файлу; відтак відкрити його піктограмою відкриття файлу і вибором у діалоговому вікні типу файлів Graphic Files або Other Source Files (інші файли) та порівняти цей символ з принциповою електричною схемою;

Приклад: 300.bsf.



б) налаштувати *інструментальну палітру* символьного редактора (розташована вертикально ліворуч): меню Tools (інструментальні засоби) > Customize Symbol Editor (налаштувати символьний редактор) > на сторінці Toolbars (інструментальні панелі) встановити облямовані кнопки великого розміру: зняти прапорець Borderless look (кнопки без облямівок), та підняти прапорці Show tooltips (показ типів інструментів) і Large buttons (великі кнопки) > на сторінці Commands вибрати категорію Symbol Editor, взятися за значок у полі Buttons і перетягнути його до відповідної позиції на палітрі та вилучити зайві кнопки перетягуванням їх із палітри до робочого поля файлу; відтак натиснути кнопку OK у вікні Customize (нижче показано в горизонтальному вигляді налаштовану палітру з одного стовпця найуживаніших інструментів, аналогічних інструментам графічного редактора).

✍ *Примітки:*

1) Символ можна відредагувати графічно, користуючись інструментами палітри: а) задля компактності змінити його розмір (підвести курсор до вузлів зовнішнього прямокутника або клацнути сторону для виділення внутрішнього прямокутника і підвести курсор до його вузлів та, коли курсор набуде форми подвійної стрілки, протягнути його в потрібному напрямку); б) змінити позиції написів і виводів (клацнути елемент і перетягнути його до потрібного місця); в) інструментами Text Tool або ліній різного типу вставити написи чи інші графічні позначення і інструментом Properties відредагувати шрифт, колір, тип ліній, тло (див. малюнок до п. 2.2,а).

2) Зінвертувати порти можна аналогічно п. 2.3,в.

2.3 ***Створити графічний проект на макрофункції*** дешифратора для реалізації заданого варіанту перетворювача до 7-сегментного коду:


а) дати ім’я новому проекту 3XX_7seg (XX – варіант) з директорією власної теки, наприклад, \RT\3br\302_7seg;

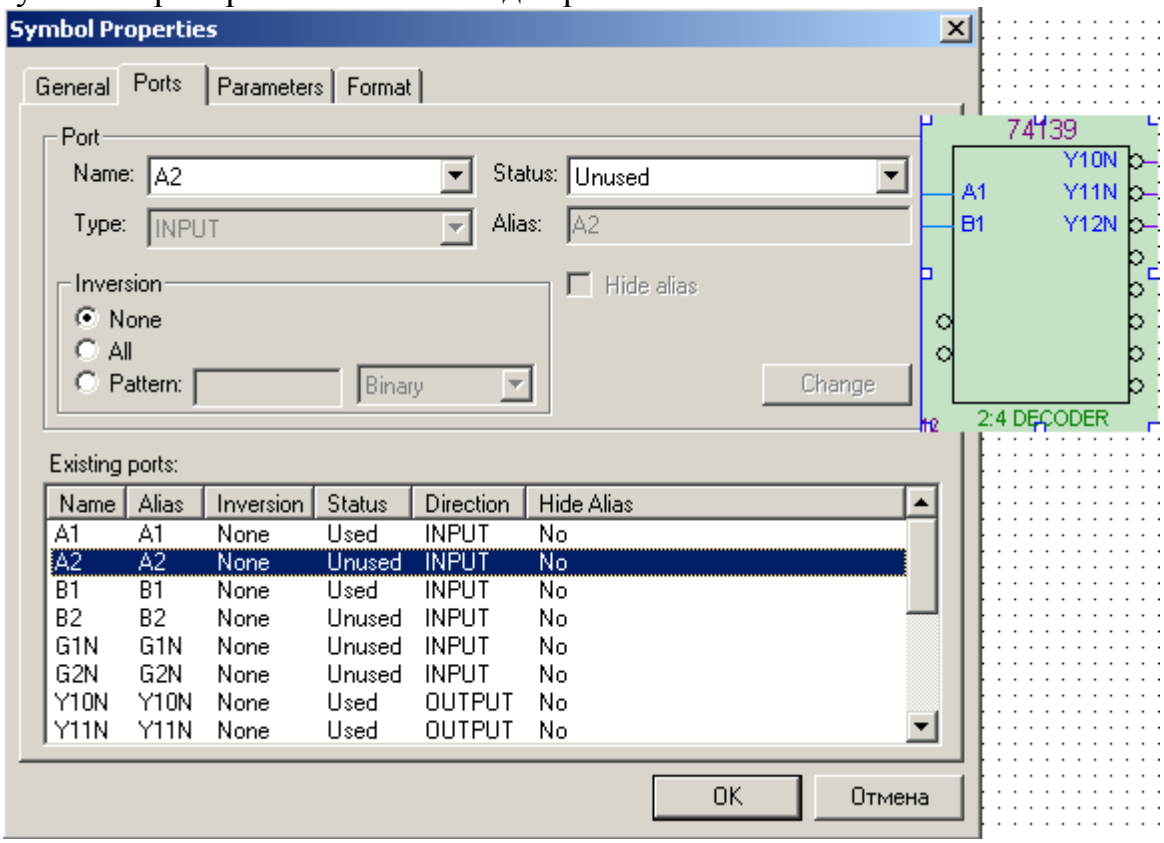

б) створити графічний файл 3XX_7seg.bdf аналогічно Л. р. № 2, п. 1.2, для чого з бібліотеки дібрати потрібну IC серії 74 (див. Л. р. № 3, п. 1.2) і вставити її розгортанням \quartus\libraries\others\maxplus2\74N (де N – тип IC, наприклад, 74139о);


в) виділити символ вставленої макрофункції і піктограмою Properties (або двічі клацнути символ, або B2 > Properties) викликати діалогове вікно Symbol Properties (властивості символу) та ***налаштувати*** її: на вкладці Ports (порти) виділити в списку існуючих портів (Existing ports) вивід, якій треба відредагувати, наприклад, A2 і у верхньому віконці Status вибрати Unused (невикористовуваний); так само можна

змінити інверсію на вході або виході: перемикач Inversion залишити в положенні None, якщо інверсія не змінюється або в положенні All, якщо змінюється на протилежну (Pattern використовується для шин); таким чином слід налаштувати всі виводи та натиснути ОК (для прикладу на ілюстрації вгорі наведено настроєну макрофункцію 74139);

✍ *Примітка:* Налаштувати параметри вікна графічного редактора можна з меню Tools > Options (див. л. р. №2, п. 1.2,в).

г) доповнити схему іншими потрібними елементами, виконати компіляцію і функціональне моделювання та перевірити правильність функціонування пристрою за часовими діаграмами.

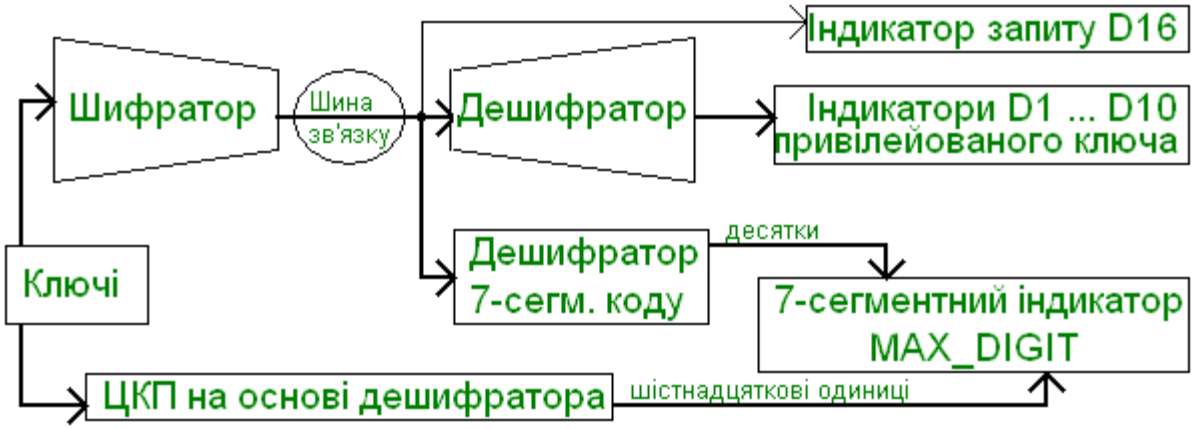


Приклад: файли 300_7seg.bdf (схема 1), .vwf.

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 до графічного файлу, як звичайно, але зі списку Project > ім'я символу діалогового вікна Symbol), виконати компіляцію і функціональне моделювання та перевірити правильність функціонування пристрою за часовими діаграмами.

Приклад: файли 300_7seg.bdf (схема 2), .vwf.

3 Створити лабораторний макет для експериментального дослідження заданого варіанту перетворювачів коду (див. структурну схему і рис. Д1 у додатках – файл 300PERETWOR_KODU.bdf).



Структурна схема

Від ДІП-перемикачів MAX_SW1, SW2 код типу X із К (активні рівні сигналу можуть діяти одночасно в кількох розрядах) надходить на входи пріоритетного шифратора, на виходах якого формується двійковий код, що відображає адресу (номер) старшого за пріоритетом ключа, а також сигнал запиту GSN, який набуває активного рівня за ввімкнення хоча б одного ключа (цей сигнал формується або самим шифратором, або поза ним додатковою логікою). Згорнутий з більшої до меншої кількості розрядів код (для передачі інформації шиною зв'язку між віддаленими блоками) розгортається на виходах шини в дешифраторі до унітарного коду 1 із К, отже, старший за пріоритетом ключ індикується одним зі світлодіодів D1 … D10. Одночасно для індикації в цифровому вигляді адреси привілейованого запиту його код через дешифратор 7-сегмнтного коду подається на старше (десятки) знакомісце 7-сегмнтного індикатора MAX_DIGIT. Сигнал запиту індикується світлодіодом D16, а за відсутності запиту гасяться цей світлодіод і розряд цифрового індикатора по входу BIN дешифратора 7-сегмент-ного коду. Додатково ключами перемикача MAX_SW2 досліджується цей дешифратор по входу перевірки світіння сегментів LTN, по входу RBIN і виходу RBON (світлодіод D15) послідовного гасіння старших нулів багаторозрядного індикатора, а також керування світінням десяткової крапки dpd.

Спроектований ЦКП на основі дешифратора (у прикладі спеціальний дешифратор 7-сегментного коду для відображення заданих шістнадцяткових цифр) досліджується за допомогою ключів 6 ... 8 (у прикладі ключі 7, 8) перемикача MAX_SW2 і молодшого (одиниці) знакомісця 7-сегментного індикатора MAX_DIGIT.

3.1 Дати ім'я новому проекту згідно із заданим варіантом і ***виконати призначення мікросхеми*** для його реалізації.

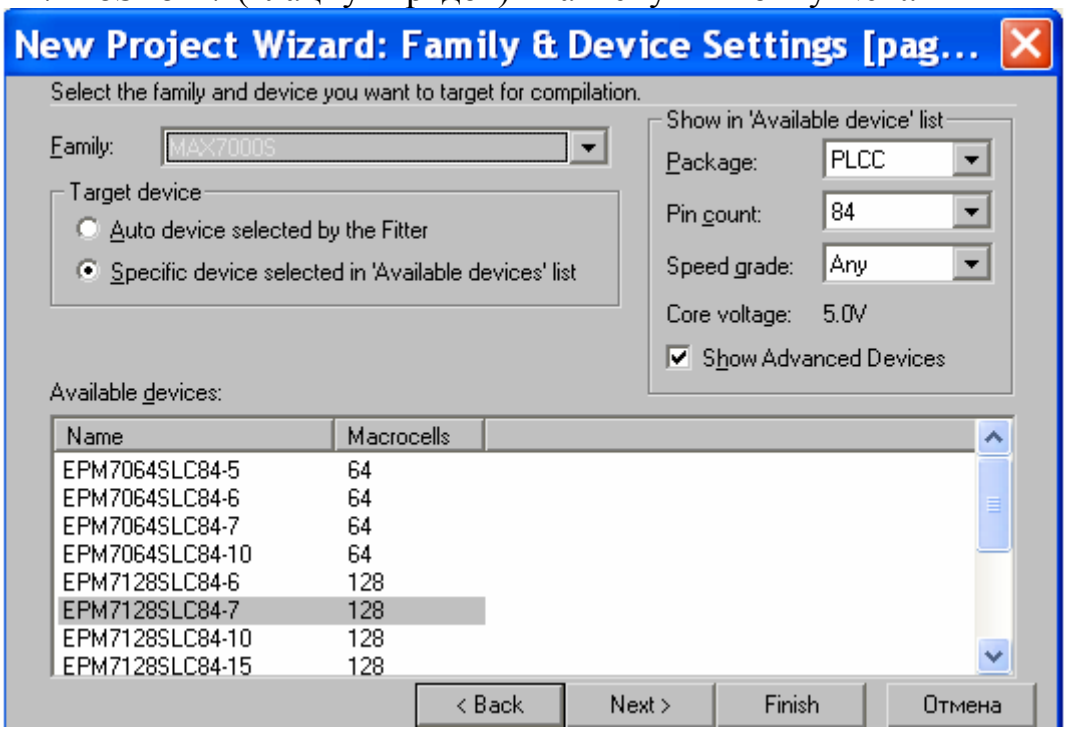
а) На першій сторінці діалогового вікна New Project Wizard ввести, як звичайно, директорію власної теки і ім'я нового проекту 3XXPERETWOR_KODU (XX – варіант) та натиснути кнопку Next.

б) У повідомленні, що ця директорія вже містить проект, на пропозицію змінити її дати негативну відповідь.

в) На другій сторінці кнопкою огляду (…) у діалоговому вікні Select File вибрати директорію і ім'я раніше створеного (п. 2.1) файлу дешифра-

тора 3XX.bdf, подвійним клацанням (або натисненням кнопки збереження) внести його до рядка File name вікна New Project Wizard і кнопкою Add включити до складу проекту – файл буде додано до списку у вікні File name. Відтак натиснути кнопку Next.

г) На третій сторінці (ілюстрацію див. нижче) вибрати: у рядку Family – родину IC, розміщену на платі, MAX7000S (залишити ввімкненим перемикач Specific device selected in 'Available devices' list – вибрати мікросхему зі списку приступних мікросхем); у розділі Show in 'Available devices' list – такі параметри (прокруткою): Package (тип корпусу) PLCC, Pin count (кількість виводів) 84, Speed grade (градація швидкодії) Any (довільна); у списку Available devises (приступні IC) – тип мікросхеми на платі EPM7128SL84-7 (клацнути рядок) і натиснути кнопку Next.



г) Четверту сторінку проминути і натиснути кнопку Next, а на п’ятій, останній сторінці продивитися підсумки і натиснути кнопку Finish – ім’я нового проекту з’явиться в рядку головного вікна.

Примітка: Виконати призначення мікросхеми для реалізації проекту можна будь-коли, для чого піктограмою Settings (або з меню Assignments > Settings) викликати діалогове вікно Settings, в якому на вкладці Category вибрати розділ Device і заповнити цю ж саму сторінку, що і в п. 3.1,г. Якщо вибір мікросхеми скориговано після компіляції, потрібно перекомпілювати проект (якщо не виконувати призначення мікросхеми взагалі, компілятором буде призначено за умовчанням довільну мікросхему).

3.2 Виконати проектування в графічному редакторі.

а) Створити графічний файл верхнього рівня, дібрати і вставити до нього потрібні макрофункції пріоритетного шифратора, дешифратора і дешифратора 7-сегмнтного коду та налаштувати їх щодо розрядності і інвертування входів/виходів. Вставити потрібні порти, дати їм імена та ви-

конати з'єднання компонентів схеми.

б) Доповнити схему спроектованим за п. 2.5 спеціальним дешифратором 7-сегментного коду для відображення заданих шістнадцяткових цифр (скопіювати його з файлу 3XX_7seg.bdf, схема 2 та перейменувати порти).

✐ *Примітка:* На всі незадіяні сегменти обох знакомісць індикатора і незадіяні світлодіоди необхідно подати лог. 1 (VCC) для їх погашення.

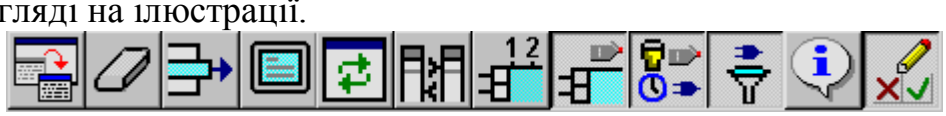
в) Виконати компіляцію і функціональне моделювання проекту.

Приклад: 300PERETWOR_KODU.bdf, .vwf.

3.3 ***Засвоїти використання редактора призначень*** Assignment Editor з метою з'єднання виводів мікросхеми із зовнішніми колами згідно з таблицями розведення на друкованій платі (див. файл ../3lab/ROZWED_MAX).

1) Якщо це не зроблено, відкрити потрібний проект та, задля наочності, файл його верхнього рівня.

2) Піктограмою Assignment Editor (або з меню Assignments > Assignment Editor) викликати діалогове вікно редактора призначень, з меню Tools > Customize Assignment Editor > на сторінці Toolbars встановити облямовані кнопки великого розміру (зняти прапорець Borderless look та підняти прапорці Show tooltips і Large buttons) > на сторінці Commands вибрати категорію Assignment Editor та ознайомитися з основними інструментами його палітри (у разі потреби, зайві інструменти можна вилучити звичайним чином). Палітру стовпця інструментів показано в горизонтальному вигляді на ілюстрації.



3) Інструментом Category Bar (або з меню View > Category Bar) ввімкнути розділ Category і вибрати в ньому категорію призначень виводів Pin (прокруткою або розгортанням розділу).

4) У разі необхідності, інструментом показу імен виводів (або з меню View > Show All Known Pin Names) розгорнути список портів електронної таблиці.

5) Покажчиком (який набуває форми хреста) двічі клацнути в комірці електронної таблиці на перетині рядка з потрібним портом та стовпця Location (місцеположення) і надрукувати тільки цифри з номером виводу мікросхеми (або клацнути цей номер зі списку, що розгортається) – у комірці позначиться номер виводу та у стовпці General Function – його загальна функція I/O (вхід / вихід). Таким чином виконати призначення для всіх портів згідно з файлом ../3lab/ROZWED_MAX.

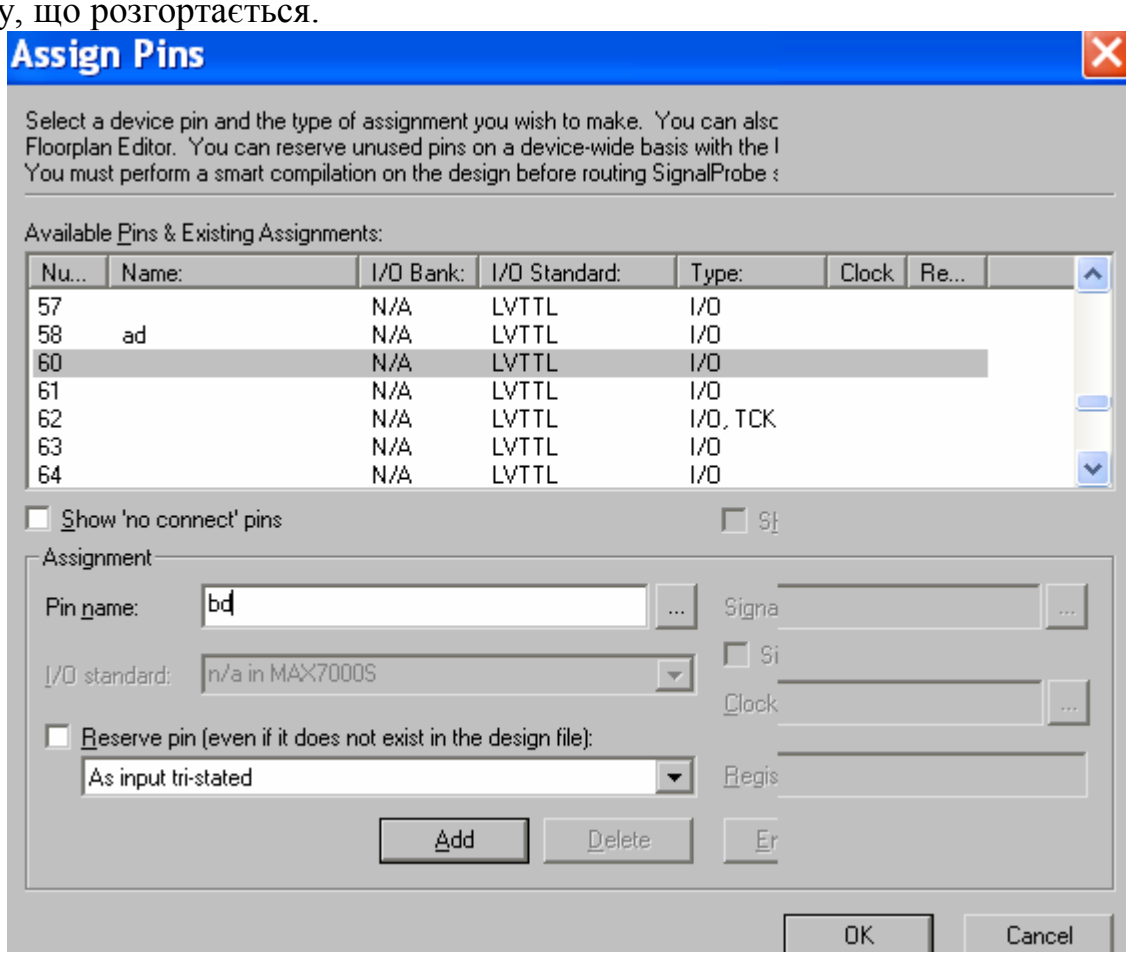
	To	Location	General Function
44	VD11	PIN_46	I/O
45	VD12	PIN_48	I/O
46	VD13	PIN_49	I/O
47	VD14		
48	<<new>>	<<new>>	

7) Зберегти файл редактора призначень – таблиці з'єднань відобра-

зяться біля портів графічного файлу. Відтак повторно виконати компіляцію (у термінах САПР – перекомпілювати проект).

✍ Примітки:

1) Таким самим чином можна відредагувати помилки: виділити комірку і ввести новий номер. Для редагування імені порту достатньо двічі клацнути комірку у стовпці To і надрукувати нове ім'я або вибрати зі списку, що розгортається.



2) Виконати з'єднання виводів мікросхеми із зовнішніми колами можна і в інший спосіб: піктограмою Settings (або з меню Assignments > Settings) викликати діалогове вікно Settings, в якому на вкладці Category вибрати розділ Assign Pins (призначити виводи) – з'явиться однойменне діалогове вікно. У розділі Available Pins & Existing Assignments (приступні виводи та існуючі призначення) прокруткою слід вибрати номер виводу (стовпець Nu…), клацнути його рядок для виділення, а в розділі Assignment (призначення) до рядка Pin name ввести назву цього виводу та натиснути кнопку Add (додати) – у виділеному рядку попереднього розділу додасться це ім'я у стовпці Name. Повторенням цих дій призначити всі виводи та натиснути кнопку ОК.

3) Таблиці з'єднань можна розташувати в позиціях, зручних для наочного відображення зв'язків, звичайним чином: взятися лівою кнопкою миші за таблицю і перетягнути її або виділити таблицю і пересунути з клавіатури клавішами зі стрілками керування курсором.

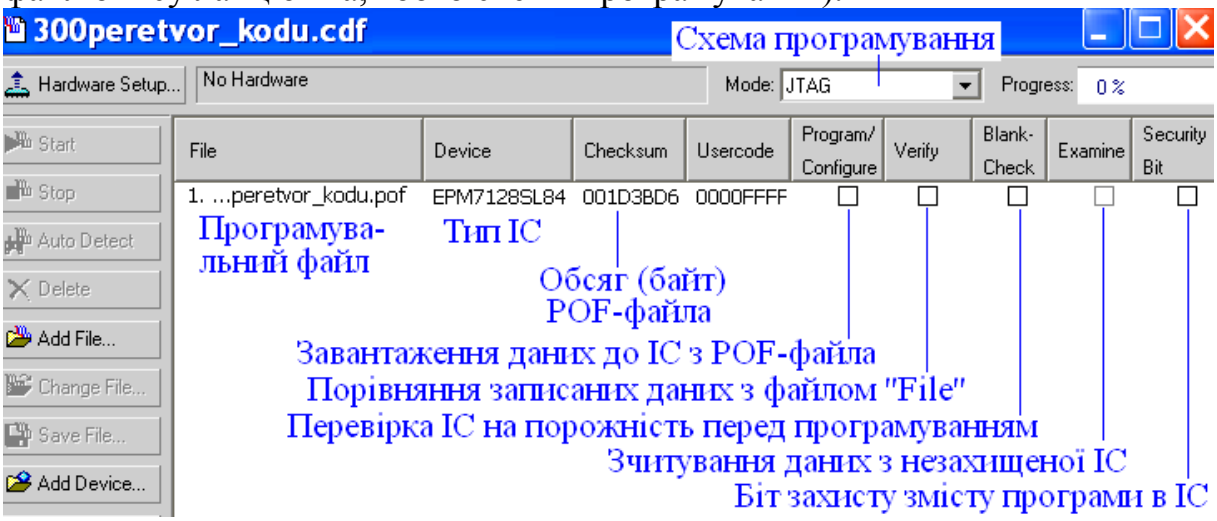
4) Таблиці з’єднань біля символів портів можна відобразити або приховати інструментом панелі Show Location Assignments (або з меню View > Show Location Assignments).

Приклад: 300PERETWOR_KODU.bdf.

3.4 ***Сформувати файл програматора.***

а) Відкрити, якщо це не зроблено, належний для реалізації скомпільований проект та (задля наочності) файл вершини ієрархії.

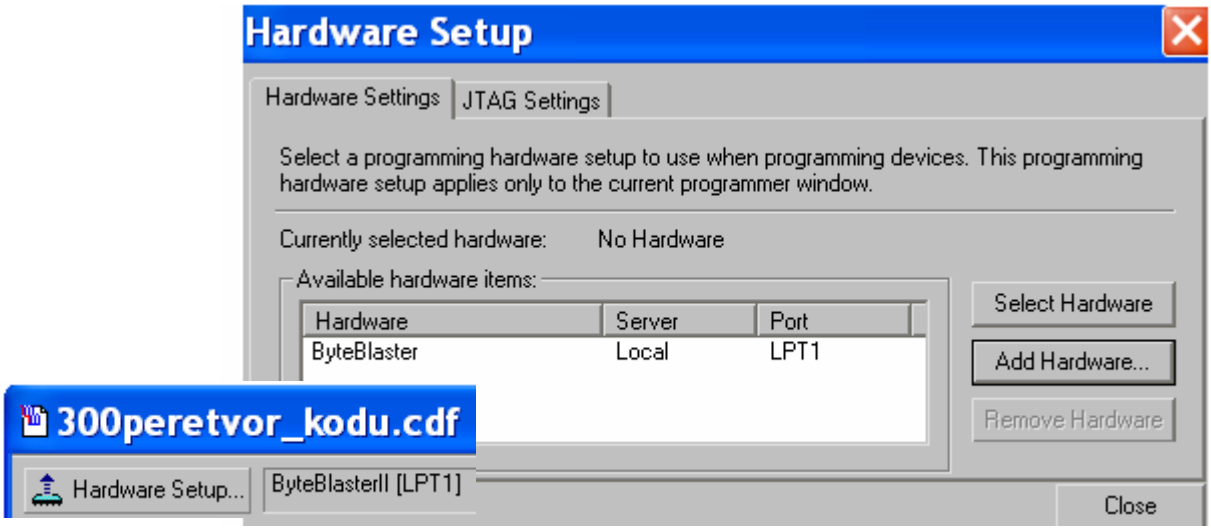
б) Піктограмою Programmer (або з меню Tools > Programmer) відкрити вікно програматора для визначення схеми фізичного програмування – з’явиться файл з розширенням .cdf (Chain Description File – файл опису ланцюжка, тобто схеми програмування).



в) У рядку Mode вибрати стандарт JTAG схеми програмування.

г) Якщо у верхньому рядку не встановлено апаратний засіб ByteBlaster II (зазначено No Hardware або іншій засіб), натиснути кнопку Hardware Setup (установка апаратного засобу) і в однойменному вікні (ілюстрацію див. нижче), на сторінці Hardware Settings переконатися, що встановлено конфігураційний пристрій ByteBlaster (якщо на сторінці Hardware Settings буде відсутній рядок з пристроєм ByteBlaster, слід вставити його кнопкою Add Hardware: в однойменному вікні прокруткою вибрати тип пристрою і порту та натиснути кнопку ОК). Відтак у розділі Available hardware items (приступні апаратні засоби) ***виділити рядок*** з пристроєм ByteBlaster, ***натиснути кнопку*** Select Hardware (вибрати апаратний засіб) та ***зачинити вікно*** Hardware Setup – у верхньому рядку вікна програматора з невеликою затримкою в часі встановиться апаратний засіб і в дужках порт комп’ютера: ByteBlaster II [LPT1].

✍ *Примітка:* Попередньо інсталювати в комп'ютері, якщо це не зроблено, програмний драйвер принтера.



г) У вікні програматора ввімкнути (клацнути прямокутники) функцій Program/Configure, Verify і Blanc-Check (функція Examine є приступною за використання спеціальних конфігураційних мікросхем), ***залишити вимкненою функцію Security Bit, записати код користувача Usercode*** та зберегти файл.

3.5 ***Виконати фізичне програмування мікросхеми.***

а) Переконатися, що на платі UP2 чотири штирьові триконтактні перемикачі (джампери) TDI, TDO, DEVICE, BOARD встановлено в режим *програмування* однієї мікросхеми – всі джампери з'єднують два верхні контакти C1 і C2 (як на ілюстрації у файлі ../3lab/ROZWED_MAX), інакше перемкнути їх.

б) З'єднати апаратним конфігураційним пристроєм ByteBlaster II рознімач JTAG_IN на платі UP2 з портом LPT1 (портом для з'єднання з принтером) персонального комп'ютера.

в) З'єднати спочатку кабелем DC_IN плату UP2 з джерелом живлення і лише після цього ввімкнути джерело в мережу 220 *В*. Наявність зовнішньої напруги індикується на платі зеленим світлодіодом POWER.

г) У вікні програматора натиснути кнопку Start. Миготінням світлодіода TCK на платі і в рядку Progress CDF-файлу індикується передача даних під час програмування, а світінням світлодіода CONF_D – завершення цього процесу.

ґ) Натиснути кнопку OK у вікні з повідомленням про успішне програмування.

3.6 ***Дослідити пристрій на запрограмованій ІС.***

4 ЛАБОРАТОРНА РОБОТА №4. ЦИФРОВІ КОМУТАТОРИ

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

ДОМАШНЄ ЗАВДАННЯ

Для заданого варіанту (див. Додатки, завдання 4): а) спроектувати ЦКП для реалізації логічної функції на мультиплексорах різної розрядності та вибрати оптимальний варіант; б) розробити схему для дослідження цифрових комутаторів.

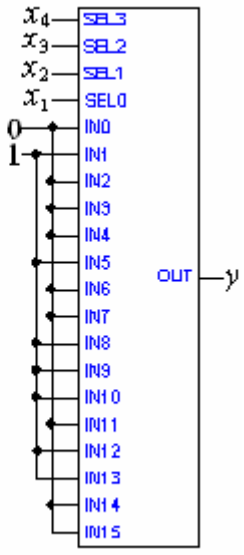
СТИСЛІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Цифрові комутатори, що здійснюють перемикання сигналів з одного каналу в декілька (демультиплексори) або, навпаки, з кількох каналів в один (мультиплексори), розглянуто в [2], де наведено також необхідні для розуміння САПР відомості про елементи арифметики, групи й шини та мегафункції (табл. 4.1, 4.2).

Проектування ЦКП на мультиплексорах

i	x_4	x_3	x_2	x_1	y
0	0	0	0	0	0
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	0
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	0

а



б

Рисунок 4.1 – УЛМ чотирьох змінних

Вихідна функція мультиплексора являє собою ДДНФ логічної функції від змінних a_i, якщо d_j є її значення на відповідних наборах (мінтермах) цих змінних. Отже, з'єднанням з адресними входами мультиплексора змінних, а з інформаційними входами – констант згідно з таблицею відповідності можна реалізувати будь-яку логічну функцію.

Утворений пристрій стає, таким чином, універсальним логічним модулем (УЛМ). УЛМ чотирьох змінних для прикладу наведено на рис. 4.1. На адресні (селекторні – SEL) входи мультиплексора 16:1 подаються чотири змінні, тому на виході OUT з'являється рівень, що відповідає значенням змінних, які діють у цей час. Наприклад, коли $x_4x_3x_2x_1 = 0101$, тобто $i = 5$, цим адресним кодом до виходу підмикається п'ятий вхід, що відповідає значенню $y = 1$.

На програмованих ВІС такий прийом використовувати доцільно, якщо він спрощує проектування, а на ІС жорсткої структури застосовувати мультиплексор для реалізації окремої функції може виявитися доцільним, якщо при цьому зменшується складність за критерієм потрібної кількості

корпусів ІС. З метою зменшення складності бажано застосовувати мультиплексори щонайменшої розрядності, якщо це надмірно не ускладнює схему на інформаційних входах, а оптимальність рішення з'ясовується шляхом порівняння варіантів схеми на мультиплексорах різної розрядності.

Методику проектування ЦКП на мультиплексорах меншої розрядності розглянемо на прикладі логічної функції чотирьох змінних (див. рис. 4.1,а).

1) За діаграмою термів y на рис. 4.2,а мінімізуємо функцію

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

і визначаємо ступінь її залежності від аргументів: змінні x_1, x_2, x_4 входять до виразу y двічі та x_3 – один раз.

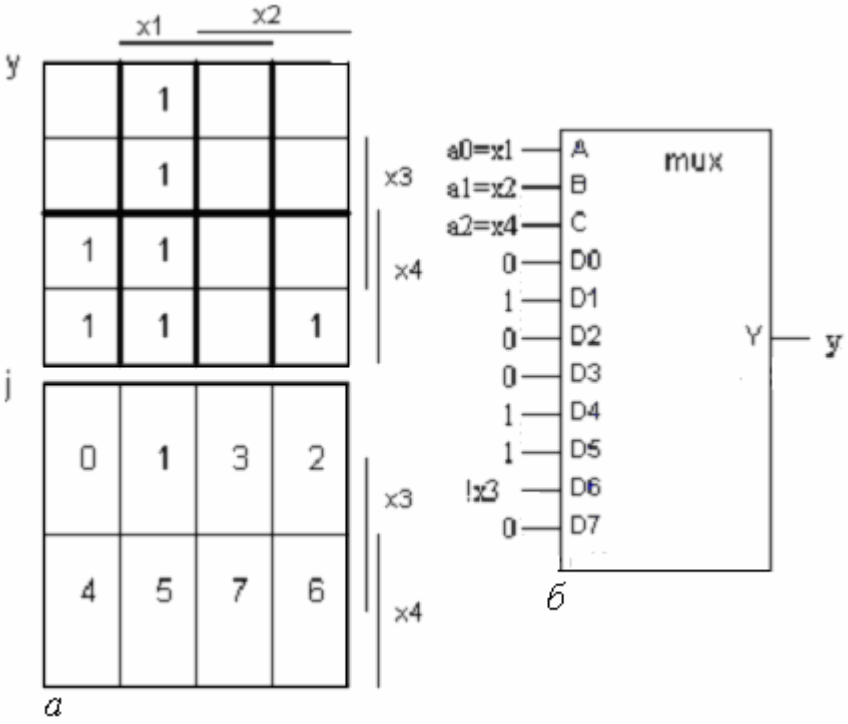


Рисунок 4.2 – Методика проектування ЦКП на мультиплексорі

2) Вибираємо мультиплексор з кількістю адресних входів k на одиницю менше кількості змінних (у прикладі $k=m-1=3$, тобто мультиплексор 8:1). До адресних входів приєднуємо змінні, від яких функція залежить більшою мірою, тобто формуємо адресний код j (у прикладі $j = a_2a_1a_0 = x_4x_2x_1$) і розбиваємо діаграму функції y на 2^k піддіаграм. Для цього у кожну клітинку діаграми j (на рис. 4.2,а нижня діаграма) вписуємо десятковий код $j = x_4x_2x_1$ відповідно до рисок цих змінних, не звертаючи уваги на аргумент x_2, що не входить до адреси. Отже, діаграма виявляється розбитою на 8 піддіаграм (з двох клітинок кожна), які і визначають номери j інформаційних входів мультиплексора, що з'єднуються з виходом y.

3) Визначаємо сигнали $d_j(x_3)$ на входах мультиплексора як функції вільної змінної, не приєднаної до адресних входів. Для цього зміст клітинок часткових діаграм з номерами j зчитуємо з діаграми y: $d_0=d_2=d_3=d_7= 0$, $d_1=d_4=d_5=1$, $d_6=\overline{x_3}$.

4) Згідно з функціями d_j будуємо ЦКП (рис. 4.2,б). Легко переконатися, що пристрій функціонує відповідно до діаграми термів. Наприклад, коли $x_4x_2x_1 = 0101$, тобто $i = 5$ маємо $y_2 = d_5 = 1$. Отже, схема виявляється вдвічі простішою відносно рис. 4.1,б – потрібен лише один додатковий інвертор на вході d_6.

5) Аналогічно проектуємо ЦКП на мультиплексорі 4:1 меншої розрядності – з двома адресними і чотирма інформаційними входами. До адресних входів приєднуємо дві змінні, від яких функція залежить більшою мірою (у прикладі один з трьох рівноцінних за цим критерієм варіантів), тобто формуємо адресний код $j = a_1a_0 = x_2x_1$ і розбиваємо діаграму функції y на чотири піддіаграми (рис. 4.3,а). Розглядаючи кожну з піддіаграм як діаграму термів функції двох змінних $d_j(x_3, x_4)$ і мінімізуючи звичайним чином, визначаємо сигнали: $d_1 = 1$, $d_3 = 0$, $d_0 = x_4$, $d_2 = \overline{x_3}\, x_4$. Шляхом порівняння з іншими двома варіантами адресного коду вибираємо перший (рис. 4.3,б) і, порівнюючи його з ЦПП на рис. 4.2,б, вважаємо кращим, бо він потребує простішого мультиплексора без надмірного ускладнення кіл для формування вхідних сигналів (потрібно по одному елементу НЕ та І).

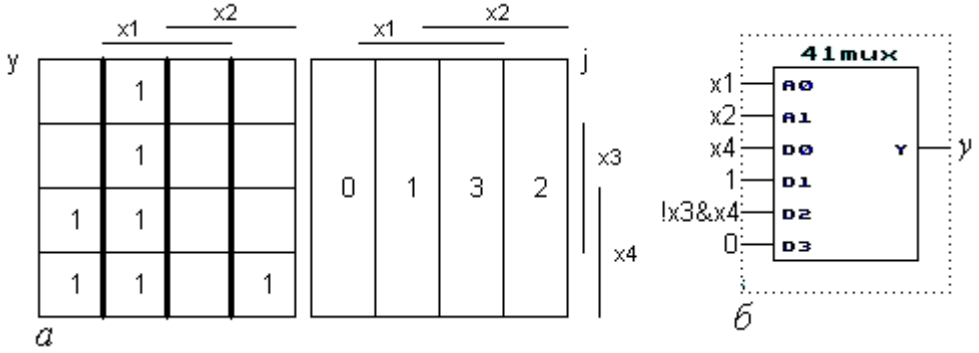


Рисунок 4.3 – Проектування ЦКП на мультиплексорі 4:1

І, нарешті, так само проектуємо ЦКП на мультиплексорі 2:1 (рис. 4.4,а), вибираючи варіант адресного входу $j = a = x_1$: $d_1 = \overline{x_2}; d_0 = \overline{x_2 x_3}\, x_4$ (рис. 4.4,б). З порівняння можна дійти висновку, що остання реалізація ЦКП потребує менше ресурсу ІС, ніж попередні.

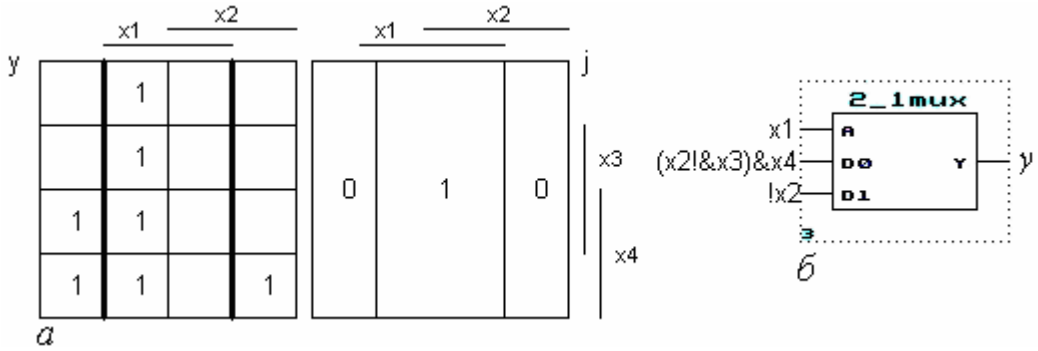
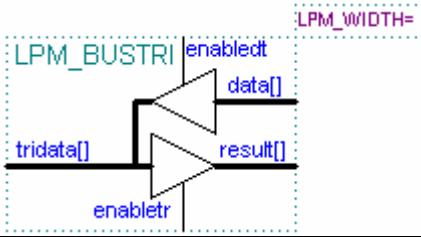
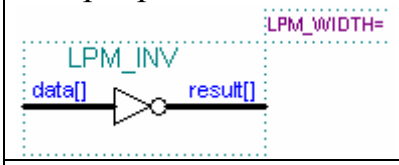
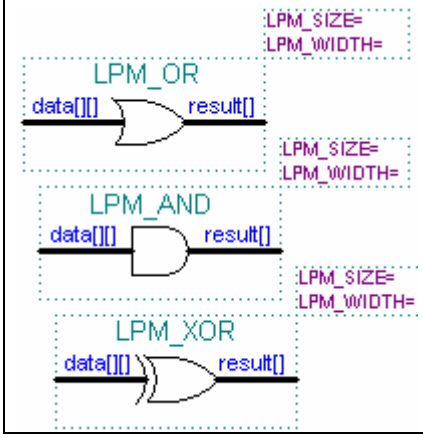


Рисунок 4.4 – Проектування ЦКП на мультиплексорі 2:1

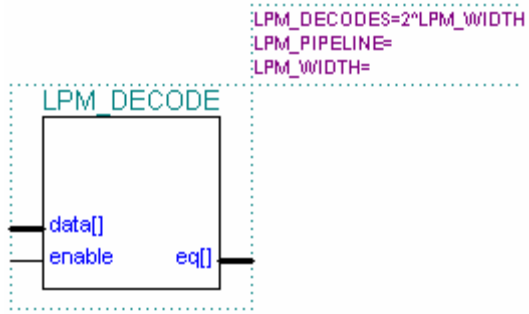
Слід відзначити, що якщо в логічному виразі функції можна винести за дужки спільний аргумент x_i, то ЦКП спрощується наданням стробовому входові значення $G = x_i$ при використанні мультиплексора-селектора зі звичайним (без третього стану) виходом. Крім того, якщо логічні функції на окремих інформаційних входах мультиплексора виявляються досить складними, то їх у свою чергу також можна реалізувати на мультиплексорах. По-

будова ЦКП з кількома виходами зводиться до проектування кількох пристроїв для кожної функції окремо. Якщо з цією метою скористатися ІС мультиплексорів зі спільними адресними входами, цим входам доцільно надати ті змінні, які найбільшу кількість разів входять до обох функцій.

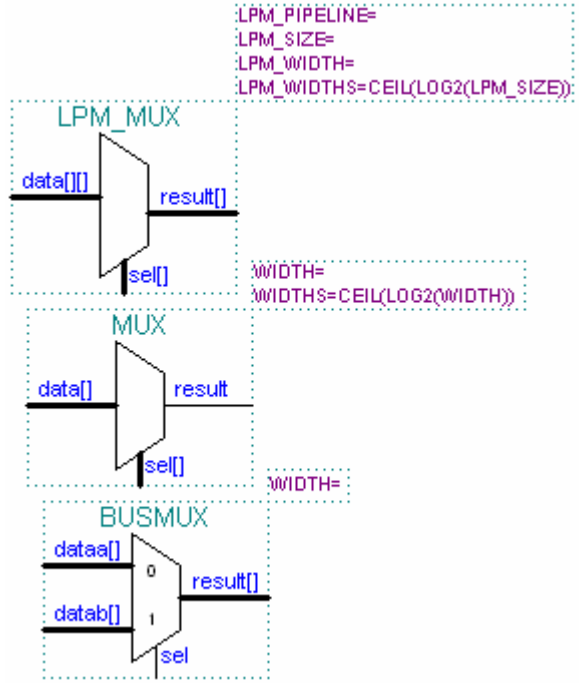
Таблиця 4.1 – Мегафункції логічних елементів

<table>
<tr><th>Символ</th><th colspan="3">Основні параметри</th></tr>
<tr><td rowspan="2"> 
</td><td>enabledt</td><td>enabletr</td><td>Напрямок передачі</td></tr>
<tr><td>0
0
1
1</td><td>0
1
0
1</td><td>tridata[]=Z; result[]=Z
tridata[] → result[]
data[] → tridata[]
data[] → tridata[],
data[] → result[]</td></tr>
<tr><td>Інвертори
 
</td><td colspan="3">lpm_bustri (Buffer), lpm_inv (NOT Gate):
LPM_WIDTH – розрядність шин і кількість елементів у кожній з них (буферів з трьома станами або інверторів).</td></tr>
<tr><td>Логічні елементи
 
</td><td colspan="3">lpm_or, lpm_and, lpm_xor:
LPM_WIDTH – кількість елементів масиву, вхідних шин і розрядність вихідної шини;
LPM_SIZE - кількість входів кожного елемента, тобто розрядність вхідних шин.
Приклад: LPM_SIZE = 2, LPM_WIDTH =3, data[LPM_SIZE-1..0][LPM_WIDTH-1..0] = =data[1..0][2..0], result[LPM_WIDTH-1..0]= result[2..0].</td></tr>
</table>

Таблиця 4.2 – Мегафункції комбінаційних пристроїв

<table>
<tr><th>Символ</th><th>Основні параметри</th></tr>
<tr><td>Дешифратор
 
</td><td>lpm_decode (Decoder):
LPM_WIDTH – розрядність двійкового вхідного коду;
LPM_DECODES – розрядність унітарного вихідного коду;
для повного дешифратора:
LPM_DECODES=2^ LPM_WIDTH.
Приклад: LPM_ WIDTH = 3, дешифратор 3:8, data[LPM_WIDTH-1..0] = data [2..0], eq[LPM_DECODES-1..0] = eq[7..0].</td></tr>
</table>

Продовження таблиці 4.2

Мультиплексори 	lpm_mux (Multiplexer): LPM_WIDTHS – розрядність адресної шини sel[]; LPM_SIZE ≤ 2^ LPM_WIDTHS – кількість вхідних шин і розрядність вихідної шини result[]; LPM_WIDTH – розрядність кожної вхідної шини. Приклад: LPM_WIDTHS = 3, LPM_WIDTH = 4, LPM_SIZE = 8, мультиплексор 8:1, sel[LPM_WIDTHS-1..0] = sel[2..0], data[LPM_SIZE-1..0][LPM_WIDTH-1..0]= = data[7..0][3..0], result[LPM_WIDTH-1..0] = result[3..0]. Окремі випадки: mux – перемикач ліній, busmux – перемикач двох шин.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Як слід каскадувати мультиплексори і демультиплексори з метою збільшення розрядності?

2. Як зв’язані між собою кількість адресних і інформаційних входів у мультиплексорі і демультиплексорі?

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

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

ЛАБОРАТОРНЕ ЗАВДАННЯ

1 Дослідити типові комутатори цифрових сигналів.

1.1 За принциповою електричною схемою та осцилограмами сигналів (4prim.bdf, .vwf) ***мультиплексора і демультиплексора на логічних елементах*** скласти таблицю відповідності і рівняння вихідних функцій, виміряти затримку вихідних імпульсів. У звіті навести схеми, умовні графічні позначення за ДСТУ, таблиці відповідності, рівняння вихідних функцій, часові діаграми, затримки, стислі пояснення принципу дії таких ЦКП.

1.2 Ознайомитися з ***макрофункціями*** стандартних мультиплексорів серії 74, у тому числі з трьома станами виходу (файл 4libr_macro.bdf, рис. 4.1...4.4), а також дешифраторів, які можуть виконувати функції демультиплексорів (файл 3libr.bdf, рис. 1). У звіті навести символи з різних груп, пояснити призначення та особливості входів і виходів.

1.3 Ознайомитися з дібраними елементами бібліотеки ***мегафункцій*** (файл 4libr_mega.bdf): 1) буфер, 2) логічні функції, 3) дешифратор, 4) мультиплексори та пояснювальними прикладами 1...10. Розглянути виконувані функції в довідці: Help > Megafunctions/LPM > вибрати категорію та ім’я мегафункції; дати тлумачення сенсу кожної з чотирьох груп мегафункцій і їх параметрів. У звіті навести принаймні по одному символу з кожної групи, пояснити призначення та особливості входів і виходів та дати стисле тлумачення основних параметрів.

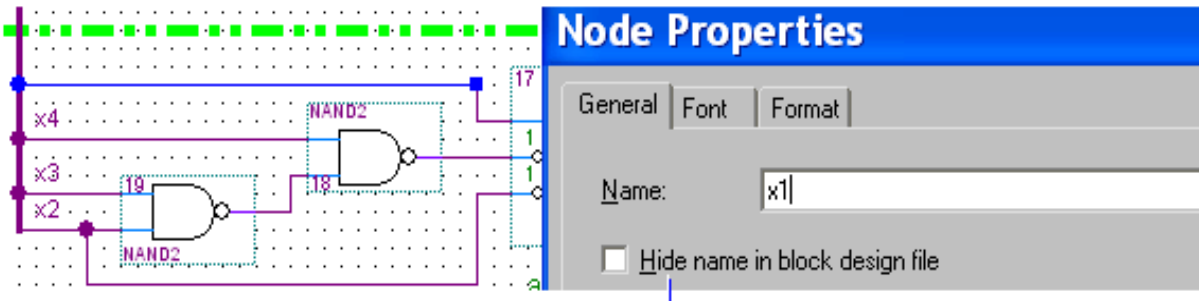
2 Застосувати *макрофункції* мультиплексорів для реалізації заданого варіанту (XX=01, 02, …) логічної функції *y*.

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: 400gr_macro.bdf (схеми 1...4), .vwf.

