МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ПОЛТАВСЬКИЙ ФАХОВИЙ КОЛЕДЖ НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ХАРЧОВИХ ТЕХНОЛОГІЙ»

Олексій Тиртишніков

Комп'ютерна логіка

Навчальний посібник для здобувачів освітньо-професійного ступеня фаховий молодший бакалавр за спеціальністю 123 Комп'ютерна інженерія

Полтава – 2023 рік

Укладач: **Тиртишніков О.І.,** кандидат техн. наук, доцент, викладач ВСП ПФК НУХТ.

Рецензенти: **Поночовний Ю. Л.**, доктор техн. наук, с.н.с., професор кафедри інформаційних систем та технологій Полтавського державного аграрного університету;
Саковець О. О., викладач ВСП ПФК НУХТ, голова циклової комісії комп'ютерної інженерії та автоматизації.

Тиртишніков О.І. Комп'ютерна логіка: навчальний посібник для здобувачів освітньо-професійного ступеня фаховий молодший бакалавр за спеціальністю 123 Комп'ютерна інженерія / Електронне видання. – Полтава: ВСП «ПФК НУХТ», 2023

У посібнику викладено логічні й арифметичні основи будови та функціонування комп'ютерів відповідно до навчальної програми дисципліни «Комп'ютерна логіка» для здобувачів освіти освітньо-професійного ступеня «фаховий молодший бакалавр» спеціальності 123 Комп'ютерна інженерія. Сформульовано понятійний апарат алгебри логіки та методи аналізу і синтезу логічних пристроїв. Розглянуто системи числення, методи кодування двійкових чисел, що застосовуються в комп'ютері, алгоритми виконання арифметичних операцій, будова й функціонування типових комбінаційних і послідовнісних функціональних цифрових вузлів обчислювальних пристроїв (цифрових автоматів). Теоретичний матеріал супроводжується та доповнюється завданнями для проведення практичних та лабораторних занять

Рекомендації щодо користування електронним навчальним посібником

Електронний навчальний посібник «Комп’ютерна логіка» виконаний у вигляді файлу формату PDF, що забезпечує легке та зручне користування ним за допомогою будь-якого браузера чи переглядача (наприклад, Adobe Acrobat Reader, STDU Viewer або подібного), із застосуванням як персонального комп’ютера, так і будь-яких мобільних пристроїв – планшетів, смартфонів, ebook-рідерів, в тому числі і в автономному режимі, при відсутності підключення до мережі Internet. Також будь-яка частина посібника (або посібник цілком) може бути легко скопійована або роздрукована, недоступне лише редагування вихідного файлу.

Особливістю посібника є власна система навігації на основі ієрархічної системи заголовків, закладок та елементів управління, яка забезпечує незалежність навігації від можливостей та особливостей конкретних браузерів, переглядачів або пристроїв. При цьому, зрозуміло, ніяк не обмежується користування убудованими можливостями навігації будь-яких програмних та апаратних засобів, що забезпечують перегляд PDF-файлів.

Засоби та способи навігації

1. Основним засобом навігації є стандартне для PDF-файлів деревовидне ієрархічне меню, що розташовано в лівій частині екрану та, в залежності від браузера або переглядача, що використовується, може називатися «Contents», «Зміст», «Структура» та ін. Як приклад, на рисунку 1 показано меню посібника в переглядачі STDU Viewer.
2. Перехід до будь-якої частини посібника можливий безпосередньо з його змісту, натисненням на назву відповідної частини. Повернення до змісту в будь-який момент може бути виконано натисненням на посилання «Зміст», що розташоване в нижньому колонтитулі на кожній сторінці зліва унизу (приклад поданий на рисунку 2). Фактично цей спосіб дублює попередній, але є корисним при перегляді посібника в повноекранному режимі, коли вікно (або закладка), що містить деревовидне ієрархічне меню, відсутнє на екрані.
3. Перехід до завдань для практичних та лабораторних занять, що розташовані в додатках А, Б, очевидно, також може здійснюватися за допомогою розглянутих раніше засобів. Додатково введена можливість

прямого переходу до конкретного завдання безпосередньо після вивчення відповідного теоретичного матеріалу без використання змісту посібника. Для цього в тексті основної частини посібника присутні відповідні посилання. Приклад такого посилання поданий на рис. 3а. Для повернення в основний текст наприкінці кожного завдання присутня кнопка «Назад» (рис. 3б).

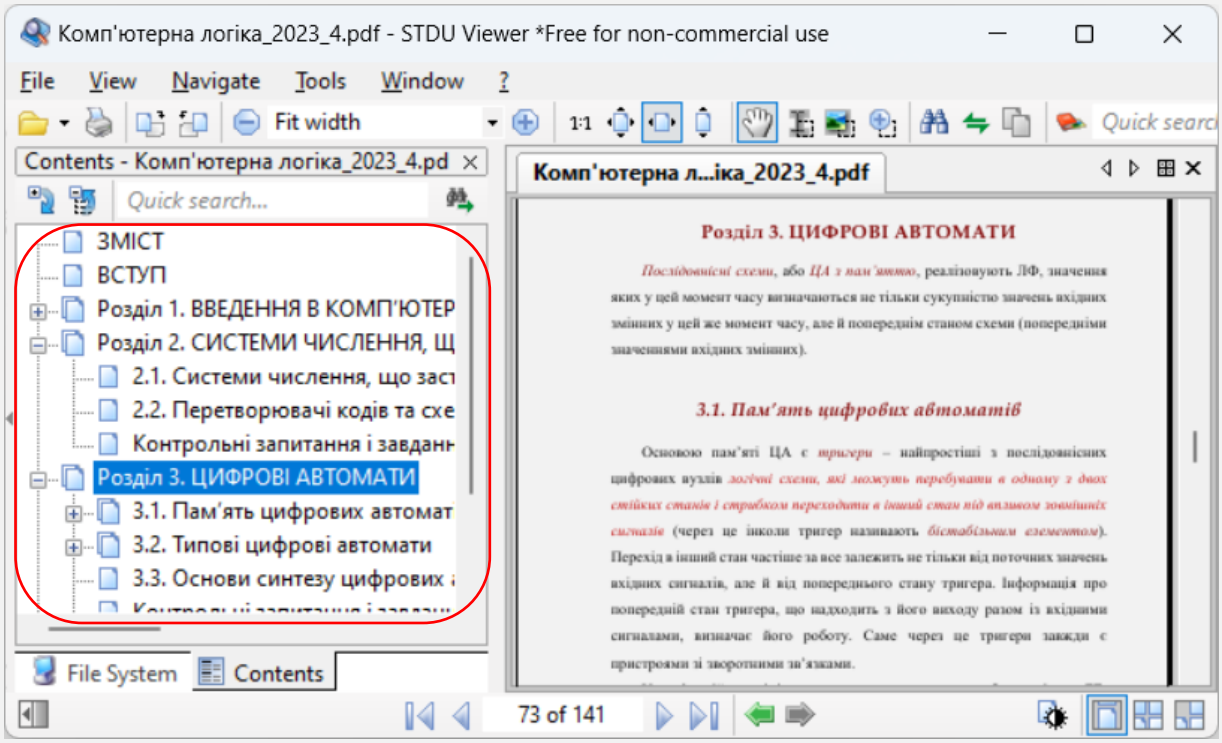


Рис. 1

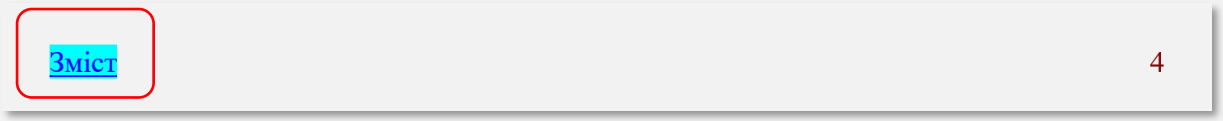


Рис. 2

а б

Рис. 3

ЗМІСТ

ВСТУП

Електронний навчальний посібник з дисципліни «Комп’ютерна логіка» є виправленим та суттєво доповненим електронним перевиданням навчального посібника (Тиртишніков О.І. Комп’ютерна логіка: навчальний посібник. – Полтава: ВСП «ПФК НУХТ», 2022), виконаним з метою підвищення його доступності, привабливості, а також зручності користування ним

Метою навчальної дисципліни «Комп’ютерна логіка» є забезпечення підготовки здобувачів освіти в галузі теорії проєктування апаратного забезпечення комп’ютерів; вивчення логічних і арифметичних основ побудови сучасних комп’ютерів, а саме:

- ✓ основних законів булевої алгебри логіки;
- ✓ задач мінімізації логічних функцій;
- ✓ побудови комбінаційних схем та функціональних вузлів у заданому логічному базисі;
- ✓ методів кодування двійкових чисел;
- ✓ етапів синтезу цифрових автоматів канонічним методом;
- ✓ основ комп’ютерної арифметики;
- ✓ основних понять про типові логічні пристрої та цифрові автомати.

Предметом дисципліни «Комп’ютерна логіка» є арифметичні та логічні основи побудови комп’ютерної техніки, а саме: алгебра логіки та мінімізація логічних функцій, методи та способи кодування інформацій у комп’ютерах, алгоритми виконання арифметичних операцій у комп’ютерах.

У посібнику викладено логічні й арифметичні основи будови та функціонування комп’ютерів відповідно до навчальної програми дисципліни «Комп’ютерна логіка» для здобувачів освіти освітньо-професійного ступеня «фаховий молодший бакалавр» спеціальності 123 Комп’ютерна інженерія. Сформульовано понятійний апарат алгебри логіки та методи аналізу і синтезу логічних пристроїв. Розглянуто системи числення, методи кодування двійкових чисел, що застосовуються в комп’ютері, алгоритми виконання арифметичних операцій, будова й функціонування типових комбінаційних і послідовнісних функціональних цифрових вузлів обчислювальних пристроїв (цифрових автоматів).

Теоретична частина посібника містить чотири розділи, наприкінці кожного з них подані запитання та завдання для самоперевірки, в додатках А, Б – завдання для проведення практичних та лабораторних занять.

Посібник вміщує переважно авторські навчально-методичні матеріали, вдосконалені з урахуванням досвіду викладання як дисципліни «Комп'ютерна логіка», так і дисциплін, які вона забезпечує, а саме «Комп'ютерна схемотехніка», «Архітектура комп'ютерів».

Дисципліна забезпечує формування фахової компетентності: знання теоретичних (логічних та арифметичних) основ побудови сучасних комп'ютерів та їх архітектури, вміння застосовувати їх у процесі побудови та експлуатації комп'ютерів при вирішенні професійних завдань.

Розділ 1. ОСНОВИ КОМП'ЮТЕРНОЇ ЛОГІКИ. АНАЛІЗ І СИНТЕЗ ЛОГІЧНИХ ПРИСТРОЇВ

1.1. Загальні відомості про цифрові автомати

Для позначення різноманітних предметів, понять і дій люди користуються словами. Слова складаються з букв, які утворюють алфавіт. У цифровій техніці використовують кодові слова на основі найпростішого алфавіту, що складається лише з двох символів – 0 та 1. Ці символи називають *логічним нулем* і *логічною одиницею*. Причина вибору такого простого алфавіту – зручність збереження й передавання інформації, закодованої за допомогою логічних нулів і одиниць. Дійсно, можна, наприклад, домовитися, що стан пристрою, коли на його виході є сигнал, відповідає логічній одиниці, а стан «немає сигналу» – логічному нулю. Очевидно, що при такому способі подання інформації цифровий сигнал являє собою кодове слово визначеної довжини (розрядності), передане як «ланцюжок букв» (послідовність символів), і має вигляд аперіодичної імпульсної послідовності, наведеної на рис. 1.1.

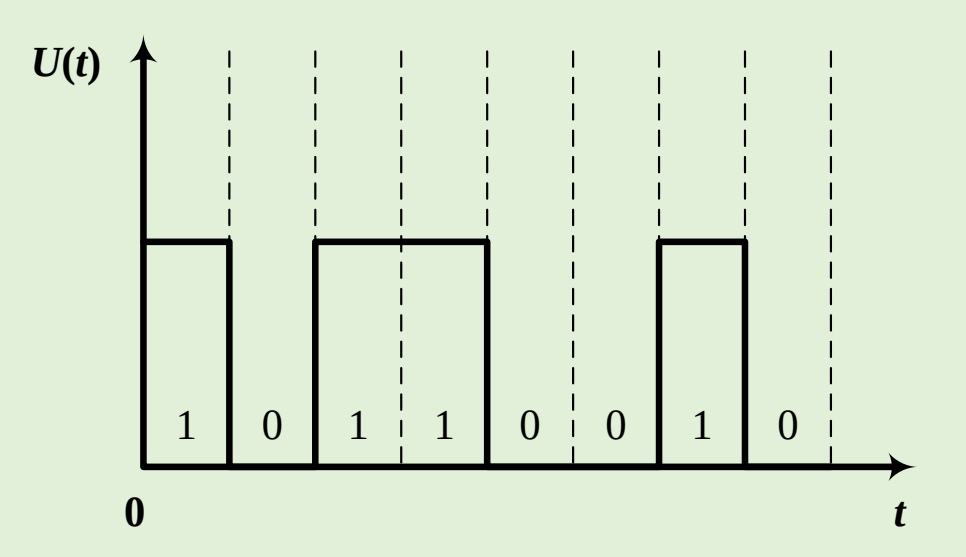


Рис. 1.1 – Цифровий сигнал

Таким чином, на входи цифрового логічного пристрою надходять вхідні кодові слова, на виході пристрою утвориться нове кодове слово, що являє собою результат обробки вхідних слів. Отже, *вихідне слово – це функція, аргументами якої є вхідні слова*. Формалізоване подання залежності між вихідними і вхідними кодовими словами у цифровій логічній схемі називають *логічними функціями* (ЛФ), або функціями алгебри логіки.

Алгебру логіки, що є математичним апаратом теорії цифрових логічних схем, часто називають булевою алгеброю на честь її засновника Джорджа Буля. Іноді для ЛФ використовують назву «перемикальні функції».

Відповідно, під логічними пристроями надалі будемо розуміти пристрої, які призначені для формування (реалізації) функцій алгебри логіки. Елементарні логічні пристрої (логічні елементи, ЛЕ) для розв’язання завдань обробки цифрової інформації об’єднуються в сукупності, що виконують певні функції (наприклад, додавання або множення) і є конструктивно завершеними пристроями обробки інформації. Такі сукупності називають функціональними вузлами цифрових пристроїв або *цифровими автоматами* (ЦА).

ЦА є основними складовими частинами цифрових систем обробки інформації взагалі й комп’ютерів зокрема. Теоретичною базою вивчення функціональних вузлів цифрових пристроїв є кібернетика, що включає в себе, як складову частину, теорію ЦА.

Кінцевим ЦА в кібернетиці називають функціональний цифровий пристрій (або сукупність пристроїв), що без прямої участі людини (автомат – від грецького *automatos* – самодіючий) виконує дискретне перетворення інформації відповідно до заданого алгоритму та має скінченне число входів, виходів і внутрішніх станів. Зрозуміло, що ЦА – це математична модель реального технічного пристрою, що виконує відповідне перетворення цифрової інформації.

ЦА поділяють на дві великі групи – комбінаційні (ЦА без пам’яті) й накопичувальні або послідовнісні (ЦА з пам’яттю).

Комбінаційні ЦА реалізовують ЛФ, значення яких у цей момент часу визначаються лише сукупністю значень (комбінацією) вхідних змінних у цей же момент часу і не залежать від попередніх значень вхідних змінних. Про такі схеми говорять, що вони не мають властивості пам’яті (передісторія функціонування не впливає на результат перетворення вхідних кодових слів), тому їх називають *автоматами без пам’яті*. Часто у літературі ЦА

без пам’яті називають логічними перетворювачами або логічними пристроями.

Послідовнісні або накопичувальні ЦА реалізовують ЛФ, значення яких у цей момент часу визначаються не тільки сукупністю значень вхідних змінних у цей же момент часу, але й попереднім станом схеми (попередніми значеннями вхідних змінних). Про такі схеми говорять, що вони мають властивість пам’яті (передісторія функціонування впливає на результат перетворення вхідних сигналів схемою).

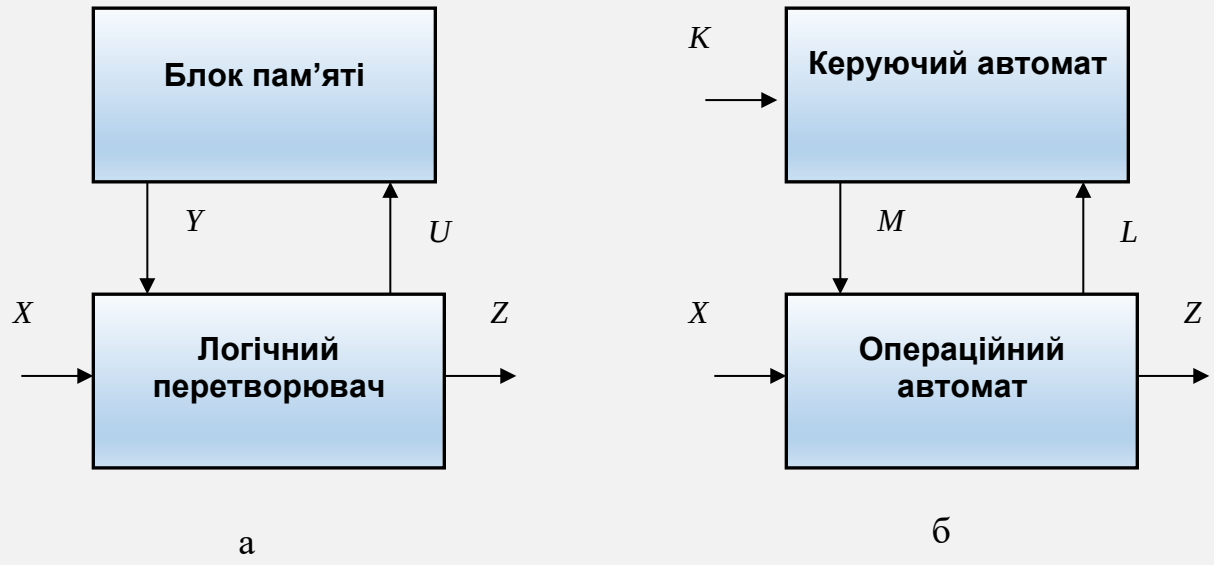
На відміну від комбінаційних, послідовнісні пристрої завжди мають у своєму складі зворотні зв’язки, по яких інформація про попередній стан із виходів пристрою надходить на його входи разом із зовнішніми керуючими сигналами. Наявністю зворотних зв’язків і пояснюється існування властивості пам’яті у послідовнісних пристроях.

ЛФ, що встановлює залежність стану, у який переходить послідовнісний пристрій із поточного стану під впливом заданих сигналів управління, має назву *функції переходів*. Переходи автоматів з пам’яттю з одного стану до іншого починаються з деякого початкового стану, визначення якого є частиною визначення умов функціонування ЦА в цілому. Наступний стан ЦА залежить від вихідного стану та вхідних сигналів. У результаті поточний стан і сигнали на виходах ЦА залежать від початкового стану та всіх попередніх вхідних сигналів, тобто послідовність вхідних сигналів визначає послідовність станів і вихідних сигналів автомата. Цим пояснюється назва «послідовнісні схеми», котру використовують для позначення ЦА з пам’яттю.

Автомати з пам’яттю в канонічному поданні розділяють на дві частини: пам’ять та комбінаційне коло (логічний перетворювач). На входи комбінаційної частини подаються вхідні сигнали та сигнали стану ЦА. На її виході формуються вихідні сигнали та сигнали, що переводять ЦА в наступний стан або сигнали керування пам’яттю (тому комбінаційну частину іноді розділяють за функціональним призначенням на два кола). Відповідна структурна схема ЦА з пам’яттю подана на рис. 1.2, а.

Тут X та Z – множини вхідних та вихідних сигналів ЦА відповідно, U – множина сигналів керування пам’яттю, Y – множина вихідних сигналів блока пам’яті, або сигнали стану ЦА.

Зрозуміло, що комбінаційний ЦА можна розглядати як окремий випадок ЦА з пам’яттю, в якого кількість внутрішніх станів зведена до одного, а елемента пам’яті немає.



а б

Рис. 1.2 – Структурні схеми ЦА: автомат з пам'яттю (а), програмований автомат (б)

Більшість складних ЦА може бути декомпозована на дві частини в інший спосіб: керуючий ЦА та операційний автомат. Перший з них, згідно з командами *K*, формує послідовність мікрокоманд (сигналів управління) *M*, які визначають алгоритм оброблення інформації операційним автоматом. Послідовність *M* може корегуватися залежно від сигналів логічних умов *L*. ЦА з такою структурою одержали назву *програмованих автоматів*.

З метою спрощення аналізу і синтезу логічний перетворювач ЦА з пам'яттю зручно подавати у вигляді двох частин: блока формування вихідних сигналів та блока керування пам'яттю (у реальних ЦА наявність чітко відокремлених двох блоків необов'язкова). Якщо вхідні сигнали подаються тільки на входи блока керування пам'яттю, то вихідні сигнали ЦА будуть функціями виключно попередніх станів ЦА. Такі автомати називають *автоматами Мура*. В *автоматах Мілі* вихідні сигнали залежать як від попередніх станів, так і від вектора вхідних змінних. Структурні схеми автоматів Мілі та Мура подані на рис. 1.3, а та 1.3, б відповідно.

Деякі ЦА відносять до *автономних автоматів*. Вони не мають інформаційних входів і під впливом тактових сигналів переходять у наступні стани за алгоритмом, що визначається структурою автомата.

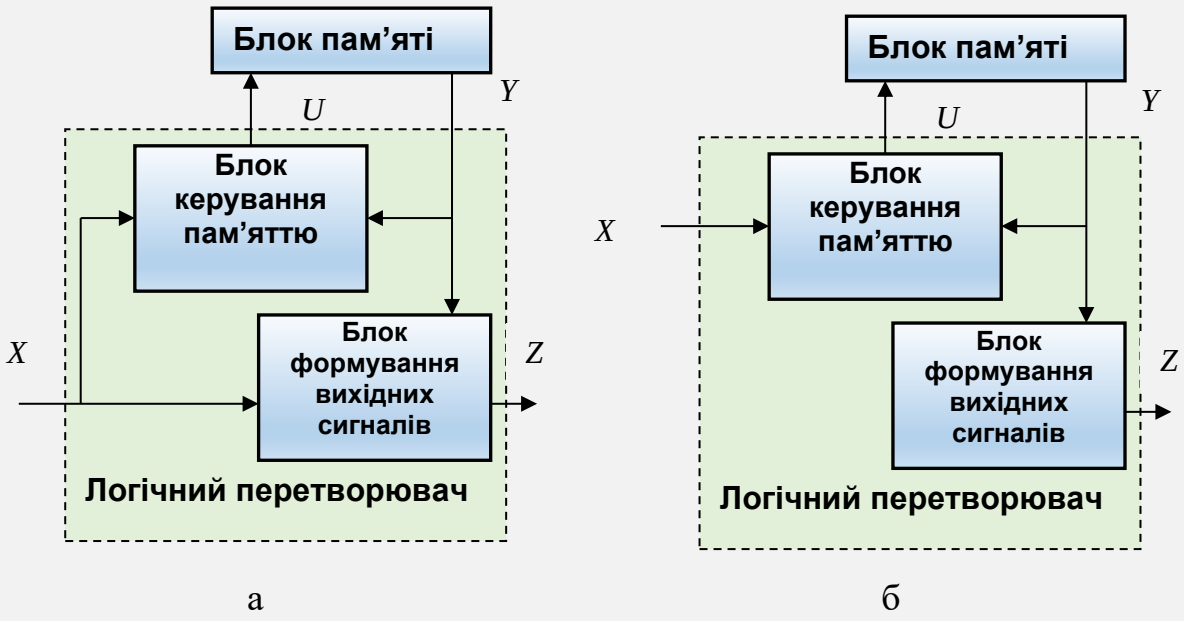


Рис. 1.3 – Структурні схеми ЦА: автомат Мілі (а), автомат Мура (б)

Класифікувати ЦА можна ще за багатьма ознаками.

✓ *За способом введення і виведення кодових слів* розрізняють ЦА послідовної, паралельної та змішаної дії.

На входи пристрою послідовної дії символи кодових слів надходять не одночасно, а послідовно, символ за символом. У такій же послідовній формі формується вихідне слово.

На входи пристрою паралельної дії всі символи вхідного кодового слова подаються одночасно (у паралельній формі). У такій же формі видається вихідне слово. Зрозуміло, що при паралельній формі прийому і видачі кодових слів у пристрої необхідно мати для кожного розряду вхідного (вихідного) слова свій окремий вхід (вихід).

У пристроях змішаної дії вхідні й вихідні кодові слова подаються у різних формах. Наприклад, вхідне слово – у послідовній формі, вихідне – у паралельній. Пристрої змішаної дії можна використовувати для перетворення кодових слів з однієї форми подання в іншу (з послідовної – у паралельну і навпаки).

✓ *За способом синхронізації* розрізняють синхронні (керовані) й асинхронні (з безпосередньою реакцією) ЦА.

У синхронному ЦА зміна його станів визначається (дозволяється або забороняється) спеціальним сигналом управління (синхросигналом). Характерною особливістю таких ЦА є те, що зміни станів автоматів відбуваються в суворо фіксовані моменти часу, які визначаються частотою

надходження синхроімпульсів. Структурна схема синхронного ЦА подана на рис. 1.4.

В асинхронних ЦА вхідні сигнали впливають на їх стан безпосередньо з моменту надходження на вхід, тобто моменти переходів з одного стану в інший завчасно не визначені.

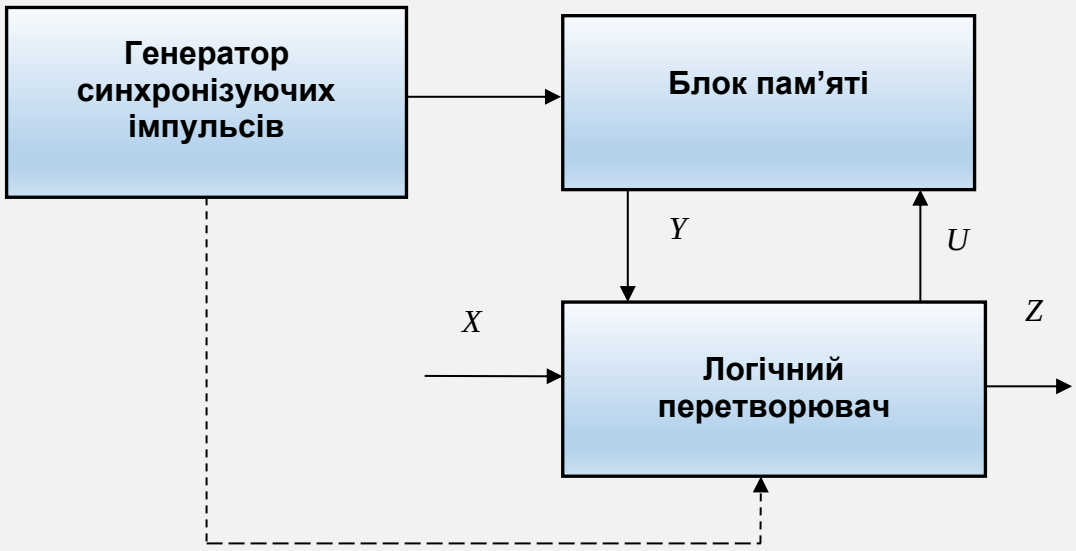


Рис. 1.4 – Структурна схема синхронного ЦА

1.2. Логічні функції і логічні елементи. Закони і правила алгебри логіки

У математиці для подання функції переважно застосовують два способи: аналітичний і табличний. Ці ж два способи використовують і для запису ЛФ.

При використанні табличного способу будується *таблиця істинності*, у якій наведені всі можливі сполучення значень аргументів і відповідні значення ЛФ. Якщо кількість аргументів ЛФ дорівнює n, то число різноманітних сполучень (наборів) значень аргументів складає 2^n, а кількість різноманітних ЛФ n аргументів – 2^{2^n}. Для $n = 1$ число всіх можливих ЛФ одного аргументу дорівнює 4. Вони можуть бути подані таблицею істинності 1.1.

Функції $F_1(x)$ і $F_4(x)$ не залежать від значень аргументу x, $F_2(x)$ дорівнює значенню аргументу. Отже, практичний інтерес має лише

$F_3(x) = \bar{x}$. Це – функція НІ (логічне заперечення, інверсія). Вираз $y = \bar{x}$ читається як «y є не x».

Таблиця 1.1

Аргумент X	Функції			
	$F_1(x)$	$F_2(x)$	$F_3(x)$	$F_4(x)$
0	0	0	1	1
1	0	1	0	1

Для двох аргументів (n = 2) кількість можливих ЛФ дорівнює 16, вони можуть бути подані таблицею істинності 1.2.

Таблиця 1.2

Аргументи		Функції															
X_1	X_2	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Очевидно, що функції $F_0(x_1, x_2) = 0$, $F_3(x_1, x_2) = x_1$, $F_5(x_1, x_2) = x_2$, $F_{15}(x_1, x_2) = 1$ не мають практичного застосування. Функції $F_{10}(x_1, x_2) = \bar{x}_2$ та $F_{12}(x_1, x_2) = \bar{x}_1$ є інверсією одного з аргументів. Функції $F_{11}(x_1, x_2)$, $F_{13}(x_1, x_2)$ (імплікація) і $F_2(x_1, x_2)$, $F_4(x_1, x_2)$ (заборона імплікації) на практиці використовуються не дуже часто, тому розглядати їх не будемо.

Шість функцій, що залишилися, розглянемо більш докладно.

1. $F_1(x_1, x_2) = x_1 \wedge x_2$ (додаткові позначення: $x_1 * x_2$, $x_1 x_2$, $x_1 \& x_2$). Це операція «логічне І» (кон'юнкція, логічне множення). Як видно з таблиці істинності, *у результаті виконання операції І утворюється функція, значення якої дорівнює 1 тільки тоді, коли значення всіх аргументів рівні 1, і дорівнює 0, коли значення хоча б одного з аргументів дорівнює 0.*

2. $F_7(x_1, x_2) = x_1 \vee x_2$ (додаткове позначення $x_1 + x_2$). Це операція «логічне АБО» (диз'юнкція, логічне додавання). *У результаті виконання операції АБО утворюється функція, значення котрої дорівнює 1, коли значення хоча б одного з аргументів дорівнює 1, і дорівнює 0 тільки в тому випадку, коли значення всіх аргументів дорівнюють 0.*

3. $F_{14}(x_1, x_2) = x_1 \uparrow x_2$. З таблиці видно, що $x_1 \uparrow x_2 = \overline{x_1 x_2}$. Тому таку операцію називають «логічним І-НІ», або запереченням кон'юнкції. Нерідко

використовується назва «штрих Шеффера». *У результаті виконання операції І-НІ утворюється функція, значення якої дорівнює 1, коли значення хоча б одного з її аргументів дорівнює 0, і дорівнює 0 тільки в тому випадку, коли значення всіх аргументів рівні 1.*

4. $F_8(x_1, x_2) = x_1 \downarrow x_2$. З таблиці видно, що $x_1 \downarrow x_2 = \overline{x_1 \vee x_2}$. Тому цю операцію називають «логічним АБО-НІ» або запереченням диз'юнкції. Також використовуються назви «стрілка Пірса» і «функція Вебба». *У результаті виконання операції АБО-НІ утворюється функція, значення котрої дорівнює 1 тільки тоді, коли значення всіх аргументів рівні 0, і дорівнює 0, коли значення хоча б одного аргументу дорівнює 1.*

5. $F_6(x_1, x_2) = x_1 \oplus x_2$ (додаткове позначення $x_1 \nabla x_2$). Це функція нерівності (виключне АБО, сума за модулем 2). *У результаті додавання за модулем 2 утворюється функція, значення якої дорівнює 1, коли значення аргументів не збігаються, і дорівнює 0 у протилежному випадку.*

Покажемо, що ця функція також може бути виражена через операції І, АБО, НІ. Таким чином,

$$x_1 \oplus x_2 = x_1\bar{x}_2 \vee \bar{x}_1 x_2 .$$

Вірність цієї рівності доведемо за допомогою таблиці істинності 1.3.

Таблиця 1.3

x_1	x_2	$x_1\bar{x}_2$	$\bar{x}_1 x_2$	$x_1\bar{x}_2 \vee \bar{x}_1 x_2$
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0

Остання колонка цієї таблиці збігається з колонкою F_6 із таблиці 2, тобто рівність правильна.

6. $F_9(x_1, x_2) = x_1 \equiv x_2$ (додаткове позначення $x_1 \leftrightarrow x_2$). Це операція рівності (еквівалентності). *У результаті виконання операції рівності утвориться функція, значення якої дорівнює 1, коли значення аргументів збігаються, і дорівнює 0 у протилежному випадку.*

Очевидно, що $x_1 \leftrightarrow x_2 = \overline{x_1 \oplus x_2}$.

Відзначимо, що функції І й І-НІ, АБО й АБО-НІ, рівності й нерівності утворять так звані дуальні (двоїсті) пари, у котрих одна функція є інверсією

іншої. Функція заперечення завжди є функцією одного аргументу, інші розглянуті елементарні ЛФ у загальному випадку є функціями довільної кількості аргументів.

Операції інверсії, диз'юнкції та кон'юнкції є основними елементарними ЛФ у тому розумінні, що всі інші ЛФ (як елементарні, так і складні) можна подати через ці три основні функції. Отже, властивості будь-яких ЛФ визначаються властивостями кон'юнкції, диз'юнкції та інверсії.

В алгебрі логіки є відмінності від правил звичайної алгебри. Головна відмінність – у неї не існує операцій віднімання і ділення, тому не можна переносити вирази з однієї частини рівності до іншої або ділити на спільний множник.

Порядок виконання операцій в алгебрі логіки такий: у першу чергу виконуються операції інверсії, потім – операції кон'юнкції, в останню чергу – операції диз'юнкції.

Наприклад, вираз $x_1 \vee x_2\bar{x}_3 \vee \bar{x}_4x_3$ припускає таку послідовність дій:

1) $\bar{x}_3$, $\bar{x}_4$;
2) $x_2\bar{x}_3$, $\bar{x}_4x_3$;
3) $x_1 \vee x_2\bar{x}_3 \vee \bar{x}_4x_3$.

Для зміни послідовності виконання (пріоритетності) логічних операцій використовуються дужки. У цьому випадку в першу чергу виконуються операції в дужках.

В алгебрі логіки є чотири основні закони: переставний (властивість комутативності), сполучний (властивість асоціативності), розподільний (властивість дистрибутивності), інверсії (теорема де Моргана). Співвідношення, що відображають ці закони для двох або трьох змінних, наведені у табл. 1.4.

Таблиця 1.4

№	Закон	Логічне додавання	Логічне множення
1	Переставний	$x_1 \vee x_2 = x_2 \vee x_1$	$x_1x_2 = x_2x_1$
2	Сполучний	$(x_1 \vee x_2) \vee x_3 = x_1 \vee (x_2 \vee x_3)$	$(x_1x_2)x_3 = x_1(x_2x_3)$
3	Розподільний	$(x_1 \vee x_2)x_3 = x_1x_3 \vee x_2x_3$	$x_1x_2 \vee x_3 = (x_1 \vee x_3)(x_2 \vee x_3)$
4	Інверсії	$\overline{x_1 \vee x_2} = \bar{x}_1\bar{x}_2$	$\overline{x_1x_2} = \bar{x}_2 \vee \bar{x}_1$

Переставний, сполучний і розподільний закони, крім другого закону дистрибутивності (у таблиці – рядок 3, стовпчик «логічне множення»), подібні відповідним законам звичайної алгебри. Закон інверсії є специфічним і аналога у звичайній алгебрі не має.

Переставний закон: від перестановки доданків і співмножників результат не змінюється.

Сполучний закон: при зміні порядку виконання однакових операцій результат не змінюється.

Розподільний закон коментарів не потребує.

Закон інверсії (теорема де Моргана) говорить про те, що операція АБО може бути виражена через операції І, НІ і навпаки: операція І може бути виражена через операції АБО, НІ. Дійсно, вирази для закону інверсії, записані в рядку 4 табл. 1.4, можуть бути подані у вигляді

$$x_1 + x_2 = \overline{\bar{x}_1 \bar{x}_2} \text{ або } x_1 x_2 = \overline{\bar{x}_1 + \bar{x}_2}.$$

Закон інверсії має фундаментальне значення для визначення так званих функціонально повних систем ЛФ, або логічних базисів.

Функціонально повною системою ЛФ (логічним базисом) називають такий набір елементарних ЛФ, за допомогою якого можна висловити будь-яку складну ЛФ.

Першою функціонально повною системою є набір елементарних ЛФ І, АБО, НІ. Його також називають універсальним, або булевим базисом. При синтезі логічних пристроїв ця система дозволяє більш просто перейти від табличного подання функції до аналітичного, а потім – до функціональної схеми на відповідних цифрових ЛЕ. Проте з теореми де Моргана витікає, що ця система має надмірність і з неї можна виключити деякі функції без втрати функціональної повноти.

Основні функціонально повні системи ЛФ:

- ✓ АБО, НІ – виключена функція І;
- ✓ І, НІ – виключена функція АБО;
- ✓ АБО-НІ;
- ✓ І-НІ.

Існують й інші функціонально повні системи ЛФ. Але найбільш широко застосовують системи, що складаються всього з однієї функції (АБО-НІ, І-НІ).

Під час аналізу складних логічних виразів із метою їх приведення до більш простого і зручного вигляду застосовують правила, подані в табл. 1.5.

У правильності перших п’яти правил неважко переконатися, аналізуючи таблиці істинності логічних функцій І, АБО, НІ.

Четверте правило (повторення), зокрема, вказує на те, що множення на коефіцієнти, що відрізняються від логічного нуля або одиниці, а також обчислення ступеня не має сенсу в алгебрі логіки.

Варто звернути увагу на властивість «симетрії» основних законів і правил булевої алгебри. Усі закони та правила у табл. 1.4 і 1.5 подані парою співвідношень (крім останнього правила в табл. 1.5).

У кожній парі одне співвідношення утворюється з іншого заміною логічного додавання множенням (і навпаки) з одночасною заміною константи 0 на константу 1 (і навпаки). Ця властивість у булевій алгебрі відома як принцип (закон) дуальності. Його можна сформулювати таким чином: *якщо доведена еквівалентність двох логічних співвідношень, то еквівалентні й дуальні їм співвідношення.*

Таблиця 1.5

№	Правило	а	б
1	Інверсії	$\bar{0}=1$	$\bar{1}=0$
2	Незмінності	$x \vee 0 = x$	$x \wedge 1 = x$
3	Універсальної і нульової множини	$x \vee 1 = 1$	$x \wedge 0 = 0$
4	Повторення	$x \vee x = x$	$x \wedge x = x$
5	Додатковості	$x \vee \bar{x} = 1$	$x \wedge \bar{x} = 0$
6	Склеювання	$x_1x_2 \vee x_1\bar{x}_2 = x_1$	$(x_1 \vee x_2)(x_1 \vee \bar{x}_2) = x_1$
7	Подвійного заперечення	$\bar{\bar{x}} = x$	

Розглянуті вище закони і правила використовують для тотожних перетворень булевих виразів, що описують складні ЛФ. Кожна ЛФ реалізовується за допомогою визначеного набору логічних пристроїв. Чим менше елементів містить логічний вираз, тим простіша схема, яка його реалізовує. Процес спрощення складної ЛФ одержав назву мінімізації.

Розрізняють аналітичні й табличні методи мінімізації. Найбільш простим аналітичним методом мінімізації є метод безпосередніх тотожних перетворень, що полягає в послідовному застосуванні до деякої складної ЛФ законів і правил алгебри логіки.

Розглянемо цей метод на прикладі. Мінімізуємо ЛФ, яка має вигляд

$$f(x_1, x_2, x_3) = (\bar{x}_1\bar{x}_2x_3 \vee x_1\bar{x}_2)(x_1 \vee x_3).$$

За розподільним законом

$$f(x_1, x_2, x_3) = \bar{x}_1\bar{x}_2x_3x_1 \vee \bar{x}_1\bar{x}_2x_3x_3 \vee x_1\bar{x}_2x_1 \vee x_1\bar{x}_2x_3.$$

У першому логічному доданку є логічний добуток $\bar{x}_1x_1 = 0$ (правило 5б табл. 1.5). Отже, на підставі правила 3б перший логічний доданок дорівнює 0 і його можна опустити (правило 2а). У другому і третьому доданках є логічні добутки $x_3x_3 = x_3$ і $x_1x_1 = x_1$ (правило 4б). Маємо

$$f(x_1, x_2, x_3) = \bar{x}_1\bar{x}_2x_3 \vee x_1\bar{x}_2 \vee x_1\bar{x}_2x_3.$$

Застосовуючи до першого і третього доданків розподільний закон, маємо

$$f(x_1, x_2, x_3) = (\bar{x}_1 \vee x_1)\bar{x}_2x_3 \vee x_1\bar{x}_2.$$

Згідно з правилами 5а і 2б

$$f(x_1, x_2, x_3) = \bar{x}_2x_3 \vee x_1\bar{x}_2.$$

В останньому виразі можна $\bar{x}_2$ винести за дужки. Остаточно маємо

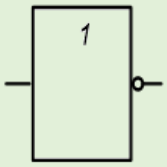
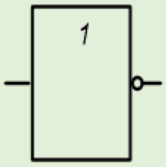
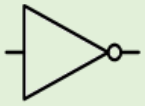
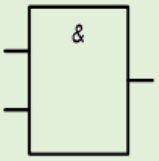
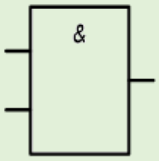
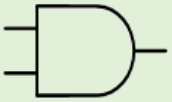
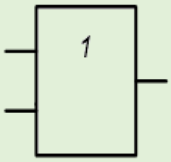
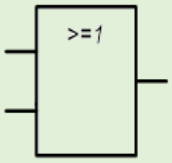
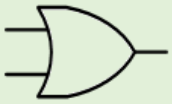
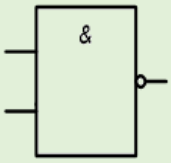
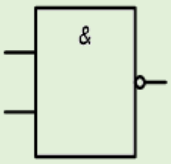
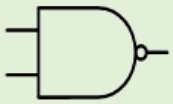
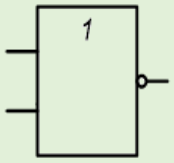
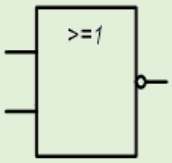
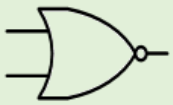
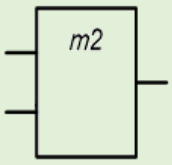
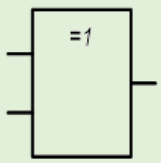
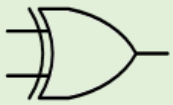
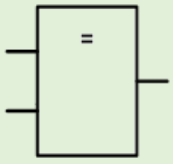
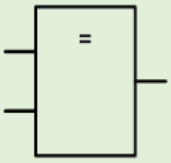
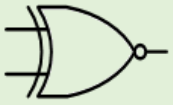
$$f(x_1, x_2, x_3) = \bar{x}_2(x_3 \vee x_1).$$

Головним недоліком методу безпосередніх тотожних перетворень є те, що він не піддається чіткій алгоритмізації. Дії, що реалізовують цей метод, визначаються виглядом вихідного виразу, кваліфікацією виконавця й іншими суб'єктивними факторами. Відсутність однозначної алгоритмізації значно підвищує можливість появи помилок і одержання не цілком мінімізованої функції. Тому такий метод найбільш придатний для відносно простих логічних виразів, коли послідовність перетворень є очевидною для виконавця. Найчастіше цей метод застосовують для остаточної мінімізації виразів, отриманих після використання інших методів.

Як вже вказувалися раніше, *логічні пристрої, що реалізовують елементарні ЛФ, називають ЛЕ*. Умовні графічні позначення (УГП) ЛЕ та їх назви подані у табл. 1.6.

З перелічених ЛЕ в схемах логічних пристроїв можуть утворюватися типові сукупності, що отримали назву складених ЛЕ. Наприклад, елемент 2І – АБО – НІ, що реалізує ЛФ $y = \overline{x_1x_2 \vee x_3x_4}$, УГП якого показано на рис. 1.5.

Таблиця 1.6

№	Логічна функція	Назва елемента	Умовне графічне позначення		
			ГОСТ 270043	Стандарт BS3939	Стандарт MIL/ANSI
1	НІ, заперечення, інверсія	НІ, інвертор			
2	І, логічне множення, кон'юнкція	І, кон'юнктор			
3	АБО, логічне додавання, диз'юнкція	АБО, диз'юнктор			
4	І – НІ, «штрих Шеффера»	І – НІ, елемент Шеффера			
5	АБО – НІ, «Стрілка Пірса»	АБО – НІ, елемент Пірса			
6	Функція нерівності, виключальне АБО, додавання по модулю 2	Виключальне АБО, суматор по модулю 2, елемент нерівності			
7	Функція рівності, еквівалентності	Елемент рівності, еквівалентності			

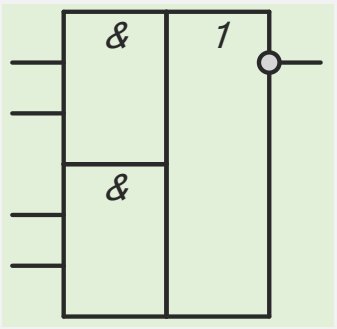


Рис. 1.5 – Складений ЛЕ 2І – АБО – НІ

Практичне завдання «Аналіз логічних схем»

Лабораторна робота «Аналіз логічних пристроїв»

1.3. Основи синтезу логічних пристроїв

У процесі проєктування будь-якого цифрового пристрою доводиться виконувати послідовність дій, що можуть бути віднесені до задач аналізу та синтезу. Виконання задач аналізу логічних пристроїв передбачає наявність готової логічної схеми, побудованої на ЛЕ заданого типу, і зводиться до запису його ЛФ в аналітичному вигляді або побудови таблиці істинності.