✐ Примітки:

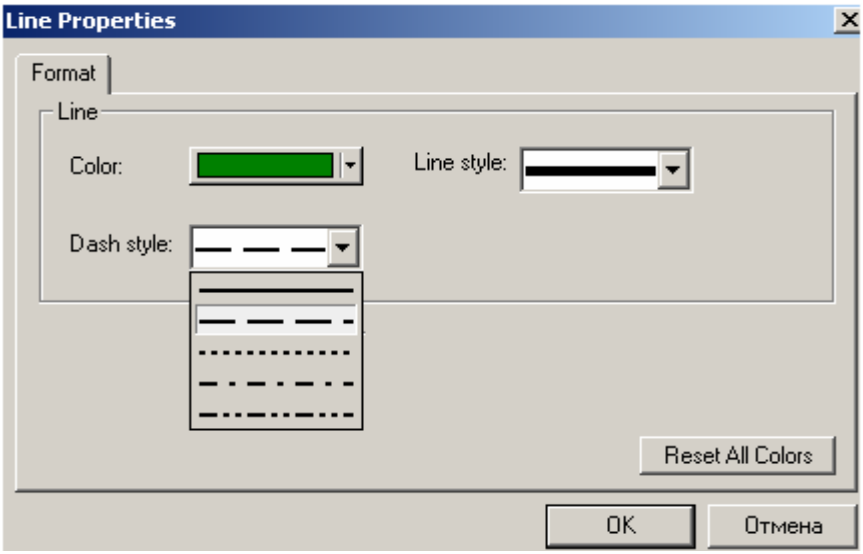
1) У випадку розгалужених шин треба вводити їх імена, а також імена з’єднаних з ними ліній: інструментом вибору виділити лінію, натиснути інструмент Properties (властивості) і в діалоговому вікні Node Properties (властивості вузла) ввести текст і натиснути ОК (або ввести його у вільному місці і перетягнути інструментом стрілка в потріб-



Приховати ім'я в графічному проектному файлі

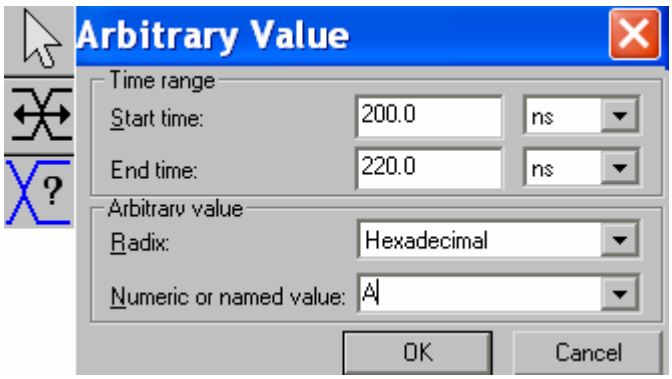
ну позицію). При цьому назви розгалужених шин (ліній) мають збігатися з назвами з'єднуваних ними вузлів (входів, виходів), інакше узгоджуємо імена за допомогою буферів Wire, які ресурсу не потребують, бо реально не існують. Приклад: на другій схемі сигнал x3 перейменовано в d6 для узгодження його імені в шинах x[] і d[].

2) 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 лівої палітри інструментів). Приклад: чотири порти x[4..1] приєднано до чотирьох дротів однойменної шини. Як видно з VWF-файла, після компіляції група набуває скороченої назви x, яка розгортається на компоненти x[4], x[3], x[2], x[1].



3) На відміну від з'єднувальних провідників і шин, довільну лінію можна відредагувати, якщо виділити її і викликати діалогове вікно Line Properties, в якому вибрати колір та стиль рисок і ліній та натиснути OK.

4) Компіляцію і моделювання доцільно виконувати кроками: спочатку змоделювати схему 1, відтак доповнити її схемою 2 і повторити моделювання.

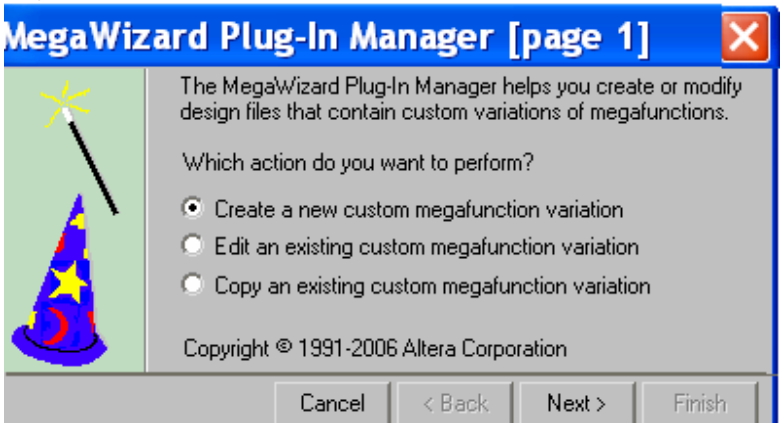


5) У VWF-файлі дані зручно вводити у вигляді групи: виділити ділянку часового інтервалу інструментом вибору або редагування часових діаграм, натиснути інструмент Arbitrary Value (або з меню Edit > Value > Arbitrary Value), в однойменному діалоговому віконці вибрати систему числення (Radix), ввести число (Numeric …) у вибраній системі числення та натиснути кнопку OK – значення відобразиться на цій ділянці часової діаграми групи. Якщо в розділі Time range (часовий інтервал) змінити час початку (Start time) та/або кінця (End time) інтервалу, можна змінити також ділянку введення даних.

3 Засвоїти основи автоматичного створення різновидів мегафункцій за допомогою майстра MegaWizard Plug-In Manager на прикладі синтезу мультиплексора найбільшої розрядності N, необхідного для реалі-

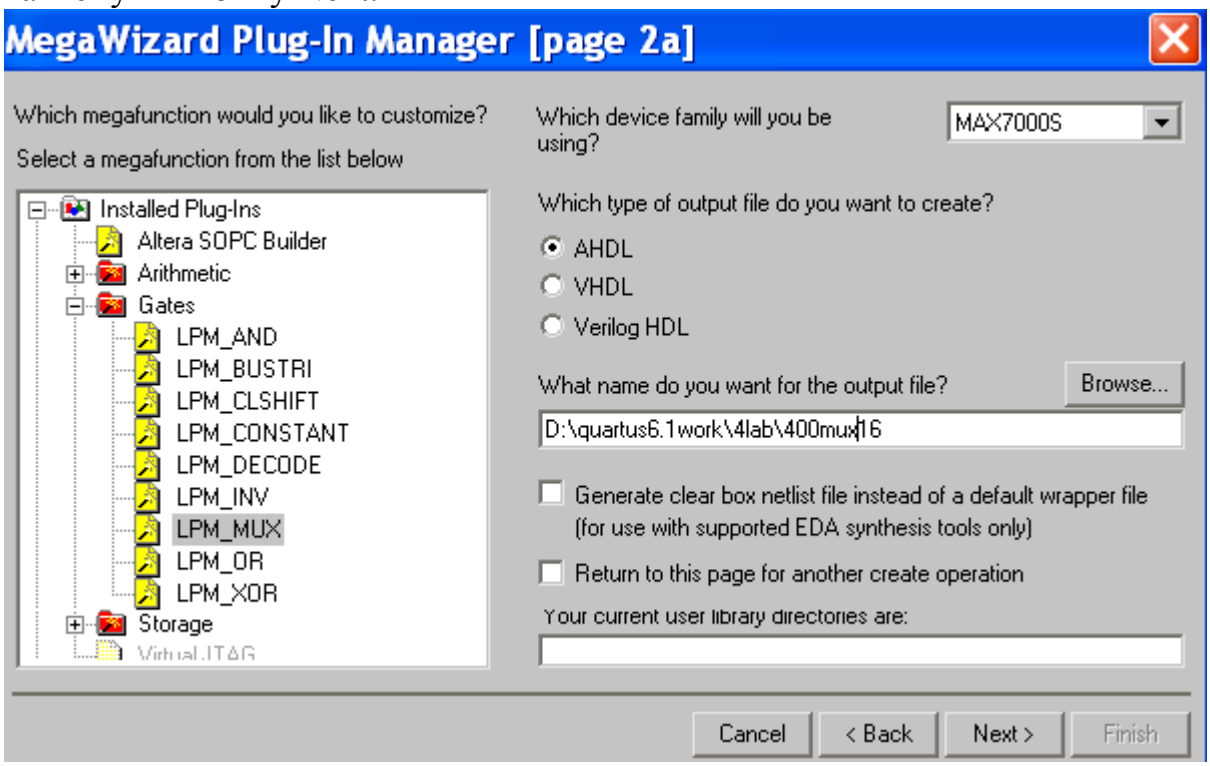
зації заданого варіанту функції (задля наочності і порівняння див. файл 400gr_macro.bdf, схема 1).

3.1 З меню Tools > MegaWizard Plug-In Manager запустити майстра створення різновидів мегафункцій і відповісти на запитання в наступних діалогових вікнах.



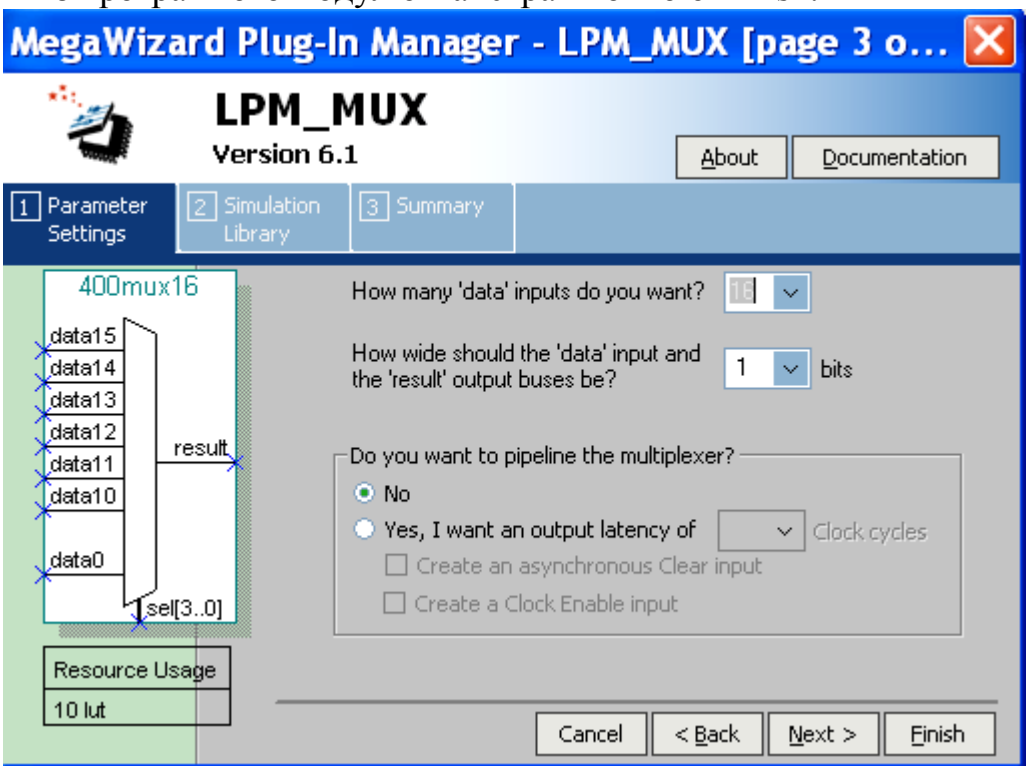
3.2 На сторінці 1 (page 1) вибрати дію, яку треба виконати – створити (Create …) новий різновид мегафункції, редагувати (Edit …) або копіювати (Copy …) існуючий різновид > Next.

3.3 На сторінці 2а ліворуч зі списку приступних мегафункцій вибрати (натиснути +) категорію (Gates) і функцію (LPM_MUX), а праворуч – родину мікросхем (MAX7000S), тип мови для створюваного файла (AHDL) і до рядка 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 рядку What name … відобразиться директорія з ім’ям). Для продовження натиснути кнопку Next.



3.4 На сторінці 3 (ілюстрацію див. нижче) у рядку How many 'data' inputs ...? ввести прокруткою або з клавіатури кількість інформаційних входів даних (до 256 входів, у прикладі – 16), у рядку 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, до 256 біт кожна). Відтак натиснути вгорі кнопку 3 – Summary (підсумкова сторінка), продивитися на сторінці 5 список автоматично створених файлів, серед них файли: текстовий (.tdf) включення (.inc) і символьний (.bsf) та вийти з програмного модулю майстра кнопкою Finish.



3.5 Продивитися утворені файли нового різновиду мегафункції із заданими конкретними параметрами, зокрема, з категорії Other Source Files (інші файли) 4XXmuxN.inc, .bsf (тут N – кількість інформаційних входів).

Приклад: файли 400mux16.inc, .bsf.

✐ Примітки:

1) Скоригувати введені дані можна будь-коли поверненням на попередні сторінки кнопкою Back.

2) З метою відредагувати існуючий різновид мегафункції так само відповідаємо на запитання: на сторінці 2b вибираємо тип файла і імена файла та мегафункції, а також, у разі потреби, директорію. Відтак на сторінці 3 коригуємо параметри і завершуємо редагування кнопкою Finish.

3) Різновид мегафункції можна створити не тільки автоматично, але й ручним способом як подано в необов'язкових п. 4.4, 4.5 роботи.

3.6 Відредагувати символьний файл для зручності користуватися ним у графічних проектах виконанням наступних дій.

а) Створити новий графічний файл 4XXmN.bdf, ввести до нього символ створеного різновиду мегафункції (зі списку Project діалогового вікна Symbol), вхідні і вихідний порти та згрупувати вхідні сигнали в шину.

б) Командою з меню File > Create / Update > Create Symbol Files for

Current File створити символ мегафункції, в якому виводи зображено компактно, у вигляді шин та відкрити його для перегляду.

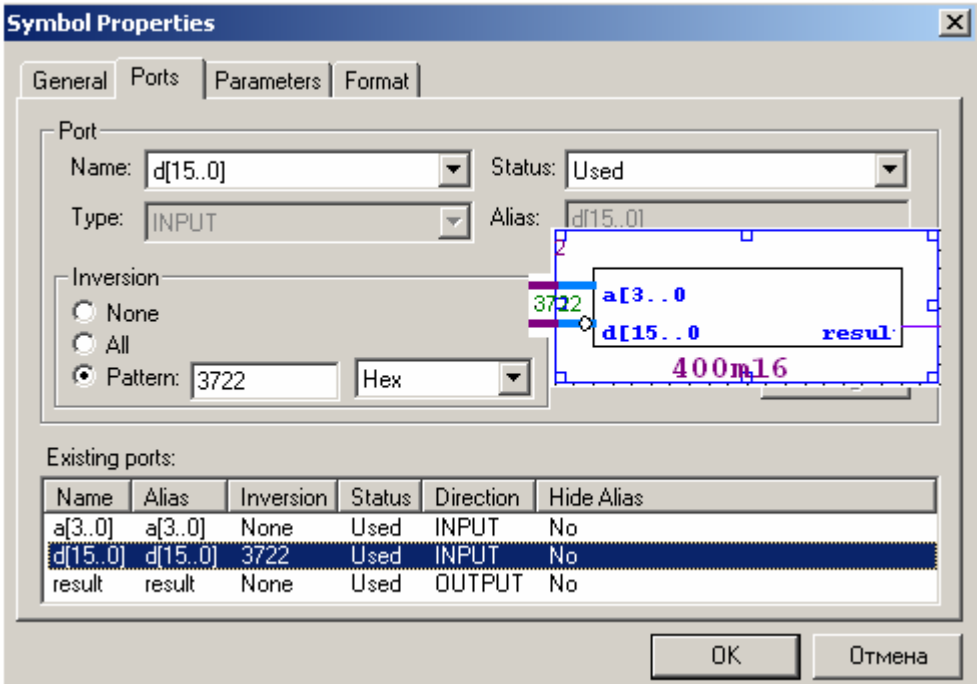
Приклад: файли 400m16.bdf, .bsf.

4 Засвоїти основи настроювання мегафункцій і застосування їх у *графічному* редакторі (на прикладі варіантів п. 2).

4.1 Реалізувати заданий варіант логічної функції на мультиплексорі найбільшої розрядності N (як приклад див. файл 400gr_mega.bdf, схема 1).

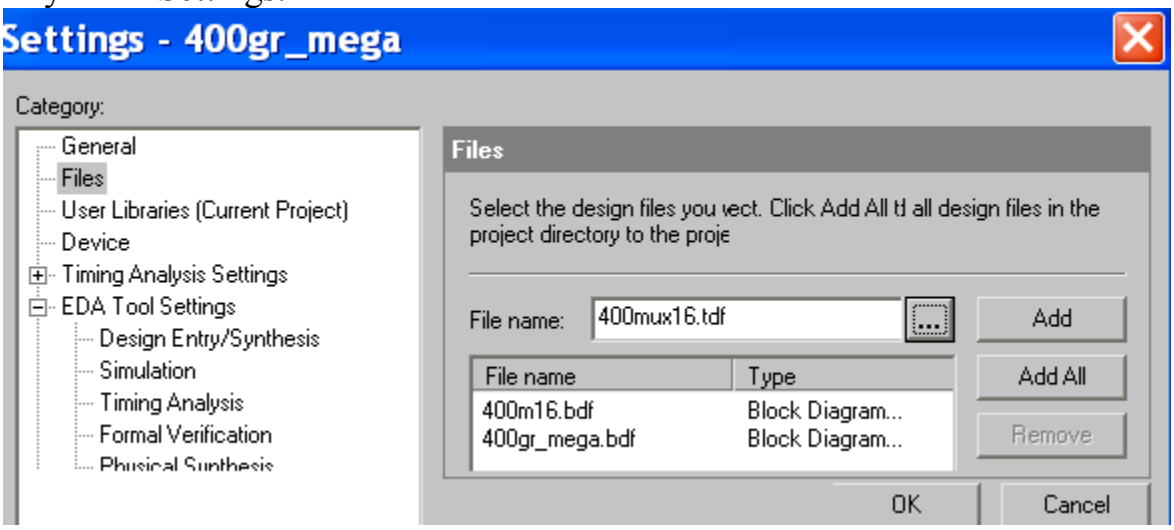
а) Створити новий проект 4XXgr_mega і однойменний графічний файл (.bdf), до якого вставити символ 4XXmN.sym відредагованого різновиду мегафункції майстра MegaWizard Plug-In Manager (зі списку Project діалогового вікна Symbol).

б) Настроїти символ: виділити його і піктограмою Properties (або B2 по символу > Properties) викликати діалогове вікно Symbol Properties (властивості символу) та призначити порти на сторінці Ports так само, як і для макрофункцій. Відмінність полягає в тому, що тепер для інвертування окремих розрядів шин застосовуємо числа в одній із систем числення (натиснути Pattern у розділі Inversion та ввести до віконця код інвертування). Користуючись, наприклад, еквівалентною схемою 1 на кшталт 400rys.bdf, з'ясовуємо, що всі інформаційні входи IN[15..0] мультиплексора з метою спрощення можна заземлити, якщо зінвертувати ті з них, на які слід подати лог. 1. У вікні редагування бітам, що інвертуються, надається значення 1, а тім, що залишаються неінвертованими, – значення 0, тому код інвертування становитиме у цьому прикладі d[15..0] = = B"0011 0111 0010 0010" = H"3722", який і вводимо до віконця Pattern в одній з чотирьох систем числення, основу якої вибираємо прокруткою. По зачиненні (OK) діалогового вікна результати редагування (див. накладений символ) відобразяться бульбашкою інверсії з кодом інвертування (на символі код подається завжди в шістнадцятковій системі числення).



в) Увести інші елементи схеми, потрібні для реалізації заданої функції, дати імена портам, а також провідникам і шинам у випадку їх розгалуження та зберегти файл.

г) Долучити до проекту потрібні складники: піктограмою Settings (або з меню Project > Add/Remove Files in Project, або з меню Assignments > Settings) викликати діалогове вікно Settings (налаштування), вибрати Files зі списку Category, кнопкою огляду (…) вставити до рядка File name графічний файл проекту 4XXgr_mega.bdf, текстовий файл 400muxN.tdf різновиду мегафункції, створений майстром MegaWizard Plug-In Manager, та відредагований файл мегафункції 4XXmN.bdf, кнопкою Add додати ці файли до списку File name (у разі потреби, виділити у списку зайві файли і вилучити їх кнопкою Remove) та натиснути кнопку OK у вікні Settings.



г) виконати компіляцію та функціональне моделювання з метою переконатися в правильності проектування.

Приклад: 400gr_mega.bdf (схема 1), .vwf.

4.2 У тому ж графічному файлі виконати п. 4.1 на мультиплексорах меншої розрядності.

Приклад: 400gr_mega.bdf (схеми 2, 3), .vwf.

✍ 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, отже й номери розрядів не обов'я-зково мають бути однаковими: a3 = x4, a2 = x3, a1 = x2, a0 = x1.

2) У п. 4.2 достатньо виконати один-два варіанти схеми на мультиплексорах різної розрядності; п. 4.4, 4.5 є *факультативні*, не обов'язкові.

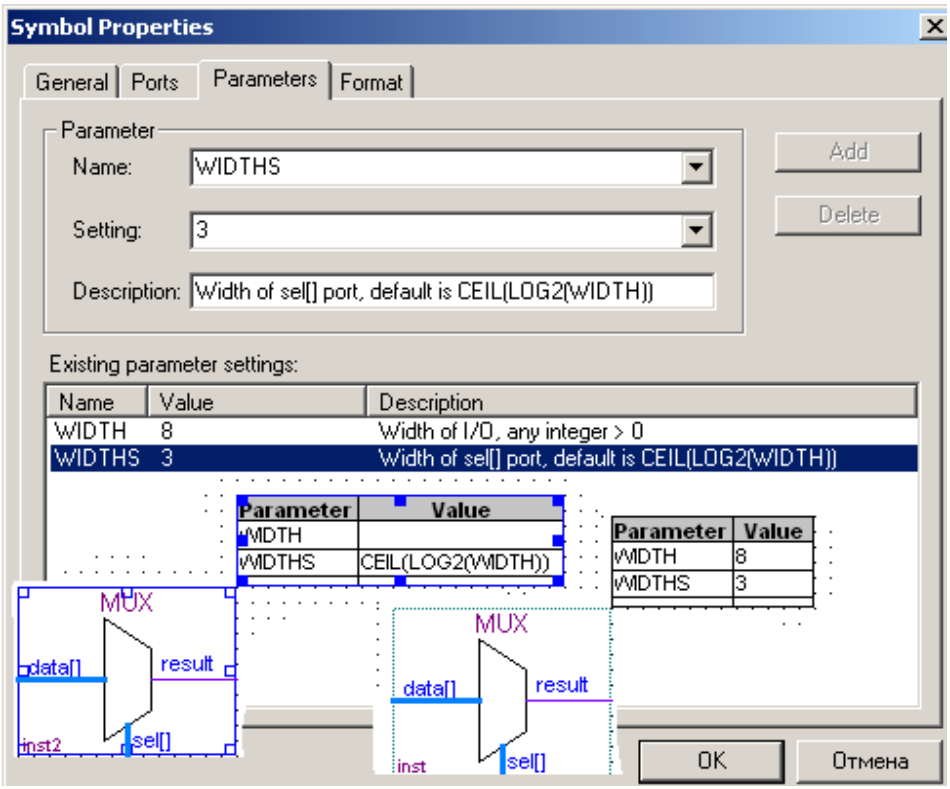
4.3 У тому ж графічному файлі виконати п. 4.1 на мультиплексорі найменшої розрядності (2:1), користуючись його символом, створеним безпосередньо менеджером MegaWizard Plug-In Manager (без згортання входів у шину за п. 3.6). Для цього подвійним клацанням у файлі (або ін-

струментом палітри Symbol Tool) викликати діалогове вікно Symbol, натиснути в ньому кнопку MegaWizard Plug-In Manager, виконати п. 3.2 … 3.4, натиснути кнопку OK у вікні Symbol та вставити символ мегафункції.

Приклад: 400gr_mega.bdf (схема 4), .vwf.

4.4 У тому ж графічному файлі виконати завдання п. 2.1 на мегафункції мультиплексора **mux** за ручного її настроювання (достатньо один варіант мультиплексора).

а) Вставити символ мегафункції до файла звичайним чином з бібліотеки мегафункцій \quartus\libraries\megafunctions\gates\mux, виділити символ і піктограмою (або B2 > Properties) викликати діалогове вікно Symbol Properties, в якому для мегафункції приступними є сторінки Ports і Parameters.



б) На сторінці Parameters прокруткою вибираємо параметр у рядку Name розділу Parameter, вводимо його значення в рядку Setting (прокруткою або з клавіатури) і фіксуємо в графі Value кнопкою Change (кнопкою Delete можна очистити значення, а відтак знов його ввести); повторюючи цю процедуру, призначаємо всі параметри, приступні в рядку Name (результат показано на вкладеному до вікна символі). Порти призначаємо на сторінці Ports так само, як і в п. 4.1,б.

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

Приклад. Для реалізації заданої логічної функції на мультиплексорі 8:1 вставляємо до файла мегафункцію mux (див. файл 400gr_mega.bdf, схема 5) і настроюємо її параметри як описано у п. 4.4,б.

Відтак редагуємо порти в закладці Ports (див. 400rys.bdf, 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 Pattern > 72 Hex (або код в іншій системі числення).

По закритті 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.

По компіляції та функціональному моделюванню (файл 400gr_mega.vwf) переконуємося в правильності проектування: функція *y*2*a* 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, da[7..0], add[2..0] (за суміщення кількох схем в одному файлі шинам кожної схеми слід надавати різні імена).

2) У мультиплексорах ширина адресної шини WIDTHS за умовчанням 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 таблицею за ввімкненого інструмента палітри Show Parameter Assignments (або командою з меню View > Show Parameter Assignments). Цю таблицю можна перетягнути у зручну позицію лівою кнопкою миші або (після виділення) клавішами зі стрілками керування курсором.

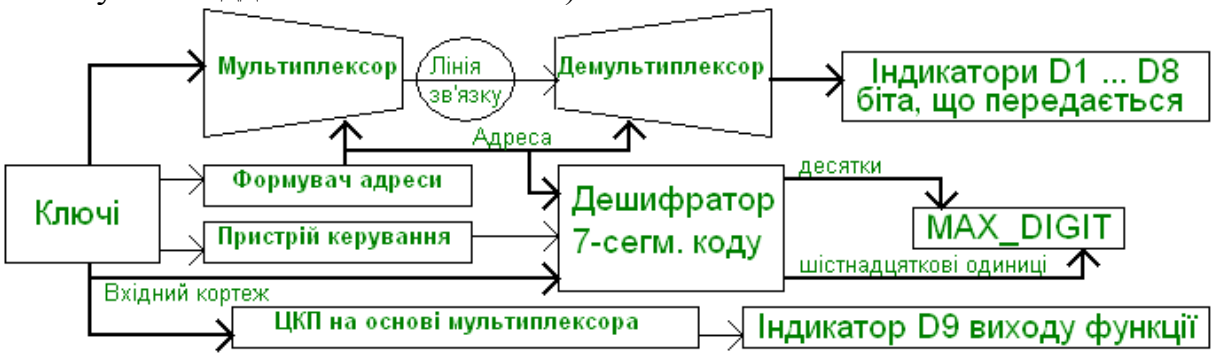
4.5 У тому ж графічному файлі виконати завдання п. 2.1 на мегафункції мультиплексора **lpm_mux** за ручного її настроювання (достатньо один варіант мультиплексора) аналогічно п. 4.4.

Приклад: 400gr_mega.bdf (схема 6), .vwf.

5 Створити проект лабораторного макету для експериментального дослідження заданого варіанту цифрових комутаторів (див. нижче структурну схему і файл 400KOMUTATORY.bdf на рис. Д2 у додатках).

Від ДІП-перемикачів MAX_SW1 вхідне слово sw1[] у паралельному коді надходить на інформаційні входи мультиплексора. Формувач адреси (лічильник) розгортає адресний код a[] в часі таким чином, що після кожного натискання по черзі двох нефіксованих перемикачів MAX_PB1 і MAX_PB2 цей код збільшується на одиницю по колу: 0, 1, ... 7, 0, 1, ..., тобто змінюється в прикладі за модулем 8. Отже, у мультиплексорі відбу-

вається перетворення паралельного коду в послідовний, бо на його виході по черзі з'являються біти вхідного слова (для передачі інформації лінією зв'язку між віддаленими об'єктами).



Структурна схема

З інформаційного входу демультиплексора (який є стробовим входом дешифратора) на вихідному боці лінії тим же самим адресним кодом біти вхідного слова по черзі спрямовуються до його виходів, що індикується одним зі світлодіодів D1 … D8. Якщо ці біти по черзі записати в розряди пристрою з пам'яттю (регістра), то послідовний код перетвориться знов у паралельний і на виходах можна буде зчитати вхідне слово sw1[] у паралельному коді.

Одночасно для індикації в десятковій системі числення адреси її двійковий код через дешифратор 7-сегментного коду подається на старше (десятки) знакомісце 7-сегментного індикатора MAX_DIGIT (якщо модуль адреси перевищує 10, її код потрібно перетворити у двійково-десятковий і індикувати кількома знакомісцями – за кількістю тетрад ДДК).

Спроектований ЦКП на основі мультиплексора (у прикладі – пристрій для реалізації заданої логічної функції) досліджується за допомогою ключів 1 ... 4 перемикача MAX_SW2 і світлодіода D9. Вхідний кортеж (набір змінних) відображається шістнадцятковим цифрами (специфічними для стандартного дешифратора 7-сегментного коду) за допомогою молодшого (одиниці) знакомісця 7-сегментного індикатора MAX_DIGIT.

Для ілюстрації принципу динамічної індикації введено пристрій керування, що складається з трьох вузлів. Восьмий ключ ДІП-перемикача MAX_SW2 керує адресним входом sel мультиплексорів шин 1 ... 3. При sel = 0 мультиплексор 1 з'єднує входи дешифратора 7-сегментного коду з адресним кодом a[], який перетворюється в 7-сегментний код на виходах y[] і з'єднується мультиплексором 2 зі старшим знакомісцем індикатора. При цьому всі сегменти молодшого знакомісця з'єднуються мультиплексором 3 з напругою живлення VCC, отже, цей розряд індикатора гасне. Адресою sel = 1, навпаки, мультиплексор 1 з'єднує входи дешифратора 7-сегментного коду з вхідним кортежем sw2[], який перетворюється в 7-сегментний код шістнадцяткової цифри на виходах y[] і з'єднується мультиплексором 3 з молодшим знакомісцем індикатора. Тепер всі сегменти старшого знакомісця з'єднуються мультиплексором 2 з напругою живлен-

ня VCC, отже, цей розряд індикатора гасне. Якщо індикатор багаторозрядний, слід застосувати мультиплексор шин типу LPM_MUX і адресу формувати аналогічно як у нашій лінії передачі даних. Таким чином, у системі з динамічною індикацією потрібен лише один дешифратор 7-сегментного коду і шляхом сканування адресним кодом кожної миті ввімкнено лише одне знакомісце, а інші погашені, що значно заощаджує ресурс мікросхеми і споживану потужність. За швидкої зміни адресного коду перемикання знакомісць непомітно для ока – здається, що одночасно світяться всі розряди індикатора.

У практичних системах з динамічною індикацією з однією шиною y[] з'єднують паралельно індикатори всіх знакомісць (при цьому не потрібно застосовувати вихідні мультиплексори шин на кшталт наших вузлів 2, 3), а перемикання знакомісць здійснюють дешифратором, який по черзі з'єднує розряди індикатора із джерелом живлення.

5.1 ***Створити проект перетворювачів коду*** в графічному редакторі згідно із заданим варіантом.

а) Створити новий проект 4XXKOMUTATORY (XX – варіант) з директорією власної теки на мікросхемі 7000S (див. Лаб. роботу №3, п. 3.1).

б) Створити однойменний графічний файл верхнього рівня, створити різновиди мегафункцій мультиплексора і демультиплексора за допомогою менеджера MegaWizard Plug-In Manager, відредагувати їх символи (згрупувати сигнали в шини) та вставити до файла верхнього рівня.

в) Вставити до файла верхнього рівня і настроїти мегафункції мультиплексорів шин і макрофункцію мультиплексора дешифратора 7-сегментного коду.

г) Вставити символ формувача адреси (подвійним клацанням у полі файлу викликати діалогове вікно Symbol, у рядку Name натиснути кнопку огляду і вставити символ ..\quartus6.1work\9lab\9addr).

ґ) Вставити також допоміжні елементи і потрібні порти та виконати з'єднання компонентів схеми.

д) Доповнити схему спроектованим на основі мультиплексора варіантом ЦКП для реалізації логічної функції (скопіювати його з файлу 4XXgr_macro.bdf або 4XXgr_mega.bdf).

е) Дати назви портам і з'єднувальним лініям у випадку розгалуження шин. Імена вхідних і вихідних портів, що з'єднують виводи мікросхеми з компонентами плати UP2, взяти точно такими, як у файлі ../4lab/MAX_pin. Якщо шинами sw1[], sw2[] приєднуються до мікросхеми не всі перемикачі, їх кількість зазначити в діапазоні групи, наприклад, sw2[4..1], а якщо деякі перемикачі відокремлено від групи (їх номери не ідуть поспіль з іншими номерами перемикачів), їх назви слід позначити двоцифровими номерами, наприклад, замість імені sw2[8] слід зазначити sw28.

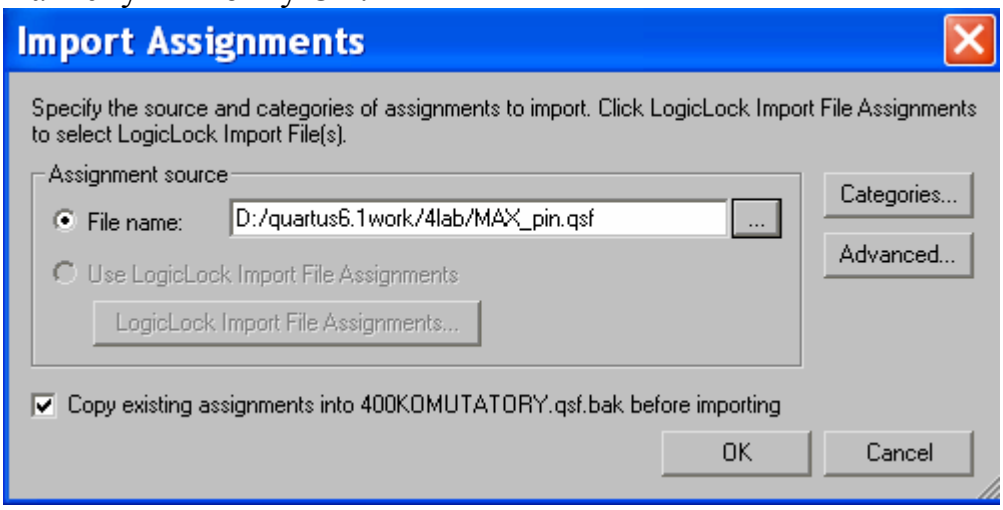
Приклад: 400KOMUTATORY.bdf.

є) Перевірити склад проекту (див. п. 4.1,г) і, у разі потреби, долучити до нього потрібні складники (файл верхнього рівня

4XXKOMUTATORY.bdf, різновиди мегафункцій мультиплексора і демультиплексора, створені за допомогою менеджера MegaWizard Plug-In Manager та відредаговані їх символи, а також формувач адреси – файли з п. 5.1,а,б,г) та вилучити зайві.

5.2 ***Імпортувати призначення виводів мікросхеми з іншого файлу,*** де виконано розведення їх на друкованій платі (файл ../4lab/MAX_pin.bdf).

а) З меню Assignments > Import Assignments викликати однойменне діалогове вікно, в якому в рядку File name кнопкою огляду (…) у вікні Select File вибрати файл ../4lab/MAX_pin.qsf (Quartus II Settings File). Двічі клацнути його ім'я (або виділити і натиснути кнопку збереження) – директорія з ім'ям файлу з'явиться в рядку File name вікна Import Assignments. Відтак натиснути кнопку OK.



✐ *Примітка:* Категорії і додаткові налаштування параметрів імпорту призначень (кнопки Categories та Advanced) можна залишити такими, що встановлені за умовчанням. Проте під час опрацювання складніших проектів ці налаштування можна змінити: за відкритого вікна Assignment Categories або Advanced Import Settings натиснути клавішу F1, ознайомитися в довідці з функціями призначень та вибрати потрібні.


б) Скоригувати імпортовані призначення виводів відповідно до портів проекту. Для цього піктограмою Assignment Editor (або з меню Assignments > Assignment Editor) відкрити файл редактора призначень (до компіляції порти буде репрезентовано зі знаком запитання), залишити в ньому категорію All, натиснути кнопку редагування Edit Bar інструментальної палітри та вилучити зайві виводи і скоригувати імена існуючих портів. Наприклад, у файлі зразка 400KOMUTATORY.bdf вилучаємо зайвий вивід Clock (потрібно клацнути будь-де в його рядку електронної таблиці покажчиком у формі хреста і натиснути інструмент палітри Delete) і так само виводи sw2[5] … sw2[7] (декілька виводів виділяються, як звичайно, за натиснутої клавіші Shift). Для виправлення імені порту слід клацнути його ім'я, яке з'явиться в рядку Edit, та внести корективи в цьому рядку з клавіатури звичайним чином,

наприклад, виправити sw2[8] на sw28 – з рядка Edit корекція одразу переноситься до таблиці. На завершення слід зберегти зміни у файлі Assignment Editor.

Примітка: Якщо клацнути комірку електронної таблиці і натиснути кнопку Information Bar інструментальної палітри, то в розділі Information з'явиться інформація щодо елемента таблиці або рекомендації, в який спосіб виконувати певні дії.

5.3 Виконати ***компіляцію*** проекту – імпортовані призначення з'являться у формі таблиць біля портів. У разі потреби, як звичайно, усунути помилки та опрацювати застереження.

Приклад: проект 400KOMUTATORY.

Примітка: Призначення біля портів позначаються таблицями за ввімкненого інструменту палітри Show Location Assignments (або командою з меню View > Show Location Assignments). Цю таблицю можна перетягнути до зручної позиції лівою кнопкою миші або (після виділення) клавішами зі стрілками керування курсором.

5.4 Виконати ***функціональне моделювання,*** переконатися в правильності проектування.

Приклад: 400KOMUTATORY.bdf, .vwf.

5.5 ***Сформувати файл програматора*** (див. Лаб. роботу №3, п. 3.5).

Приклад: 400KOMUTATORY.cdf.

5.6 ***Виконати фізичне програмування мікросхеми*** (див. Лаб. роботу №3, п. 3.6).

5.7 ***Дослідити пристрій на запрограмованій ІС.***

5 ЛАБОРАТОРНА РОБОТА №5.
СТВОРЕННЯ ТЕКСТОВИХ ПРОЕКТІВ

Мета роботи: основи створення проекту за його текстового введення, запровадження до текстового файлу макрофункцій, символів та мегафункцій; дослідження запрограмованого пристрою.

ДОМАШНЄ ЗАВДАННЯ

Для заданого варіанту (див. Додатки, завдання 5): а) записати апаратною мовою AHDL логічну функцію завдання 2; б) для наочності підготувати графічні файли – аналоги текстових файлів.

СТИСЛІ ТЕОРЕТИЧНІ ВІДОМОСТІ

1 Основи проектування в текстовому редакторі

1.1 *Основні складники проекту*

Мови опису апаратних засобів HDL (Hardware Description Language) типу Verilog HDL та VHDL є універсальними для різних повноциклових САПР. Програмні пакети MAX+PLUS II і Quartus II підтримують також мову високого рівня AHDL (Altera Hardware Description Language), пристосовану для мікросхем фірми Altera (тут розглядатимемо проектування на основі цієї мови). Пристрій, створений у текстовому редакторі, можна згорнути до символу і ввести до графічного файлу. З іншого боку, графічні об'єкти, наприклад, макрофункції і мегафункції можна як готові модулі вводити до текстових файлів. Текстовий опис, вислідом якого є проектний файл з розширенням **.tdf** (Text Design File), містить розгалужену систему засобів, таких як логічні рівняння, таблиці відповідності та інші оператори.

Структурно проект (Design) складається із секцій (Section), операторів або підсекції (Statement); секція змінних (Variable Section) і логічна секція (Logic Section), у свою чергу, складаються з окремих операторів. З понад трьох десятків таких складників розглядатимемо лише біля половини, які можна вважати основними. У рекомендованій структурі проекту вони розташовуються в такій послідовності.

Title Statement
Include Statement
Function Prototype Statement
Subdesign Section
Variable Section
    Instance Declaration
    Node Declaration
    Register Declaration
State Machine Declaration
Logic Section
    Boolean Equations
    Case Statement
    If Then Statement
Truth Table Statement

Обов'язковими для кожного TDF-файлу є секція підпроекту (Subdesign Section) і логічна секція (Logic Section), а застосування інших складників визначається, здебільшого, типом використовуваної елементної бази

та вибраними засобами проектування. Кожний такий структурний блок має суворо відповідати правилам свого синтаксису. Аби пришвидшити складання текстового файлу (TDF-файлу) 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).

Наведемо тут лише короткий опис основних проектних блоків (оператори Register Declaration та State Machine Declaration подаються під час вивчення послідовнісних пристроїв), а докладно питання процедури введення і синтаксису з конкретними прикладами розглядаються безпосередньо в лабораторному завданні.

1.2 ***Заголовок (Title Statement)***

Необов'язковий оператор заголовка є зручний для документування. Він використовується також компілятором у коментарях файлу повідомлення (.rpt – від Report). Оператор може застосовуватися у файлі лише один раз і має бути приміщений поза іншими секціями.

1.3 ***Оператор включення (Include Statement)***

Структурні компоненти, для яких існує файл включення (з розширенням **.inc**), зокрема, мегафункції зручно запроваджувати до проекту за допомогою оператора включення. Оператор містить лише назву файлу включення, яка надається об'єкту під час його створення або існує в бібліотеці комплекту. Для стандартних мегафункцій файл включення компілятор сам вибирає з бібліотеки, достатньо лише вказати її назву і доповнити розширення .inc. Для різновидів мегафункцій менеджер автоматично створює одночасно сам різновид у текстовому файлі (.tdf) і файл включення (.inc).

Синтаксис оператора не дозволяє вказувати директорію компонента, тому якщо його різновид створено менеджером в іншій теці, файл включення слід перенести до проекту. Оператор має бути приміщений поза інших секцій, може вводитися до файлу довільну кількість разів і не може бути вкладеним.

1.4 ***Оператор прототипу функцій (Function Prototype Statement)***

До графічного файлу можна вставляти не тільки елементи, але й готові структурні компоненти, наприклад, макрофункції, які є в бібліотеці програми, або блоки, створені користувачем в інших файлах і згорнуті до символу. Для цього використовуються символи цих модулів з позначеними іменами входів і виходів. У текстовому файлі роль символу відіграє прототип функції у вигляді її назви і тими ж самими іменами входів і виходів. Слід зауважити, що оператор прототипу функцій має розташовуватися у файлі поза секцією підпроекту, але перед секцією, в якій ці функції використовуються.

1.5 ***Секція підпроекту (Subdesign Section)***

Секція призначена для оголошення вхідних та вихідних портів і

тому є обов'язковою. Вона містить назву проекту, *таку саму*, що й ім'я TDF-файлу (без розширення), та списки зовнішніх сигналів – вхідних і вихідних портів.

1.6 ***Секція змінних*** **(Variable Section)**

Секція складається з низки підсекцій для оголошення змінних різного типу і є необов'язковою. Змінна мовою програмування високого рівня може означати як просто сигнал, так і деякий внутрішній щодо програмного забезпечення логічний модуль, наприклад, макрофункцію, мегафункцію або символ, створений користувачем.

1.7 ***Оператор*** о***голошення вузла (Node Declaration)***

Якщо є сигнали, не оголошені в секції підпроекту або в інших операторах секції змінних, їх можна ввести оператором оголошення вузла цієї секції. Оператор дозволяє оголосити проміжні сигнали, які зручно застосувати в логічних рівняннях з метою їх спрощення, особливо якщо такі сигнали використовуються в Булевих рівняннях неодноразово.

1.8 ***Оператор оголошення модуля (Instance Declaration)***

Під час використання в графічному файлі деякого функціонального модуля, наприклад, примітива, макрофункції або мегафункції, його конкретному екземплярові надається ідентифікаційний номер (позначається в лівому нижньому куту). У текстовому файлі, аби розрізнити окремі екземпляри функціональних модулів, кожному з них у секції змінних (Variable Section) надається символічне власне ім'я зразка, наприклад, двом дешифраторам можна дати символічні імена dc1 та dc2.

Синтаксис оператора містить символічне ім'я зразка і після двокрапки ім'я функціонального модуля, попередньо визначене операторами прототипу функцій Function Prototype Statement або включення Include Statement. Подібний синтаксис, наприклад, для транзисторів означав би їх ідентифікаційні номери в схемі і тип на кшталт VT1 : КТ315; VT2 : КТ315 тощо.

1.9 ***Логічна секція (Logic Section)***

Логічна секція є обов'язковою в TDF-файлі, бо саме вона визначає залежність вихідних сигналів від вхідних з використанням інших змінних, оголошених у файлі. До цієї секції вставляються оператори, основні з яких розглядаються нижче.

1.10 ***Булеві рівняння (Boolean Equation)***

За допомогою логічних рівнянь здійснюється зв'язок між вихідними і вхідними сигналами через всі примітиви, макрофункції, мегафункції та інші складники проекту. Ліва сторона рівняння містить сигнали, що визначаються логічним виразом правої сторони (який у найпростішому випадку може бути змінною).

1.11 ***Оператор таблиці відповідності (Truth Table Statement)***

Як і в звичайній таблиці відповідності, таблиця мовою AHDL містить комбінації вхідних змінних і значення вихідних функцій. Для частково визначених функцій частина вхідних наборів може подаватися за допомогою невизначених станів X (через це символ *x* без індексу не можна за-

стосовувати в текстових файлах для позначення змінної).

1.12 ***Умовний оператор*** (***If Then Statement***)

Оператор логічної секції If Then Statement (якщо, тоді) містить умову для подальшого розгалуження алгоритму роботи пристрою. За виконання цієї умови активізуються одні інструкції у вигляді списків логічних виразів, а за її невиконання – інші інструкції.

1.13 ***Оператор вибору (Case Statement)***

Оператор вибору (Case – випадок) містить булів вираз (у найпростішому випадку групу змінних), від значень якого залежить дія пристрою. У тілі оператора наводяться постійні значення цього виразу і інструкції (булеві рівняння, інші оператори, наприклад, таблиця відповідності), які діють за наведених значень. За невиконання жодного з цих значень можуть подаватися альтернативні інструкції.

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.

2 Логічні оператори

Операнди логічних рівнянь з'єднуються логічними операторами, черговість виконання яких визначається пріоритетом (табл. 1).

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

Приклад:

$$y_1 = \overline{(x_1 + x_3 + \overline{x_4})\left[\ x_2 + \overline{x_1 x_4 + (x_1 \oplus x_4)}\ \right]} = \overline{(a)\left[x_2 + \overline{(b + c)}\right]},$$

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

Таблиця 1

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
# (фунт)	АБО (OR)	4	$x_1 + x_2$	x1 # x2
!#	АБО-НЕ (NOR)	4	$\overline{x_1 + x_2}$	x1 !# x2

У складних випадках вираз доцільно поділити на частини, завдяки чому спрощується запис мовою AHDL:

y1 = a !& (x2 # (b !# c)) = (x1 # x3 # !x4) !& (x2 # (x1&x4 !# x1$x4)).

Останній вираз зв’язує безпосередньо вихідний порт (функцію) y1 з вхідними портами (змінними) x1... x4, які є зовнішніми виводами мікросхеми і оголошуються в секції підпроекту (Subdesign Sektion). Проте з метою уникнути помилок і забезпечити легкість для читання складні вирази можна вводити до логічної секції (Logic Sektion) частинами

a = x1 # x3 # !x4; b = x1 & x4; c = x1 $ x4; y1 = 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,

!y1 = a & (x2 # (b !# c)).

ЛАБОРАТОРНЕ ЗАВДАННЯ

1 Засвоїти основи створення проекту за його *текстового* введення апаратною мовою AHDL.

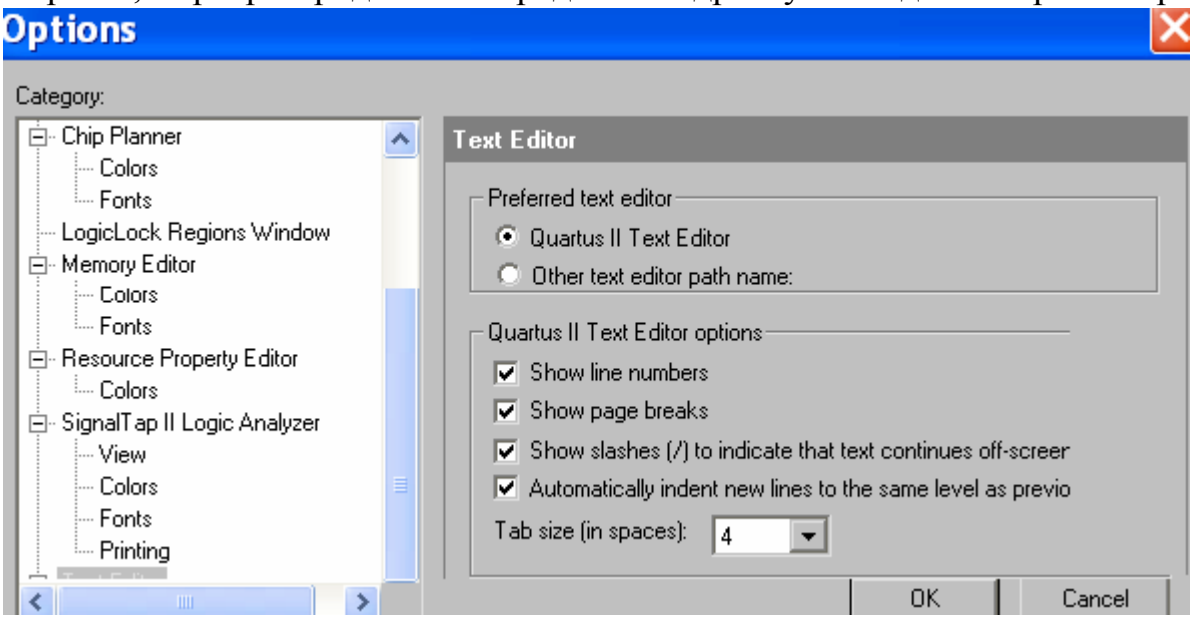
1.1 ***Засвоїти синтез із використанням основних секцій і оператора оголошення вузлів*** на прикладі виконання завдання 5.1 (як зразок див. файл 500node.tdf).