Синтез логічного пристрою передбачає побудову його логічної схеми, тобто визначення складу необхідних ЛЕ та сполучень між ними, при яких вхідні ЛФ будуть перетворюватися на вихідні відповідно до заданого алгоритму роботи пристрою. Також у процесі синтезу необхідно прагнути мінімізувати апаратні витрати на реалізацію пристрою. Ця мінімізація безпосередньо пов’язана з мінімізацією ЛФ, яка описує алгоритм роботи пристрою.

1.3.1. Алгоритм синтезу логічного пристрою з одним виходом

Алгоритм синтезу логічного пристрою містить такі основні етапи:

1. Запис умов функціонування пристрою. Ці умови частіше за все задають у вигляді таблиці істинності або ЛФ. Також для порівняно простих

пристроїв можливе визначення умов функціонування у вигляді словесного опису.

2. Запис та мінімізація ЛФ. Якщо на першому етапі функція вже була задана в аналітичному вигляді, то виконується лише її мінімізація.

3. Запис мінімізованої ЛФ у заданому базисі.

4. Зображення отриманої структурної схеми, тобто зображення потрібних ЛЕ і зв'язків між ними.

З перелічених етапів найбільш складним і трудомістким є другий – запис та мінімізація ЛФ. Але функція у вигляді однієї з так званих канонічних форм – *досконалої диз'юнктивної нормальної форми* (ДДНФ) і *досконалої кон'юнктивної нормальної форми* (ДКНФ) – легко може бути отримана безпосередньо з таблиці істинності, що описує функціонування пристрою.

ЛФ, що подана у вигляді логічної суми (диз'юнкції) логічних добутків аргументів або їх інверсій, називається *диз'юнктивною нормальною формою* (ДНФ).

Приклад: $f(x_1, x_2, x_3) = \bar{x}_1\bar{x}_2x_3 \vee x_1\bar{x}_2 \vee x_1\bar{x}_2$.

Якщо кожний доданок ДНФ містить повний набір аргументів функції або їх інверсій, то така ДНФ називається *досконалою* (ДДНФ).

Приклад: $f(x_1, x_2, x_3) = \bar{x}_1x_2\bar{x}_3 \vee x_1\bar{x}_2x_3$.

За аналогією до диз'юнктивних форм можливі відповідні кон'юнктивні нормальні форми (КНФ) і досконалі КНФ (ДКНФ).

ЛФ, що подана у вигляді логічного добутку (кон'юнкції) логічних сум аргументів або їх інверсій, називається *кон'юнктивною нормальною формою* (КНФ).

Якщо кожний доданок КНФ містить повний набір аргументів функції або їх інверсій, то така КНФ називається *досконалою* (ДКНФ).

Приклади: $f(x_1, x_2, x_3) = (x_1 \vee x_2)(x_1 \vee \bar{x}_2 \vee \bar{x}_3)$ – КНФ;

$f(x_1, x_2, x_3) = (x_1 \vee x_2 \vee x_3)(x_1 \vee \bar{x}_2 \vee \bar{x}_3)$ – ДКНФ.

ДДНФ отримують на основі таблиці істинності за таким алгоритмом:

1. Визначають ті набори аргументів (вхідні слова), на яких функція має значення логічної одиниці.

2. Для кожного з таких наборів записують кон'юнкцію всіх аргументів або їх інверсій. При цьому ті аргументи, які мають значення 1, записуються без інверсії, а ті, що мають значення 0, – з інверсією.

3. Отримані таким чином кон’юнкції (їх буде стільки, скільки одиниць має рядок значень функції у таблиці) поєднуються між собою операціями диз’юнкції.

Розглянемо застосування цього алгоритму на прикладі обчислення ДДНФ для ЛЕ АБО з двома входами (таблиця істинності та елементи ДДНФ наведені в табл. 1.7).

Таблиця 1.7

x_1	x_2	y	**Кон’юнкція**
0	0	0	–
0	1	1	$\overline{x_1}x_2$
1	0	1	$x_1\overline{x_2}$
1	1	1	x_1x_2

Одержимо ДДНФ функції

$$y = \overline{x_1}x_2 + x_1\overline{x_2} + x_1x_2.$$

Цілком зрозуміло, що отримана функція не є мінімальною (найбільш простою) формою подання ЛФ АБО ($y = x_2 + x_1$). Мінімізуємо одержаний вираз за допомогою безпосередніх тотожних перетворень. Застосовуючи до другого та третього елементів функції правило склеювання ($x_1 = x_1\overline{x_2} + x_1x_2$), отримуємо

$$y = \overline{x_1}x_2 + x_1.$$

Відповідно до розподільного закону маємо

$$y = (x_1 + \overline{x_1})(x_1 + x_2).$$

Приймаючи до уваги, що $x_1 + \overline{x_1} = 1$, одержуємо

$$y = x_2 + x_1.$$

На підставі таблиці істинності досить легко отримати і ДКНФ, якщо брати до уваги не одиничні значення ЛФ, а нульові. Треба також, згідно з принципом дуальності, поміняти місцями операції АБО та І.

ДНФ та КНФ використовують у багатьох методах мінімізації для запису в алгебраїчній формі ЛФ, що задані таблицею істинності.

1.3.2. Мінімізація логічних функцій із застосуванням карт Карно

Карта Карно є специфічною формою зображення таблиці істинності ЛФ, що подана у вигляді ДДНФ.

Загальний вигляд карти Карно для функцій двох, трьох і чотирьох змінних поданий на рис. 1.6. У клітинках карт також наведені, як довідкова інформація для полегшення їх заповнення, двійкові набори аргументів та їх десяткові порядкові номери у порядку їх звичайного розташування у таблицях істинності – від набору $x_1 = \ldots = x_n = 0$ до набору $x_1 = \ldots = x_n = 1$.

Слід звернути увагу на те, що така нумерація відповідає порядку слідування аргументів $x_1x_2x_3x_4$ у кодовому слові (аргумент з більшим індексом – молодший). Для зворотного порядку індексації нумерація клітинок карти буде іншою.

$n = 2$

	x_2	$\overline{x_2}$
x_1	11 **3**	10 **2**
$\overline{x_1}$	01 **1**	00 **0**

$n = 3$

	x_2	x_2	$\overline{x_2}$	$\overline{x_2}$
x_1	110 **6**	111 **7**	101 **5**	100 **4**
$\overline{x_1}$	010 **2**	011 **3**	001 **1**	000 **0**
	$\overline{x_3}$	x_3	x_3	$\overline{x_3}$

$n = 4$

	x_2	x_2	$\overline{x_2}$	$\overline{x_2}$	
x_1	1100 **12**	1110 **14**	1010 **10**	1000 **8**	$\overline{x_4}$
x_1	1101 **13**	1111 **15**	1011 **11**	**1001 9**	x_4
$\overline{x_1}$	0101 **5**	0111 **7**	0011 **3**	0001 **1**	x_4
$\overline{x_1}$	0100 **4**	0110 **6**	0010 **2**	0000 **0**	$\overline{x_4}$
	$\overline{x_3}$	x_3	x_3	$\overline{x_3}$	

Рис. 1.6 – Загальний вигляд карт Карно для функцій двох, трьох і чотирьох змінних

Кількість клітинок карти дорівнює кількості можливих варіантів вхідних слів (наборів аргументів) і при кількості аргументів n дорівнює 2^n. У кожну клітинку ставиться 1 тільки у тому випадку, якщо набір аргументів, який відповідає цій клітинці, наявний у запису функції у вигляді ДДНФ, тобто функція на цьому наборі має значення одиниці (нульові значення функції в карті розміщати необов'язково).

Фактично при заповненні карти Карно одиницями аргументи ЛФ та їх інверсії, які записані навколо карти, використовуються як координати відповідних клітинок.

Для визначення мінімізованої ДНФ (МДНФ) за допомогою карти Карно необхідно поєднати усі клітинки, які вміщують одиницю, у контури (замкнені прямокутні області з кількістю клітинок у кожній, що дорівнює 2^k, де k=0,1,2,3,4).

Для знаходження оптимального (найкращого) покриття карти Карно контурами необхідно враховувати деякі особливості розв'язання даної задачі.

1. Контури можуть перетинатися і таж сама клітинка може належати кільком контурам.
2. У кожному контурі має бути хоча би одна «унікальна» одиниця, що належить тільки даному контуру, тобто контур, всі одиниці якого вже належать іншим контурам, є надлишковим.
3. Кількість контурів має бути якомога меншою, а кількість клітинок у кожному контурі – якомога більшою. За цих умов кількість аргументів у членах МДНФ ЛФ буде мінімальною.
4. Права та ліва границі карти – це сусідні стовпчики, так само, як верхній та нижній рядки карти Карно – це сусідні рядки (наприклад, чотири одиниці, розташовані в кутах карти, утворюють один контур).

На рис. 1.7 зображено три варіанти вибору замкнених областей на тій самій карті Карно. Оптимальним, з урахуванням вказаних особливостей, є варіант, зображений ліворуч.

Кожна замкнена область на карті Карно у МДНФ буде подана кон'юнкцією, кількість аргументів якої на k менша загального числа аргументів функції (тобто n-k). У кон'юнкції будуть наявні тільки ті аргументи (або їх інверсії), які в межах відповідного контуру не змінюють свого значення, тобто мають значення тільки з інверсією, або тільки без інверсії.

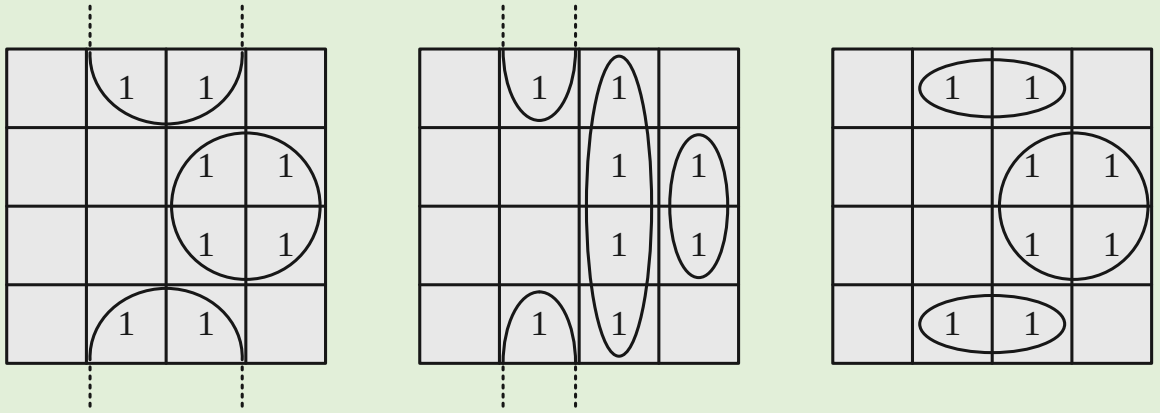


Рис. 1.7 – Варіанти вибору замкнених областей на карті Карно

Пояснити це можна таким чином. Для ЛЕ АБО з двома входами (ДДНФ $y = \overline{x_1}x_2 + x_1\overline{x_2} + x_1x_2$) карта матиме вигляд, показаний на рис. 1.8.

	x_2	$\overline{x_2}$
x_1	**1**	**1**
$\overline{x_1}$	**1**	**0**

Рис. 1.8 – Карта Карно для ДДНФ елемента АБО з двома входами

Мінімізуючи цю ДДНФ у прикладі, який був розглянутий раніше, ми користувалися правилом склеювання

$$x_1\overline{x_2} + x_1x_2 = x_1.$$

Цей запис відповідає області, що охоплює дві верхні клітинки карти. Ми бачимо, що в результаті склеювання у нас залишився тільки аргумент x_1, який в обох клітинках цієї області наявний тільки без інверсії. Аргумент x_2, який у лівій клітинці має значення без інверсії, а у правій – з інверсією, виключений з остаточного виразу.

Для області, що охоплює дві ліві клітинки карти, маємо

$$x_1x_2 + \overline{x_1}x_2 = x_2.$$

Остаточно

$$y = x_2 + x_1.$$

Таким чином, фактично за допомогою даної карти Карно ми виконали дві операції склеювання.

Треба також зазначити, що якщо для карти Карно кількість нулів значно менша за кількість одиниць, то зручніше склеювати саме їх. Не треба тільки забувати, що у цьому випадку ми отримаємо не саму мінімальну функцію, а її інверсію, яку на завершення треба проінвертувати.

Практичне завдання «Мінімізація ЛФ методом карт Карно»

Інколи при проєктуванні конкретних цифрових пристроїв заздалегідь відомо, що деякі вхідні слова ніколи не використовуються (використання їх у цій схемі заборонено). На цих наборах ЛФ можна довизначити за власним бажанням аби максимально спростити її мінімізацію. Наведемо приклад такого довизначення ЛФ під час її мінімізації. Нехай на рис. 1.9 картою Карно подана деяка ЛФ, невизначені значення якої позначені літерою *X*.

	x_2	x_2	$\overline{x_2}$	$\overline{x_2}$	
x_1	**1**	**X**	**1**	**X**	$\overline{x_4}$
x_1	**0**	**1**	**X**	**0**	x_4
$\overline{x_1}$	**X**	**0**	**0**	**0**	x_4
$\overline{x_1}$	**0**	**X**	**0**	**0**	$\overline{x_4}$
	$\overline{x_3}$	x_3	x_3	$\overline{x_3}$	

	x_2	x_2	$\overline{x_2}$	$\overline{x_2}$	
x_1	**1**	**1**	**1**	**1**	$\overline{x_4}$
x_1	**0**	**1**	**1**	**0**	x_4
$\overline{x_1}$	**0**	**0**	**0**	**0**	x_4
$\overline{x_1}$	**0**	**0**	**0**	**0**	$\overline{x_4}$
	$\overline{x_3}$	x_3	x_3	$\overline{x_3}$	

Рис. 1.9 – Приклад довизначення ЛФ

Як можна побачити на остаточному варіанті карти Карно, котрий показано праворуч від початкового варіанта карти, три верхні невизначені значення функції було замінено на одиниці, а два нижні значення – на нулі. Таким чином вдалося створити лише два контури по чотири одиниці, тобто зменшити кількість членів МДНФ до двох, у кожному з яких є тільки два аргументи.

Розглянемо приклад застосування алгоритму синтезу логічного пристрою з одним виходом, ЛФ котрого є функцією чотирьох змінних і подана у вигляді табл. 1.8.

Таблиця 1.8

x_1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
x_2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
y	0	0	0	0	1	1	1	X	1	1	1	1	0	0	0	0

Відповідна ЛФ у вигляді ДДНФ

$$y = \bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 + \bar{x}_1 x_2 x_3 \bar{x}_4 + \bar{x}_1 x_2 x_3 x_4 + x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 + x_1 \bar{x}_2 x_3 \bar{x}_4 + x_1 \bar{x}_2 x_3 x_4 + x_1 x_2 \bar{x}_3 \bar{x}_4 + x_1 x_2 x_3 x_4.$$

Будуємо карту Карно, на якій позначаємо як визначені, так і невизначені значення функції, як показано на рис. 1.10.

	x_2	x_2	$\bar{x}_2$	$\bar{x}_2$	
x_1	**0**	**0**	**1**	**1**	$\bar{x}_4$
x_1	**0**	**0**	**1**	**1**	x_4
$\bar{x}_1$	**1**	**X**	**0**	**0**	x_4
$\bar{x}_1$	**1**	**1**	**0**	**0**	$\bar{x}_4$
	$\bar{x}_3$	x_3	x_3	$\bar{x}_3$	

	x_2	x_2	$\bar{x}_2$	$\bar{x}_2$	
x_1	**0**	**0**	**1**	**1**	$\bar{x}_4$
x_1	**0**	**0**	**1**	**1**	x_4
$\bar{x}_1$	**1**	**1**	**0**	**0**	x_4
$\bar{x}_1$	**1**	**1**	**0**	**0**	$\bar{x}_4$
	$\bar{x}_3$	x_3	x_3	$\bar{x}_3$	

Рис. 1.10 –

Довизначення ЛФ, заданої таблицею 1.8

МДНФ матиме вигляд

$$y = \bar{x}_1 x_2 + x_1 \bar{x}_2 .$$

Пояснити відсутність змінних x_3 та x_4 можна дуже просто. Це означає, що для кожного набору змінних, де функція має значення логічної 1, можна знайти такий самий набір (за винятком змінних x_3 та x_4, які на ньому мають протилежні значення), де функція теж має значення логічної 1. Це свідчить про те, що функція y не залежить від указаних вище аргументів.

Як приклад, запишемо мінімізовану ЛФ у базисі АБО-НІ. Для цього застосуємо закон інверсії, або правило де Моргана,

$$y = \overline{\overline{x_1} + \overline{x_2}} + \overline{\overline{x_1} + x_2}.$$

Структурну схему синтезованого логічного пристрою зображено на рис. 1.11.

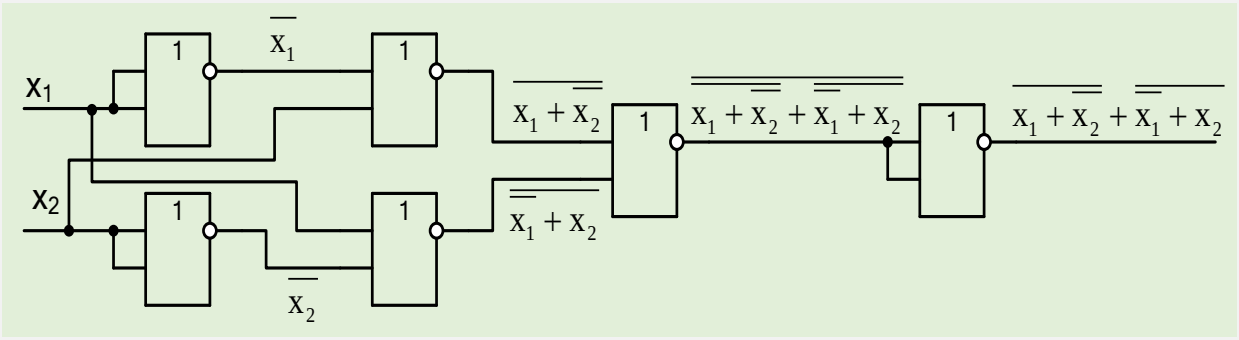


Рис. 1.11 – Схема синтезованого логічного пристрою

Отримана схема, очевидно, не є найпростішою, оскільки в результаті мінімізації ми отримали ЛФ суматора за модулем 2.

Таким чином, карта Карно фактично дозволяє одночасно виконати всі операції склеювання, які можливі для ЛФ, поданої у ДДНФ. Після застосування карти Карно іноді можлива подальша мінімізація ЛФ – за умови переходу до логічних базисів, котрі містять більш складні ЛЕ (наприклад, суматори по модулю два), або за рахунок застосування розподільного закону.

Практичне завдання «Мінімізація неповністю визначених ЛФ. Синтез логічних пристроїв»

1.3.3. Особливості синтезу складних логічних пристроїв

Синтез логічних пристроїв із великою кількістю входів

Раніше розглядалася мінімізація ЛФ із кількістю аргументів до чотирьох. Подання й мінімізація ЛФ за допомогою карт Карно істотно ускладняються, якщо кількість аргументів перевищує чотири. На рис. 1.12 показано приклад подання ЛФ п’яти аргументів за допомогою карт Карно на чотири аргументи.

Карта тут складається з двох половин, кожна з яких являє собою карту для чотирьох аргументів. Одна з них відповідає $x_5 = 1$, друга – $x_5 = 0$. Ці

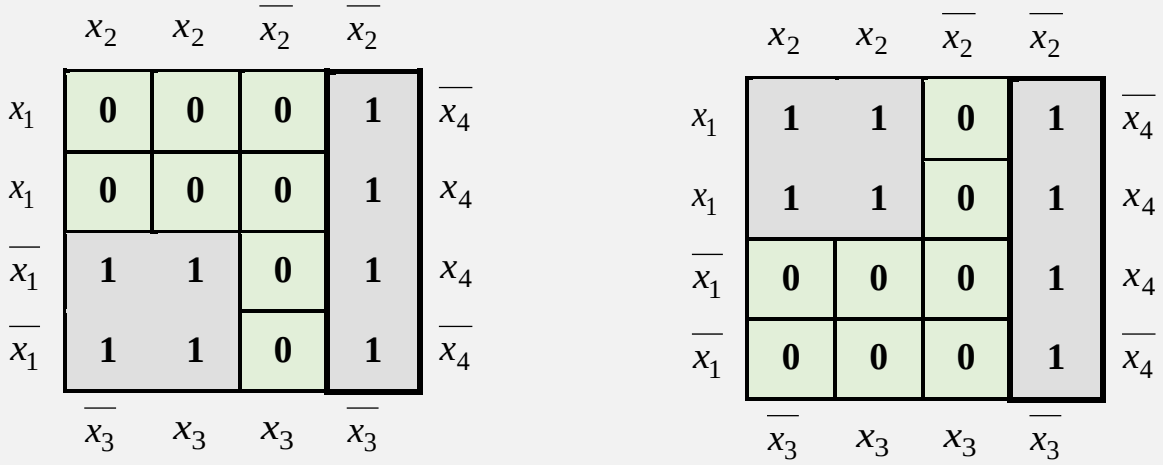
карти можна уявити собі розташованими одна над одною. При цьому контури можуть бути тривимірними, тобто одна область може охоплювати клітинки обох половин карти. На рис. 1.11 такий тривимірний контур охоплює вісім клітинок (праві стовпчики обох половин карти).

Відповідно МДНФ ЛФ має вигляд

$$Y = \overline{x_1}x_2x_5 + x_1x_2x_5 + \overline{x_2}\,\overline{x_3}\,.$$

Очевидно, що якщо контур розташований в одній з половин карти (не є тривимірним), то у відповідному елементі МДНФ наявний аргумент x_5 – з інверсією або без неї. В елементі МДНФ, що відповідає тривимірному контурові, аргумент x_5, навпаки, відсутній.

Для мінімізації функцій з кількістю аргументів, більшою від п’яти, карти Карно виявляються незручними. Мінімізація таких функцій може бути виконана іншими методами, наприклад методом Квайна.



$x_5 = 1$ $x_5 = 0$

Рис. 1.12 – Подання функції п’яти аргументів за допомогою карти Карно

Синтез логічних пристроїв з декількома виходами

Припустимо, що синтезований логічний пристрій має n входів і m виходів. На кожному з виходів повинна бути сформована визначена ЛФ вхідних змінних.

Ця задача могла б бути розв’язана шляхом синтезу m роздільно діючих вузлів, кожний з яких реалізував би визначену вихідну ЛФ. Однак навіть якщо кожний з цих вузлів буде побудований мінімальним способом, логічний пристрій у цілому може виявитися не мінімальним. Дійсно,

найчастіше такий пристрій може бути ще мінімізований шляхом спільного використання загальних елементів, що реалізовують у різних вузлах ті самі фрагменти ЛФ.

Отже, приведення кожної з вихідних ЛФ окремо до мінімальної форми не є умовою отримання логічного пристрою, мінімального в цілому. При мінімізації багатофункціонального пристрою в цілому деякі з реалізованих ним ЛФ можуть виявитися поданими не в мінімальній формі.

Пояснимо сказане вище на прикладі.

Припустимо, що пристрій реалізовує дві ЛФ – Y_1 і Y_2, подані картами Карно (рис. 1.13).

Необхідно синтезувати мінімальну схему логічного пристрою в булевому базисі.

З карт Карно отримаємо ЛФ Y_1 і Y_2 у МДНФ:

$$Y_1 = \overline{x_1} x_4 + \overline{x}_2\, \overline{x_3} + x_2 x_4;$$

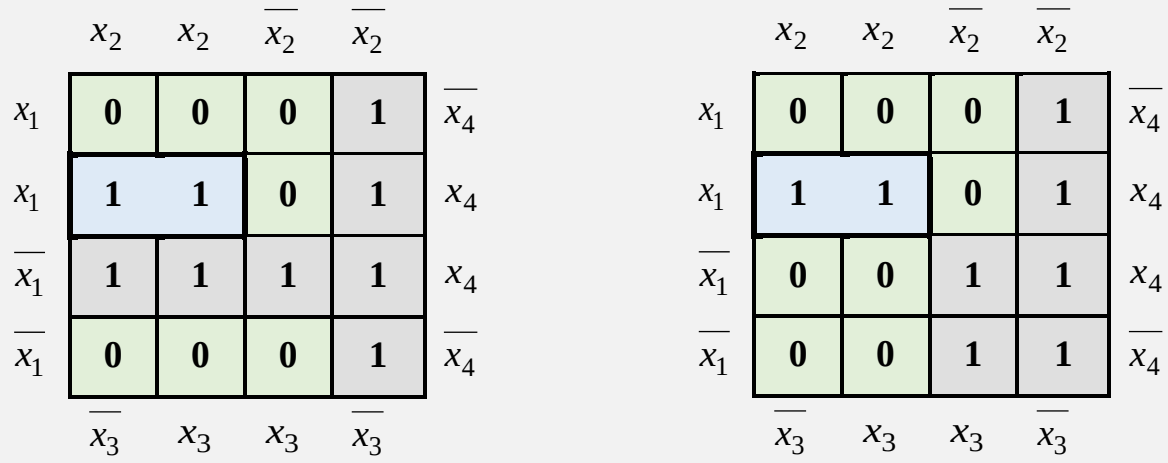


Рис. 1.13 – Приклад ЛФ, реалізованих одним пристроєм

$$Y_2 = \overline{x}_1\, \overline{x}_2 + \overline{x}_2\, \overline{x_3} + x_1 x_2 x_4.$$

Очевидно, що в цих ЛФ є спільна кон'юнкція $\overline{x}_2\, \overline{x_3}$, що може бути реалізована в схемі спільним елементом І. Однак це не єдиний спільний елемент, що може бути виділений у такій схемі. Інша спільна кон'юнкція $x_1 x_2 x_4$ може бути отримана при неповній мінімізації функції Y_1 (відповідний загальний контур на картах рис. 1.13 виділений жирною рамкою)

$$Y_1 = \overline{x_1} x_4 + \overline{x}_2 \overline{x_3} + x_1 x_2 x_4 .$$

Схема, що реалізовує необхідні ЛФ, зображена на рис. 1.14.

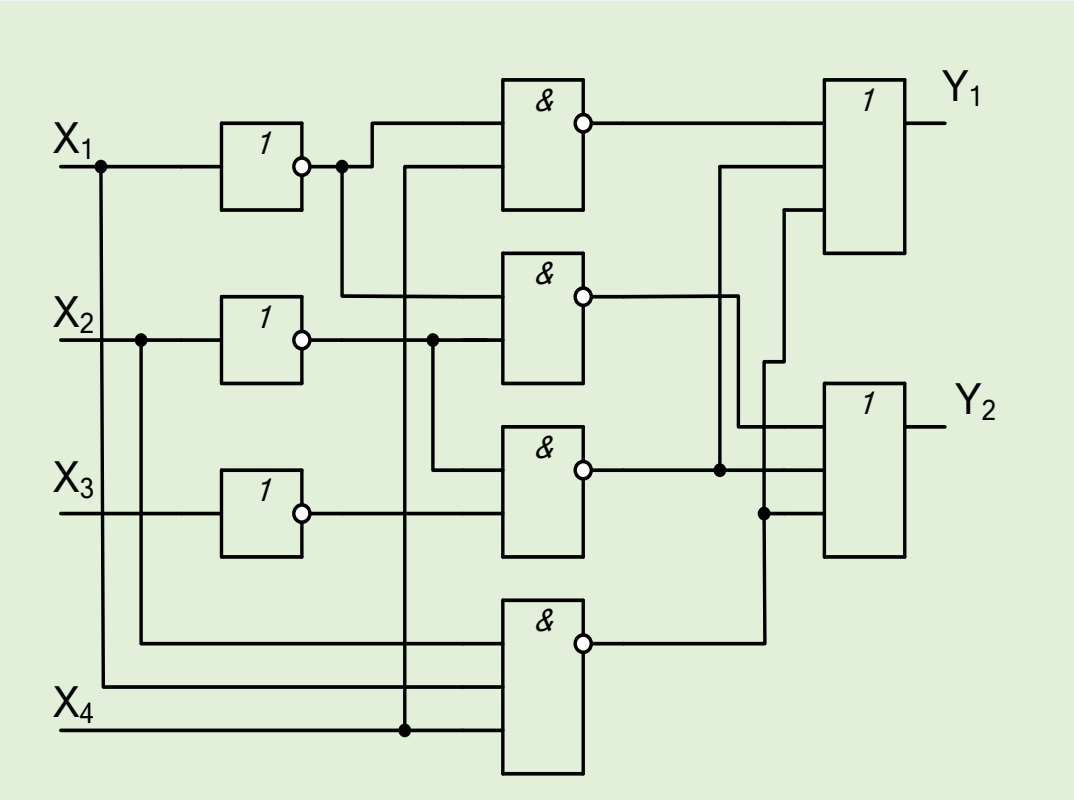


Рис. 1.14 – Схема синтезованого логічного пристрою

1.3.4. Особливості побудови реальних логічних пристроїв

У різних серіях реальних мікросхем низького ступеню інтеграції, як правило, передбачається наявність наборів елементів, які виконують ту саму ЛФ, але мають різну кількість входів. Тому для побудови цифрових пристроїв у більшості випадків можуть бути використані елементи саме з тією кількістю входів, що потрібна в окремих елементах структурної схеми.

Але іноді, наприклад з вимог мінімізації кількості корпусів мікросхем у схемі конкретного логічного пристрою, доводиться використовувати елементи, кількість входів у яких більша або менша необхідної. Нижче розглядаються особливості побудови реальних логічних пристроїв, що виникають у цих випадках.

Припустимо, що у наявності є ЛЕ з трьома входами, а для подачі необхідних вхідних змінних досить двох входів. Надлишковий вхід міг би бути залишений вільним (не підключеним до будь-яких електричних кіл).

Однак у такому випадку знижується завадостійкість схеми через перешкоди, що наводяться на вільні входи, тому таке використання вільних входів елементів небажане.

Можливі різні способи підключення надлишкових входів. Наприклад, вхід, що не використовується, може бути підключений до будь-якого з використовуваних входів того ж елемента, як показано на рис. 1.15.

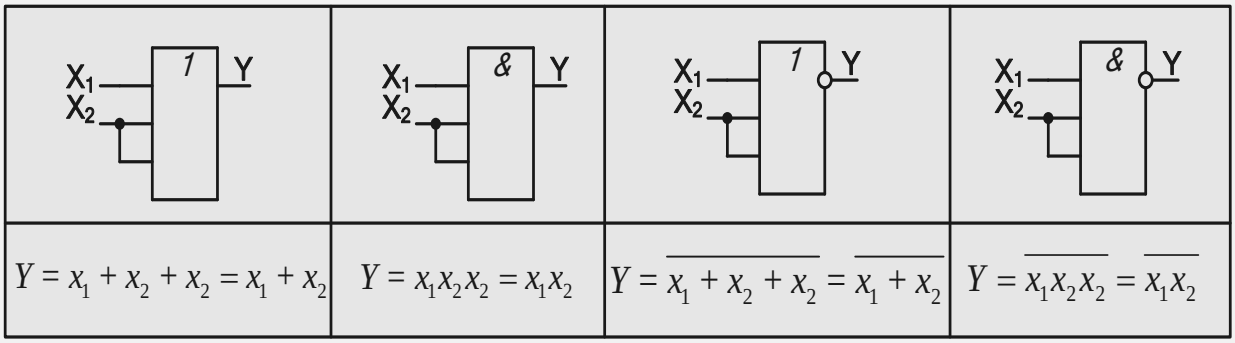


Рис. 1.15 – Підключення входу, що не використовується, до будь-якого з використовуваних входів ЛЕ

Недоліком такого способу підключення надлишкових входів є те, що об’єднання входів призводить до збільшення навантаження на вихід попереднього елемента, що у свою чергу збільшує затримку поширення сигналу і знижує швидкодію елемента.

Тому кращим є спосіб підключення, при якому на вхід, котрий не використовується, подається потенціал, що відповідає рівневі логічного 0 або 1. Правило вибору рівня полягає у тому, що на вхід, який не використовується, повинен подаватися сигнал пасивного рівня, тобто такий, котрий не викликає зміни стану ЛЕ. Тобто на вільні входи елементів АБО і АБО-НІ необхідно подавати логічний нуль, а елементів І, І-НІ – логічну одиницю, як показано на рис. 1.16.

При цьому рівень логічного нуля може бути поданий підключенням входу, що не використовується, до корпусу схеми, а рівень логічної одиниці – підключенням до плюса джерела живлення через обмежуючий опір.

Елементи І-НІ, АБО-НІ, у яких використовується лише один вхід, а входи, що не використовуються, підключені будь-яким з розглянутих раніше способом, виконують операцію НІ (функцію інвертора). Варіанти підключення входів, що не використовуються, для цього випадку показані на рис. 1.17.

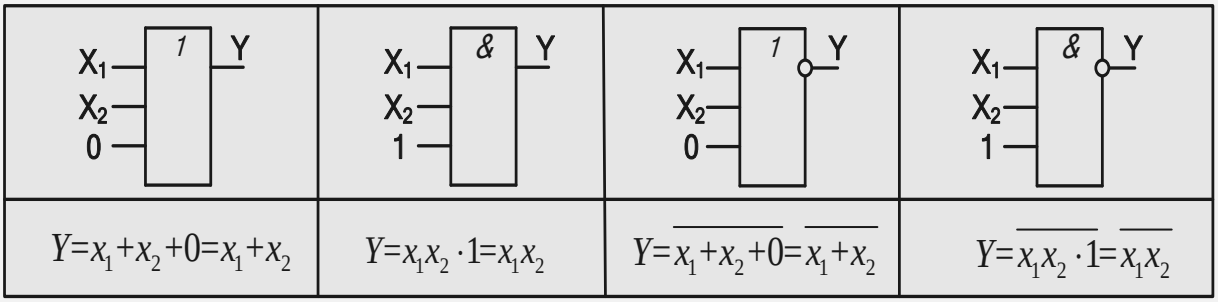


Рис. 1.16 – Подача рівнів логічних нулів і одиниць на входи, що не використовуються

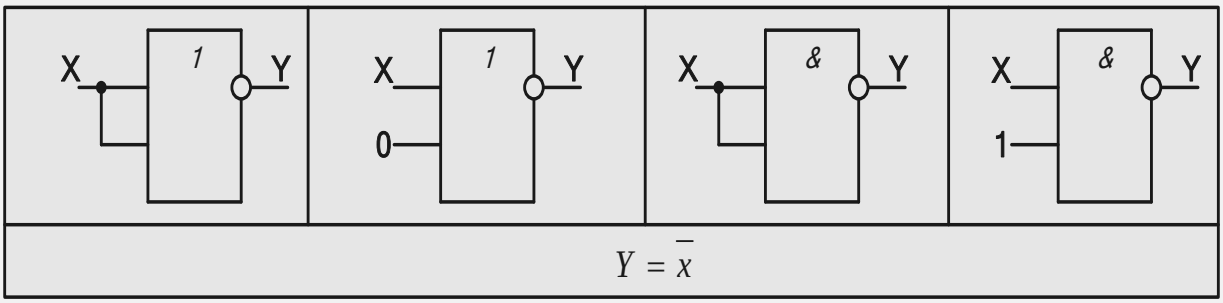


Рис. 1.17 – Застосовування елементів І-НІ, АБО-НІ як інверторів

Якщо необхідно побудувати який-небудь ЛЕ з великою кількістю входів на основі елементів того ж типу, але з меншим числом входів, варто використовувати каскадне з'єднання елементів. При цьому входи елементів І, АБО підключаються безпосередньо до виходів попередніх елементів, а елементів І-НІ, АБО-НІ − через інвертори, як показано на рис. 1.18.

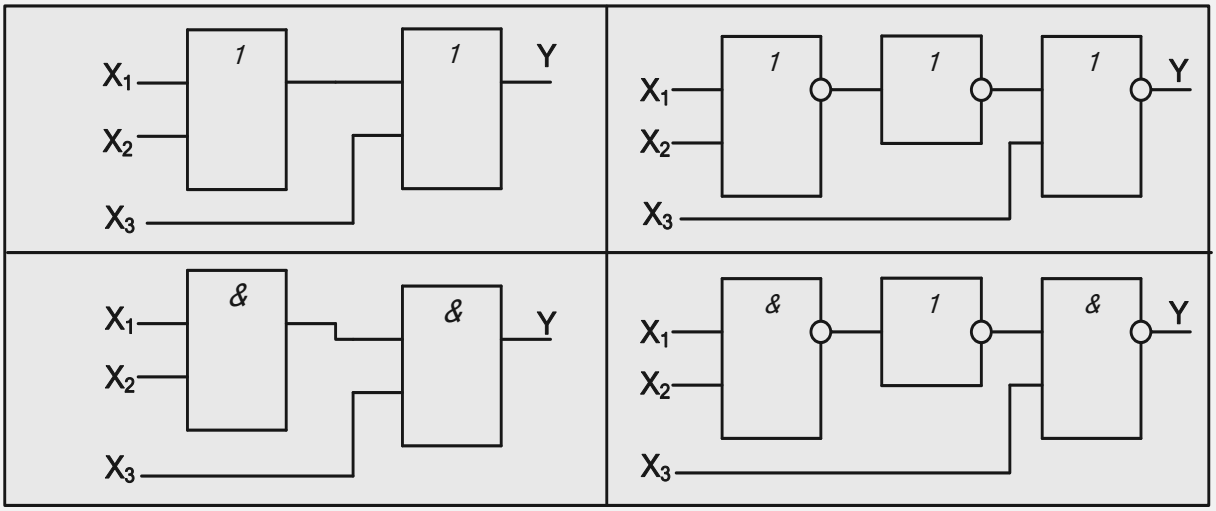


Рис. 1.18 – Побудова ЛЕ із трьома входами на основі елементів із двома входами

Більш докладно особливості використання реальних цифрових мікросхем, в тому числі і наборів ЛЕ, розглядаються в дисциплінах «Комп'ютерна електроніка» та «Комп'ютерна схемотехніка».

1.4. Типові логічні пристрої

До типових логічних пристроїв (комбінаційних цифрових вузлів) належать: шифратори і дешифратори; мультиплексори і демультиплексори; перетворювачі кодів; схеми порівняння кодових слів, комбінаційні суматори та ін.

1.4.1. Шифратори та дешифратори

Шифраторами називають комбінаційні ЦА, які призначені для перетворення однорозрядного позиційного коду на багаторозрядний код. Вхідний сигнал активного рівня на такому пристрої може з'явитися лише на одному з його входів; у той же час на його виході буде отримана комбінація декількох активних вихідних сигналів, що відповідає визначеному алгоритмові перетворення коду.

Розглянемо приклад синтезу шифратора, що виконує перетворення десяткового коду в натуральний двійковий.

Припустимо, що шифратор має 10 входів, послідовно пронумерованих десятковими цифрами (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), та n виходів. Подача сигналу на один із входів приводить до появи на виходах n-розрядного двійкового числа, яке відповідає десятковому номеру активного входу. Кількість виходів n може бути визначена як мінімальна, що достатня для подання найбільшої вхідної величини, тобто $9_{10} = 1001_2$; відповідно $n = 4$.

Взагалі, якщо кількість входів настільки велика, що в шифраторі використовуються всі можливі комбінації сигналів на виході, то такий шифратор називається повним, якщо не всі, то неповним. Число входів і виходів в повному шифраторі пов'язано співвідношенням $n = 2^m$, де n – число входів, m – число виходів.

Так, для перетворення десяткових цифр в чотирирозрядний двійковий код достатньо лише 10 входів, в той час як повне число можливих входів дорівнюватиме 16 ($n = 2^4 = 16$), тому шифратор 10 × 4 (з 10 в 4) буде неповним.

УГП такого шифратора зображено на рис. 1.19. Абревіатура *CD* в УГП утворена із двох літер англійського слова *coder*. Десяткові порядкові номери

входів та відповідні комбінації вихідних сигналів (натуральні двійкові коди чисел від 0 до 9) подані у табл. 1.9.

З наведеної таблиці функціонування шифратора випливає, що, наприклад, вихідна змінна X_1 матиме рівень логічної одиниці за умови, що одна із вхідних змінних Y_1, Y_3, Y_5, Y_7, Y_9 теж має цей рівень. Таким чином,

$$X_1 = Y_1 \vee Y_3 \vee Y_5 \vee Y_7 \vee Y_9 .$$

Аналогічно

$$X_2 = Y_2 \vee Y_3 \vee Y_6 \vee Y_7,\ X_4 = Y_4 \vee Y_5 \vee Y_6 \vee Y_7,\ X_8 = Y_8 \vee Y_9 .$$

Такій системі ЛФ відповідає схема, зображена на рис. 1.20.

Таблиця 1.9

Десяткове число Y	Двійковий код 8421			
	X_4	X_3	X_2	X_1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

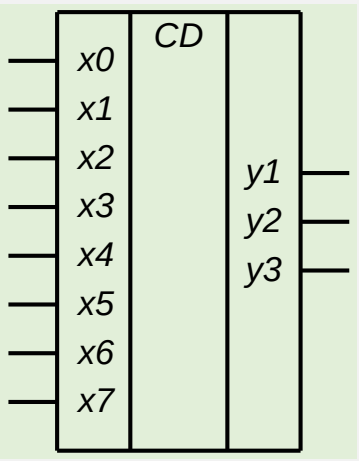


Рис. 1.19 – УГП шифратора

Для отримання схеми шифратора, побудованої на елементах І-НІ, треба застосувати до виразів, що відображають функціональну залежність сигналів на його виходах від сигналів на входах, правило де Моргана:

$$X_1 = \overline{\overline{Y_1} \cdot \overline{Y_3} \cdot \overline{Y_5} \cdot \overline{Y_7} \cdot \overline{Y_9}},$$

$$X_2 = \overline{\overline{Y_2} \cdot \overline{Y_3} \cdot \overline{Y_6} \cdot \overline{Y_7}},$$

$$X_4 = \overline{\overline{Y_4} \cdot \overline{Y_5} \cdot \overline{Y_6} \cdot \overline{Y_7}},$$

$$X_8 = \overline{\overline{Y_8} \cdot \overline{Y_9}}.$$

На рис. 1.21 зображена відповідна схема шифратора на елементах І-НІ.

У другій схемі необхідно передбачити подачу на входи інверсних значень сигналів, тобто для отримання на виході двійкового аналога певної десяткової цифри достатньо на відповідний вхід подати логічний нуль, а на решту входів – логічні одиниці.

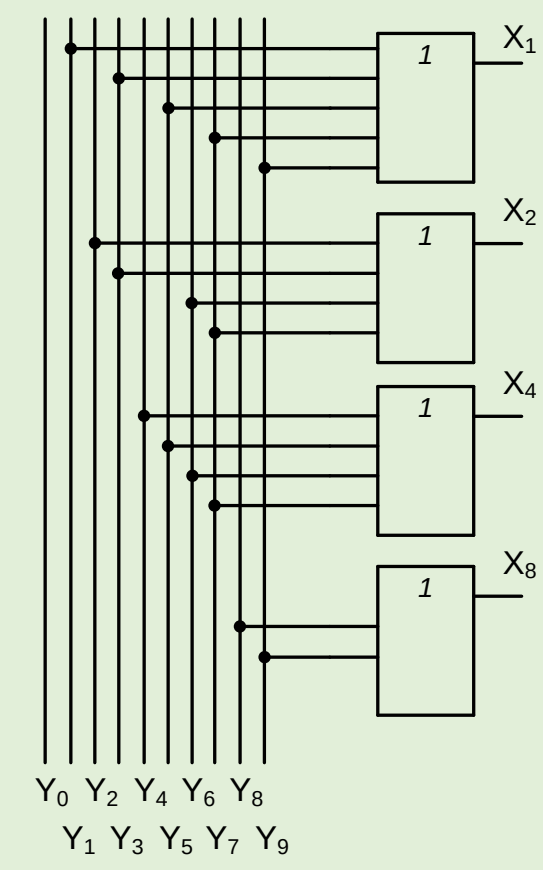


Рис. 1.20 – Схема шифратора на елементах АБО

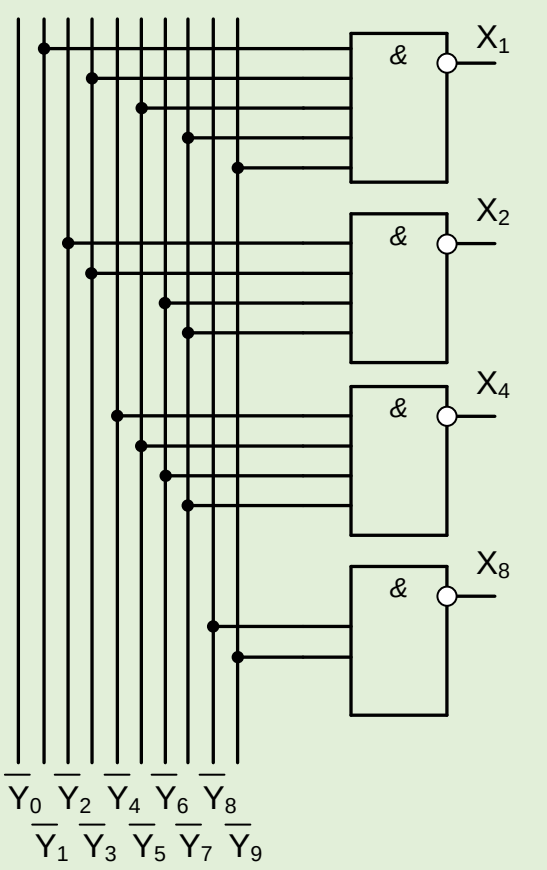


Рис. 1.21 – Схема шифратора на елементах І-НІ

Аналогічним чином можуть бути побудовані шифратори, які виконують перетворення десяткових чисел у двійкові з використанням будь-якого двійкового коду. Тому шифратори можна розглядати як окремий різновид перетворювачів кодів.

Шифратори широко застосовують у різноманітних пристроях ручного введення інформації до цифрових систем. Такі системи мають клавіатури, кожна клавіша яких пов’язана з певним входом дешифратора. При натисканні обраної клавіші подається сигнал на відповідний вхід шифратора, і на його виході виникає двійкове число, яке відповідає коду символу, що зображений на клавіші.

На практиці часто використовують шифратори з пріоритетом. У таких шифраторах код двійкового числа відповідає найвищому номеру входу, на який подано сигнал «1» (якщо вхідні сигнали надходять одночасно на декілька входів). На пріоритетний шифратор допускається подавати сигнали на кілька входів, а він виставляє на виході код числа, відповідного старшому входу.

Дешифратори (декодери) виконують зворотну порівняно з шифратором функцію та *призначені для перетворення багаторозрядних кодів на однорозрядний позиційний код (наприклад, двійкових чисел у десяткові).*

Входи дешифраторів призначені для подачі на них кодів двійкових чисел, а виходи послідовно пронумеровані десятковими числами. При подачі на вхід дешифратора двійкового числа сигнал активного рівня з'явиться лише на виході, який відповідає цьому двійковому числу.

Дешифратор – це комбінаційний ЦА, який у будь-який момент часу створює сигнал активного рівня тільки на одному виході – на тому, десятковий порядковий номер якого відповідає коду двійкового числа, створеного вхідними сигналами. Повний дешифратор має p входів та $m = 2^p$ виходів.

Складемо таблицю істинності для дешифратора на два входи, вважаючи, що активний рівень вихідних сигналів – рівень логічного нуля (табл. 1.10).

Таблиця 1.10

Вхідний код 8421		**Вихід**				**Логічна функція**
X_1	**X_0**	**D_3**	**D_2**	**D_1**	**D_0**	
0	0	1	1	1	0	$D_0 = X_1 + X_0$
0	1	1	1	0	1	$D_1 = X_1 + \overline{X_0}$
1	0	1	0	1	1	$D_2 = \overline{X_1} + X_0$
1	1	0	1	1	1	$D_3 = \overline{X_1} + \overline{X_0}$

У цьому випадку кожній з чотирьох комбінацій на вході буде відповідати рівень логічного нуля на одному з чотирьох виходів, на інших виходах будуть рівні логічних одиниць. Кількість виходів такого дешифратора $m = 4$. Очевидно, що ЛФ кожного із чотирьох виходів може бути подана як операція АБО над самими значеннями аргументів або їх інверсіями. ЛФ виходів дешифратора записані у відповідних рядках

табл. 1.10. Неважко помітити, що у кожній з функцій інвертуються аргументи, які мають у цьому вхідному наборі значення логічної одиниці.

Схема дешифратора, побудованого відповідно до отриманих виразів, подана на рис. 1.22. На рис. 1.23 наведені варіанти УГП дешифратора 3×8 (три входи, вісім виходів) на принципових схемах. Абревіатура *DC* утворена з двох літер слова *decoder*. Ліворуч показані входи із вказаними на них ваговими коефіцієнтами кожного двійкового входу. Праворуч – виходи, пронумеровані десятковими числами, що відповідають комбінаціям вхідного двійкового коду. Дешифратор може мати як однофазні (рис. 1.23, а), так і парафазні входи (рис. 1.23, б) для подачі як вхідних змінних, так і їх інверсій.

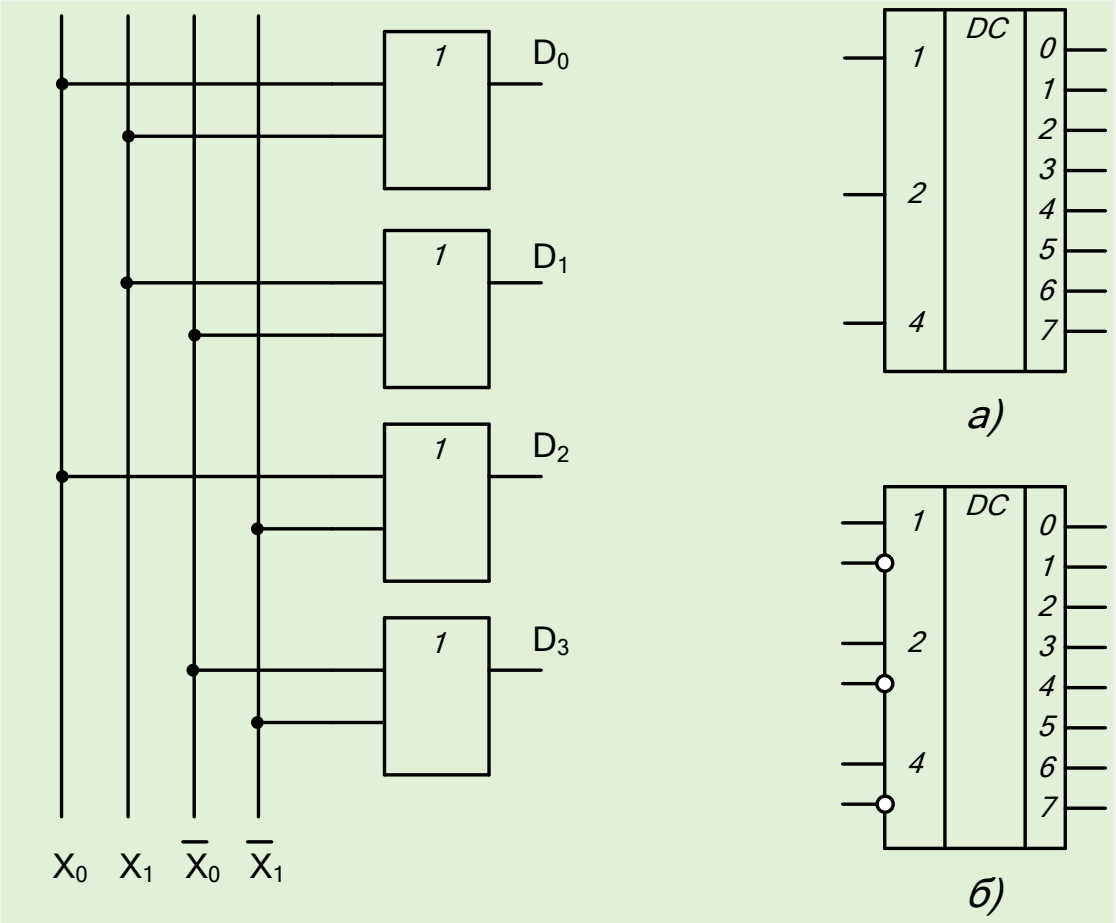


Рис. 1.22 – Схема дешифратора 2x4 з нульовим активним рівнем сигналів

Рис. 1.23 – Варіанти УГП дешифратора 3x8

Аналогічно можна отримати таблицю істинності й ЛФ дешифратора з одиничним активним рівнем вихідних сигналів (табл. 1.11). Для запису ЛФ у цьому випадку доцільно використовувати операцію І. Схема такого

дешифратора буде відрізнятися від зображеної на рис. 3.4 тільки типом використаних ЛЕ (І замість АБО) і зворотним порядком нумерації виходів.

Зрозуміло, що подібним чином можна побудувати схеми дешифраторів на будь-яку кількість входів. Але дешифратори такої структури, яка зветься лінійною, мають суттєвий недолік – для їх реалізації потрібна велика кількість ЛЕ. Так, наприклад, для реалізації повного дешифратора восьмирозрядних двійкових чисел потрібно $m = 2^8 = 256$ елементів АБО, кожний з яких повинен мати 8 входів.

Аналогічно можна отримати таблицю істинності й ЛФ дешифратора з одиничним активним рівнем вихідних сигналів (табл. 1.11). Для запису ЛФ у цьому випадку доцільно використовувати операцію І. Схема такого дешифратора буде відрізнятися від зображеної на рис. 3.4 тільки типом використаних ЛЕ (І замість АБО) і зворотним порядком нумерації виходів.

Таблиця. 1.11

Вхідний код 8421		**Вихід**				**Логічна функція**
X_1	**X_0**	**D_3**	**D_2**	**D_1**	**D_0**	
0	0	0	0	0	1	$D_0 = \overline{X}_1\overline{X}_0$
0	1	0	0	1	0	$D_1 = \overline{X}_1 X_0$
1	0	0	1	0	0	$D_2 = X_1\overline{X}_0$
1	1	1	0	0	0	$D_3 = X_1 X_0$

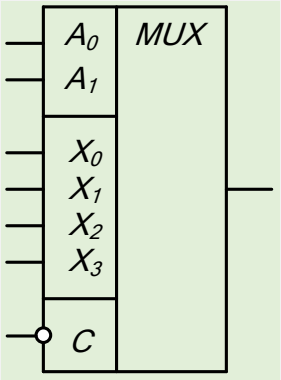
Зрозуміло, що подібним чином можна побудувати схеми дешифраторів на будь-яку кількість входів. Але дешифратори такої структури, яка зветься лінійною, мають суттєвий недолік – для їх реалізації потрібна велика кількість ЛЕ. Так, наприклад, для реалізації повного дешифратора восьмирозрядних двійкових чисел потрібно $m = 2^8 = 256$ елементів АБО, кожний з яких повинен мати 8 входів.

Лабораторна робота «Дослідження комбінаційних цифрових вузлів»

1.4.2. Мультиплексори і демультиплексори

Мультиплексором називається комбінаційний ЦА, який забезпечує комутацію на виході одного з декількох інформаційних вхідних сигналів відповідно до заданого коду на керуючих входах. Іноді мультиплексори називають комутаторами декількох входів на один вихід.

На рис. 1.24 подано УГП мультиплексора із чотирма інформаційними входами, функціонування котрого описується таблицею 1.12.



Таблиця 1.12

A_1	A_0	C	Y
0	0	0	X_0
0	1	0	X_1
1	0	0	X_2
1	1	0	X_3

Рис. 1.24 – УГП мультиплексора на чотири входи

Замість абревіатури *MUX* в УГП може використовуватися *MS* – мультиплексор-селектор.

Кожному інформаційному входу мультиплексора присвоюється двійковий порядковий номер (у цьому випадку 00…11), який називається адресою. При подачі синхронізуючого імпульсу на вхід *C* мультиплексор вибирає один з інформаційних входів, адреса котрого задається двійковим кодом на адресних входах A_0, A_1 і підключає його до виходу *Y*.

За відсутності синхронізуючого сигналу (відзначимо, що вхід *C* – інверсний, отже, його активне значення – нульове) зв'язок між інформаційними входами і виходом відсутній. Тому в табл. 1.12, яка містить алгоритм функціонування мультиплексора, рядки для $C = 1$, ($Y = 0$) відсутні.

Використовуючи табл. 1.12, запишемо реалізовану ЛФ у ДДНФ:

$$Y = \overline{A}_1\overline{A}_0\overline{C}X_0 + \overline{A}_1A_0\overline{C}X_1 + A_1\overline{A}_0\overline{C}X_2 + A_1A_0\overline{C}X_3.$$

Аналізуючи отриманий вираз, можна помітити, що для кожного входу *X* комбінації адресних сигналів A_1, A_0 у мультиплексорі такі ж, як у дешифраторі на два входи з одиничним активним рівнем вихідного сигналу (див. табл. 1.11). Отже, складовою частиною мультиплексора є дешифратор адресних сигналів, що забезпечує одержання адресних комбінацій $\overline{A}_1\overline{A}_0$, $\overline{A}_1A_0$, $A_1\overline{A}_0$, A_1A_0. З урахуванням цього отримуємо схему, що реалізовує потрібну ЛФ із застосуванням дешифратора, яка зображена на рис. 1.25.

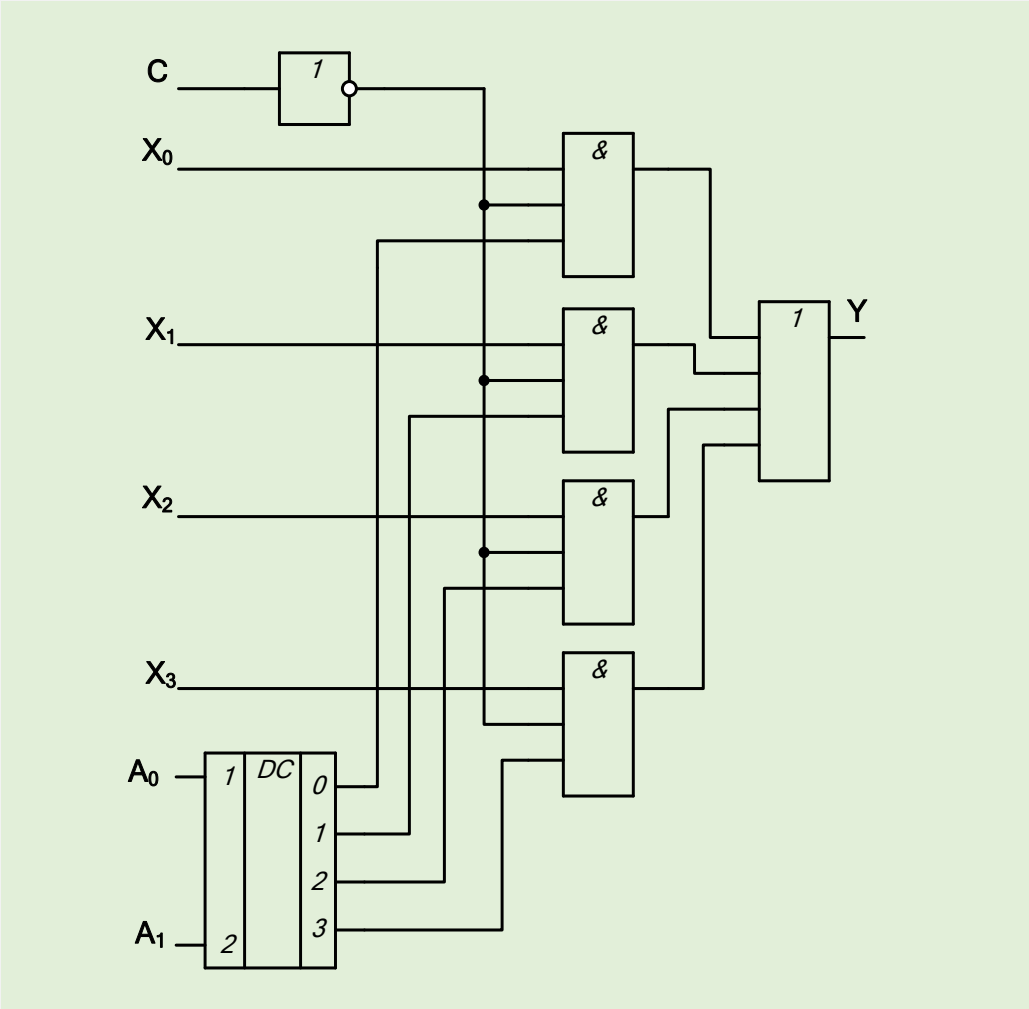


Рис. 1.25 – Мультиплексор на основі дешифратора

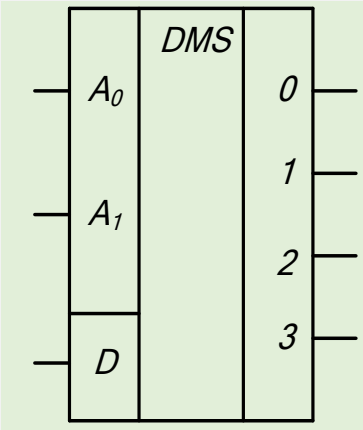
Мультиплексори, як і дешифратори, можуть застосовуватися як гоові частини для синтезу складних логічних пристроїв. На мультиплексорах з чотирма входами може бути реалізована будь-яка ЛФ трьох змінних; на мультиплексорах, що мають вісім входів, будь-яка ЛФ чотирьох змінних.

Для збільшення розрядності даних, що передаються, використовують паралельне з'єднання мультиплексорів.

Демультиплексор – це комбінаційний ЦА, який забезпечує комутацію інформаційного вхідного сигналу на один з декількох виходів, відповідно до заданого коду на керуючих входах. Таким чином, демультиплексор виконує функцію, зворотну мультиплексуванню.

На рис. 1.26 подано УГП демультиплексора (абревіатура *DMS* означає «демультиплексор-селектор») із чотирма інформаційними входами, функціонування котрого описується таблицею 1.13. У цій же таблиці наведені ЛФ для всіх інформаційних виходів.

З порівняння табл. 1.11 і 1.13 та аналізу ЛФ для виходів $Y_0 \ldots Y_3$ очевидно, що цей демультиплексор є також дешифратором із додатковим входом синхронізації *D*.



Таблиця 1.13

A1	A0	D	Вихід
0	0	1	$Y_0 = \overline{A}_1 \overline{A}_0 D$
0	1	1	$Y_1 = \overline{A}_1 A_0 D$
1	0	1	$Y_2 = A_1 \overline{A}_0 D$
1	1	1	$Y_3 = A_1 A_0 D$

Рис. 1.26 – УГП демультиплексора на чотири виходи

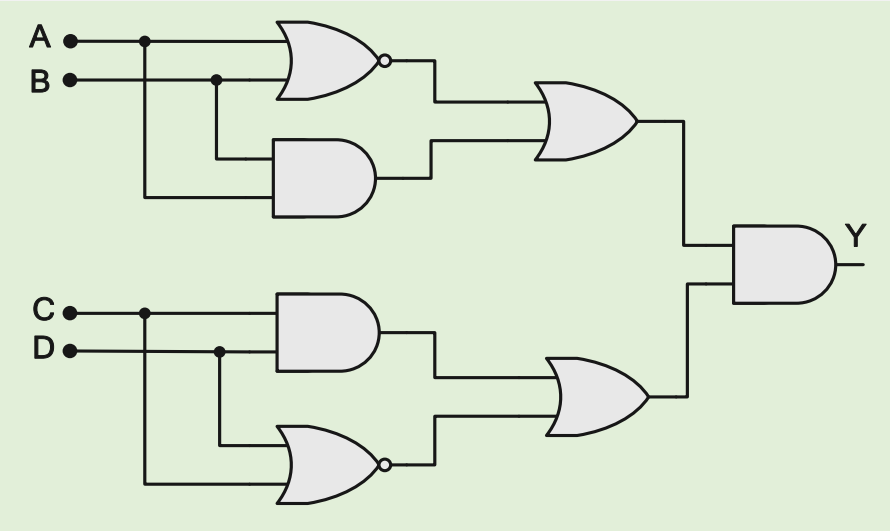
Тому демультиплексори будуються як синхронні дешифратори, інформаційні входи яких виконують функцію адресних входів демультиплексора, а вхід синхронізації – функцію інформаційного входу.

Мультиплексори і демультиплексори широко застосовуються для побудови мультиплексованих каналів передачі даних для передавання декількох цифрових сигналів одночасно по тім самим фізичним лініям зв’язку в режимі розподілу часу. Прикладом такого каналу є системна шина персонального комп’ютера.

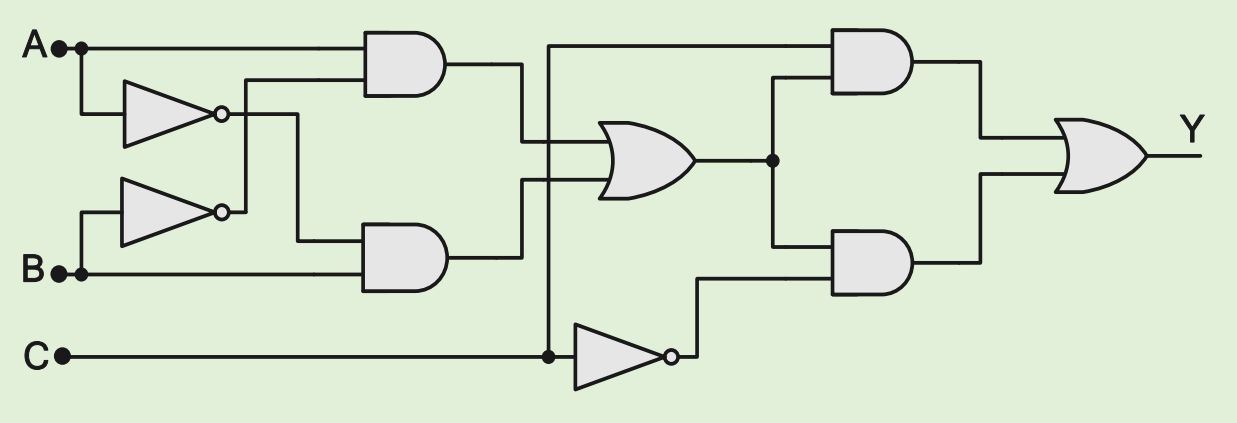
Більш докладно будова та застосування шифраторів, дешифраторів, мультиплексорів і демультиплексорів розглядаються в дисципліні «Комп’ютерна схемотехніка».

Запитання і завдання

1. Яка змінна називається логічною? Назвіть способи подання ЛФ.
2. Назвіть основні закони та правила алгебри логіки.
3. Сформулюйте та поясніть принцип дуальності.
4. Що називають функціонально повною системою ЛФ або логічним базисом? Наведіть приклади логічних базисів.
5. Наведіть і поясніть класифікацію логічних пристроїв.
6. Надайте приклади кон'юнктивної та диз'юнктивної нормальної форми подання логічних функцій. Коли КНФ та ДНФ вважаються досконалими?
7. Запишіть ЛФ, що реалізують схеми, зображені на рис. 1.27, та мінімізуйте їх методом тотожних перетворень.
8. Для функцій, отриманих при виконанні завдання 7, складіть таблиці істинності.
9. Спростить та перетворіть ЛФ $F = \overline{C} \cdot B \cdot \overline{A} \vee C \cdot B \cdot A \vee \overline{B} \cdot \overline{A}$ в базис І-НІ, зобразіть схему, яка цю функцію реалізовує.
10. Спростить та перетворіть ЛФ $F = \overline{D} \cdot \overline{B} \cdot \overline{A} \vee \overline{D} \cdot C \cdot B \vee \overline{D} \cdot \overline{C} \cdot B \cdot A$ в базис АБО-НІ, зобразіть схему, котра цю функцію реалізовує.
11. Що передбачає синтез логічного пристрою? Назвіть і поясніть основні етапи синтезу логічних пристроїв.
12. Сформулюйте та поясніть алгоритм отримання ДДНФ ЛФ на основі її таблиці істинності.
13. Сформулюйте та поясніть алгоритм отримання ДКНФ логічної функції на основі її таблиці істинності.
14. У чому полягає сутність мінімізації ЛФ методом карт Карно? Яких правил необхідно дотримуватися при виборі замкнутих областей на картах Карно?
15. Поясніть особливості синтезу логічних пристроїв із декількома виходами.



а)



б)

Рис. 1.27 – Схеми логічних пристроїв

16. Для функцій, які подані у табл. 1.14, знайдіть ДДНФ та МДНФ.

Таблиця 1.14

		Вхідні слова															
Аргументи	X_1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	X_2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	X_3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
	X_4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Функція	F_1	0	0	0	1	0	1	0	1	0	0	1	1	1	1	1	0
	F_2	0	0	1	0	0	1	1	1	0	1	1	0	0	1	1	0

17. Мінімізуйте ЛФ чотирьох аргументів, яка задана табл. 1.15, за допомогою карти Карно (з урахуванням невизначених значень, що позначені символом *X*).

Таблиця 1.15

		Вхідні слова															
Аргументи	X_1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	X_2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	X_3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
	X_4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Функція	F	1	X	1	X	0	1	0	1	0	1	0	0	0	1	0	1

Побудуйте схему, котра реалізовує отриману функцію, в базисах: а) І-НІ; б) АБО-НІ; в) І, НІ; г) АБО, НІ.

18. Поясніть особливості застосування ЛЕ з різною кількістю входів при побудові реальних логічних пристроїв.

19. Поясніть призначення й зобразіть логічну схему шифратора.

20. Поясніть призначення й зобразіть логічну схему найпростішого дешифратора (*а* – з нульовим активним рівнем вихідного сигналу; *б* – з одиничним активним рівнем вихідного сигналу).

21. Поясніть призначення й зобразіть логічну схему мультиплексора.

22. Поясніть призначення й зобразіть логічну схему демультиплексора.

Розділ 2. СИСТЕМИ ЧИСЛЕННЯ, ЩО ЗАСТОСОВУЮТЬСЯ В КОМП'ЮТЕРІ. КОДУВАННЯ ДВІЙКОВИХ ЧИСЕЛ ТА ПЕРЕТВОРЕННЯ КОДІВ

2.1. Системи числення, що застосовуються в комп'ютері

Як відомо, число – це величина, яка виражає кількість однорідних об'єктів. *Системою числення називають визначену сукупність символів – знаків і цифр – для запису чисел, а також правила їх запису.* Розрізняють позиційні й непозиційні системи числення.

У *непозиційних* системах числення значення кожної цифри не залежить від її позиції (розташування) у числі. Прикладом непозиційної системи є римська система числення. Недоліками непозиційних систем є необмежена кількість різних цифр, необхідних для подання чисел та складність виконання в них арифметичних операцій. Непозиційні системи числення в обчислювальній техніці не використовуються.

Позиційними називають такі системи числення, в яких значення кожної цифри у числі знаходяться у прямій залежності від її позиції в ряді цифр, що зображують число. Позиція визначається розташуванням цієї цифри щодо коми, яка відділяє цілу частину числа від дробової.

Будь-яке число у позиційній системі числення можна подати у вигляді

$$N = x_n q^n + \ldots + x_1 q^1 + x_0 q^0 + x_{-1} q^{-1} + \ldots + x_{-m} q^{-m},$$

де q – основа системи числення; $x_n \ldots x_{-m}$ – цифри від 0 до $q - 1$.

Наприклад, для десяткової системи числення число 621,75 можна записати у вигляді суми

$$621{,}75 = 6 \times 10^2 + 2 \times 10 + 1 + 7 \times 10^{-1} + 5 \times 10^{-2}.$$

Таким чином, щоб одержати значення числа, потрібно кожну цифру (розряд числа) помножити на число, яке називається вагою (або ваговим

коефіцієнтом) розряду. Вага окремих розрядів – це геометрична прогресія зі знаменником, що дорівнює основі системи числення.

В обчислювальній техніці використовують двійкову, вісімкову, шістнадцяткову, десяткову системи числення:. Всі вони належать до класу систем з безпосереднім поданням чисел, у яких кожній цифрі відповідає окремий символ. Наприклад, у двійковій системі числення використовують лише дві цифри – 0 і 1, у вісімковій – вісім: 0, 1, 2, 3, 4, 5, 6, 7; у десятковій – десять: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9; у шістнадцятковій – шістнадцять: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *A, B, C, D, E, F*. Таким чином, для таких систем числення основа системи *q* дорівнює кількості різних символів, що використовують для подання цифр.

Як правило, у всіх системах числення з основою менше 10 для подання цифр застосовуються арабські символи, а в системах числення з основою більше 10 – ще й літери латинського алфавіту.

Двійкова система числення є основною системою подання інформації у цифрових пристроях взагалі та в комп’ютерах зокрема внаслідок простоти реалізації логічних 0 і 1 у вигляді двох рівнів напруги – низького рівня (*L*-рівня) та високого рівня (*H*-рівня). Двійкові цифри 0 і 1 називаються бітами (*bit –BInary digiT*). Один *біт* – це найменша кількість інформації, що відповідає одному з рівно можливих повідомлень (так-ні). Повідомлення, в якому міститься набір з декількох бітів, називається *словом*. Слово з 8 бітів називається *байтом*. Але двійкова система незручна для людини: внаслідок малої кількості цифр числа виходять довгими та «одноманітними».

Десяткова система в комп’ютерах використовується в пристроях введення-виведення, тобто для «спілкування» комп’ютера з людиною-оператором.

Вісімкова система числення виконує в обчислювальній техніці допоміжну роль і використовується для компактного запису двійкових кодів чисел та машинних команд комп’ютера. Відмінною рисою системи є те, що її основою є третій степінь числа 2, тому для подання одного вісімкового розряду потрібні три двійкових розряди (тріада).

Шістнадцяткова система числення в обчислювальній техніці використовується з тією ж метою, що й вісімкова. Унаслідок того, що основою шістнадцяткової системи є четвертий ступінь числа 2, для подання одного шістнадцяткового розряду потрібні чотири двійкові розряди (тетрада). У сучасних комп’ютерах шістнадцятковій системі віддається

перевага над вісімковою, тому що вона забезпечує більш компактне подання інформації (числа мають меншу кількість розрядів).

Фактично вісімкову та шістнадцяткову системи можна розглядати як «буферні» або «інтерфейсні» між людиною й комп'ютером – числа в цих системах порівняно легко сприймаються людиною, а їх переклад у двійкову систему, на відміну від десяткових, не викликає труднощів у комп'ютера.

Також в комп'ютерах використовується двійково-десятковий код (ДДК), котрий іноді називають двійково-десятковою системою числення. ДДК, як і двійкова система числення, призначений для кодування десяткових чисел послідовністю нулів та одиниць, але він належить до класу систем числення з кодованим поданням чисел. У таких системах кількість символів менша, ніж кількість цифр, а кожну цифру кодують певною комбінацією кількох символів. У ДДК для подання кожної з десятьох цифр десяткової системи використовують перші десять чотирирозрядних двійкових чисел-тетрад від 0000 до 1001. Шість старших кодових комбінацій, що залишилися, є забороненими.

Приклад: $73_{(10)} = 1001001_{(2)} = 0111\,0011_{(2-10)} = 111_{(8)} = 49_{(16)}$.

З прикладу видно, що, незважаючи на зовнішню схожість двійково-десяткового числа з двійковим числом, вони є різними формами подання чисел.

Запишемо двійкове число 1001001 у вигляді суми розрядів, помножених на їх вагові коефіцієнти,

$$1001001_{(2)} = 1\times 2^6 + 0\times 2^5 + 0\times 2^4 + 1\times 2^3 + 0\times 2^2 + 0\times 2^1 + 1\times 2^0.$$

64 32 16 8 4 2

Такий спосіб двійкового подання чисел часто називають *натуральним кодом або кодом 8421* (назва складена з вагових коефіцієнтів розрядів двійкового числа). Він є природною формою подання двійкових чисел.

У табл. 2.1 наведено подання десяткових чисел від 0 до 15 у різних системах числення.

Ця таблиця корисна для виконання перетворення цілих чисел з однієї системи числення в іншу.

Метод, котрий використовують для перетворення чисел із десяткової системи числення у двійкову і навпаки, залежить від системи, у якій виконуються арифметичні операції, необхідні для перетворення. При перетворенні чисел вручну операції будуть виконуватися у десятковій

системі числення. Якщо перетворення виконується цифровим пристроєм, то арифметичні операції будуть виконуватися над числами, поданими у двійковій системі числення.

Таблиця 2.1

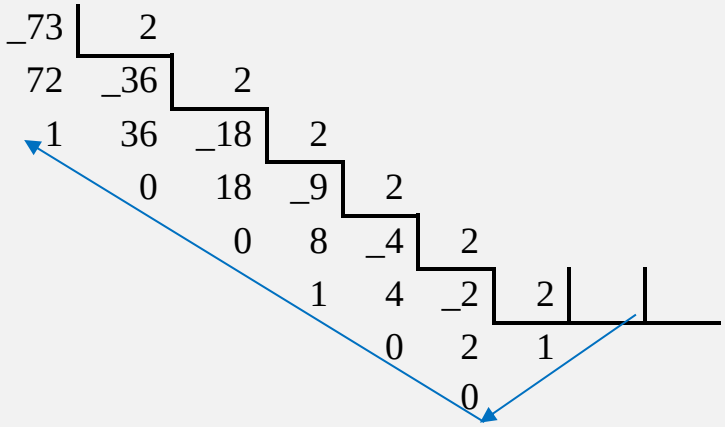
Основа системи числення				
10	**2**	**8**	**16**	**2-10**
0	0	0	0	0000
1	1	1	1	0001
2	10	2	2	0010
3	11	3	3	0011
4	100	4	4	0100
5	101	5	5	0101
6	110	6	6	0110
7	111	7	7	0111
8	1000	10	8	1000
9	1001	11	9	1001
10	1010	12	A	0001 0000
11	1011	13	B	0001 0001
12	1100	14	C	0001 0010
13	1101	15	D	0001 0011
14	1110	16	E	0001 0100
15	1111	17	F	0001 0101

Для перетворення цілих чисел з однієї системи числення в іншу потрібно: записати число у вихідній системі й послідовно цілочисельно ділити його на основу нової системи, подану у вихідній системі числення; одержати частку й остачу; отриману частку знову ділити на ту ж основу; знову знайти частку й остачу. Так треба продовжувати доти, доки частка не стане меншою за основу, на яку ділили, – це буде остання остача. Тепер треба записати один за одним усі отримані остачі, починаючи з останньої. Отриманий запис дає шукане число у новій системі.

Це правило діє для позиційних систем числення з будь-якою основою. Особливо воно зручне для перетворення чисел із десяткової системи числення в інші, тому що при цьому арифметичні операції виконуються у десятковій системі.

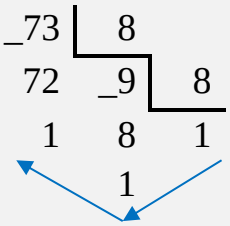
Приклади:

1. Перетворення десяткового числа на двійкове.



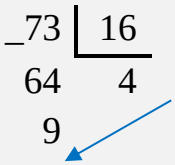
Відповідь: $73_{(10)} = 1001001_{(2)}$.

2. Перетворення десяткового числа на вісімкове.



Відповідь: $73(10) = 111_{(8)}$.

3. Перетворення десяткового числа на шістнадцяткове.



Відповідь: $73_{(10)} = 49_{(16)}$.

Для переведення чисел у десяткову систему з інших систем числення доцільно користуватися таким алгоритмом: записати над розрядами вихідного числа десяткові вагові коефіцієнти цих розрядів; помножити коефіцієнти на значення розрядів (крім розрядів із нульовими значеннями); додати отримані добутки.

Приклади:

64	32	16	8	4	2	1	
1	0	0	1	0	0	$1_{(2)}$	$= 64 + 8 + 1 = 73_{(10)}$;

64	8	1	
1	1	$1_{(8)}$	$= 64 + 8 + 1 = 73_{(10)}$;

16	1	
4	$9_{(16)}$	$= 64+9 = 73_{(10)}$.

Основи вісімкової й шістнадцяткової систем числення можуть бути подані цілим ступенем двійки. Цим пояснюється простота перетворення чисел, поданих у цих системах, у двійкову систему числення і навпаки.

Для перетворення чисел із вісімкової системи числення на двійкову достатньо кожну цифру вісімкового числа подати як трирозрядне двійкове число (тріаду).

Приклад: $762_{(8)} = 111\ 110\ 010_{(2)}$.

⇩ ⇩ ⇩

7 6 2

Переведення шістнадцяткових чисел у двійкову систему числення досягається поданням цифр шістнадцяткового числа чотирирозрядними двійковими числами (тетрадами).

Приклад: $A7B_{(16)} = 1010\ 0111\ 1011_{(2)}$.

⇩ ⇩ ⇩

A 7 B

При зворотному переведенні цілих чисел із двійкової системи у вісімкову або шістнадцяткову треба розряди двійкового числа, рахуючи їх справа наліво, розбити на тріади (у випадку переведення у вісімкову систему) чи на тетради (у випадку переведення у шістнадцяткову систему). Якщо старша (крайня зліва) група є неповною, то вона доповнюється нулями. Потім кожна двійкова група подається цифрою тієї системи числення, у яку переводиться число.

Приклади: $001\ 111 = 17_{(8)}$, $0101\ 1100 = 5C_{(16)}$.

⇩ ⇩ ⇩ ⇩

1 7 5 C

Переведення десяткових чисел у ДДК і зворотне перетворення виконується аналогічно перетворенню шістнадцяткових чисел на двійкові й навпаки (із використанням двійкових тетрад).

Приклад: $735_{(10)} = 0111\ 0011\ 0101_{(2\text{-}10)}$.

⇩ ⇩ ⇩

7 3 5

Для чисел, що мають як цілу, так і дробову частину, переведення з однієї системи числення в іншу здійснюється окремо для цілої та дробової частин. Для перетворення правильного дробу X, записаного у системі числення з основою p, на його еквівалент у системі числення з основою q

слід послідовно множити X на q, причому множити потрібно тільки дробові частини (метод послідовного множення). Еквівалент X у системі числення з основою q подається у вигляді послідовності цілих частин результатів множення у порядку їхнього одержання, причому старший розряд є цілою частиною першого результату. Якщо необхідно виконати перетворення з точністю q^{-k}, то кількість послідовних множень дорівнює k.

Наприклад, необхідно записати десятковий дріб $0{,}366_{10}$ з точністю 2^{-8} у двійковій системі числення. Для цього дробові частини заданого числа та чисел, що утворюються у результаті множення, вісім разів послідовно множимо на 2:

0,366*2 = 0,732 ціла частина 0
0,732*2 = 1,464 ціла частина 1
0,464*2 = 0,928 ціла частина 0
0,928*2 = 1,856 ціла частина 1
0,856*2 = 1,712 ціла частина 1
0,712*2 = 1,424 ціла частина 1
0,424*2 = 0,848 ціла частина 0
0,848*2 = 1,696 ціла частина 1

Результат: $0{,}366_{10} = 0{,}01011101_2$.

Зворотне перетворення двійкового числа на десятковий еквівалент може бути виконано, як звичайно, множенням вагових коефіцієнтів розрядів числа на значення розрядів та додаванням отриманих добутків

$$0{,}01011101_2 = 0 \times 2^0 + 2^{-1} \times 0 + 2^{-2} \times 1 + 2^{-3} \times 0 + 2^{-4} \times 1 + 2^{-5} \times 1 + 2^{-6} \times 1 +$$
$$+ 2^{-7} \times 0 + 2^{-8} \times 1 = 0 \times 1 + 0{,}5 \times 0 + 0{,}25 \times 1 + 0{,}125 \times 0 + 0{,}0625 \times 1 +$$
$$+ 0{,}03125 \times 1 + 0{,}015625 \times 1 + 0{,}0078125 \times 0 + 0{,}00390625 \times 1 = 0{,}36328125_{10}.$$

Отримане число $0{,}36328125_{10}$ наближує задане число 0,366 з точністю $2^{-8} = 0{,}0039$.

Практичне завдання «Перетворення чисел в різні позиційні системи числення»

2.2. Кодування двійкових чисел та перетворення кодів

У цифрових пристроях часто виникає необхідність перетворення інформації з одного двійкового коду в інший. Коди, що відрізняються від найбільш простого натурального 8421, наприклад, застосовуються:

- ✓ у цифрових пристроях, комп'ютерах та системах передачі даних для виявлення і корекції помилок (код з контролем на парність; код Хеммінга, циклічні коди);
- ✓ у перетворювачах аналогових фізичних сигналів у цифрові сигнали для забезпечення погрішності перетворення, що не перевищує одиниці молодшого розряду (код Грея);
- ✓ при виконанні арифметичних операцій в комп'ютерах (прямий, обернений, доповняльний коди); для введення в комп’ютер даних (ДДК);
- ✓ для побудови цифрових індикаторів (семисегментний код).

Для синтезу перетворювачів кодів використовуються два методи:

- ✓ перетворення вихідного двійкового коду в десятковий і наступне перетворення десяткового коду в потрібний двійковий код;
- ✓ використання комбінаційного логічного пристрою, що безпосередньо реалізовує необхідне перетворення.

Перший метод структурно реалізовується з’єднанням дешифратора і шифратора та є зручним у випадках, коли можна використовувати стандартні дешифратори і шифратори в інтегральному виконанні. Використання другого методу нерідко зменшує апаратні витрати на реалізацію перетворювача.

3.2.1. Синтез перетворювача натурального двійкового коду в код Грея

Розглянемо синтез перетворювача другим методом на прикладі трирозрядного перетворювача натурального двійкового коду у код Грея, що також називають циклічним або рефлексно-двійковим.

Особливістю коду Грея є те, що в ньому кодові комбінації двох сусідніх чисел різняться тільки в одному двійковому розряді. Відповідність трирозрядних двійкових чисел їх кодам Грея показано в табл. 2.2.

Таблиця 2.2

Код 8421			Код Грея		
X_2	X_1	X_0	Y_2	Y_1	Y_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

Безпосередньо з таблиці запишемо ЛФ виходів Y_0 і Y_1 у ДДНФ:

$$Y_0 = \overline{X}_2\overline{X}_1X_0 + \overline{X}_2X_1\overline{X}_0 + X_2\overline{X}_1X_0 + X_2X_1\overline{X}_0,$$
$$Y_1 = \overline{X}_2X_1\overline{X}_0 + \overline{X}_2X_1X_0 + X_2\overline{X}_1\overline{X}_0 + X_2\overline{X}_1X_0.$$

Застосовуючи до отриманих виразів правило склеювання, маємо (у першому виразі склеюються перший і третій, другий і четвертий додаток, у другому – перший і другий, третій і четвертий):

$$Y_0 = X_1\overline{X}_0 + \overline{X}_1X_0 = X_0 \oplus X_1,$$
$$Y_1 = \overline{X}_2X_1 + X_2\overline{X}_1 = X_1 \oplus X_2.$$

Що стосується функції виходу Y_2, то з таблиці видно, що

$$Y_2 = X_2.$$

Таким чином, найпростіше цей перетворювач реалізовується на двох суматорах за модулем два. Відповідна схема зображена на рис. 2.1, а.

У цілому, схема *n*-розрядного перетворювача натурального двійкового коду в код Грея може бути побудована на основі такого правила: старші розряди вхідного та вихідного кодів збігаються, а будь-який наступний розряд Y_k коду Грея дорівнює сумі по модулю два відповідного X_k та попереднього X_{k-1} розрядів натурального коду

$$Y_k = X_k \oplus X_{k-1}.$$

Узагальнений варіант УГП перетворювача кодів показаний на рис. 2.1, б. Допускається заміна літер X і Y позначеннями типу подання

вхідної і вихідної інформації відповідно. Зокрема, для двійкового коду (стандартного) можна використовувати літеру *B*, для десяткового коду – скорочення *DEC*, для коду Грея – *G*, для семисегментного коду – 7*S* і т.д. Таким чином, позначення синтезованого трирозрядного перетворювача може мати вигляд, зображений на рис. 2.1, в.

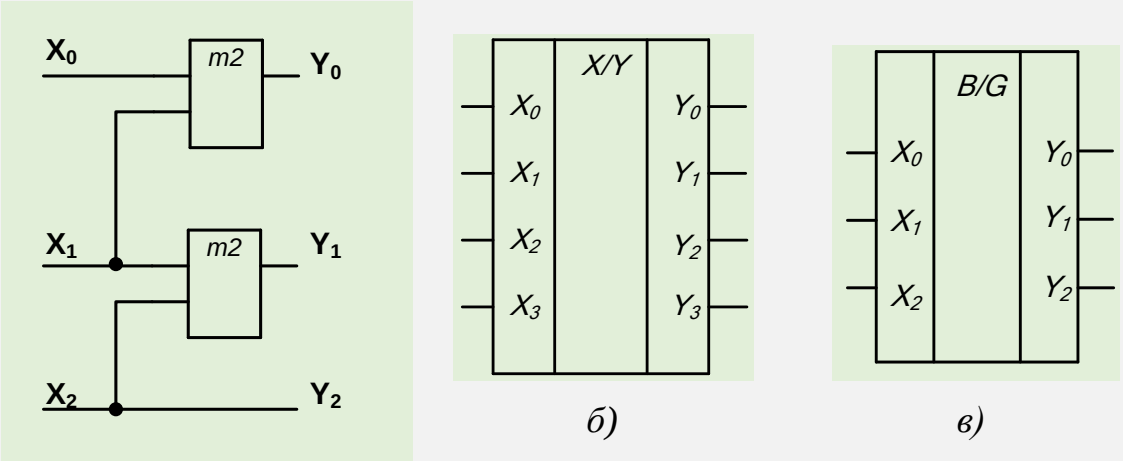


Рис. 2.1 – Схема та УГП трирозрядного перетворювача натурального коду у код Грея: а – схема, б – загальне УГП трирозрядного перетворювача кодів, в – УГП трирозрядного перетворювача натурального коду у код Грея

3.2.2. Поняття про завадостійке кодування кодових слів

Внаслідок дії завад під час передавання, обробки та збереження двійкових кодів можуть статися помилки, наприклад, прийом логічної одиниці замість логічного нуля чи навпаки.

Одним з найбільш ефективних шляхів захисту інформації є *кодування, стійке до завад*, яке здійснюється введенням у кодові слова додаткових бітів, призначених або для виявлення і виправлення помилок, або тільки для виявлення помилок. Відповідно до цього коди, стійкі до завад, поділяють на коригуючі, котрі виявляють і виправляють помилки, та коди, які тільки виявляють помилки.