а) Дати ім'я новому проекту: 5XXnode у власній теці.

б) ***Створити текстовий файл*** піктограмою New Text File текстового редактора (або з меню File > New > Device Design Files > AHDL File > OK) і надати ім'я з розширенням .tdf безіменному (AhdlN.tdf) файлу: піктограмою збереження файлу (або з меню File > Save) викликати діалогове вікно збереження, в якому встановити тип файлу AHDL File (*.tdf), ввести ім'я файлу 5XXnode і натиснути кнопку збереження. У рядку заголовка файлу відобразиться назва 5XXnode.tdf.

в) Ознайомитися з інструментальною палітрою текстового редактора (на ілюстрації показано основні інструменти), з меню Tools > Customize Text Editor встановити облямовані кнопки великого розміру (зайві інструменти можна вилучити звичайним чином).

г) Налаштувати опції текстового редактора: меню Tools > Options > вибрати категорію Text Editor і на однойменній сторінці ввімкнути перемикач Quartus II Text Editor та в розділі Quartus II Text Editor options підняти прапорці всіх опцій (показувати номери рядків, маркери розмежування сторінок, маркери продовження рядка за кадром у вигляді косої риски пра-



воруч, автоматично вирівнювати відступ нових рядків такий самий, як у попередніх рядках), а також вибрати розмір табуляції – зсув рядків, наприклад, на 4 знаки. Відтак вибрати категорію Text Editor > Fonts і на сторінці Fonts встановити шрифт, наприклад, Courier New і розмір 14 та натиснути кнопку OK у вікні Options.

ґ) Рекомендується (але не обов'язково) ввести ***заголовок*** логічного блоку: інструментом вставлення шаблону Insert Template (або з меню Edit > Insert Template, або кнопкою B2 > Insert Template) викликати діалогове вікно Insert Template (далі ці дії стисло позначатимемо: IT), в якому на лівій вкладці залишити синтаксис AHDL, а на правій Template section (секції шаблону) двічі клацнути Title Statement (або виділити і натиснути OK). Поставити курсор на початку після лапок, ввести довільну назву (до 255 символів), у кінці залишити лапки і крапку з комою, а все інше стерти, наприклад:

```
TITLE "Логічна функція";
```

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

д) Ввести обов'язкову секцію ***Subdesign*** (підпроект) для оголошення вхідних змінних і вихідних функцій (відповідають портам – зовнішнім виводам мікросхеми): IT > Subdesign Section > OK та заповнити цей розділ:

– після ключового слова Subdesign ввести назву проекту (до 32 символів) – ***ту саму, що в заголовку*** файлу без розширення (5XXnode), а інше стерти, наприклад:

```
SUBDESIGN 500node
```

– 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 умовчанням GND чи VCC для функцій АБО та І відповідно), наприклад:

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

відтак все зайве слід стерти, а ***курсор перевести*** на початок нового рядка після дужки.

☝ *Примітки:*

1) Імена змінних і вихідної функції мають точно відповідати файлу ../2lab/2XX.bdf з метою подальшого аналізу правильності проектування.

2) 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.

е) У разі необхідності, оголосити внутрішні вузли (відповідають проміжним змінним, не пов'язаним із зовнішніми виводами мікросхеми) вводять необов'язкову ***секцію змінних:*** IT > Variable Section > OK (відтак у наступному рядку курсор перевести з відступом Tab) та ***підсекцію оголошення вузлів:*** IT > Node Declaration > OK, в яку вставляють *через кому*

імена зазначених змінних, залишають двокрапку, ключове слово та крапку з комою (відтак все зайве слід стерти, а ***курсор перевести*** на початок нового рядка), наприклад (варіант 00):

```
VARIABLE
a, b, c       : NODE;
```

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

```
BEGIN
    y1= a !& (x2 # (b !# c)); a = x1 # x3 # !x4; b = x1 & x4; c = x1 $ x4;
END;
```

Приклад: 500node.tdf.

Примітки:

1) Ім'я в секції Subdesign обов'язково має збігатися з ім'ям проекту.

2) Необов'язкові прогалини між складниками у форматуванні формул, відступи і інтервали не мають значення, вони лише покращують стиль, легкий для читання та перевірки. Проте оператори мають точно відповідати синтаксису мови AHDL (інакше під час компіляції видається повідомлення про помилки та їх локалізацію – зазначаються номери рядків тексту і що в них є помилкове).

3) У TDF-файлі ключові слова виділяються синім кольором, дані для компілятора (у тому числі розділові знаки) та інформація, взята в лапки, – чорним, а довільна інформація (у тому числі коментарі, оточені з обох боків знаками „%" або відокремлені на початку рядка подвійним тире, 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 файлу призначені за умовчанням кольори можна з меню Tools > Options > Text Editor > Colors.

ж) Зберегти файл, перевірити його на ***синтаксичні помилки*** (у разі потреби) та виконати ***компіляцію*** проекту, натиснути OK в інформаційних віконцях за відсутності помилок (іна-кше виправити) та зачинити вікна компілятора.

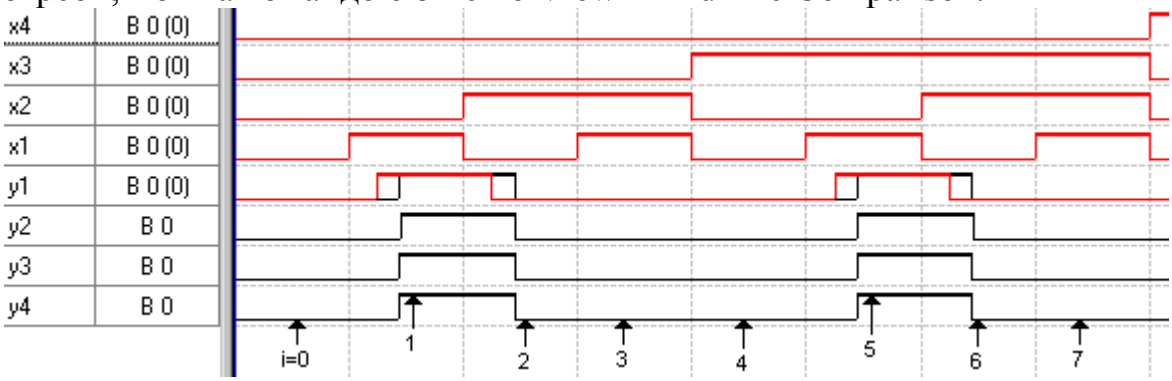
з) Виконати ***функціональне моделювання*** за допомогою редактора часових діаграм Waveform Editor.

Приклад: 500node.vwf.

и) Виконати ***автоматичне порівняння часових діаграм*** з метою проаналізувати результати проектування: відкрити вікно одного з порівнюваних файлів (2XX.vwf з Лаб. роботи №2, діаграми якого вважатимуться початковими) > натиснути інструмент порівняння (або ко-

манду з меню View > Compare...) > у діалоговому вікні двічі клацнути другий з порівнюваних VWF-файл (5XXnode.vwf).

Відтак *однойменні* діаграми другого файлу автоматично накладаються на діаграми початкового файлу і набувають червоного кольору на ділянках, де вони збігаються; діаграми початкового файлу, відсутні в другому файлі, залишаються чорними, а діаграми другого файлу, відсутні в початковому, не відображаються. При цьому в полі Value рівні сигналів другого файлу зазначаються в дужках. Зняти накладання діаграм, у разі потреби, можна командою з меню View > End File Comparison.

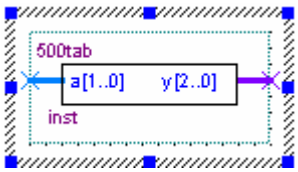


1.2 ***Виконати синтез із використанням оператора таблиці відповідності Truth Table Statement*** на прикладі завдання 5.2, для чого дати ім'я новому проекту 5XXtab, створити однойменний текстовий файл і зберегти його із включенням до проекту. Відтак виконати такі дії.

а) Ввести заголовок (Title Statement), вхідні і вихідні вектори (Subdesign Section) та логічну секцію (Logic Section) – ці дії докладно описано у п. 1.1,г,д,є, приклад див. нижче.

б) Між ключовими словами (BEGIN та END;) розташувати курсор (з відступом Tab) і ввести ***підсекцію таблиці відповідності*****:** IT > Truth Table Statement та вставити в утворений таким чином шаблон мовою AHDL: 1) 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 та 2) тіло таблиці в тому ж форматі, що й заголовок (кожний рядок 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.

в) Виконати компіляцію, функціональне моделювання та автоматичне порівняння утворених таким чином часових діаграм з файлом 3XX.vwf (Лаб. робота №3).



г) Створити символьний файл: за відкритого файлу 5XXtab.tdf в меню File вибрати Create / Update > Create Symbol File for Current File (створити символьний файл для поточного файлу), натиснути ОК у повідомленні про створення такого файлу; відтак відкрити символьний файл піктограмою відкриття файлу і вибором у діалоговому вікні типу файлів Graphic

Files або Other Source Files (інші файли); у разі потреби, відредагувати графічний символ та порівняти його з принциповою електричною схемою.

Приклад: 500tab.tdf, .vwf, .bsf.

```
TITLE "Дешифратор 2:3 (таблиця)";
SUBDESIGN 500tab
(
    a[1..0]      : INPUT;
    y[3..0]      : OUTPUT;
)
BEGIN                    %   Logic Section          %
    TABLE                %   Truth Table Statement  %
        a[] => y[];
        0   =>  1;
        1   =>  2;
        2   =>  4;
        3   =>  X;
    END TABLE;
END;
```

1.3 ***Виконати синтез із використанням умовного оператора If Then*** (якщо, тоді) у проекті 5XXif_then (на прикладі завдання 5.3) шляхом створення текстового файлу (.tdf) і аналізу результатів так само, як у п. 1.2 за винятком п. 1.2,б, в якому ввести умовний оператор If Then**:** IT > 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 лог. 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 лог. 0), а все інше слід стерти;

в) виконати так само компіляцію, функціональне моделювання та автоматичне порівняння утворених часових діаграм з файлом 4prim.vwf.

Приклад: 500if_then.tdf, .vwf.

```
TITLE "Демультиплексор (умовний оператор)";
SUBDESIGN 500if_then
(
    a[1..0], G          : INPUT = VCC;
    y[3..0]             : OUTPUT;
)
BEGIN
  IF G THEN
      y0 = !a1&!a0; y1 = !a1&a0; y2 = a1&!a0; y3 = a1&a0;
  ELSE
        y[] = GND;
  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.

1.4 ***Виконати синтез із використанням оператора вибору Case*** (випадок) у проекті 5XXcase (на прикладі завдання 5.4 – для наочності див. файл 4prim.bdf з Лаб. роботи №4 за відсутності входу d3) шляхом створення текстового файлу (.tdf) і аналізу результатів так само, як у п. 1.2 за винятком п. 1.2,б, в якому ввести оператор вибору Case: IT > 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 функції (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 п. б);

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

Приклад: 500case.tdf, .vwf.

```
TITLE  "Мультиплексор 3:1 (оператор вибору)";
SUBDESIGN 500case
(
      a[1..0], d[2..0]      : INPUT;
      y                     : OUTPUT;
)
BEGIN
    CASE a[] IS
    WHEN 0 => y=d[0];
       WHEN 1 => y=d[1];
       WHEN 2 => y=d[2];
       WHEN OTHERS => y=X;
    END CASE;
END;
```

2 Засвоїти запровадження макрофункцій, символів та мегафункцій до проекту за його *текстового* введення.

2.1 Засвоїти включення до проекту ***макрофункції*** (на прикладі виконання завдання 5.5): створити новий проект 5XXmacro і однойменний текстовий файл (.tdf) із включенням його до проекту, відтак виконати такі дії (для наочності див. графічний файл 3XX_7seg.bdf, схема 1 з Лаб. роботи №3).

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

```
TITLE "Спец. дешифратор 7-сегментного коду на макрофункції";
```

2) До шаблону секції Function Prototype Statement вставити прототип макрофункції двійкового дешифратора (описати використовувані входи і виходи згідно із синтаксисом мови безпосередньо за графічним символом або повністю скопіювати з довідки Help пакету MAX+PLUS II).

```
FUNCTION 74139o (g1n, b1, a1)
    RETURNS (y10n, y11n, y12n, y13n);
```

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

```
SUBDESIGN 500macro
(
    a1, a0                  : INPUT;
    a, b, c, d, e, f, g, dp : OUTPUT;
)
```

4) Ввести секцію змінних Variable Section.

```
VARIABLE
```

5) Вставити підсекцію оголошення зразка Instance Declaration і надати ім’я зразку (для наочності у прикладі 300_7seg.bdf позначено: dc – ім’я зразка дешифратора) та через двокрапку ввести ім’я макрофункції (74139o).

```
    dc    :   74139o;
```

6) Ввести логічний блок Logic Section, між ключовими словами (BEGIN та END;) розташувати курсор (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.y11n # !dc.y12n;
    c=!dc.y12n; d=!dc.y10n; e=GND; f=GND; g=!dc.y12n; dp=VCC;
END;
```

✍ Примітки:

1) Логічні зв’язки між виходами та входами макрофункції містяться в її прототипі (п. 2.1.2), 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 крапку 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, наприклад, інвертор на виході: a = !dc.y11n.

2) Імена макрофункції та її параметрів мають суворо відповідати формату: FUNCTION <ім’я функції> (вхідні порти через кому) RETURNS (вихідні порти через кому) <крапка з комою>, про який можна дізнатися з

прототипу функції в довідці пакету MAX+PLUS II (Help > Old-Style Macrofunctions > Категорія > Ім'я макрофункції > AHDL Function Prototype) або з графічного файлу в тому ж пакеті (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 500macro.tdf, .vwf.

2.2 Засвоїти включення до проекту ***символу*** (на прикладі виконання завдання 5.6): створити новий проект 5XXsymb і однойменний текстовий файл (.tdf), включити до проекту цей файл і текстовий файл символу 5XXtab.tdf та виконати такі самі дії, як і в п. 2.1 (для наочності див. графічний файл 3XX_7seg.bdf, схема 2 з Лаб. роботи №3) із такими відмінностями:

а) у п. 2.1.2 до шаблону секції Function Prototype Statement ввести із заздалегідь створеного текстового файлу символу 5XXtab.tdf (п. 1.3) назву і параметри із секції підпроекту; наприклад, із файлу 500tab.tdf

```
SUBDESIGN 500tab
(
        a[1..0]                 : INPUT;
        y[2..0]         : OUTPUT;
)
```

вводимо до нашого файлу, дотримуючись шаблона дані:

```
FUNCTION 500tab(a[1..0])
        RETURNS (y [2..0]);
```

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

```
VARIABLE
        dc      :  500tab;
```

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

```
BEGIN
          dc.a[1..0]=a[1..0]; a2=dc.y[1]; b2=dc.y[1]#dc.y[2]; c2=dc.y[2];
    d2=dc.y[0]; e2=GND; f2=GND; g2=dc.y[2]; dp2=VCC;
END;
```

Приклад: файли 500symb.tdf, .vwf.

2.3 Засвоїти включення до проекту ***мегафункцій*** (на прикладі виконання завдання 5.7; задля наочності див. файл 400gr_mega.bdf): створити новий проект 5XXmega і однойменний текстовий файл (.tdf), зберегти із включенням до проекту цей файл, ввести заголовок (Title Statement) та виконати такі дії (під кожним пунктом наводяться фрагменти файлу 500mega.tdf).

```
TITLE "y на основі мегафункцій ";
```

1) Включити до складу проекту принаймні один різновид мегафункції мультиплексора певної розрядності (за власним вибором), створений менеджером MegaWizard Plug-In Manager у Лаб. роботі №4, наприклад, 4XXmux8.inc, для чого:

а) вставити шаблон оператора включення **Include Statement** і в лап-

ках ввести після двох крапок частину директорії, відмінну від директорії проекту ../4lab, та через косу риску ім'я різновиду мегафункції (з розширенням .inc) /4XXmux8.inc, надане під час його створення, наприклад:

```
INCLUDE "../4lab/400mux8.inc";
```

б) піктограмою Settings (або з меню Project > Add/Remove Files in Project) викликати діалогове вікно Settings, відкрити сторінку Files і кнопкою огляду (…) ввести до складу проекту файл включення (.inc) і текстовий файл (.tdf) зазначеного різновиду мегафункції (з Лаб. роботи №4, тип файлів Other Source Files). Низку файлів з діалогового вікна Select File можна перенести до проекту одним заходом: виділити їх за натиснутої клавіші Ctrl та натиснути кнопку збереження.

✐ *Примітка:* Альтернативою використання готового різновиду мегафункції є створення за допомогою менеджера MegaWizard Plug-In Manager нового її різновиду з директорією теки проекту.

2) Включити до складу проекту принаймні один із типів мегафункцій мультиплексора (за власним вибором), що є в бібліотеці програмного пакету, наприклад, **mux** оператором включення Include Statement.

```
INCLUDE "mux.inc";
```

✐ *Примітка:* Стандартні мегафункції компілятор бере безпосередньо з бібліотеки за назвою, тому й немає потреби вводити директорію.

3) Ввести вхідні і вихідні порти в секції підпроекту (Subdesign Section), використовуючи замість x[] інші імена змінних (бо літера x зарезервована для невизначеного стану), наприклад, in[].

```
SUBDESIGN 500mega
(
    in[4..1]    : INPUT;
    y2, y2a     : OUTPUT;
)
```

4) Вставити секцію змінних Variable Section.

```
VARIABLE
```

5) Надати імена компонентам схеми, для яких створено різновиди мегафункцій за допомогою менеджера, для чого викликати підсекцію оголошення непараметризованого зразка **Instance Declaration (non-parametererized)** і ввести в її шаблон ім'я зразка функції, через двокрапку – ім'я мегафункції та залишити крапку з комою.

```
    mx2         : 400mux8;
```

6) Надати імена компонентам схеми, в яких використовуються стандартні мегафункції, для чого викликати підсекцію оголошення параметризованого зразка **Instance Declaration (parametererized)** і ввести в її шаблон ім'я зразка функції, через двокрапку – ім'я мегафункції, після ключового слова WITH – її параметри (кількість входів даних WIDTH і розрядність адресної шини WIDTHS) та залишити крапку з комою.

```
mx2a        : mux
    WITH (WIDTH = 8, WIDTHS = 3);
```

✐ *Примітка:* Під час створення різновиду мегафункції за допомо-

гою менеджера його параметри вже містяться у файлі, наприклад, 400mux8.tdf, тому достатньо оголосити зразок без зазначення параметрів (непараметризований зразок). Проте за використання стандартної мегафункції, наприклад, mux параметри зразка потрібно визначити в підсекції оголошення параметризованого зразка (на кшталт налаштування мегафункції в графічному редакторі).

7) Ввести логічний блок Logic Section.

```
BEGIN
END;
```

8) Між ключовими словами (BEGIN та END;) ввести оператор булевих рівнянь Boolean Equation та вставити в утворений таким чином шаблон рівняння через крапку з комою, якими з'єднати входи мегафункції з вхідними портами та вихідний порт – з виходом мегафункції.

```
BEGIN
    mx2.sel[2..0] = (in4, in[2..1]); mx2.data6 = !in3;
    mx2.(data7, data[5..0]) = H"32";  y2 = mx2.result;
    mx2a.sel[2..0] = (in4, in[2..1]); mx2a.data6 = !in3;
    mx2a.(data7, data[5..0]) = H"32"; y2a = mx2a.result;
END;
```

9) Виконати компіляцію і функціональне моделювання та переконатися в правильності проектування (для наочності див. файли 400gr_mega.bdf, 400gr_mega.vwf).

Приклад: 500mega.tdf, .vwf.

3 Створити в текстовому редакторі проект лабораторного макету для експериментального дослідження ЦКП на мультиплексорі згідно з варіантом 5.7 (див. структурну схему і задля наочності графічний аналог проекту 500gr_СКР.bdf на рис. Д3 у додатках).

Від ДІП-перемикачів MAX_SW1 вхідні змінні надходять на входи мультиплексорів, з виходів яких реалізована функція в інверсному вигляді індукується світлодіодами. Набори вхідних змінних через перетворювач двійкового коду в двійково-десятковий подаються на дешифратори 7-сегментного коду для подальшої індикації двома знакомісцями цифрового індикатора в десятковій системі. У прикладі за чотирьох змінних потрібно відобразити числа 0…15, тобто індикатор старшого розряду має або бути погашений, або індикувати цифру 1. Для цього потрібно молодшу тетраду ДДК подати на входи дешифратора 7-сегментного коду безпосередньо, а зі старшої тетради ДДК скористатися лише молодшим розрядом для гасіння або засвічування сегментів b та c (усі інші сегменти мають бути погашені).

3.1 Засвоїти ***сумісне включення*** до текстового проекту ***мегафункцій***

і ***макрофункцій,*** для чого створити новий проект 5XXCKP і однойменний текстовий файл (.tdf), зберегти із включенням до проекту цей файл та виконати такі дії (див. файл 500CKP.tdf).

а) Вставити до текстового файлу мегафункції аналогічно п. 2.3. Для цього можна скопіювати текстовий файл 5XXmega.tdf, залишити в ньому частину для реалізації пристрою на одній-двох мегафункціях, а інше видалити та виправити заголовок (Title Statement) і ім'я проекту (Subdesign Section). Відтак аналогічно п. 2.3.1,б ввести до складу проекту файли включення (.inc) і текстовий (.tdf) різновиду мегафункції, створеного менеджером (з Лаб. роботи №4, тип файлів Other Source Files).

б) Аналогічно п. 2.1 вставити до текстового файлу макрофункції перетворювача двійкового коду у двійково-десятковий і дешифратора 7-сег-ментного коду.

в) Доповнити файл портами (Subdesign Section), оголошенням зразків (Variable Section) та необхідними з'єднаннями компонентів (Boolean Equation). Виконати компіляцію і функціональне моделювання проекту.

✍ ***Примітка.*** Імена портів використовувати відповідно до файлу розведення їх на друкованій платі ../4lab/MAX_pin.

3.2 Імпортувати призначення виводів мікросхеми (див. Лаб. роботу №4, п. 5.2) з файлу ../4lab/MAX_pin.

3.3 Виконати компіляцію і функціональне моделювання проекту та переконатися в правильності проектування.

Приклад: 500CKP.tdf, .vwf.

3.4 Сформувати файл програматора і виконати фізичне програмування мікросхеми (див. Лаб. роботу №3, п. 3.5, 3.6).

Приклад: 500CKP.cdf.

3.5 Розробити методику та виконати необхідні експериментальні дослідження. Порівняти їх результати з даними проектних файлів, зробити висновки.

6 ЛАБОРАТОРНА РОБОТА №6. АРИФМЕТИЧНІ ПРИСТРОЇ

Мета роботи: дослідження типових арифметичних пристроїв – суматорів, арифметико-логічних пристроїв, помножувачів, подільників і компараторів; побудова ЦКП на суматорах і компараторах; засвоєння основ створення ієрархічного проекту на рівні блок-схеми.

ДОМАШНЄ ЗАВДАННЯ

Спроектувати заданий згідно з варіантом (див. Додатки) пороговий елемент на суматорах і компараторах.

СТИСЛІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Арифметичні пристрої: двійкові суматори, комбінаційні помножувачі, арифметико-логічні пристрої, цифрові компаратори розглянуто в [2]. Основні з них подано в табл. 6.1, 6.2.

Застосування арифметичних пристроїв для побудови ЦКП

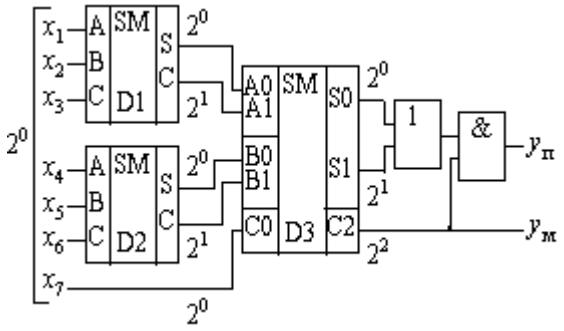
Арифметичні пристрої можуть використовуватись не тільки для виконання арифметичних операцій, але й для побудови пристроїв, пов'язаних із порівнянням кодів, визначенням кількості вхідних сигналів, що набувають значення лог. 1 тощо. До таких ЦП належать, зокрема, *порогові елементи* – ЦКП, на виході яких встановлюється рівень лог. 1, якщо не менш, ніж на k із довільної кількості m входів сигнали набувають рівня лог. 1. Окремим випадком порогових пристроїв є *мажоритарний елемент* – ЦКП, на виході якого виникає рівень лог. 1, якщо на більшості з його непарної кількості m входів діють рівні лог. 1.

Приклад. Для прикладу розглянемо принцип побудови мажоритарного елемента на $m = 7$ входів, тобто пристрою з порогом $k = 4$. Потрібна для цього кількість елементів І-НЕ різко зростає зі збільшенням m і стає неприйнятною. Ідея використання суматорів полягає в тому, що спочатку підраховуємо кількість логічних одиниць на входах, а потім встановлюємо, чи не перевищує вона потрібний поріг k.

Нагадаємо попередньо, що індекси i на вхідному і вихідному полях умовного графічного позначення суматора (див. рис. 6.1,*є*) вказують на вагу його розрядів 2^i. З огляду на це підраховуємо кількість одиниць на входах $x_1 ... x_7$ за допомогою двох однорозрядних та одного дворозрядного суматорів (рис. 1,*а*), виконуючи з'єднання з урахуванням ваги розрядів. Підсумок зчитуємо з виходів останнього суматора як $S = c_2 2^2 + s_1 2^1 + s_0 2^0$, тобто у вигляді двійкового числа $c_2 s_1 s_0$. Перенесення $y_М = c_2$ і є виходом мажоритарного елемента "4 з 7", бо $S \geq 4$ при $c_2 = 1$.

Аналогічно проектуємо й пороговий елемент, наприклад, при $m = 7$, $k = 5$. Згідно з таблицею (рис. 6.1,*б*) мінімізуємо його функцію (рис. 6.1,*в*) $y_П = (s_0 + s_1)\, c_2$ та реалізуємо її на основі тієї ж схеми (див. рис. 6.1,*а*).

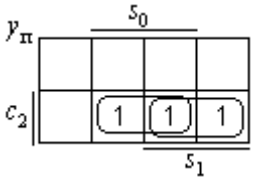
Отже, на виході $y_М$ встановлюється рівень лог. 1 за виникнення одиниць не менш, ніж на 4 входах, а на виході $y_П$ – не менш, ніж на 5 входах із 7.



а)

c_2	s_1	s_0	$y_П$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

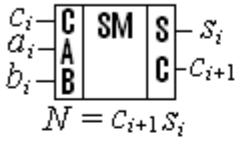
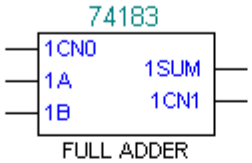
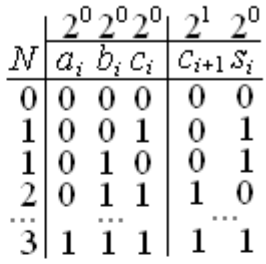
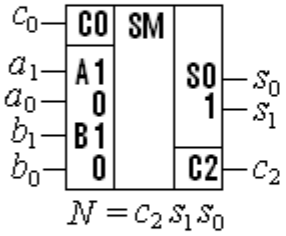
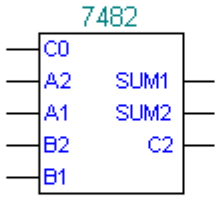
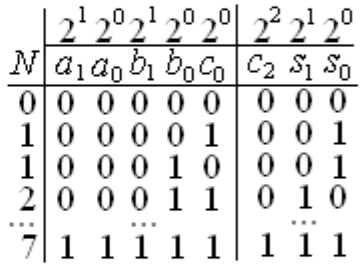
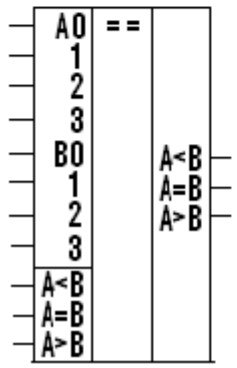
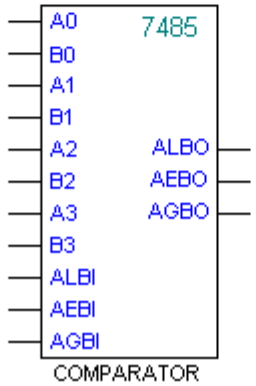
б)



в)

Рисунок 6.1

Таблиця 6.1 – Макрофункції арифметичних пристроїв

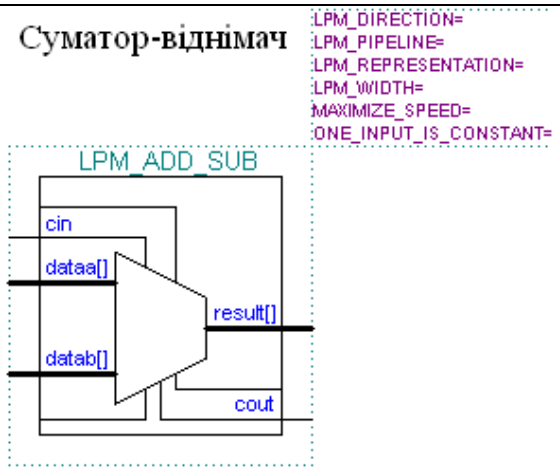
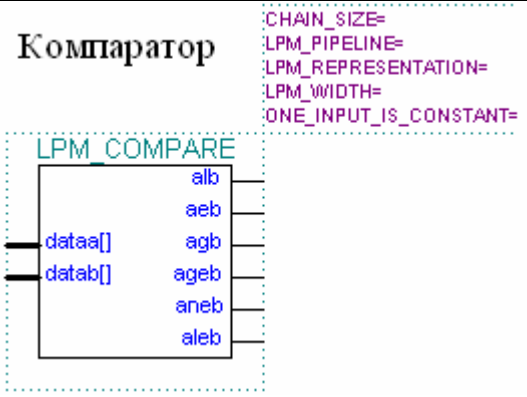
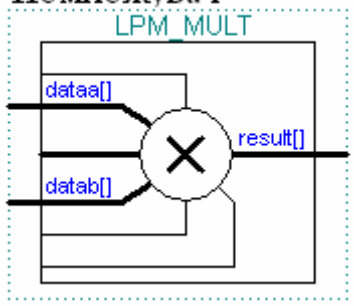
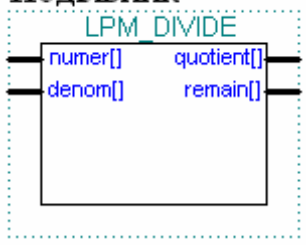
Позначення за ДЕСТУ	Типова макрофункція	Таблиця відповідності
Суматор повний		
		
Суматор багаторозрядний		
		
Компаратор цифровий		
		

N	2^0 a_i	2^0 b_i	2^0 c_i	2^1 c_{i+1}	2^0 s_i
0	0	0	0	0	0
1	0	0	1	0	1
1	0	1	0	0	1
2	0	1	1	1	0
…		…		…	
3	1	1	1	1	1

N	2^1 a_1	2^0 a_0	2^1 b_1	2^0 b_0	2^0 c_0	2^2 c_2	2^1 s_1	2^0 s_0
0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	1
1	0	0	0	1	0	0	0	1
2	0	0	0	1	1	0	1	0
…			…				…	
7	1	1	1	1	1	1	1	1

Входи A, B	Виходи ALB	AEB	AGB
A<B	1	0	0
A=B	0	1	0
A>B	0	0	1

Таблиця 6.2 – Мегафункції арифметичних пристроїв

Символ	Основні параметри
Суматор-віднімач 	lpm_add_sub (Adder/Subtractor): LPM_WIDTH – розрядність вхідних і вихідної шин; LPM_REPRESENTATION – зображення даних: зі знаком ("SIGNED"), без знаку ("UNSIGNED"); LPM_DIRECTION – тип операції: додавання ("ADD"), віднімання ("SUB"); cin (Carry-in), cout (Carry-out) – вхідний і вихідний переноси.
Компаратор 	lpm_compare (Comparator): LPM_WIDTH – розрядність вхідних шин dataa[] = A та datab[] = B. *Входи* *Функція* A < B alb A = B aeb A > B agb A ≥ B ageb A ≠ B aneb A ≤ B aleb
Помножувач 	lpm_mult (Multiplier): LPM_WIDTHA – розрядність множеного dataa[]; LPM_WIDTHB – розрядність множника datab[]; LPM_WIDTHP = LPM_WIDTHA + + LPM_WIDTHB – розрядність добутку result[].
Подільник 	lpm_divide (Divider): LPM_WIDTHN – розрядність діленого numer[] і частки quotient[]; LPM_WIDTHD – розрядність дільника denom[] і залишку remain[].

КОНТРОЛЬНІ ЗАПИТАННЯ

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

2. Реалізуйте на основі заданих в дужках пристроїв і, за необхідністю, 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) мажорритарні елементи: 1) на 5 входів, 2) на 9 входів;

б) порогові елементи: 3) 2 з 5, 4) 3 з 6, 5) 3 з 7, 6) 4 з 5, 7) 4 з 6, 8) 4 з 8, 9) 4 з 9, 10) 5 з 7, 11) 5 з 8, 12) 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: 6) 4-розрядних компараторів, 7) 4-розрядних суматорів, 8) 4-розрядних АЛП.

ЛАБОРАТОРНЕ ЗАВДАННЯ

1 Дослідити типові арифметичні пристрої.

1.1 Дослідити суматори: за принциповими електричними схемами (файл 6sum.bdf) та осцилограмами сигналів (6sum.vwf) ***півсуматора*** (схема 1) і однорозрядного повного ***суматора*** (схема 2) на логічних елементах І-АБО-НЕ скласти таблиці 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).

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).

1.3 Дослідити компаратори: за принциповими електричними схемами (файл 6comp.bdf) та осцилограмами сигналів (6 comp.vwf) ***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).

1.4 Ознайомитися зі ***стандартними арифметичними пристроями*** серії 74 (макрофункціями) та параметризованими мегафункціями (файл 6libr.bdf): суматорів і суматора-віднімача (Adder Macrofunctions, Parameterized Adder/Subtractor Megafunction); АЛП (Arithmetic Logic Unit Macrofunctions); помножувачів і подільника (Multiplier Macrofunctions, Parameterized Multiplier Megafunction, Parameterized Divider Megafunction) 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.

2 У текстовому редакторі створити проект нижнього рівня ієрархії як блоку для проекту верхнього рівня ієрархії

2.1 Задля наочності створити проект у *графічному* редакторі блоку суматора, призначеного для підсумовування кількості сигналів одиничного рівня на входах заданого варіанту порогового елемента.

а) Визначити ім'я проекту 6XXgr_sum і створити однойменний графічний файл блоку із включенням його до складу цього проекту.

б) Зібрати пристрій для підсумовування кількості N одиничних рівнів вхідних сигналів. Перший ступінь скласти на макрофунк-

ціях повних суматорів дібраного типу (з бібліотеки 6libr.bdf або з пакету MAX+ PLUS II > Old Style Macrofunctions). У другому ступені за допомогою вбудованого майстра MegaWizard Plug-In Manager вставити мегафункцію багаторозрядного суматора з іменем зразка, наприклад, sum2 (з категорії arithmetic > LPM_ADD_SUB, режим операції Addition only – тільки додавання, якщо потрібно, встановити прапорці Create a carry input та Create a carry output для створення вхідного cin і вихідного cout переносів). Відтак вставити до файлу вхідний і вихідний порти, виконати з'єднання компонентів схеми та дати імена портам і лініям у місцях розщеплення шин або об'єднання до шин. На завершення зберегти зміни у файлі.

✐ *Примітки:*

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

2) У прикладі введено додатковий нульовий старший розряд dataa[3] = 0 для зручності з'єднання виходів суматора dataa[3..0] з вхідною тетрадою дешифратора семисегментного коду (якщо його вхід D вилучити, то згідно з довідкою MAX+PLUS II за умовчанням він з'єднується з логічною одиницею VCC).

в) Скомпілювати проект, створити файл часових діаграм, виконати функціональне моделювання та переконатися в правильності проектування пристрою.

***Приклад**:* 600gr_sum.bdf, .vwf.

✐ *Примітки:*

1) Результати моделювання автоматично переносяться на файл часових діаграм проекту, якщо з меню Tools > Options > General > Processing на вкладці Processing ввімкнути прапорець Overwrite simulation input file with simulation result та натиснути кнопку OK. Проте повторне моделювання після внесення змін у вхідних сигналах може виявитися зручнішим без цієї опції: можна порівняти діаграми у звітному файлі після моделювання з діаграмами до внесення змін.

2) Вхідний і вихідний коди зручно зображати в шістнадцятковій і десятковій системі числення відповідно. Для зміни системи слід виділити діаграму групи і піктограмою Properties (або B2 > Properties) викликати діалогове вікно Node Properties (властивості вузла) та в рядку Radix (система числення) прокруткою ввести Hexadecimal (шістнадцяткова система) або Unsigned Decimal (десяткова система числення без знаку).

2.2 Користуючись графічним файлом 6XXgr_sum як зразком, створити проект блоку суматора у *текстовому* редакторі.

а) Створити новий проект з ім'ям 6XXsum, яке має точно збігатися з іменем цього блоку в проекті верхнього рівня, та включити до складу проекту файл sum2 суматора другого ступеня з проекту 6XXgr_sum (вставити його кнопками огляду та Add на другій сторінці вікна New Project Wizard).

б) Створити новий текстовий файл 6XXsum.tdf і включити його до проекту 6XXsum: натиснути піктограму збереження файлу, у діалоговому вікні встановити директорію своєї теки, вибрати тип файлу AHDL File (*.tdf), ввести ім'я 6XXsum, підняти прапорець Add file to current project та натиснути кнопку збереження.

в) Ввести заголовок Title Statement (під кожним пунктом наводиться приклад зразка 600sum.tdf).

```
TITLE "Блок суматора для файлу блок-схеми";
```

г) У секції Include Statement включити до проекту автоматично створений різновид мегафункції другого ступеня суматора sum2.

```
INCLUDE  "sum2.inc";
```

ґ) У секції Function Prototype Statement (non-parameterized) ввести прототипи макрофункцій суматорів першого ступеня (записати дані із символу або скопіювати з довідки Help пакету MAX+PLUS II).

```
FUNCTION 74183 (1cn0, 1a, 1b, 2cn0, 2a, 2b)
    RETURNS (1sum, 1cn1, 2sum, 2cn1);
```

☝ *Примітка:* Прототип макрофункції певного типу вставляється тільки один раз, навіть якщо зразок цієї макрофункції використовується в проекті неодноразово. Якщо вставляється декілька різних макрофункцій, їх прототипи вводяться по черзі повністю: FUNCTION <ім'я 1> (…) RETURNS (…); FUNCTION <ім'я 2> (…) RETURNS (…); і т. д.

д) У секції Subdesign Section повторити ім'я проекту 6XXsum та оголосити вхідні і вихідні порти.

```
SUBDESIGN 600sum
(
    sw1[7..1]    : INPUT;
    dataa[3..0]  : OUTPUT;
)
```

☝ *Примітка:* Порти sw1[7..1] є зовнішніми виводами мікросхеми, тому названі стандартно, як у файлі розведення ../4lab/MAX_pin.

е) Ввести секцію змінних Variable Section.

```
VARIABLE
```

є) Вставити підсекцію Instance Declaration (non-parameterized) і ввести імена зразків та (через двокрапку) типи оголошених у п. 2.2, г, ґ макро- і мегафункцій (для наочності у файлі 600gr_sum.bdf імена зразків позначено для суматорів першого і другого ступеня).

```
    sm1   : 74183;  sm2 :  sum2;
```

ж) Викликати логічну секцію Logic Section;

```
BEGIN
END;
```

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

```
BEGIN
    sm1.1cn0=sw1[2]; sm1.1a=sw1[3]; sm1.1b=sw1[4];
```

```
sm1.2cn0=sw1[5];   sm1.2a=sw1[6]; sm1.2b=sw1[7]; sm2.cin=sw1[1];
sm2.dataa[]=(sm1.1cn1, sm1.1sum); sm2.datab[]=(sm1.2cn1, sm1.2sum);
   dataa[]=(GND, sm2.cout, sm2.result[]);
END;
```

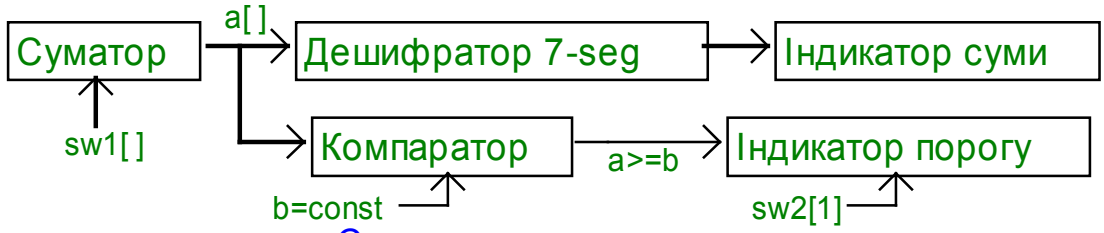
и) Виконати компіляцію і моделювання та переконатися в правильності функціонування пристрою за часовими діаграмами 6XXsum.vwf.

Приклад: 600sum.tdf, .vwf.

і) Відкрити (якщо це не зроблено) файл вершини текстового проекту 6XXsum.tdf і створити символьний файл блоку суматора: меню File > Create/Update > Create Symbol Files for Current File > OK в інформаційному віконці про успішне генерування символьного файлу. Відтак відчинити вікно символьного файлу (з типу файлів Graphic Files або Other Source Files) та, у разі потреби, зменшити розмір символу: клацнути сторону внутрішнього прямокутника, скоригувати його розмір, клацнути для виділення написи, перетягнути їх ближче до центру та скоригувати розмір зовнішнього прямокутника і зберегти файл.

Приклад: 600sum.bsf.

3 Засвоїти основи створення ієрархічного проекту на рівні блок-схеми для побудови порогового елемента, заданого згідно з варіантом XX.



Структурна схема

Від перемикача MAX_SW1 (див. файл ../6lab/600POR_EL.bdf на рис. Д1) сигнали sw1[] надходять на входи суматора, на виходах якого формується код dataa[], що відображає кількість одиничних рівнів на входах. Для індикації цієї кількості сума через дешифратор 7-сегмнтного коду подається на 7-сегментний індикатора MAX_DIGIT (якщо сума перевищує 9, її двійковий код попередньо необхідно перетворити в ДДК на кшталт ..5lab/500gr_CKP). У компараторі сума dataa[] порівнюється з пороговою величиною b і коли досягає її, на виході компаратора утворюється рівень лог. 1, що індикується світлодіодом D1. Дозвіл на індикацію порогу задається перемикачем MAX_SW2.

3.1 ***Визначити ім'я проекту та створити файл блок-схеми*** (файл верхнього рівня ієрархії проекту).

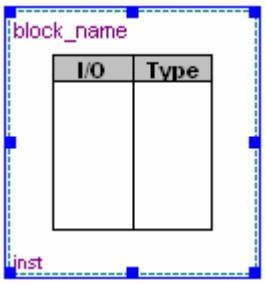
Примітка. Для виконання наступних дій доцільно користуватися зразком блок-схеми у файлі ../6lab/600POR_EL.bdf (рис. Д1).

а) Створити новий проект 6XXPOR_EL, на другій сторінці майстра New Project Wizard включити до складу проекту заздалегідь створені текстові файли блоку суматора 6XXsum.tdf (п. 2.2) та

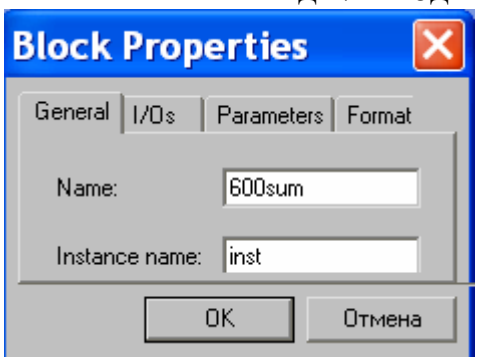
sum2.tdf (п. 2.1,б), на четвертій – вибрати родину мікросхем MAX7000S, на п'ятій – тип корпусу PLCC, кількість виводів 84, швидкодію Any (будь-яка) та мікросхему зі списку приступних типів EPM7128SL84-7.

б) Створити новий графічний файл з безіменною назвою BlockN.bdf (N – довільний номер).

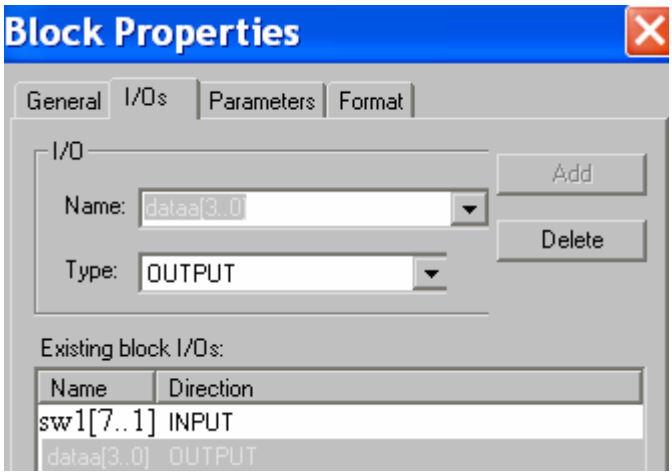
3.2 ***Вставити до блок-схеми функціональний блок.***



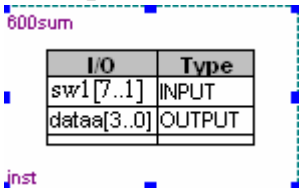
а) Інструментом палітри вставлення блоку Block Tool натиснути в полі файлу лівою кнопкою миші і, утримуючи її, протягнути по діагоналі для утворення прямокутної області приблизно потрібного розміру. По відпусканні кнопки з'явиться виділений безіменний блок з порожньою мапою його зовнішніх входів/виходів (I/O).



б) Виділити, якщо це не зроблено, блок і натиснути інструмент Properties (властивості) для виклику діалогового вікна Block Properties (або B2 > Block Properties). На вкладці General (загальні) у полі Name ввести ім'я блоку суматора 6XXsum, таке саме, що й у текстовому файлі, в якому створено цей блок (п. 2.2).



в) Перейти на вкладку I/Os (входи/виходи), у полі Name ввести ім'я групи входів, наприклад, sw1[7..1], у полі Type вибрати INPUT і натиснути кнопку Add (додати) – у списку існуючих входів/виходів блоку з'явиться рядок групи входів. Відтак повторити ці дії для групи виходів: dataa[3..0] > OUTPUT > Add. По натисненні кнопки OK зроблені призначення переносяться на мапу всередині блоку.



г) На блоці натиснути праву кнопку миші B2 і в контекстному меню вибрати команду AutoFit (або виділити блок і в меню Edit вибрати AutoFit) – блок автоматично набирає такого розміру, щоб до нього добре вписалась мапа входів/виходів.

ґ) Таким самим чином (п. а ... г) вставити інші блоки, якщо вони є. Відтак натиснути піктограму збереження файлу Save, у діалоговому вікні підняти прапорець Add file to current project для включення файлу до проекту та натиснути кнопку збереження – файлові автоматично надається назва верхнього рівня проекту і в ньому зберігаються зміни.

✍ *Примітка:* Процедура вставлення блоку за п. 3.2 є універсальною: файл блоку можна створити як до, так і після створення файлу блок-схеми. Проте у випадку, коли файл блоку і його символ створено заздалегідь (як у нашому прикладі – у п. 2.2), блок можна вставити як символ: двічі клацнути в полі файлу (або натиснути інструмент вставлення символу) і в діалоговому вікні Symbol вибрати Project > 6XXsum > Insert symbol as block > OK.

3.3 ***Створити і вставити до блок-схеми примірник мегафункції.***

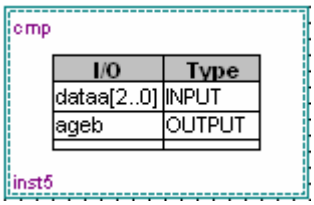
а) Двічі клацнути в полі файлу (або натиснути інструмент вставлення символу) і в діалоговому вікні Symbol натиснути кнопку MegaWizard Plug-In Manager. На першій сторінці вибрати створення нового зразка мегафункції (Create …).

б) На другій сторінці зі списку мегафункцій ліворуч вибрати Installed Plug-Ins (установки майстра) > категорію Arithmetic > тип LPM_CO-MPARE. У верхніх рядках праворуч залишити раніше встановлену родину IC MAX7000S (інакше вибрати її прокруткою) та тип вихідного файлу AHDL. У наступному рядку до встановленої директорії теки (інакше вибрати її кнопкою огляду Browse) ввести без прогалини ім’я вихідного файлу, наприклад, cmp.

в) На третій сторінці у верхньому віконці прокруткою вибрати розрядність у бітах порівнюваних слів на входах компаратора ‘dataa’ та ‘datab’. Виходячи з максимально можливого числа (7) на виході суматора, яке подаватимемо на вхід‘dataa’ компаратора, вибираємо у нашому прикладі розрядність 3 біта. Відтак вибрати вихідну функцію компаратора встановленням прапорця a > = b (greater than or equal – більше або дорівнює), а інші прапорці мають бути зняті.

г) На четвертій сторінці у верхній закладці на питання, чи має бути на вхідній шині ‘datab’ константа, дати ствердну відповідь (Yes) і ввести десяткове значення константи, наприклад, для порогового елемента “5 з 7” вводимо число 5. На питання про тип порівнюваних чисел слід залишити Unsigned (без знаку).

ґ) П’яту і шосту сторінки слід проминути (Next), на сьомій, останній сторінці продивитися список сформованих файлів та натиснути кнопку Finish – у вікні Symbol з’явиться символ налаштованого різновиду мегафункції компаратора (скоригувати свої дії можна поверненням на попередні сторінки кнопкою Back).



д) У вікні Symbol встановити прапорець Insert symbol as block (вставити символ як блок) та натиснути кнопку ОК – це вікно зачиниться, а символ мегафункції буде прив’язаний до покажчика миші. Для вставлення його до файлу блок-схеми клацнути в полі файлу (у разі потреби, натиснути інструмент вибору – стрілку, щоб звільнити покажчик від символу) та піктограмою зберегти файл (зникне зірочка в рядку його заголовка).