Можливість виявлення помилок за наявності додаткових бітів обумовлена тим, що для передавання інформації використовуються не всі можливі комбінації *n*-розрядного двійкового коду, а лише деяка частина з них. Дозволені комбінації вважаються безпомилковими, інші є забороненими. Поява заборонених комбінацій розглядається як помилка.

Слід уявляти, що можливі такі помилки, за яких одна дозволена комбінація переходить в іншу. У цьому випадку помилки не виявляються.

Найпростішим поширеним кодом, стійким до завад, який використовується у мікропроцесорній техніці, є код з контролем на парність. У цьому коді до інформаційних бітів праворуч додається один контрольний біт. Якщо кількість одиниць в інформаційних бітах є парною, то значення контрольного біта дорівнює 0, у протилежному випадку – 1. Отже, у будь-якому випадку кількість одиниць у повній послідовності (кодовому слові з контрольним бітом) є парною. Якщо під час перевірки після передавання кількість одиниць є непарною, то це означає, що відбулася помилка. Код із контролем на парність дозволяє виявляти всі помилки непарної кратності (в одному біті кодового слова) і не дозволяє виявляти помилки парної кратності (у двох бітах кодового слова одночасно).

Неважко помітити, що алгоритм отримання додаткового контрольного біта для *n*-розрядного паралельного кодового слова збігається з ЛФ «Виключне АБО» *n*-аргументів.

Схема отримання контрольного біта *P* також може бути реалізована на суматорах за модулем 2 (елементах «Виключне АБО» на 2 входи). У цьому разі схема будується за багатоярусним принципом: спочатку попарно додаються значення окремих розрядів кодового слова, потім отримані результати також попарно додаються за допомогою суматорів за модулем 2 другого ярусу і т.д., наприклад:

$$P = (((a_1 \oplus a_2) \oplus (a_3 \oplus a_4)) \oplus ((a_5 \oplus a_6) \oplus (a_7 \oplus a_8)) \oplus \ldots$$

Багатоярусні схеми на суматорах за модулем 2 часто називають схемами згортання. На рис. 2.2 зображена схема отримання двійкового коду з контролем на парність, де схема згортання позначена прямокутником з написом $2k + 1$.

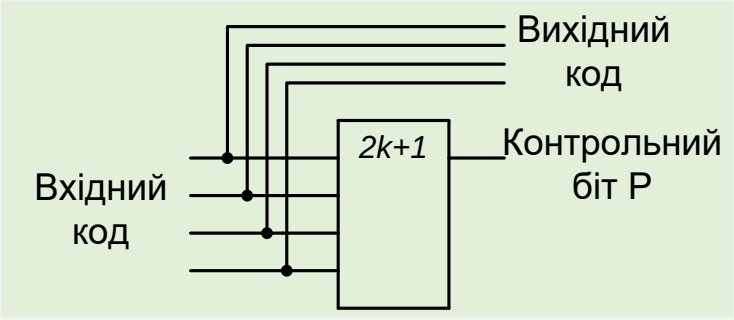


Рис. 2.2 – Схема отримання двійкового коду з контролем на парність

Очевидно, що схема, яка перевіряє на парність прийняті кодові слова (схема контролю на парність), також являє собою ЛЕ «Виключне АБО» $n + 1$ аргументів (з урахуванням контрольного біта).

Іншим поширеним кодом є код Хеммінга, що виявляє і виправляє одноразові помилки. Кожній з $2^n - 1$ ненульових комбінацій n-розрядного кодового слова відповідає комбінація з $n + k$ бітів. Значення контрольних бітів отримують у результаті додавання за модулем 2 значень бітів у деяких визначених інформаційних розрядах. Із загальної кількості $2^{n+k} - 1$ можливих помилок код Хеммінга може виявити та виправити $2^k - 1$ помилок.

Припустимо, що треба передати або обробити 15 різних двійкових повідомлень. Без кодування для цього достатньо чотирьох інформаційних бітів ($n = 4$). Потрібну кількість додаткових контрольних бітів обчислюють за формулою $2^k - 1 = n + k$, звідки визначають кількість перевірних розрядів та кількість одноразових помилок, які можуть бути виявлені та виправлені. У цьому випадку кількість додаткових розрядів $k = 3$, а кількість одноразових помилок – $2^k - 1 = 7$.

Контрольні біти k_i розташовують у послідовності інформаційних бітів u_j на позиціях із номерами 2^{i-1}, як показано у табл. 2.3.

Таблиця 2.3

Позиція	**1**	**2**	**3**	**4**	**5**	**6**	**7**
	001	010	011	100	101	110	111
Біт	k_1	k_2	u_1	k_3	u_2	u_3	u_4

Значення перевірних бітів k_i обчислюється додаванням за модулем 2 значень бітів, у двійковому виразі номерів яких наявна одиниця в i-му розряді. Відповідно для обчислення значення k_1 потрібно додати за модулем 2 значення бітів із непарними номерами

$$k_1 = u_1 \oplus u_2 \oplus u_4.$$

Для визначення k_2 треба додати за модулем 2 біти, у двійковому виразі номерів яких наявна одиниця у другому розряді, тобто

$$k_2 = u_1 \oplus u_3 \oplus u_4.$$

Контрольний біт k_3 визначається додаванням за модулем 2 бітів, у двійковому виразі номерів котрих наявна одиниця у третьому розряді,

$$k_3 = u_2 \oplus u_3 \oplus u_4.$$

Схема перетворювача чотирирозрядних кодових слів у код Хеммінга зображена на рис. 2.3.

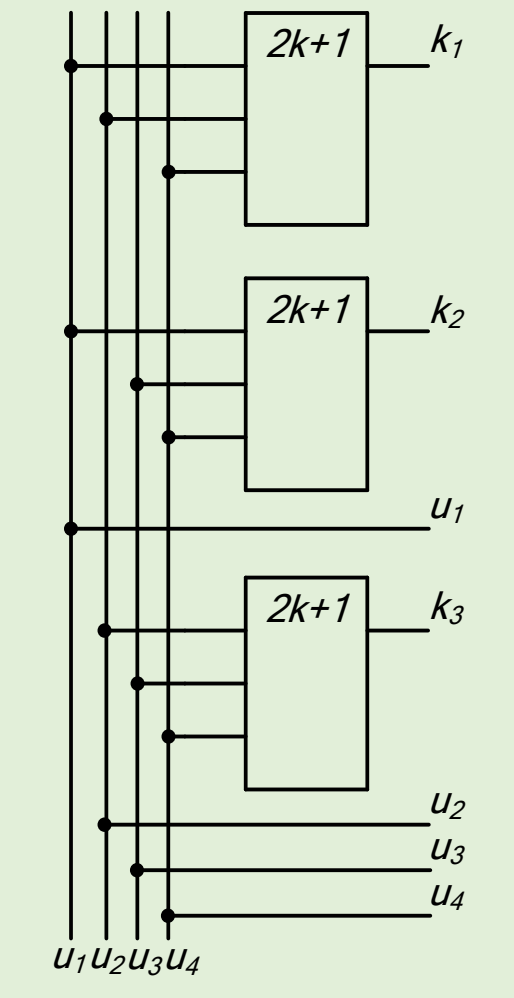


Рис. 2.3 – Схема перетворювача чотирирозрядних кодових слів у код Хеммінга

Визначення та виправлення помилок здійснюється *k*-перевірками. При кожній перевірці додаються за модулем 2 біти прийнятої послідовності інформаційних та контрольних розрядів, двійкові номери яких мають одиницю в першому, другому і подальших розрядах. Якщо під час передавання не було збою, то результати всіх перевірок дорівнюють нулю. Якщо збій відбувся, то хоча б одна перевірка не дорівнює нулю. У цьому випадку треба сформувати кодову комбінацію з результатів перевірок, яка буде вказувати на розряд, де відбувся збій (він має назву *синдром*). Молодший розряд коду результатів перевірок формує перша перевірка,

старший – остання. Інверсія біта в розряді з одержаним номером виправить помилку.

Наприклад, необхідно сформувати код Хеммінга, що виявляє та виправляє одноразові помилки у послідовності

1	1	0	0
u_1	u_2	u_3	u_4

Відповідно

$$k_1 = u_1 \oplus u_2 \oplus u_4 = 1 \oplus 1 \oplus 0 = 0,$$
$$k_2 = u_1 \oplus u_3 \oplus u_4 = 1 \oplus 0 \oplus 0 = 1,$$
$$k_3 = u_2 \oplus u_3 \oplus u_4 = 1 \oplus 0 \oplus 0 = 1.$$

Послідовність, що закодована кодом Хеммінга, буде мати вигляд

0	1	1	1	1	0	0
k_1	k_2	u_1	k_3	u_2	u_3	u_4

Нехай після передачі відбувся збій в одному розряді та прийнята послідовність 0110100 (помилка в четвертому розряді – k_3). Тоді перша й друга перевірки дадуть значення 0, а третя – 1:

$$k_1 \oplus u_1 \oplus u_2 \oplus u_4 = 0 \oplus 1 \oplus 1 \oplus 0 = 0,$$
$$k_2 \oplus u_1 \oplus u_3 \oplus u_4 = 1 \oplus 1 \oplus 0 \oplus 0 = 0,$$
$$k_3 \oplus u_2 \oplus u_3 \oplus u_4 = 0 \oplus 1 \oplus 0 \oplus 0 = 1.$$

Код 100, що створює результати перевірок, указує, що відбувся збій у четвертому розряді. Якщо проінвертувати четвертий розряд, то одержимо виправлену послідовність 0111100.

Практичне завдання «Завадостійке кодування двійкових чисел»

Із розглянутого прикладу видно, що схема пристрою контролю повинна містити такі складові частини: схеми згортання (елементи «Виключне АБО») відповідно до кількості перевірок, що виконуються; дешифратор, який на основі отриманого синдрому керує інверсією помилкового розряду та елементів, що виконують саму інверсію. Така схема зображена на рис. 2.4 (схеми згортання позначені прямокутником з написом

$2k + 1$). Як «керовані інвертори» у схемі використовуються суматори за модулем 2.

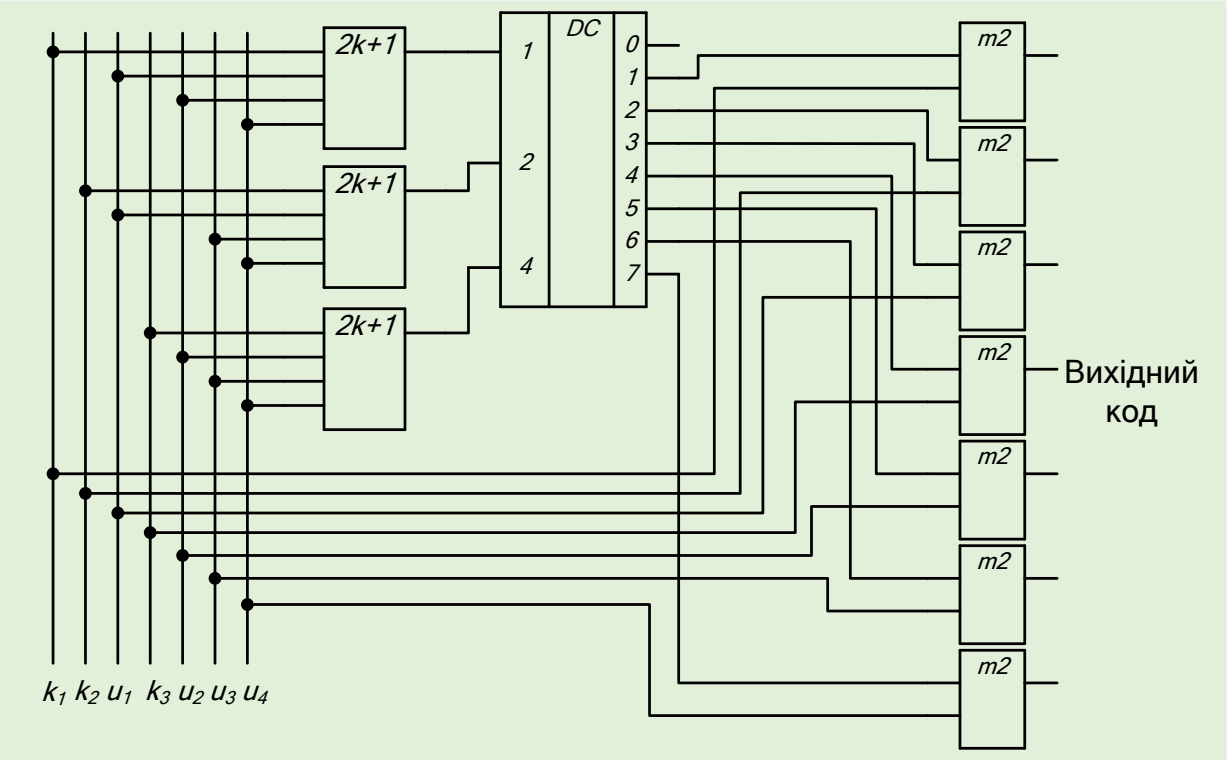


Рис. 2.4 – Схема пристрою контролю

Аналізуючи схеми перетворювачів кодів і схем контролю, які розглядалися вище, неважко помітити, що в більшості з них використовуються суматори за модулем 2 або схеми згортання, побудовані на основі цих суматорів.

3.2.3. Порівняння кодових слів

Різновидом схем контролю є *компаратори слів* (цифрові схеми порівняння) – це комбінаційні цифрові вузли, які *виконують функцію порівняння двох кодових слів визначеної розрядності*. Основними операціями, що виконуються схемами порівняння, є визначення ознаки рівності або нерівності двох n-розрядних чисел, причому операція порівняння може супроводжуватися визначенням знаку нерівності.

Як приклад, розглянемо синтез схеми порівняння двох трирозрядних кодових слів $X_2X_1X_0$ і $Y_2Y_1Y_0$, яка має три виходи ($Y = X$, $Y > X$, $Y < X$), за умови, що активний рівень сигналів – логічна 1.

Таблиця істинності схеми буде мати такий вигляд:

Таблиця 2.3

№ набору	X_2	X_1	X_0	Y_2	Y_1	Y_0	$Y = X$	$Y < X$	$Y > X$
0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	1	0	0	1
2	0	0	0	0	1	0	0	0	1
...	...	...	...	...	...	...	…	…	…
56	1	1	1	0	0	0	0	1	0
57	1	1	1	0	0	1	0	1	0
...	...	...	...	...	...	...	…	…	…
63	1	1	1	1	1	1	1	0	0

Очевидно, що синтез схеми традиційним методом або поданням вихідних ЛФ у вигляді ДДНФ та подальшою їх мінімізацією буде занадто складним, оскільки потребує мінімізації трьох функцій шести змінних. Тому виконаємо синтез схеми з використанням евристичних прийомів і без визначення обмежень на застосування елементів тих чи інших типів.

По-перше, очевидно, що схема, яка виконує функцію $Y = X$, може бути реалізована порозрядним порівнянням слів $X_2X_1X_0$ та $Y_2Y_1Y_0$, за допомогою елементів рівності та елемента «І» (якщо всі розряди двох кодових слів попарно дорівнюють один одному, то і кодові слова в цілому еквівалентні). Відповідна схема зображена на рис. 2.5.

По-друге, будь-яка з трьох вихідних функцій може бути виражена через дві інші. Наприклад, якщо Y не менше X та Y не дорівнює X, то $Y > X$. Це твердження можна подати таблицею істинності (табл. 2.4). Отримана таблиця істинності може бути реалізована елементом АБО-НІ на два входи, як показано на рис. 2.6.

Таблиця 2.4

Вхідні функції		Вихідна функція
$Y = X$	$Y < X$	$Y > X$
0	0	1
0	1	0
1	0	0
1	1	0

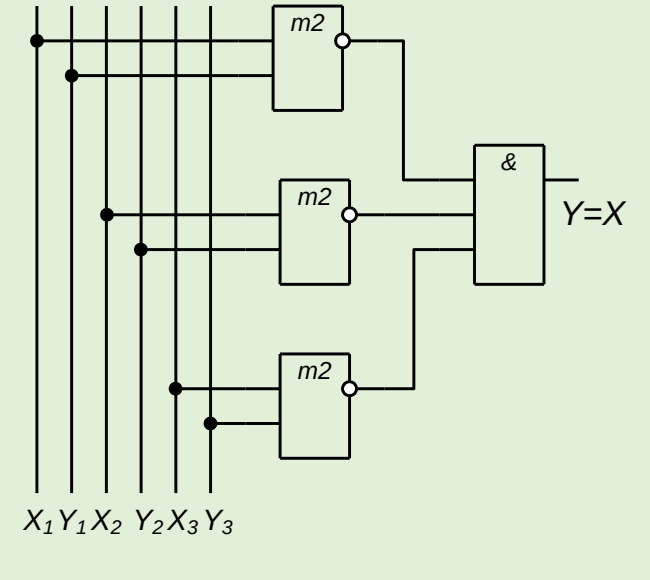


Рис. 2.5 – Схема, що реалізовує функцію Y = X

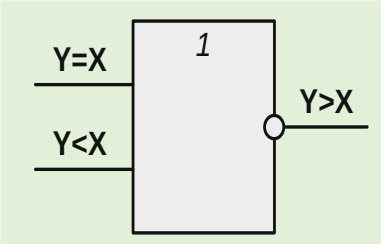


Рис. 2.6 – Схема, що реалізовує функцію Y > X

Таким чином, залишилося синтезувати схему, яка реалізовуватиме функцію $Y < X$. Вона може бути отримана на підставі наступного твердження: *для того, щоб одне кодове слово було більше, ніж друге, достатньо, щоб старший розряд першого слова був більшим, ніж старший розряд другого, або щоб будь-який розряд першого слова був більшим відповідного розряду другого слова за умови, що старші розряди обох слів попарно рівні.*

Функція порівняння відповідних окремих розрядів двох кодових слів за умови $X_n > Y_n$ може бути отримана на підставі таблиці істинності 2.5.

$$F_{x>y} = X_n \overline{Y_n}\,.$$

Таблиця 2.5

X	Y	$F_{X>Y}$
0	0	0
0	1	0
1	0	1
1	1	0

Отже, схема порівняння відповідних окремих розрядів двох кодових слів за умови $X > Y$ являє собою логічний елемент І, а результати порівняння старших розрядів кодових слів за умови $Y = X$ можуть бути отримані з виходів відповідних елементів рівності схеми, що зображена на рис. 2.5.

Трирозрядна схема порівняння відповідних розрядів двох кодових слів за умови $X > Y$ повинна реалізовувати ЛФ

$$F_{X>Y} = F_1 + F_2 + F_3,$$

де:

$$F_1 = X_2\overline{Y_2};$$
$$F_2 = X_1\overline{Y_1}(X_2 \leftrightarrow Y_2);$$
$$F_2 = X_0\overline{Y_0}(X_2 \leftrightarrow Y_2)(X_1 \leftrightarrow Y_1).$$

Якщо у схемах порівняння окремих розрядів вхідних кодових слів використовуються інверсії розрядів слова Y, доцільно для зменшення загальної кількості входів схеми перетворити схему, що виконує функцію $Y = X$ (рис. 2.5) таким чином, щоб вона також використовувала інверсії розрядів слова Y як вхідні аргументи. Це може бути зроблено таким чином:

$$x \leftrightarrow y = xy + \overline{x}\,\overline{y} = x\overline{y} + \overline{x}\,\overline{y} = x \oplus \overline{y}.$$

Тобто в схемі, що виконує функцію $Y = X$, будуть використовуватися замість елементів рівності суматори за модулем 2.

Отримана схема компаратора трирозрядних кодових слів зображена на рис. 2.7. УГП чотирирозрядного компаратора наведено на рис. 2.8.

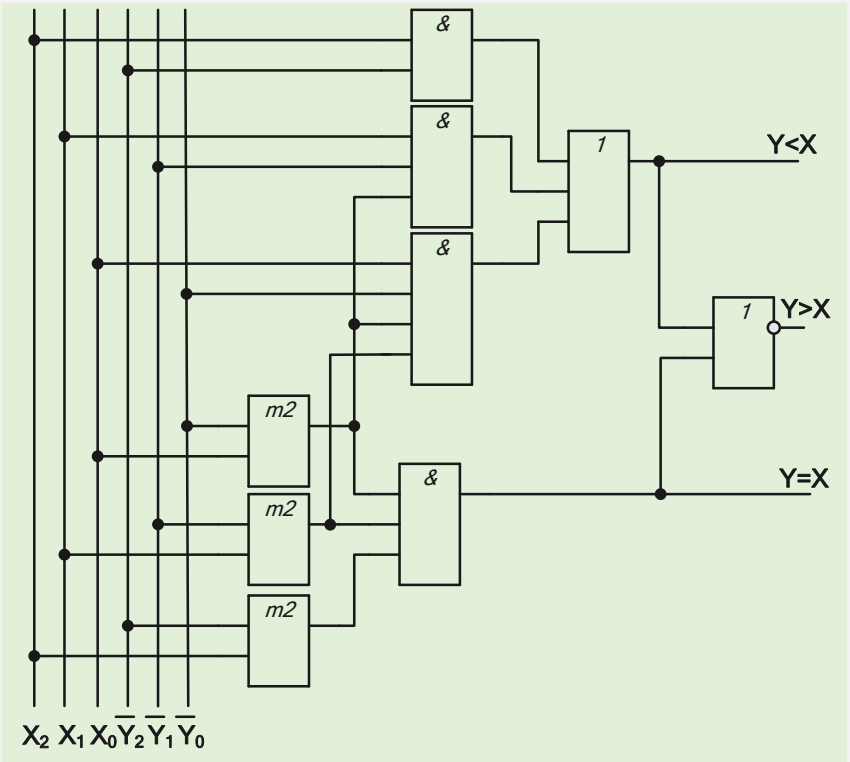


Рис. 2.7 – Схема синтезованого компаратора

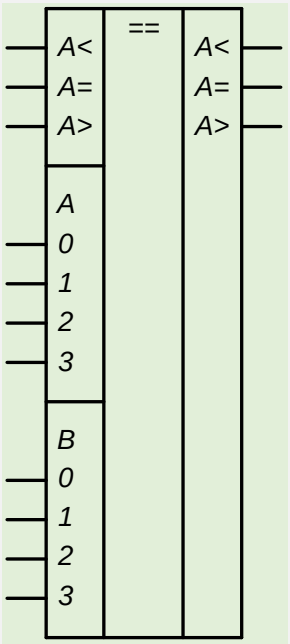


Рис. 2.8 – УГП компаратора чотирирозрядних кодових слів

Запитання і завдання

1. В чому полягає різниця між позиційними та непозиційними системами числення?
2. Поясніть алгоритм перетворення десяткового числа на двійкове.
3. Поясніть алгоритм перетворення десяткового числа на вісімкове.
4. Поясніть алгоритм перетворення десяткового числа на шістнадцяткове.
5. Поясніть алгоритм перетворення десяткового числа на двійкове.
6. Поясніть алгоритм перетворення десяткового числа на ДДК.
7. Поясніть алгоритм перетворення двійкового числа на десяткове.
8. Поясніть алгоритм перетворення двійкового числа на вісімкове.
9. Поясніть алгоритм перетворення двійкового числа на шістнадцяткове.
10. Поясніть призначення й зобразіть логічну схему перетворювача стандартного двійкового коду в код Грея.
11. Поясніть сутність та практичну реалізацію кодування з контролем на парність.
12. Що називають схемою згортання? Для чого вони використовуються?
13. Знайдіть значення контрольного біта в кодах з контролем на парність у кодових комбінаціях: а) 00101101; б) 011101110; в) 11100000.
14. Що називають компараторами слів? Які функції вони реалізовують?

Розділ 3. ЦИФРОВІ АВТОМАТИ

Послідовнісні схеми, або *ЦА з пам'яттю*, реалізовують ЛФ, значення яких у цей момент часу визначаються не тільки сукупністю значень вхідних змінних у цей же момент часу, але й попереднім станом схеми (попередніми значеннями вхідних змінних).

3.1. Пам'ять цифрових автоматів

Основою пам'яті ЦА є *тригери* – найпростіші з послідовнісних цифрових вузлів *логічні схеми, які можуть перебувати в одному з двох стійких станів і стрибком переходити в інший стан під впливом зовнішніх сигналів* (через це інколи тригер називають *бістабільним елементом*). Перехід в інший стан частіше за все залежить не тільки від поточних значень вхідних сигналів, але й від попереднього стану тригера. Інформація про попередній стан тригера, що надходить з його виходу разом із вхідними сигналами, визначає його роботу. Саме через це тригери завжди є пристроями зі зворотними зв'язками.

У цифровій техніці використовують тригери, побудовані на ЛЕ. Тригери у свою чергу є основою для побудови складних функціональних цифрових вузлів різного призначення – лічильників і розподільників імпульсів, дільників частоти слідування імпульсів, регістрів, запам'ятовуючих пристроїв.

Інтегральні тригери класифікуються за способом отримання інформації, за принципом побудови та функціональними можливостями.

За способом отримання інформації розрізняють синхронні й асинхронні тригери. *Асинхронні тригери* сприймають інформаційні сигнали та реагують на них безпосередньо в момент їх появи на інформаційних входах тригера. *Синхронні тригери* реагують на інформаційні сигнали за умови наявності дозволяючого сигналу на спеціальному керуючому вході *C*, який називають входом синхронізації. Синхронні тригери у свою чергу поділяються на тригери зі статичним та динамічним управлінням по синхровходу.

Тригери зі статичним управлінням (керовані рівнем сигналу) сприймають інформаційні сигнали за умови надходження на синхровхід

рівня логічної одиниці (прямий *C*-вхід) або нуля (інверсний *C*-вхід). *Тригери з динамічним управлінням* (керовані фронтом сигналу) сприймають інформаційні сигнали при зміні сигналу на *C*-вході з 0 на 1 (прямий динамічний *C*-вхід) або з 1 на 0 (інверсний динамічний *C*-вхід).

За принципом побудови синхронні тригери можна поділити на одно- та двоступеневі. *Одноступеневі тригери* мають лише один ступінь запам'ятовування інформації, а у двоступеневих тригерах таких ступенів два. Спочатку інформація записується у перший ступінь, потім переноситься у другий і потрапляє на вихід тригера. *Двоступеневі тригери* також називають тригерами типу *MS* (від англійського *Master – Slave*, тобто «майстер – помічник»). Ця абревіатура відображає характер роботи тригера: вхідний ступінь виробляє нове значення вихідної змінної *Q*, а вихідний ступінь його копіює.

За функціональними можливостями (або за способом організації логічних зв'язків) розрізняють:

1. *Тригер з окремим встановленням станів 0 та 1 (RS-тригер)*. *R* (від англійського *RESET* – скидання) – окремий вхід встановлення у стан 0. *S* (від англійського *SET* – встановлення) – окремий вхід встановлення тригера у стан 1.
2. *Універсальний тригер з інформаційними входами J та K (JK-тригер).* Тут *J* – вхід для встановлення універсального тригера у стан 1. *K* – вхід для встановлення універсального тригера у стан 0.
3. *Тригер, який отримує інформацію лише через один вхід D – тригер затримки, або D-тригер* (D від англійського DELAY – затримка). Тут вхід D – інформаційний вхід для встановлення тригера у стан, який збігається з логічним рівнем на цьому вході.
4. *Тригер із лічильним входом – T-тригер або лічильний тригер.* Тут вхід T – лічильний вхід.
5. *Комбіновані тригери,* у яких сполучені декілька типів тригерів. Наприклад, тригер типу RST – лічильний тригер, що також має входи встановлення та скидання.

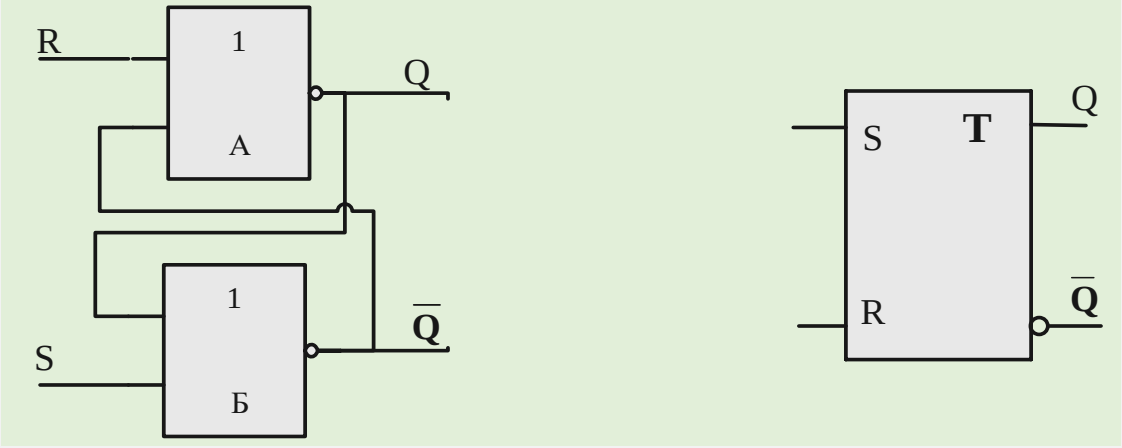
З класифікації тригерів за їх функціональними можливостями стає зрозумілим, що назва тригера за цією ознакою цілком визначається типами його входів. Тригер будь-якого типу має два виходи: прямий *Q* та інверсний $\overline{Q}$. Стан тригера визначається за прямим виходом. Функціональні можливості визначаються саме кількістю та типом входів тригера.

Для повного визначення тригера достатньо задати його структурну схему на підставі базових логічних елементів (частіше за все використовують елементи І-НІ, АБО-НІ) та закон функціонування тригера у вигляді логічної функції або таблиці переходів.

Основою тригерів усіх типів є *RS*-тригер з прямими або інверсними входами.

3.1.1. *RS*-тригери

Асинхронний RS-тригер з прямими входами має два інформаційні входи *R* та *S*, які використовуються для встановлення його відповідно у стан 0 і 1, а також два виходи – прямий *Q* та інверсний $\overline{Q}$. Цей тригер побудований на двох ЛЕ АБО-НІ, які охоплені зворотними зв'язками. Схема й УГП тригера подані на рис. 3.1 а, б відповідно.



а б

Рис. 3.1 – Схема та УГП асинхронного RS-тригера з прямими входами

У цій схемі вихід кожного ЛЕ АБО-НІ підключений до одного із входів іншого елемента. Саме таке з'єднання й забезпечує два стійких стани тригера. У *RS*-тригерах із прямими входами сигналів управління є тільки одиничні рівні сигналів. Взагалі, *вхідні сигнали, які приводять до перемикання елементів, називають активними, а ті, котрі не приводять до такого перемикання, – пасивними*. Для ЛЕ АБО-НІ активним сигналом є сигнал логічної 1.

Нехай ми маємо на входах тригера $R = 0$ та $S = 0$. Якщо початковий стан тригера $Q = 0$, то з виходу *Q* логічний 0 подається до одного із входів ЛЕ Б; при цьому на обох його входах діють логічні нулі, тому на виході ЛЕ є

сигнал логічної одиниці ($\overline{Q}=1$). З виходу ЛЕ Б логічна 1 потрапляє на вхід ЛЕ А, що забезпечує на його виході рівень логічного 0. Це один із стійких станів тригера. У стані 1 тригера $Q=1$ і відповідно $\overline{Q}=0$, при цьому на обох входах ЛЕ А діють нульові логічні рівні, що забезпечує $Q=1$.

Таким чином, у кожному з двох стійких станів тригера ЛЕ А і В знаходяться у протилежних станах. Перемикання тригера з одного стійкого стану на інший відбувається при надходженні активних сигналів на входи. Якщо R = 1, тобто, якщо тригер був у стані 0 ($Q=0$), то цей стан не зміниться. Якщо ж тригер знаходився у стані 1, то при надходженні сигналу R = 1 він перейде до стану 0. Аналогічно, якщо S = 1, то $Q=1$.

Одночасне надходження активних сигналів 1 на обидва входи (S = R = 1) є неприпустимим через те, що при цьому на обох входах установлюється нульовий стан, а після припинення дії активних сигналів стан тригера лишатиметься невизначеним: через випадкові чинники тригер може перейти до стану 1 або 0. Наведений вище алгоритм функціонування тригера може бути наочно поданий за допомогою таблиці переходів (табл. 4.1). У цій таблиці Q^t – початковий стан тригера, Q^{t+1} – наступний стан тригера, у який він перейде після надходження на його входи відповідної комбінації сигналів R та S.

Таблиця 3.1

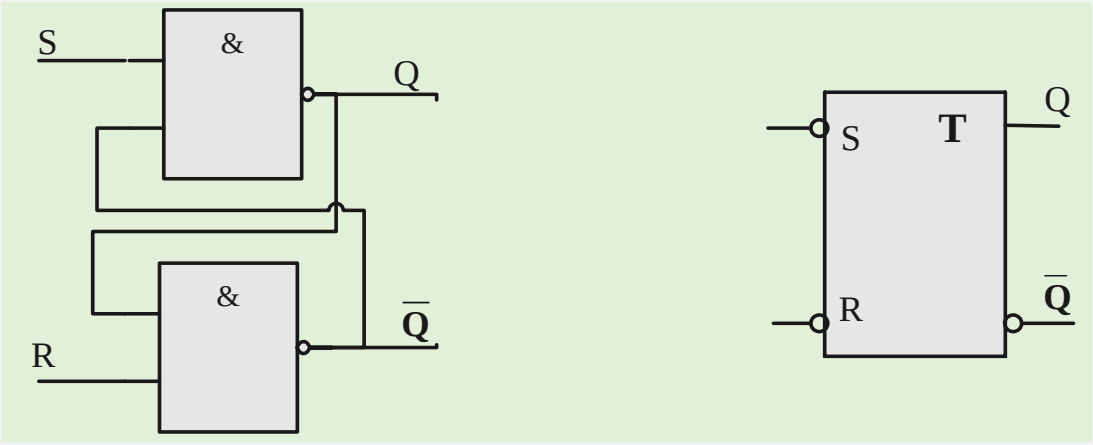
R	S	Q^t	Q^{t+1}		Режим роботи тригера
0 0	0 0	0 1	0 1	Q^t	Режим зберігання інформації
0 0	1 1	0 1	1 1	1	Встановлення 1
1 1	0 0	0 1	0 0	0	Встановлення 0 (скидання)
1 1	1 1	0 1	- -	-	Комбінація заборонена

Таким чином:

- ✓ при S = R = 0 тригер залишається у попередньому стані (режим зберігання інформації);
- ✓ при R = 1; S = 0 тригер переходить до стану 0 незалежно від попереднього стану;
- ✓ при R = 0; S = 1 тригер переходить до стану 1 незалежно від попереднього стану;

✓ комбінація вхідних сигналів $S = R = 1$ є забороненою для *RS*-тригера з прямими входами.

Асинхронний *RS*-тригер з інверсними входами побудований на ЛЕ І-НІ. При цьому активним логічним рівнем на його входах є рівень логічного 0, а пасивним – рівень логічної 1. Схема та УГП такого тригера подані відповідно на рис. 3.2, а, б.



а б

Рис. 3.2 – Схема та УГП асинхронного RS-тригера з інверсними входами

Можливі стани тригера показані у таблиці переходів (табл. 3.2).

Таблиця 3.2.

S	***R***	Q^{t+1}
0	0	-
0	1	1
1	0	0
1	1	Q^t

Таким чином:

✓ при $S = R = 1$ тригер залишається у попередньому стані;

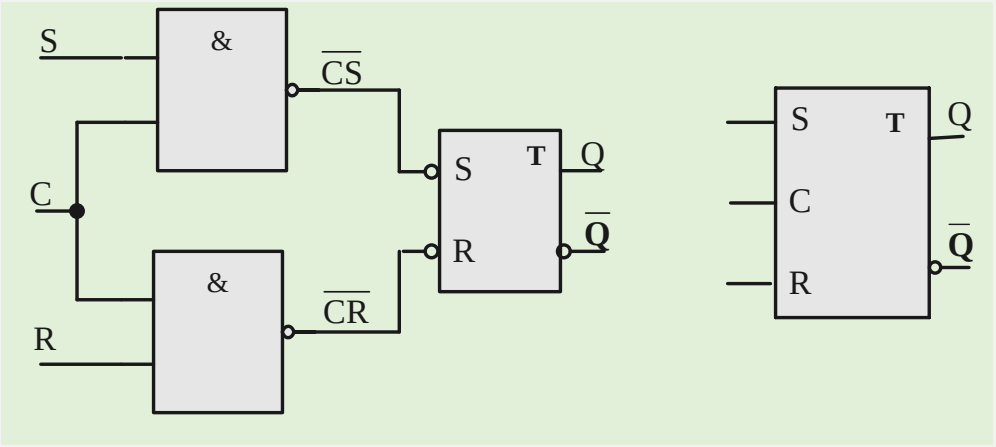
✓ при $R = 1$; $S = 0$ $Q^{t+1} = 1$;

✓ при $R = 0$; $S = 1$ $Q^{t+1} = 0$;

✓ комбінація $S = R = 0$ є забороненою.

Синхронний RS-тригер відрізняється від асинхронного наявністю *C*-входу, на який надходять синхронізуючі (тактові) сигнали. Синхронний тригер складається з асинхронного *RS*-тригера та комбінаційного

цифрового пристрою, як показано на рис. 3.3, а. УГП такого тригера подано на рис. 3.3, б.



а *б*

Рис. 3.3 – Схема та УГП синхронного RS-тригера

За допомогою ЛЕ І-НІ, які утворюють вхідний комбінаційний пристрій, забезпечується передавання активних рівнів сигналів на інформаційних входах *S* та *R* синхронного тригера на інверсні входи внутрішнього асинхронного тригера лише за умови наявності логічної 1 на синхровході *C*. При $C = 1$ стан тригера визначається сигналами на його входах аналогічно до розглянутого вище асинхронного тригера. При $C = 0$, тригер не реагує на рівні сигналів на входах *S* та *R*. Алгоритм функціонування тригера поданий у таблиці переходів (табл. 3.3).

Рядки, де $C = 0$, у таблиці відсутні через те, що вони не несуть корисної інформації.

Очевидно, що таблиця переходів такого синхронного тригера практично збігається з таблицею переходів асинхронного *RS*-тригера із прямими входами (якщо не брати до уваги стовпець *C* табл. 3.3), хоча у схемі застосований асинхронний *RS*-тригер з інверсними входами.

Таблиця 3.3

S	***R***	***C***	Q^{t+1}
0	0	1	Q^t
0	1	1	0
1	0	1	1
1	1	1	-

Справа у тому, що всередині схеми, наведеної на рис. 3.3, двічі відбувається інверсія тих самих сигналів – на виходах ЛЕ І-НІ

комбінаційного пристрою та на інверсних входах асинхронного тригера, що відповідно до закону подвійної інверсії ($\overline{\overline{X}} = X$) означає відсутність інверсії взагалі. Таким чином, у схемі можна виключити всі позначки внутрішніх операцій інверсії, при цьому для отриманої схеми буде справедлива та ж сама таблиця переходів.

Як видно, наявність входу синхронізації *C* розширює можливості тригерів. Тому практично всі реальні інтегральні тригери мають такий вхід. При цьому залишається можливість функціонування в асинхронному режимі: для цього лише потрібно подати на вхід *C* сталий активний рівень сигналу.

Головним недоліком *RS*-тригерів є наявність заборонених комбінацій сигналів. Статичні *RS*-тригери застосовуються для побудови регістрів пам’яті.

3.1.2. *JK*-тригери

Універсальний *JK*-тригер функціонує майже так само, як і звичайний *RS*-тригер. При цьому вхід *J* виконує роль входу *S*, а вхід *K* – входу *R*. Таким чином, активний сигнал (рівень логічної 1), поданий на вхід *J*, переводить тригер у стан 1, а поданий на вхід *K* – у стан 0. Різниця полягає лише у тому, що при *J* = *K* = 1 тригер змінює свій стан на протилежний. *JK*-тригер не має заборонених комбінацій вхідних сигналів, наявність яких була головною вадою *RS*-тригерів.

Найпростіший *JK*-тригер можна побудувати з того ж набору елементів, що й синхронний *RS*-тригер. Схема такого тригера подана на рис. 3.4. Алгоритм його функціонування поданий у таблиці переходів (табл. 3.4).

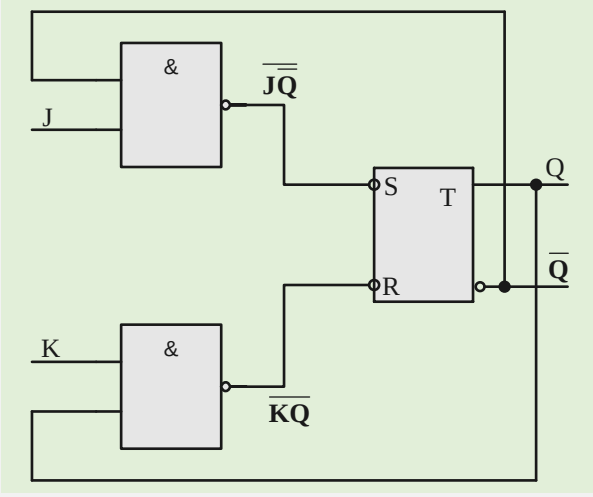


Рис. 3.4 – Схема найпростішого JK-тригера

Таблиця 3.4

J	***K***	Q^{t+1}
0	0	Q^t
0	1	0
1	0	1
1	1	$\overline{Q}^t$

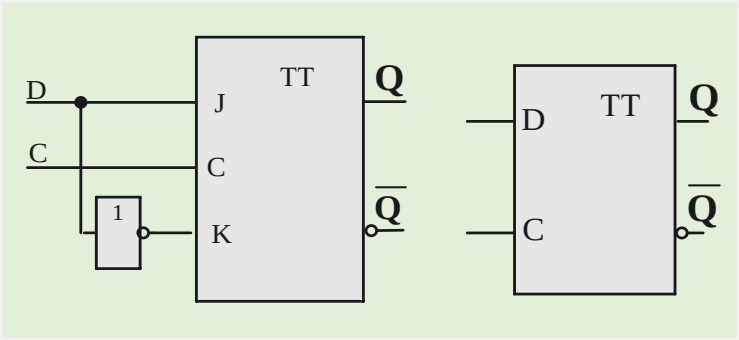
Припустимо, що тригер знаходиться у стані 0 ($Q = 0$, $\overline{Q} = 1$), а на його входах діють сигнали $J = K = 0$. При цьому на виходах обох ЛЕ І-НІ – рівні логічної 1. Відповідно до рядка 4 з табл. 3.4 така комбінація вхідних сигналів ($R = S = 1$) в асинхронному *RS*-тригері з інверсними входами забезпечує режим зберігання інформації (стан тригера залишається незмінним). Якщо ж подати на входи тригера сигнали $J = K = 1$, то вихідний сигнал другого ЛЕ І-НІ не зміниться, а на виході першого ЛЕ з'явиться сигнал логічного 0. При такій комбінації вхідних сигналів ($R = 1$, $S = 0$) асинхронний *RS*-тригер з інверсними входами переходить до стану 1 (рядок 2 табл. 3.4). Аналогічні міркування можна навести стосовно одиничного початкового стану тригера. Таким чином, *JK*-тригер при надходженні на його входи сигналів $J = K = 1$ дійсно переходить до протилежного стану.

Треба зазначити, що на практиці застосовуються схеми *JK*-тригерів, значно складніші від наведеної на рис. 3.4.

3.1.3. *D*- і *T*-тригери

D-тригер має один інформаційний вхід (*D*-вхід) і вхід синхронізації *C*. Головне призначення *D*-тригера – затримка сигналу на один такт синхронізації. *D*-тригер може бути отриманий з *JK*-тригера об'єднанням входу *K* із входом *J* через інвертор так, як це зображено на рис. 3.5, а.

Функціонування *D*-тригера відображено таблицею переходів, у якій виключені рядки для $C = 0$ (табл. 3.5). При $C = 0$ тригер може необмежений час зберігати раніше встановлений стан. При $C = 1$ інформація, що надходить на вхід *D*, потрапляє до тригера, але на виході його вона з'явиться із затримкою на один такт. УГП *D*-тригера подано на рис. 3.5, б.



а *б*

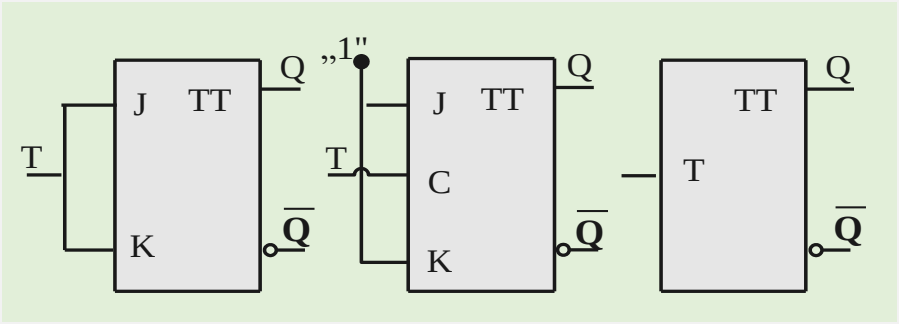
Рис. 3.5 – Схема та УГП D-тригера

Таблиця 3.5

D	C	Q^{t+1}
1	1	1
0	1	0

Як було розглянуто раніше, при надходженні на обидва інформаційні входи *JK*-тригера рівня логічної 1 ($J = K = 1$) він переходить до протилежного стану. Із цієї точки зору найбільш доцільним способом перетворення *JK*-тригера на асинхронний *T*-тригер є просте об'єднання його *J*-, *K*-входів для отримання лічильного *T*-входу, як це зображено на рис. 3.6, а.

На практиці частіше за все використовують синхронну схему *T*-тригера, у якій як *T*-вхід використовують вхід *C* синхронного *JK*-тригера, а на *J*-, *K*-входи подають рівень логічної 1, як це зображено на рис. 3.6, б. Алгоритм функціонування *T*-тригера однаковий для схем, зображених на рис. 3.6 а, б (якщо не враховувати сигнал синхронізації), поданий у вигляді таблиці переходів (табл. 3.6). УГП тригера зображено на рис. 3.6, в.



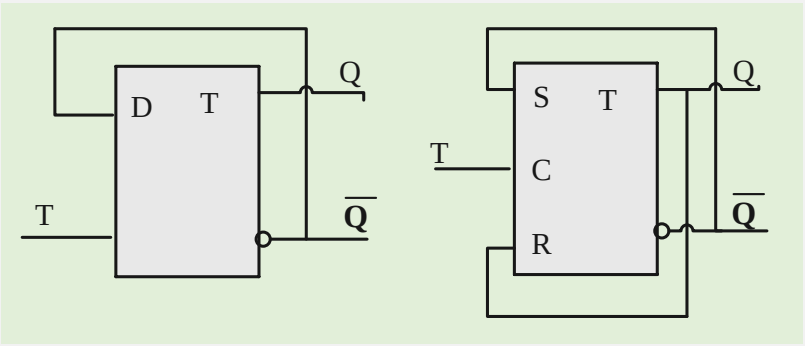
а б в

Таблиця 3.6

T	Q^{t+1}
1	1
0	0

Рис. 3.6 – Варіанти побудови та УГП Т-тригера

T-тригер також може бути побудований на основі тригерів інших типів. На рис. 3.7 наведені варіанти схем асинхронного *T*-тригера на *D*-тригері та на синхронному *RS*-тригері.



а б

Рис. 3.7 – Варіанти побудови Т-тригера

Лабораторна робота «Аналіз функціонування тригерів»

3.2. Типові цифрові автомати

До типових послідовнісних цифрових вузлів відносять: тригери, регістри, лічильники, дільники частоти слідування імпульсів, схеми порівняння поточного і попередніх станів; перетворювачі паралельної двійкової інформації у послідовну та навпаки, накопичувальні суматори й ін.

3.2.1 Регістри

Регістром називається послідовнісний ЦА, що реалізовує функції прийому, зберігання і подальшої видачі інформації, поданої у вигляді m-розрядної кодової комбінації (кодового слова).

Регістри є найбільш розповсюдженими вузлами цифрових пристроїв. Основою побудови регістрів, як і інших послідовнісних функціональних вузлів, є тригери.

При введенні додаткових ЛЕ у схеми регістрів за їх допомогою можна здійснювати не тільки збереження, але й деяке перетворення інформації, наприклад перетворення прямого коду числа на обернений код і, навпаки, перетворення кодового слова, що надходить у послідовній формі, на паралельну, і навпаки, зсув двійкового числа на декілька розрядів праворуч або ліворуч, одержання певних числових послідовностей шляхом замикання регістра у кільце та ін.

Відповідно до виконуваних функцій регістри поділяються на *регістри без зсуву інформації* (регістри пам’яті, накопичувальні або статичні регістри) і *регістри зі зсувом інформації* (регістри зсуву). Різновидами регістрів зсуву є *кільцеві й рекурентні регістри*.

За способом подання (формою) вхідних і вихідних кодових слів розрізняють такі типи регістрів:

- ✓ із паралельним прийомом та видачою інформації (паралельні регістри);
- ✓ із послідовним прийомом і видачою інформації (послідовні регістри);
- ✓ із послідовним прийомом та паралельною видачою інформації чи навпаки (паралельно-послідовні регістри).

Регістри з послідовним прийомом або видачою інформації є регістрами зсуву. За напрямком зсуву розрізняють *регістри зі зсувом праворуч, ліворуч або реверсивні* (з можливістю зсуву в обох напрямках). Залежно від

кількості вхідних і вихідних каналів регістри поділяються на *однофазні* (сигнали передаються по одному каналу) та *парафазні* (сигнали передаються по двох каналах – прямому й інверсному). Парафазні регістри реалізовуються на *RS*- або *JK*-тригерах, однофазні – на *D*-тригерах. Регістри характеризуються числом розрядів оброблюваного кодового слова і швидкодією, яка обумовлена максимальною тактовою частотою приймання, передавання та зсуву інформації.

Регістри пам’яті

Регістр, що виконує запис числа, збереження і видачу його прямим кодом або оберненим кодом усіма розрядами одночасно називається регістром паралельної дії (регістром пам’яті). Схема такого регістра зображена на рис. 3.8.

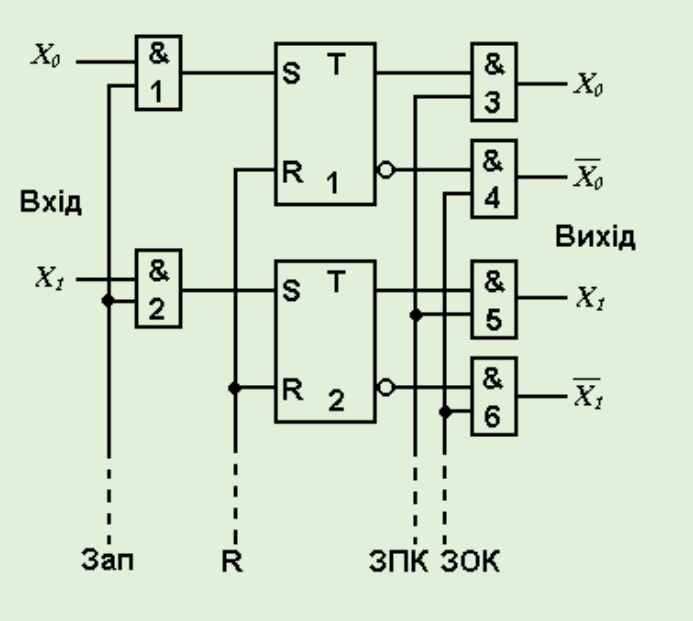


Рис. 3.8 – Схема паралельного регістра

Вихідний стан (усі тригери встановлені у «0») забезпечується подачею імпульсу на вхід *R* (скидання). Вхідне кодове слово подається всіма розрядами одночасно на входи $X_0 \ldots X_n$. Якщо в момент подачі вхідного кодового слова на вході «Зап» (запис коду) є рівень логічної 1, то розряди слова надходять на входи *S* відповідних *RS*-тригерів. Припустимо, що $X_0 = 1$, $X_1 = 0$. Тоді перший *RS*-тригер буде встановлений в одиничний стан ($Q = 1$), а другий залишиться в нульовому стані. Таким чином, вхідне кодове слово буде записано у регістр і може зберігатися там як завгодно довго (до подачі

імпульсу скидання), якщо на вході «Зап» діє рівень логічного нуля (якщо на цьому вході діє логічний нуль, то на виходах елементів І № 1 і № 2 – теж логічні нулі, і зміна X_0, X_1 не впливає на стан тригерів).

Інформація, що записана в регістрі, може бути отримана в будь-який момент часу в ПК або в ОК. Якщо на входах «ЗПК» (зчитування прямим кодом) і «ЗОК» (зчитування оберненим кодом) – рівні логічного нуля, то на виходах $X_0, \overline{X_0}, X_1, \overline{X_1}$ логічні нулі незалежно від стану тригерів, тобто інформація, що зберігається, на виходи регістра не потрапляє.

Якщо на вхід «ЗПК» поданий рівень логічної 1, то на виходи X_0, X_1 регістра надходить збережене кодове слово у ПК ($X_0 = 1, X_1 = 0$). Якщо на вхід «ЗОК» поданий рівень логічної 1, то на виходи $\overline{X_0}, \overline{X_1}$ регістра надходить збережене кодове слово в ОК ($\overline{X_0} = 0, \overline{X_1} = 1$).

Таким чином, розглянутий регістр дійсно виконує функції запису, збереження і видачі кодових слів у ПК або ОК усіма розрядами одночасно.

УГП чотирирозрядного паралельного регістра показано на рис. 3.9.

Із статичних регістрів шляхом їх паралельного з'єднання складаються блоки регістрової пам'яті – регістрові файли. Регістрові файли широко застосовуються, наприклад, як статичні оперативні запам'ятовуючи пристрої.

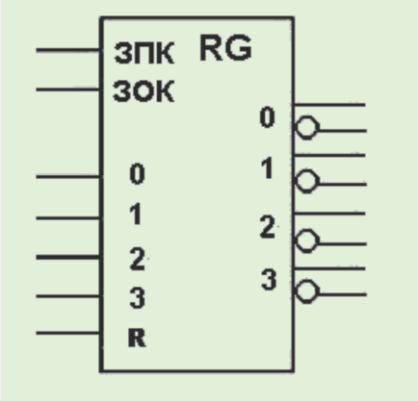


Рис. 3.9 – УГП паралельного регістра

Регістри зсуву

Регістр зсуву – це група тригерів, з'єднаних таким чином, що інформація з кожного з них може передаватися до наступного тригера, зсуваючи код, записаний у ньому. На рис. 3.10 показана схема регістра зсуву, що складається з послідовно з'єднаних *D*-тригерів із динамічними входами синхронізації.

Розряди кодового слова подаються на вхід X послідовно. По передньому фронту синхронізуючого імпульсу C інформація із входу X записується в перший тригер ($Q_1^{t+1} = X$). Ця інформація (0 або 1) зберігається у тригері $T1$ один такт до надходження наступного синхроімпульсу.

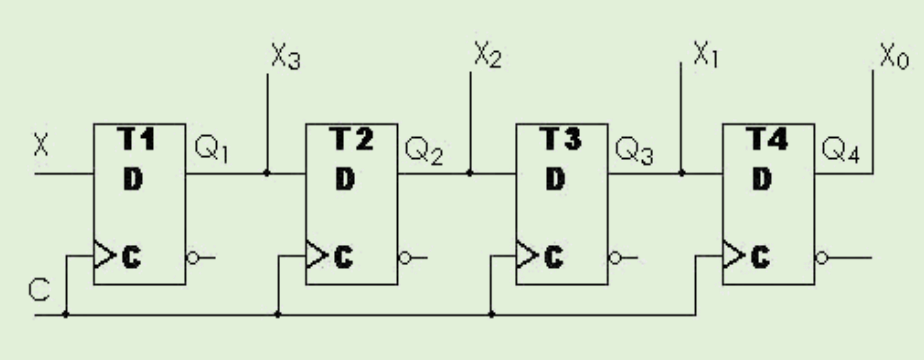


Рис. 3.10 – Схема регістра зсуву

У наступному такті ця інформація надійде до тригера $T2$, а до тригера $T1$ – наступний розряд кодового слова, тобто в такому регістрі відбувається зсув даних від тригера до тригера

$$Q_i^{t+1} = Q_{i-1}^t (i = 2,3,...,m).$$

Нехай на вхід регістра надходить кодове слово 1011 молодшим розрядом уперед. Тоді функціонування регістра описується таблицею переходів (табл. 3.7).

Таблиця 3.7

Такти	**Стани тригерів регістра**			
	Q_1	Q_2	Q_3	Q_4
Початковий стан	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	0	1	1	0
4	1	0	1	1

З таблиці видно, що після чотирьох тактів роботи регістра чотирирозрядне кодове слово буде повністю записано у регістр.

У розглянутому випадку зсув інформації здійснювався в напрямку молодших розрядів (праворуч). Також використовуються і регістри зі

зсувом ліворуч та реверсивні, що змінюють напрямок зсуву інформації залежно від значення спеціального сигналу керування.

Напрямок зсуву кодового слова, якщо це необхідно, може бути вказаний стрілкою в УГП регістра, як показано у табл. 3.8. Зміна напрямку зсуву інформації у регістрі досягається шляхом відповідної комутації з'єднань тригерів регістра через додаткові логічні кола за допомогою спеціальних сигналів керування.

Таблиця 3.8

Вид регістра	Регістр зі зсувом праворуч	Регістр зі зсувом ліворуч	Реверсивний регістр
Елемент УГП	RG →	RG ←	RG ↔

Регістри зсуву застосовуються для:

- ✓ перетворення паралельних кодових слів у послідовні й навпаки;
- ✓ у схемах множення і ділення (зсув двійкового числа ліворуч або праворуч відповідає його множенню чи діленню на два);
- ✓ для затримки інформації, що передається, на *n* тактів;
- ✓ для побудови регістрів спеціального призначення – кільцевих та рекурентних.

3.2.2. Лічильники

Лічильником називають послідовнісний ЦА, що забезпечує збереження кодового слова і виконання над ним операції рахування. Операція рахування полягає у зміні значення числа *C* у лічильнику на задану константу (частіше за все на одиницю).

Основним параметром лічильника є *модуль рахування M* (інша назва – ємність), тобто максимальне число імпульсів, що може бути перелічене лічильником. Лічильник, який має *n* двійкових розрядів, може знаходитися у $0,1,2,\ldots,2^n-1$ станах. При надходженні на вхід додаючого лічильника 2^n-го імпульсу він переходить із стану 2^n-1 у стан 0. Таким чином, *n*-розрядний додаючий двійковий лічильник має модуль рахування $M = 2^n$.

Лічильники характеризуються також швидкодією, що визначається припустимою частотою вхідних сигналів (імпульсів) і часом встановлення стану лічильника. Реалізовуються лічильники частіше за все на *T*- або *JK*-тригерах.

Лічильники класифікують за різними ознаками:

1. За напрямком рахування: лічильник, що реалізовує мікрооперацію $C: = C + 1$ (інкрементацію) називають *додаючим*, а той, що виконує мікрооперацію $C: = C - 1$ (декрементацію), – *віднімаючим.* Лічильник називається *реверсивним*, якщо може реалізувати рахування в обох напрямках.

2. *За способом організації схеми переносу* (за способом з'єднання тригерів між собою, тобто способом передавання сигналу з молодшого розряду до старшого), лічильники можуть бути *з послідовним, паралельним, наскрізним і з комбінованим переносами*.

3. Залежно від наявності синхронізації лічильники можуть бути *синхронні та асинхронні*.

4. За способом кодування внутрішніх станів лічильника розрізняють *двійкові лічильники, лічильники Джонсона, лічильники з кодом «1 із N»* й ін.

Двійкові лічильники з послідовним переносом

Схема й УГП трирозрядного додаючого двійкового лічильника з послідовним переносом зображені на рис. 3.11. Такий лічильник може реалізовувати послідовність рахування від 0 до $2^3 - 1 = 7$. Кожний стан відповідає трирозрядному двійковому числу від 000 до 111. Початковий стан 000 встановлюється подачею імпульсу на вхід *R* усіх *T*-тригерів одночасно.

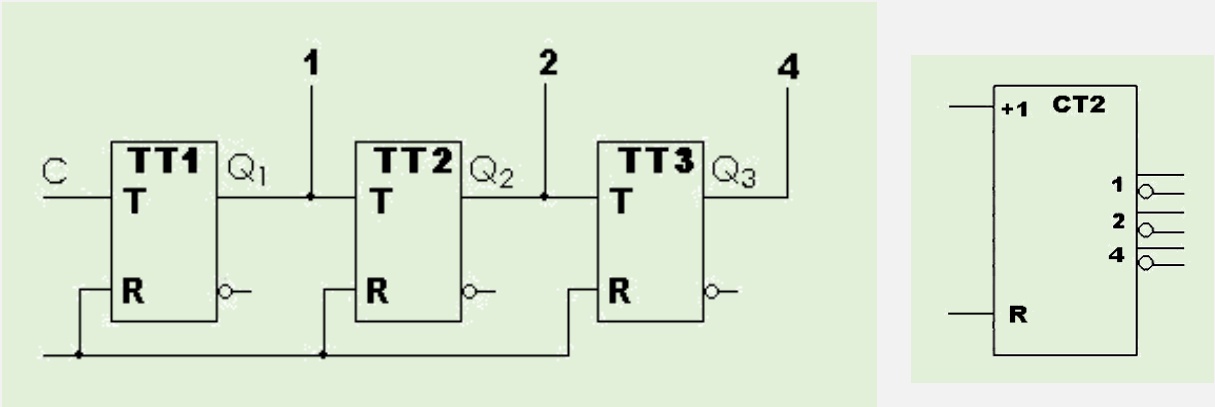


Рис. 3.11 – Трирозрядний двійковий додаючий лічильник

Тригери, з яких складається лічильник, побудовані таким чином, що вони переходять у протилежний стан за умови наявності на вході зміни рівня вхідної напруги з 1 на 0, тобто при проходженні заднього фронту вхідного імпульсу. Принцип роботи лічильника пояснюється часовою

діаграмою сигналів на його виходах, поєднаною з таблицею переходів, як показано на рис. 3.12.

З надходженням першого імпульсу на вхід *C* тригер *TT*1 переходить у стан 1 (Q_1 = 1). На входах тригерів *TT*2 і *TT*3 не відбувається зміна рівнів вхідної напруги з 1 на 0, ці тригери зберігають свій стан незмінним. У лічильнику записане число 001. З надходженням другого імпульсу тригер 1 переходить у стан 0, тригер 2 – у стан 1. Тригер 3 зберігає свій стан незмінним. У лічильнику записане тепер число 010. Тригер 3 перейде до стану 1 лише при надходженні на лічильний вхід четвертого імпульсу.

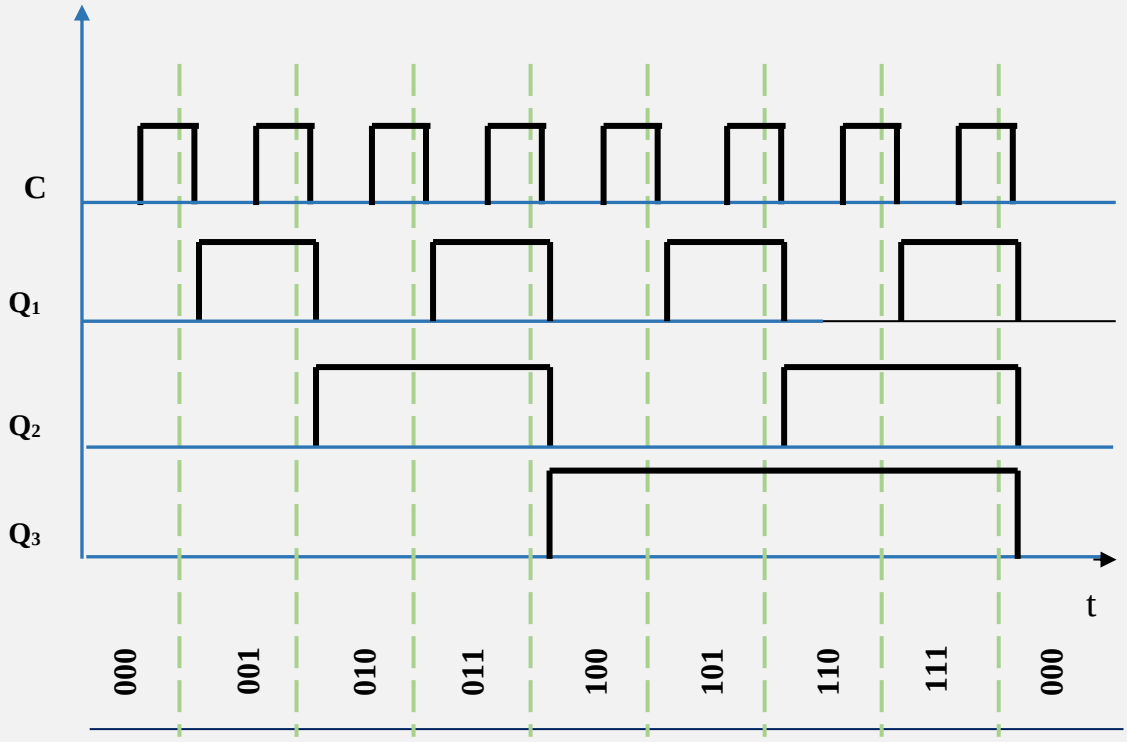


Рис. 3.12 – Часова діаграма роботи трирозрядного додаючого лічильника

До моменту приходу восьмого імпульсу на виходах тригерів Q_1, Q_2, Q_3 буде встановлений рівень 1. Після закінчення його дії всі тригери лічильника перейдуть у стан 0. Лічильник тепер готовий рахувати нову імпульсну послідовність з восьми імпульсів.

З рис. 3.12 видно, що частота проходження імпульсів на виході першого тригера вдвічі менша, ніж на вході *C*, на виході другого тригера – у 4 рази, на виході третього – у 8 разів, тобто кожний тригер лічильника зменшує частоту проходження імпульсів удвічі. Ця властивість лічильників і обумовила можливість їхнього застосування як дільників частоти проходження імпульсів на число 2^n, тобто, використовуючи лічильник,

можна одержати з однієї імпульсної послідовності декілька синхронізованих послідовностей кратних частот, необхідних, наприклад, для погодженого керування (тактування) вузлами окремих вузлів складного цифрового пристрою (наприклад, системної плати комп'ютера) у цілому.

Схема трирозрядного віднімаючого двійкового лічильника, яка подана на рис. 3.13, відрізняється від розглянутої схеми додаючого лічильника тим, що:

- ✓ сигнал на вхід кожного наступного тригера подається не з прямого, а з інверсного виходу попереднього тригера;
- ✓ замість входу *RESET* («*Встановлення* 0») наявний вхід *SET* («*Встановлення* 1») для забезпечення початкового стану 111. У такому лічильнику з надходженням кожного імпульсу на вхід *C* відбувається зменшення записаного числа на 1. Після надходження восьмого імпульсу в лічильнику встановлюється початковий стан 111.

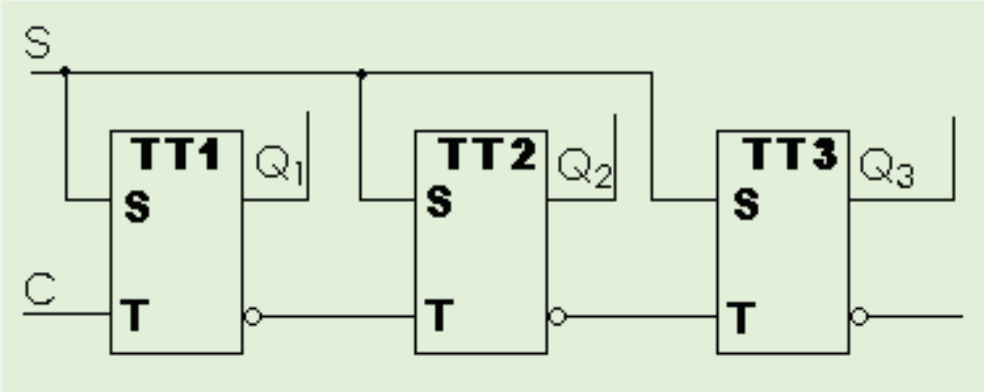


Рис. 3.13 – Двійковий трирозрядний віднімаючий лічильник

Реверсивний лічильник

Для отримання схеми реверсивного лічильника у міжрозрядні кола переносу додаючого або віднімаючого лічильника необхідно додати ЛЕ, що дозволяють підключати вхід кожного наступного тригера до прямого або інверсного виходу попереднього тригера. На рис. 3.14 зображена схема реверсивного лічильника з керуючим *RS*-тригером.

При встановленні керуючого *RS*-тригера у стан 1 (подачею імпульсу на вхід «+») відкриваються ЛЕ І, через які до входів наступних тригерів підключаються прямі виходи попередніх. При цьому лічильник буде працювати як додаючий. Встановлення вихідного стану 000 в цьому випадку здійснюється подачею відповідного імпульсу на вхід *S*.

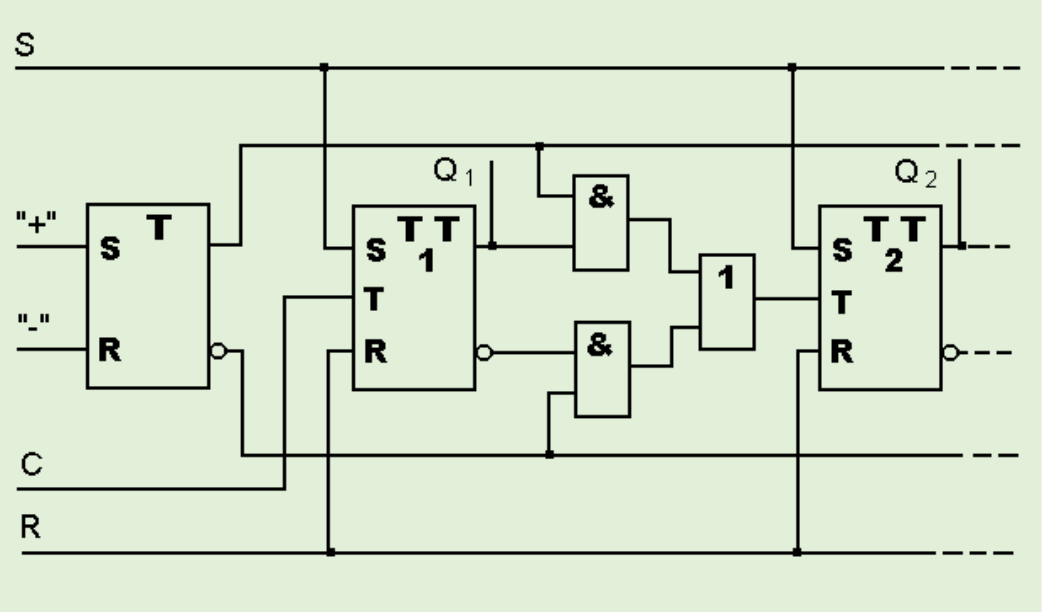


Рис. 3.14 – Схема двійкового реверсивного лічильника

При встановленні *RS*-тригера у стан 0 (подачею імпульсу на вхід «-») відкриваються ЛЕ І, через які до входів наступних тригерів підключаються інверсні виходи попередніх. Лічильник буде працювати як віднімаючий. Відповідно встановлення вихідного стану 111 здійснюється подачею імпульсу встановлення на вхід *R* схеми. ЛЕ АБО в схемі реверсивного лічильника необхідні для розв'язки виходів комутуючих елементів І.

Крім розглянутих найпростіших двійкових лічильників, існують лічильники з модулем рахування $K_c \neq 2^n$. З них частіше за все застосовуються лічильники, що працюють у десятковому коді або у ДДК (так звані декади). Також широко застосовуються лічильники з програмованим модулем рахування – наприклад, у складних схемах синхронізації. На основі лічильників будуються *дільники частоти слідування імпульсів* – пристрої, які при надходженні на їх вхід періодичної імпульсної послідовності формують на виході таку ж послідовність, але із частотою повторення імпульсів, яка у визначене число разів $K_д$ менша, ніж частота повторення імпульсів вхідної послідовності.

Більш докладно будова, функціонування та застосування типових цифрових функціональних вузлів розглядається в дисципліні «Комп'ютерна схемотехніка».

3.3. Основи синтезу цифрових автоматів

Проєктування функціональних вузлів будь-якого типу – як комбінаційних, так і послідовнісних – складається з послідовного розв'язання задач синтезу й аналізу ЦА. Однак взагалі синтез автоматів з пам'яттю значно складніший, ніж синтез комбінаційних схем, що обумовлено наявністю елементів пам'яті та, відповідно, зворотних зв'язків.

Головною метою синтезу ЦА з пам'яттю є визначення всіх його можливих станів та переходів відповідно до заданого алгоритму функціонування та отримання функцій збудження всіх входів тригерів, з яких складається автомат. Цього достатньо для складання логічної схеми ЦА з урахуванням заданого схемотехнічного базису.

Багатоваріантність можливих реалізацій ЦА пов'язана з вибором типу тригерів та способу побудови його комбінаційної частини. Теоретично будь-який ЦА може бути побудований на тригерах будь-якого типу. Найбільш розповсюджені в схемотехніці *D*- та *JK*-тригери. *JK*-тригер має більш розвинені логічні можливості, тому для нього можна отримати більш прості функції збудження, але кількість функцій буде вдвічі більшою, ніж для *D*-тригера. Яке рішення буде оптимальним для конкретного ЦА, заздалегідь невідомо.

Алгоритм синтезу ЦА з пам'яттю містить такі основні етапи:

1. *Запис та формалізація умов функціонування автомата*. Як і для комбінаційних пристроїв, вихідне завдання функціонування ЦА з пам'яттю може виконуватися в різних формах, у тому числі й словесній. У результаті формалізації необхідно отримати таблиці або формули, що повно та однозначно описують алгоритм функціонування ЦА в усіх можливих режимах роботи.

2. *Мінімізація та кодування станів.* На цьому етапі необхідно визначити мінімальну кількість усіх можливих станів ЦА з урахуванням усіх необхідних напрямків переходу. Кодування станів найчастіше виконується з використанням двійкових кодів. Якщо ЦА може функціонувати в декількох різних режимах, доцільно скласти граф переходів (діаграму станів), що наочно відображає як можливі стани ЦА, так і напрямки переходів у різних режимах функціонування.

3. *Складання таблиці переходів.* На основі діаграми станів (для ЦА, режим роботи яких завжди однаковий, – безпосередньо на основі вихідної

таблиці) складається таблиця переходів, у котрій необхідно показати не тільки попередні та наступні стани тригерів, але й сигнали на їх входах.

4. *Визначення функцій збудження тригерів*. Функції збудження тригерів отримують із таблиці переходів, зазвичай у вигляді ДДНФ, таким же чином, що і для комбінаційних схем. Попередні стани тригерів при цьому використовуються як вхідні аргументи функцій.

5. *Мінімізація функцій збудження тригерів*. Функції збудження тригерів реалізовуються комбінаційною частиною ЦА, що в загальному випадку має $n + m$ входів (n – кількість вхідних сигналів ЦА, m – кількість виходів усіх тригерів ЦА) та $l + k$ виходів (l – кількість вихідних сигналів ЦА, k – кількість входів усіх тригерів ЦА). Тобто на цьому етапі виконується спільна мінімізація $l + k$ логічних функцій $n + m$ аргументів. Для мінімізації функцій можуть використовуються будь-які методи, на власний розсуд проєктувальника.

6. *Перехід до заданого базису та складання логічної схеми ЦА* виконуються практично таким же чином, що і при синтезі комбінаційних схем. Однак слід мати на увазі, що поняття логічного базису застосовується лише до комбінаційної частини ЦА, але не стосується пам’яті автомата (тип тригерів, що реалізовують пам’ять ЦА, визначається вихідними умовами задачі або вибирається проєктувальником).

Завершується проєктування ЦА його аналізом, тобто моделюванням чи макетуванням отриманої схеми з метою перевірки правильності її функціонування.

Приклад синтезу цифрового автомата з пам’яттю

1. *Поставлення завдання (вихідне завдання функціонування)*: необхідно синтезувати трирозрядний двійковий лічильник прямого рахування на основі T-тригерів, алгоритм функціонування якого визначає керуючий сигнал M. Якщо $M = 0$, лічильник працює як звичайний лічильник прямого рахування; якщо $M = 1$, лічильник працює в коді Грея. Зміна керуючого сигналу M відразу веде до зміни режиму роботи, тобто наступний стан лічильника буде належати вже іншому коду.

Синтез проведемо без обмежень на використовуваний логічний базис з метою отримання схеми з мінімальною кількістю елементів.

На основі опису ЦА можливо відразу отримати його структурну схему, що зображена на рис. 3.15.

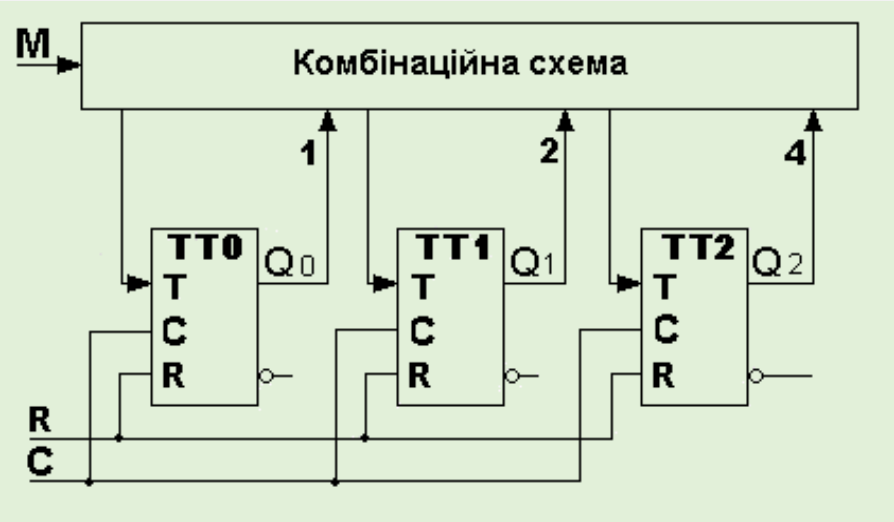


Рис. 3.15 – Структурна схема ЦА

Комбінаційна схема реалізовує необхідні функції збудження *T*-тригерів на основі їх станів та керуючого сигналу *M*. Усі тригери мають загальні входи скидання та синхронізації.

2. *Формалізоване завдання функціонування. Мінімізація та кодування станів автомата.* Алгоритм функціонування лічильника в обох заданих режимах може бути поданий табл. 3.9.

На підставі табл. 3.9 складемо граф переходів (діаграму станів лічильника, що показана на рис. 3.16).

Напрямки переходів на діаграмі вказані: для рахування в коді 8421 – синіми лініями, для рахування в коді Грея – зеленими. Ділянки діаграми, де напрямки переходів у двох режимах збігаються, вказані подвійними лініями.

Таблиця 3.9

Такт	M = 0; код 8421			M = 1; код Грея		
	X_2	X_1	X_0	X_2	X_1	X_0
Початковий стан	0	0	0	0	0	0
1	0	0	1	0	0	1
2	0	1	0	0	1	1
3	0	1	1	0	1	0
4	1	0	0	1	1	0
5	1	0	1	1	1	1
6	1	1	0	1	0	1
7	1	1	1	1	0	0

3. *Складання таблиці переходів автомата*. На підставі діаграми станів автомата з урахуванням алгоритму функціонування *T*-тригера складаємо таблицю переходів ЦА (табл. 3.10).

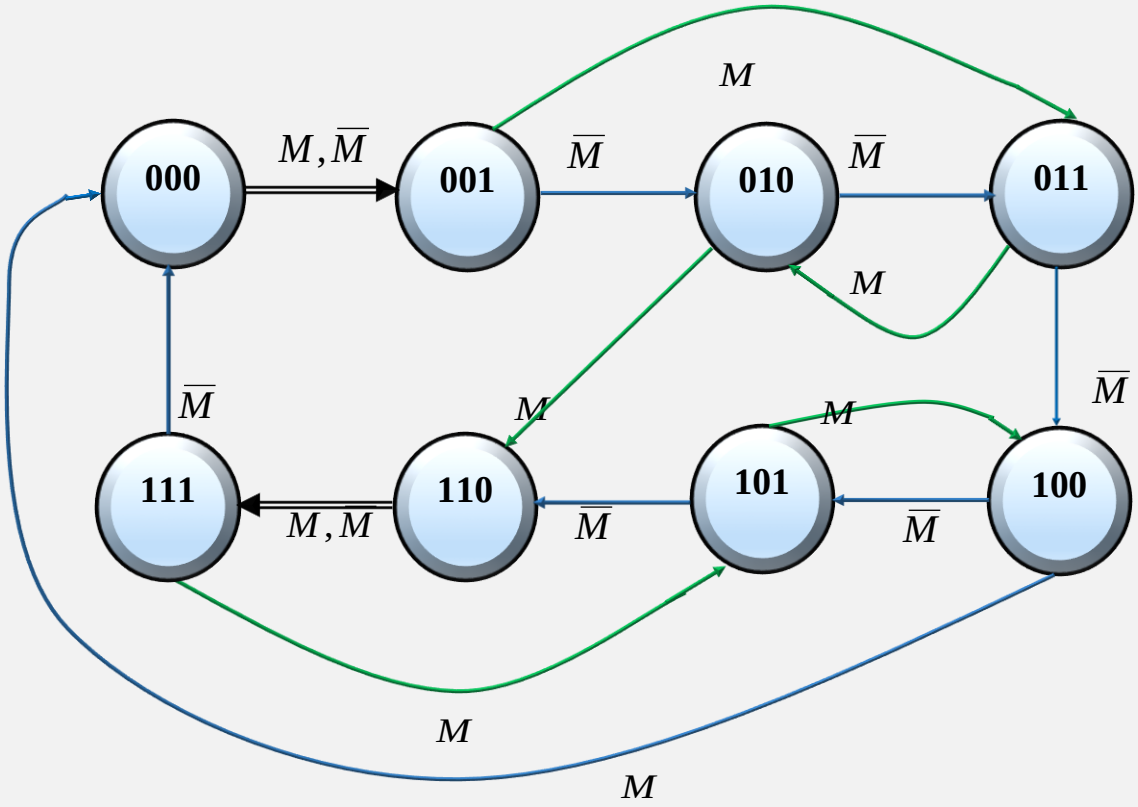


Рис. 3.16 – Діаграма станів лічильника

4. *Отримання функції збудження тригерів ЦА* (у вигляді ДДНФ):

$$T_0 = \overline{M}\,\overline{Q}_2\overline{Q}_1\overline{Q}_0 + \overline{M}\,\overline{Q}_2\overline{Q}_1Q_0 + \overline{M}\,\overline{Q}_2Q_1\overline{Q}_0 + \overline{M}\,\overline{Q}_2Q_1Q_0 + \overline{M}Q_2\overline{Q}_1\overline{Q}_0 + \overline{M}Q_2\overline{Q}_1Q_0 +$$
$$+\overline{M}Q_2Q_1\overline{Q}_0 + \overline{M}Q_2Q_1Q_0 + M\overline{Q}_2\overline{Q}_1\overline{Q}_0 + M\overline{Q}_2Q_1Q_0 + MQ_2\overline{Q}_1Q_0 + MQ_2Q_1\overline{Q}_0;$$
$$T_1 = \overline{M}\,\overline{Q}_2\overline{Q}_1Q_0 + \overline{M}\,\overline{Q}_2Q_1Q_0 + \overline{M}Q_2\overline{Q}_1Q_0 + \overline{M}Q_2Q_1Q_0 + M\overline{Q}_2\overline{Q}_1Q_0 + MQ_2Q_1Q_0;$$
$$T_2 = \overline{M}\,\overline{Q}_2Q_1Q_0 + \overline{M}Q_2Q_1Q_0 + M\overline{Q}_2Q_1\overline{Q}_0 + MQ_2\overline{Q}_1\overline{Q}_0.$$

5. *Мінімізація функції збудження тригерів.* Перший етап мінімізації функцій збудження *T*-тригерів виконаємо із застосуванням карт Карно (рис. 3.17). Отримаємо функції збудження тригерів у вигляді МДНФ:

$$T_0 = \overline{M} + \overline{Q}_2\overline{Q}_1\overline{Q}_0 + \overline{Q}_2Q_1Q_0 + Q_2\overline{Q}_1Q_0 + Q_2Q_1\overline{Q}_0;$$
$$T_1 = \overline{M}Q_0 + \overline{Q}_2\overline{Q}_1Q_0 + Q_2Q_1Q_0;$$

$$T_2 = \overline{M}Q_1Q_0 + M\overline{Q}_2Q_1\overline{Q}_0 + MQ_2\overline{Q}_1\overline{Q}_0.$$

Таблиця 3.10

M	Початковий стан			Наступний стан			Сигнали на входах тригерів		
	Q_2^t	Q_1^t	Q_0^t	Q_2^{t+1}	Q_1^{t+1}	Q_0^{t+1}	T_2	T_1	T_0
0	0	0	0	0	0	1	0	0	1
0	0	0	1	0	1	0	0	1	1
0	0	1	0	0	1	1	0	0	1
0	0	1	1	1	0	0	1	1	1
0	1	0	0	1	0	1	0	0	1
0	1	0	1	1	1	0	0	1	1
0	1	1	0	1	1	1	0	0	1
0	1	1	1	0	0	0	1	1	1
1	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	1	0	1	0
1	0	1	0	1	1	0	1	0	0
1	0	1	1	0	1	0	0	0	1
1	1	0	0	0	0	0	1	0	0
1	1	0	1	1	0	0	0	0	1
1	1	1	0	1	1	1	0	0	1
1	1	1	1	1	0	1	0	1	0

Використовуючи відомі співвідношення (розподільний закон, правило де Моргана та ін.), остаточно одержимо:

$$T_0 = \overline{M} + \overline{Q}_2(\overline{Q_1 \oplus Q_0}) + Q_2(Q_1 \oplus Q_0) = \overline{M} + \overline{Q_2 \oplus Q_1 \oplus Q_0} = \overline{M(Q_2 \oplus Q_1 \oplus Q_0)};$$

$$T_1 = \overline{M}Q_0 + \overline{Q}_2\overline{Q}_1Q_0 + Q_2Q_1Q_0 = Q_0(\overline{M} + \overline{Q_2 \oplus Q_1}) = Q_0\overline{M(Q_2 \oplus Q_1)};$$

$$T_2 = \overline{M}Q_1Q_0 + M\overline{Q}_2Q_1\overline{Q}_0 + MQ_2\overline{Q}_1\overline{Q}_0 = \overline{M}Q_0Q_1 + \overline{Q_0 + \overline{M(Q_2 \oplus Q_1)}}.$$

6. *Складання логічної схеми автомата.* Логічна схема комбінаційної частини ЦА, що формує необхідні функції збудження тригерів, подана на рис. 3.17.

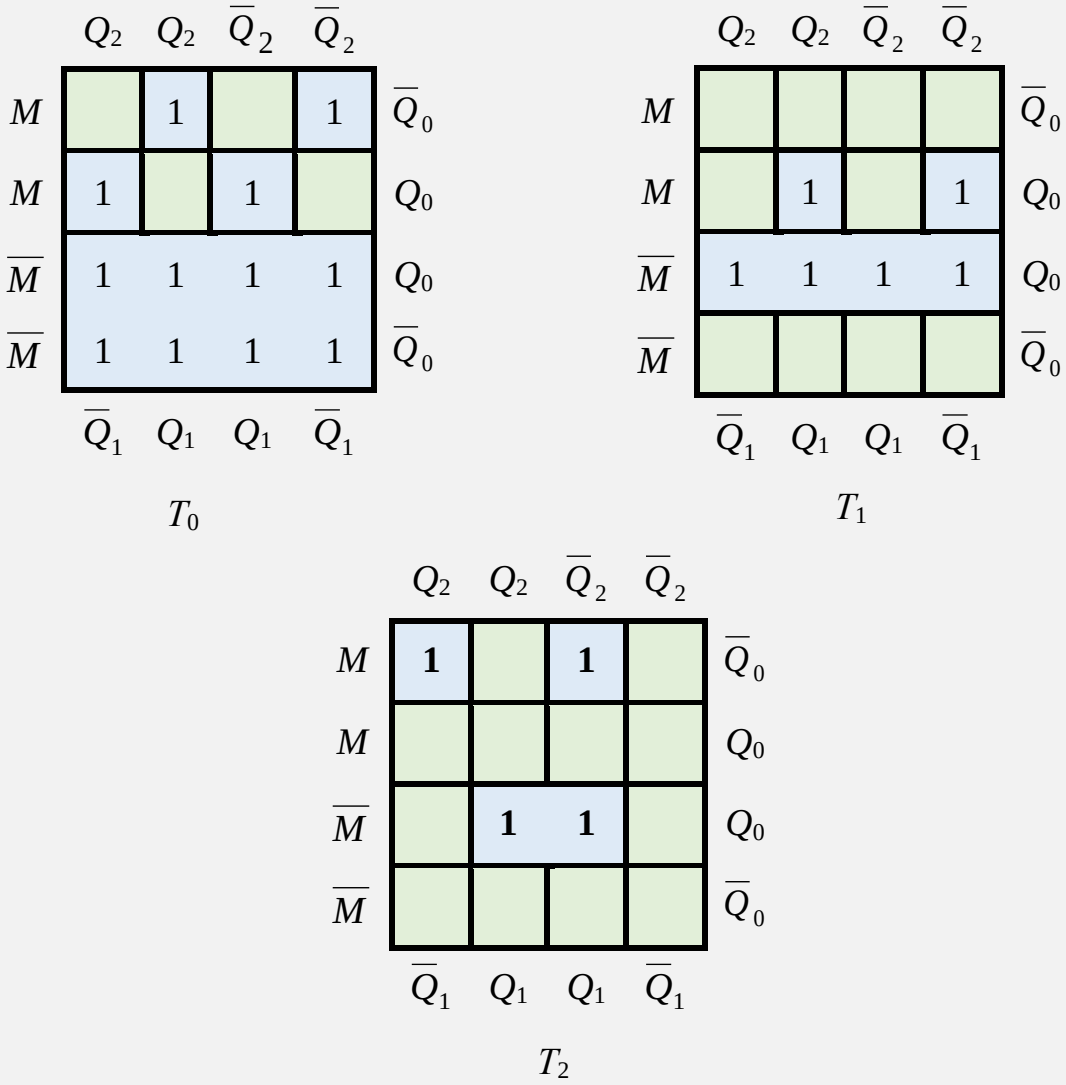


Рис. 3.16 – Мінімізація функцій збудження тригерів за допомогою карт Карно

Особливості синтезу ЦА з пам’яттю на основі *JK*-тригерів

При виконанні синтезу ЦА на основі *JK*-тригерів необхідно враховувати деякі особливості, пов’язані з тим, що цей тип тригерів порівняно з іншими має найбільш розвинуті логічні можливості.

Таблицю переходів *JK*-тригера можна подати у вигляді табл. 3.11. Із її аналізу можна зробити деякі важливі висновки.