✍ *Примітка:* На символі зразка мегафункції відображено основні

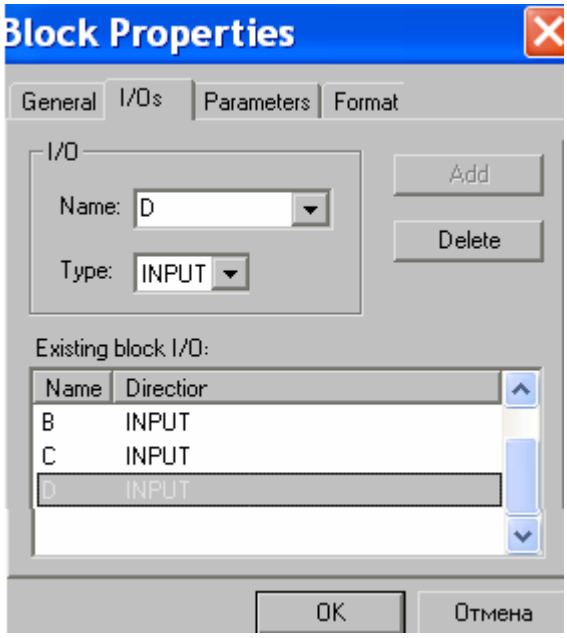
параметри, тому команда Show Parameter Assignments (показати призначення параметрів) і відповідний інструмент палітри для такого символу не діють. Продивитися параметри можна в текстовому файлі cmp.tdf.

е) Відкрити і продивитися текстовий файл cmp.tdf створеного примірника мегафункції та включити його до складу проекту: меню Project > Add Current File to Project.

3.4 ***Вставити до блок-схеми макрофункцію.***

а) З метою індикації кількості одиничних сигналів, визначених суматором, вибрати з файлу ../3lab/3libr.bdf макрофункцію дешифратора семисегментного коду з інверсними виходами і вставити його до файлу блок-

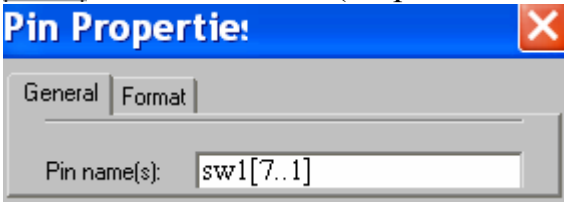
схеми. Для цього двічі клацнути в полі файлу (або натиснути інструмент вставлення символу) і в діалоговому вікні Symbol розгортанням каталогу бібліотеки Libraries вибрати c:/quartus/libraries > others > maxplus2 > ім'я макрофункції, наприклад, 74246. Відтак встановити прапорець Insert symbol as block (вставити символ як блок) та натиснути кнопку ОК – це вікно зачиниться, а символ макрофункції буде прив'язаний до покажчика миші. Для вставлення його до файлу блок-схеми клацнути в полі файлу, та зберегти файл (зникне зірочка в рядку його заголовка).



б) Настроїти параметри макрофункції, для чого виділити (у разі потреби) її символ і натиснути інструмент Properties (властивості) для виклику діалогового вікна Block Properties (або B2 > Block Properties). З огляду на те, що в прикладі не використовуватимемо вхід гасіння BIN та входи і вихід керування RBIN, LTN, RBON, їх слід вилучити з блоку. Для цього на вкладці I/Os (входи/виходи), у розділі Existing block I/Os (існуючі входи/виходи блоку) виділити рядок BIN і натиснути кнопку Delete (видалити). Так само вилучаємо інші зазначені виводи та натискаємо кнопку ОК – зміни переносяться на символ файлу блок-схеми. Кнопкою B2 > AutoFit нормалізуємо розмір блоку. У разі потреби, так само вставити до файлу блок-схеми інші макрофункції повторенням п. 3.4.

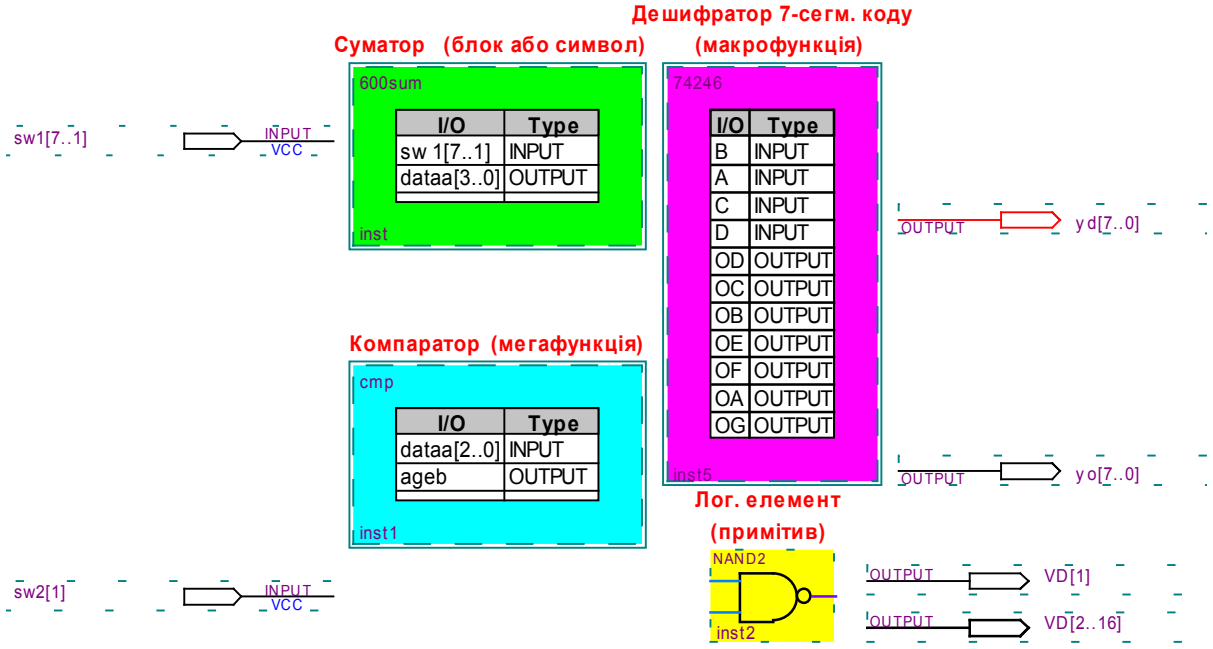
3.5 ***Вставити до блок-схеми примітиви і ввести імена портів.***

а) Вставити звичайним чином до файлу блок-схеми логічний елемент I-НЕ (c:/quartus/libraries > primitives > logic > nand2) та вхідні і вихідні порти (c:/quartus/libraries > primitives > pin > input або output).



б) Дати імена входам і виходам, для чого викликати діалогове вікно Pin

Properties (властивості виводу): двічі клацнути символ порту (або виділити символ і натиснути інструмент Properties, або клацнути B2 по символу > Properties). Відтак на вкладці General (загальні) до рядка Pin name(s) ввести ім'я сигналу, наприклад, sw1[7..1] та натиснути кнопку OK. Повторенням цих дій дати імена всім виводам IC.



3.6 ***Виконати з'єднання компонентів блок-схеми.***

а) Інструментом ортогональних шин (Orthogonal Bus Tool) провести шини між групою вхідних портів sw1[7..1], блоками 600sum та 74246 і групою вихідних портів yd[7..0]. Крім того, вивести шину зі входів компаратора cmp для подальшого з'єднання з блоком 600sum, а також для погашення невикористовуваних світлодіодів і знакомісця індикатора з'єднати порти VD[2..16] та yo[7..0] з примітивами VCC (див. ілюстрацію нижче).

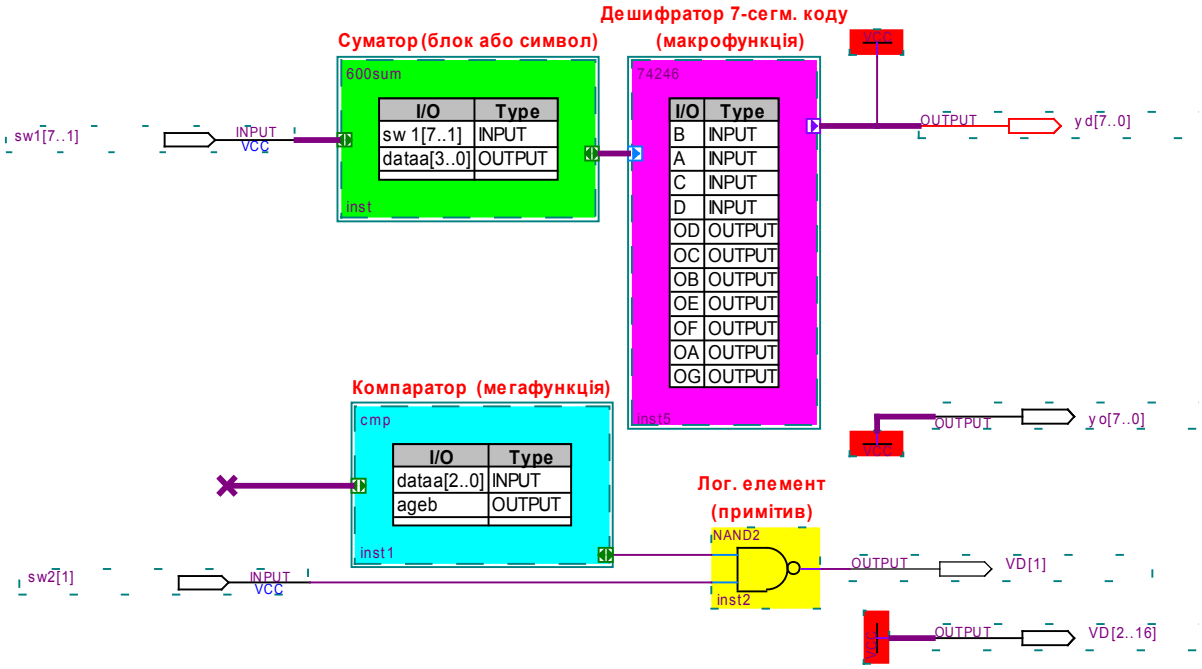
б) Інструментом ортогональних ліній (Orthogonal Node Tool) з'єднати входи елемента I-НЕ з виходом компаратора cmp і портом sw2[1], а його вихід – з портом VD[1] та зберегти файл.

Примітки:

1) Для зручності виконання з'єднань доцільно нанести сітку з меню View > Show guidelines (позиціонування елементів у графічному редакторі завжди є прив'язане до сітки, а в символьному редакторі це можна зробити з меню Tools > Options > Block / Symbol Editor > Snap to grid > OK).

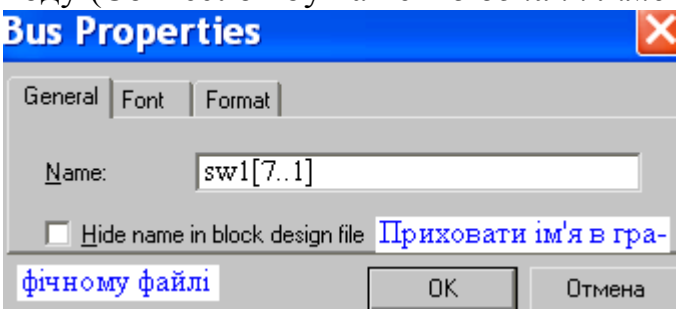
2) Більшість з'єднань можна виконати інструментом вибору і гнучкого рисування Selection and Smart Drawing Tool (стрілка), який автоматично перемикається на рисування необхідних ліній при підведенні його до компонентів, блоків або виконаних з'єднань.

3) Під час з'єднання лінії з блоком на його облямівці з'являється символ трасувальника (Mapper).



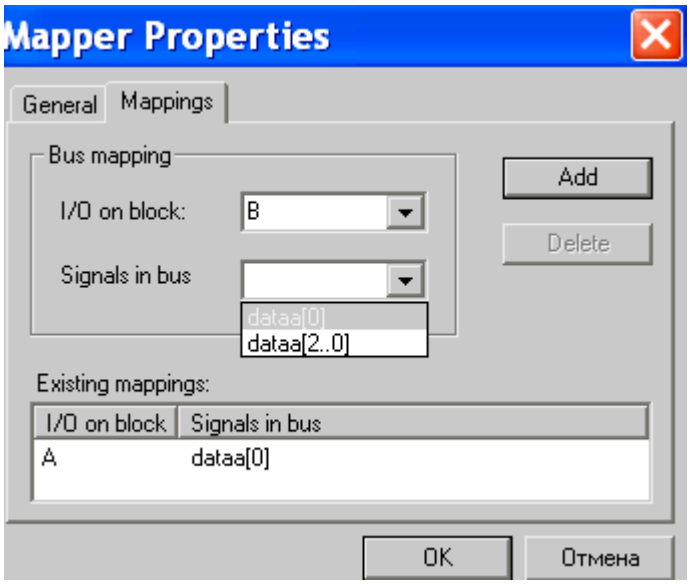
3.7 ***Виконати розведення сигналів між блоками.***

а) Якщо з'єднувальна лінія має таку саму назву, що й вхід або вихід блоку (подані на мапі всередині блоку), то вона приєднується до цього виводу (Connection by name – *з'єднання іменами*). Щоб з'єднати іменами вхідні сигнали sw1[7..1] зі входами блоку 600sum, необхідно двічі клацнути шину (або клацнути кнопкою B2 і в контекстному меню вибрати Properties, або виділити шину і натиснути на панелі інструмент Properties). У діалоговому вікні Bus Properties (властивості шини) на вкладці General (загальні), у віконці Name (ім'я) ввести назву шини sw1[7..1] та натиснути OK. Так само з'єднати іменами вихід блоку 600sum із шиною dataa[3..0], вихід компаратора cmp зі входом елемента І-НЕ введенням у діалоговому вікні Node Properties (властивості лінії) ім'я лінії ageb. Інший вхід елемента І та його вихід з'єднано з портами безпосередньо лініями, які не потребують введення імен (див. рис. Д4 у додатках).



б) Якщо імена двох ліній збігаються, то вони з'єднуються *автоматично* (Smart), без рисування з'єднань. Такий тип з'єднання можна створити між виходами блоку 600sum і входами блоку cmp наданням відповідним шинам однакової назви. У прикладі шина dataa[2..0] автоматично з'єднується з однойменними розрядами шини dataa[3..0] (див. рис. Д4 у додатках).

в) Якщо імена входів/виходів на мапі блоку і з'єднувальних ліній різні, то розведення сигналів виконують за допомогою *трасувальників* (Mappers). Для з'єднання розрядів шини dataa[3..0] зі входами блоку 74246 слід підвести інструмент вибору об'єктів до трасувальника в місці з'єднання



шини із цим блоком і, коли біля стрілки з'явиться символ таблиці, двічі клацнути трасувальник (або B2 > Mapper Properties). У діалоговому вікні Mapper Properties (властивості трасувальника) на вкладці General (загальні) вибрати тип виводу стосовно даного блоку INPUT (вхід). На вкладці Mappings (трасування сигналів) у рядку I/O on block (вхід/вихід блоку) прокруткою вибрати вхід A, а в рядку Signals in bus (сигнали в шині) – dataa[3..0] виправити на dataa[0] та натиснути кнопку Add (додати). У віконці Existing mappings (існуючі трасування) відобразиться з'єднання входу A з молодшим бітом шини. Так само з'єднати входи B, C, D з розрядами шини dataa[1], dataa[2], dataa[3] та натиснути кнопку OK – результати відобразяться в трасувальній таблиці, прив'язаній до трасувальника. Аналогічним чином розвести сигнали на виходах блоку дешифратора 74246 та зберегти зміни у файлі (див. рис. Д4 у додатках).

✍ *Примітки:*

1) Для погашення десяткової крапки на лінію yd[7] її сегменту індикатора подано лог. 1 (VCC).

2) Під час утворення трасувальних таблиць символи трасувальника на входах і виходах набувають різного вигляду за напрямком стрілки і кольором.

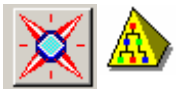
3) Таблиці трасувань біля символів трасувальників відображаються або приховуються командою меню View > Show Mapper Tables. Трасувальні таблиці можна перетягнути в зручну позицію звичайним чином лівою кнопкою миші (або виділити і пересунути з клавіатури клавішами керування курсором зі стрілками). Командою меню View > Show Block I/O Tables можна відобразити або приховати мапи входів/виходів всередині блоків.

4) Зображення лінії і тло блоку можна виділити кольором піктограмою Properties на вкладці Format відповідного діалогового вікна.

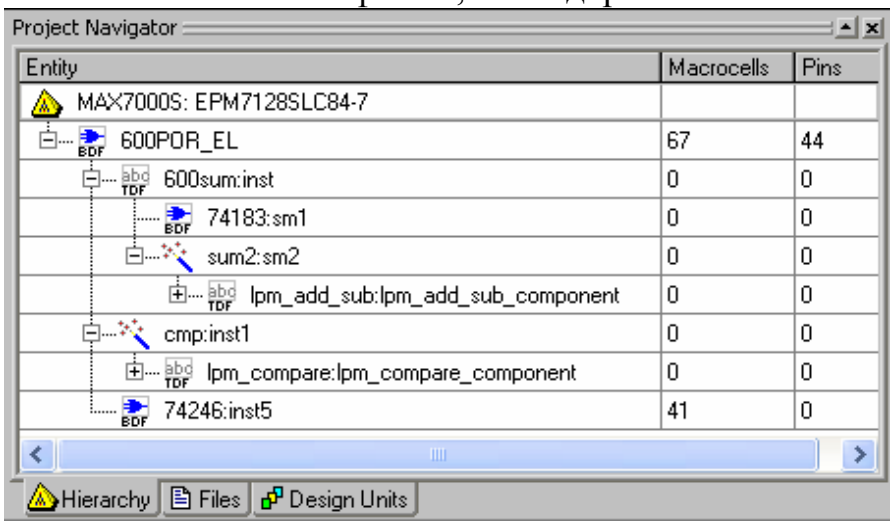
3.8 ***Імпортувати призначення виводів мікросхеми*** (див. Лаб. роботу №4, п. 5.2) з файлу ../4lab/MAX_pin.

3.9 ***Виконати компіляцію і функціональне моделювання*** проекту та переконатися в правильності проектування.

Приклад: 600POR_EL.bdf, .vwf.



3.10 ***Продивитися складники проекту***: піктограмою Project Navigator (або з меню View > Utility Windows > Project Navigator) відкрити вікно навігатора і кнопкою Hierarchy внизу цього вікна перейти на вкладку ієрархії. Розгортанням (+) та згортанням (-) гілок ієрархічного дерева спостерігати компоненти проекту. Подвійним клацанням по значках або іменах файлів розкрити їх і ознайомитися в загальних рисах зі змістом. Кнопкою Files внизу вікна перейти на однойменну вкладку і ознайомитися зі списком файлів, які відкриваються так само. Кнопкою Design Units внизу вікна перейти на вкладку модулів проекту, які відкриваються так само, подвійним клацанням. Відтак піктограмою навігатора (або значком „x") зачинити його вікно.



3.11 ***Сформувати файл програматора і виконати фізичне програмування мікросхеми*** (див. Лаб. роботу №3, п. 3.5, 3.6).

Приклад: 600POR_EL.cdf.

3.12 ***Розробити методику та виконати необхідні експериментальні дослідження.*** Порівняти їх результати з даними проектних файлів, зробити висновки.

7 ЛАБОРАТОРНА РОБОТА №7. ТРИГЕРИ

Мета роботи: дослідження основних типів тригерів і поширених радіотехнічних кіл на тригерах; засвоєння основ застосування тригерів у проектах, створення ієрархічного проекту на рівні блок-схеми з використанням тригерів.

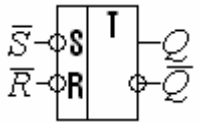
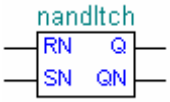
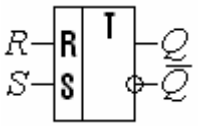
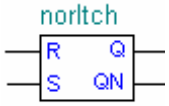
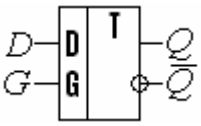
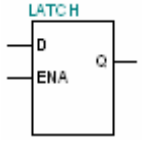
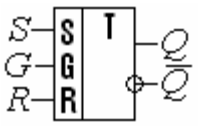
ДОМАШНЄ ЗАВДАННЯ

Для заданого варіанту (див. Додатки, завдання 7) розробити лінію передачі даних із фіксацією вихідного коду на тригерах.

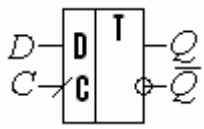
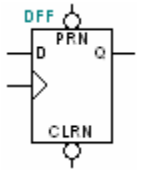
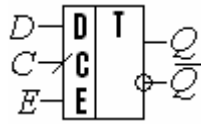
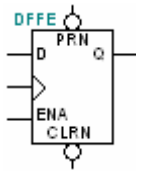
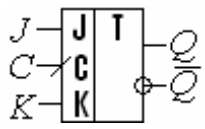
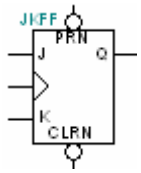
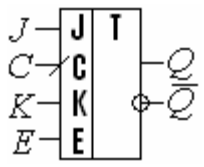
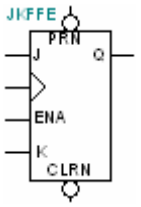
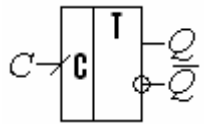
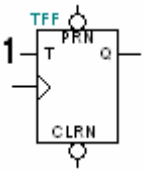
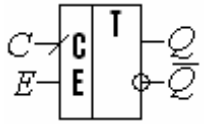
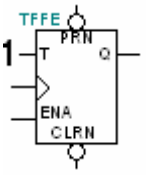
СТИСЛІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Означення, класифікацію та схемну реалізація тригерів розглянуто в [2]. Основні типи асинхронних тригерів і синхронних тригерів зі статичним керуванням (Latches) подано в табл. 1, а тригерів з динамічним керуванням (Flipflops) – у табл. 2. Перетворення тригерів з динамічним керуванням наведено на рис. 7.1.

Таблиця 7.1 –Асинхронні і синхронні тригери зі статичним керуванням

Тип	Позначення		Таблиця	
	ДЕСТУ	САПР	перемикальна	переходів
$\overline{R}\,\overline{S}$			$\overline{R}\,\overline{S}\mid Q^+$ 0 0 | X 0 1 | 0 1 0 | 1 1 1 | Q	$Q Q^+ \mid \overline{R}\,\overline{S}$ 0 0 | X 1 0 1 | 1 0 1 0 | 0 1 1 1 | 1 X
$R\,S$			$R\,S\mid Q^+$ 0 0 | Q 0 1 | 1 1 0 | 0 1 1 | X	$Q Q^+ \mid R\,S$ 0 0 | X 0 0 1 | 0 1 1 0 | 1 0 1 1 | 0 X
D			$G\,D\mid Q^+$ 0 X | Q 1 0 | 0 1 1 | 1	$Q Q^+ \mid D$ 0 0 | 0 0 1 | 1 1 0 | 0 1 1 | 1 (G = 1)
RSG		–	$G\,R\,S\mid Q^+$ 0 X X | Q 1 0 0 | Q 1 0 1 | 1 1 1 0 | 0 1 1 1 | X	$Q Q^+ \mid R\,S$ 0 0 | X 0 0 1 | 0 1 1 0 | 1 0 1 1 | 0 X (G = 1)

Таблиця 7.2 –Синхронні тригери з динамічним керуванням

Тип	Позначення		Таблиця	
	ДЕСТУ	САПР	перемикальна	переходів
D			$\begin{array}{cc\|c} C & D & Q^+ \\ \hline - & X & Q \\ \uparrow & 0 & 0 \\ \uparrow & 1 & 1 \end{array}$ "_" = **0**, **1**, ↓	$\begin{array}{cc\|c} Q & Q^+ & D \\ \hline 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{array}$ ($C = \uparrow$)
DE			$\begin{array}{ccc\|c} E & C & D & Q^+ \\ \hline 0 & X & X & Q \\ 1 & - & X & Q \\ 1 & \uparrow & 0 & 0 \\ 1 & \uparrow & 1 & 1 \end{array}$	$\begin{array}{cc\|c} Q & Q^+ & D \\ \hline 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{array}$ ($E = 1$, $C = \uparrow$)
JK			$\begin{array}{ccc\|c} C & J & K & Q^+ \\ \hline - & X & X & Q \\ \uparrow & 0 & 0 & Q \\ \uparrow & 0 & 1 & 0 \\ \uparrow & 1 & 0 & 1 \\ \uparrow & 1 & 1 & \overline{Q} \end{array}$	$\begin{array}{cc\|cc} Q & Q^+ & J & K \\ \hline 0 & 0 & 0 & X \\ 0 & 1 & 1 & X \\ 1 & 0 & X & 1 \\ 1 & 1 & X & 0 \end{array}$ ($C = \uparrow$)
JKE			$\begin{array}{cccc\|c} E & C & J & K & Q^+ \\ \hline 0 & X & X & X & Q \\ 1 & - & X & X & Q \\ 1 & \uparrow & 0 & 0 & Q \\ 1 & \uparrow & 0 & 1 & 0 \\ 1 & \uparrow & 1 & 0 & 1 \\ 1 & \uparrow & 1 & 1 & \overline{Q} \end{array}$	$\begin{array}{cc\|cc} Q & Q^+ & J & K \\ \hline 0 & 0 & 0 & X \\ 0 & 1 & 1 & X \\ 1 & 0 & X & 1 \\ 1 & 1 & X & 0 \end{array}$ ($E = 1$, $C = \uparrow$)
T			$\begin{array}{c\|c} C & Q^+ \\ \hline - & Q \\ \uparrow & \overline{Q} \end{array}$	$\begin{array}{cc\|c} Q & Q^+ & C \\ \hline 0 & 0 & - \\ 0 & 1 & \uparrow \\ 1 & 0 & \uparrow \\ 1 & 1 & - \end{array}$
TE			$\begin{array}{cc\|c} E & C & Q^+ \\ \hline 0 & X & Q \\ 1 & - & Q \\ 1 & \uparrow & \overline{Q} \end{array}$	$\begin{array}{cc\|c} Q & Q^+ & E \\ \hline 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{array}$ ($C = \uparrow$)

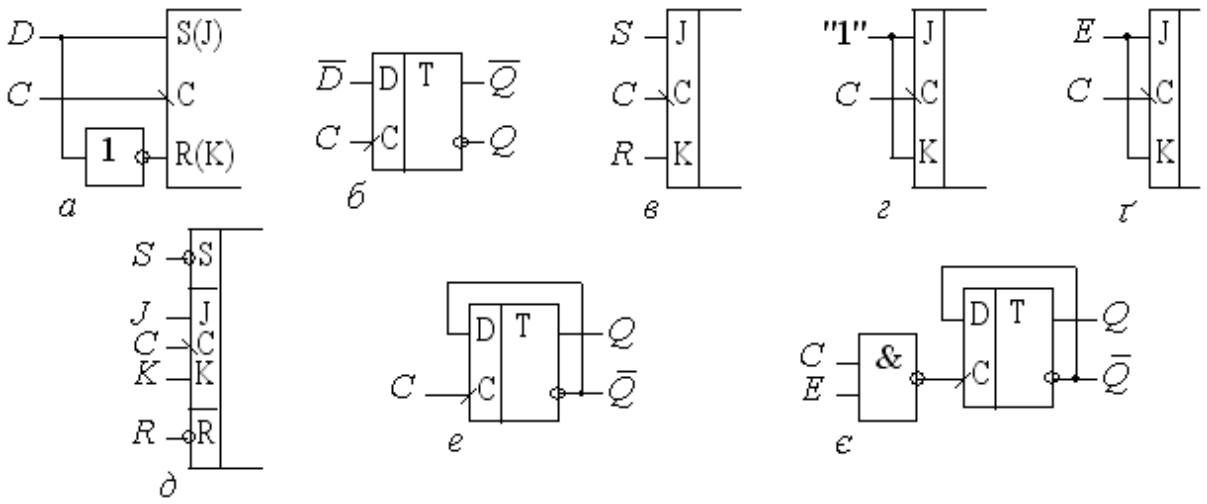


Рисунок 7.1

КОНТРОЛЬНІ ЗАПИТАННЯ

1. 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.

2. Побудуйте граф перемикань та за допомогою позицій станів на схемі і часових діаграм схарактеризуйте 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: 3) RSC – І-НЕ, 4) RSC – АБО-НЕ, 5) $\overline{R}\,\overline{S}\,\overline{C}$ –І-АБО-НЕ, 6) RSC – І-АБО-НЕ, 7) DC – АБО-НЕ, 8) D$\overline{C}$ – І-АБО-НЕ та НЕ, 9) DC –І-АБО-НЕ, 10) DC – І-НЕ, 11) D$\overline{C}$ – АБО-НЕ, 12) $\overline{D}\,\overline{C}$ – АБО-НЕ;

в) за схемою MS: 13) RSC - з інвертором, 14) JKC - із забороненими зв’язками, 15) T - з різнополярним керуванням на синхронних RS-тригерах, 16) D - з різнополярним керуванням на однофазних D-тригерах;

г) за схемою трьох тригерів: 17) DC, 18) JK, 19) T, 20) TE;

д) з імпульсно-потенціальним керуванням: 21) JKC, 22) DC, 23) TE.

ЛАБОРАТОРНЕ ЗАВДАННЯ

1 Дослідити основні типи тригерів.

1.1 Дослідити ***асинхронний RS-тригер*** з інверсними входами (/RS-тригер) на елементах І-НЕ: за принциповою електричною схемою та осцилограмами сигналів (файли 7asyn.bdf, .vwf) 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.

1.2 Дослідити ***синхронний D-тригер зі статичним керуванням*** (примітив Latch – тригер-за́скочка) на елементах І-НЕ за п. 1.1 (файли 7stat.bdf, .vwf) та розглянути особливості побудови і перемикання макрофункції D-тригера та схеми синхронного RS-тригера (RSC-тригер).

1.3 Дослідити ***синхронний JK-тригер з динамічним керуванням*** структури MS (схема 1) за п. 1.1 (файли 7dyn.bdf, .vwf); розглянути особливості побудови і перемикання базових примітивів JKFF, TFF і DFF (2, 3, 4), T-тригера на основі D-тригера (5), макрофункції D-тригера expdff (6) та T-тригера на основі типової макрофункції JK-тригера (7).

1.4 Ознайомитися з ***різновидами тригерів*** бібліотеки бази даних (файл 7libr.bdf): примітивами та макрофункціями (у тому числі вибраними ІС серії 74). У звіті навести також умовне графічне позначення за ДСТУ таких тригерів, пояснити призначення та особливості входів і виходів.

2 Розглянути приклади застосування тригерів.

2.1 Дослідити вимірювальний перетворювач на основі RS-тригера і детекторів фронтів (файли 7vymir.bdf, .vwf). У звіті навести схему, часові діаграми та пояснити спосіб перетворення різниці фаз між вхідними сигналами в пачку імпульсів квантування.

2.2 Дослідити антидеренчливий пристрій на основі RS-тригера (файли 7ander.bdf, .vwf). У звіті навести схему, часові діаграми та пояснити спосіб усунення деренчання фронтів вихідного сигналу Q2 внаслідок тремтіння контактів під час перемикання механічних перемикачів (на виході Q3 спостерігається деренчання фронтів).

2.3 Дослідити селектор імпульсів на основі тригерів з динамічним керуванням (файли 7select.bdf, .vwf). У звіті навести схему, часові діаграми та пояснити спосіб виокремлення одиничного синхроімпульсу, першого з їх послідовності після натиснення нефіксованого ключа.

2.4 Дослідити імпульсний фільтр на асинхронних тригерах і типовій макрофункції серії 74 (файли 7if.bdf, .vwf). У звіті навести схему, часові діаграми та пояснити спосіб виокремлення інформаційного імпульсу x від коротких імпульсних завад за допомогою двох послідовностей імпульсів a, b тривалістю не нижче тривалості імпульсів завад.

3 Створити згідно із заданим варіантом на основі DE-тригерів фіксатор у паралельному коді вхідного слова, біти якого надходять до входів тригерів послідовно в часі.

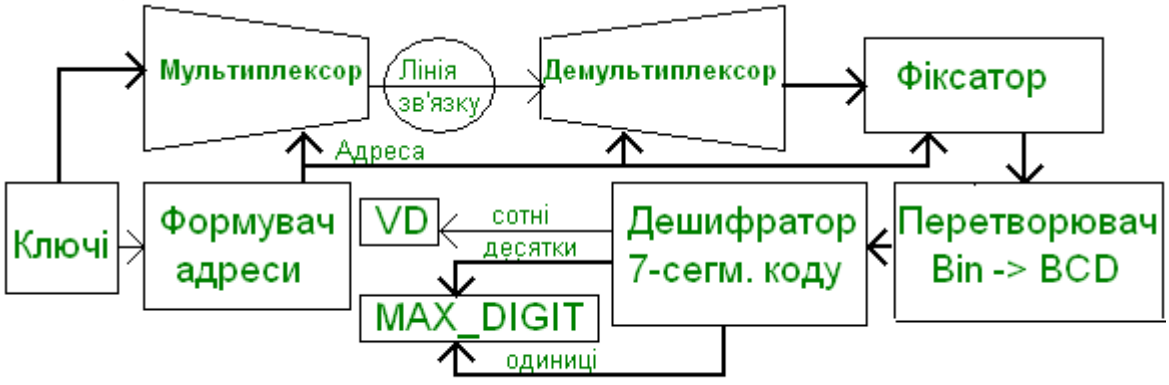
3.1 За допомогою програмного майстра New Project Wizard створити новий проект 7XXfixator на IC родини MAX7000S (тип та інші параметри IC можна не визначати), зібрати в графічному файлі фіксатор потрібної розрядності на примітивах DE-тригерів (директорія c:/quartus/libraries/primitives/storage/dffe), створити і вставити до файла зразок мегафункції дешифратора адреси, виходи якого з'єднати зі входами дозволу тригерів, скомпілювати проект, створити файл часових діаграм, виконати функціональне моделювання та переконатися в правильності роботи пристрою.

Приклад: 700fixator.bdf, .vwf.

3.2 Командою з меню File > Create / Update > Create Symbol Files for Current File створити символьний файл та відкрити його для перегляду (з типу файлів Graphic Files або Other Source Files).

Приклад: 700fixator.bsf.

4 Створити ієрархічний проект лабораторного макету на рівні блок-схеми для експериментального дослідження лінії передачі даних (ЛПД) з фіксацією вихідного коду на тригерах згідно з варіантом завдання 7 (див. структурну схему і як взірець файл 700LPD.bdf – рис. Д5 у додатках).



Структурна схема

Від ДІП-перемикачів MAX_SW1 вхідне слово sw1[] у паралельному коді надходить на інформаційні входи мультиплексора. Формувач адреси (лічильник) розгортає адресний код a[] в часі таким чином, що після кожного натискання по черзі двох нефіксованих перемикачів MAX_PB1 і MAX_PB2 цей код збільшується на одиницю по колу: 0, 1, ... 7, 0, 1, ..., тобто змінюється в прикладі за модулем 8. Отже, у мультиплексорі відбувається перетворення паралельного коду в послідовний, бо на його виході по черзі з'являються біти вхідного слова (для передачі інформації лінією зв'язку між віддаленими об'єктами). З інформаційного входу демультиплексора (який є стробовим входом дешифратора) на вихідному боці лінії тим же самим адресним кодом біти вхідного слова по черзі спрямовуються

до інформаційних входів тригерів у блоці фіксатора. Послідовно в часі, біт за бітом слово записується в розрядні тригери синхроімпульсами Clk, що надходять з формувача адреси.

У такий спосіб послідовний код перетворюється в паралельний, отже, після n тактів (n – кількість розрядів коду) на виходах фіксатора вхідне слово sw1[] зчитується в паралельному коді. У разі потреби, тригери можна обнулити натисненням і відтисненням ключа на вході очищення CLRN. Для індикації слова sw1[] в десятковій системі його двійковий код попередньо перетворюється в тетради ДДК за допомогою перетворювача двійкового коду у двійково-десятковий. Кожна тетрада надходить на свій дешифратор 7-сегментного коду і подається на відповідне знакомісце 7-сегмент-ного індикатора MAX_DIGIT. Якщо значення коду перевищує 99, для індикації потребується більше двох знакомісць (за кількістю тетрад ДДК), але в прикладі наявність сотні індикується світлодіодом D1 (кількість сотень більше однієї можна індикувати у двійковій системі кількома світлодіодами).

4.1 Визначити ім’я проекту 7XXLPD, вибрати родину мікросхем MAX7000S та тип EPM7128SL84-7.

4.2 Ввести до проекту мегафункції.

а) Створити графічний файл для мультиплексора, дати йому ім’я, наприклад, mx із включенням до проекту (у вікні збереження підняти прапорець Add file to current project). За допомогою програмного майстра MegaWizard Plug-In Manager створити потрібний різновид мегафункції, наприклад, mux7_1 і вставити його до файла, ввести вхідні і вихідний порти, об’єднати входи до шини, дати портам імена та зберегти зміни у файлі із включенням його до проекту.

б) Командою з меню File > Create / Update > Create Symbol Files for Current File створити символьний файл та відкрити його для перегляду (з типу файлів Graphic Files або Other Source Files).

***Приклад**:* mx.bdf, .bsf, mux7_1.tdf.

в) Аналогічним чином створити і ввести до складу проекту потрібний різновид мегафункції демультиплексора.

***Приклад**:* dmx.bdf, .bsf, dmx1_7.tdf.

4.3 Створити файл блок-схеми і вставити до нього у вигляді блоків символи мультиплексора і демультиплексора за п.4.2, фіксатора 7XXfixator.bsf за п. 3, формувача адреси з файла D:/quartus6.1work/9lab/9addr.bdf (у разі необхідності змінити розрядність адресного коду слід у файлі-зразку 700LPD клацнути символ 9addr кнопкою B2 > Open Design File, перенести його схему до свого нового файла, внести зміни і зберегти із включенням до проекту та створити символ), а також перетворювача двійкового коду у двійково-десятковий (у вікні Symbol натиснути кнопку огляду та вибрати символ ../3lab/Bin8_BCD). Відтак вставити у вигляді блоків символи макрофункцій дешифратора 7-сегментного коду (до нового графічного файла вставити макрофункцію та вхідний і ви-

хідний порти, згорнути сигнали до шин, зберегти файл із включенням до проекту та створити символ), а також ввести вхідні і вихідні порти та, у разі потреби, додаткові елементи.

4.4 З'єднати компоненти блок-схеми і розвести сигнали між блоками (див. Лаб. роботу №6, п. 3.6, 3.7).

4.5 Імпортувати призначення виводів мікросхеми (див. Лаб. роботу №4, п. 5.2) з файлу ../4lab/MAX_pin та скоригувати їх.

***Приклад**:* 700LPD.bdf.

4.6 Переконатися, що в проекті блок-схеми описано всі його складники (інакше відсутні додати, а зайві вилучити), виконати компіляцію проекту і налагодити його в разі виявлення помилок, переглянути застереження у вікні повідомлень та врахувати або знехтувати їх залежно від наслідків.

4.7 Виконати функціональне моделювання та переконатися в правильності проектування пристрою.

***Приклад**:* 700LPD.vwf.

4.8 Відкрити вікно програматора, визначити режим програмування, зберегти CDF-файл та виконати фізичне програмування мікросхеми.

4.9 Розробити методику і виконати необхідні експериментальні дослідження. Порівняти результати досліджень з даними проектних файлів, зробити висновки.

8 ЛАБОРАТОРНА РОБОТА №8. 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 8 (див. Додатки).

СТИСЛІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Означення і класифікацію регістрів, принцип побудови та схемну реалізацію паралельних регістрів і регістрів зсуву розглянуто в [2], основні їх типи зведено в табл. 8.1, 8.2 (у кінці розділу). Тут наведемо методику проектування ЦПП на основі регістрів зсуву.

1 Генератори цифрових кодів і розподільники

Крім зберігання інформації і виконання арифметичних операцій регістри 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 розподілу 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 здобули 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 (лог. 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 комбіновані, бо РІ утворюють з РР.

Згадані пристрої циклічної дії, що ґрунтуються на кодах зі сталим сполученням нулів і одиниць, можна побудувати на тригерах або 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 праворуч 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 $N = Q_2Q_1Q_0$ = (000111) = 0, 1, 3, 7, 6, 4, які є різні, 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.

Замість попередньо записувати слово до кільцевого регістра, в ГКП і РІР формують послідовність символів запровадженням зворотного зв'язку між розрядами регістра. В ІС регістра зсуву приступним є лише вхід послідовного введення, тому зв'язок здійснюють між цим входом і виходами регістра за допомогою комбінаційних кіл.

2 Проектування ГКП

Вважатимемо заданою послідовність символів, яку легко визначити, виходячи з міркувань потрібних модуля, послідовності вихідних кодів, часових діаграм тощо. Для прикладу розглядатимемо послідовність символів з періодом (000111). Проектування можна виконати в такому порядку.

1) Визначаємо модуль M = 6, що дорівнює кількості символів у послідовності, та мінімальну кількість розрядів ГКП, яка становить n = 3, бо при n = 2, як зазначено вище, деякі з вихідних кодів будуть однаковими.

Для наочності будуємо робочий цикл перемикального графу ГКП. Під час зсуву числа з послідовності символів на один крок у бік старших розрядів значення початкового коду $N = Q_2Q_1Q_0$ подвоюється (наприклад, 001 $\rightarrow$ 010) і до молодшого розряду додається ще наступний символ x_N послідовності, тобто з надходженням чергового синхроімпульсу матимемо код $N^+ = 2N + x_N$. Але якщо є одиниця в старшому розряді, у наступному такті зсуву вона зникає, тому її вагу 2^{n-1} перед подвоєнням треба відняти. Отже, наступний код можна визначити з виразу:

$$N^+ = 2N + x_N, \text{ якщо } N < 2^{n-1}; \; N^+ = 2(N - 2^{n-1}) + x_N, \text{ якщо } N \geq 2^{n-1}. \quad (8.1)$$

Так, у прикладі при n = 3 перший код утворюється трьома нулями послідовності символів (000111), тобто $N = Q_2Q_1Q_0 = 0$. Другий код N^+=2·0+1=1, бо $N < 4$ і наступний символ послідовності $x_N = 1$; так само третій код становить $N^+ = 2\cdot1 + 1 = 3$ і т.д. Після останнього символу з періоду послідовності переходимо до його початку. У підсумку визначаємо робочий цикл $N = Q_2Q_1Q_0$ = (000111) = 0, 1, 3, 7, 6, 4, який відображаємо перемикальним графом (рис. 8.1,а).

✐ *Примітка:* Робочий цикл можна визначити і безпосередньо з по-

слідовності символів 000111000111... просуванням по ній трьох розрядів, але це не зручно для великих модулів і, як буде видно з подальшого, для побудови повного графу.

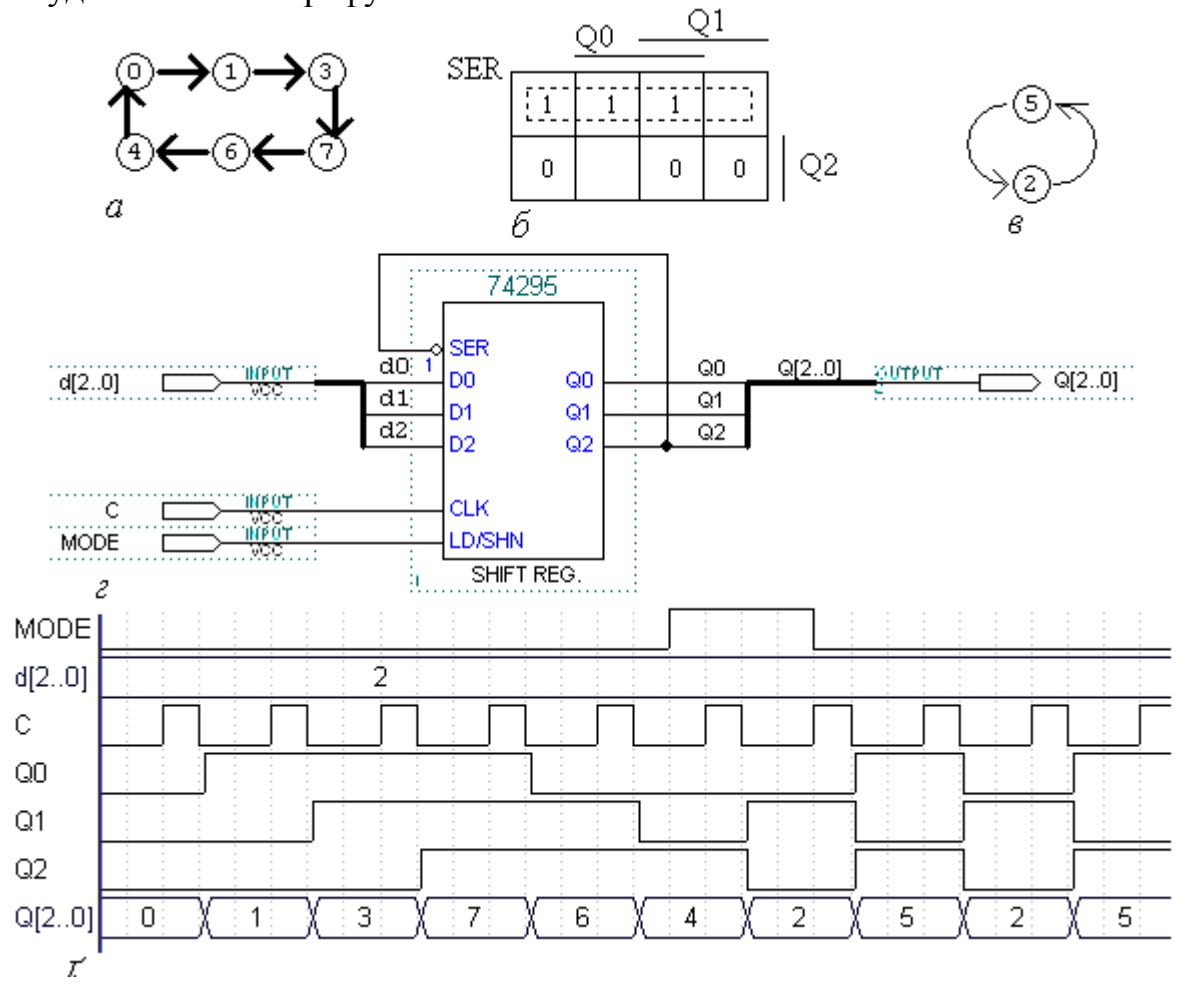


Рисунок 8.1 – Проектування ГКП

2) Мінімізуємо функцію збудження по входу послідовного введення *SER*, який для регістра прямого зсуву є входом молодшого розряду, отже, для регістра на D-тригерах функція має вигляд $SER = Q_0^+$. Тому діаграму термів (Вайча-Карно) можна побудувати і без таблиці відповідності (за багатьох розрядів така таблиця стає неозорою). Виходимо з того, що парні коди $N = Q_2Q_1Q_0$ закінчуються на 0, а непарні – на 1, тому в клітинку діаграми з номером N вписуємо значення $Q_0^+ = 0$, якщо наступний код робочого циклу є парний, і $Q_0^+ = 1$, якщо непарний. Так, на діаграмі рис. 8.1,б номери клітинок відповідають станам $N = Q_2Q_1Q_0$, що є вершинами робочого циклу графу, тому до нульової клітинки $N = 0$ вписуємо 1, бо наступний код $N^+ = 1$ є непарний; переходимо до вершини $N = 1$ і вписуємо до клітинки з цим номером також одиницю, бо наступний код $N^+ = 3$ теж є непарний і т. д. У підсумку заповнюємо шість клітинок відповідно до робочого циклу, а в двох порожніх клітинках функція має факультативні значення, бо у станах, що їм відповідають, за нормальної роботи пристрій не перебуває. Мінімізуючи звичайним чином, отримуємо функцію збудження

$$SER = \overline{Q_2}\,. \qquad (8.2)$$

3) Перевіряємо функціонування пристрою, якщо випадково (внаслідок збою, дії завади або по ввімкненні джерела живлення) він опиниться в хибному стані поза межами робочого циклу. Пристрій, що автоматично за кілька тактів повертається до робочого циклу, є *самовідновний*, інакше – несамовідновний. З метою перевірки на самовідновність будуємо повний перемикальний граф. Для цього беремо будь-який стан поза робочим циклом і за (1) визначаємо наступний стан, але значення x_N підставляємо тепер з діаграми термів: для тих кодів, які згідно з виконаною мінімізацією входять до об'єднань одиниць, $x_N = 1$, а для всіх інших $x_N = 0$. Так, зі стану N=2 з надходженням синхроімпульсу ГКП перейде до стану N^+=2·2+1=5, бо $N< 4$ і $x_N = 1$, а зі стану $N = 5$ повернеться до N^+=2(5–4)+0=2, тобто замкнеться хибний цикл (рис. 1,в) з модулем 2, отже, ГКП є несамовідновний.

Такий ГКП, реалізований згідно з (2) на регістрі зсуву з перехресним зворотним зв'язком та інвертуванням біта старшого розряду, який називається також лічильником Джонсона (рис. 8.1,г), функціонує за часовими діаграмами на рис. 1,г. З метою випробування регістр керується сигналом *MODE = LD/SHN* (завантаження/зсув): при *MODE* = 0 він функціонує як регістр зсуву, а при *MODE* = 1 з надходженням синхроімпульсу до нього завантажується в паралельному коді слово зі входів *d*[2..0], яке імітує збій з переходом до хибного стану. З діаграм видно, що в кожному розряді відповідно до тактів синхроімпульсів за нормальної роботи пристрою просувається послідовність символів по три нулі і три одиниці поспіль (отже, безпосередньо з діаграми легко записати період послідовності символів). Зі зсувом у розрядах ці послідовності утворюють вихідні коди згідно з робочим циклом графу, але після збою послідовність порушується і пристрій не в змозі самостійно вийти з хибного циклу.

4) Відновити нормальну роботу пристрою можна двома шляхами: а) коли регістр потраплятиме до хибного циклу, примусово встановити один зі станів робочого циклу (у прикладі можна обнулити) або б) скоригувати функцію збудження таким чином, щоб хибні цикли не утворювалися.

У ГКП застосовують другий шлях, тому з метою надати властивість самовідновлення пристрою коригуємо функцію збудження. Для цього без зміни робочого циклу необхідно розірвати хибний цикл (або всі хибні цикли, якщо їх декілька) зміною, принаймні, одного з переходів останнього. Якщо перевизначити п'яту клітинку одиницею (помічена зірочкою на рис. 8.2,а), дістанемо нову функцію збудження

$$SER = Q_0\overline{Q_1} + \overline{Q_2} = \overline{\overline{Q_0}\,\overline{\overline{Q_1}}\,Q_2}\,. \qquad (8.3)$$

За наявності інших варіантів розглядаємо їх і вибираємо функцію меншої складності (у прикладі варіант довизначення другої клітинки нулем є рівноцінний, але в мішаному елементному базисі).

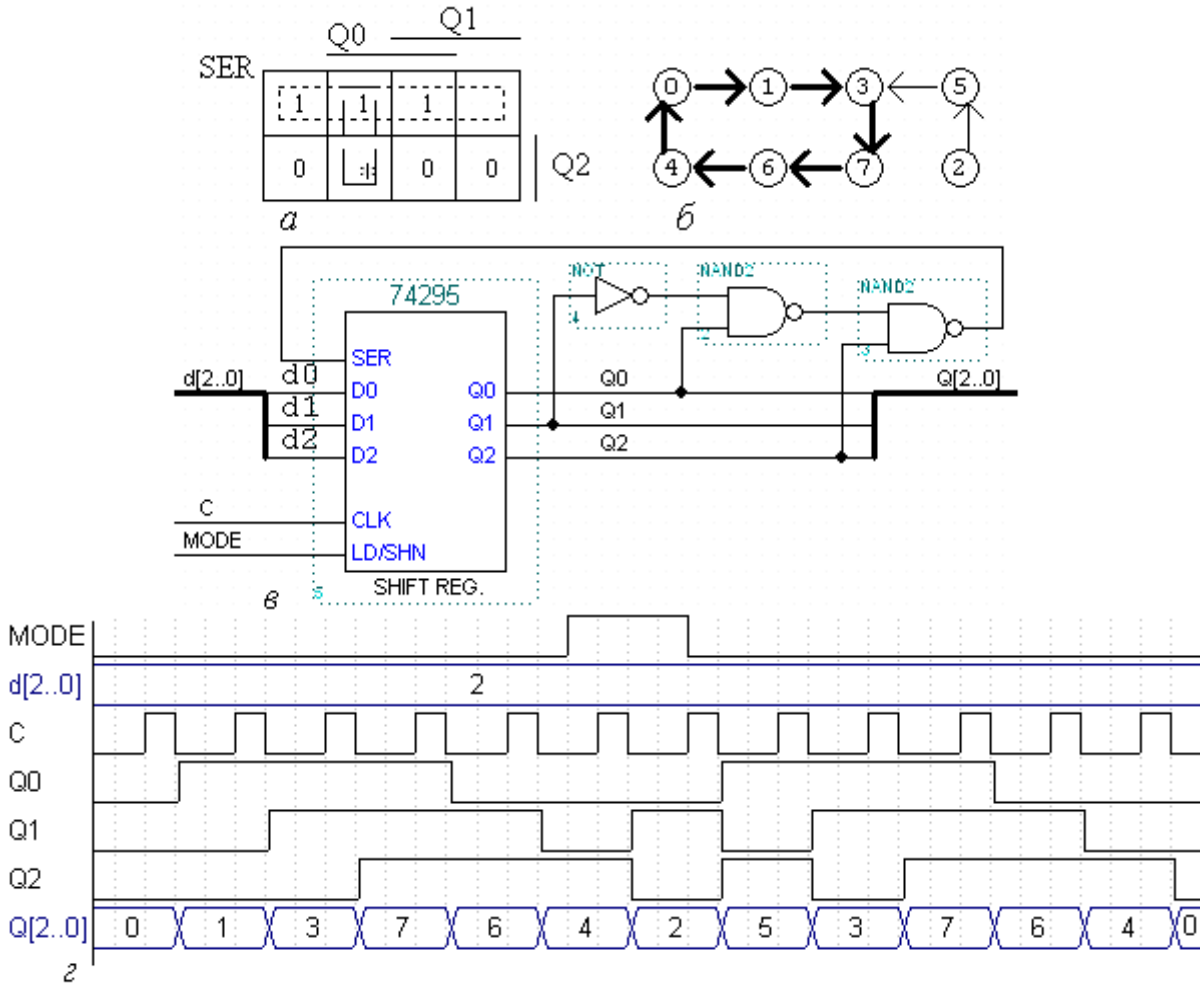


Рисунок 8.2– Проектування ГКП (розірвання хибного циклу)

Враховуючи нові значення символів x_N після останньої мінімізації, будуємо так само скоригований повний перемикальний граф (рис. 8.2,б) і переконуємось, що при новій функції збудження ГКП є самовідновний, інакше добираємо інші варіанти перевизначення факультативних клітинок. Згідно з (3) коригуємо схему ГКП (рис. 8.2,в) і так само випробовуємо її шляхом переведення сигналом *MODE* до хибного стану (рис. 8.2,г). Як бачимо, за два такти ГКП автоматично повертається до робочого циклу. У випадку необхідності скоротити самовідновлення до одного такту потрібно інакше довизначити факультативні значення функції (тут: всі клітинки).

3 Проектування розподільників

ГКП, в яких період послідовності символів містить лише одну одиницю, і є найпростішим типом розподільників рівнів. Записана до одного розряду одиниця циркулюватиме в кільцевому регістрі, утворюючи в кожному такті активний рівень по черзі в каналах розподільника. Тому й методика проектування РР аналогічна методиці проектування ГКП.

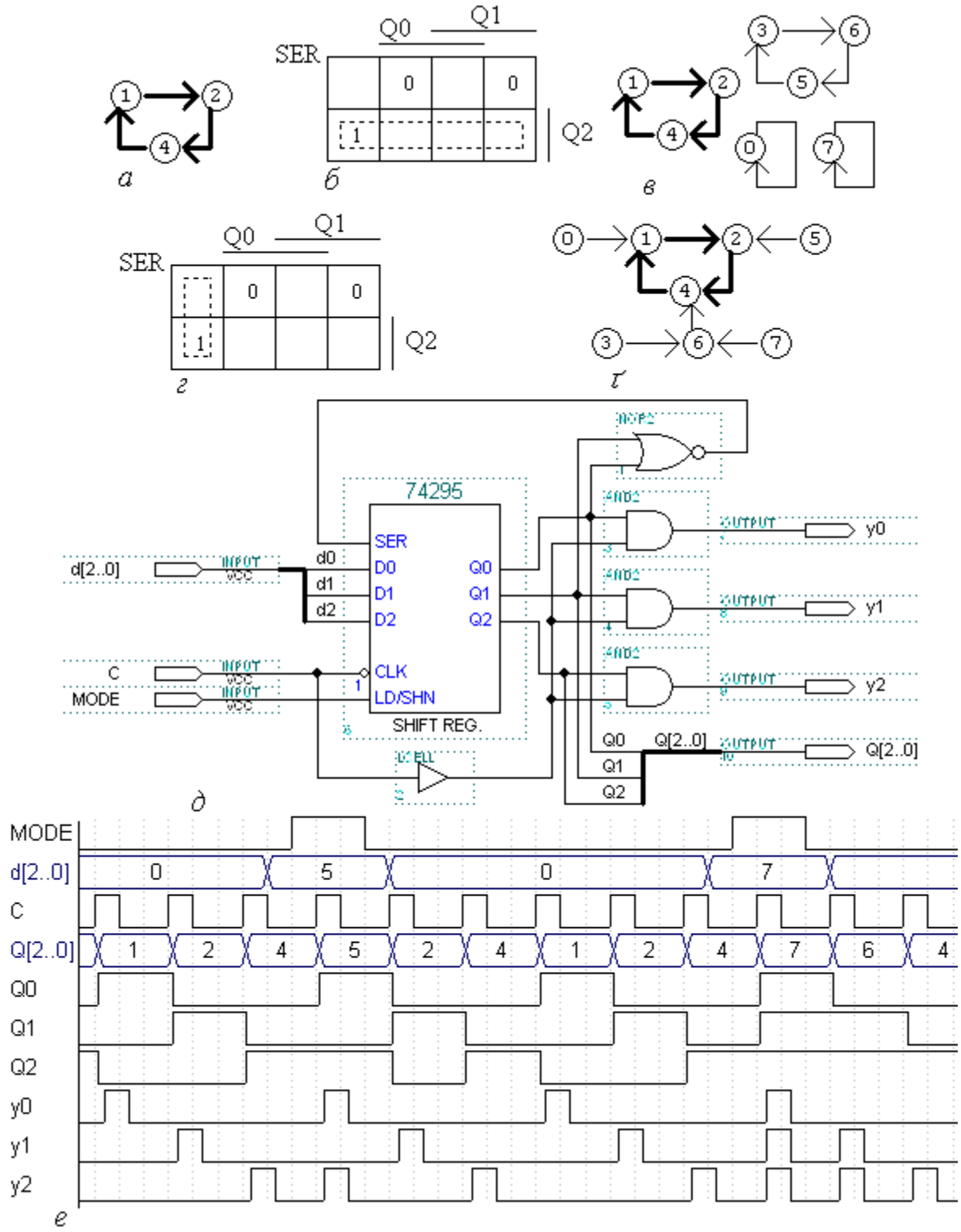


Рисунок 8.3 – Проектування РР

1) Задана кількість каналів РР *n* визначає і кількість розрядів регістра *n* та період послідовності з *n* символів, серед яких є одна одиниця, а решта нулі. Наприклад, для триканального РР період становитиме (001). Якщо в каналах є паузи між тактами або, навпаки, активний рівень є в кількох тактах поспіль, період визначаємо за потрібними часовими діаграмами так само, як і в ГКП. Відтак будуємо робочий цикл перемикального графу за виразом (1) або безпосередньо з послідовності символів (рис. 8.3,а).

2) Користуючись цим графом, будуємо діаграму термів (рис. 8.3,б) і мінімізуємо функцію збудження для входу послідовного введення $SER = Q_2$.

3) З метою перевірки на самовідновність будуємо повний перемикальний граф (рис. 8.3,в), який у прикладі крім робочого містить три хибні цикли: один за модулем 3, але з неправильним розподілом імпульсів, та два стани (0 і 7), з яких розподільник не зможе вийти самостійно у випадку потрапляння до них. Отже, пристрій є несамовідновний.

4) Коригуємо функцію збудження (рис. 8.3,г) з метою надати властивості самовідновлення пристрою:

$$SER = \overline{Q_0}\,\overline{Q_1} = \overline{Q_0 + Q_1}, \qquad (8.4)$$

будуємо скоригований повний перемикальний граф (рис. 8.3,ґ), переконуємось, що при новій функції збудження пристрій є самовідновний та згідно з (4) складаємо схему РР (на рис. 8.3,д частина без урахування виходів y_i).

У кожному такті активний рівень лог.1 (у випадку інверсних сигналів – лог. 0) діє тільки на одному з виходів Q_i (на рис. 8.3,е інтервал до появи сигналу *MODE* = 1), тому пропусканням синхроімпульсів через елементи збігу І легко утворити *розподільник імпульсів* (на рис. 8.3,д,е частина по виходах y_i). Аби синхроімпульси не потрапляли на краї сигналів Q_i, їх можна затримати за допомогою додаткових логічних елементів (у прикладі – буфер LCELL).

Штучним переведенням розподільника до хибного стану сигналом *MODE* (див. рис. 8.3,е) шляхом паралельного завантаження регістра переконуємось, що за один або два такти після збою пристрій автоматично повертається до робочого циклу – так само, як на повному перемикальному графі.

4 Перетворення ГКП у розподільники

Перевагою розглянутого типу розподільників є простота і висока швидкодія, а недоліком – велика розрядність регістра і, крім того, для розподільників імпульсів ще велика потрібна кількість двовходових елементів І, що може виявитися неприйнятним для побудови багатоканальних РІ та РР. Послабити цей недолік можна шляхом перетворення у розподільники ГКП з модулем, удвічі більшим за розрядність регістра. З огляду на те, що всі вихідні коди ГКП є різні, шляхом їх дешифрування дістанемо кількість каналів розподільника, яка дорівнює модулю ГКП. Розглянемо порядок зазначеного перетворення.

1) Вибираємо модуль M ГКП, що відповідає потрібній кількості каналів розподільника, отже, і кількості символів у періоді їх послідовності та за методикою п. 2 проектуємо ГКП. Наприклад, для шестиканального РР задаємося періодом (000111) і одержуємо ГКП як у п. 2 (відповідає схемі і часовим діаграмам відносно виходів Q_i на рис. 8.4,а,б).

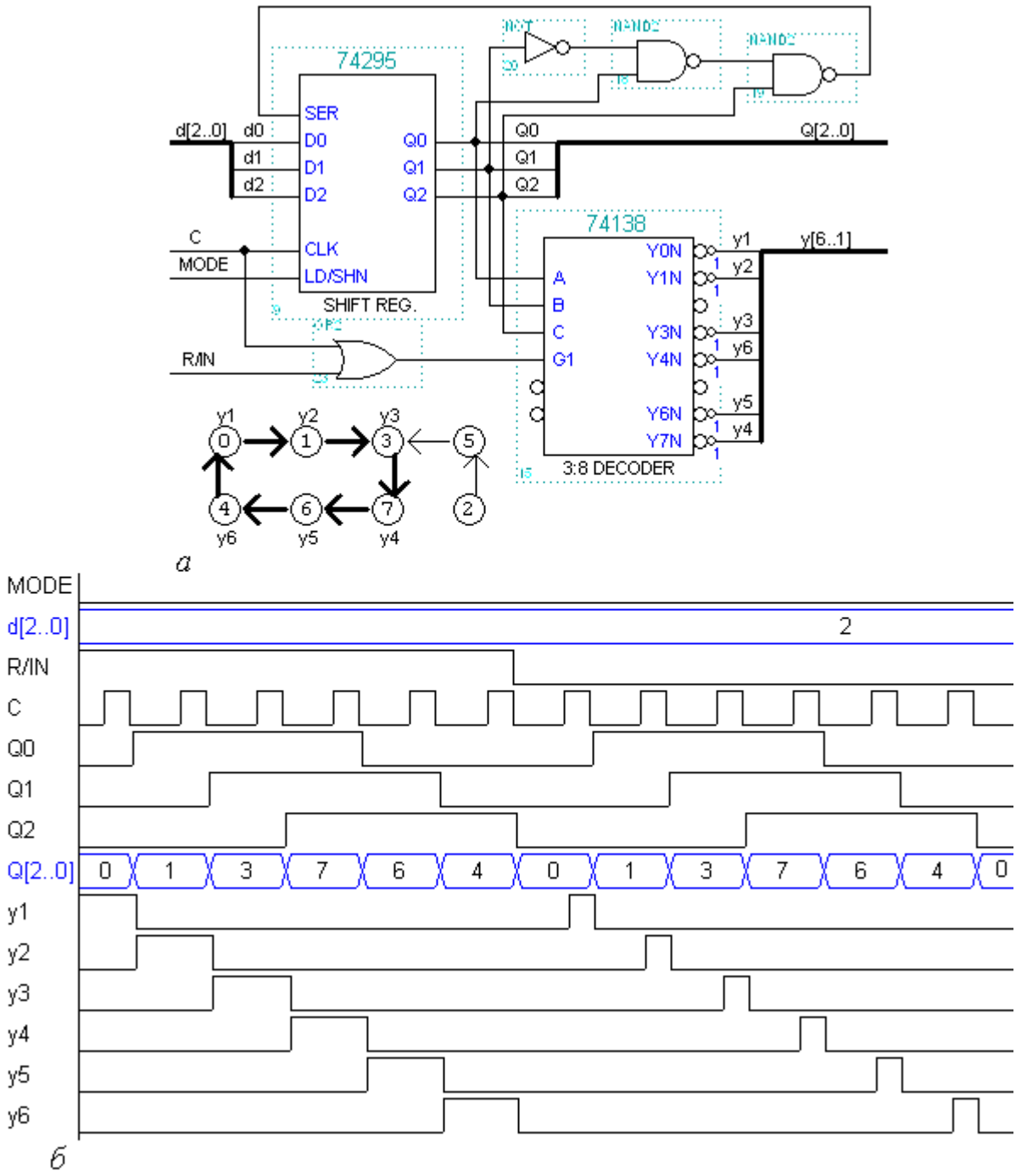


Рисунок 8.4 – Перетворення ГКП у розподільник

2) На виходах ГКП утворюється двійковий код $Q[2..0]$, а РР має забезпечити рівень лог. 1 протягом такту тільки в одному каналі, що відповідає унітарному кодові на його виходах. Отже, завдання полягає в перетворенні двійкового коду в унітарний. Таку операцію, як відомо, здійснює дешифратор. Відповідно до розрядності вихідного коду ГКП $n = 3$ вибираємо двійковий дешифратор 3:8 (див. рис. 8.4,а без урахування стробового входу $G1$).

3) Визначаємо канали РР згідно з черговістю з’явлення в них активного рівня. Повний дешифратор 3:8 має вісім виходів, кожний з яких активізується відповідно до вхідного адресного коду, відображеного робо-

чим циклом перемикального графу (див. рис. 8.4,а). Тому два виходи з номерами хибних станів (2 і 5) не використовуються, а інші шість каналів y_1... y_6 нумеруємо в порядку активізації їх у часі згідно з переходами графу. У підсумку отримуємо РР (схема на рис. 4,а по виходах y_1... y_6), який функціонує за часовими діаграмами на інтервалі $R/IN = 1$ на рис. 8.4,б: на виходах y_1... y_6 по черзі протягом такту діє рівень лог. 1.

4) З'єднанням каналів РР y_1... y_6 з елементами І та пропусканням через них синхроімпульсів можна отримати РІ так само, як на рис. 8.3,д. Проте доцільно скористатися стробованим дешифратором, який виконує функцію демультиплексора, якщо на стробовий вхід ($G1$ на рис. 8.4,а) подати синхроімпульси. Комбінований розподільник утворюється з додатковим елементом АБО: керувальним сигналом $G_1 = R/IN = 1$ дешифратор стає перетворювачем до унітарного коду і пристрій функціонує як РР, а за рівня $R/IN = 0$ елемент АБО пропускає синхроімпульси на вхід G_1 і демультиплексор комутує їх до каналів y_1... y_6, тому пристрій функціонує як РІ (див. рис. 8.4,б). Як і ГКП, перевірити розподільник на самовідновність можна за допомогою сигналу *MODE*.

5 Особливості побудови реверсивних ГКП і розподільників

У випадку необхідності змінювати напрямок надходження кодів з виходів ГКП та сигналів у каналах розподільників природним є застосування реверсивних регістрів в основі побудови таких пристроїв. Розгляд почнемо з ГКП, бо на них ґрунтуються й розподільники та для наочності й стислості зупинимося на тому самому прикладі послідовності символів з періодом (000111), що й у п. 2. Основа проектування реверсивного ГКП така сама, як і односпрямованого, тому достатньо відзначити лише її особливості.

1) Вважаючи ГКП односпрямованим, розрядність регістра $n = 3$ визначаємо так само і за відсутністю трирозрядних вибираємо чотирирозрядний реверсивний регістр (символ SHIFT REG. на рис. 8.5,а). Керувальним кодом $S[1..0] = 1$ утворюється регістр прямого зсуву (праворуч) зі входом послідовного введення SRSI, а кодом $S[1..0] = 2$ – у регістр зворотного зсуву (ліворуч) зі входом послідовного введення SLSI.

2) Визначаємо виходи регістра, на яких формується потрібна кодова послідовність. Виходимо з того, що для побудови односпрямованого ГКП на кільцевому регістрі з прямим зсувом необхідно, як і раніше, зворотний зв'язок здійснювати через вхід молодшого розряду *SRSI*, а виходами пристрою є виходи трьох молодших розрядів $Q[2..0] = (QC, QB, QA)$. Для зворотного зсуву доступ є тільки до входу старшого розряду *SLSI* з виходом *QD*, отже, у трирозрядному регістрі необхідно використовувати виходи $Q[2..0] = (QD, QC, QB)$.

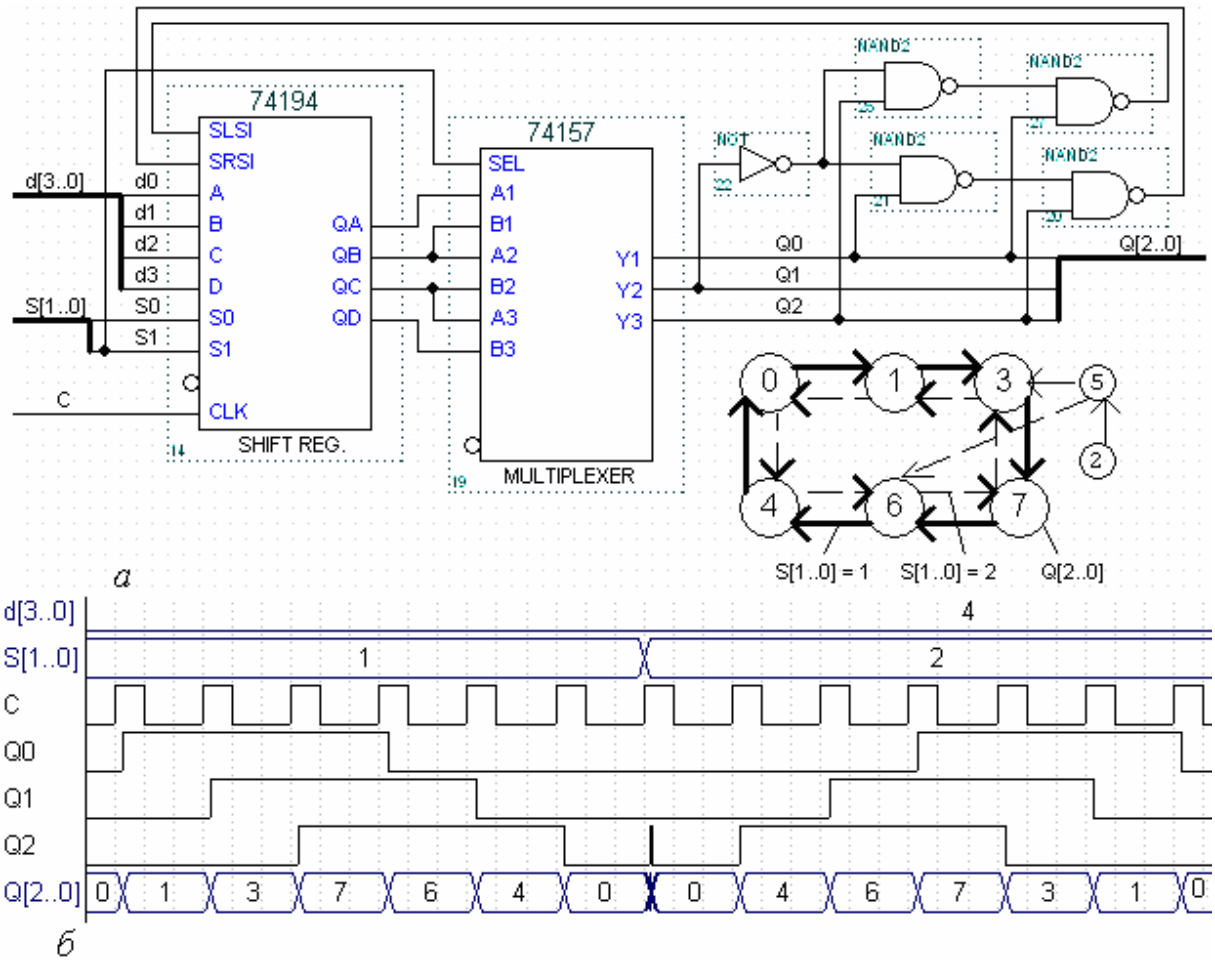


Рисунок 8.5 – До особливостей побудови реверсивних ГКП і розподільників

3) Формуємо вихідну шину ГКП $Q[2..0]$, з якої знімається змінювана за напрямком кодова послідовність. Аби спрямувати пряму та зворотну послідовності до цієї шини, потрібно перемикати виходи регістра, що можна здійснити логічними елементами, але найдоцільніше скористатися мультиплексорами (в одному корпусі стандартних ІС міститься по чотири мультиплексори 2:1 зі спільним адресним входом *SEL*). У прикладі за адресний вхід може правити розряд S_1, який перемикає до виходів Y_i під час прямого зсуву рівнем $S_1 = 1$ входи A_i, а під час зворотного зсуву рівнем $S_1 = 0$ – входи B_i. Після зазначених з'єднань входів мультиплексора схему (див. рис. 8.5,а) можна уявляти еквівалентним трирозрядним регістром відносно виходів $Q[2..0]$ та входів послідовного введення. Внаслідок цього для утворення реверсивного кільцевого регістра достатньо з'єднати $SRSI = Q_2$ та $SLSI = Q_0$.

4) Проектуємо односпрямований самовідновний ГКП за п. 2 і отримуємо схему з коригувальним ЦКП у колі зворотного зв'язку (див. рис. 8.2,в), який переносимо на нашу схему (елемент НЕ та два нижні елементи І-НЕ). Цілком зрозуміло, що проектувати такий самий ГКП зворотного напрямку немає сенсу, адже якщо перейменувати розряди, регістр зворотного зсуву перетворюється на регістр прямого зсуву. У нашому ви-

падку достатньо поміняти місцями розряди Q_2 і Q_0 для коригувального ЦКП у регістрі зворотного зсуву (елемент НЕ і два верхні елементи І-НЕ). Природно, у регістрах більшої розрядності слід змінювати на протилежну всю нумерацію від старшого розряду до молодшого.

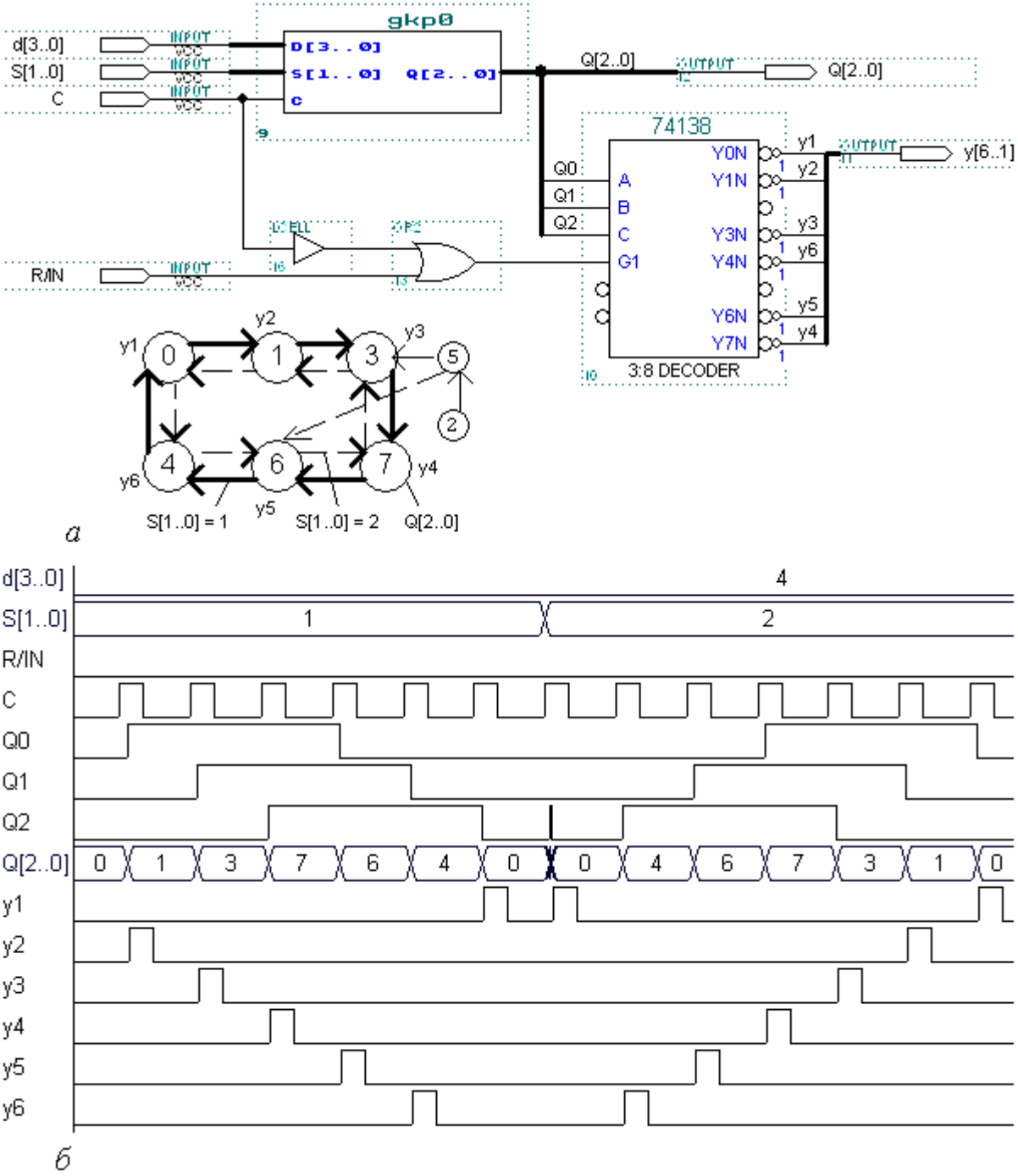


Рисунок 8.6

5) Через це робочі цикли перемикального графу (на рис. 8.5,а) при прямому (суцільні лінії) та зворотному (пунктирні лінії) зсуві перетворюються один в одного, якщо коди станів в одному графі читати в прямому напрямку, а в іншому – у зворотному. Наприклад код $1_{10} = 001_2$ при читанні у зворотному напрямку перетворюється на $4_{10} = 100_2$. Взаємозворотними є і коди 3 та 6, тому перехід від хибних станів 2, 5 (через си-

метричність читаються однаково в обох напрямках) відбувається в одному графі до стану 3, а в іншому – до стану 6 (зображено суцільною та пунктирною лініями).

6) Перемикання напрямку кодової послідовності сигналом $S[1..0]$ наведено на часових діаграмах рис. 8.5,б. Для випробування ГКП на самовідновність сигналом $S[1..0] = 3$ у паралельному коді до регістра записується відповідно до найгіршого хибного стану число $2_{10} = 010_2$, яке при прямому зсуві слід подати до розрядів *C, B, A*, а при зворотному зсуві – до розрядів *D, C, B*. Відносно групи паралельного введення в останньому випадку воно становитиме $d[3..0] = (D, C, B, A) = 0100_2 = 4_{10}$ (цей код відображено на епюрі).

7) Перетворення реверсивних ГКП у розподільники можна виконати так само, як і односпрямованих ГКП (див. п. 4). Якщо згорнути схему реверсивного ГКП (див. рис. 8.5,а) до символу (gkp0 на рис. 8.6,а), то утворюється типова структура такого перетворення. При керувальному сигналі *R/IN* = 1 пристрій функціонує як реверсивний РР, а при *R/IN* = 0 – як реверсивний РІ. Часові діаграми (рис. 8.6,б) наведено для режиму РІ.

8) Проектування простих реверсивних розподільників на основі ГКП з періодом послідовності символів, серед яких є лише одна одиниця, здійснюється за п. 3 з урахуванням особливостей, зазначених у п. 5.1 ... 5.6. Приклад односпрямованого розподільника, розглянутий у п. 3, для реверсивного РІР ілюструється на рис. 7(самостійно доповніть його робочий цикл для утворення повного перемикального графу). Фрагменти випробування щодо самовідновності такого РІР наведено на рис. 8.7.

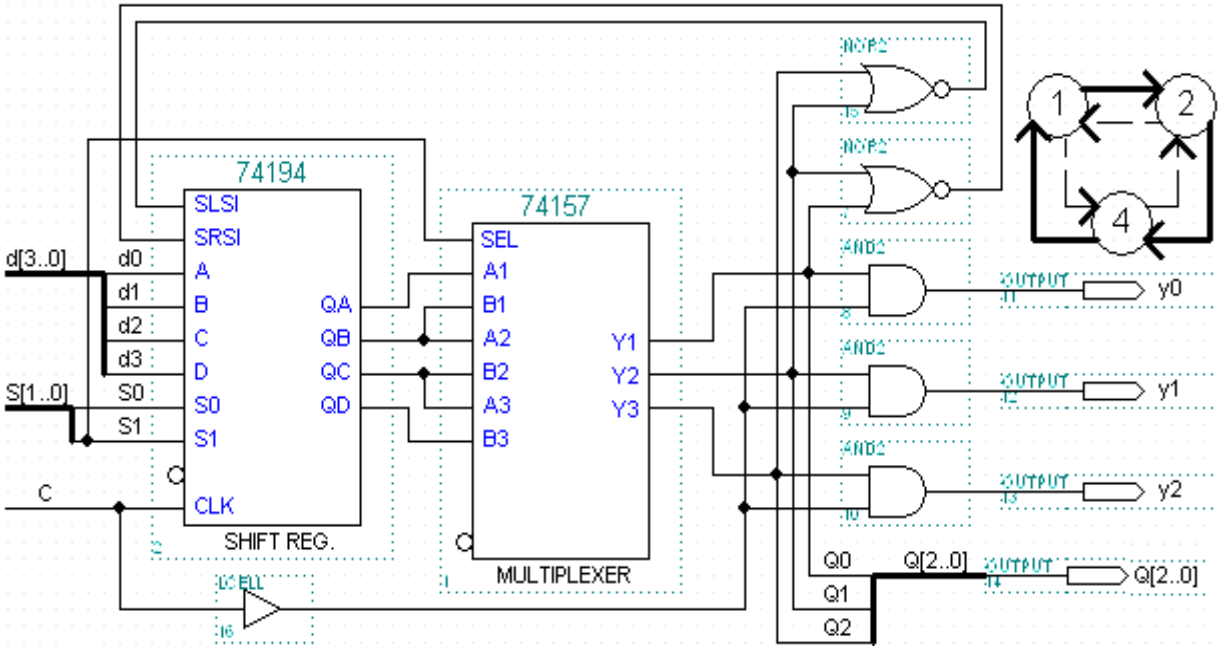


Рисунок 8.7 – Фрагменти випробування РІР щодо самовідновності

КОНТРОЛЬНІ ЗАПИТАННЯ

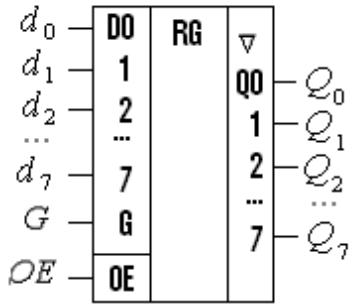
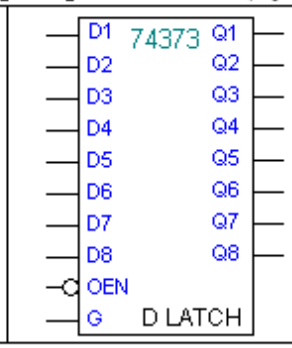
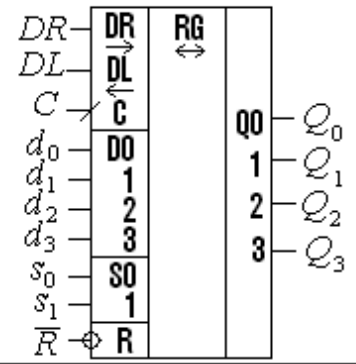
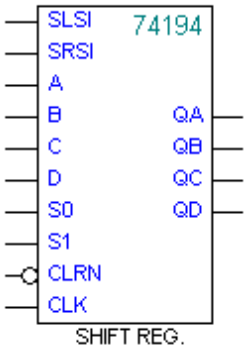
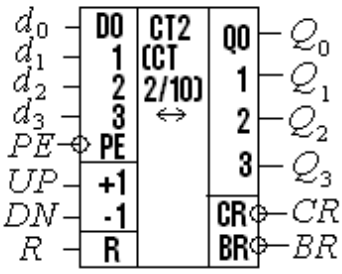
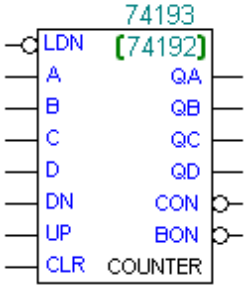
1 Тригери якого типу керування можна застосовувати в паралельних регістрах і регістрах зсуву?

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.

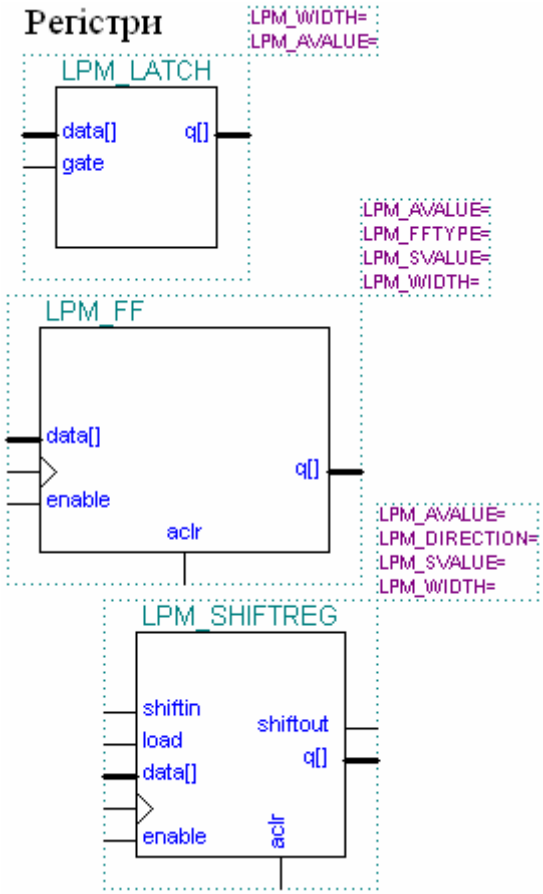
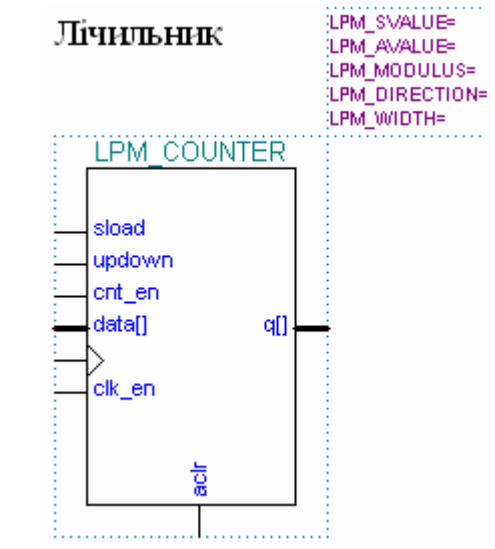
3 Користуючись умовними графічними позначеннями, складіть з ІС 4-розрядних регістрів схему 12-розрядного регістра: а) паралельного, б) зсуву.

4 Перетворіть трирозрядний паралельний регістр у регістр зсуву.

Таблиця 8.1 – Макрофункції послідовнісних пристроїв

Позначення за ДЕСТУ	Типова макрофункція	Таблиця перемикальна
Регістр паралельний (буферний)		
		OE G D | Q⁺ 0 X X | Z 1 0 X | Q 1 1 D | D Q = Q[7..0] D = D[7..0]
Регістр зсуву реверсивний (комбінований)		
		C s_1 s_0 DR DL D | Q_0^+ Q_1^+ Q_2^+ Q_3^+ - X X X X X | $Q_0 Q_1 Q_2 Q_3$ ↑ 0 0 X X X | $Q_0 Q_1 Q_2 Q_3$ ↑ 0 1 D_R X X | $D_R Q_0 Q_1 Q_2$ ↑ 1 0 X D_L X | $Q_1 Q_2 Q_3 D_L$ ↑ 1 1 X X D | d_0 d_1 d_2 d_3 "-" = 0; 1; ↓; $\overline{R} = CLRN = 1$; D = d [3..0]
Лічильник двійковий (двійково-десятковий) реверсивний		
		PE D UP DN | Q^+ CR BR 0 D X X | D X X 1 X ↑ 1 | Q+1 1 1 15 (9) 0 1 1 X 1 ↑ | Q-1 1 1 0 1 0 $R = CLR = 0$

Таблиця 8.2 – Мегафункції послідовнісних пристроїв

Символ	Основні параметри
	**lpm_latch/lpm_ff** – паралельні регістри зі статичним/динамічним керуванням; **lpm_shiftreg** – регістр зсуву (універсальний реверсивний регістр): LPM_WIDTH – розрядність вхідної data[] і вихідної q[] шин; LPM_AVALUE/LPM_SVALUE – константа для асинхронного/синхронного завантаження; LPM_FFTYPE – тип тригерів: DFF або TFF; LPM_DIRECTION – напрямок зсуву: ліворуч/праворуч ("LEFT"/"RIGHT"); aclr/sclr, aset/sset – входи асинхронного/синхронного скидання до $q^+[] = 0...0$ та передустановлення до $q^+[] = 1...1$ або до $q^+[] =$ LPM_(A/S)VALUE; load – вхід синхронного паралельного завантаження $q^+[] =$ data[]; enable – дозвіл на проходження синхроімпульсів clock; shiftin, shiftout – вхід/вихід послідовного введення/виведення.
	**lpm_counter:** LPM_WIDTH, LPM_AVALUE/LPM_SVALUE, aclr/sclr, aset/sset – те саме, що і в регістрів; LPM_MODULUS – модуль лічби; LPM_DIRECTION, updown – напрямок лічби: додавання (up) або віднімання (down); aload/sload – входи асинхронного/синхронного паралельного завантаження $q^+[] =$ data[]; clk_en та cnt_en – дозвіл на проходження лічильних імпульсів clock та дозвіл лічби.

ЛАБОРАТОРНЕ ЗАВДАННЯ

1 Дослідити основні типи регістрів.

1.1 Дослідити ***паралельний регістр*** з трьома станами виходів на основі D-тригерів зі статичним керуванням: за принциповою електричною схемою та осцилограмами сигналів (файли 8par.bdf, .vwf) визначити умови режимів запису, 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 осцилограми сигналів.

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 8zsuw.bdf, .vwf) та розглянути особливості побудови і перемикання регістра зсуву ліворуч (зворотного зсуву) і реверсивного регістра (схеми 2, 3).

1.3 Ознайомитися з ***різновидами регістрів*** бібліотеки бази даних (файл 8libr.bdf): макрофункціями (вибраними ІС серії 74) паралельних регістрів і регістрів зсуву та мегафункціями. У звіті навести функціональні прототипи (FUNCTION), таблиці відповідності та інші довідкові дані регістрів з кожної групи, пояснити призначення та особливості входів і виходів.

2 Застосувати регістр зсуву в *графічному* редакторі для побудови заданого варіанту XX генератора кодової послідовності (ГКП) або розподільника імпульсів/рівнів (РІР).

2.1 Зібрати найпростіший (несамовідновний) пристрій на ***макрофункції*** регістра зсуву вибраного типу в графічному файлі 8XXgkp(ri)1.bdf проекту 8XXgkp(ri)1, виконати компіляцію і моделювання та випробувати його на самовідновність за часовими діаграмами 8XXgkp(ri)1.vwf.

Приклади: 800gkp1.bdf, vwf (схема 1); 800ri1.bdf, vwf (схема 1).

✍ *Примітки:*

1) У наведених прикладах пристрій переводиться до станів поза робочим циклом шляхом паралельного завантаження коду зі входів d_i сигналом *MODE*=1.

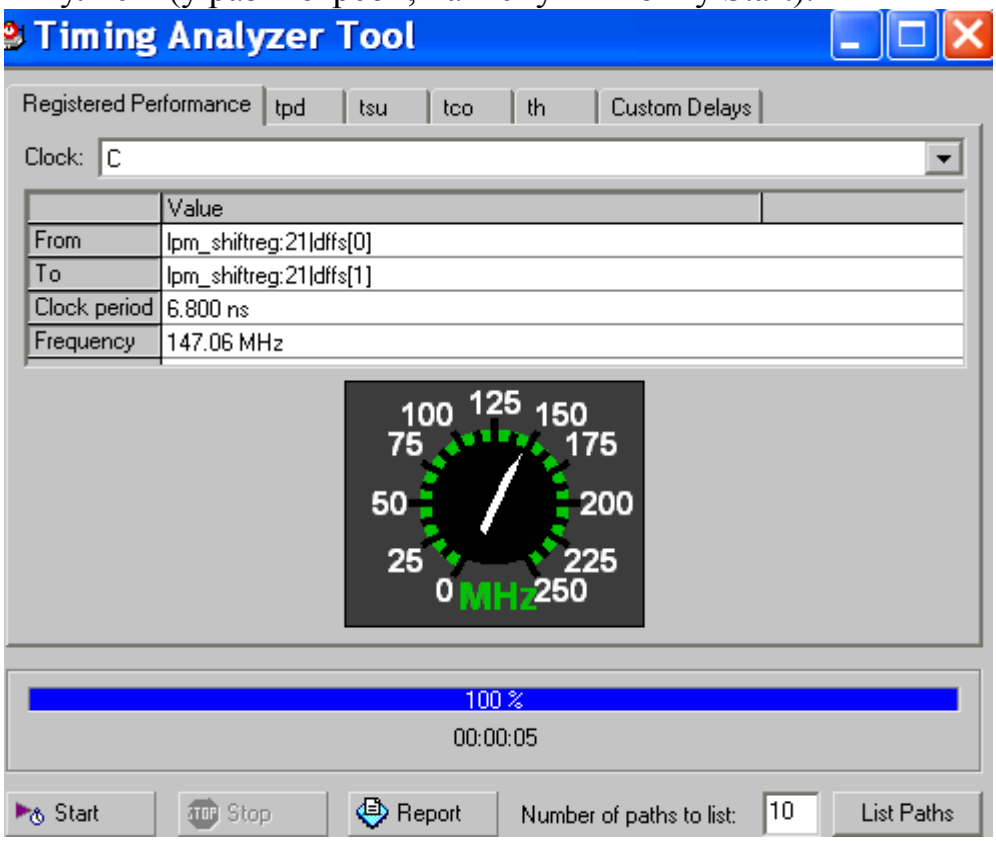
2) Аби уникнути завад на краях імпульсів Q_i, у розподільниках імпульсів/рівнів вибрано регістри з інверсним динамічним керуванням (інакше синхровхід слід зінвертувати під час настроювання макрофункції). У цьому випадку в елементах збігу синхроімпульси розташовуються всередині імпульсів з розрядних виходів, що добре видно на часових діаграмах.

2.2 Скоригувати схему для утворення самовідновного пристрою, виконати п. 2.1 (можна в тому самому файли) та скласти повний перемикальний граф.

2.3 Визначити швидкодію пристрою (максимальну частоту перемикання відносно синхровходу), користуючись звітним файлом компіляції: Compilation Report > Timing Analyzer > Timing Analyzer Summary.

2.4 Визначити швидкодію пристрою (максимальну частоту

перемикання відносно синхровходу) безпосередньо часовим аналізатором: піктограмою Timing Analyzer Tool (або з меню Tools > Timing Analyzer) викликати його вікно Timing Analyzer Tool, перейти на вкладку Re-gistered Performance (швидкодія ЦПП) та зчитати для критичного шляху (From, To) найгіршу швидкодію у формі періоду і частоти надходження синхроімпульсів (у разі потреби, натиснути кнопку Start).



***Приклади**:* 800gkp1.bdf, vwf (схема 2), 800ri1.bdf, vwf (схема 2); реверсивні пристрої: 800gkp0.bdf, vwf, 800ri2.bdf, vwf, 800ri3.bdf, vwf.

2.5 Зберегти таблицю результатів часового аналізу в текстовому файлі: у вікні Timing Analyzer Tool натиснути кнопку Report – з'явиться вікно звіту Compilation Report з розділом Timing Analyzer Summary; клацнути правою кнопкою миші будь-де в полі таблиці звіту (або меню Tools) > Save Current Section As і в діалоговому вікні Save Current Report Section As вибрати тип Timing Analysis Output File (*.tao) та натиснути кнопку збереження; продивитися зазначений файл: натиснути піктограму відкриття файла, прокруткою встановити тип файлів Output Files, вибрати зі списку файл 8XXgkp(ri)1 Timing Analyzer Summary та натиснути кнопку відкриття.

***Приклад**:* 800gkp(ri)1 Timing Analyzer Summary.tao.

2.6 У проекті 8XXgkp(ri)1 (схема 3) зібрати та дослідити самовідновний пристрій на ***мегафункції*** регістра зсуву (можна в тому самому графічному файли).

***Приклад**:* 800gkp1.bdf, vwf (схема 3).

2.7 У проекті 8XXgkp(ri)1 (схема 4) зібрати та дослідити самовідновний пристрій на автоматично створеному за допомогою ***менеджера MegaWizard Plug-In Manager*** різновиді мегафункції регістра зсуву

8XXwiz_reg.bsf (можна в тому самому графічному файли).

Приклад: 800wiz_reg.bsf, .tdf; 800gkp1.bdf, vwf (схема 4).

3 Засвоїти застосування регістра зсуву в *текстовому* редакторі виконанням завдання за п. 2.

3.1 Створити та дослідити пристрій на основі ***макрофункції*** регістра вибраного типу в тестовому файлі (.tdf) проекту 8XXreg_decl на кшталт самовідновної схеми на макрофункції з графічного файла, для чого (як взірець див. файл 800reg_decl.tdf):

а) після заголовку TITLE вставити шаблон Function Prototype Statement (non-parameterized), в який ввести параметри прототипу регістра (можна безпосередньо із символу або скопіювати і вставити копію з довідки пакету MAX+PLUS II):

```
FUNCTION 74295 (ld/shn, clk, ser, d[2..0])
         RETURNS (q[2..0]);
```

б) після оголошення портів у секції підпроекту вставити в секцію змінних (Variable Section) шаблон оголошення регістрів (***Register Declaration***), де ввести умовне ім'я зразка та через двокрапку назву вибраного типу регістра:

```
VARIABLE                        %  Variable Section        %
    rg      : 74295;            %  Register Declaration    %
```

в) до логічної секції (Logic Section) вставити шаблон підсекції булевих рівнянь (Boolean Equation), за допомогою яких утворити з'єднання на зразок графічного файла:

```
BEGIN                                          %  Logic Section        %
rg.ld/shn = MODE; rg.d[2..0] = d[2..0];  %  Boolean Equation     %
rg.clk = C; rg.ser = (q0 !& !q1) !& q2;
Q[2..0] = rg.q[2..0];
END;
```

г) виконати компіляцію і функціональне моделювання та перевірити правильність функціонування пристрою за часовими діаграмами.

Приклад: 800reg_decl.tdf**,** .vwf (на кшталт 800gkp1.bdf, схема 2).