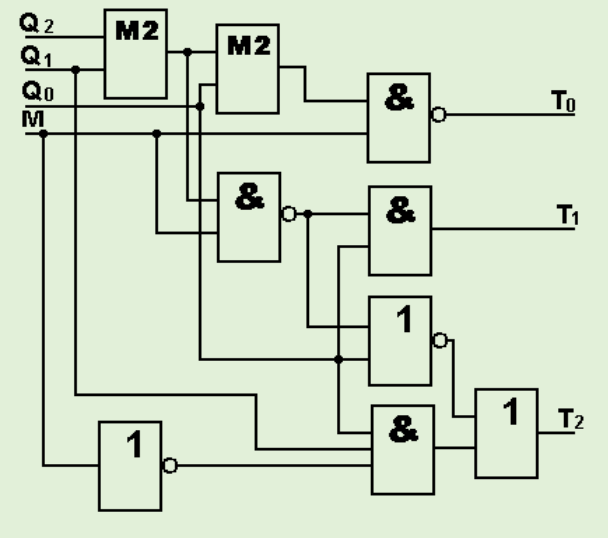


Рис. 3.17 – Схема комбінаційної частини синтезованого ЦА

Таблиця 3.11

Q^t	J	K	Q^{t+1}
0	0	*	0
0	1	*	1
1	*	1	0
1	*	0	1

По-друге, перехід тригера з нульового стану в будь-який інший (0 або 1) визначається виключно значенням J, а перехід тригера з одиничного стану – виключно значенням K. Це твердження можна записати таким чином:

$$Q^{t+1}\Big|_{Q^t=0} = J^t; \quad Q^{t+1}\Big|_{Q^t=1} = \overline{K^t}.$$

Ця властивість дозволяє отримати дуже прості функції збудження, для чого необхідно складати таблицю переходів синтезованого ЦА з урахуванням вмісту табл. 3.11. Наприклад, таблиця переходів для синтезованого дворежимного лічильника з використанням *JK*-тригерів (табл. 3.12) містить удвічі більше стовпчиків для вхідних сигналів тригерів (функцій збудження), ніж табл. 3.10, але в кожному рядку цих стовпчиків наявні невизначені значення сигналів.

Як розглядалося раніше, при використанні карт Карно ці значення довизначаються на власний розсуд проєктувальника з метою підвищення ефективності мінімізації, що дозволяє отримати більш прості ЛФ.

Таблиця 3.12

M	Початковий стан			Наступний стан			Сигнали на входах тригерів					
	Q_2^t	Q_1^t	Q_0^t	Q_2^{t+1}	Q_1^{t+1}	Q_0^{t+1}	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	0	1	0	*	0	*	1	*
0	0	0	1	0	1	0	0	*	1	*	*	1
0	0	1	0	0	1	1	0	*	*	0	1	*
0	0	1	1	1	0	0	1	*	*	1	*	1
0	1	0	0	1	0	1	*	0	0	*	1	*
0	1	0	1	1	1	0	*	0	1	*	*	1
0	1	1	0	1	1	1	*	0	*	0	1	*
0	1	1	1	0	0	0	*	1	*	1	*	1
1	0	0	0	0	0	1	0	*	0	*	1	*
1	0	0	1	0	1	1	0	*	1	*	*	0
1	0	1	0	1	1	0	1	*	*	0	0	*
1	0	1	1	0	1	0	0	*	*	0	*	1
1	1	0	0	0	0	0	*	1	0	*	0	*
1	1	0	1	1	0	0	*	0	0	*	*	1
1	1	1	0	1	1	1	*	0	*	0	1	*
1	1	1	1	1	0	1	*	0	*	1	*	0

Практичне завдання «Синтез ЦА канонічним методом»

Запитання і завдання

1. Поясніть поняття ЦА послідовнісного типу. Чим вони відрізняються від комбінаційних ЦА?
2. Які логічні схеми називаються тригерами? Наведіть та поясніть класифікацію тригерів.
3. Зобразіть схему асинхронного *RS*-тригера на елементах АБО-НІ й поясніть її роботу.
4. Зобразіть схему асинхронного *RS*-тригера на елементах І-НІ та поясніть її роботу.
5. Зобразіть схему синхронного *RS*-тригера й поясніть її роботу.
6. Зобразіть схему найпростішого *JK*-тригера та поясніть її роботу.
7. Зобразіть схему синхронного *JK*-тригера й поясніть її роботу.
8. З якою метою більшість тригерів будують за двоступеневою схемою?
9. Як побудувати *T*-тригер на основі: а) *JK*-тригера; б) *RS*-тригера; в) *D*-тригера?
10. Наведіть та поясніть класифікацію регістрів.
11. Зобразіть схему регістра пам’яті й поясніть її роботу.
12. Зобразіть схему регістра зсуву: а) праворуч; б) ліворуч та поясніть її роботу.
13. Наведіть та поясніть класифікацію лічильників.
14. Зобразіть схему трирозрядного лічильника: а) додаючого; б) віднімаючого. Поясніть її роботу.
15. Зобразіть схему (фрагмент) реверсивного лічильника й поясніть її роботу.
16. Перелічіть та поясніть основні етапи синтезу ЦА з пам’яттю.
17. Поясніть особливості синтезу ЦА з пам’яттю на основі *JK*-тригерів.

Розділ 4. ВИКОНАННЯ АРИФМЕТИЧНИХ ОПЕРАЦІЙ В КОМП'ЮТЕРІ

4.1. Форми подання двійкових чисел в комп'ютері

Двійкові числа в комп'ютерах розміщуються у комірках пам'яті, причому для кожного розряду числа виділяється окрема комірка, що зберігає один біт інформації. *Сукупність комірок, призначених для розміщення одного двійкового числа, називають розрядною сіткою*. Довжина розрядної сітки (число комірок *n* у розрядній сітці) обмежена і залежить від конструктивних особливостей комп'ютера (його архітектури). В сучасних комп'ютерах довжина розрядної сітки може розглядатися як змінна величина, тобто зберігаються та обробляються числа різної розрядності – 8 (байт), 16 (слово), 32 (подвійне слово) або 64 (довге слово) розрядів.

Розміщення розрядів числа у розрядній сітці може відбуватися різними способами. Спосіб розміщення визначається формою подання двійкових чисел в комп'ютері. Розрізняють дві форми подання двійкових чисел: із *фіксованою комою* і з *«плаваючою» комою*. Іноді ці форми називають відповідно *природною* і *напівлогарифмічною*.

Припустимо, що в розрядній сітці необхідно розмістити двійкове число, що містить цілу і дробову частини. Якщо для розміщення цілої частини числа виділяється *k* комірок *n*-розрядної сітки, то (якщо не враховувати знак) для розміщення дробової частини залишиться *n* - *k* вільних комірок (рис. 4.1).

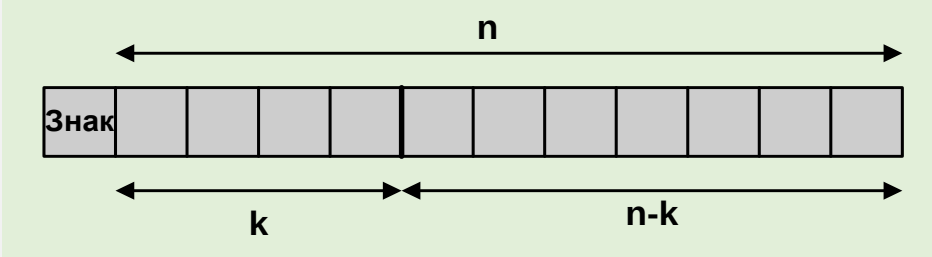


Рис. 4.1 – Форма подання двійкових чисел із фіксованою комою

Така форма подання двійкових чисел називається *представленням з фіксованою комою*.

Якщо кількість розрядів у дробовій частині числа перевищують n - k, то деякі молодші розряди виходять за межі розрядної сітки і не будуть сприйматися обчислювальним пристроєм. Отже, будь-яке двійкове число, менше, ніж одиниця молодшого розряду розрядної сітки, сприймається як нуль і називається *машинним нулем*.

У результаті відкидання молодших розрядів дробової частини числа, розташованої за межами розрядної сітки, виникає похибка подання. Максимальне значення абсолютної похибки подання не перевищує одиниці молодшого розряду сітки.

В універсальних комп'ютерах форма з фіксованою комою, у зв'язку з властивою їй низькою точністю, застосовується лише для подання цілих чисел. Основною є форма подання чисел з рухомою (або «плаваючою», float) комою. Її використання дозволяє суттєво розширити діапазон і зменшити відносну похибку.

У цій формі числа подаються у вигляді суми деякого ступеня основи системи числення і цифрової частини, що має вигляд правильного дробу,

$$N = \pm aq^{\pm p},$$

де p – порядок числа; a – його мантиса. Мантиса і порядок є знаковими числами. Тому для позначення знаків у розрядній сітці відводяться два додаткові розряди. Знак усього числа збігається зі знаком мантиси.

Під час запису двійкового числа у показовій формі в розрядній сітці використовуються дві групи розрядів (без урахування знакових розрядів мантиси і порядку). Перша група (k розрядів) призначена для розміщення коду мантиси, друга (n - k розрядів) – для розміщення коду порядку (рис. 4.2).

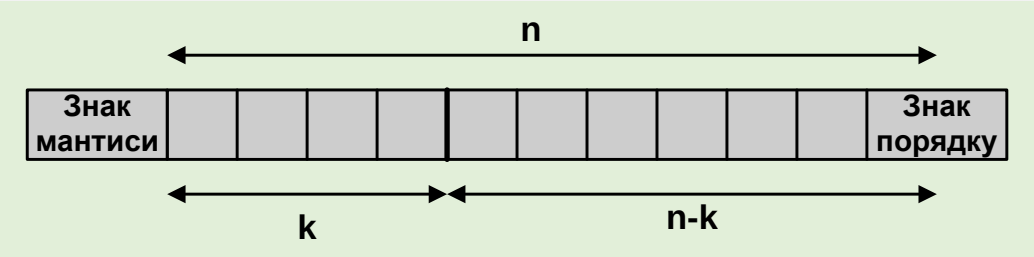


Рис. 4.2 – Форма подання двійкових чисел із рухомою комою

Отже, мантиса числа може мати необмежену кількість різних значень, менших за одиницю, при відповідних значеннях порядку (тобто кома може «плавати»). *Подання числа у показовій формі, що не має в старшому розряді*

мантиси нуля, називають нормалізованим. Усі інші подання є ненормалізованими. У нормалізованій формі значення мантиси завжди більші або дорівнюють 1/2, але не перевищують одиниці.

В обчислювальних пристроях із «плаваючою» комою всі числа зберігаються в нормалізованому вигляді, при цьому не втрачаються молодші розряди мантиси і підвищується точність обчислень. Якщо після виконання будь-якої арифметичної операції результат виявляється ненормалізованим, то перед занесенням числа в пам’ять виконують його нормалізацію, тобто зсув мантиси ліворуч на відповідну кількість розрядів, і зменшення порядку числа на відповідну кількість одиниць.

Показова форма подання чисел має й свої вади, основною з яких є порівняно висока складність виконання арифметичних операцій, а отже, і більша вимогливість до ресурсів обчислювального пристрою. Це обмежує її застосування, наприклад, у спеціалізованих радіотехнічних обчислювальних пристроях, у системах управління технологічними процесами, передачі та обробки інформації у реальному часі в телекомунікаційних системах.

4.2. Натуральний, прямий, обернений і доповняльний коди двійкових чисел

Залежно від способу обробки бітів, розміщених у розрядній сітці, розрізняють два види кодів: паралельний, коли в кожний момент часу всі розряди сітки доступні для обробки, і послідовний, коли в кожний момент часу доступний один розряд сітки. Числа, подані паралельним кодом, доступні за один такт, а числа, подані послідовним кодом, – за n тактів, де n – розрядність сітки. Якщо розрядність числа перевищує довжину сітки, то його обробку виконують частинами.

Натуральним кодом називають подання числа як цілого беззнакового у двійковій системі числення. Діапазон подання чисел у натуральному коді для n-розрядної сітки становить від 0 до 2^n - 1, тобто, наприклад, для 8-розрядної сітки – від 0 до 255.

Для подання цілих знакових чисел використовують прямий, обернений і доповняльний коди. Старший розряд сітки є знаковим. Значення цього розряду дорівнює 0 для додатних чисел і 1 – для від’ємних. В інших розрядах розміщується модуль числа.

Якщо до натурального коду цілого числа додати знаковий розряд, то одержуємо запис числа у *прямому коді* (ПК). Домовимося знаковий розряд розташовувати зліва і відокремлювати від розрядів модуля числа точкою, наприклад: $+ 6_{(10)} = 0, 110_{(ПК)}$; $- 6_{(10)} = 1, 110_{(ПК)}$.

Використання ПК забезпечує виконання операції додавання двох додатних чисел звичайним способом без будь-яких труднощів – не варто лише робити перенос одиниці старшого розряду модуля суми у знаковий розряд. Тобто при виконанні арифметичних операцій над ПК двійкових чисел знаковий розряд і розряди модуля не можна розглядати як єдине ціле.

У цьому можна переконатися, розглянувши такий приклад:

Правильно:	Неправильно:
0, 0110	0,0110
+ 0, 1010	+ 0,1010
0,10000	1,0000

Однак виконання операції віднімання одного числа від іншого шляхом безпосереднього додавання їхніх ПК неможливо. Неважко також помітити, що в ПК нуль має два можливі зображення: - 0 = 1,000... і + 0 = 0,000..., що ускладнює інтерпретацію результатів виконання арифметичних операцій в комп’ютері.

Іншою формою запису двійкових чисел є *обернений код* (ОК).

ОК двійкового від’ємного числа утворюється з ПК, рівного йому за модулем додатного числа, шляхом інвертування значень усіх його розрядів. Або: *ОК від’ємного числа* утворюється шляхом інверсії всіх розрядів модуля цього числа, записаного у ПК. Знаковий розряд при цьому зберігає значення 1. Наприклад, $6_{(10)} = 1.110_{(ПК)} = 1.001_{(ОК)}$.

При виконанні арифметичних операцій над двійковими числами, поданими в ОК, знаковий розряд і розряд модуля числа можна розглядати як єдине ціле (перенос одиниці зі старшого розряду модуля суми в знаковий розряд не приводить до помилкового результату), але нуль, як і раніше, має два зображення – «додатне» і «від’ємне». Слід зазначити, що отриманий при додаванні від’ємний результат також утворюється в ОК. У цьому випадку число може бути перетворене у ПК інверсією всіх значущих розрядів (розрядів модуля).

Наприклад:

(+6) + (-6) = (-0).

$$\begin{array}{r} 0,110 \\ +\underline{1,001} \\ 1,111_{(ОК)} = 1,000_{(ПК)} \end{array}$$

Найбільш поширеним в комп'ютерах є подання від'ємних двійкових чисел за допомогою *доповняльного коду* (ДК).

ДК від'ємного числа утворюється з його прямого коду за правилом:

- ✓ у знаковому розряді залишається одиниця;
- ✓ розряди модуля числа інвертуються;
- ✓ до молодшого розряду додається одиниця.

Очевидно, що ДК від'ємного числа утворюється з його ОК додаванням одиниці до молодшого розряду.

Наприклад: $- 6_{(10)} = 1,010_{(ДК)}$.

Дійсно, для числа - 6 маємо:

$$\begin{array}{r} 1,110_{(ПК)} \\ 1,001_{(ОК)} \\ +\quad\underline{1} \\ 1,010_{(ДК)}. \end{array}$$

Зворотний перехід від ДК до ПК або ОК відбувається за тими ж правилами. Головною перевагою ДК є те, що цифра 0 у ньому має єдине подання: 0.000... Саме тому для подання від'ємних чисел у сучасних комп'ютерах використовується переважно ДК.

Неправильний дріб (число, що має цілу частину) зі знаком записують у різних кодах за допомогою традиційного роздільника – коми між цілою та дробовою частиною. Наприклад: $- 118,375_{(10)} = 1.0001,101_{(ДК)}$.

Слід пам'ятати, що для кодування додатних чисел застосовується тільки ПК, хоча можна сказати, що для таких чисел ДК і ОК збігаються з прямим кодом.

Операція одержання ДК від'ємного числа з ПК, рівного йому за модулем додатного числа, називається *операцією доповнення*. Ця операція полягає в інвертуванні всіх розрядів вихідного коду (включаючи знаковий) та додаванні до молодшого розряду одиниці.

Таким чином, сформулюємо таке правило: *у системі двійкових чисел зі знаком заміна додатного числа на рівне йому за модулем від'ємне і, навпаки, від'ємного на додатне, здійснюється шляхом застосування до коду цього числа операції доповнення*.

Така властивість подання від'ємних чисел у ДК дозволяє при виконанні арифметичних операцій взагалі відмовитися від операції віднімання, замінивши її операцією додавання із числом, що має знак, протилежний знаку числа, яке віднімається.

4.3. Алгоритми виконання арифметичних операцій над двійковими числами зі знаком

Додавання двійкових чисел зі знаком

Очевидно, що при додаванні чисел із знаком можуть виникати переноси одиниці зі старшого розряду модуля суми до знакового розряду (домовимося позначати його P_1) та із знакового розряду – ліворуч за межі розрядної сітки, у розряд переповнення (P_2). Через використання розглянутих раніше кодів, у яких знак числа позначається тими ж цифрами, що і розряди модуля, переповнення розрядної сітки може виникати навіть у випадку додавання чисел із різними знаками, коли модуль результату не перевищує модуля будь-якого операнда. При додаванні ж двох від'ємних чисел перенесення одиниці до розряду переповнення відбувається завжди.

При виникненні переповнення розрядної сітки для одержання правильного результату додавання необхідно застосовувати таке правило:

– якщо $P_1 \oplus P_2 = 0$, одиниця в розряді переповнення ігнорується (відкидається);

– якщо $P_1 \oplus P_2 = 1$, необхідно зсунути число на один розряд праворуч (або зсунути позицію точки на один розряд ліворуч).

Додавання дробових і цілих двійкових чисел, поданих у формі з фіксованою комою, відбувається однаково, тобто порядок додавання не залежить від розташування коми. Тому операцію додавання розглянемо на прикладі додавання цілих чисел.

Приклади:

1) додавання двох додатних чисел (без переповнення розрядної сітки):

$$\begin{array}{rr} 0{,}100111 & 39 \\ \underline{+\ 0{,}001101} & \underline{+13} \\ 0{,}110100 & 52 \end{array} \quad ;$$

$P_1 \oplus P_2 = 0$ – результат коректний і остаточний.

2) додавання двох додатних чисел (з переповненням розрядної сітки):

$$\begin{array}{rr} 0{,}01101 & 13 \\ \underline{+\ 0{,}10011} & \underline{+19} \\ 1{,}00000 & 32 \end{array} ;$$

$P_1 \oplus P_2 = 1$. Результат некоректний, тому що відбулося переповнення розрядної сітки. Зсуваючи число на один розряд праворуч, остаточно маємо $0.100000_{(ПК)} = 32_{(10)}$;

3) додавання двох чисел із різними знаками (без переповнення розрядної сітки):

$$\begin{array}{rr} 1{,}001100 & -\ 52 \\ \underline{+\ 0{,}001101} & \underline{+\ 13} \\ 1{,}011001 & -\ 39 \end{array} ;$$

$P_1 \oplus P_2 = 0$. Результат коректний, але тому що він є від'ємним, для перевірки правильності розв'язання необхідно перетворити його у прямий код. Остаточно маємо $1.100111_{(ПК)} = 39_{(10)}$;

4) додавання двох чисел, рівних за модулем і різних за знаком:

$$\begin{array}{rr} 1{,}011001 & -\ 39 \\ \underline{+\ 0{,}100111} & \underline{+\ 39} \\ 10{,}000000 & 0 \end{array} ;$$

$P_1 \oplus P_2 = 0$. Результат коректний, якщо не брати до уваги одиницю у розряді переповнення.

Додавання двох від'ємних чисел виконується аналогічно до прикладів 1, 2 (залежно від значення виразу $P_1 \oplus P_2$). Оскільки результат у цьому випадку завжди від'ємний, для перевірки правильності розв'язання необхідно перетворити його у прямий код, аналогічно до прикладу 3.

Таким чином:

✓ правильність виконання операцій додавання обов'язково повинна перевірятися шляхом аналізу значення виразу $P_1 \oplus P_2$, щоб уникнути одержання некоректного результату, що виникає при переповненні розрядної сітки, при цьому: якщо $P_1 \oplus P_2 = 0$, одиниця в розряді переповнення ігнорується (відкидається); якщо $P_1 \oplus P_2 = 1$, необхідно зсунути число на один розряд праворуч;

✓ правило перевірки коректності результату додавання двійкових чисел також можна сформулювати в такий спосіб: якщо знак операндів однаковий, а знак суми протилежний, результат є некоректним. При додаванні двох операндів із різними знаками результат завжди коректний, якщо не брати до уваги одиницю у розряді переповнення.

Множення і ділення двійкових чисел із фіксованою комою

Множення двійкових чисел звичайно виконують у ПК. Знак добутку визначають за знаковими розрядами співмножників згідно з таким загальновідомим правилом: якщо знаки операндів однакові, то результат додатний, у протилежному випадку – знак добутку від'ємний.

Знак добутку двох чисел не впливає на алгоритм виконання операції множення модулів цих чисел.

Часто використовують спосіб множення, процедура якого аналогічна до процедур множення вручну. У цьому випадку результат одержують додаванням часткових добутків. Кожний частковий добуток удвічі перевищує попередній, що відповідає його зсуванню ліворуч на один розряд.

Наприклад:

```
   1101          13
 x 1011        x 11
   1101          13
  1101         + 13
+ 0000           143
1101
10001111
```

Характерно, що розрядність добутку двійкових чисел удвічі перевищує розрядність співмножників. Якщо у множенні беруть участь мантиси, тобто правильні дроби, то молодші розряди, що виходять за межі розрядної сітки, можуть бути відкинуті без округлення або з округленням.

Операція ділення також виконується способом, аналогічним до застосовуваного при діленні вручну, що наочно ілюструє приклад ділення двох чисел 506 : 2 = 22, тобто 0.111111010 : 0.10111 = 0.10110. Знак частки визначають аналогічно до знака добутку. Застосоване при діленні віднімання дільника виконують шляхом додавання його доповняльного коду.

У цьому прикладі використаний так названий алгоритм без відновлення остачі.

0.111111010	– ділене додатне –
+ 1.01001	перше віднімання дільника
10.010001	1 – результат додатний
+ 1.01001	– друге віднімання дільника
1.110100	0 – від'ємний результат
+ 0.10111	– додавання дільника
100.010111	1 – результат додатний
+ 1.01001	– третє віднімання дільника
10.000000	1 – остача дорівнює нулю

Він передбачає таку послідовність дій:

- ✓ із діленого віднімається дільник (додається дільник, записаний у доповняльному коді);
- ✓ якщо остача додатна, перша цифра частки дорівнює одиниці, у протилежному випадку – 0;
- ✓ остача зсувається ліворуч, і до неї додається дільник із знаком, зворотним знаку остачі;
- ✓ знак наступної остачі визначає наступну цифру частки;
- ✓ ці дії повторюють доти, поки не утвориться необхідне число розрядів частки або нульова остача.

Слід зазначити, що оскільки цей алгоритм передбачає додавання чисел (остач і дільника) тільки з протилежними знаками, то всі розряди проміжних сум, старші за знаковий, слід ігнорувати.

Практичне завдання «Виконання арифметичних операцій в комп'ютері»

Виконання арифметичних операцій у пристроях із рухомою комою

Операція додавання у пристроях із рухомою комою відбувається у чотири етапи:

1. Порівнюються порядки доданків: менший порядок збільшується до більшого. При цьому відповідним чином корегується мантиса числа, яке перетворюється.
2. Виконується перетворення мантис у додаткові коди.
3. Виконується додавання мантис за правилами, розглянутими вище для чисел із фіксованої комою.

4. До суми приписується порядок доданків, і в разі необхідності виконується нормалізація результату.

Операція множення чисел, поданих у формі з рухомою комою, також виконується у чотири етапи:

1. Визначається знак добутку.
2. Перемножуються мантиси співмножників за правилами для чисел із фіксованою комою.
3. Обчислюється порядок добутку алгебраїчним додаванням порядків співмножників за правилами додавання цілих чисел із знаком.
4. Виконується нормалізація отриманого результату у випадку її необхідності.

Ділення чисел у пристроях із плаваючою комою виконується так само, як і множення.

Практичне завдання «Виконання арифметичних операцій над числами з рухомою комою»

Висновки

1. Операція віднімання в обчислювальних пристроях виконується як операція додавання із числом протилежного знака (доповненням модуля від'ємника).
2. Операції множення і ділення в обчислювальних пристроях найчастіше виконуються у вигляді циклічної послідовності операцій додавання та зсування.

Таким чином, виконання всіх арифметичних операцій в комп'ютері так чи інакше зводиться до виконання послідовності дій, серед яких основною є додавання чисел, допоміжними – зсування, інверсія й ін. Тому всі ЦА, що виконують арифметичні операції, містять у своєму складі один або декілька суматорів.

4.4. Суматори двійкових чисел

Суматором називається ЦА, що здійснює операції арифметичного додавання кодів двох чисел.

Суматори широко застосовуються у цифровій техніці. Наприклад, головна частина мікропроцесора – арифметико-логічний пристрій (АЛП,) основою якого є багаторозрядний суматор.

За ознакою реалізованої мікрооперації суматори поділяють на *суматор за модулем два*, *напівсуматор* та *повний суматор*.

За кількістю розрядів оброблюваних двійкових чисел суматори бувають *одноразрядні* й *багаторозрядні*.

За способом подання вхідних і вихідних кодових слів суматори можуть бути *паралельні* й *послідовні*.

За способом функціонування суматори можуть бути як *комбінаційними*, так і *послідовнісними* (*накопичувальними*).

За ознакою використовуваної системи числення суматори поділяються на *двійкові* й *десяткові*.

Комбінаційні одноразрядні суматори

Якщо подати правила додавання двійкових чисел, що були розглянуті раніше, у табличному вигляді, отримуємо таблицю функціонування двійкового одноразрядного суматора (табл. 4.1), у якій: A_i, B_i – одноразрядні доданки: S_i – молодший розряд суми; P_i – перенос до i-го розряду з попереднього; P_{i+1} – перенос до наступного старшого розряду з i-го (старший розряд суми).

P_{i+1} ← ← P_i

A_i

\+ B_i

S_i

З аналізу цієї таблиці видно, що її верхня половина без урахування стовпчиків P_i і P_{i+1} (виділений фрагмент) є таблицею істинності логічного елемента «Виключне АБО» (суматора за модулем два), ЛФ якого має вигляд

$$A_i \oplus B_i = A_i\overline{B_i} \vee \overline{A_i}B_i = (A_i \vee B_i)(\overline{A_i} \vee \overline{B_i}).$$

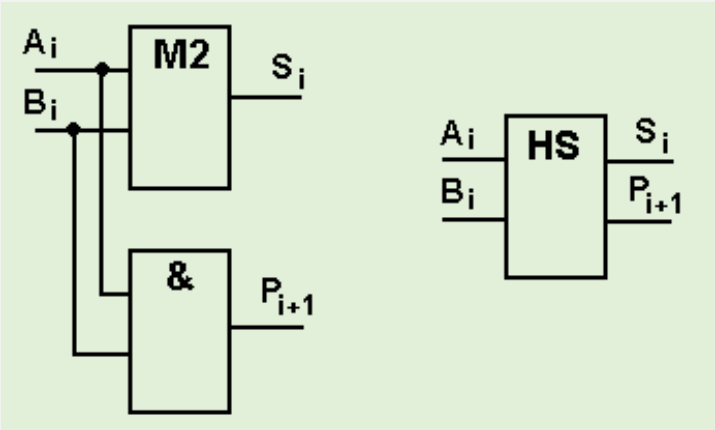
З аналізу значень стовпчика P_{i+1} тієї ж верхньої половини таблиці (для $P_i = 0$) очевидно, що для реалізації операції додавання двох одноразрядних двійкових чисел з урахуванням переносу одиниці до старшого розряду ($P_{i+1} = 1$ тільки при $A_i = B_i = 1$, таким чином $P_{i+1} = A_iB_i$) суматор по модулю два треба доповнити елементом І, як показано на рис. 4.3, а. Такий функціональний елемент отримав назву одноразрядного напівсуматора, або одноразрядного суматора на два входи (ОС-2). УГП напівсуматора показано на рис. 4.3, б.

Таблиця 4.1

P_i	A_i	B_i	S_i	P_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Очевидним недоліком одноразрядного напівсуматора є те, що він при виконанні операції додавання не враховує

переносу з попереднього розряду P_i. Цей недолік усунене у схемі повного однорозрядного суматора, або однорозрядного суматора на три входи − (ОС-3).



а *б*

Рис. 4.3 – Двійковий однорозрядний напівсуматор

На підставі табл. 4.1 запишемо ЛФ виходів Si і Pi+1:

$$S_i = \overline{P_i}\,\overline{A_i}B_i \vee \overline{P_i}A_i\overline{B_i} \vee P_i\overline{A_i}\,\overline{B_i} \vee P_iA_iB_i;$$
$$P_{i+1} = \overline{P_i}A_iB_i \vee P_i\overline{A_i}B_i \vee P_iA_i\overline{B_i} \vee P_iA_iB_i.$$

Використовуючи різні варіанти перетворення цих функцій, можна реалізувати велику кількість структур однорозрядних двійкових суматорів. Як приклад розглянемо побудову схеми суматора з використанням суматорів по модулю два і ЛЕ І-НІ.

Застосовуючи розподільний закон і правило де Моргана, отримаємо

$$S_i = \overline{P_i}(\overline{A_i}B_i \vee A_i\overline{B_i}) \vee P_i(\overline{A_i}\,\overline{B_i} \vee A_iB_i) = \overline{P_i}(A_i \oplus B_i) \vee P_i\overline{(A_i \vee B_i)(\overline{A_i} \vee \overline{B_i})} =$$
$$= \overline{P_i}(A_i \oplus B_i) \vee P_i\overline{(A_i \oplus B_i)} = A_i \oplus B_i \oplus P_i.$$

Таким чином, для формування молодшого розряду суми S_i у схемі ОС-3 необхідно мати два суматори по модулю два.

Аналізуючи стовпчики A_i, B_i, P_i, P_{i+1} табл. 4.1, можна помітити, що вихідний сигнал схеми формування переносу до старшого розряду (старшого розряду суми) P_{i+1} збігається з більшістю вхідних сигналів A_i, B_i, P_i (якщо на входах одиниць більше, ніж нулів, то вихідний сигнал дорівнює 1 і навпаки). Такий комбінаційний вузол називають *мажоритарним елементом*.

Мінімізуємо ЛФ для P_{i+1} за допомогою карти Карно (рис. 4.4).

	A_i	A_i	$\overline{A_i}$	$\overline{A_i}$
B_i	1	1	1	0
$\overline{B_i}$	0	1	0	0
	$\overline{P_i}$	P_i	P_i	$\overline{P_i}$

Рис. 4.4. Карта Карно для мажоритарного елемента з трьома входами

Одержимо ЛФ мажоритарного елемента з трьома входами у вигляді мінімальної ДНФ

$$P_{i+1} = A_i B_i \vee P_i B_i \vee P_i A_i .$$

Подаємо отриману функцію у базисі І-НІ

$$P_{i+1} = \overline{\overline{A_i B_i} \wedge \overline{P_i B_i} \wedge \overline{P_i A_i}} .$$

Схема однорозрядного комбінаційного суматора, побудованого на основі отриманих виразів, подана на рис. 4.5.

Інший варіант побудови однорозрядного комбінаційного суматора можна одержати, якщо, використовуючи розподільний закон і операцію склеювання, подати вираз для P_{i+1} у вигляді

$$P_{i+1} = \overline{P_i} A_i B_i \vee P_i \overline{A_i} B_i \vee P_i A_i \overline{B_i} \vee P_i A_i B_i = P_i (\overline{A_i} B_i \vee A_i \overline{B_i}) \vee A_i B .$$

Якщо взяти до уваги, що

$$\overline{A_i} B_i \vee A_i \overline{B_i} = A_i \oplus B_i = S_{iOC-2}, \text{ а } P_{i+1} = P_i S_{iOC-2} \vee P_{i+1OC-2},$$

де S_{iOC-2} і $P_{i+1OC-2}$ – ЛФ суми й переносу на виході однорозрядного напівсуматора, остаточно отримаємо

$$P_{i+1} = P_i S_{iOC-2} \vee P_{i+1OC-2} .$$

Відповідно схема однорозрядного суматора на напівсуматорах має вигляд, показаний на рис. 4.6, а. УГП однорозрядного суматора показано на рис. 4.6, б.

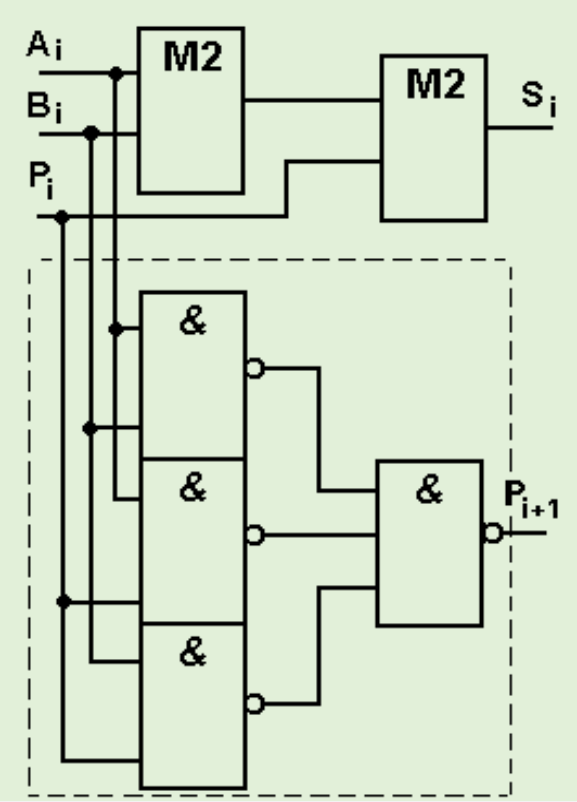
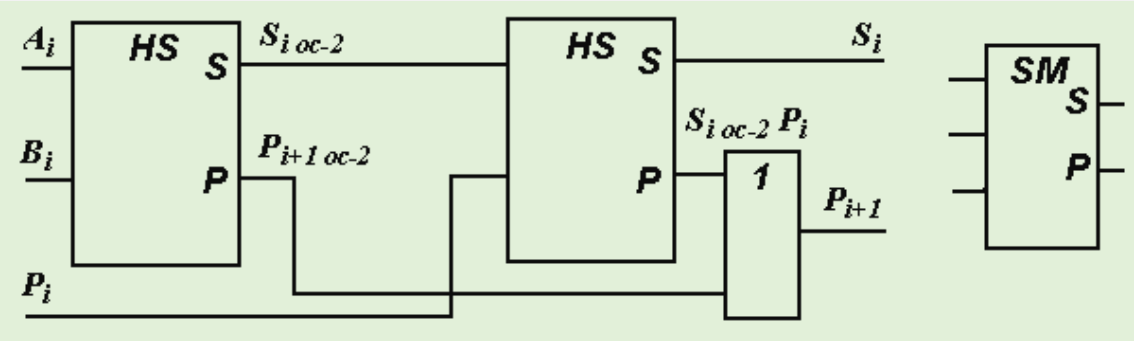


Рис. 4.5 – Однорозрядний комбінаційний суматор



а б

Рис. 4.6 – Однорозрядний комбінаційний суматор та його УГП

В УГП напівсуматора та суматора дозволяється використовувати замість сполучень літер *HS* і *SM* символ ∑.

Багаторозрядні комбінаційні суматори

Принцип додавання багаторозрядних двійкових чисел полягає в тому, що в кожному з розрядів виконуються однотипні дії: визначається цифра суми шляхом додавання по модулю 2 цифр доданків і переносу з попереднього розряду та формується перенос до наступного розряду. Ці дії реалізовуються двійковим однорозрядним суматором. Така однотипність

дій при додаванні окремих розрядів багаторозрядних чисел дозволяє реалізувати багаторозрядні суматори як у послідовному вигляді – за рахунок послідовного виконання додавання розрядів за допомогою одного одноразрядного суматора, так і в паралельному – за допомогою схеми, що вміщує кілька однотипних фрагментів (відповідно до кількості розрядів доданків), кожний з яких містить свій одноразрядний суматор.

Паралельний багаторозрядний суматор складається з такої кількості одноразрядних суматорів, яка дорівнює кількості розрядів чисел, які додаються.

Усі розряди доданків повинні одночасно надходити до пристрою додавання. Сигнал переносу передається від розряду до розряду послідовно, утворюючи на виході значення старшого розряду суми. Схема паралельного дворозрядного комбінаційного суматора з послідовним переносом показана на рис. 4.7.

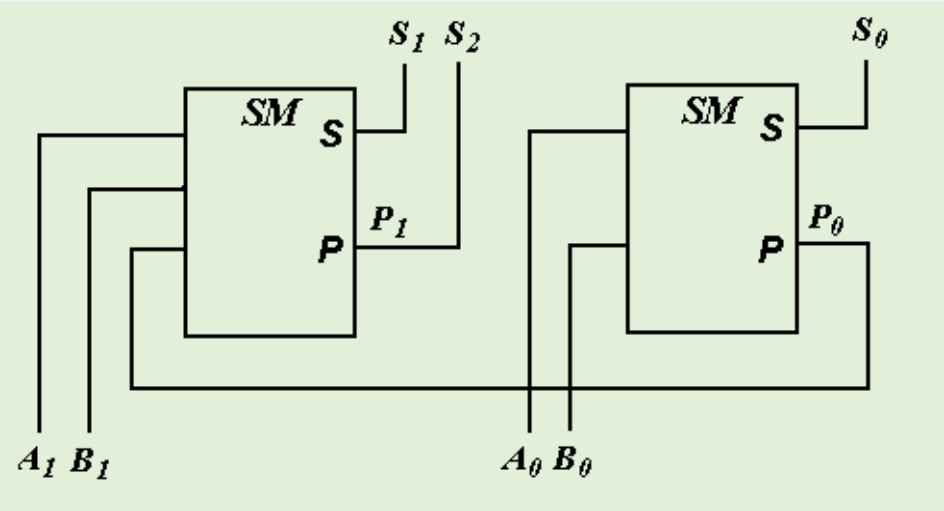


Рис. 4.7 – Дворозрядний комбінаційний паралельний суматор з послідовним переносом

Послідовний багаторозрядний суматор складається з одного одноразрядного суматора й елемента затримки – D-тригера, що здійснює затримку сигналу переносу на один робочий такт – до надходження до входів суматора наступних (старших) розрядів доданків. Схема послідовного суматора показана на рис. 4.8.

Явна перевага суматора послідовної дії полягає у малих апаратних витратах на його побудову. Але такий суматор порівняно з більш складним паралельним має меншу швидкодію.

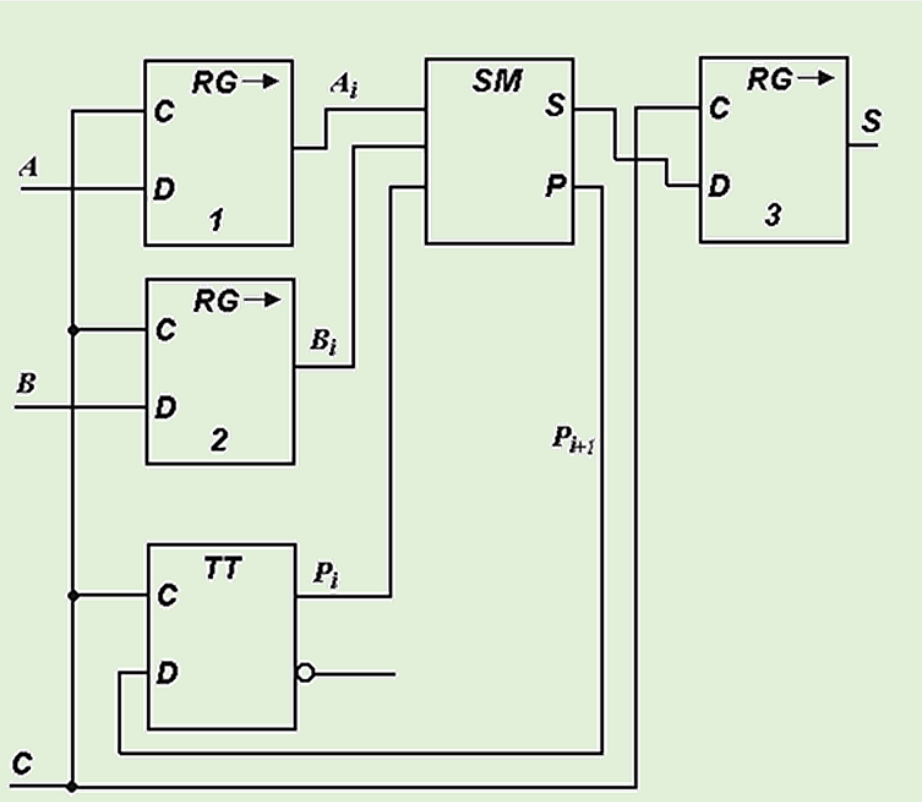


Рис. 4.8 – Послідовний багаторозрядний суматор

Пристрій додавання та віднімання чисел із знаком

Комбінаційний суматор разом з деякими допоміжними елементами може бути використаний не тільки для додавання, але також для віднімання двійкових чисел із знаком. Зокрема, для виконання віднімання у доповняльних кодах за допомогою суматора до від'ємника необхідно застосувати операцію доповнення, що відповідає зміні знака операнда на протилежний. Для цього в схемі пристрою додавання і віднімання необхідно мати пристрій, що виконує за необхідності доповнення від'ємника (інверсію всіх розрядів вихідного коду, включаючи знаковий, та додавання одиниці до молодшого розряду).

Схема чотирирозрядного пристрою додавання та віднімання двійкових чисел у доповняльних кодах подана на рис. 4.9.

Тут

- ✓ $[X\partial] = \{x_1, x_2, x_3, x_4\}$ – доповняльний код операнда X (оскільки X додатний, доповняльний код збігається з прямим);
- ✓ $[Y\partial] = \{y_1, y_2, y_3, y_4\}$ – доповняльний код операнда Y;
- ✓ $[S\partial] = \{s_1, s_2, s_3, s_4\}$ – доповняльний код результату.

Суматор SM виконує операцію

$$[S\partial]=[X\partial]+[Y\partial]+P1.$$

Якщо сигнал керування $Z = P1 = 0$, то коди операндів подаються на входи суматора без зміни, і відбувається їх додавання:

$$[S\partial]=[X\partial]+[Y\partial].$$

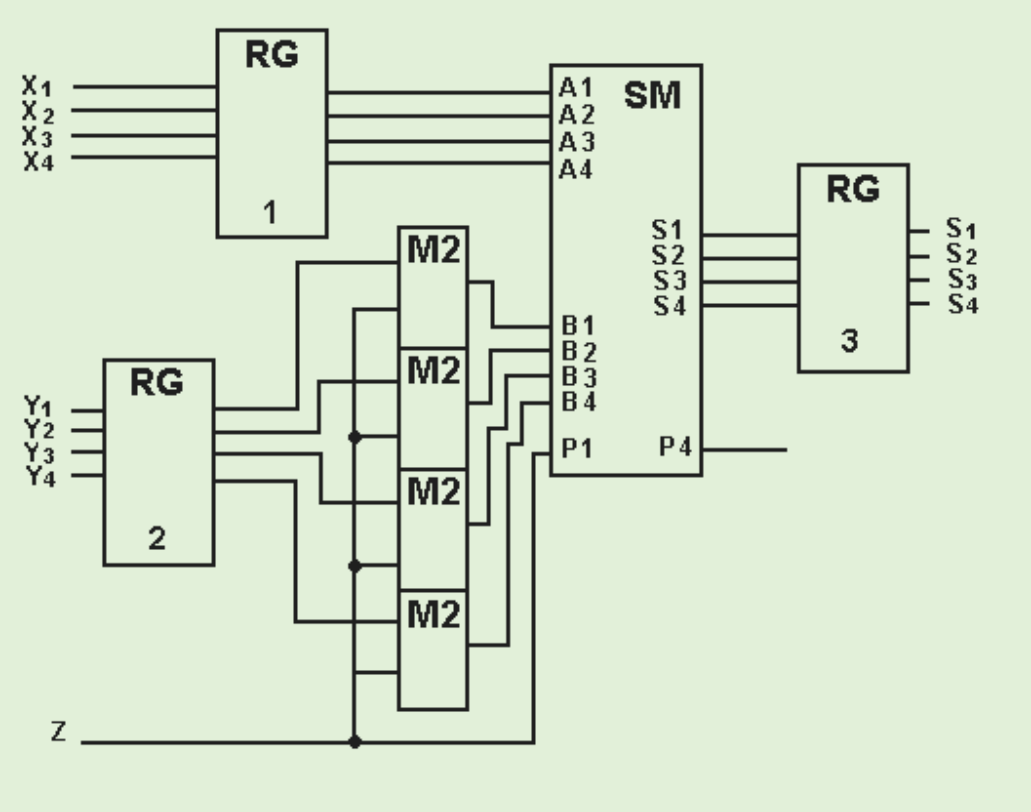


Рис. 4.9 – Схема пристрою додавання і віднімання двійкових чисел

Якщо сигнал керування $Z = P1 = 1$, то відбувається інвертування значень усіх розрядів коду $[Y\partial]$ і по каналу переносу $P1$ у молодший розряд додається одиниця:

$$[S\partial]=[X\partial]+[Y\partial]+1.$$

Ураховуючи те, що $[-Y\partial]=\left[\overline{Y\partial}\right]+1$, остаточно отримуємо

$$[S\partial]=[X\partial]+[-Y\partial],$$

що еквівалентно виконанню операції віднімання.