3.2 Виконати п. 3.1 на ***мегафункції*** регістра у тестовому файлі (.tdf) проекту 8XXmega_gkp(ri) на кшталт схеми на мегафункції з графічного файла. Для цього включити до складу проекту мегафункцію регістра зсуву lpm_shiftreg із зазначенням її імені і параметрів в підсекції Instance Declaration (parametererized) секції змінних (Variable Section) та до логічної секції (Logic Section) вставити шаблон підсекції булевих рівнянь (Boolean Equation), за допомогою яких утворити з'єднання на зразок графічного файла. Скомпілювати і змоделювати проект та перевірити правильність функціонування пристрою за часовими діаграмами.

Приклад: 800mega_gkp.tdf, .vwf (на кшталт 800gkp1.bdf, схема 3).

3.3 Виконати п. 3.1 на автоматично створеному за допомогою ***менеджера MegaWizard Plug-In Manager*** різновиді мегафункції регістра у тестовому файлі (.tdf) проекту 8XXwiz_gkp(ri) на кшталт аналогічної схе-

ми з графічного файла. Для цього включити до складу проекту створений у п. 2.5 різновид мегафункцій регістра (Include Statement) без зазначення параметрів у підсекції Instance Declaration (non-parametererized) секції змінних (Variable Section) та до логічної секції (Logic Section) вставити шаблон підсекції булевих рівнянь (Boolean Equation), за допомогою яких утворити з'єднання на зразок графічного файла. Скомпілювати і змоделювати проект та перевірити правильність функціонування пристрою за часовими діаграмами.

Приклад: 800wiz_reg.tdf, 800wiz_gkp.tdf, .vwf (на кшталт схеми 4 файла 800gkp1.bdf).

4 Засвоїти основи побудови послідовнісного пристрою як *скінченного автомата* (State Machine) виконанням завдання за п. 2.

4.1 За перемикальним графом (файл 8XXgkp(ri)1.bdf, рис. 8.5) створити пристрій з використанням ***оператора вибору*** Case Statement у тестовому файлі (.tdf) проекту 8XXsmcase_gkp(ri), для чого після оголошення портів (див. 800smcase_gkp.tdf):

а) у секцію змінних Variable Section вставити шаблон підсекції оголошення скінченного автомата ***State Machine Declaration***, в який ввести символічну 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 символічну назву розрядів автомата, наприклад, (q2, q1, q0) або (q[2..0]); нарешті, після ключових слів WITH STATES у круглих дужках ввести через кому символічні 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 залишити крапку з комою:

```
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);
```

✍ *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).

б) до логічної секції Logic Section вставити шаблон булевих рівнянь

Boolean equations, до якого ввести *булеві рівняння керування*, що зв'язують порти (входи та виходи) автомата із зовнішніми портами пристрою.

```
BEGIN
gkp.clk = C;  Q[] = gkp.q[];
END;
```

✐ *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 зв'язувати з переходами автомата іншими логічними рівняннями або гілками умовних операторів чи таблиці, як, наприклад, у наступному проекті 800smtabl_gkp.

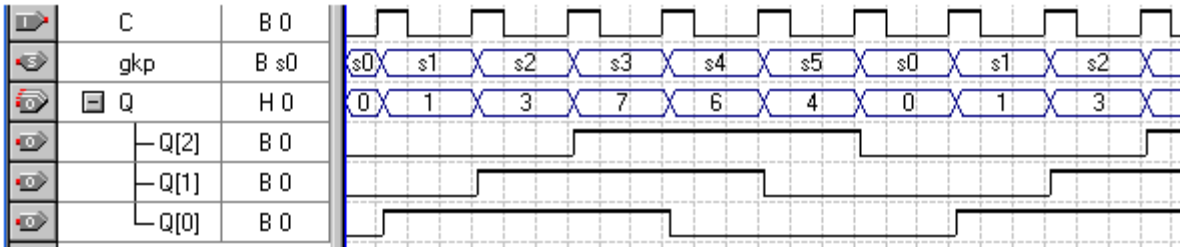
в) після рівнянь до логічної секції вставити шаблон умовного оператора Case Statement (див. Лаб. робота №5, п. 1.4), до якого ввести переходи між станами автомата, так само, як у перемикальному графі:

```
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) по компіляції і функціональному моделюванню перевірити результати проектування за часовими діаграмами 8XXsmcase_gkp(ri).vwf.

Приклад: 800smcase_gkp.tdf, vwf; перемикальний граф 5, 800gkp1.bdf.

✐ ***Примітка.*** Замість оператора вибору Case так само можна використовувати умовний оператор IF THEN.



4.2 За перемикальним графом 6 у файлі 8XXgkp(ri)1.bdf створити пристрій з використанням ***таблиці відповідності*** в тестовому файлі (.tdf) проекту 8XXsmtabl_gkp(ri), для чого виконати п. 4.1, де у п. 4.1,в до логічної секції вставити шаблон таблиці відповідності Truth Table Statement (див. Лаб. робота №5, п. 1.2), до якого ввести переходи між станами автомата, так само, як у перемикальному графі.

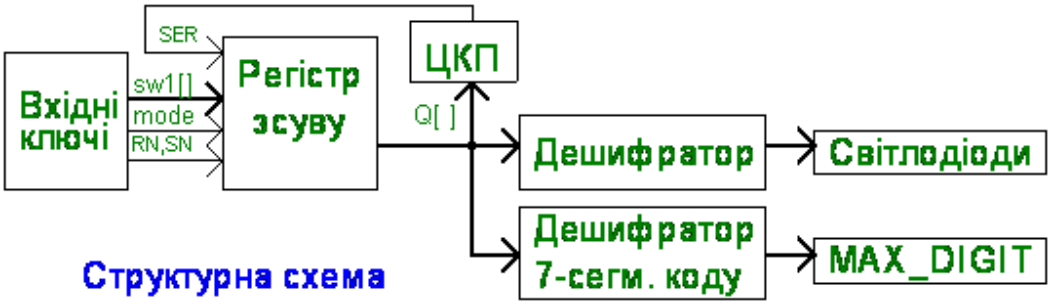
```
TITLE "ГКП - State Machine - перемикальна таблиця";
SUBDESIGN 800smtabl_gkp
(
      C, R, LD     : INPUT;
      Q[2..0]      : OUTPUT;
)
VARIABLE                                  % Variable Section        %
      gkp    : MACHINE                    % State Machine Declaration  %
      OF BITS (q[2..0])
      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: **800**smtabl_gkp.tdf, vwf; перемикальний граф 6, 800gkp1.bdf.

5 Створити проект лабораторного макету на рівні блок-схеми для експериментального дослідження ГКП (PIP) на основі регістра зсуву згідно з варіантом завдання 8 (див. структурну схему і як взірець файл 800GENERATOR_KP.bdf – рис. Д6 у додатках).

Схема власне ГКП складається з регістра зсуву і ЦКП. Виходи пристрою через дешифратор семисегментного коду з'єднано із цифровим індикатором для індикації кодів у десятковій системі і через дешифратор – для відображення розрядів на світлодіодах (у прикладі утворено дві

гірлянди по колу). Синхросигнал утворюється антидеренчливим пристроєм одноразовим натисненням по черзі ключів на входах RN, SN. Для переведення ГКП до стану поза основним циклом з метою перевірки його на самовідновність від вхідних ключів у паралельному коді записується слово sw1[] (натисненням ключа на вході mode, одноразовим натисненням по черзі ключів на входах RN, SN і відтисненням ключа на вході mode).



Структурна схема

5.1 В окремому проекті 8XXGENERATOR_KP(RIR) відтворити ГКП (PIP) за п. 2.2 (шляхом копіювання частини файлів) на мікросхемі родини MAX7000S, типу EPM7128SL84-7. Включити до складу проекту файл дешифратора 7-сегментного коду ../7lab/dc7seg.bdf та створений в окремому файлі і згорнутий до символу дешифратор. Імпортувати призначення виводів мікросхеми з файлу ../4lab/MAX_pin та скоригувати їх. Виконати компіляцію і функціональне моделювання з метою переконатися в правильності проектування.

***Приклад**:* 800GENERATOR_KP.bdf, vwf; 800dc.bdf, .bsf.

5.2 Створити програмувальний CDF-файл, перевірити режим і схему програмування, переконатися, що чотири джампери на платі TDI, TDO, DEVICE, BOARD встановлено в режим програмування однієї мікросхеми та виконати її програмування.

***Приклад**:* 800GENERATOR_KP.cdf.

5.3 Розробити методику та виконати експериментальні дослідження пристрою на запрограмованій ІС. Порівняти результати експериментальних досліджень з даними проектних файлів, зробити висновки.

9 ЛАБОРАТОРНА РОБОТА №9. ЛІЧИЛЬНИКИ

Мета роботи: дослідження типових лічильників; застосування лічильників у проектах; засвоєння методів створення недвійкових лічильників і подільників частоти.

ДОМАШНЄ ЗАВДАННЯ

Спроектувати згідно з варіантом завдання 9 (див. Додатки): а) недвійковий лічильник шляхом перетворення двійкового лічильника, б) недвійковий лічильник на тригерах зі зворотними зв'язками.

СТИСЛІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Означення, класифікацію і параметри лічильників, схемну реалізацію двійкових лічильників з послідовним, паралельним і груповим переносом, а також принцип побудови реверсивних лічильників викладено в [2]. Тут розглянемо проектування недвійкових лічильників.

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

$$n = Ceil\ (log_2 M), \tag{9.1}$$

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

$$2^{n-1} < M < 2^n, \tag{9.2}$$

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.

Існує кілька 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 спочатку на ІС жорсткої структури, коли приступними є лише її зовнішні виводи, зокрема, вхід скидання двійкового лічильника R (рис. 9.1,а). 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.

Методику проектування розглянемо на прикладі перетворення двійкового лічильника в декадний з модулем $M = 10$.

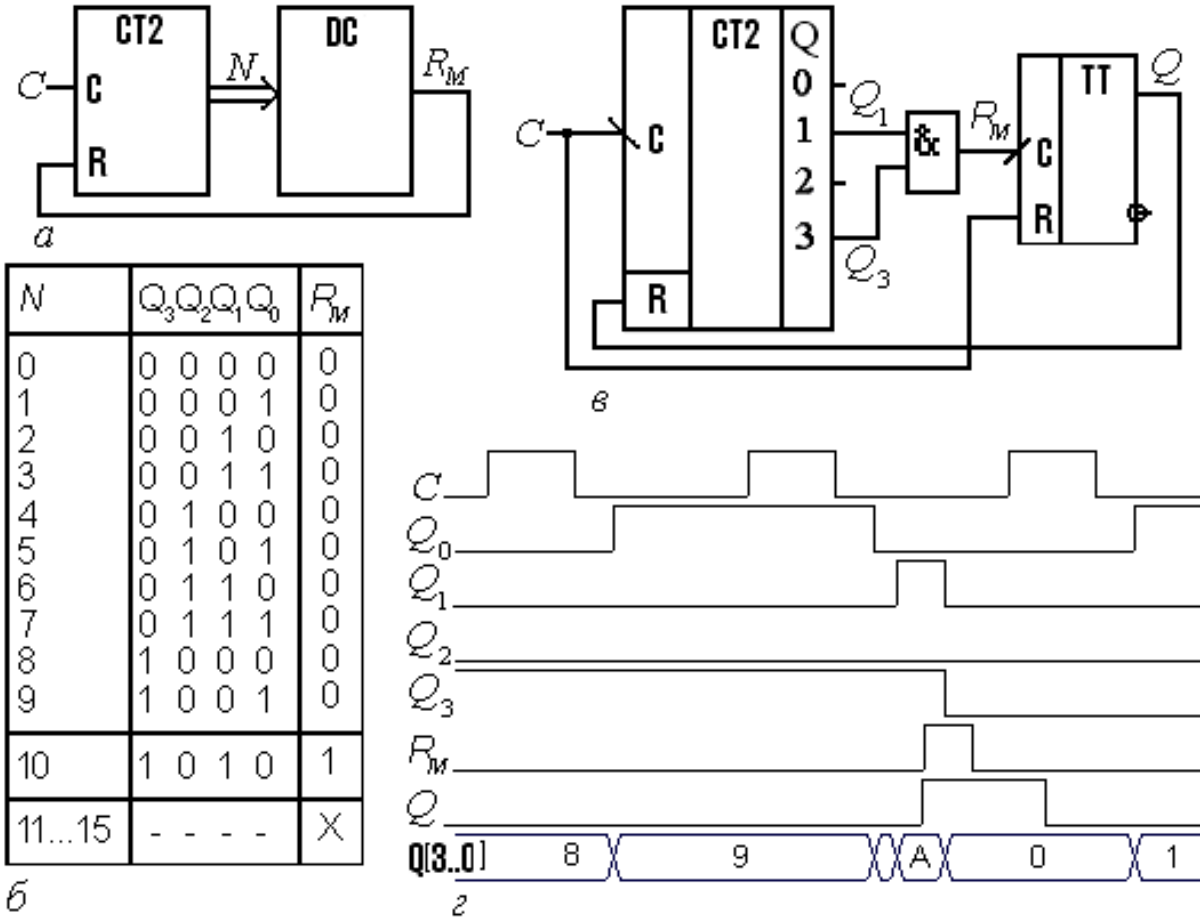


Рисунок 9.1 – Перетворення двійкових лічильників у недвійкові

1) За співвідношеннями (9.1, 9.2) визначаємо потрібну кількість розрядів $n = 4$ і вибираємо ІС лічильника зі входом скидання.

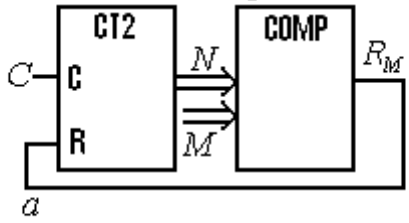
2) Для наочності зазначаємо в перемикальній таблиці (рис. 9.1,б), що коли лічильник переходить до стану $N = M = 10$, виникає сигнал скидання $R_M = 1$, яким він повертається до нульового стану, тому старші коди $N = 11 \ldots 15$ не використовуються.

3) Мінімізуємо звичайним чином за діаграмою термів шукану функцію $R_M = Q_1Q_3$, згідно з якою дешифрування виконує елемент І (рис. 9.1,в).

Проте якщо сигнал R_M подати безпосередньо на вхід скидання лічильника R, певні його розряди можуть скинутися раніше інших, внаслідок чого на виході елемента І встановиться пасивний рівень, коли лічильник не встигне повністю обнулитися. У випадку можливості такої небезпеки слід розширити сигнал R_M затримкою на виході елемента І, наприклад, за допомогою додаткового тригера (див. рис. 9.1,в). Вихідним кодом $N = Q[3..0] = \ = 10_{10} = A_{16}$ встановлюється сигнал $R_M = 1$ (рис. 9.1,г), позитивним перепадом якого тригер перемикається до стану $Q = 1$ і запам'ятовує його до надходження нового імпульсу C. За цей час лічильник надійно обнуляється, а за позитивним перепадом наступного імпульсу C тригер скидається до стану $Q = 0$, відновлюється пасивний рівень на вході R лічильника, тому за негативним перепадом імпульсу C починається новий цикл лічби.

2 Програмовані недвійкові лічильники

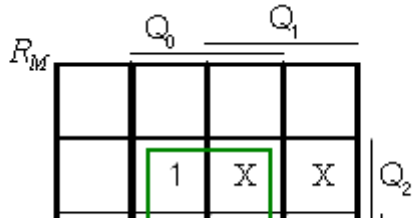
Програмований лічильник за методом примусового скидання двійкового лічильника можна побудувати заміною дешифратора в узагальненій схемі перетворення (див. рис. 9.1,а) на цифровий компаратор (рис. 9.2,а). Коли, під час лічби вихідний код N сягає потрібного модуля M, сигналом R_M лічильник обнуляється, а з надходженням наступних імпульсів на вхід C цикл лічби повторюється.



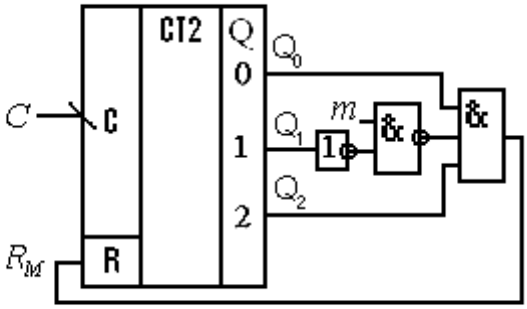
а

N	i	m $Q_2Q_1Q_0$	R_M
0	0	0 0 0 0	0
1	1	0 0 0 1	0
2	2	0 0 1 0	0
3	3	0 0 1 1	0
4	4	0 1 0 0	0
-	5	0 1 0 1	1
-	6, 7	0 1 1 0	X
0	8	1 0 0 0	0
1	9	1 0 0 1	0
2	10	1 0 1 0	0
3	11	1 0 1 1	0
4	12	1 1 0 0	0
5	13	1 1 0 1	0
6	14	1 1 1 0	0
-	15	1 1 1 1	1

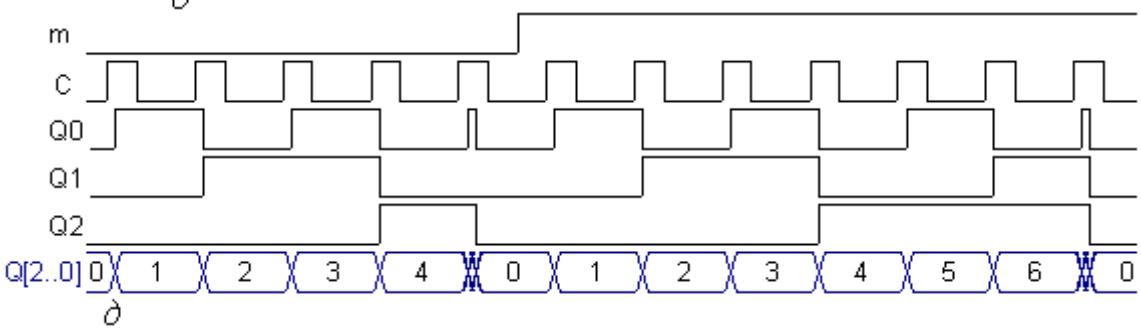
б



в



г



д

Рисунок 9.2 – Програмовані недвійкові лічильники

Особливості проектування програмованого лічильника такого типу розглянемо на прикладі змінюваного модуля лічби $M = 5$ і $M = 7$. За максимальним модулем $M = 7$ вибираємо розрядність лічильника $n = 3$, а за кількістю модифікацій модуля – розрядність керувального коду. У прикладі достатньо обмежитися однорозрядним керувальним кодом m: за його значення $m = 0$ програмуємо модуль на величину $M = 5$, а за значення $m = 1$ – на величину $M = 7$. Якщо кількість модифікацій модуля не перевищує чотирьох, вибираємо дворозрядний керувальний код m_1m_0, якщо восьми – трирозрядний код $m_2m_1m_0$ і т. д.

Аналогічно рис. 9.1,б складаємо перемикальну таблицю (рис. 9.2,б),

в якій зазначаємо, що лічильник скидається сигналом $R_M = 1$, коли він опиняється в проміжних станах $Q_2Q_1Q_0 = 5$ при $m = 0$ та $Q_2Q_1Q_0 = 7$ при $m = 1$. Також позначаємо десяткові значення наборів змінних $i = mQ_2Q_1Q_0$, за якими будуємо діаграму термів (рис. 9.2,в), відтак мінімізуємо функцію, застосовуючи для спрощення редукцію

$$R_M = Q_0Q_2 \setminus m\overline{Q_1} = Q_0\overline{m\overline{Q_1}}Q_2, \qquad (9.3)$$

та складаємо схему (рис. 9.2,г). Затримка комбінаційної частини зазвичай достатня для надійного обнуління лічильника (рис. 9.2,д), але, у разі потреби, імпульси скидання розширюють, наприклад, на кшталт рис. 9.1,в.

На основі трирозрядного двійкового лічильника можна створити програмований лічильник з модулем лічби $M = 2 \ldots 8$ із застосуванням керувального коду $m_2m_1m_0$ шляхом узагальнення (9.3):

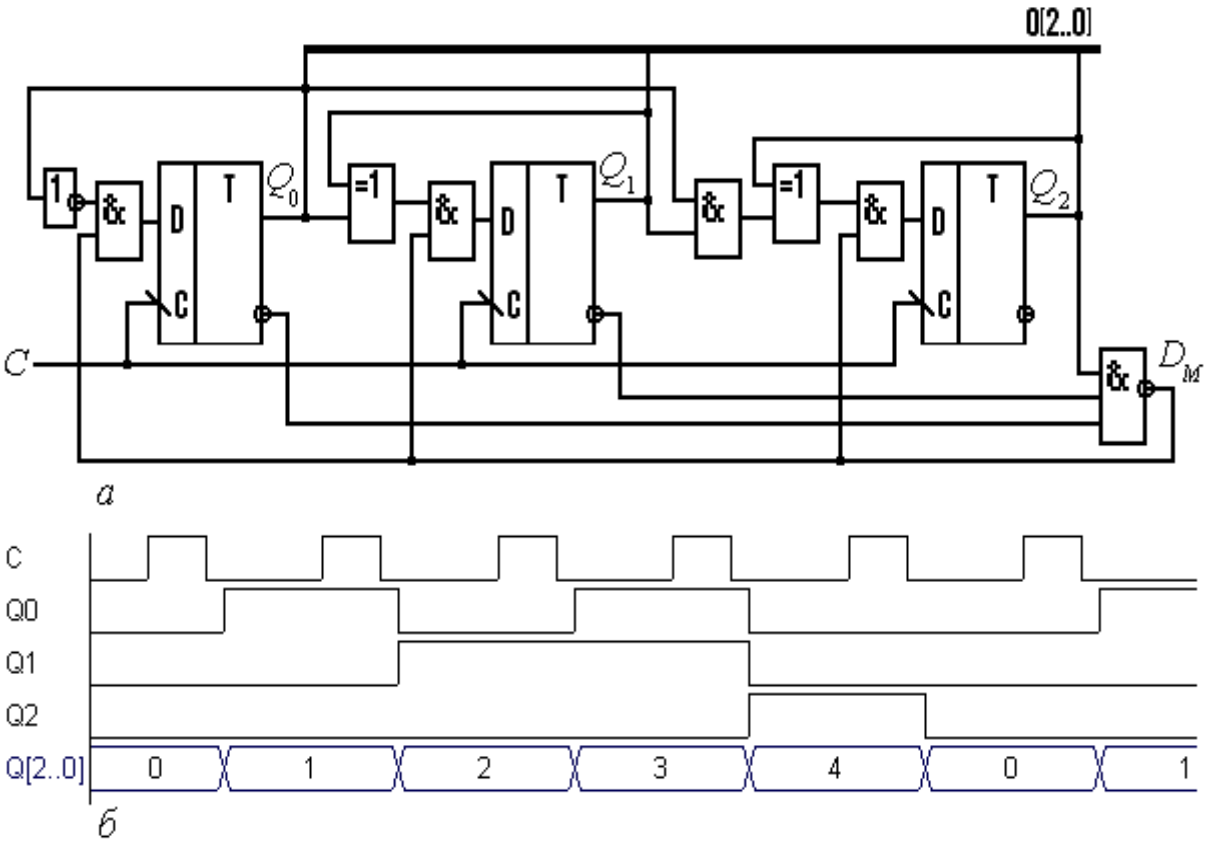
$$R_M = \overline{m_0\overline{Q_0}}\,\overline{m_1\overline{Q_1}}\,\overline{m_2\overline{Q_2}}, \qquad (9.4)$$

отже, аналогічно і для лічильника довільної розрядності.

Слід враховувати, що під час примусового скидання виникають проміжні стани лічильника, тому вихідний код можна знімати по закінченні пе рехідного процесу його усталення. Цей недолік усувають в програмованих ІС шляхом ***модифікації структури*** лічильника. Принцип такої модифікації розглянемо на прикладі перетворення двійкового лічильника за основною схемою на D-тригерах (див. [2], рис. 9.3,б) у лічильник з програмованим модулем лічби $M = 2 \ldots 8$.

З цією метою на входах D_i тригерів (рис. 9.3,а) увімкнемо додаткові елементи І та за допомогою елемента І-НЕ сформуємо керувальний сигнал D_M. Під час лічби залишається рівень $D_M = 1$, який не впливає на функції збудження тригерів, а коли вихідний код набуває значення $M - 1$, цей рівень змінюється на $D_M = 0$, тому з надходженням чергового імпульсу лічильник скидається до нульового стану. Для програмування на потрібний модуль лічби достатньо з'єднати зі входами елемента І-НЕ виходи розрядів відповідно до коду $M - 1$. Так, лічильник з модулем $M = 5$ дістанемо реалізацією числа $M - 1 = 4_{10} = 100_2 = Q_2\overline{Q_1}\,\overline{Q_0}$ (див. рис. 9.3,а). Паралельний лічильник з довільним модулем такого типу перемикається, як і двійковий, без проміжних станів під час зміни вихідного коду (рис. 9.3,б).

Перевагою лічильників з керованим скиданням є природний порядок лічби, тобто їх можна використовувати за прямим призначенням – для підраховування кількості імпульсів. Проте коли порядок лічби не має значення, наприклад, у подільниках частоти, застосовують також способи перетворення двійкових лічильників у недвійкові шляхом усунення проміжних або молодших станів. В останньому випадку по досягненні максимального коду (всі розряди в одиничному стані) здійснюють *примусове нараховування* коду на кількість надлишкових станів зі входів передустановлення лічильника. Це спрощує дешифрування за ознакою негативного перепаду на виході старшого розряду.



а

б

Рисунок 9.3 – Перетворення двійкового лічильника за основною схемою на D-тригерах у лічильник з програмованим модулем лічби *M*=2...8

3 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 – можна шляхом блокування перенесень усунути будь-які надлишкові стани, отже, дістати лічильники з довільним модулем і порядком лічби. Якщо будувати в такий спосіб лічильники з послідовним або комбінованим переносом між окремими розрядами, потрібно визначати функції збудження як інформаційних, так і синхровходів тригерів, а в паралельних лічильників синхровхід всіх розрядів є спільний, 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 на рис. 9.4.

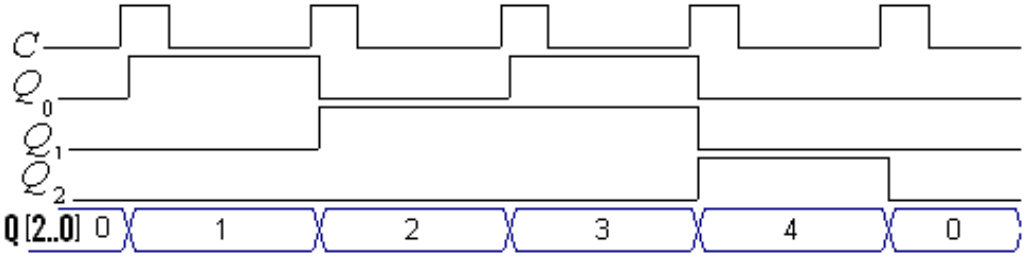


Рисунок 9.4 – Часові діаграми паралельного лічильника з модулем *M*=5 і природним порядком лічби

1) За потрібною кількістю розрядів $n = 3$ у перемикальній таблиці (рис. 9.5,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 надлишкові.

2) Вибираємо тип тригерів (для прикладу розглянемо варіанти лічильника на JK-, D- і TE-тригерах) та заповнюємо стовпці таблиці для функцій збудження на їх інформаційних входах. Так, JK-тригерові 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 для TE-тригерів занотовуємо $E_i = 0$, 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. 9.5,б порожнім клітинкам відповідають факультативні значення X):

$$J_2 = D_2 = Q_0Q_1;\ J_1 = K_1 = E_1 = Q_0;\ J_0 = E_0 = \overline{Q_2};$$

$$D_1 = Q_0 \oplus Q_1;\ D_0 = \overline{Q_0 + Q_2};\ E_2 = Q_0Q_1 + Q_2.$$

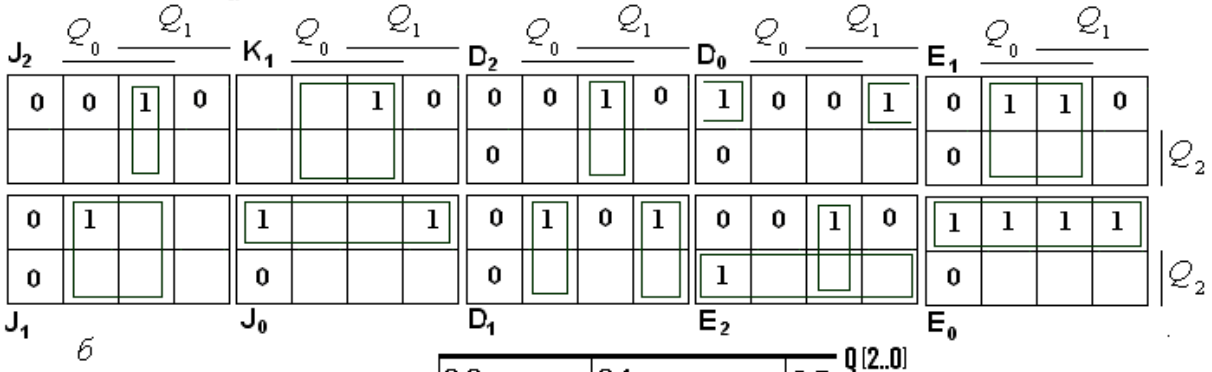
4) З'єднуючи синхровходи тригерів зі спільним лічильним входом C, а інші входи – згідно з функціями збудження, отримуємо варіанти реалізації лічильника на JK-, D- і TE-тригерах (рис. 5,в,г,ґ відповідно). Такі лічильники є синхронні, отже, перемикаються без проміжних станів вихідного коду (див. рис. 9.4).

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. 9.5,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 переносом.

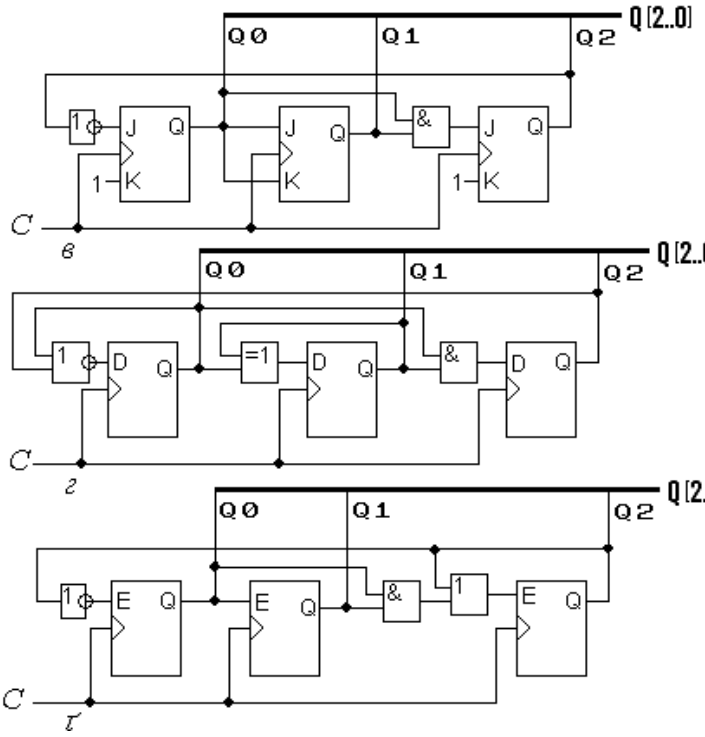
Проектування реверсивних лічильників. Особливості проектування розглянемо на прикладі побудови реверсивного паралельного лічильника з модулем $M = 5$ і природним порядком лічби на JK-тригерах. Перемикальну таблицю доповнюємо стовпцем модифікації схеми (рис. 9.6,а): при $m = 0$ лічильник працює в режимі додавання, а при $m = 1$ – в режимі віднімання, а також стовпцем десяткового коду набору змінних $i = mQ_2Q_1Q_0$. В іншому методика проектування аналогічна.

N	$Q_2 Q_1 Q_0$	$Q_2^+ Q_1^+ Q_0^+$	$J_2 K_2$ $J_1 K_1$ $J_0 K_0$	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* *ґ*

Рисунок 9.5

Безпосередньо з таблиці маємо $K_2 = K_0 = 1$, а інші функції збудження дістанемо з діаграм термів (на рис. 9.6,б порожнім клітинкам відповідають факультативні значення X), застосовуючи прийоми їх сумісної мінімізації:

$$J_0 = Q_2 m + \overline{Q_2}\,\overline{m} + Q_1 = \overline{Q_2 \oplus m} + Q_1 = A + Q_1 \left(\text{де } A = \overline{Q_2 \oplus m}\right);$$

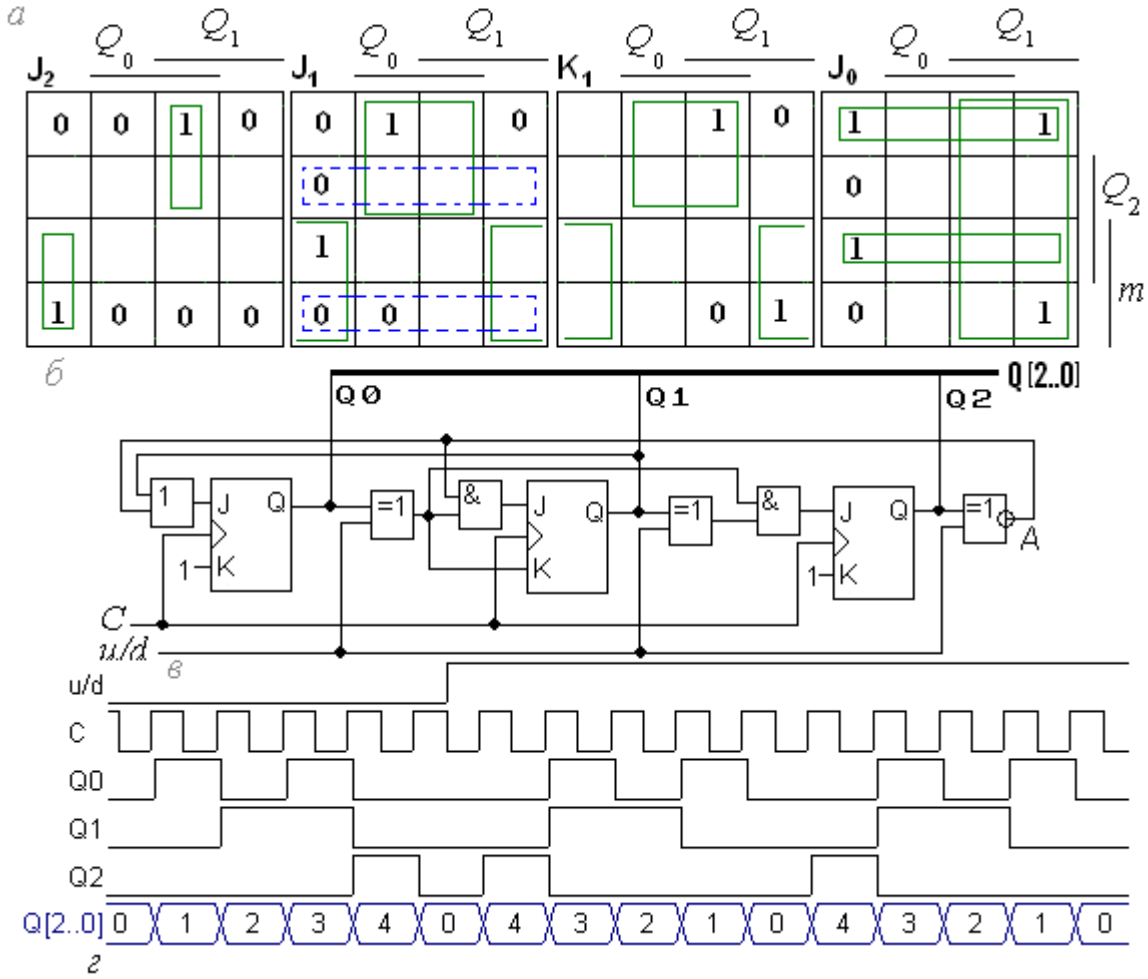
$$K_1 = Q_0 \overline{m} + \overline{Q_0} m = Q_0 \oplus m;\ J_1 = K_1 \setminus \left(Q_2 \overline{m} + \overline{Q_2} m\right) = K_1 A;$$

$$J_2 = K_1 \setminus \left(Q_1 m + \overline{Q_2}\,\overline{m}\right) = K_1 \setminus \overline{Q_1 \oplus m} = \left(Q_1 \oplus m\right) K_1.$$

Цим рівнянням відповідає схема лічильника (рис. 9.6,в), напрямок лічби якого керується сигналом *u/d* = *m* (рис. 9.6,г).

N	i	m	$Q_2Q_1Q_0$	$Q_2^+Q_1^+Q_0^+$	J_2K_2 J_1K_1 J_0K_0	N	i	m	$Q_2Q_1Q_0$	$Q_2^+Q_1^+Q_0^+$	J_2K_2 J_1K_1 J_0K_0
0	0	0	0 0 0	0 0 1	0X 0X 1X	4	12	1	1 0 0	0 1 1	X1 1X 1X
1	1	0	0 0 1	0 1 0	0X 1X X1	3	11	1	0 1 1	0 1 0	0X X0 X1
2	2	0	0 1 0	0 1 1	0X X0 1X	2	10	1	0 1 0	0 0 1	0X X1 1X
3	3	0	0 1 1	1 0 0	1X X1 X1	1	9	1	0 0 1	0 0 0	0X 0X X1
4	4	0	1 0 0	0 0 0	X1 0X 0X	0	8	1	0 0 0	1 0 0	1X 0X 0X
-	-	-	- - -	- - -	XX XX XX	-	-	-	- - -	- - -	XX XX XX

а



б *в* *г*

Рисунок 9.6

4 Подільники частоти

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.

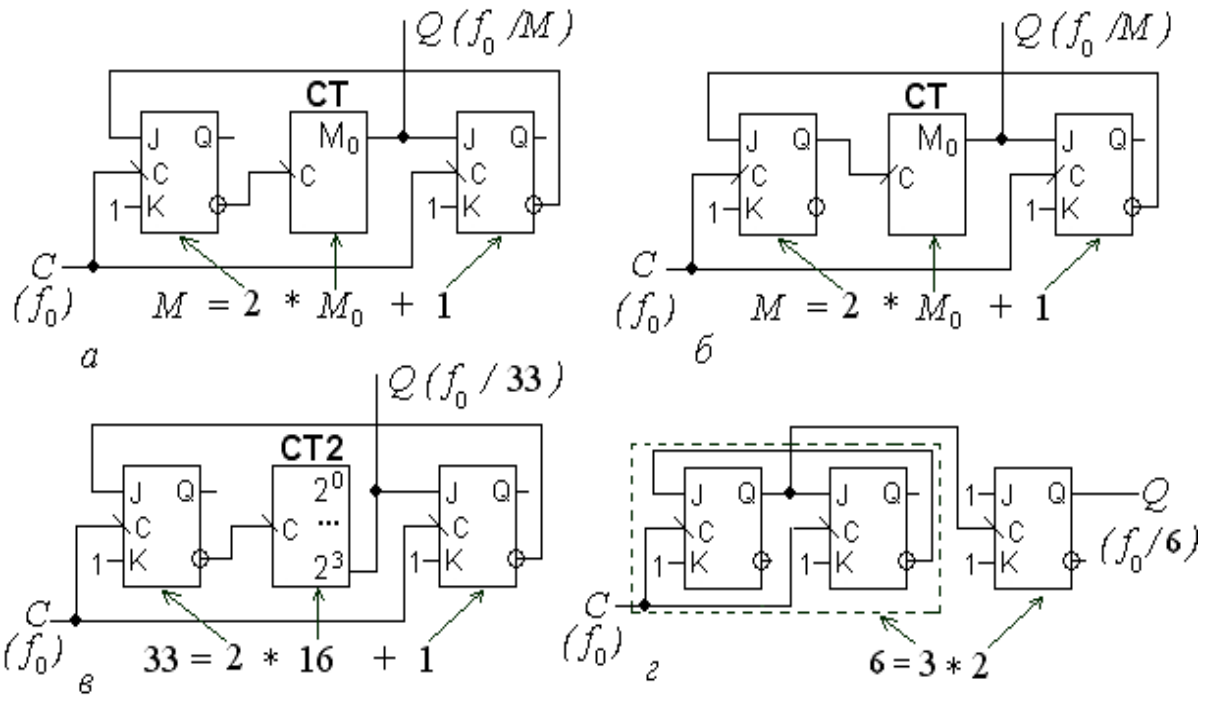


Рисунок 9.7

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. 9.7,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. 9.7,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. 9.7,г). Для отримання парного модуля лічби досить послідовно ввімкнути лічильний тригер на вході або на виході. В останньому випадку (див. рис. 9.7,г) вихідні імпульси Q матимуть форму меандру.

Проектування. Побудову і аналіз функціонування безвентильного лічильника розглянемо на прикладі подільника частоти з модулем лічби (коефіцієнтом поділу) $M = 11$.

1) Розкладаємо модуль на числа степеня 2 та додаткові одиниці: $M = 11 = 10 + 1 = 2 * 5 + 1 = 2 * (4 + 1) + 1 = 2 * (2^2 + 1) + 1$. Підраховуємо потрібну кількість розрядів $n = n_2 + n_1$ як суму степенів два n_2 та додаткових одиниць n_1, яка в прикладі становить $n = 3 + 2 = 5$, тобто потрібно на один тригер більше мінімальної їх кількості за (4). Якщо є варіанти розкладу числа M, вибираємо такий, що потребує меншої кількості розрядів (наприклад, модуль $M = 39$ можна розкласти двома варіантами з розрядністю $n = 7$ та $n = 8$).

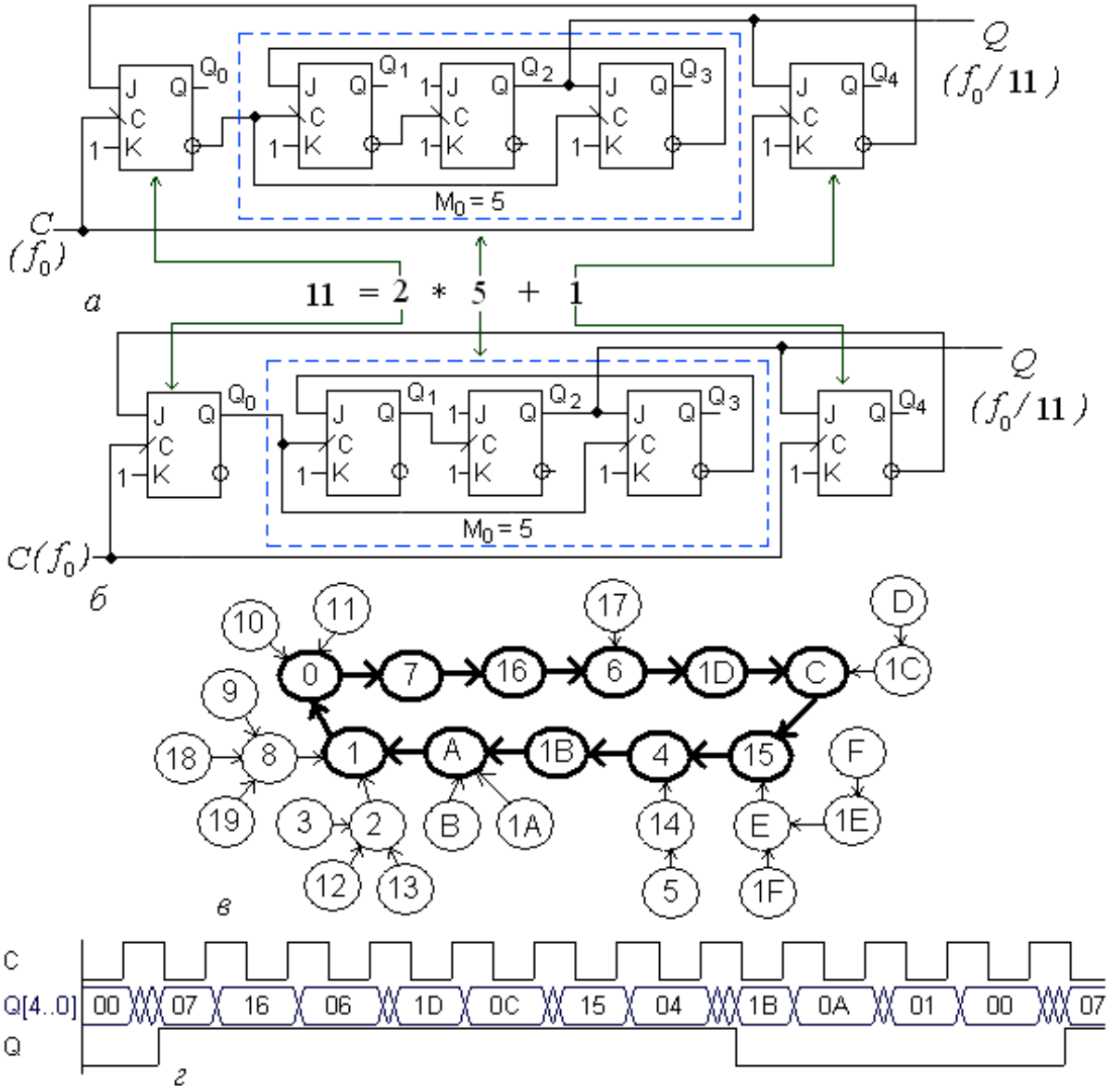


Рисунок 9.8

2) Поділяємо розклад на блоки у вигляді (2 * M_0 + 1), наприклад, M = = 11 = [2 * (2 * 2 + 1) + 1] та реалізуємо ці блоки за типовою схемою (див. рис. 7,а,б) з використанням лічильних тригерів або лічильників для степенів числа 2 і одиничних тригерів для одиниць (рис. 9.8,а,б).

3) Аналіз спроектованого подільника частоти розглянемо докладно, бо він є типовий для будь-якого ЦПП взагалі.

Зазвичай функціонування пристрою зображають перемикальним графом. Для його побудови зручно скористатися перемикальною таблицею (табл. 1), до якої доцільно внести початковий стан N, наприклад, у шістнадцятковій системі числення, його двійковий код Q_4 … Q_0 відповідно до номерів тригерів, функції збудження всіх розрядів $C_iJ_iK_i$, включаючи синхросигнали C_i (бо лічильник не є паралельний), відтак стани Q_i^+, до яких перемкнуться тригери по надходженні чергового лічильного імпульсу, і, нарешті, шістнадцятковий код N^+ нового стану лічильника. Задля наочності позначимо над функціями збудження черговість перемикання роз-

рядів, а під ними – значення цих функцій, які зчитуємо безпосередньо зі схеми (зупинимося для прикладу на схемі рис. 9.8,б).

Таблиця 9.1

N	$Q_4Q_3Q_2Q_1Q_0$	1 $C_4J_4K_4$ $dCQ_2$1	2 $C_3J_3K_3$ $dQ_0Q_2$1	3 $C_2J_2K_2$ $dQ_1$1 1	2 $C_1J_1K_1$ $dQ_0\overline{Q}_3$1	1 $C_0J_0K_0$ $dC\overline{Q}_4$1	$Q_4^+Q_3^+Q_2^+Q_1^+Q_0^+$	N^+
00	0 0 0 0 0	1 0 1	1 0 1	1 1 1	1 1 1	1 1 1	0 0 1 1 1	07
07	0 0 1 1 1	1 1 1	-	-	-	1 1 1	1 0 1 1 0	16
16	1 0 1 1 0	1 1 1	-	-	-	1 0 1	0 0 1 1 0	06
06	0 0 1 1 0	1 1 1	1 1 1	-	1 1 1	1 1 1	1 1 1 0 1	1D
1D	1 1 1 0 1	1 1 1	-	-	-	1 0 1	0 1 1 0 0	0C
0C	0 1 1 0 0	1 1 1	1 1 1	-	1 0 1	1 1 1	1 0 1 0 1	15
15	1 0 1 0 1	1 1 1	-	-	-	1 0 1	0 0 1 0 0	04
04	0 0 1 0 0	1 1 1	1 1 1	1 1 1	1 1 1	1 1 1	1 1 0 1 1	1B
1B	1 1 0 1 1	1 0 1	-	-	-	1 0 1	0 1 0 1 0	0A
0A	0 1 0 1 0	1 0 1	1 0 1	-	1 0 1	1 1 1	0 0 0 0 1	01
01	0 0 0 0 1	1 0 1	-	-	-	1 1 1	0 0 0 0 0	00
02	0 0 0 1 0	1 0 1	1 0 1	-	1 1 1	1 1 1	0 0 0 0 1	01
...	...	...	...	...	...	...	...	...
1A	1 1 0 1 0	1 0 1	-	-	-	1 0 1	0 1 0 1 0	0A

З лічильного входу C імпульси потрапляють безпосередньо до першого і останнього тригерів, першочерговість перемикання яких позначаємо цифрою 1, перепад цих імпульсів на їх синхровходах – у вигляді $C_4 = \ = C_0 = dC$ та зчитуємо зі схеми сигнали $J_4 = Q_2$, $K_4 = 1$, $J_0 = \overline{Q_4}$, $K_0 = 1$.

Наслідком перемикання молодшого розряду є перепад dQ_0, яким керуються тригери з виходами Q_1, Q_3, тому черговість їх спрацьовування позначаємо цифрою 2, функції синхровходів – $C_3 = C_1 = dQ_0$ та зчитуємо зі схеми сигнали на входах J і K. І, нарешті, в останню чергу (цифра 3) перемикається середній тригер перепадом $C_2 = dQ_1$.

Відтак вносимо початковий код, наприклад, N = 00 і заповнюємо клітинки таблиці з функціями першої черги, позначаючи активний перепад $dC = 1$ (або стрілкою догори, бо в нас активними є позитивні перепади), а вихідні стани тригерів Q_i беремо з лівої частини таблиці: $J_4 = Q_2 = 0$, $J_0 = \overline{Q_4} = 1$. За цими функціями збудження визначаємо наступний стан двох тригерів по надходженні лічильного імпульсу і вносимо його до правої частини таблиці: $Q_4^+ = 0$, $Q_0^+ = 1$. Позитивний перепад сигналу $Q_0 \rightarrow Q_0^+ = 0 \rightarrow 1$ позначаємо для тригерів другої черги як $dQ_0 = 1$, так само заното-

вуємо їх функції збудження та наступні стани $Q_3^+ = 0$, $Q_1^+ = 1$. У третю чергу позитивним перепадом $dQ_1 = 1$ перемикається до протилежного стану середній тригер (бо $J_2 = K_2 = 1$), тому занотовуємо $Q_2^+ = 1$.