Розглянемо виконання операції віднімання чисел із знаком на прикладі (-5) - (-3) = -2.

На вхід пристрою операнди надходять у доповняльних кодах: $[X\partial]=1.011$; $[Y\partial]=1.101$. Після інверсії коду ($\left[\overline{Y\partial}\right]=0.010$) і додавання одиниці до молодшого розряду одержимо $[-Y\partial]=\left[\overline{Y\partial}\right]+0.001=0.011$.

ДК різниці $[S\partial]$ знайдемо як суму $[X\partial]$ і $[-Y\partial]$:

$$\begin{array}{r} 1.011 \\ \underline{+0.011} \\ 1.110 \end{array}$$

Отриманий код дійсно є ДК результату $(-2)_{10} = 1.010_2$.

Лабораторна робота «Дослідження суматорів двійкових чисел»

4.5. Помножувачі двійкових чисел

Для побудови помножувачів двійкових чисел використовуються два принципово різні методи. Перший полягає в тому, що операція множення виконується апаратним способом як послідовність операцій додавання часткових добутків, окремі розряди яких є кон'юнкціями відповідних розрядів множників. Розглянемо цей спосіб на прикладі множення дворозрядних двійкових чисел $a_1\ a_0$ і $b_1\ b_0$.

Схема апаратного помножувача, що реалізовує такий спосіб, подана на рис. 4.10. Отриману структуру називають матричним помножувальним блоком.

$$\begin{array}{cccc} & & a_1 & a_0 \\ \times & & b_1 & b_0 \\ \hline & & b_0\,a_1 & b_0\,a_0 \\ + & b_1\,a_1 & b_1\,a_0 & \\ \hline M_3 & M_2 & M_1 & M_0 \end{array}$$

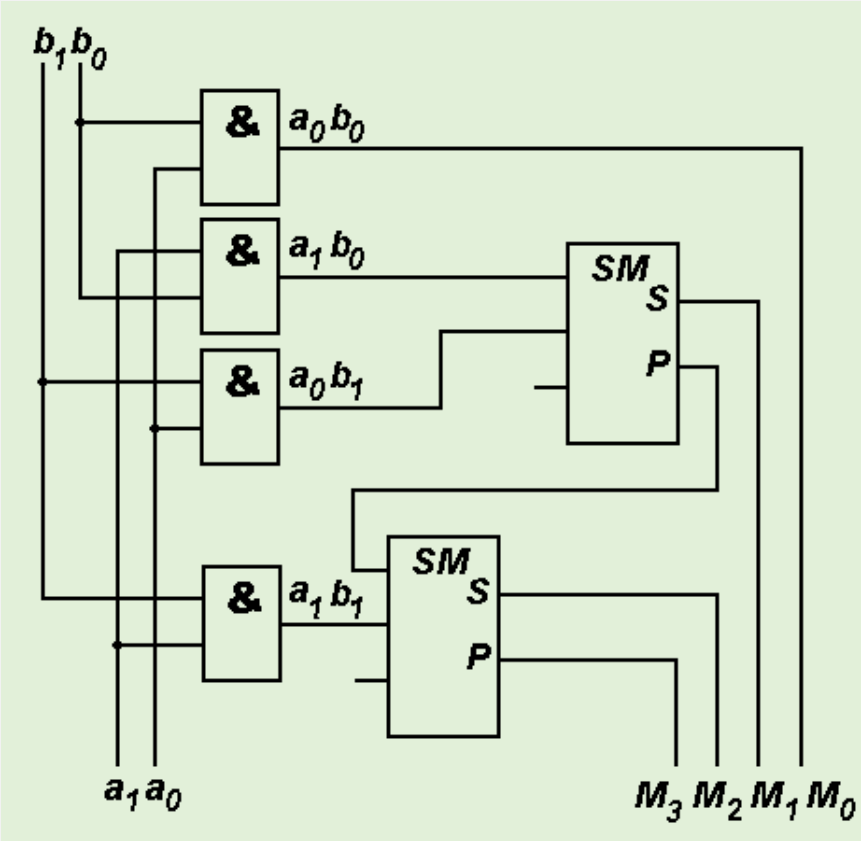


Рис. 4.10 – Матричний помножувальний блок

Аналогічнім чином можна синтезувати схеми помножувачів, що будуть працювати з кодовими словами довільної розрядності. Для підвищення швидкодії обчислювальних систем, які працюють у реальному масштабі часу, наприклад процесорів цифрових сигналів, використовують саме такі спеціалізовані багаторозрядні помножувачі двійкових чисел.

За другим алгоритмом множення виконується як циклічна послідовність елементарних операцій (мікрооперацій) формування часткових добутків (добутків множеного на цифри окремих розрядів множника) і додавання цих добутків із попереднім зсувом ліворуч на відповідну кількість розрядів.

Цей алгоритм достатньо просто реалізовується з використанням одного багаторозрядного суматора й декількох регістрів, що виконують операції зсуву та збереження операндів і результату.

Спрощена структурна схема помножувача двійкових чисел, що реалізовує цей метод, подана на рис. 4.11.

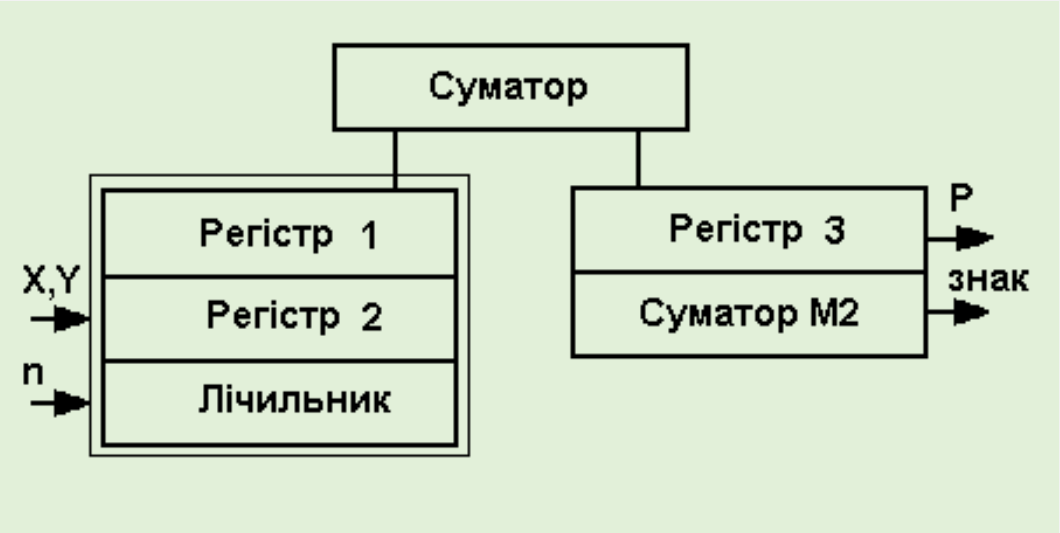


Рис. 4.11 – Помножувач двійкових чисел

Пристрій множення складається з суматора й трьох регістрів: двох для розміщення множників (регістри 1 і 2) та одного для накопичення результату (регістр 3, який звичайно називають акумулятором).

Підрахунок числа *n* виконаних операцій додавання і зсуву часткових добутків виконується за допомогою лічильника зворотного рахування (число *n* заздалегідь записується в лічильник як початковий стан).

Знак добутку визначається окремо за загальновідомим алгоритмом: якщо знаки множників однакові, добуток додатний; якщо знаки різні – від'ємний. Очевидно, що такий алгоритм формування знакового розряду добутку реалізовується за допомогою суматора за модулем два, якщо знак «плюс» кодувати, як звичайно, логічним нулем, а знак «мінус» – логічною одиницею.

Відзначимо деякі суттєві особливості функціонування помножувача. По-перше, в загальному випадку розрядність добутку вдвічі більша розрядності множників. Тому, якщо розрядність усіх регістрів схеми однакова, використовується спосіб множення, коли один з операндів поступово витісняється з регістра 2 розрядами добутку таким чином, що наприкінці виконання операції результат опиняється у парі регістрів 2 та 3.

По-друге, для функціонування цього помножувача необхідні зовнішні сигнали керування, тому він може працювати тільки разом зі спеціальним керуючим пристроєм.

4.6. Поняття про арифметико-логічний пристрій та мікропрограмний автомат

З порівняння пристрою додавання і віднімання з пристроєм множення, що реалізовує алгоритм з використанням одного багаторозрядного паралельного суматора, видно, що більшість елементів цих схем збігається, що говорить про можливість їх об'єднання у вигляді єдиного універсального пристрою, здатного виконувати повний набір основних арифметичних і логічних операцій. Такий пристрій називають *арифметико-логічним пристроєм*. Він виконує додавання чисел апаратним способом, а множення – програмним, як циклічну послідовність розглянутих раніше елементарних дій. Саме такі АЛП використовуються в сучасних універсальних мікропроцесорах.

Очевидно, що для організації роботи обох розглянутих пристроїв необхідне застосування зовнішніх сигналів керування (наприклад, сигналу z у пристрої віднімання і сигналу n встановлення у початковий стан лічильника у пристрої множення). Тому універсальний АЛП повинен працювати разом із пристроєм керування.

Арифметико-логічний пристрій (операційний пристрій) призначений для виконання арифметичних і логічних операцій над числами (словами), що надходять до нього, за сигналами з пристрою керування.

Пристрій керування призначений для організації процесу обчислень. Він координує дії АЛП, генеруючи у визначеній часовій послідовності керуючі сигнали, під дією яких у вузлах АЛП виконуються необхідні операції.

Сукупність АЛП і пристрою керування називають *процесорним пристроєм*, або просто *процесором*. Сучасні процесори, як правило, реалізовуються у вигляді однієї мікросхеми і називаються *мікропроцесорами*.

Структура процесорного пристрою показана на рис. 4.12.

Формування керуючих сигналів $y_1,\ldots, y_N$ для виконання визначених мікрокоманд може залежати від стану вузлів АЛП, обумовленого сигналами $x_1,\ldots, x_S$, які передаються по відповідних колах з виходів АЛП на входи керуючого пристрою. Керуючі сигнали $y_1,\ldots, y_N$ можуть залежати також від зовнішніх сигналів $x_{S+1},\ldots, x_L$. Результати оброблення вхідних даних, що виконано у АЛП, знімають з його виходів $z_1,\ldots, z_M$.

Процес функціонування АЛП поділяється на визначену послідовність елементарних дій в його вузлах. З таких елементарних дій можна виділити:

- ✓ встановлення регістра в деякий стан (наприклад, запис у регістр числа 0, що позначається як $Rg \leftarrow 0$);
- ✓ інвертування вмісту розрядів регістра, що позначається як $Rg \leftarrow \overline{(Rg)}$;

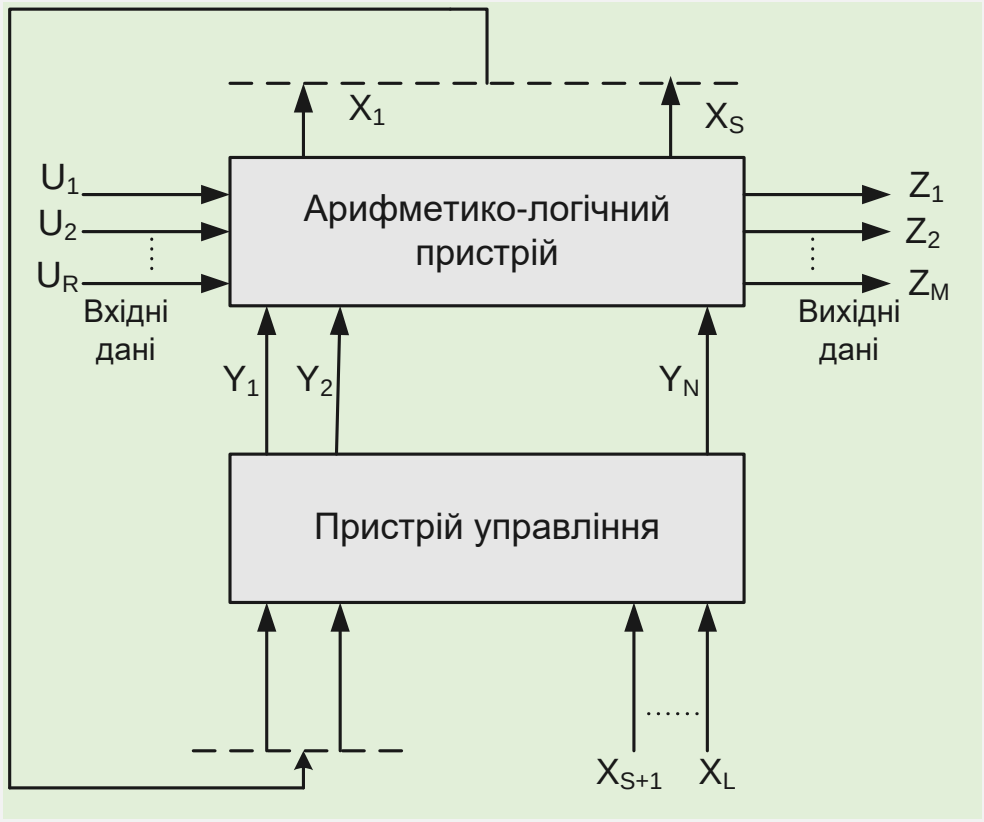


Рис. 4.12 – Схема процесорного пристрою

- ✓ пересилання вмісту одного вузла в інший вузол (наприклад, пересилання вмісту регістра Rg_1 в регістр Rg_2, що позначається як $Rg_2 \leftarrow (Rg_1)$);
- ✓ зсув умісту вузла ліворуч, праворуч (наприклад, зсув на один розряд праворуч умісту регістра, що позначається як $Rg \leftarrow Зсв\ П(Rg)$);
- ✓ рахування, коли число в лічильнику збільшується або зменшується на одиницю ($Cm \leftarrow (Cm) \pm 1$);
- ✓ додавання (наприклад, $Rg_2 \leftarrow (Rg_2) + (Rg_1)$);
- ✓ порівняння вмісту регістра з деяким числом;
- ✓ деякі логічні операції.

Кожна елементарна дія, яка виконується в одному з вузлів АЛП протягом одного тактового періоду, називається *мікрооперацією*. Сукупність мікрооперацій, що виконуються за один такт, називається *мікрокомандою*, а весь набір мікрокоманд, призначений для розв'язання визначеної задачі, – *мікропрограмою*.

Таким чином, якщо в АЛП передбачається можливість виконання N різних мікрооперацій, то з пристрою керування виходять N керуючих кіл, кожне з яких відповідає визначеній мікрооперації. Якщо в АЛП необхідно виконати деяку мікрооперацію, то досить із керуючого пристрою по певному керуючому колу подати в АЛП сигнал (наприклад, рівень логічної 1). Внаслідок того, що пристрій керування визначає мікропрограму, тобто які й у якій часовій послідовності повинні виконуватися мікрооперації, він одержав назву *мікропрограмний автомат*.

Запитання і завдання

1. Назвіть порівняльні переваги і недоліки прямого, оберненого й доповняльного кодів.
2. Яка операція називається операцією доповнення двійкового числа до двох і для чого вона застосовується в комп'ютері?
3. Виконайте додавання чисел із знаком у двійковій системі числення з використанням доповняльного коду: а) 34_{10} та 75_{10}, б) – 34_{10} та 75_{10}, в) 34_{10} та – 75_{10}, г) – 34_{10} та – 75_{10}. Перевірте отримані результати у десятковій системі.
4. Які ознаки переповнення розрядної сітки при додаванні двійкових чисел із знаком у доповняльному коді?
5. Виконайте множення чисел 110110_2 і 1011_2. Перевірте результат у десятковій системі.
6. Який існує зв'язок між розрядністю добутку та співмножників при виконанні множення двох чисел у формі з фіксованою комою? Визначте розрядність результату множення чисел 11011100_2 і 11101111_2.
7. Виконайте ділення числа 286_{10} на 13_{10} у двійковій системі числення з використанням алгоритму без відновлення остачі, результат перевірте у десятковій системі.

8. Поясніть порядок додавання та множення двійкових чисел у формі з плаваючою комою. Як відбувається додавання двох чисел, поданих у формі з плаваючою комою, якщо числа мають різні порядки?

9. Що називають суматором? Наведіть й поясніть класифікацію суматорів.

10. Зобразіть схему та поясніть роботу двійкового однорозрядного напівсуматора.

11. Зобразіть схему й поясніть роботу двійкового однорозрядного суматора на напівсуматорах.

12. Зобразіть схему й поясніть роботу двійкового однорозрядного суматора на мажоритарному елементі.

13. Зобразіть схему й поясніть роботу матричного помножувального блока.

14. Зобразіть схему й поясніть роботу помножувача двійкових чисел, що реалізовує метод додавання часткових добутків зі зсувом.

15. Для чого призначений АЛП?

16. З яких компонентів складається процесорний пристрій? Пояснити призначення його складових частин.

17. Надайте визначення мікрокоманди. Наведіть декілька прикладів мікрокоманд.

ЛІТЕРАТУРА

1. Жабін В.І., Жуков І.А., Клименок І.А., Ткаченко В.В. Прикладна теорія цифрових автоматів: Навч. посібник. – К.: Книжкове вид-во НАУ, 2007. – 364 с.
2. Матвієнко М.П. Комп’ютерна логіка. Навч. посібник. – К.: Вид-во Ліра-К, 2015. – 288 с.
3. Кочубей О.О. Прикладна теорія цифрових автоматів. Логічні основи: Навч. посібник / О.О. Кочубей, О.В. Сопільник. – Д.: РВВ ДНУ; Вид-во ДНУ, 2009, – 264 с.
4. Лахно В.А. Комп'ютерна логіка: Навч. посібник / В.А. Лахно, Б.С. Гусєв, Д.Ю. Касаткін – К.: Вид-во КОМПРІНТ, 2018. – 422 с.
5. Корнійчук В.І., Тарасенко В.П., Тарасенко-Клятченко О.В. Основи комп’ютерної арифметики. – К.: Вид-во «Корнійчук», 2006. – 164 с.
6. Лахно В. А. Лабораторний практикум з Прикладної теорії цифрових автоматів: для студ. спец. 7.091501 „Комп’ютерні системи та мережі” / В.А. Лахно, Г.А. Могильний; Держ. закл. «Луган. нац. ун-т імені Тараса Шевченка». – Луганськ: Вид-во ЛНУ імені Тараса Шевченка, 2010. – 138 с.
7. Булатецький В.В. Алгебра логіки та проєктування основних операційних вузлів: Навч. посібник. / В.В. Булатецький, Л.В. Булатецька, О. М. Собчук; ВНУ ім. Лесі Українки. – Луцьк: ВНУ ім. Лесі Українки, 2021. – 150 с.
8. Тарарака В.Д. Прикладна теорія цифрових автоматів: Навч. посібник. – Житомир: ЖДТУ, 2019. – 183с.

Додаток А. Завдання для практичних занять

Заняття 1. Аналіз логічних схем

Для заданої логічної схеми на 4 входи (рис. А.1) записати ЛФ. Для отриманої функції скласти таблицю істинності.

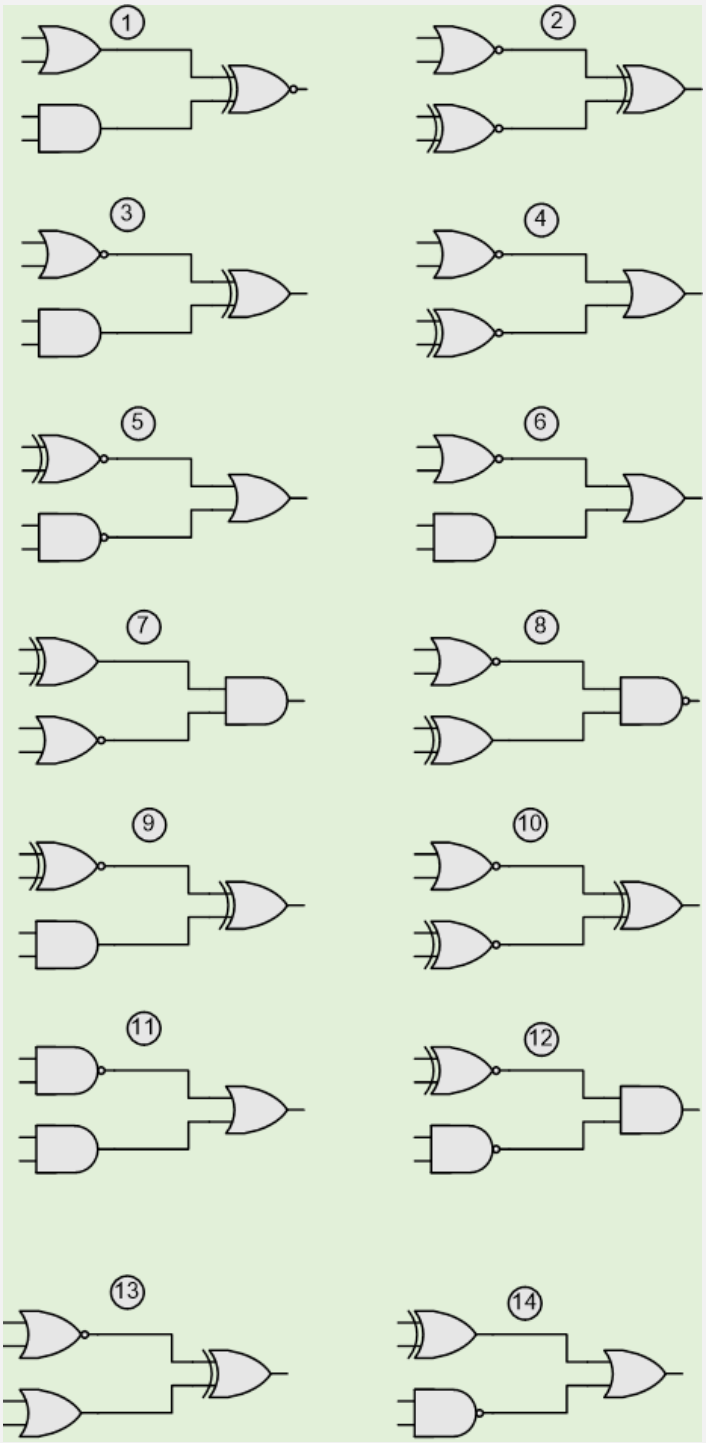


Рис. А.1

Заняття 2. Мінімізація логічних функцій методом карт Карно

Мінімізувати ЛФ, задану таблицею істинності, методом карт Карно.

1. Записати задану таблицею істинності А.1 ЛФ у вигляді ДДНФ

.

Таблиця А.1

АРГУМЕНТИ																
	X_1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	X_2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	X_3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
	X_4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Варіанти (логічні функції)	F_1	1	1	0	0	1	0	1	0	1	1	0	0	1	0	1	0
	F_2	0	1	0	1	1	0	0	0	1	1	0	1	0	1	0	0
	F_3	1	0	1	1	0	1	0	1	0	1	0	0	0	1	0	1
	F_4	1	0	1	0	1	0	0	0	0	0	1	1	0	0	1	1
	F_5	0	0	1	1	0	0	1	1	1	0	1	0	1	0	1	0
	F_6	1	1	1	1	0	0	1	0	1	0	1	0	0	0	0	0
	F_7	1	1	1	0	0	0	1	0	0	0	0	0	1	1	1	1
	F_8	0	0	0	0	1	1	1	0	1	1	1	1	0	0	0	0
	F_9	1	1	0	0	1	1	1	1	1	0	0	0	1	0	0	0
	F_{10}	1	1	0	1	1	0	0	1	1	0	0	1	0	0	0	1
	F_{11}	0	0	1	1	1	0	1	0	0	0	1	1	1	0	1	0
	F_{12}	0	1	0	1	0	0	1	1	0	1	0	1	0	0	1	0
	F_{13}	1	1	0	0	0	1	0	1	1	1	0	0	0	1	0	1
	F_{14}	0	1	0	1	0	1	0	1	0	0	1	1	0	0	1	1
	F_{15}	0	0	1	1	1	1	1	1	0	1	0	1	0	0	0	0
	F_{16}	1	1	1	1	1	0	0	0	0	0	0	0	1	1	0	0
	F_{17}	0	0	1	0	1	0	1	0	0	0	0	0	1	1	1	1
	F_{18}	1	0	1	0	0	0	0	0	1	1	1	1	0	0	1	0
	F_{19}	0	0	1	0	1	1	1	1	0	0	1	0	0	0	1	0
	F_{20}	0	0	0	1	0	1	0	1	0	0	0	1	1	1	0	1
	F_{21}	0	1	0	0	1	0	1	0	0	0	0	0	1	1	1	1
	F_{22}	1	1	0	1	1	1	0	0	1	1	0	1	0	0	0	0
	F_{23}	0	0	1	1	0	1	1	1	0	0	0	1	0	1	0	1
	F_{24}	1	1	0	0	0	0	0	0	1	0	1	1	0	0	1	1
	F_{25}	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0	1
	F_{26}	1	1	1	1	1	0	0	0	1	1	0	0	0	0	0	0
	F_{27}	1	0	0	0	0	0	1	0	1	0	1	0	1	1	1	1
	F_{28}	0	0	1	0	0	0	0	0	1	1	1	1	0	1	0	1
	F_{29}	1	0	0	0	1	1	1	1	1	1	0	1	0	0	0	0
	F_{30}	0	1	0	1	0	0	0	1	0	1	0	1	0	1	0	1

2. Подати таблицю істинності у вигляді карти Карно.
3. Проаналізувати отриману карту та вибрати оптимальне покриття її контурами.
4. Мінімізувати ЛФ за допомогою карти Карно та записати її у вигляді МДНФ.
5. При наявності можливості, виконати остаточну мінімізацію отриманої МДНФ методом безпосередніх тотожних перетворень.

Заняття 3. Мінімізація неповністю визначених логічних функцій методом карт Карно. Синтез нескладних логічних пристроїв

Мінімізувати неповністю визначену ЛФ, задану таблицею істинності, методом карт Карно. Побудувати логічну схему, що реалізує отриману ЛФ.

1. Записати задану таблицею істинності А.2 ЛФ у вигляді ДДНФ, без урахування невизначених значень функції (в таблиці істинності позначені як X).
2. Окремо записати кон’юнкції всіх аргументів або їх інверсій, що відповідають невизначеним значенням ЛФ.
3. Подати таблицю істинності у вигляді карти Карно. Невизначені значення ЛФ на карті позначити як X.
4. Проаналізувати можливі варіанти розташування контурів на карті та замінити невизначені значення функції на 0 чи 1 з метою максимального спрощення мінімізованої ЛФ.
5. Мінімізувати довизначену ЛФ за допомогою карти Карно та записати її у вигляді МДНФ.
6. При наявності можливості, виконати остаточну мінімізацію отриманої МДНФ методом безпосередніх тотожних перетворень.
7. Побудувати логічну схему, що реалізує отриману ЛФ.

Таблиця А.2

АРГУМЕНТИ	X_1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	X_2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	X_3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
	X_4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Варіанти (логічні функції)	F_1	1	X	0	0	1	0	1	0	1	1	0	0	1	0	1	0
	F_2	0	1	0	1	1	X	0	0	1	1	0	1	X	1	0	0
	F_3	1	X	1	X	0	1	0	1	0	1	0	0	0	1	0	1
	F_4	1	0	1	0	1	0	X	0	0	0	1	1	0	0	1	1
	F_5	0	0	1	1	0	0	1	1	1	0	1	0	X	0	1	0
	F_6	1	1	1	1	0	0	1	X	1	0	1	0	0	0	0	0
	F_7	1	1	1	0	0	0	X	0	0	0	0	0	1	1	1	1
	F_8	0	0	0	0	1	1	1	X	1	1	1	1	0	0	0	0
	F_9	1	X	0	0	1	1	1	1	1	X	0	0	1	0	0	0
	F_{10}	1	X	0	1	1	0	0	1	1	X	0	1	0	0	0	1
	F_{11}	0	0	1	X	1	0	1	0	0	0	1	1	1	0	1	0
	F_{12}	0	1	0	1	0	0	1	1	0	1	0	1	0	0	1	X
	F_{13}	1	1	0	0	0	1	0	1	1	X	0	0	0	1	0	1
	F_{14}	0	1	0	X	0	1	0	1	0	0	1	1	0	0	1	1
	F_{15}	0	X	1	1	1	1	1	1	0	1	0	X	0	0	0	0
	F_{16}	1	1	1	1	1	X	0	0	0	0	0	0	1	1	0	0
	F_{17}	X	0	1	0	1	0	1	0	0	0	0	0	1	1	1	1
	F_{18}	1	0	1	0	0	0	0	0	1	1	1	1	0	0	1	X
	F_{19}	0	0	1	X	1	1	1	1	0	0	1	0	0	0	1	0
	F_{20}	0	0	0	1	0	1	0	1	0	0	0	1	1	1	X	1
	F_{21}	0	1	0	0	1	X	1	0	0	X	0	0	1	1	1	1
	F_{22}	1	1	0	1	1	X	0	0	1	1	X	1	0	0	0	0
	F_{23}	0	0	1	1	0	1	X	1	0	0	X	1	0	1	0	1
	F_{24}	1	1	0	0	0	0	0	0	1	X	1	1	X	0	1	1
	F_{25}	0	0	1	1	0	0	1	1	0	X	1	1	0	0	X	1
	F_{26}	1	1	1	1	1	X	0	0	1	1	0	0	X	0	0	0
	F_{27}	X	0	0	0	0	0	1	X	1	0	1	0	1	1	1	1
	F_{28}	0	0	1	X	0	0	0	0	1	1	1	1	X	1	0	1
	F_{29}	X	0	0	0	1	1	1	1	1	1	X	1	0	0	0	0
	F_{30}	0	1	0	1	0	X	0	1	0	1	X	1	0	1	0	1

Заняття 4. Перетворення чисел в різні позиційні системи числення

Виконати перетворення чисел, заданих таблицею А.3 із однієї системи числення в іншу.

Таблиця А.3

№ варіанта	А	Б	В	Г
1	276	1000011	717	4A
2	493	1000100	726	4B
3	480	1000101	735	4C
4	404	1000110	704	4D
5	354	1000111	673	4E
6	297	1001000	652	4F
7	384	1001001	671	74
8	333	1001010	660	84
9	321	1001011	657	94
10	434	1001100	646	CC
11	447	1001101	635	3E
12	360	1001110	624	3F
13	473	1001111	613	3A
14	486	1010000	602	3B
15	306	1010001	561	3C
16	291	1010011	577	83
17	345	1010100	566	93
18	458	1010101	555	AA
19	261	1010110	544	BB
20	374	1010111	533	2C
21	487	1011000	522	2D
22	300	1011001	511	2E
23	313	1011010	500	2F
24	426	1011011	457	2A
25	270	1011100	446	2B
26	342	1011101	435	92
27	455	1011110	424	DD
28	268	1011111	413	EE
29	381	1100000	402	FF
30	328	11001110	388	AB

1. Перетворити десяткове число із стовпчика А на двійкове число.
2. Перетворити десяткове число із стовпчика А на вісімкове число.

3. Перетворити десяткове число із стовпчика А на шістнадцяткове число.
4. Перетворити десяткове число із стовпчика А на двійково-десятковий код.
5. Перетворити двійкове число із стовпчика Б на десяткове число.
6. Перетворити двійкове число із стовпчика Б на вісімкове число.
7. Перетворити двійкове число із стовпчика Б на шістнадцяткове число.
8. Перетворити вісімкове число із стовпчика В на двійкове число.
9. Перетворити шістнадцяткове число із стовпчика Г на двійкове число.

Заняття 5. Завадостійке кодування двійкових чисел

Закодувати заданий байт (таблиця А.4):

1. *Кодом Грея.* Правило перетворення: старші розряди вхідного та вихідного кодів збігаються, а будь-який наступний розряд Y_k коду Грея дорівнює сумі по модулю два відповідного X_k та старшого X_{k+1} розрядів натурального коду.

2. *Кодом з контролем парності.* Правило перетворення: якщо у вихідному байті парна кількість одиниць, в контрольному розряді (наприклад, старшому), записуємо 0, якщо непарна – 1.

3. *Молодшу половину заданого байта закодувати кодом Хеммінга.*

Контрольні біти k_i (для розглянутого прикладу їх 3) розташовують у послідовності інформаційних бітів u_j, як показано у таблиці:

Позиція	**1**	**2**	**3**	**4**	**5**	**6**	**7**
	001	010	011	100	101	110	111
Біт	k_1	k_2	u_1	k_3	u_2	u_3	u_4

Значення перевірних бітів k_i обчислюється додаванням за модулем 2 значень бітів, у двійковому виразі номерів яких наявна одиниця в i-му розряді. Відповідно для обчислення значення k_1 потрібно додати за модулем 2 значення бітів із непарними номерами позицій

$$k_1 = u_1 \oplus u_2 \oplus u_4 .$$

Для визначення k_2 треба додати за модулем 2 біти, у двійковому виразі номерів яких наявна одиниця у другому розряді, тобто

$$k_2 = u_1 \oplus u_3 \oplus u_4.$$

Контрольний біт k_3 визначається додаванням за модулем 2 бітів, у двійковому виразі номерів котрих наявна одиниця у третьому розряді,

$$k_3 = u_2 \oplus u_3 \oplus u_4$$

.

Таблиця А.4

Варіант	Байт	Варіант	Байт
1	10000111	16	10100111
2	10101011	17	10111011
3	10001010	18	10111010
4	10100110	19	10100100
5	10011110	20	11011100
6	10010001	21	10010101
7	10010011	22	11010010
8	10110101	23	10111101
9	10010111	24	10110110
10	10011000	25	10111100
11	01001101	26	01001010
12	01001110	27	01101010
13	11001111	28	01001111
14	10100000	29	10100011
15	11010001	30	11011101

Приклад виконання

Варіант	Байт
1	10101100

1. Перетворення в код Грея:

	X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0
Вхідний код	1	0	1	0	1	1	0	0
Вихідний код	$Y_7=X_7$	$Y_6=X_7\oplus X$	$Y_5=X_6\oplus X$	$Y_4=X_5\oplus X$	$Y_3=X_4\oplus X$	$Y_2=X_3\oplus X$	$Y_1=X_2\oplus X$	$Y_0=X_1\oplus X$
Результат	1	1	1	1	1	0	1	0

2. Перетворення в код з контролем парності. Оскільки в байті 10101100 парна кількість одиниць (4), маємо: **0**10101100.

3. Перетворення в код Хеммінга. Молодша половина заданого байта – 1100. Обчислимо контрольні біти
:

$$k_1 = u_1 \oplus u_2 \oplus u_4 = 1\oplus1\oplus0=0,$$
$$k_2 = u_1 \oplus u_3 \oplus u_4 = 1\oplus0\oplus0=1,$$
$$k_3 = u_2 \oplus u_3 \oplus u_4 = 1\oplus0\oplus0=1.$$

Результат записуємо в таблицю.

Позиція	**1**	**2**	**3**	**4**	**5**	**6**	**7**
	001	010	011	100	101	110	111
Біт	k_1	k_2	$u_1=1$	k_3	$u_2=1$	$u_3=0$	$u_4=0$
Результат	0	1	1	1	1	0	0

Заняття 6. Синтез цифрових автоматів канонічним методом

Синтезувати логічну схему простого ЦА на заданому типі тригерів (Т або JK). Алгоритм функціонування ЦА (послідовність станів) задана таблицею варіантів А.5. Оскільки автомати дуже прості (мають всього по два тригери та один циклічний режим функціонування, побудова графа функціонування недоцільна).

Для виконання завдання необхідно:

1. Скласти таблицю переходів тригерів ЦА. Результат виконання цього етапу – таблиця переходів тригерів, що має вигляд:

✓ для JK-тригерів

Такт	Початковий стан		Наступний стан		Сигнали на входах тригерів			
	Q_1^t	Q_0^t	Q_1^{t+1}	Q_0^{t+1}	J_1	K_1	J_0	K_0
0								
…								
4								

✓ для T-тригерів

Такт	Початковий стан		Наступний стан		Сигнали на входах тригерів	
	Q_1^t	Q_0^t	Q_1^{t+1}	Q_0^{t+1}	T_1	T_0
0						
…						
4						

2. На підставі таблиці переходів записати функції збудження тригерів (вхідні ЛФ) у вигляді ДДНФ.

3. Якщо є можливість, мінімізувати функції збудження тригерів. Оскільки функції збудження тригерів будуть дуже простими, застосування карт Карно тут недоцільно, достатньо мінімізації методом безпосередніх тотожних перетворень. Результат виконання цього етапу – мінімізовані ЛФ входів тригерів у довільному логічному базисі.

4. Побудувати логічну схему синтезованого ЦА.

Таблиця А.5

№ варіанта	Стани автомата Q_1Q_0					Тип тригерів
	Початковий стан	Такти				
		1	2	3	4	
1	00	10	01	11	00	T
2	00	11	10	01	00	T
3	00	01	11	10	00	T
4	00	10	11	01	00	T
5	00	11	01	10	00	T
6	11	10	00	01	11	T
7	11	01	10	00	11	T
8	11	01	00	10	11	T
9	11	10	01	00	11	T
10	11	10	00	01	11	T
11	00	10	01	11	00	JK
12	00	11	10	01	00	JK
13	00	01	11	10	00	JK
14	00	10	11	01	00	JK
15	00	11	01	10	00	JK
16	11	10	00	01	11	JK
17	11	01	10	00	11	JK
18	11	01	00	10	11	JK
19	11	10	01	00	11	JK
20	11	10	00	01	11	JK

Заняття 7. Виконання арифметичних операцій в комп'ютері

Виконати заданий набір арифметичних операцій над двійковими числами, поданими в таблиці А.6.

1. Десяткові числа із стовпчиків А і В перевести у двійкову систему числення та додати (з урахуванням знака), вважаючи обидва числа додатними, використовуючи для цього процедуру додавання чисел в прямих кодах.

Таблиця А.6

Варіант	A	B	C	D	E	F
1	91	75	15	17	25	5
2	86	65	13	13	28	4
3	77	52	13	15	30	5
4	94	48	12	18	30	6
5	65	33	11	18	36	6
6	96	50	10	19	40	4
7	87	54	19	11	44	4
8	85	64	15	12	45	9
9	44	63	17	10	45	5
10	66	37	14	12	48	8
11	68	35	17	15	20	4
12	78	42	14	18	24	4
13	78	75	13	19	50	5
14	76	60	12	13	50	10
15	95	77	11	17	56	8
16	69	45	19	17	56	4
17	73	51	17	13	63	7
18	72	56	18	10	63	9
19	92	61	13	12	72	8
20	88	70	14	17	35	5
21	76	44	16	19	35	7
22	71	35	15	10	49	7
23	70	42	14	12	54	6
24	65	39	13	18	54	9
25	89	84	12	14	80	8
26	90	78	11	15	90	9
27	62	90	10	10	56	8
28	63	43	17	12	56	4
29	64	37	15	11	63	7
30	89	84	15	10	72	8

2. Десяткові числа із стовпчиків A і B перевести в двійкову систему числення. З числа A відняти число B, використовуючи для цього процедуру додавання числа із стовпчика A в прямому коді з числом із стовпчика B в доповняльному коді.

3. Десяткові числа із стовпчиків A і B перевести в двійкову систему числення. Додати числа у доповняльних кодах, вважаючи обидва числа від’ємними.

4. Десяткові числа із стовпчиків C та D перевести у двійкову систему числення та перемножити.

5. Десяткові числа із стовпчиків E і F перевести у двійкову систему числення та поділити перше число на друге, використовуючи алгоритм без відновлення остачі.

6. Для кожного пункту завдання правильність отриманих результатів перевірити шляхом їх переведення у десяткове подання.

Заняття 8. Виконання арифметичних операцій над числами з рухомою комою

Виконати додавання двох чисел із стовпчиків X, Y таблиці А.7 як чисел з рухомою комою (тут X – ціле число, Y – дробове). Розрядність мантиси – 1 байт, порядок незміщений.

Таблиця А.7

Варіант	X	Y	Варіант	X	Y
1	52	46,875	16	56	32,875
2	63	18,75	17	57	23,75
3	48	43,625	18	58	19,625
4	55	11,375	19	59	29,375
5	47	17,5	20	60	42,5
6	49	39,25	21	61	37,25
7	53	21,125	22	62	28,125
8	44	35,875	23	64	21,875
9	45	37,75	24	65	27,75
10	41	26,625	25	66	27,625
11	43	33,375	26	67	33,375
12	42	38,5	27	45	38,5
13	46	46,25	28	52	26,25
14	31	49,125	29	48	29,125
15	54	45,875	30	64	43,5

Приклад виконання

Варіант	X	Y
31	68	29,125

1. Переводимо числа X, Y в двійкову систему числення: $X=68_{10}=1000100_2$; $Y=29,125_{10}=11101,001$;

2. Оскільки розрядність мантиси – 1 байт:

$M_x=0,10001000$; $P_x=0111_2=7_{10}$; $M_y=0,11101001$; $P_y=0101_2=5_{10}$.

3. Оскільки X>Y, порядок числа Y приводиться до порядку числа X (зсув коми мантиси M_y на $P_x-P_y=2$ розряди ліворуч): $M_y=0,00111010$; $P_y=0111_2=7_{10}$.

4. Додавання мантис: $M_s=M_x+M_y=$

$$\begin{array}{r} 0,10001000 \\ \underline{0,00111010} \\ 0.11000010 \end{array}$$

Корекція результату в даному випадку не потрібна.

5. Якщо необхідно, виконується операція нормалізації результату – приведення мантиси до стандартного вигляду змінюванням порядку (в даному прикладі результат вже є нормалізованим).

Результат: $M_s=0,11000010$; $P_s=0111_2=7_{10}$; S=1100001,0

$S=X+Y=97,125_{10}=1100001,001_2$.

Додаток Б. Завдання для лабораторних занять

Заняття 1. Аналіз функціонування логічних пристроїв

Мета:

- ✓ вивчення прийомів моделювання, аналізу та синтезу логічних схем за допомогою програми Multisim;
- ✓ дослідження складених ЛЕ.

Теоретична частина.
Робота з логічним перетворювачем

Необхідні теоретичні відомості про ЛФ та ЛЕ викладені в п.1.2 посібника «Логічні функції і логічні елементи. Закони і правила алгебри логіки».

Логічний перетворювач (Logic Converter) виконує перетворення подання схеми й цифрових сигналів та використовується для аналізу цифрових схем. Реального аналога цей прилад не має.

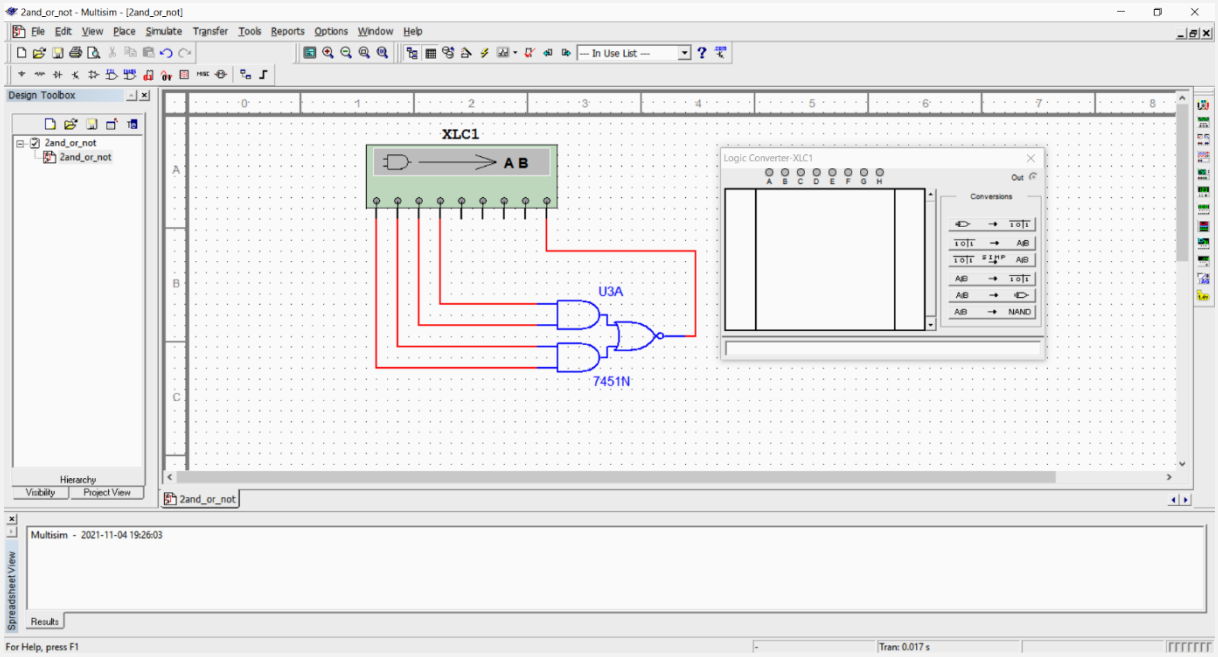
Для того щоб додати логічний перетворювач у робоче поле програми, необхідно натиснути на його піктограму на панелі «Прилади» і розмістити його за допомогою миші в необхідному місці на схемі. Для того щоб відобразити панель приладу, необхідно двічі клацнути лівою кнопкою миші на його зображенні на схемі. Рис. Б.1 демонструє панель перетворювача, його зображення на схемі, а також приклад його підключення до схеми.

Прилад має вісім входів та один вихід. Принцип з'єднання логічного перетворювача з елементами схеми такий самий, як і для інших компонентів схеми.