Далі, за початковий беремо новий стан 07 і в другому рядку таблиці визначаємо так само переходи тригерів, але відсутність активного перепаду на синхровходах для стислості позначаємо рискою (негативний перепад dQ_0 не спричиняє перемикання ланцюжка тригерів другої, отже, і третьої черги), тому немає сенсу занотовувати значення на входах J та K.

Продовжуємо таблицю, доки не замкнеться робочий цикл лічильника (у прикладі він зі стану 01 повертається до початкового стану 00) і зображаємо його на перемикальному графі, наприклад, грубими лініями як на рис. 9.8,в (у вершинах графа стани подано, як і в перемикальній таблиці шістнадцятковим кодом).

Для побудови повного перемикального графа вносимо до таблиці стан поза робочим циклом, наприклад, 02 і так само визначаємо переходи тригерів, доки не переберемо всі можливі 2^n станів та переносимо їх на граф (тонкі лінії на рис. 9.8,в). Відсутність хибних циклів перемикального графа свідчить про самовідновність лічильників за схемами на рис. 9.8,а,б та за узагальненими схемами на рис. 9.7,а,б.

4) За перемикальним графом та таблицею будуємо часові діаграми вихідного сигналу подільника частоти Q та, у разі потреби, на всіх розрядних виходах (рис. 9.8,г). Як бачимо, частота на виході Q в 11 разів менша за вхідну, а повний час усталення вихідного коду становить $t_Л = 3t_T$ внаслідок послідовного перемикання трьох черг тригерів. Проте за використання лічильника для поділу частоти швидкодія визначається лише часом перемикання тригера молодшого розряду.

Таким чином, безвентильні подільники частоти відрізняються простотою схеми, хоч і можуть містити кількість тригерів, більшу мінімальної, що визначається за (1).

КОНТРОЛЬНІ ЗАПИТАННЯ

1 Порівняйте лічильники з переносом 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}, є) 2^{11}, ж) 2^{15}.

3 Спроектуйте лічильник з модулем 2778 і природним порядком лічби (для вимірювача фазових зсувів) на основі: а) макрофункцій двійкових лічильників, б) макрофункцій декадних лічильників, в) мегафункції лічильника.

ЛАБОРАТОРНЕ ЗАВДАННЯ

1 Дослідити основні типи лічильників.

1.1 Дослідити ***двійковий підсумовувальний лічильник з послідовним переносом*** (послідовний лічильник) на основі Т-тригерів: за принциповою електричною схемою і осцилограмами сигналів (схема 1, файли 9dwij.bdf, .vwf) 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.

1.2 Дослідити ***двійковий підсумовувальний лічильник з паралельним переносом*** (паралельний лічильник) на основі ТЕ-тригерів за п. 1.1 (схема 2, файли 9dwij.bdf, .vwf). Розглянути особливості побудови і перемикання базової схеми на D-тригерах (схема 3, файли 9dwij.bdf, .vwf) та застосування її для побудови формувача адреси шляхом розгортання коду в часі з антидеренчливим пристроєм на вході (файли 9addr.bdf, .vwf). У звіті навести схеми, осцилограми сигналів, відомості щодо швидкодії.

1.3 Розглянути особливості побудови і перемикання двійкового лічильника в режимі віднімання (схема 4, файли 9dwij.bdf, .vwf) та дослідити ***двійкові реверсивні лічильники*** з подаванням вхідних імпульсів до однієї точки (схема 5, файли 9dwij.bdf, .vwf) і двох точок схеми (схема 6, файли 9dwij.bdf, .vwf). У звіті навести схему реверсивного лічильника на D-триге-рах, осцилограми сигналів, стисле пояснення принципу його дії.

1.4 Дослідити ***недвійковий лічильник***, побудований шляхом ***перетворення двійкового*** (двійковий лічильник репрезентовано файлами 9m16.bdf, .bsf, .vwf) скиданням його надлишкових станів за допомогою дешифратора (репрезентовано файлами 9dc.bdf, .bsf) на прикладі декадного лічильника (схема 1, файли 9nedw.bdf, .vwf). Розглянути особливості побудови і перемикання лічильника, в якому за дешифратор править елемент І (схема 2, файли 9nedw. bdf, .vwf), ***декадного лічильника*** на стандартній ***макрофункції*** – ІС серії 74 (схема 3, файли 9nedw.bdf, .vwf) та реверсивного декадного лічильника на ***мегафункції*** (схема 4, файли 9nedw.bdf, .vwf). У звіті навести схеми 1 або 2 та 4, осцилограми сигналів таких лічильників, стисле пояснення принципу їх дії.

1.5 Ознайомитися з ***різновидами двійкових лічильників*** бібліотеки бази даних (файл 9libr.bdf): 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 та 6) з універсальною ***мегафункцією*** лічильника. У звіті навести принаймні по одому символу з груп 1...6, пояснити призначення та особливості входів і виходів, а також параметри мегафункції.

2 Спроектувати недвійковий лічильник із заданим модулем *M* (згідно з варіантом завдання 9,а) **і природним порядком лічби шляхом *перетворення* двійкового лічильника.**

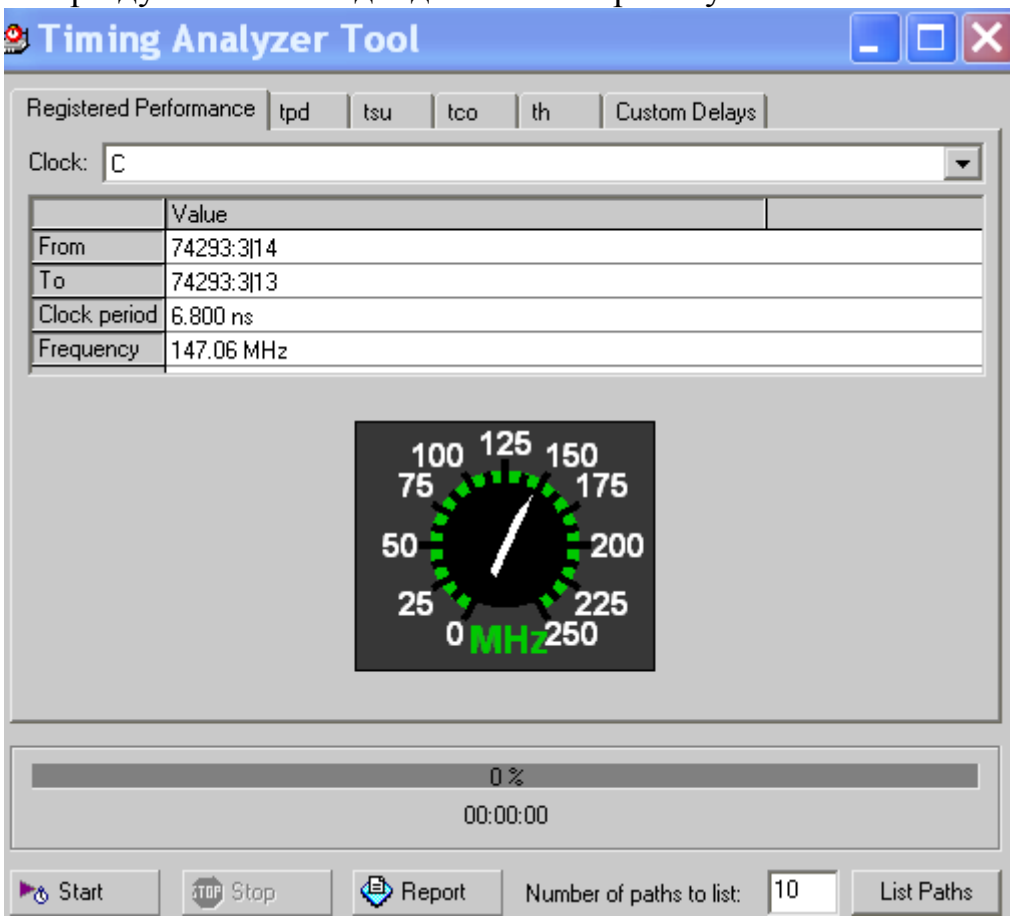
2.1 Побудувати лічильник на основі *макрофункції* вибраного типу за *графічного* (.bdf) введення проекту 9XXper_gr, виконати компіляцію та функціональне моделювання (.vwf).

***Приклад**:* програмований лічильник з природним порядком лічби і модулем *M* = 5, 7 на макрофункції 74293 у файлах 900per_gr.bdf, .vwf.

2.2 В автоматичному режимі визначити швидкодію спроектованого послідовнісного пристрою:

а) у разі потреби, послідовністю наведених піктограм відкрити аналізований проект (переконатися, що його директорію і назву відображено в рядку заголовка головного вікна) та задля наочності відкрити файл верхнього рівня проекту (у нашому випадку це є графічний файл);

б) піктограмою Timing Analyzer Tool (або з меню Tools > Timing Analyzer Tool) відкрити вікно часового аналізатора, перейти на вкладку визначення швидкодії послідовнісних пристроїв Registered Performance і зчитати для критичного шляху (From, To) найгіршу швидкодію у формі періоду і частоти надходження синхроімпульсів.



в) зберегти таблицю результатів часового аналізу в текстовому файлі: у вікні Timing Analyzer Tool натиснути кнопку Report – з'явиться вікно звіту Compilation Report з розділом Timing Analyzer Summary; клацну-

ти правою кнопкою миші будь-де в полі таблиці звіту (або меню Tools) > Save Current Report Section As і в діалоговому вікні Save Current Report Section As вибрати тип файла Timing Analyzer Output File (*.tao) та натиснути кнопку збереження;

г) продивитися зазначений файл: натиснути піктограму відкриття файла, прокруткою встановити тип файлів Output Files, вибрати зі списку файл 9XXper_gr Timing Analyzer Summary та натиснути кнопку відкриття.

***Приклад**:* 900per_gr Timing Analyzer Summary.tao.

2.3 Виконати п. 2.1 за *текстового* (.tdf) введення проекту 9XXper_tx.

***Приклад**:* програмований лічильник з природним порядком лічби і модулем $M = 5, 7$ на макрофункції 74293 у файлах 900per_tx.tdf, .vwf.

✍ *Примітка:* У текстовому файлі (підсекція Boolean Equation) імена виводів зразка макрофункції мають точно відповідати його функціональному прототипові, який можна записати безпосередньо з графічного символу або скопіювати з довідки Help САПР MAX+PLUS II, 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).

3 Спроектувати паралельний лічильник *зі зворотними зв'язками* із заданими модулем *M* і порядком лічби (згідно з варіантом завдання 9,б) **та дослідити його, користуючись часовими діаграмами.**

3.1 Побудувати лічильник на *тригерах* заданого типу за *графічного* (.bdf) введення проекту 9XXzz_gr, виконати компіляцію та функціональне моделювання (.vwf).

***Приклади**:* лічильник з природним порядком лічби і модулем 5 на тригерах типу JK (схема 1), D (схема 2), TE (схема 3) та реверсивний лічильник на JK-тригерах (схема 4) у файлах 900zz_gr.bdf, .vwf.

3.2 Виконати п. 3.1 на *мегафункції* лічильника в тому самому проекті (примірник мегафункції можна ввести і одночасно настроїти з діалогового вікна Symbol: кнопкою MegaWizard Plug-In Manager викликати однойменне вікно, на сторінці 2а ліворуч вибрати категорію arithmetic > мегафункцію LPM_COUNTER і праворуч після риски ввести ім'я, наприклад, CT5; на сторінці 3 ввести ширину шини в бітах, наприклад, 3 і ввімкнути перемикач напрямку лічби, наприклад, нижній для створення входу 'updown': 1 – додавання, 0 – віднімання; на сторінці 4 ввімкнути перемикач Modulus і ввести модуль лічби, наприклад, 5). На завершення включити створений різновид мегафункції CT5.tdf до проекту.

***Приклад**:* реверсивний лічильник з природним порядком лічби і модулем 5 на мегафункції (схема 5) у файлах 900zz_gr. bdf, .vwf.

3.3 Виконати п. 3.2 за *текстового* (.tdf) введення проекту 9XXzz_tx.

Для цього доцільно скористатися створеним у п. 3.2 за допомогою менеджера MegaWizard Plug-In Manager різновидом мегафункції лічильника, наприклад, CT5. Файл включення (у прикладі CT5.inc) вставляється оператором Include Statement перед секцією підпроекту Subdesign Section.

***Приклад**:* реверсивний лічильник з природним порядком лічби і модулем 5 на зразку мегафункції CT5.inc у файлах 900zz_ tx.tdf, .vwf.

✍ Примітки:

1) Зауваження щодо відповідності імен стосується і мегафункції. Наприклад, з файла включення автоматично створеного її зразка CT5.inc (відкривається з типу файлів Other Source Files) зчитуємо прототип

```
FUNCTION CT5.inc (clock, updown)
                 RETURNS (q[2..0]);
```

і використовуємо в рівняннях імена виводів ct.clock, ct.updown, ct.q[]. Імена виводів можна взяти також із символьного файла CT5.bsf (відкривається з типу файлів Graphic Files або Other Source Files).

2) У разі потреби, зразок мегафункції можна створити безпосередньо менеджером MegaWizard Plug-In Manager, який викликається з меню Tools > MegaWizard Plug-In Manager.

4 Створити проект лабораторного макету для експериментального дослідження лічильника на прикладі заданого варіанту недвійкового лічильника за п. 2 або 3 за вибором (див. як взірець файл 900LICHYLNYK.bdf – рис. Д7 у додатках).

Для цього вставити зазначений лічильник до графічного файлу, який доповнити іншими компонентами, потрібними для індикації вихідного коду і модулю лічби (у прикладі модуль лічби індукується світлодіодами VD1, VD2). Імпортувати призначення виводів мікросхеми з файлу ../4lab/MAX_pin.qsf та скоригувати їх. Виконати компіляцію і функціональне моделювання з метою переконатися в правильності проектування. Створити програмувальний CDF-файл і виконати програмування мікросхеми. Розробити методику та виконати експериментальні дослідження пристрою на запрограмованій ІС. Порівняти результати експериментальних досліджень з даними проектних файлів, зробити висновки.

***Приклад**:* програмований лічильник $M = 5, 7$ – файли 900LICHYLNYK.bdf, .vwf, .cdf.

10 ЛАБОРАТОРНА РОБОТА №10. ПРОГРАМОВАНІ МІКРОСХЕМИ

Мета роботи: використання мікросхем різних типів на прикладі побудови подільників частоти; програмовані логічні матриці, архітектура ПЛІС, інтерфейс JTAG, схеми програмування і конфігурування; редактор фізичного розміщення; конфігурування мікросхеми.

ДОМАШНЄ ЗАВДАННЯ

Спроектувати згідно з варіантом завдання 10 (див. Додатки) безвентильний подільник частоти на JK-тригерах.

СТИСЛІ ТЕОРЕТИЧНІ ВІДОМОСТІ

У [2] викладено загальні відомості про сучасні ПЛІС (їх класифікацію за способами програмування структури, програмовані елементи ПЛІС, принцип побудови програмованих логічних матриць), елементи структури (макрокомірка, блок керування вводом/виводом), а також відомості щодо програмування і конфігурування (режими програмування і конфігурування, інтерфейс JTAG, схеми програмування і конфігурування). Тут розглянемо стисло питання архітектури ПЛІС і наведемо для прикладу відомості про дві родини ПЛІС фірми Altera з довідки САПР Quartus II.

1 Архітектура ПЛІС

1.1 *План розміщення ПЛІС.*

ПЛІС, що крім багатьох програмованих логічних матриць (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. 1,а. Компілятор автоматично призначає елементи схеми, необхідні для реалізації проекту, виконує потрібні з'єднання і вносить всю цю інформацію до об'єктного файлу (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 приступний для кожного елемента набору. На плані ЛБ позначено літерами A, B біля лотків, в яких відображаються призначення відповідного блоку. ІС можуть вміщувати до 32 ЛБ.

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). Призначені (задіяні) комірки на плані забарвлюються, а вільні залишаються білими.

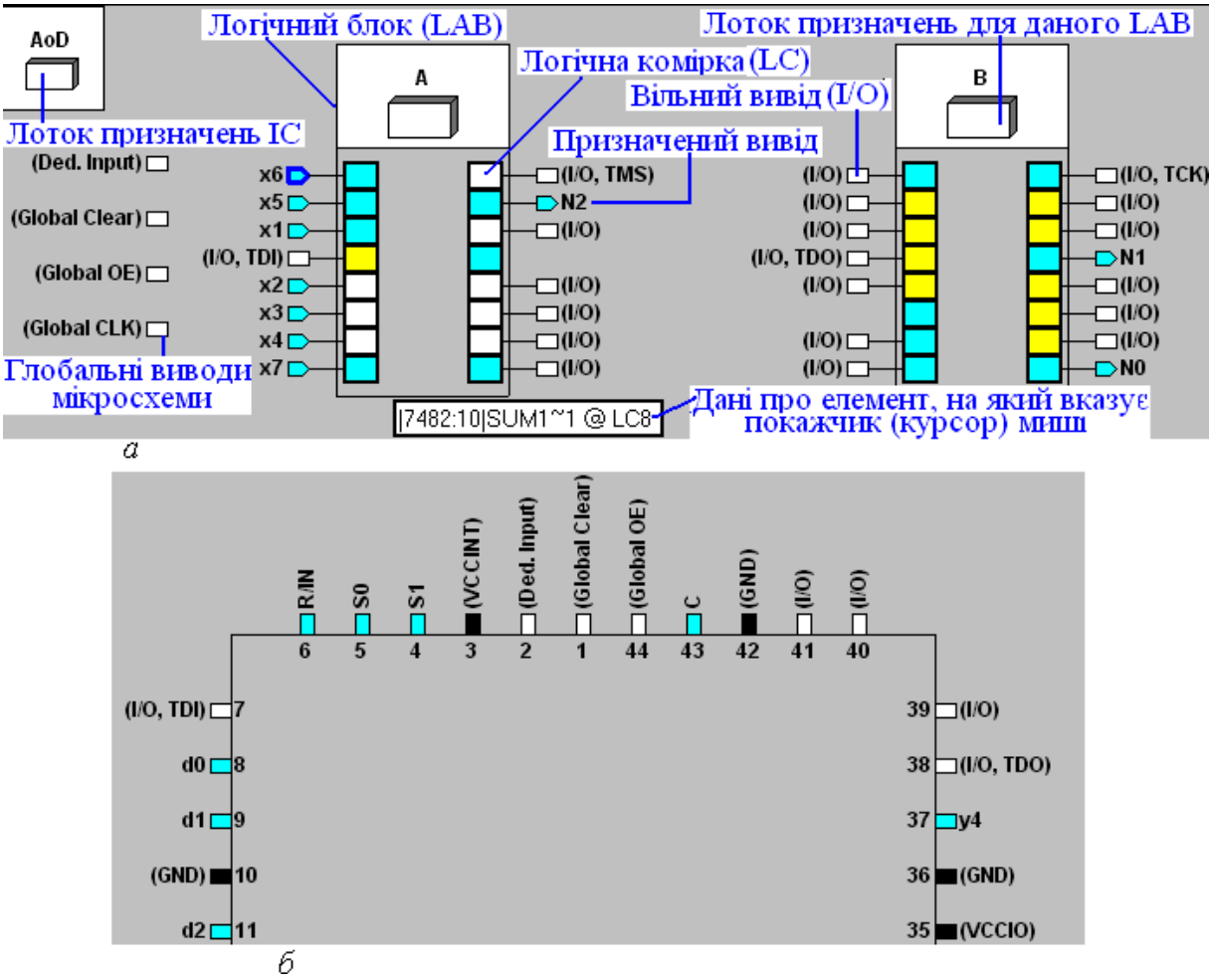


Рисунок 10.1

Так само і виводи (контакти, штирки – Pin) мікросхеми на плані (див. рис. 10.1,а) і на цоколівці (на рис. 10.1,б подано її фрагмент), що є призначені, забарвлено, а вільні – білі. Основна їх 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 файлу. Інші виводи є спеціалізовані (Dedicated). Перш за все це GND – земля та VCC або VCCINT і VCCIO – напруги живлення; ці виводи, як і незадіяні (N.C. – No Connect), якщо вони є, наводяться лише на цоколівці (виводи GND у схемі не можна залишати вільними, навіть якщо їх декілька). Наступна група спеціалізованих виводів, призначена для фізичного програмування (TDI, TDO, TCK, TMS), є неприступною для користувача. 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), GCLRn – Global Clear (спільний сигнал скидання, зазвичай інверсний), Global OE – дозвіл виходу для компонентів з трьома станами, Ded. Input (від Dedicated – спеціалізований) – глобальний вхід за призначенням користувача.

Будь-який вивід IC, 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 зв'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.

1.2 ***Структура ПЛІС з електричним стиранням*** **(EPLD).**

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. 10.2 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 обабіч ярусами розташовуються логічні блоки ЛБ, кожний зі своїм блоком керування вводом/виводом БКВВ. Показані знизу інтерфейсні кола і сигнали діють у режимі програмування.

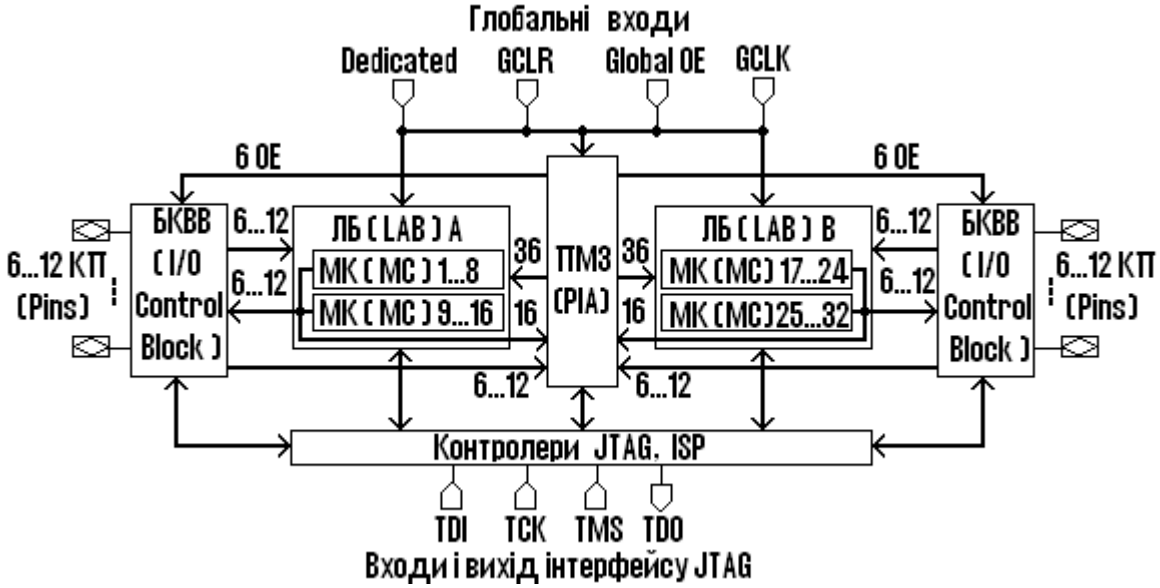


Рисунок 10.2 – Зв'язки між компонентами ПЛІС

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 двотактних схем), інформаційні сигнали з контактних площинок КП входів/виходів загального користування через

БКВВ (показано обабіч нижніми шинами розрядністю 6...12 – за кількістю КП, що обслуговуються даним ЛБ) та сигнали з виходів макрокомірок ЛБ (показано 16-розрядними шинами – за кількістю МК у кожному ЛБ).

Усі сигнали ПМЗ стають приступними 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 вводом/виводом 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 ОЕ), досягається гнучкість керування блоком.

1.3 ***Особливості архітектури ПЛІС із тригерною пам'яттю конфігурації.***

З розвитком технології і зростанням ступеня інтеграції з'явилися ІС комбінованого типу, що сполучають 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. 10.3. Як і для ПЛІС типу CPLD (див. рис. 10.1,а), зв'язки між компонентами такої ПЛІС так само можна відображати на її плані графічно або логічними рівняннями.

Призначення (Assignments) для компонентів містяться в лотках, 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".

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 лі-

чильників 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".

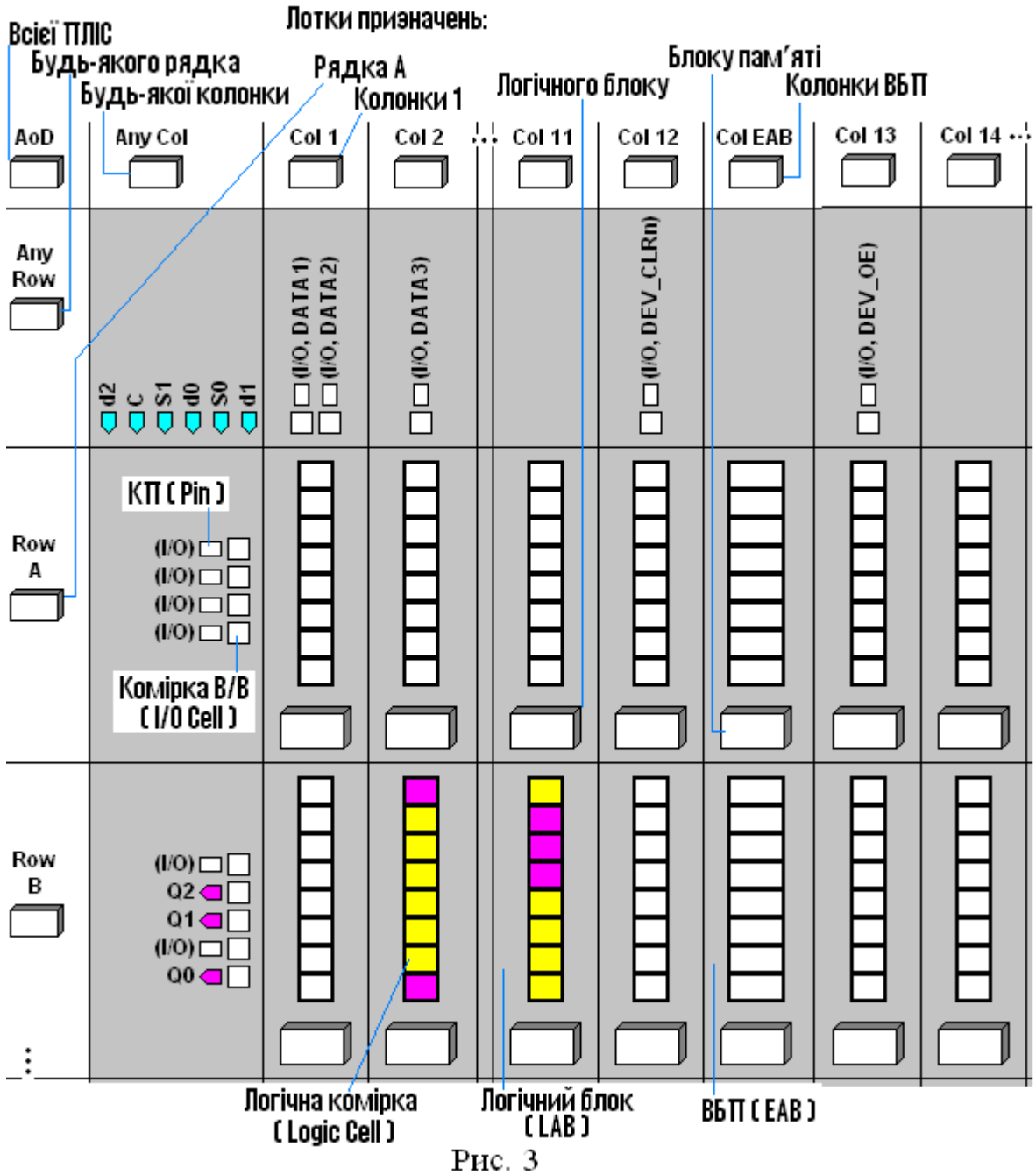


Рис. 3

Рисунок 10.3 – План розміщення (Floorplan) ПЛІС FLEX 10K фірми Altera

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 вводу-виводу КВВ (I/O Cell) біля кожного з них. Аналогічно блоку керування вводом-виводом БВВ 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).

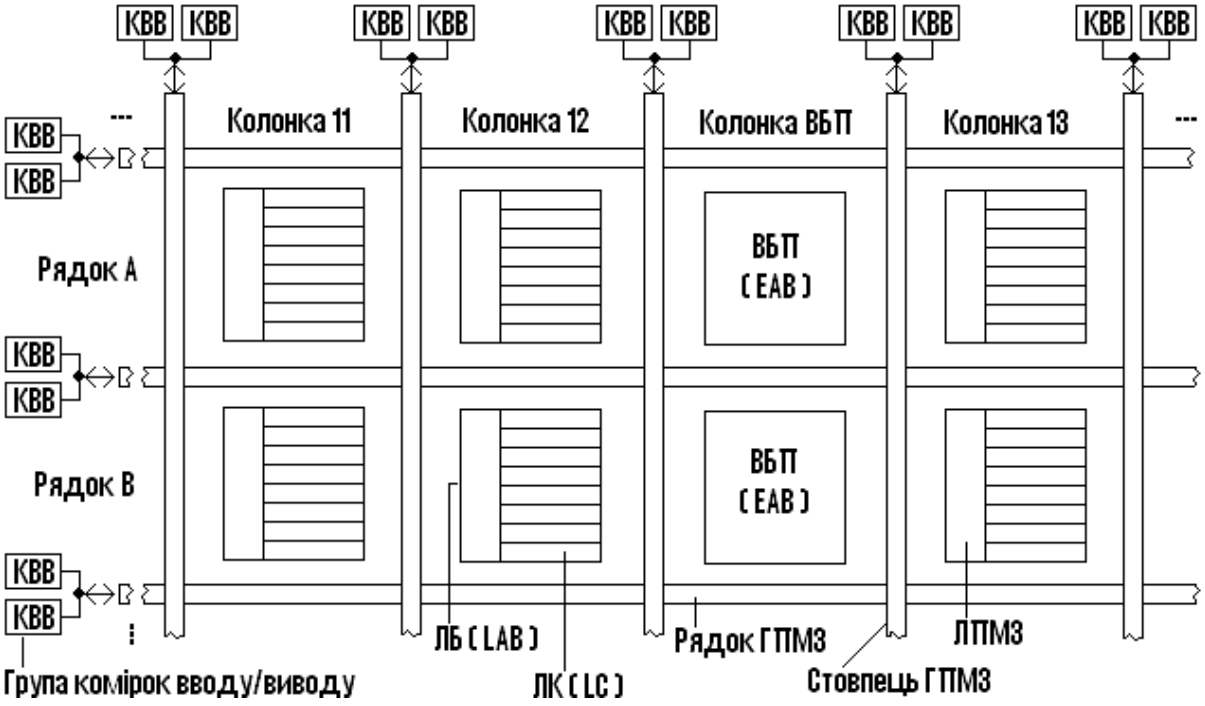


Рисунок 10.4 – Дворівнева система зв’язків блоків за допомогою глобальної програмованої матриці з’єднань ГПМЗ, поділеній на рядки і стовпці

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. 4), 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.

2 Довідкові відомості

Родини мікросхем подано в довідці пакету Help > Devices & Adapters. ПЛІС однієї родини однотипні і відрізняються кількістю ЛБ. Так, родини MAX7000 (рис. 10.5,а) найменшої складності містять два ЛБ, а найбільшої складності – 32 ЛБ.

MAX 7000 Devices	EPM7032AE EPM7032B EPM7032S EPM7064AE EPM7064B EPM7064S EPM7128AE EPM7128B EPM7128S EPM7160S EPM7192S EPM7256AE EPM7256B EPM7256S EPM7512AE EPM7512B

а

Device *Note (1)*	Package *Note (2)*	Macrocells	Registers	I/O *Note (11)*
EPM7128S	84L, 100Q, 100T, 160Q	128	128	64, 80, 80, 96

б

FLEX 10K Devices	EPF10K10 EPF10K10A EPF10K20 EPF10K30 EPF10K30A EPF10K30E EPF10K40 EPF10K50 EPF10K50S EPF10K50V EPF10K70 EPF10K100E EPF10K100A EPF10K130E EPF10K200S

в

Device *Note (1)*	Package *Note (2)*	EABs	Logic Elements	Registers	Memory Bits	I/O *Note (11)*	Configuration Device
EPF10K70	240R	10	3,744	4,096	18,432	183	EPC1, EPC2, EPC4, EPC8, EPC16

г

Рисунок 10.5 - Довідкові відомості родин мікросхем

Кожний ЛБ розглядуваного типу ПЛІС складається з 16 макрокомірок, а їх загальна кількість у мікросхемі визначає її назву. Найпростіші мікросхеми типу EPM7032(AE, B, S) містять у двох логічних блоках 32 макрокомірки, а найскладніші EPM7512(AE, B) – 512 МК у складі 32-х ЛБ.

Натисканням у списку ІС на родину потрібного типу легко вийти на файл довідки з даними про типи корпусів (Package), кількість макрокомірок (Macrocells) і приступних для користувача виводів (I/O) та інші дані (на рис. 10.5,б подано фрагмент даних для ІС типу EPM7128S родини

MAX7000S). Докладну інформацію про цоколівку виводів тощо можна отримати в п. 3 лабораторного завдання або на Altera website (у попередніх версіях Quartus II така інформація міститься в довідці).

У назвах ПЛІС родин типу FLEX (рис. 10.5,в) складність позначається в тисячах еквівалентних вентилів. Так у родині FLEX 10K мікросхема EPF10K10 найменшої складності має логічну ємність 10 тис. еквівалентних вентилів (72 ЛБ, 576 ЛК, 3 ВБП обсягом 2048 біт кожний), а ІС цієї родини EPF10K250 найбільшої складності має логічну ємність 250 тис. еквівалентних вентилів (1520 ЛБ, 12160 ЛК, 20 ВБП).

Такі ПЛІС придатні для побудови складних пристроїв обробки інформації. У довідці для ІС такого типу (на рис. 10.5,в подано фрагмент даних для ІС типу EPF10K70 родини FLEX 10K) макрокомірки називаються «логічними елементами» (Logic Elements) та подаються також відомості щодо вбудованих блоків пам'яті (EABs) і типи конфігураційних мікросхем (Configuration Device), призначених для зберігання інформації про конфігурацію ІС, яка по ввімкнені джерела живлення автоматично завантажується до ПЛІС.

КОНТРОЛЬНІ ЗАПИТАННЯ

1 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.

2 Чим відрізняються сучасні програмовані ПЛІС від мікропроцесорних ІС, робота яких основана на виконанні програми? Порівняйте можливості таких ІС щодо складності виконуваних функцій і швидкодії.

3 Схарактеризуйте режими реконфігурування структури ПЛІС типу EPLD з електричним стиранням. Які елементи пам'яті конфігурації застосовуються для таких ПЛІС? Що розуміється в САПР під термінами „програмування" і „конфігурування"?

4 Зобразіть схему ПЛМ для реалізації одного з варіантів функції завдання 2 (див. додаток до Лаб. роботи №2).

5 Поясніть архітектуру ПЛІС типу EPLD: призначення сигналів і складників та їх взаємодію.

6 Поясніть поняття інтерфейсу JTAG і схеми програмування однієї ПЛІС та їх ланцюжка.

ЛАБОРАТОРНЕ ЗАВДАННЯ

1 Спроектувати *подільник частоти* із заданими коефіцієнтом поділу – модулем *M* (згідно з варіантом завдання 10).

1.1 Побудувати *безвентильний подільник* частоти з використанням JK-тригерів за *графічного* (.bdf) введення проекту 10XXpod, виконати компіляцію і моделювання (.vwf), 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 1000pod.bdf, vwf.

✍ *Примітка:* Інверсні асинхронні входи передустановлення SN[4..0] і скидання RN (схема 2) призначені для випробування лічильника на самовідновність і побудови повного перемикального графа. Для цього інверсним кодом, наприклад, SN[4..0] = 00_{16} (як подано на часових діаграмах) можна асинхронно записати до всіх розрядів лічильника одиниці і по моделюванні імітатором простежити шлях повернення до робочого циклу: 31 $\rightarrow$ 14 $\rightarrow$ 21 $\rightarrow$ ...; відтак записати з метою наочності інший стан поза робочим циклом в іншому місці (або в цьому ж місці і знов змоделювати) та простежити шлях і так само повторити цю процедуру для потрібної кількості станів.

1.2 Створити подільник частоти за *текстового* (.tdf) введення проекту 10XXpod_sm як *скінченний автомат* (State Machine) за власним вибором оператора логічної секції для позначення його переходів (див. Лаб. робота №8, п.4) на кшталт файла 10XXpod.bdf та дослідити його. Користуючись часовими діаграмами (.vwf), визначити тип пристрою, утворений автоматичним синтезатором як подільник частоти.

***Приклад**:* подільник частоти із коефіцієнтом поділу *M* = 11 – файли 1000pod_sm.tdf, vwf.

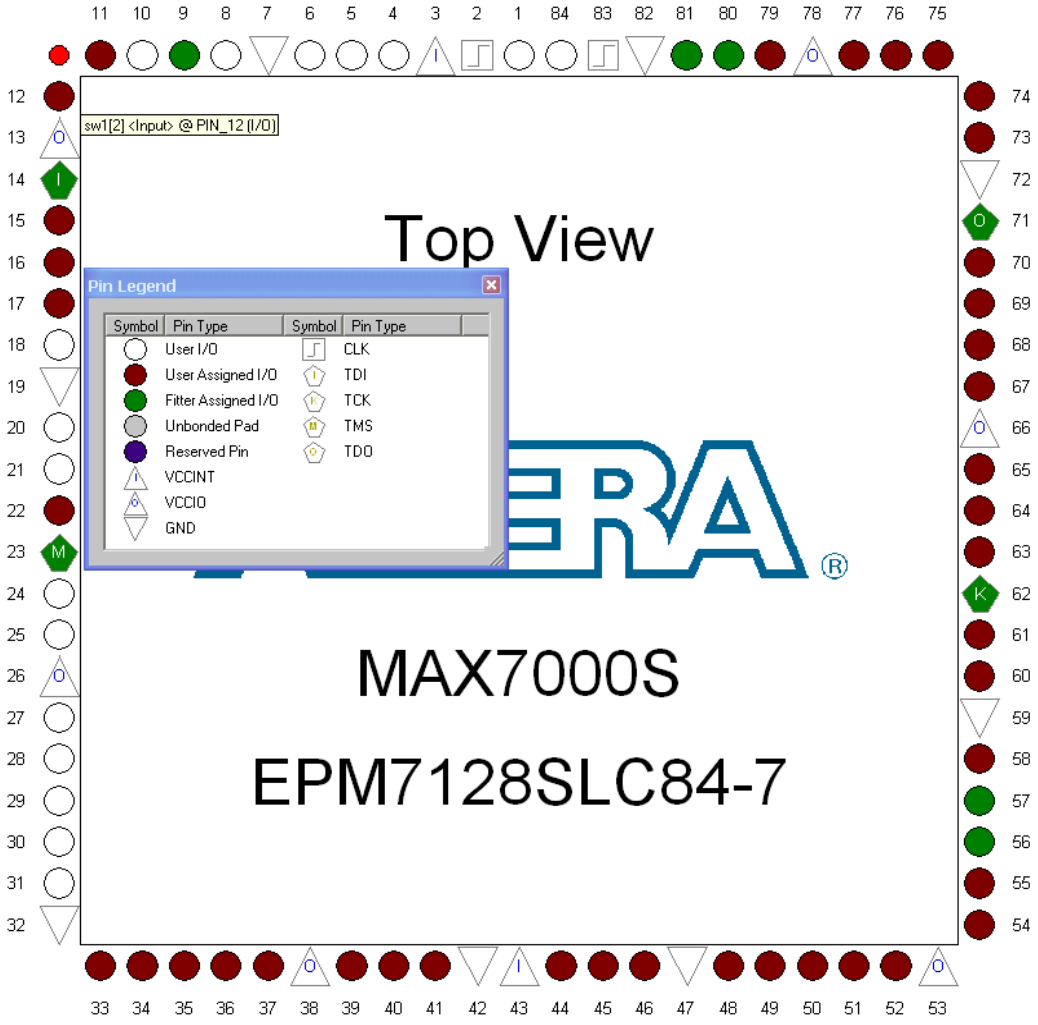
2 Створити проект 10XXPODIL_MAX лабораторного макету для експериментального дослідження ЦПП на мікросхемі MAX7000S на прикладі заданого варіанту подільника частоти за п. 1.1. (див. як взірець файл 1000PODIL_MAX.bdf – рис. Д8 у додатках).

Для цього створити символ подільника за п. 1.1 (як приклад див. файли 1000pod11.bdf, bsf), вставити його до графічного файла, який доповнити перетворювачем двійкового коду в ДДК і дешифраторами 7-сегментного коду, потрібними для індикації вихідного коду, а також іншими компонентами (у прикладі кількість тактів вхідного сигналу в періоді вихідного сигналу, тобто коефіцієнт поділу можна простежити за допомогою світлодіода VD1). Імпортувати призначення виводів мікросхеми з файлу ../4lab/MAX_pin та скоригувати їх. Виконати компіляцію і функціональне моделювання з метою переконатися в правильності проектування. Створити програмувальний CDF-файл і виконати програмування мікросхеми. Розробити методику та виконати експериментальні дослідження пристрою на запрограмованій ІС. Порівняти результати експериментальних досліджень з даними проектних файлів, зробити висновки.

Приклад: подільник частоти із коефіцієнтом поділу $M = 11$ – файли 1000PODIL_MAX.bdf, .vwf, .cdf.

3 Ознайомитися з основними функціями редактора фізичного розміщення Floorplan Editor.

3.1 У разі потреби, активізувати належний проект 10XXPODIL_MAX (ввести його назву до рядка заголовку головного вікна), піктограмою горизонтальної панелі Pin Planner (або з меню Assignments > Pin Planner) відкрити вікно із цоколівкою виводів і розгорнути його, зачинивши вікна зі списками. Налаштувати інструментальну палітру (меню Tools > Customize Pin Planner): встановити великі облямовані кнопки та вилучити зайві інструменти, зокрема, які є на горизонтальній панелі. Натиснути інструмент вертикальної палітри Pin Legend та ознайомитися з позначенням виводів і розташуванням їх на цоколівці мікросхеми. Аби з'ясувати призначення виводу, досить підвести до нього покажчик і зчитати дані в рамці, що з'являється. З меню View > All Pins List відкрити вікно зі списком виводів, виділити вивід на цоколівці та зчитати дані в цьому вікні чи, навпаки, виділити вивід у вікні і продивитися його розташування на цоколівці. Так само з меню View > Groups List відкрити вікно зі списком груп і аналогічним чином з'ясувати зв'язок між списком і цоколівкою.

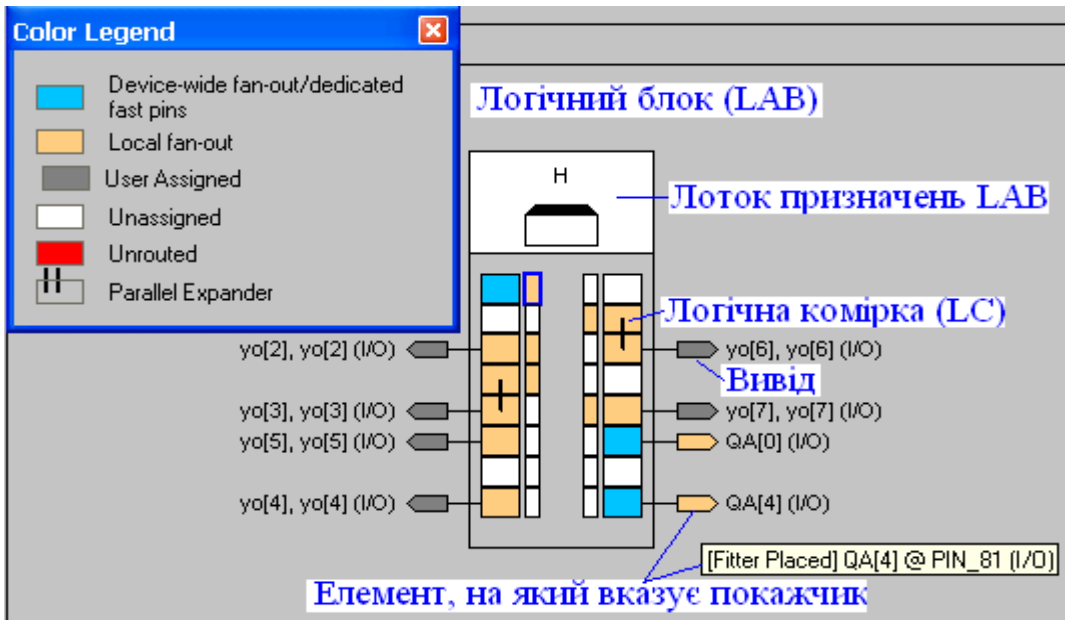


✍ *Примітки:*

1) У зазначеному вікні All Pins можна також відредагувати призначення виводів: клацнути будь-де в рядку потрібного виводу правою кнопкою миші і вибрати в контекстному меню Node Properties для зміни номера виводу або Pin Properties для зміни імені виводу. Відтак в однойменному діалоговому вікні (Node Properties або Pin Properties) в графі Location або Node name прокруткою встановити або надрукувати відповідно номер виводу чи його ім'я. Призначити або відредагувати номер виводу можна і безпосередньо подвійним клацанням у стовпці Location і прокруткою списку номерів. Так само можна відредагувати призначення виводів і у вікні Groups.

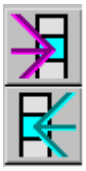
2) У зазначених діалогових вікнах (Node Properties або Pin Properties) у графі I/O standard прокруткою можна вибрати також стандарт входу/виходу. За умовчанням (default) для родин MAX7000S та FLEX10K встановлено TTL I/O standard – звичайний стандарт ТТЛ з напругою живлення 5 *В* (зокрема, через це зазначені родини є популярні серед масових користувачів). Проте для всіх родин ІС, що підтримуються пакетом Quartus II, можуть використовуватися також низьковольтні (LV) стандарти напруги 3,3 *В* стосовно вхідних/вихідних сигналів (VCCIO) і напруги 3,3 *В*/2,5 *В* стосовно вхідних/вихідних буферів. Такі стандарти для елементної бази ТТЛ (TTL) і КМОНТЛ (CMOS) позначаються в САПР відповідно LVTTL I/O standard та LVCMOS I/O standard.

3.2 Піктограмою Timing Closure Floorplan (або з меню Assignments > Timing Closure Floorplan) відкрити вікно редактора фізичного розміщення Floorplan Editor і розгорнути його. Налаштувати інструментальну палітру (меню Tools > Customize Timing Closure Floorplan): встановити великі облямовані кнопки та вилучити зайві інструменти, зокрема, які є на горизонтальній панелі. Натиснути інструменти вертикальної палітри Show User Assignment (показ призначень користувача), Show Fitter Placements (показ розміщень трасувальника) і View Color Legend (перегляд кольорових позначень) та ознайомитися з ви-

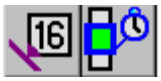


глядом внутрішньої структури ПЛІС, логічними блоками (LAB), їх комірками (LC) і умовними позначеннями елементів.

Примітки:



1) Для перегляду логічних зв'язків між елементами (між логічними комірками або комірками і зовнішніми виводами) слід виділити один або декілька елементів (клацнути при утримуваній клавіші Ctrl), натиснути інструмент палітри Show Node Fan-In (показати вхідні зв'язки кіл), або Show Node Fan-Out (показати вихідні зв'язки кіл) і спостерігати логічні зв'язки у вигляді кольорових ліній.



2) Для перегляду кількості з'єднань між виділеними елементами або затримок між ними слід натиснути інструмент Show Connection Count або Show Routing Delays.

3) Для перегляду логічних рівнянь і їх зв'язок з виділеними елементами плану мікросхеми слід натиснути інструмент View Equations.

4) На плані мікросхеми можна також відредагувати розміщення виводів і інших призначень, після чого перекомпілювати проект.

4 Створити проект 10XXPODIL_ FLEX лабораторного макету для експериментального дослідження ЦПП на мікросхемі FLEX10K на прикладі заданого варіанту подільника частоти за п. 1.1. (див. як взірець файл 1000PODIL_FLEX.bdf – рис. Д9 у додатках).

4.1 Вставити до графічного файлу 10XXPODIL_ FLEX.bdf копію лабораторного макету 10XXPODIL_MAX.bdf, вилучити світлодіоди, скоригувати імена вхідних портів sw[] (якщо якійсь вивід відокремлено від групи, його слід іменувати без квадратних дужок, наприклад, замість sw[8] позначити sw8), відповідно до цих портів скоригувати також трасувальні таблиці блоку подільника частоти та ввести до складу проекту всі потрібні файли. Розвести виводи мікросхеми на друкованій платі відповідно до файлу ../10lab/ROZWED_FLEX.bdf або імпортувати їх призначення з файлу ../10lab/FLEX_pin.bdf і скоригувати відповідно до портів проекту.

4.2 Виконати компіляцію і функціональне моделювання.

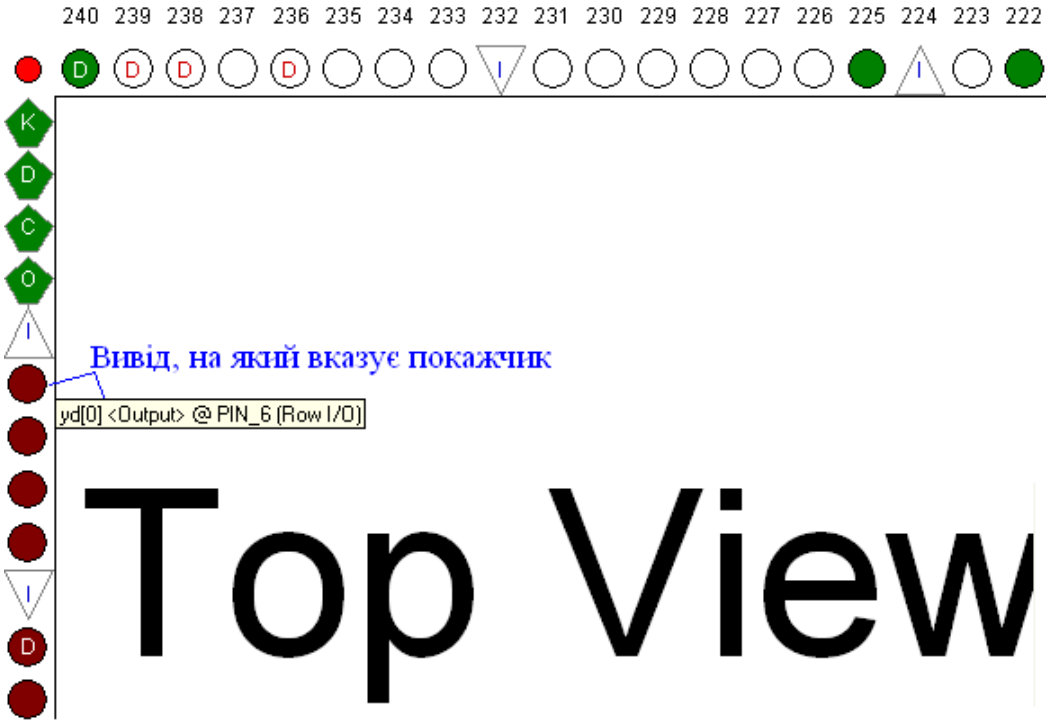
4.3 Створити програмувальний CDF-файл і виконати програмування мікросхеми. Виконати фізичне конфігурування мікросхеми (джампери встановити згідно з файлом ../10lab/ FLEX_pin.bdf).

4.4 Розробити методику та виконати експериментальні дослідження пристрою. Порівняти результати експериментальних досліджень з даними проектних файлів, зробити висновки.

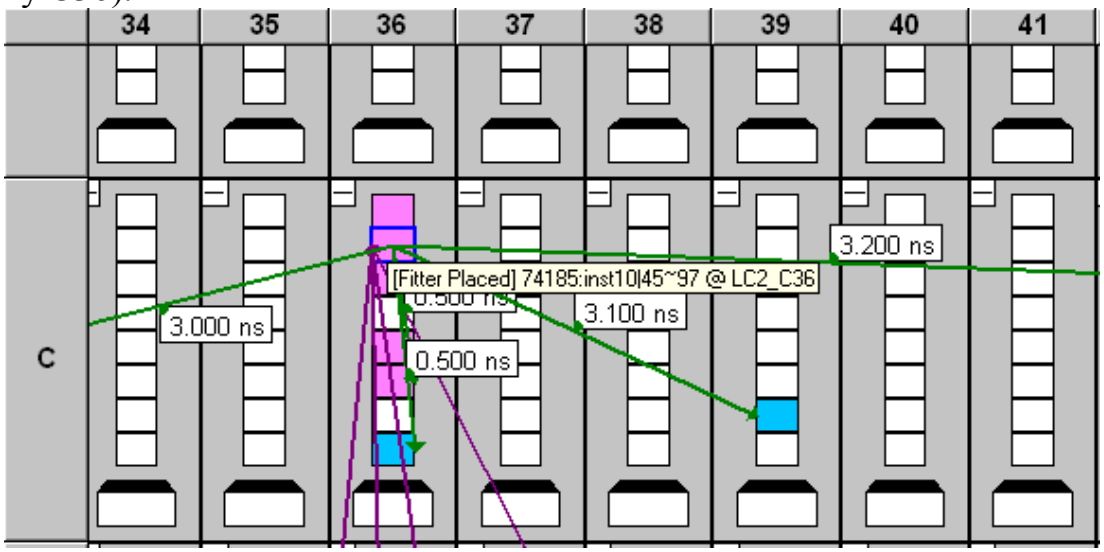
Приклад: подільник частоти із коефіцієнтом поділу $M = 11$ – файли 1000PODIL_FLEX.bdf, vwf.

5 Переглянути особливості фізичного розміщення мікросхеми на плані Floorplan Editor.

5.1 Аналогічно п. 3.1 ознайомитися із цоколівкою виводів, користуючись інструментами регулювання масштабу.



5.2 Аналогічно п. 3.2 ознайомитися з виглядом внутрішньої структури ПЛІС (на фрагменті подано затримки зв'язків комірки LC2 логічного блоку C36).



ЛІТЕРАТУРА

1. Кофанов В.Л. **Математичні та схемотехнічні основи цифрових пристроїв**: Навч. посібник. – Вінниця: УНІВЕРСУМ-Вінниця, 2005. – 165 с. – *Теоретичні відомості щодо логічних функцій, методика їх мінімізації і проектування на рівні логічних елементів, приклади.*

2. Кофанов В.Л., Осадчук О.В., Гаврілов Д.В.. **Лабораторний практикум з дослідження цифрових пристроїв на основі САПР MAX+PLUS II:** Навч. посібник. – Вінниця: УНІВЕРСУМ-Вінниця, 2007. – …? с. (друкується). – *Стислі теоретичні відомості, відомості щодо ВІС програмованої структури, методика використання САПР MAX+PLUS II.*

3. **Справочник по цифровой схемотехнике** / В.И. Зубчук, В.П. Сигорский, А.Н. Шкуро – К.: Техніка, 1990. – 448 с. – *Стислі теоретичні відомості і довідкові матеріали щодо ІС жорсткої структури.*

4. Угрюмов Е.П. **Цифровая схемотехника:** Учеб. пособие. – СПб.: БХВ – Петербург, 2002. – 528 с. – *Відомості щодо ВІС програмованої структури.*

5. Кофанов В.Л. **Базовые элементы цифровых интегральных микросхем:** Учеб. пособие. – К.: УМК ВО, 1988. – 116 с. – *Відомості щодо ІС жорсткої структури, їх класифікації і параметрів.*

6. Ерофеев Ю.Н. **Импульсные устройства**: Учеб. пособие для вузов по спец. «Радиотехника». – М.: Высшая шк., 1989. – 527 с. – *Теоретичні відомості щодо основ ЦП на ІС жорсткої структури.*

7. Калабеков Б.А. **Цифровые устройства и микропроцессорные системы**: Учебник для техникумов связи. – М.: Горячая линия – Телеком, 2002. – 336 с. – *Популярне викладення основ ЦП на ІС жорсткої структури.*

8. **University Program UP2 Education Kit.** – Altera Corporation, v. 3.1, 2004. – *Технічний опис лабораторної плати UP2.*

9. Кофанов В. Л. **Проектування цифрових пристроїв**: Навч. посібник (електронний варіант). – *Основи проектування ЦП на типових структурах (ЦКП і ЦПП) та з використанням САПР Quartus II, приклади.*

ДОДАТКИ

ВАРІАНТИ ЗАВДАННЯ 1

Наведіть для кожної схеми (*а...і*) заданого рисунку значення константи або умовне позначення еквівалентного логічного елемента, якщо елемент *А* виконує таку логічну функцію (далі наводиться варіант – рисунок – функція): 1) Д1 – АБО, 2) Д1 – І, 3) Д1 – Виключне АБО, 4) Д1 – Заборона, 5) Д1 – АБО-НЕ, 6) Д1 – І-НЕ, 7) Д1 – Виключне АБО-НЕ, 8) Д1 – Імплікація, 9) Д2 – АБО, 10) Д2 – І, 11) Д2 – Виключне АБО, 12) Д2 – Заборона, 13) Д2 – АБО-НЕ, 14) Д2 – І-НЕ, 15) Д2 – Виключне АБО-НЕ, 16) Д2 – Імплікація.

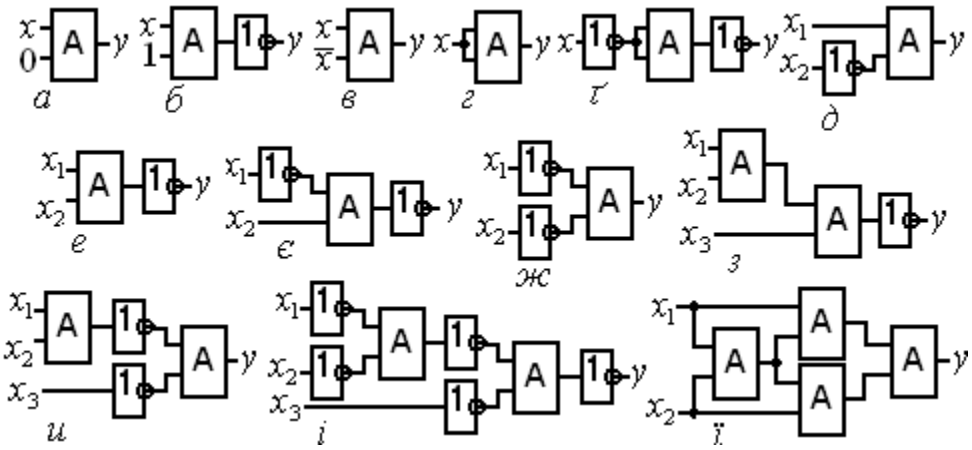


Рисунок Д1

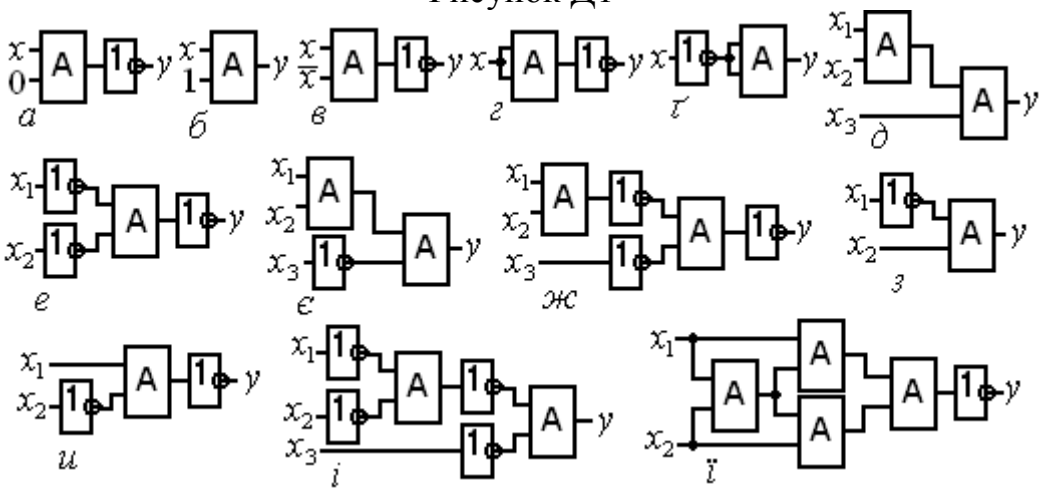


Рисунок Д2

Приклад. Дано: $A = \overline{x_1 \oplus x_2}$.

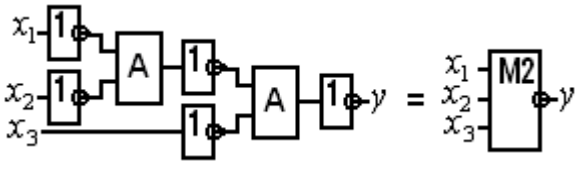


Рисунок Д3

Підставляючи змінні до схеми, маємо

$$y = \overline{\overline{x_1} \oplus \overline{x_2} \oplus \overline{x_3}} = \overline{x_1} \oplus \overline{x_2} \oplus \overline{x_3} =$$

$$= \overline{x_1 \oplus x_2 \oplus x_3} = \overline{M_2},$$

де M_2 – суматор за модулем два (рис. Д3).

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; *ґ)* навести схеми за п. *в*, *г*.

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) Спроектувати на основі двійкового дешифратора перетворювач (дешифратор) до семисегментного коду для індикації шістнадцяткових цифр:

1) A, b, C, d; 2) A, b, C, d, E; 3) A, b, C, d, E, F; 4) A, C, E; 5) C, d, E, F; 6) b, C, d, E, F; 7)A, C, E, F; 8) A, E, F; 9) A, C, d, E, F; 10) C, d, E, F.

б) Розробити схему передачі даних заданої розрядності на основі шифратора і дешифратора (як взірець див. рис. Д1).

Варіант*	1	2	3	4	5	6	7	8	9	10
Розрядність, n	9	8	7	6	11	5	12	15	13	14

***)** При n > 10 виключити дослідження DC 7-seg. або (та) ЦКП.

ВАРІАНТИ ЗАВДАННЯ 4

а) Спроектувати ЦКП для реалізації логічної функції, заданої згідно з варіантом завдання 2 (до лабораторної роботи №2), на мультиплексорах різної розрядності та вибрати оптимальний варіант.

б) Розробити схему передачі даних заданої розрядності на основі мультиплексора і демультиплексора (як взірець див. рис. Д1).

Варіант*	1	2	3	4	5	6	7	8	9	10
Розрядність, n	5	4	7	6	11	10	12	9	13	14

***)** При n > 8 доповнити формувач адреси виходом a[3].

ВАРІАНТИ ЗАВДАННЯ 5

1) Записати апаратною мовою AHDL логічну функцію завдання 2 з Лаб. роботи №2 (первісний вираз, але без операндів заборони і імплікації), розбити її на частини з метою введення логічних рівнянь за допомогою оператора Node; *приклад* див. на стор. 6.

2) Відновити графічний файл двійкового дешифратора згідно із завданням 3 (за п. 2.1 Лаб. роботи №3) і скласти його таблицю відповідності з метою введення оператора цієї таблиці.

3) Відновити символ демультиплексора згідно із завданням 4 (за п. 5.1 Лаб. роботи №4) і скласти його таблицю відповідності з метою введення умовного оператора.

4) Відновити графічний файл для реалізації мультиплексора розрядністю за власним вибором згідно із завданням 4 (за п. 2.1 Лаб. роботи №4) і скласти його таблицю відповідності з метою введення оператора вибору.

5) Відновити графічний файл, що реалізує спеціальний дешифратор 7-сегментного коду для відображення заданих шістнадцяткових цифр на основі макрофункції двійкового дешифратора (за п. 2.3 Лаб. роботи №3) з метою включення до проекту макрофункції і створення текстового файлу.

6) Відновити графічний файл, що реалізує спеціальний дешифратор 7-сегментного коду для відображення заданих шістнадцяткових цифр на основі символу двійкового дешифратора (за п. 2.5 Лаб. роботи №3), з метою включення до проекту символу і створення текстового файлу.

7) Відновити графічний файл, що реалізує задану логічну функцію на основі створеного різновиду мегафункції (за п. 4.1 ... 4.3 Лаб. роботи №4), з метою включення до проекту мегафункції і створення текстового файлу.

ВАРІАНТИ ЗАВДАННЯ 6

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 7

Розробити лінію передачі даних заданої розрядності з фіксацією вихідного коду на тригерах (як взірець див. рис. Д1).

Варіант*	1	2	3	4	5	6	7	8	9	10
Розрядність, n	5	4	8	6	11	10	12	9	13	14

***)** При n > 8 доповнити формувач адреси виходом a[3].

ВАРІАНТИ ЗАВДАННЯ 8

Спроектувати на основі регістра зсуву ЦПП (далі наводиться тип пристрою – варіант – дані у вигляді періоду кодової послідовності M або модулю N):

ГКП реверсивний – 1) N = (0000101); 5) M = 5; 9) N = (00101);

РІР – 2) N = (01111); 6) M = 7; 10) M = 9;

ГКП– 3) M = 7; 7) N = (01011); 11) N = (0001101);
РІР реверсивний – 4) M = 5; 8) N = (00011); 12) M = 6.

ВАРІАНТИ ЗАВДАННЯ 9

Спроектувати згідно з варіантом:

а) недвійковий лічильник із заданим модулем *M* і природним 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 природним 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.

ВАРІАНТИ ЗАВДАННЯ 10

Спроектувати згідно з варіантом 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.

***Приклад**:* див. Лаб. робота №9, п. 4.2.

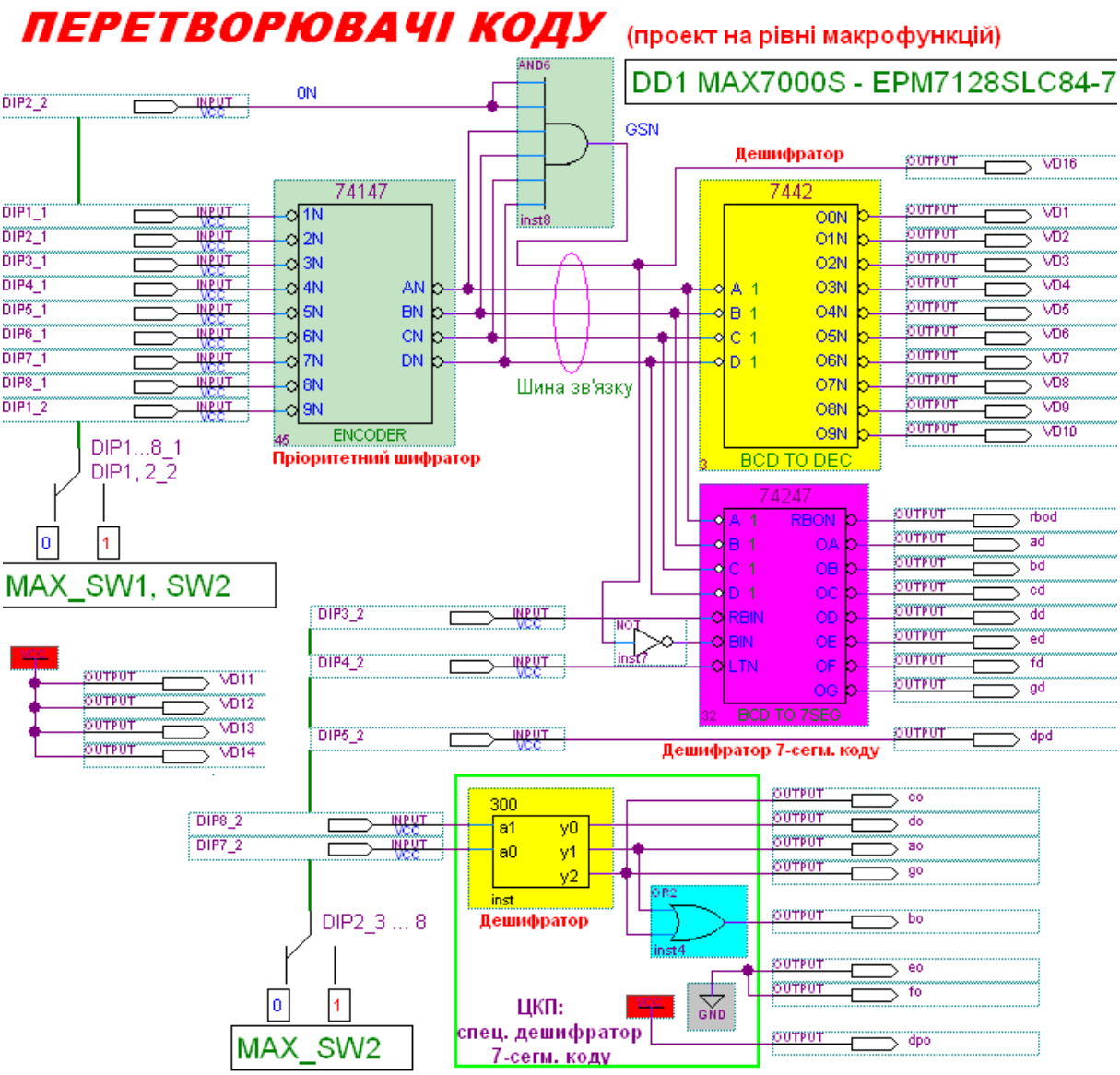

ПЕРЕТВОРЮВАЧІ КОДУ
(проект на рівні макрофункцій)
DD1 MAX7000S - EPM7128SLC84-7
Дешифратор
Пріоритетний шифратор
Шина зв'язку
Дешифратор 7-сегм. коду
ЦКП:
спец. дешифратор
7-сегм. коду
MAX_SW1, SW2
MAX_SW2
DIP1...8_1
DIP1, 2_2
DIP2_3 ... 8
ENCODER
BCD TO DEC
BCD TO 7SEG
74147
7442
74247
300
Дешифратор

Рисунок Д1

ЦИФРОВІ КОМУТАТОРИ (проект на рівні мегафункцій)

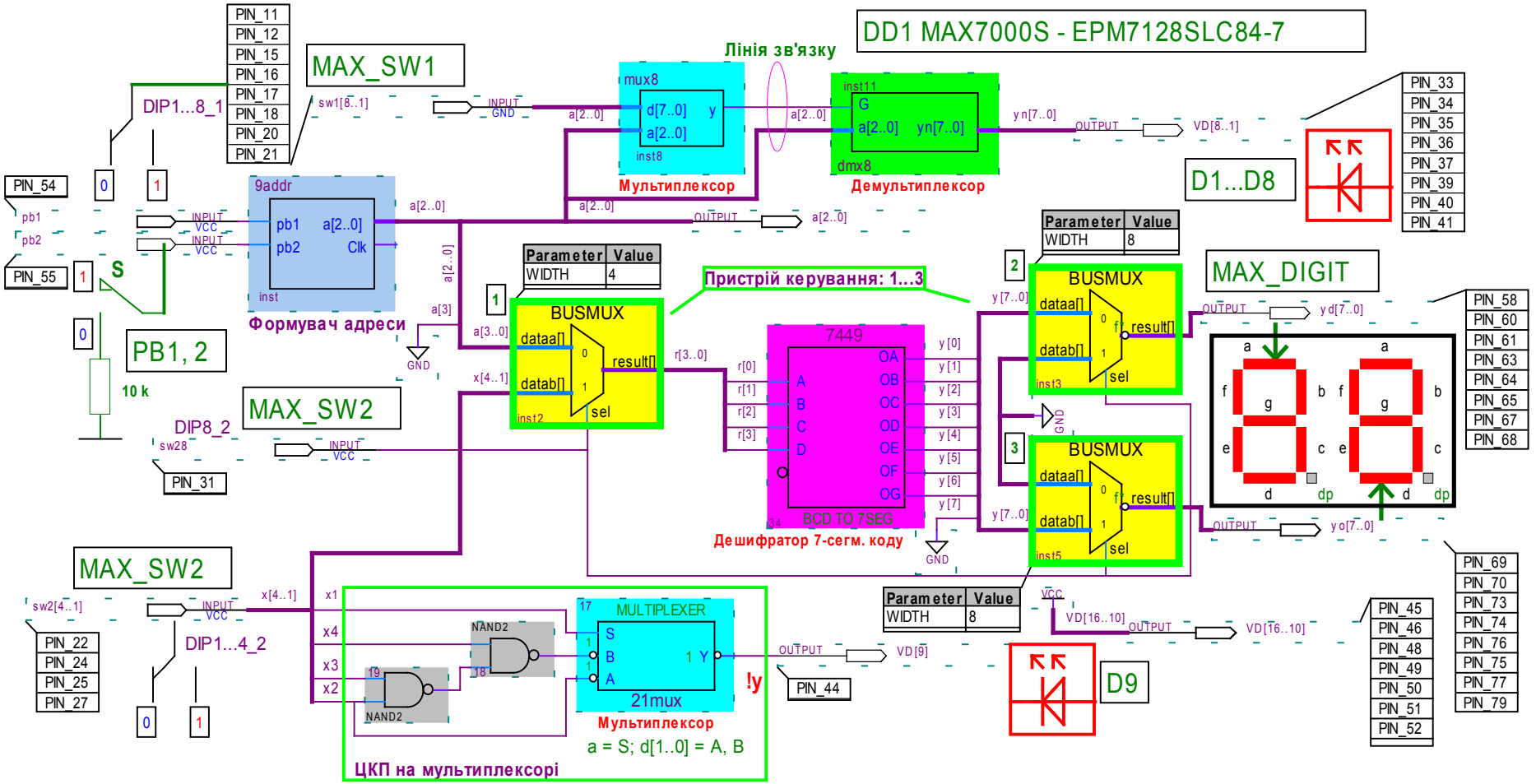


Рисунок Д2

ЦКП на MUX [Ілюстрація для текстового проекту 500СКР]

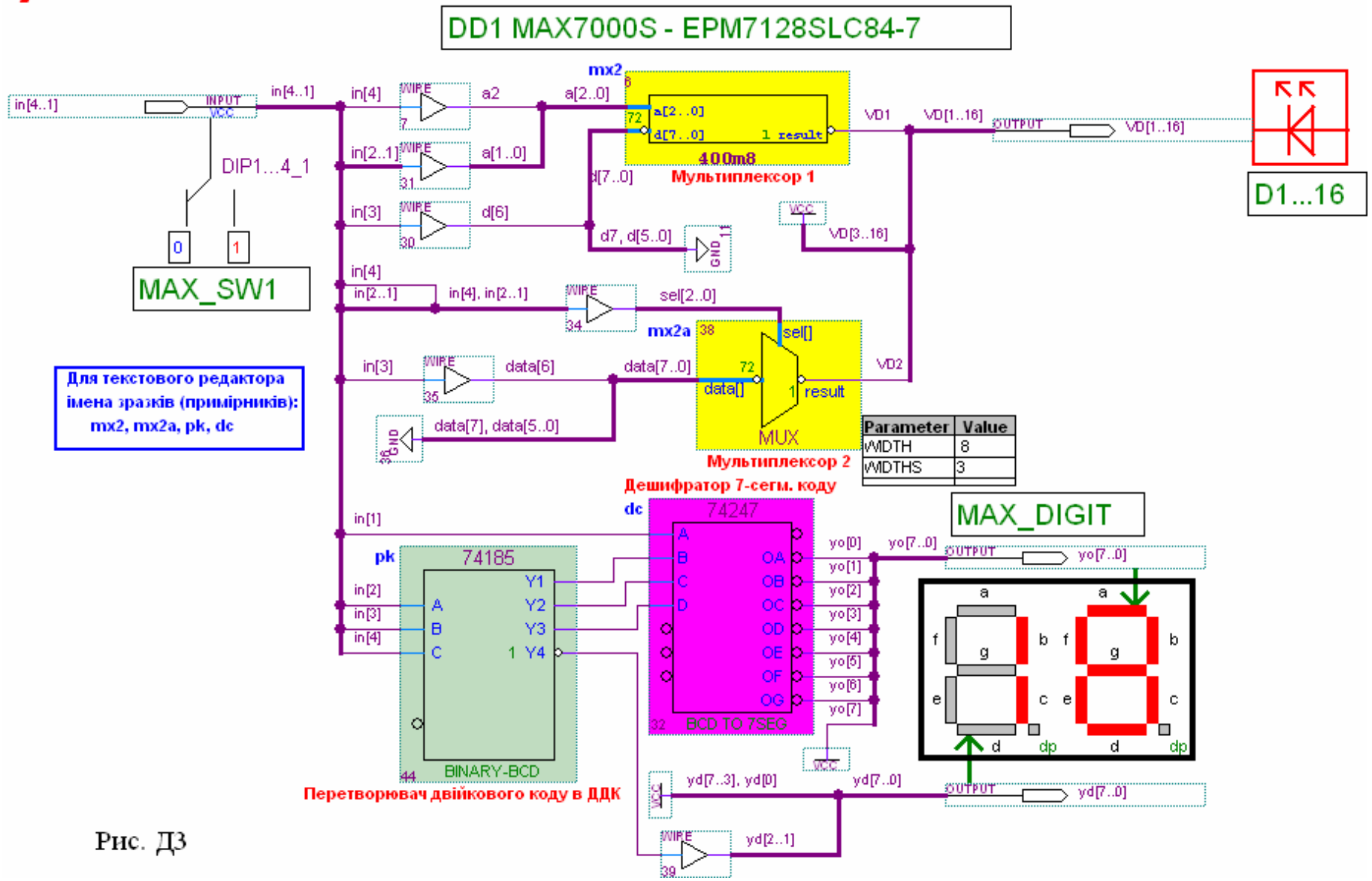


Рис. Д3

ПОРОГОВИЙ ЕЛЕМЕНТ 5 з 7 (проект на рівні блок-схеми)

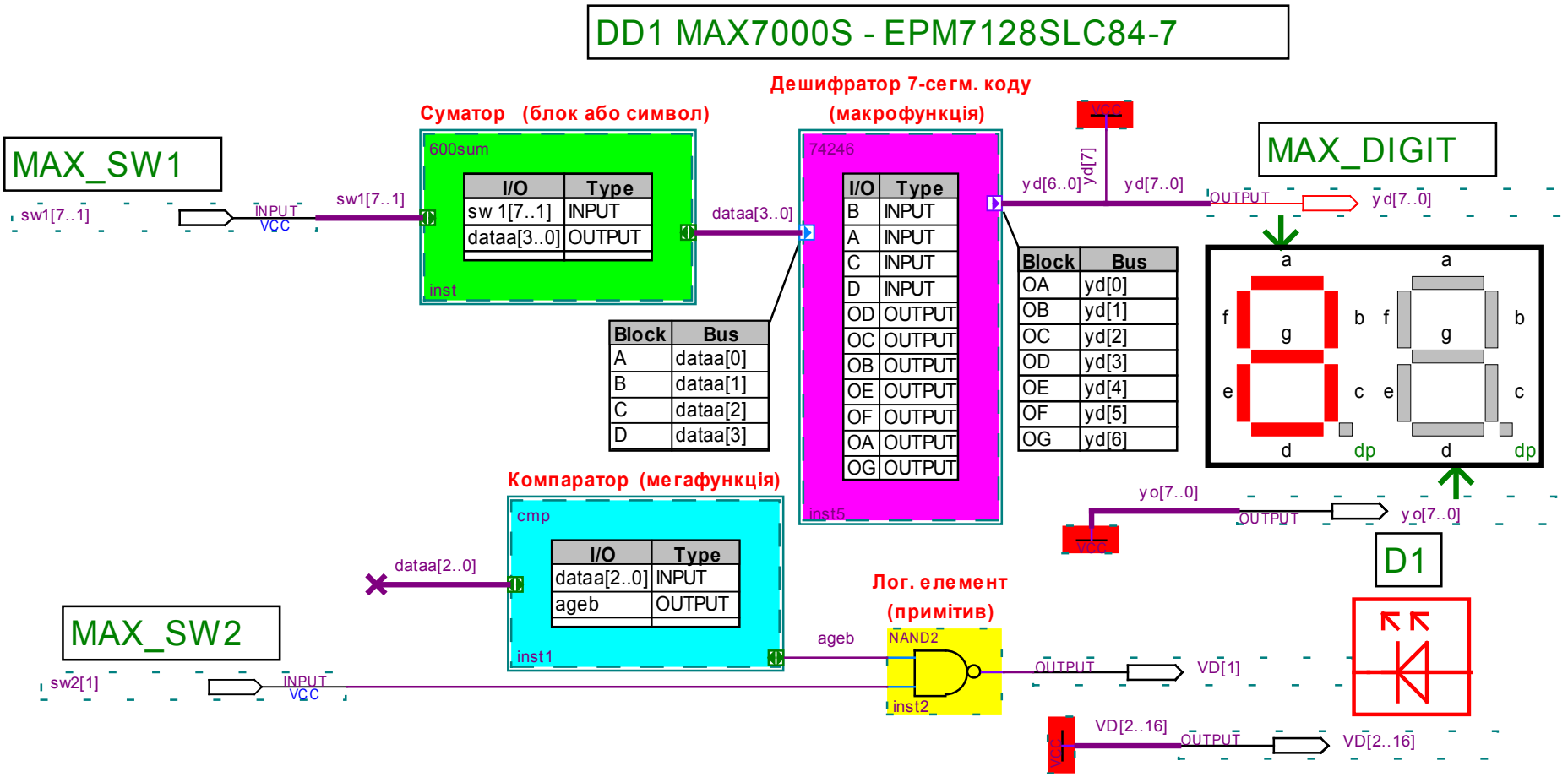


Рисунок Д4

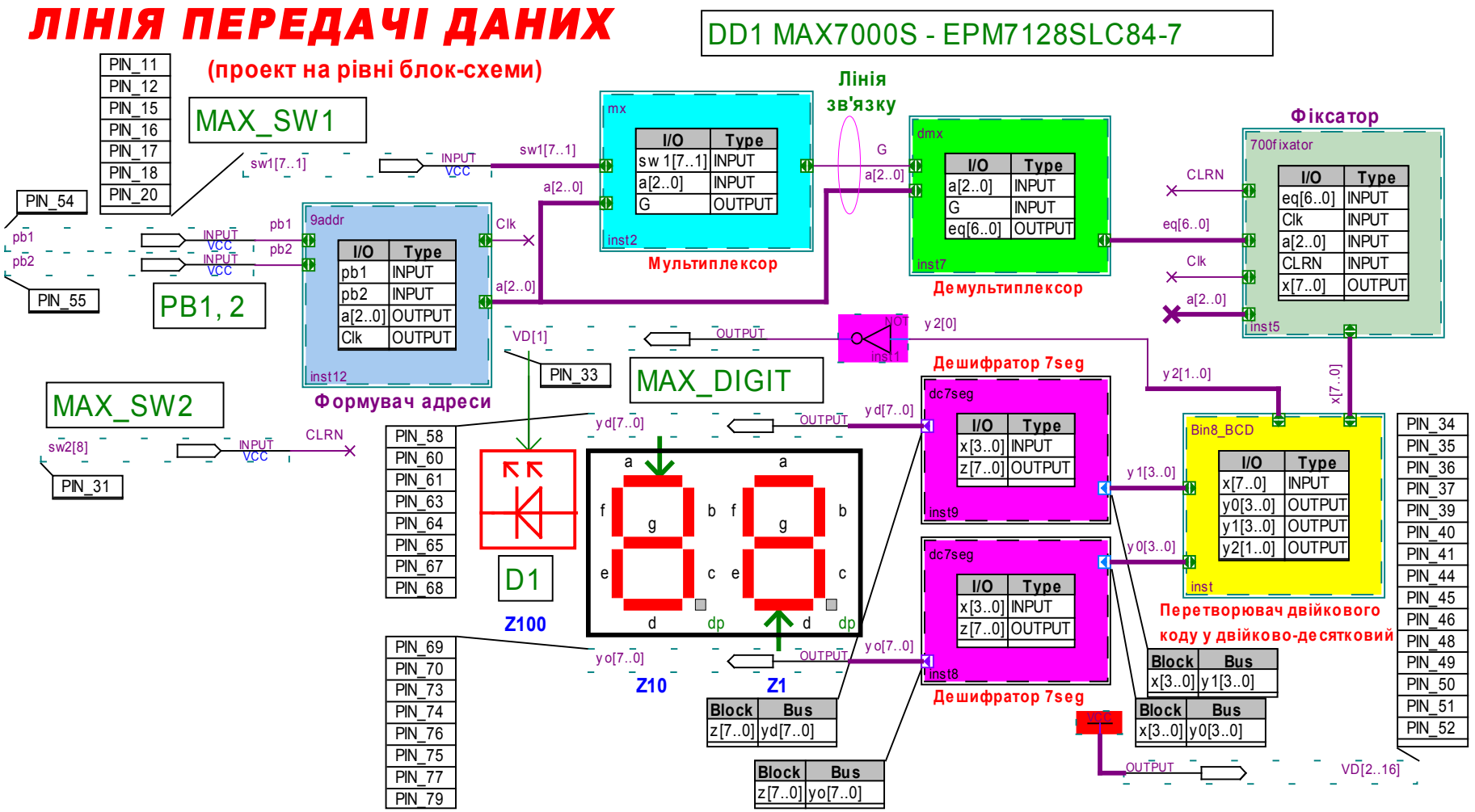

ЛІНІЯ ПЕРЕДАЧІ ДАНИХ
(проект на рівні блок-схеми)
DD1 MAX7000S - EPM7128SLC84-7
MAX_SW1
MAX_SW2
PB1, 2
MAX_DIGIT
Лінія зв'язку
Мультиплексор
Демультиплексор
Фіксатор
Формувач адреси
Дешифратор 7seg
Дешифратор 7seg
Перетворювач двійкового коду у двійково-десятковий
D1
Z100
Z10
Z1
PIN_11
PIN_12
PIN_15
PIN_16
PIN_17
PIN_18
PIN_20
PIN_54
PIN_55
PIN_31
PIN_33
PIN_58
PIN_60
PIN_61
PIN_63
PIN_64
PIN_65
PIN_67
PIN_68
PIN_69
PIN_70
PIN_73
PIN_74
PIN_76
PIN_75
PIN_77
PIN_79
PIN_34
PIN_35
PIN_36
PIN_37
PIN_39
PIN_40
PIN_41
PIN_44
PIN_45
PIN_46
PIN_48
PIN_49
PIN_50
PIN_51
PIN_52
VD[1]
VD[2..16]
VD[0..7]
mx
dmx
700fixator
9addr
dc7seg
Bin8_BCD
inst2
inst7
inst5
inst12
inst9
inst8
inst11
inst
I/O Type
sw 1[7..1] INPUT
a[2..0] INPUT
G OUTPUT
a[2..0] INPUT
G INPUT
eq[6..0] OUTPUT
eq[6..0] INPUT
Clk INPUT
a[2..0] INPUT
CLRN INPUT
x[7..0] OUTPUT
pb1 INPUT
pb2 INPUT
a[2..0] OUTPUT
Clk OUTPUT
x[3..0] INPUT
z[7..0] OUTPUT
x[7..0] INPUT
y0[3..0] OUTPUT
y1[3..0] OUTPUT
y2[1..0] OUTPUT
Block Bus
z[7..0] yd[7..0]
z[7..0] yo[7..0]
x[3..0] y1[3..0]
x[3..0] y0[3..0]

Рисунок Д5

ГЕНЕРАТОР КОДОВИХ ПОСЛІДОВНОСТЕЙ

(проект на основі регістра зсуву)

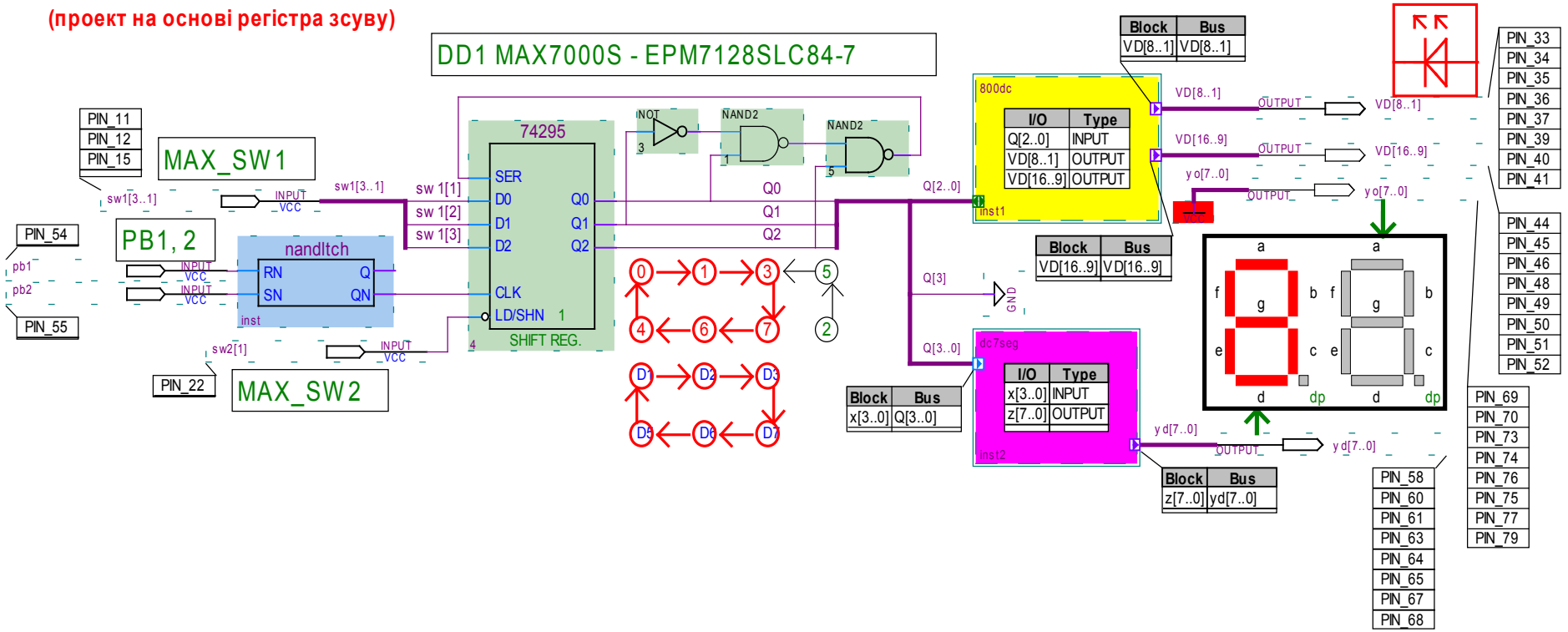


Рисунок Д6

ПРОГРАМОВАНИЙ ЛІЧИЛЬНИК (М=5, 7)

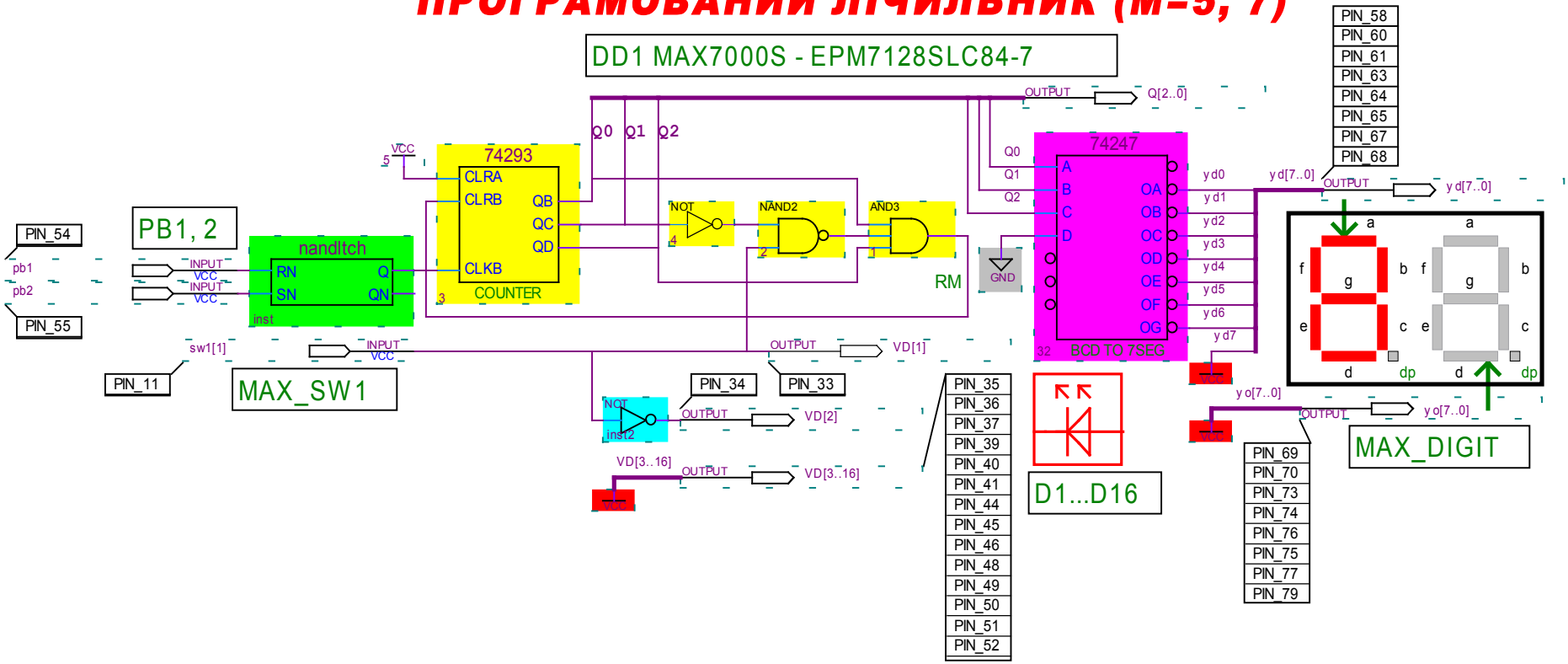
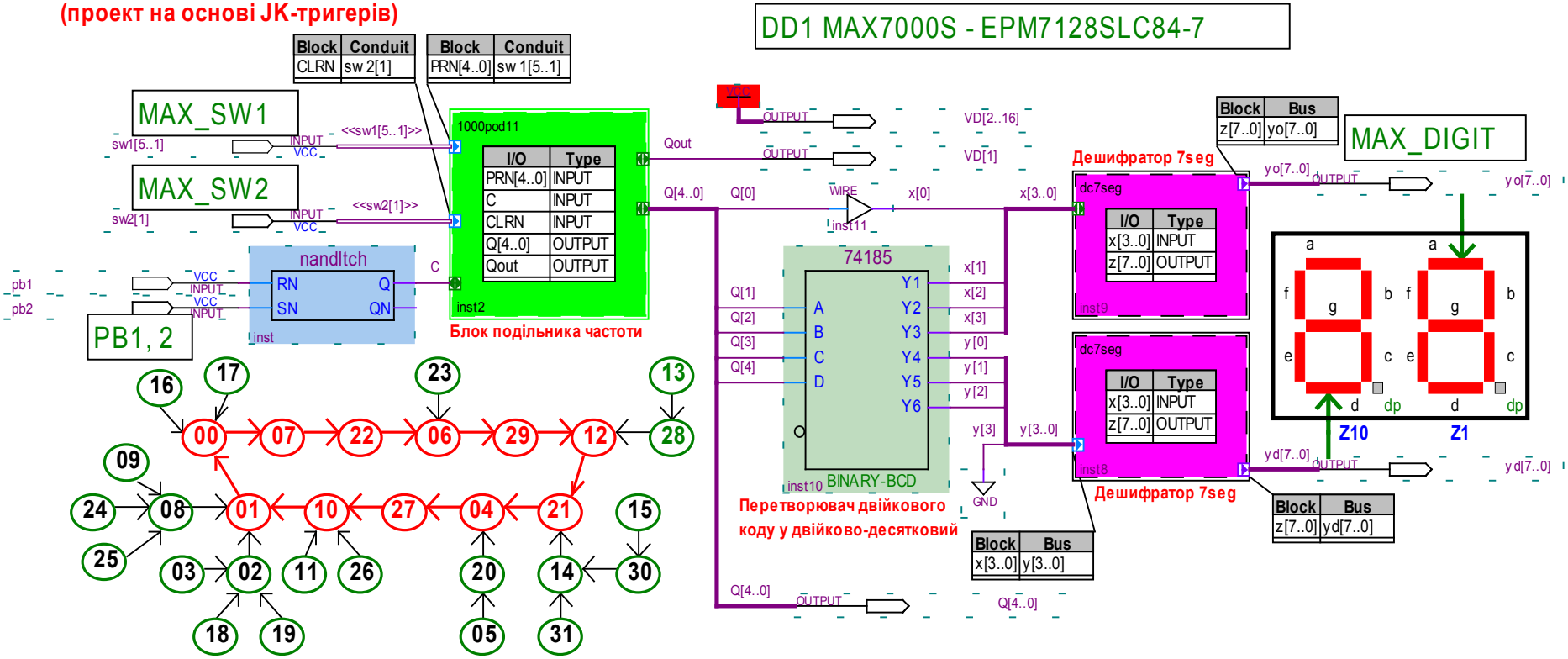


Рисунок Д7

ПОДІЛЬНИК ЧАСТОТИ (М=11)

(проект на основі JK-тригерів)

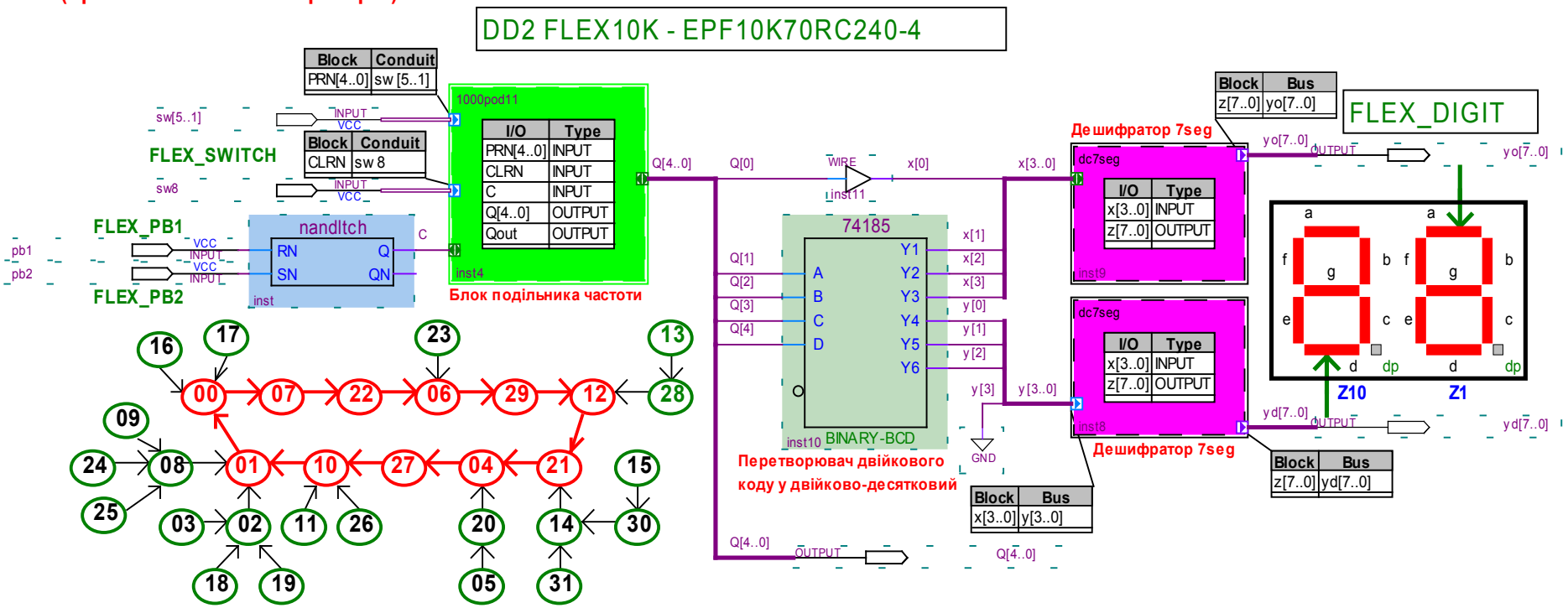


Примітка: 3і входів PRN[ ] стан поза основним циклом слід завантажувати в інверсному коді.

Рисунок Д8

ПОДІЛЬНИК ЧАСТОТИ (М=11)

(проект на основі JK-тригерів)



Примітка: 3і входів PRN[ ] стан поза основним циклом слід завантажувати в інверсному коді.

Рисунок Д9

Навчальне видання

Віктор Леонідович Кофанов
Олександр Володимирович Осадчук
Дмитро Володимирович Гаврілов

ПРАКТИКУМ З ЦИФРОВИХ ПРИСТРОЇВ НА ОСНОВІ САПР QUARTUS II

Навчальний посібник

Оригінал-макет підготовлено авторами

Редактор

Навчально-методичний відділ ВНТУ
Свідоцтво Держкомінформу України
серія ДК № 746 від 25.12.2001
21021, м. Вінниця, Хмельницьке шосе, 95, ВНТУ

Підписано до друку Гарнітура Peterburg
Формат 29,7х42 1/4 Папір офсетний
Друк різографічний Ум. друк. арк.
Тираж прим.
Зам. №

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