Розглянемо панель логічного перетворювача більш детально. У лівій частині панелі знаходиться вікно таблиці істинності досліджуваної схеми. Стовпці таблиці відповідають входам логічного перетворювача (A, B, C, D, E, F, G, H). Над кожним стовбцем таблиці розташований кружок, який відображається білим кольором у разі, коли вхід перетворювача використовується і сірим – коли вхід вільний.

Останній стовпчик таблиці істинності відповідає виходу логічного перетворювача. Значення даного стовбця можна змінювати для кожної вхідної умови, для чого необхідно клацнути по ньому лівою кнопкою миші,

перемикаючись між трьома можливими установками: логічний 0, логічна 1, значення X. досліджуваної схеми.



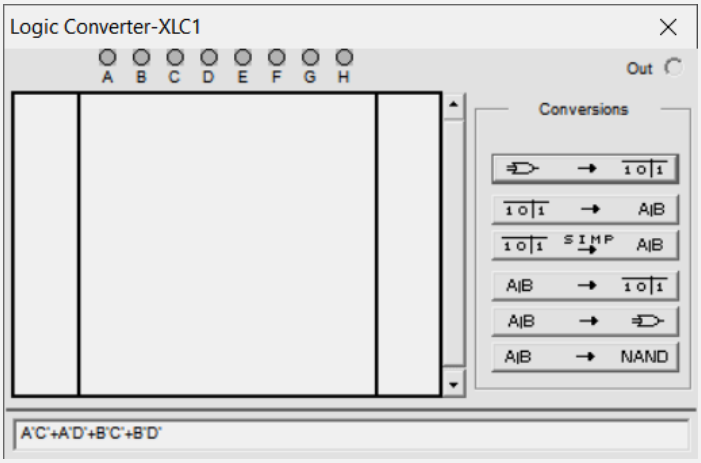


Рис. Б.1

У нижній частині панелі знаходиться рядок, в якому відображається ЛФ, що відповідає досліджуваній схемі. ЛФ у дане полі можна ввести і вручну в тому випадку, коли є необхідність побудувати таблицю істинності згідно із заданою функцією або синтезувати схему, що реалізує введену ЛФ. У правій частині панелі пристрою розташовано шість кнопок вибору перетворення:

- ✓ побудова таблиці істинності досліджуваної схеми;
- ✓ отримання ЛФ відповідно до таблиці істинності;
- ✓ отримання ЛФ у спрощеній формі відповідно до таблиці істинності;

- ✓ побудова таблиці істинності згідно з ЛФ;
- ✓ побудова логічної схеми згідно з ЛФ;
- ✓ побудова логічної схеми у базисі І-НІ згідно з ЛФ.

Побудова таблиці істинності згідно з ЛФ здійснюється шляхом введення логічного виразу в рядок функцій та наступного натискання на кнопку «Побудова таблиці істинності згідно з ЛФ». Введення ЛФ здійснюється відповідно до таких правил:

- ✓ у ЛФ для позначення аргументів можуть використовуватися тільки значення букв, що збігаються з назвами входів логічного перетворювача (тобто: A, B, C, D, E, F, G, H);
- ✓ логічна операція додавання позначається знаком «+»;
- ✓ логічна операція множення не позначається;
- ✓ при складанні логічного виразу при необхідності можуть використовуватися дужки «( )».
- ✓ інверсія позначається знаком ' після відповідного аргументу або фрагменту ЛФ, охопленому дужками (наприклад, A'C'+A'D'+B'C'+B'D).

Побудова схеми, яка реалізує ЛФ в булевому базисі (на елементах І, АБО, НІ), здійснюється шляхом введення у рядку функцій логічного виразу та наступного натискання на кнопку «Побудова схеми на логічних вентилях згідно з ЛФ». В результаті логічним перетворювачем буде виведено на робоче поле програми схема, яка реалізує ЛФ, що описує введений у рядку функцій вираз. Приклад цього перетворення представлений на рис. Б.2.

Побудова схеми в базисі І-НІ, що реалізує заданий логічний вираз, здійснюється шляхом введення в рядку функцій цього виразу та наступного натискання на кнопку «Побудова схеми на логічних вентилях у базисі І-НІ згідно з ЛФ». Внаслідок чого логічним перетворювачем буде виведено на робоче поле програми схема, що відповідає заданій ЛФ, реалізована винятково на вентилях І-НІ. Приклад цього перетворення представлений на рис. Б.3.

Порядок виконання роботи

1. Дослідження ЛЕ 2І-АБО-НІ

- ✓ Скласти в Multisim схему для дослідження складеного ЛЕ 2І-АБО-НІ (рис. Б.1) на основі мікросхеми 7451 (Plase TTL – 7451N). УГП та логічна структура мікросхеми показані на рис. Б.4. Оскільки мікросхема складається з двох однакових частин, достатньо вибрати лише одну з них (наприклад, А).

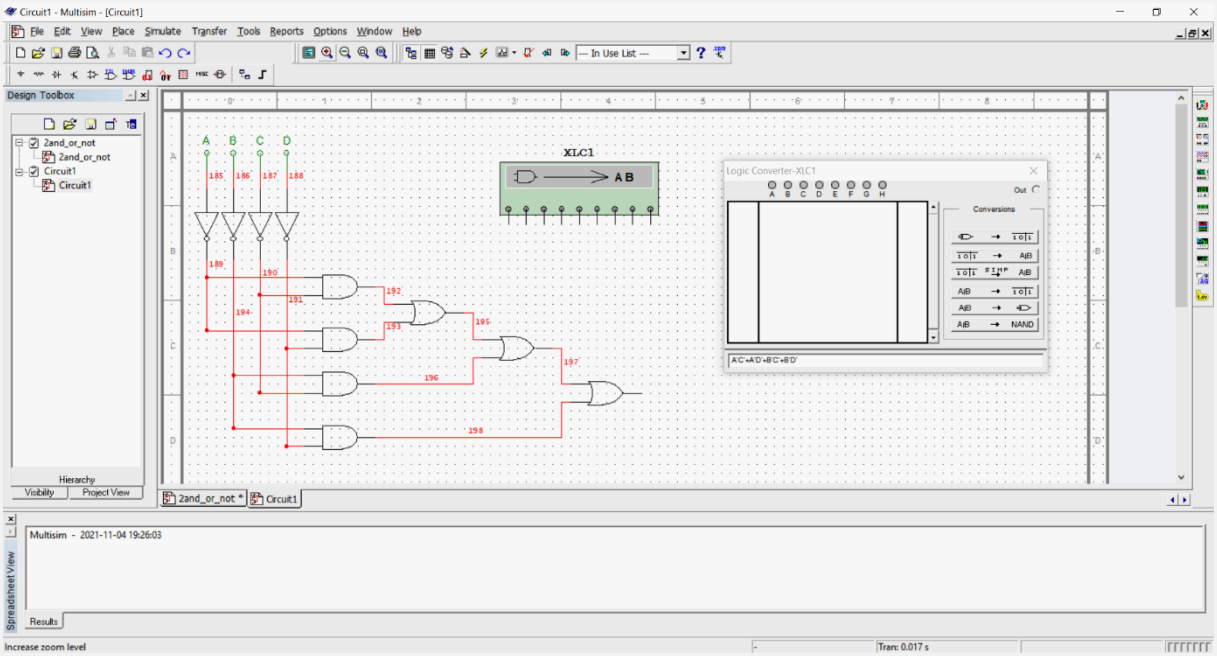


Рис. Б.2

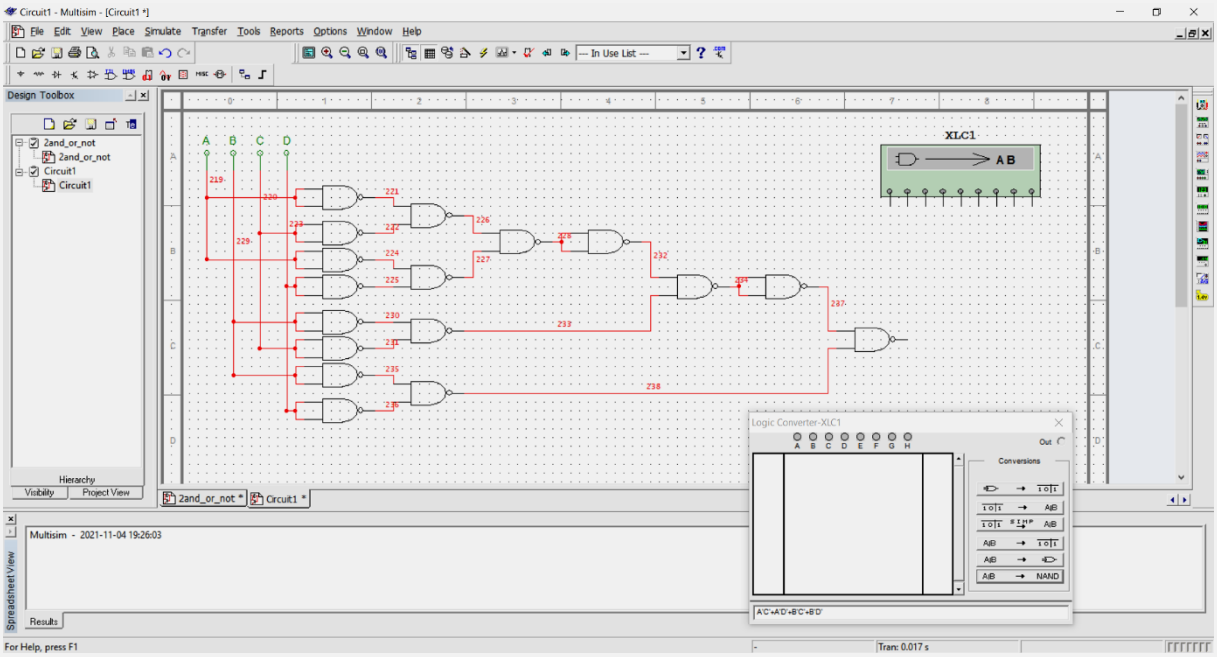


Рис. Б.3

✓ Отримати таблицю істинності складеного ЛЕ, схему в булевому базисі (приклад на рис. Б.2), схему в базисі І-НІ (приклад на рис. Б.3) та відповідні ЛФ у вказаних логічних базисах. Схеми доцільно створювати в нових файлах. Необхідна ЛФ може бути скопійована з відповідного рядка логічного конвертора за допомогою буферу обміну.

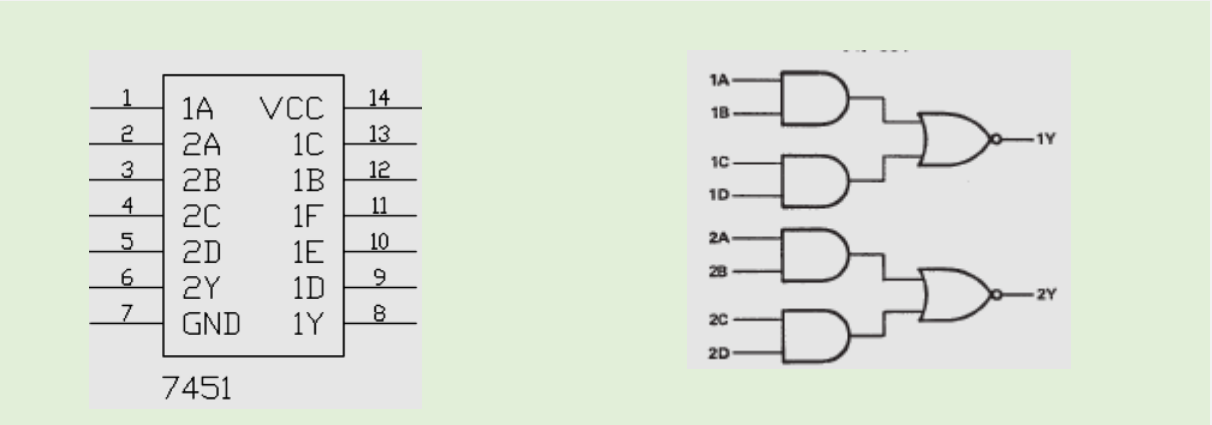


Рис. Б.4

Зверніть увагу на те, що обидві схеми є складнішими, чим вихідна. Пояснить причину цього у висновках.

В звіті мають бути: схема моделювання (приклад на рис. Б.1), таблиця істинності (у вигляді вікна логічного конвертора), еквівалентні схеми в базисах булевому та І-НІ (приклади на рис Б.2, Б.3), ЛФ у вказаних базисах.

2. Дослідження ЛЕ «Виключне АБО» на 4 входи

Скласти в Multisim схему для дослідження складеного ЛЕ «Виключне АБО» на 4 входи (рис. Б.5) на основі мікросхеми 7486 (Plase TTL – 7486N). УГП та логічна структура мікросхеми показані на рис. Б.6.

Далі, аналогічно першої частини завдання, отримати таблицю істинності складеного ЛЕ, схему в булевому базисі, схему в базисі І-НІ та відповідні ЛФ у вказаних логічних базисах.

Використовуючи метод безпосередніх тотожних перетворень, доведіть, що отримана ЛФ дійсно еквівалентна функції: $A \oplus B \oplus C \oplus D$.

Оформити та подати на перевірку викладачеві звіт, що містить:

- ✓ назву та мету заняття;
- ✓ для обох частин завдання: отримані схеми, таблиці істинності та ЛФ;
- ✓ висновок, у якому бажано сформулювати для кожної дослідженої схеми складеного ЛЕ, у вигляді словесного опису, правило (алгоритм) її функціонування (наприклад: значення ЛФ дорівнює одиниці, якщо значення аргументів…).

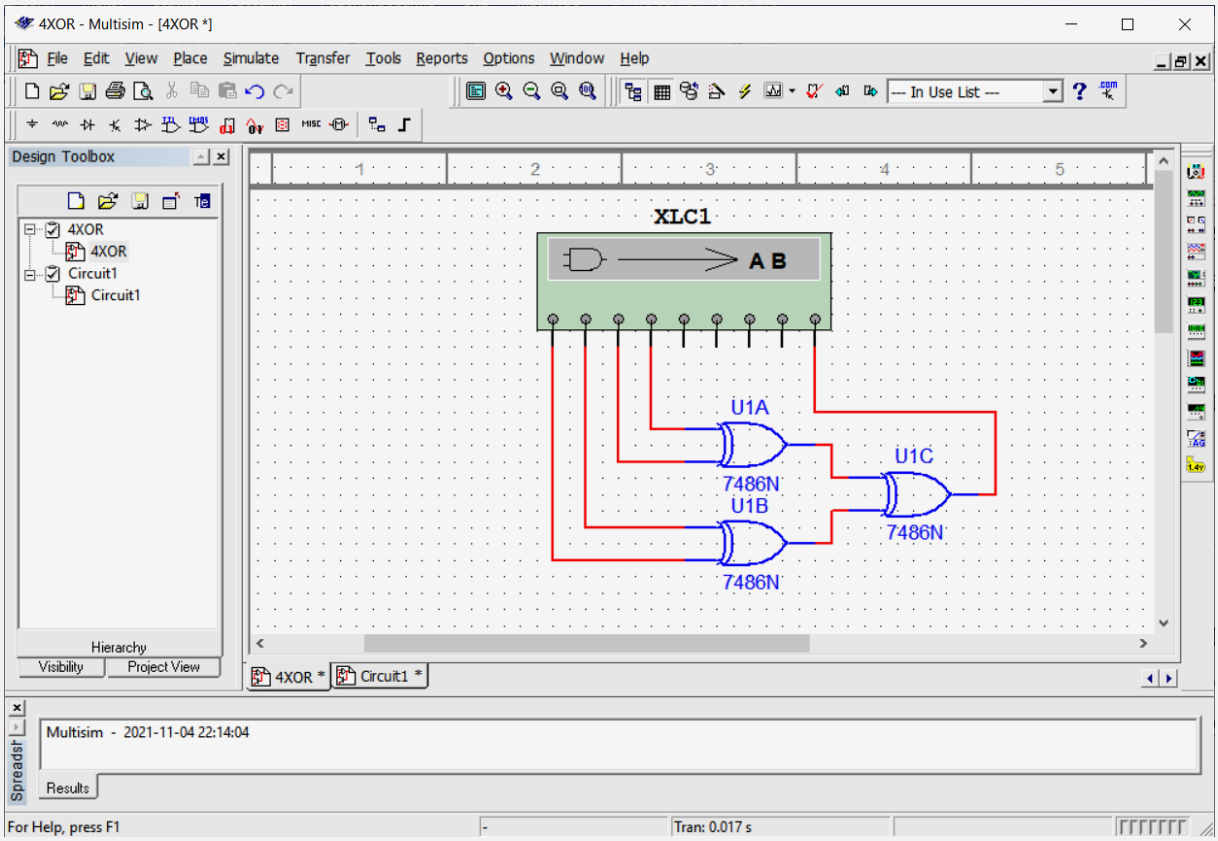

4XOR - Multisim - [4XOR *]
File Edit View Place Simulate Transfer Tools Reports Options Window Help
Design Toolbox
4XOR
4XOR
Circuit1
Circuit1
Hierarchy
Visibility
Project View
XLC1
A B
U1A
7486N
U1B
7486N
U1C
7486N
4XOR *
Circuit1 *
Multisim - 2021-11-04 22:14:04
Results
Spreadsh
For Help, press F1
Tran: 0.017 s

Рис. Б.5

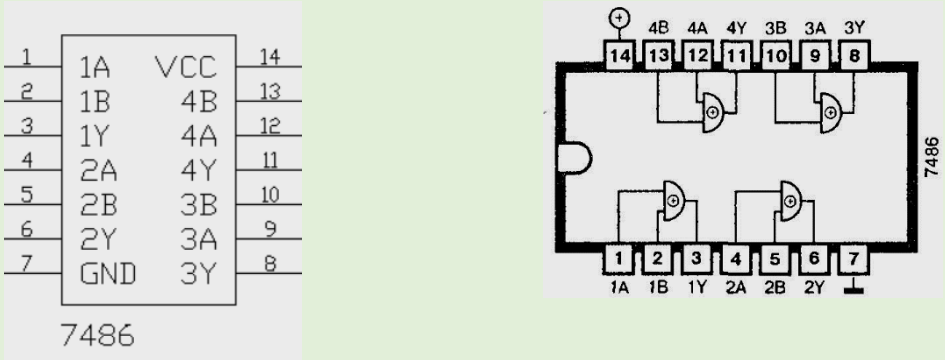

1A
1B
1Y
2A
2B
2Y
GND
VCC
4B
4A
4Y
3B
3A
3Y
7486
4B 4A 4Y 3B 3A 3Y
14 13 12 11 10 9 8
1 2 3 4 5 6 7
1A 1B 1Y 2A 2B 2Y
7486

Рис Б.6.

Заняття 2. Дослідження комбінаційних цифрових вузлів

Мета:

✓ вивчення прийомів моделювання та дослідження нескладних комбінаційних вузлів схем за допомогою програми Multisim 8;

✓ дослідження функціонування типових комбінаційних цифрових вузлів (шифраторів та дешифраторів).

Теоретична частина

Необхідні теоретичні відомості про комбінаційні цифрові вузли викладені в п. 1.4.1. посібника «Шифратори та дешифратори».

Порядок виконання роботи

1. Дослідження шифратора

✓ Скласти (або отримати) схему моделювання шифратора на основі мікросхеми 74HC147N (рис. Б.7);

✓ скласти таблицю істинності шифратора (таблиця Б.1). Вхідні сигнали по черзі подавати на один з дев'яти входів дешифратора за допомогою перемикачів J1 – J9, що перемикаються відповідними цифровими клавішами. Стани входів та виходів відображаються індикаторами X1 – X9 (входи) та X10 – X13. При введенні одного з вхідних сигналів на виходах схеми отримаємо певну комбінацію з вихідних сигналів;

✓ проаналізувати отриману таблицю і зробити висновки щодо призначення та особливостей функціонування шифратора.

2. Дослідження дешифратора

✓ Скласти або отримати схему моделювання дешифратора (рис. Б.8);

✓ скласти таблицю істинності шифратора (таблиця Б.2). На входи дешифратора подаються вхідні кодові слова за допомогою перемикачів X3 – X1, що перемикаються відповідними цифровими клавішами. Стани виходів відображаються відповідними сегментами індикатора (бараграфа), який в даній схемі виконує функцію набору індикаторних світлодіодів. При

введенні будь-якої вхідної кодової комбінації отримаємо сигнал активного рівня на одному з виходів дешифратора;

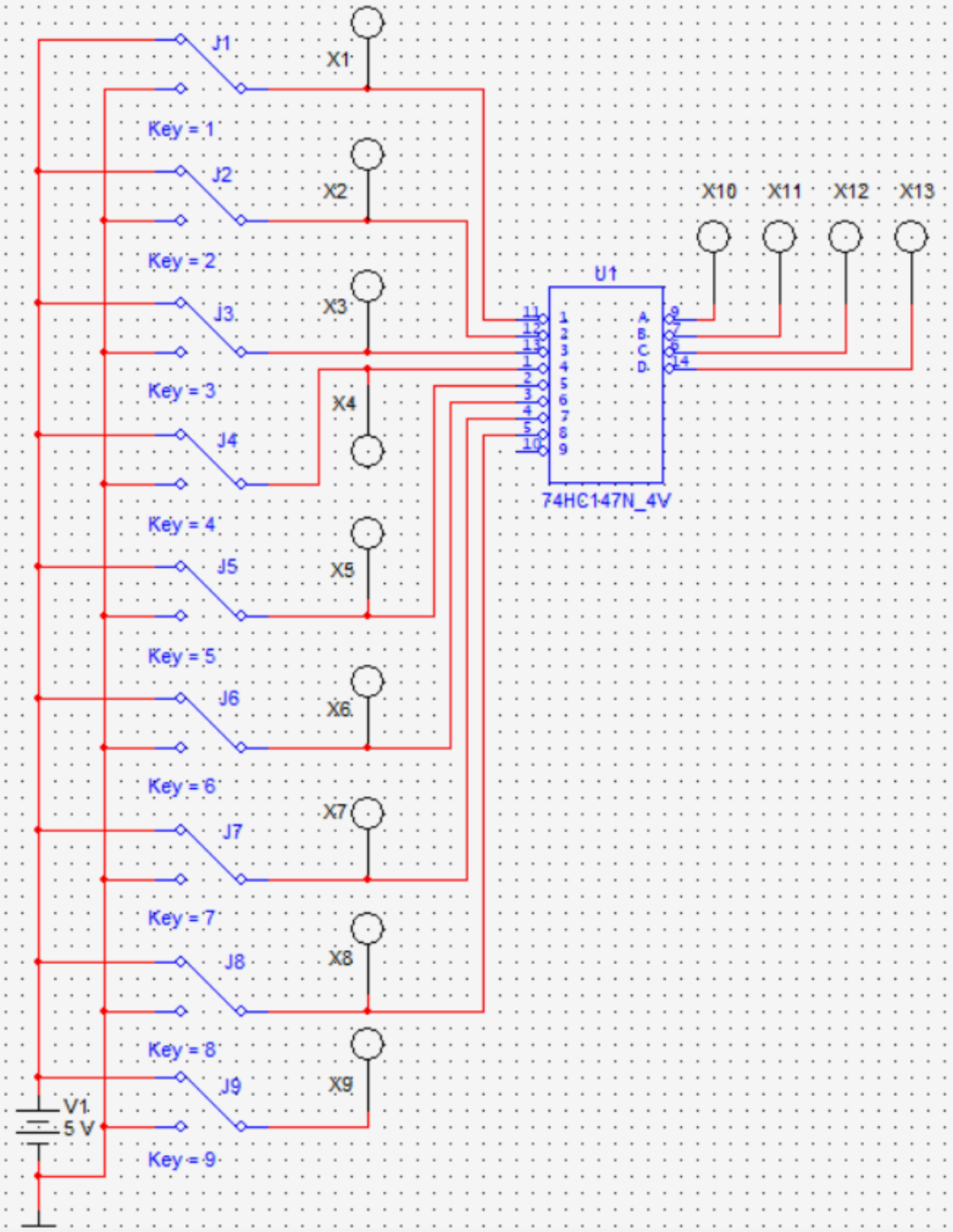


Рис. Б.7 – Схема моделювання шифратора

✓ проаналізувати отриману таблицю і зробити висновки щодо призначення та особливостей функціонування дешифратора.

Таблиця Б.1

Вхід	Виходи (двійковий код 8421)			
	8 (X10)	4 (X11)	2 (X12)	1 (X13)
1				
2				
3				
4				
5				
6				
7				
8				
9				

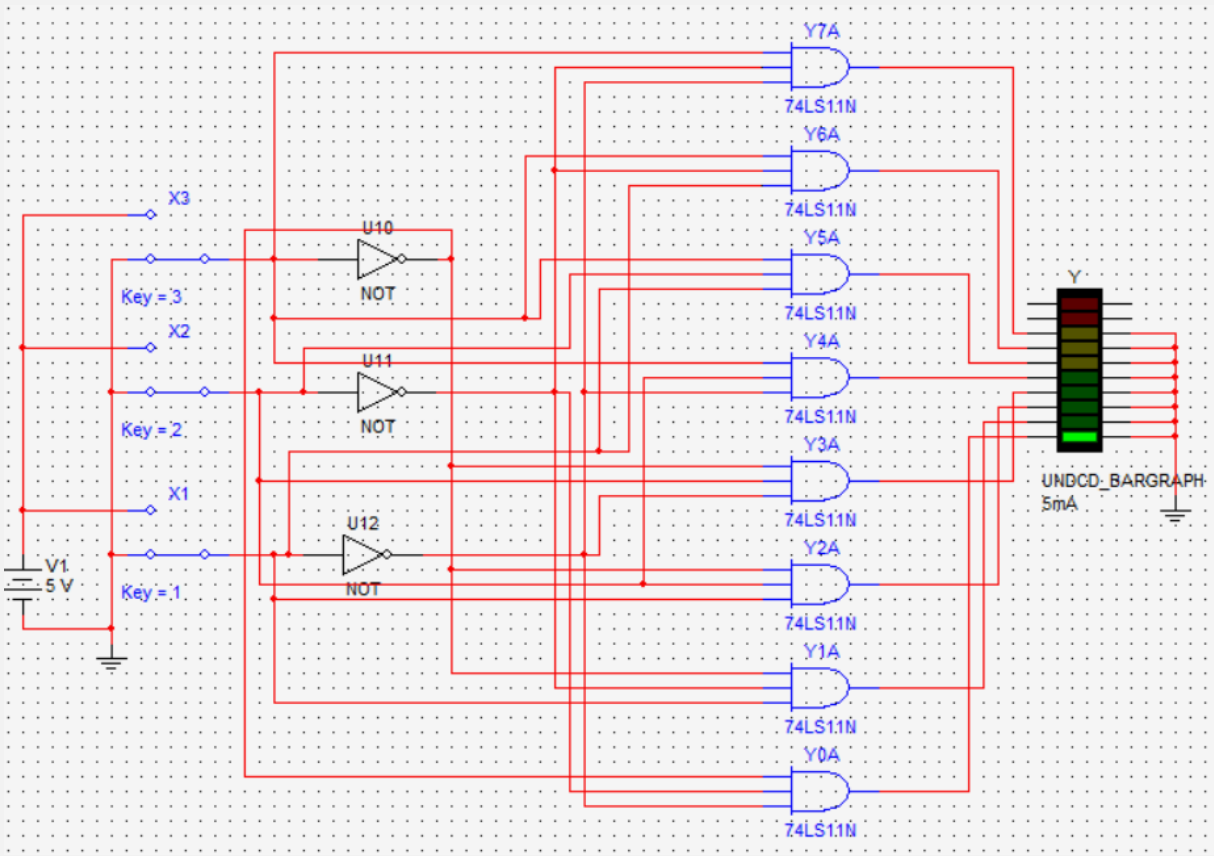


Рис. Б.8 – Схема моделювання дешифратора

Таблиця Б.2

Входи			Виходи							
X3	X2	X1	Y7A	Y6A	Y5A	Y4A	Y3A	Y2A	Y1A	Y0A
0	0	0								
0	0	1								
0	1	0								
0	1	1								
1	0	0								
1	0	1								
1	1	0								
1	1	1								

Заняття 3. Аналіз функціонування тригерів

Мета:

✓ вивчення прийомів моделювання цифрових автоматів за допомогою програми Multisim;

✓ аналіз функціонування JK, D, T тригерів.

Теоретична частина

Необхідні теоретичні відомості про тригери викладені в п. 3.1 посібника «Пам’ять цифрових автоматів».

Порядок виконання роботи

Скласти або отримати схему моделювання JK, D, T тригерів (зображена на рис. Б.9).

В роботі досліджується функціонування тригерів трьох типів на основі JK- тригера із додатковими входами S, R. Базовий JK-тригер має наступні особливості:

✓ одиничний активний рівень інформаційних сигналів;

✓ синхровхід базового JK-тригера – динамічний інверсний, тобто зміна стану тригера відбувається при переході 1-0 на вході C (синхронізація заднім фронтом синхроімпульсу);

✓ додаткові входи базового JK-тригера (вхід S уверху, вхід R – унизу) та перемикачі, позначені відповідними літерами, призначені для примусового встановлення тригера, при необхідності, в стани 0 та 1. Вказані входи мають, порівняно із входами J, K вищій пріоритет, тому, після встановлення необхідного початкового стану, необхідно встановити відповідний перемикач у розімкнуте положення. Крім того, одночасна подача високого рівня на входи S та R є неприпустимою (заборонена комбінація для RS-тригера з прямими входами).

Вибір типу тригера здійснюється перемикачами JK or T_D, T or D:

✓ JK-тригер – перемикач JK or T_D у верхньому положенні (як показано на рис. Б.9), при цьому положення перемикача T or D не має значення, інформаційні входи, відповідно, J, K;

✓ T-тригер – перемикач JK or T_D у нижньому положенні, перемикач T or D – у верхньому положенні, тобто, T-тригер утворюється із JK-тригера об’єднанням входів J, K, вхід J виконує функцію входу T;

✓ D-тригер – перемикач JK or T_D у нижньому положенні, перемикач T or D також у нижньому положенні, вхід J виконує функцію входу D.

Початковий стан тригера у всіх випадках встановлюється відповідним положенням перемикачів S, R.

Необхідно отримати та заповнити таблиці переходів для JK, D, T тригерів (таблиці Б.3, Б.4, Б.5 відповідно.), з’ясувати алгоритми функціонування вказаних тригерів.

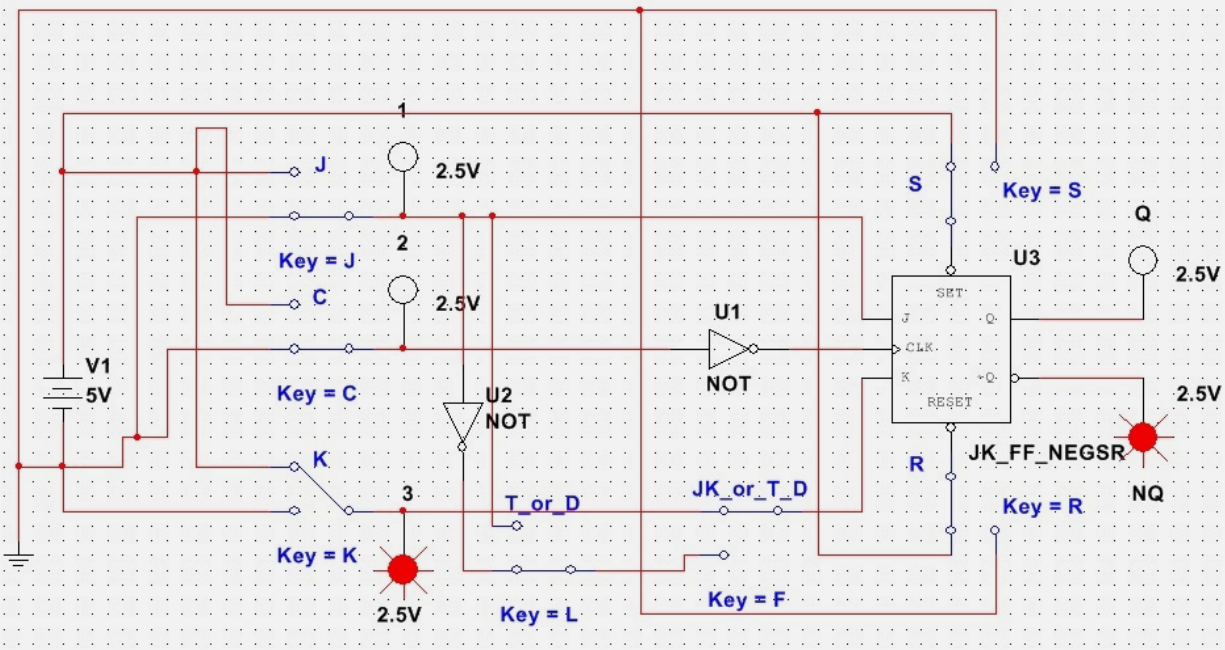


Рис. Б.9 – Схема моделювання JK, D, T тригерів

Таблиця Б.3

JK-тригер					
Входи			Виходи		
Інформаційні		Синхровхід	Початковий стан	Наступний стан	
J	K	C	Q	Q^{t+1}	not Q^{t+1}
0	0	0-1	0		
			1		
1	0		0		
			1		
0	1		0		
			1		
1	1		0		
			1		

Таблиця Б.4

D-тригер					
Входи			Виходи		
Інформаційні		Синхровхід	Початковий стан	Наступний стан)	
D (J)	K	C	Q	Q^{t+1}	not Q^{t+1}
0	J	0-1	0		
			1		
1			0		
			1		

Таблиця Б.5

T-тригер					
Входи			Виходи		
Інформаційні		Синхровхід	Початковий стан	Наступний стан	
T (J)	K	C	Q	Q^{t+1}	not Q^{t+1}
0	$\bar{J}$	0-1	0		
			1		
1			0		
			1		

Оформити та подати на перевірку викладачеві звіт, який містить:

- ✓ назву та мету заняття;
- ✓ схему моделювання та заповнені таблиці переходів для тригерів досліджених типів;
- ✓ висновки, в яких бажано стисло сформулювати алгоритми функціонування тригерів досліджених типів.

Заняття 4. Дослідження суматорів двійкових чисел

Мета: вивчення прийомів моделювання та аналіз функціонування арифметичних пристроїв за допомогою програми Multisim;

Теоретична частина

Необхідні теоретичні відомості про суматори двійкових чисел викладені в п. 4.4 посібника «Суматори двійкових чисел».

Порядок виконання роботи

1. Дослідження трирозрядного комбінаційного паралельного суматора

✓ Скласти або отримати схему моделювання трирозрядного комбінаційного паралельного суматора, зображену на рис. Б.10;

✓ отримати номер варіанта (варіанти подані таблицею Б.6, визначаються викладачем);

Таблиця Б.6

Варіант / Числа	1	2	3	4	5	6	7	8	9	10
A	5	3	1	2	3	1	2	5	2	3
B	2	4	6	3	1	4	4	1	1	3
C	7	4	3	4	2	3	4	7	5	7
D	6	7	5	5	7	6	6	5	6	3

✓ перевести числа A, B, C, D в трирозрядний натуральний двійковий код;

✓ за допомогою перемикачів A0 – A2 та B0 – B2, що перемикаються цифровими клавішами 0 – 5 відповідно, набрати в двійковому коді числа A і B;

✓ запустити симуляцію, перевірити по індикаторам правильність набору чисел і отриманого результату арифметичного додавання;

✓ за допомогою перемикачів A0 – A2 та B0 – B2, що перемикаються цифровими клавішами 0 – 5 відповідно, набрати в двійковому коді числа C і D;

✓ запустити симуляцію, перевірити по індикаторам правильність набору чисел і отриманого результату арифметичного додавання;

✓ заповнити таблицю Б.7;

Таблиця Б.7

Числа	Десяткове подання	Двійковий код 8421			
A		A_2	A_1	A_0	
B		B_2	B_1	B_0	
A + B		P_{i+1}	S_2	S_1	S_0
C		C_2	C_1	C_0	
D		D_2	D_1	D_0	
C + D		P_{i+1}	S_2	S_1	S_0

✓ проаналізувати заповнену таблицю, сформулювати висновки у вигляді стислої характеристики дослідженого суматора.

2. Дослідження пристрою додавання та віднімання двійкових чисел

✓ Скласти або отримати схему моделювання пристрою додавання та віднімання двійкових чисел, зображену на рис. Б.11;

✓ записати числа A, B, C, D в ПК (старшим розрядом записати знаковий розряд);

✓ за допомогою перемикачів Y1-Y3 і X1-X3, що перемикаються цифровими клавішами 1 – 8 відповідно, набрати числа A і B (знаки чисел встановлюються перемикачами Y3, X3);

✓ перемикач Z встановити у ніжне положення (+ – виконання операції додавання);

✓ результат виконання операції записати в таблицю Б.8 (рядок A+B);

✓ аналогічно додати числа C, D, результат виконання операції записати в таблицю Б.8 (рядок C+D);

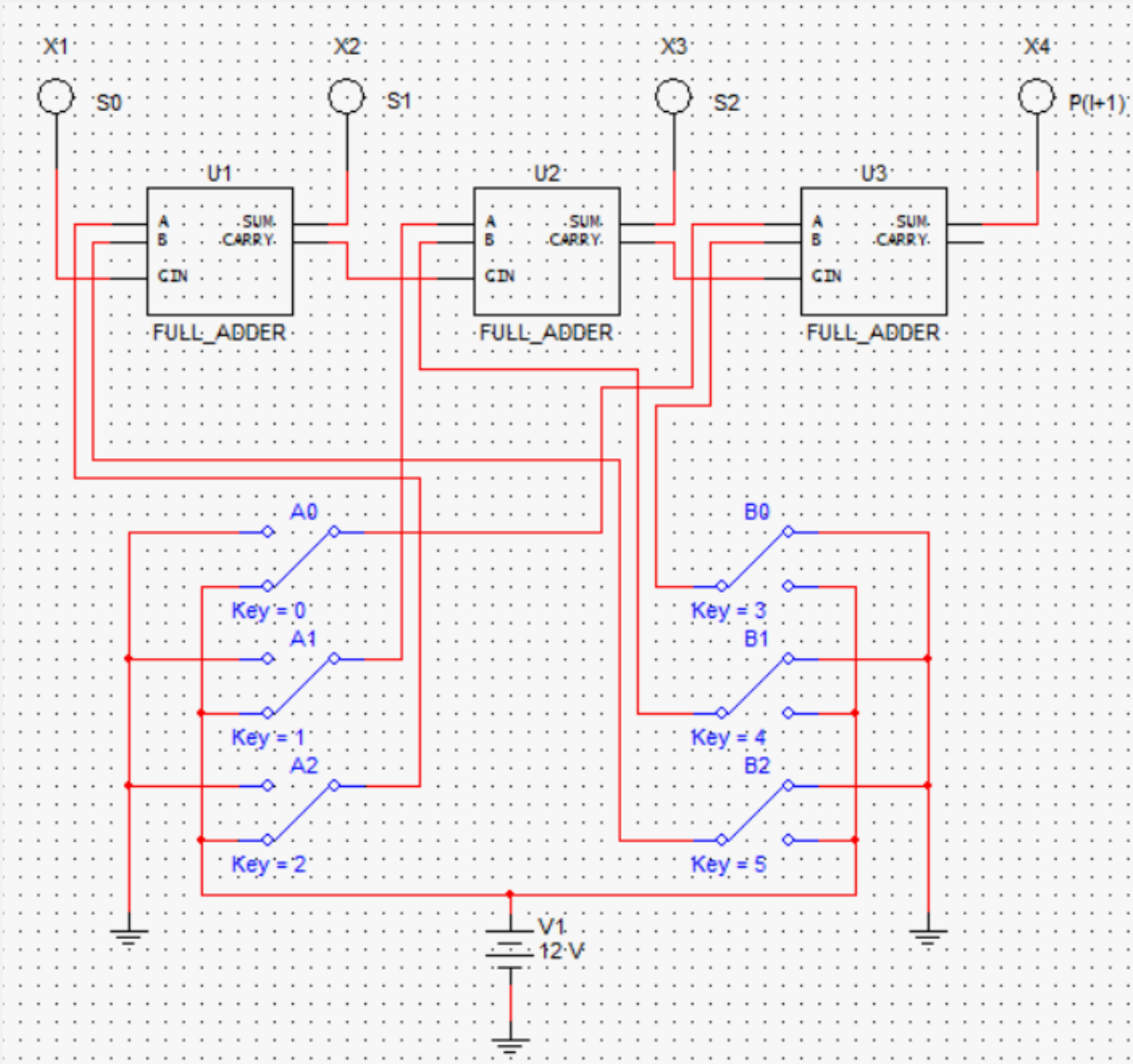


Рис. Б.10 – Модель трирозрядного комбінаційного паралельного суматора

Таблиця Б.8

Число	Знак	X3	X2	X1	Число	Знак	Y1	Y2	Y3	Операція	Результат				
											Overflow	S4	S3	S2	S1
A					B					A+B					
										A-B					
C					D					C+D					
										C-D					

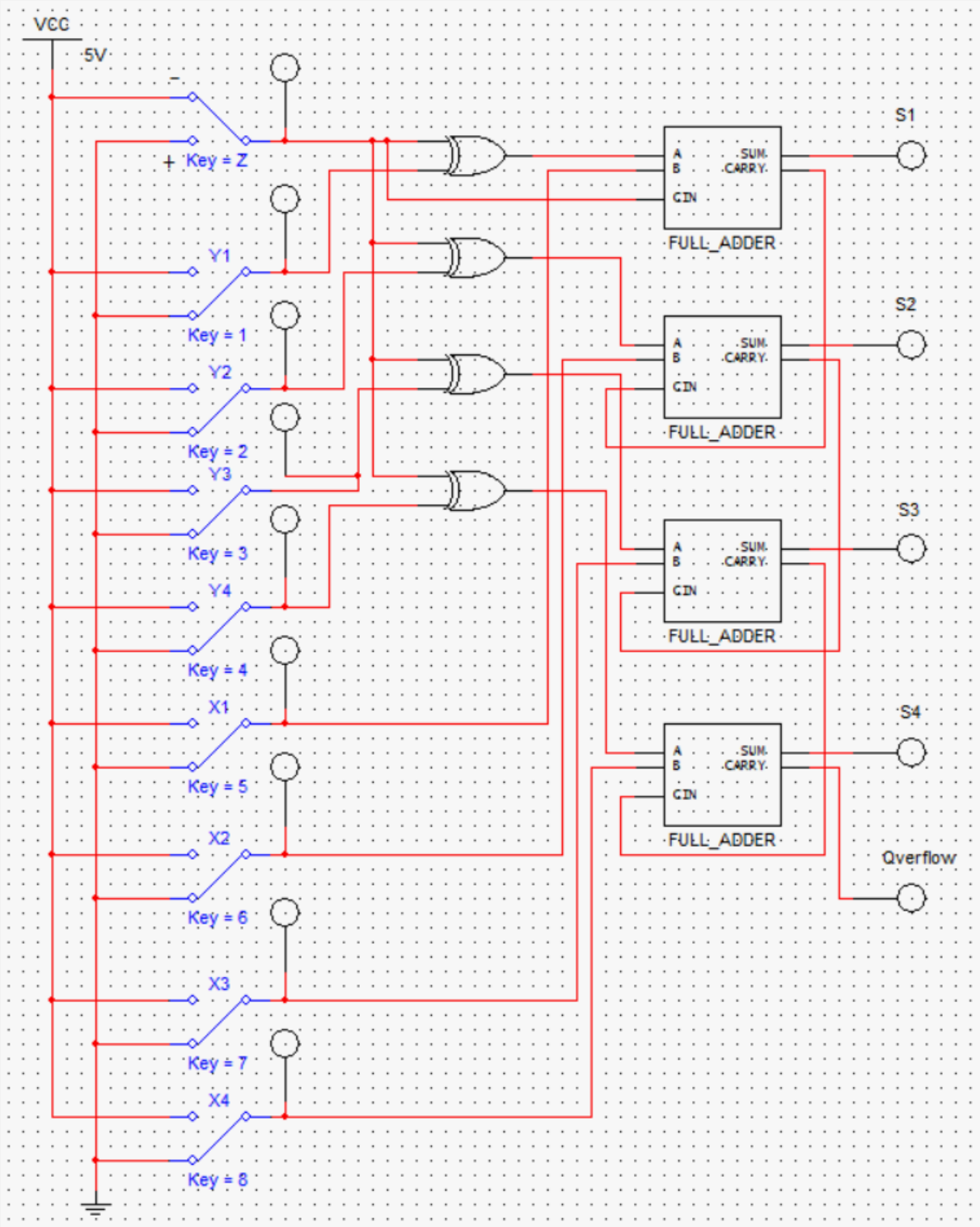


Рис. Б.11 – Схема моделювання пристрою додавання та віднімання

✓ записати числа B, D в ДК (старшим розрядом записати знаковий розряд);

✓ за допомогою перемикачів Y1-Y3 і X1-X3, що перемикаються цифровими клавішами 1 – 8 відповідно, набрати числа A і B (A – в ПК, B – в ДК, знаки чисел встановлюються перемикачами Y3, X3);

✓ перемикач Z встановити у верхнє положення (- – виконання операції віднімання);

✓ результат виконання операції записати в таблицю Б.8 (рядок A-B);

✓ аналогічно з числа C відняти число D, результат виконання операції записати в таблицю Б.8 (рядок C-D);

Скласти та подати на перевірку викладачеві звіт, який містить:

✓ назву та мету заняття;

✓ схеми моделювання та заповнені таблиці функціонування арифметичних пристроїв;

✓ висновки, в яких пояснити формування результатів виконання арифметичних операцій дослідженими пристроями.