МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Методичні вказівки

для лабораторних робіт студентів деннної форми навчання,

з дисципліни

«КОМП'ЮТЕРНА СХЕМОТЕХНІКА ТА АРХІТЕКТУРА КОМП'ЮТЕРІВ»

Напрям підготовки: – комп'ютерні науки

Одеса 2014

Методичні вказівки для виконання лабораторних робіт для студентів II- III курсів денної форми навчання. Дисципліна « Комп'ютерна схемотехніка та архітектура комп'ютерів ». Напрям підготовки – комп'ютерні науки. Препелиця Г.П. к.т.н., доц. , ст. викладач Пономаренко О.Л. – Одеса: ОДЕКУ, 2014р. – 88 с.

Передмова

При вивченні типових пристроїв цифрової техніки важливе місце займають питання експериментального дослідження їх структури, архітектури, а також питання автоматизованого логічного і схемотехнічного проектування зазначених пристроїв на рівні регістрів.

Освоєння принципів роботи типових пристроїв цифрової техніки базується, в основному, на синтезі цих пристроїв за допомогою формальних методів логічного проектування комбінаційних схем і цифрових автоматів. Методи аналізу роботи цифрових пристроїв грають у структурі розглянутого курсу допоміжну роль.

Поява другого видання навчального посібника обумовлена достатньо успішним досвідом застосування в учбовому процесі попереднього видання, а також необхідністю усунення виявлених друкарських помилок і додавання актуального, на думку автора, матеріалу.

Дійсний навчальний посібник повинен закріпити у студентів знання основного курсу “Комп’ютерна схемотехніка та архітектура комп’ютерів”, поглибити теоретичні знання по окремих розділах, яки мають найбільше практичне значення. Крім того, посібник повинен розвити практичні навички, необхідні для розробки і створення апаратних частин цифрових пристроїв по заданому закону їх функціонування. Наприклад, знання і навички, яки здобуваються, можуть бути використані для розробки програмно-апаратного комплексу систем екологічного моніторингу навколишнього середовища.

Навчальний посібник складається з трьох розділів: проектування і дослідження типових схем цифрової техніки, вивчення способів машинного перекладу чисел з довільної системи числення в довільну і докладний опису тем курсового проектування.

Лабораторний практикум, як і основний курс “Комп’ютерна схемотехніка та архітектура комп’ютерів ”, розрахований на проведення протягом двох семестрів. Відповідно до програми курсу він містить у собі лабораторні роботи з арифметичних і логічних основ цифрової техніки. Проектування і дослідження типових цифрових схем проводиться на універсальному лабораторному стенді (УЛС), що містить усі необхідні апаратні компоненти для виконання лабораторних робіт у повному обсязі. Вивчення і дослідження методів перекладу чисел може проводитися як на програмувальних калькуляторах, так і на персональних комп'ютерах. Для цього розроблені програми для різних засобів і на різних мовах. При бажанні пропоновані алгоритми можуть бути реалізовані на мовах і програмних засобах, обраних самими студентами за узгодженням з викладачем. Дві програми перекладу, реалізовані по розглянутим у теоретичній частині алгоритмам, приведені в додатках з відповідними коментарями.

Опис тем лабораторних робіт і курсового проектування є досить докладним, щоб служити матеріалом для самостійної роботи. Освоєння теоретичної частини може бути проведено самим студентом за списком контрольних питань до кожної лабораторної роботи. До речі, для допуску до виконання робіт використовуються тільки контрольні питання, на які студент відповідає по пам'яті на основі домашньої переробки матеріалу.

Кожна робота включає мету і зміст, домашнє завдання, теоретичні відомості, опис лабораторного стенда, порядок виконання роботи і рекомендації по змісту звіту.

Приведений наприкінці посібника список літератури дозволить студентам розширити свої представлення з досліджуваних питань, а контрольні питання, які є в кожній лабораторній роботі, дадуть можливість цілеспрямовано здійснювати домашню самопідготовку до проведення практикуму.

Опис завдань на проектування виконано досить докладно для того, щоб можна було б самостійно приступити до виконання проектування. Для полегшення роботи й однозначності одержаних результатів у завданнях приведена структура операційної частини (операційного автомата), докладний словесний опис роботи проектованого пристрою і мікропрограма мовою мікрооперацій функціонування керуючого автомата. Послідовне виконання всіх перерахованих пунктів проектування неминуче повинно привести до створення логічної схеми закінченого пристрою.

Більшість лабораторних робіт і тем курсових проектів входить у перелік типової програми з дисципліні “Комп’ютерна схемотехніка та архітектура комп’ютерів ” для спеціальності “Інформаційні-управляючі системи і технології”. Деякі з них можуть бути використані також при підготовці фахівців з інших споріднених технічних спеціальностей. На виконання лабораторної роботи з одним завданням потрібно п'ять годин попередньої підготовки і п'ять годин занять у лабораторії. Кількість пропонованих у навчальному посібнику робіт трохи перевищує число встановлених програмою за курсом “Комп’ютерна схемотехніка та архітектура комп’ютерів ”, що дає можливість, у залежності від конкретних умов і матеріально-технічного забезпечення вузу, відібрати необхідні.

Навчальний посібник складений на основі досвіду проведення лабораторних робіт на кафедрі “Інформаційні технології” Одеського державного екологічного університету. Книга може бути корисна також фахівцям розробляючих організацій, зайнятих проектуванням і розробкою вузлів і пристроїв цифрової техніки.

Лабораторна робота № 1

ВИВЧЕННЯ АЛГОРИТМУ ТА ПРОГРАМИ ПЕРЕТВОРЕННЯ ЦІЛИХ ЧИСЕЛ З ДОВІЛЬНОЇ СИСТЕМИ ЧИСЛЕННЯ У ДЕСЯТКОВУ ТА НАВПАКИ

1.1 Опис алгоритму та програми

1.1.1 Загальні етапи рішення задач на ПК. Використання персональних комп'ютерів забезпечує можливість комплексного рішення задач, зв'язаних з числовими розрахунками, та припускає закінченість дій – від словесної (вербальної) постановки завдання до одержання результату. Поза залежності від змісту завдання можна виділити наступні етапи її рішення.

1. Змістовий опис завдання, тобто її точне словесне формулювання з використанням однозначних математичних визначень та понять.
2. Формалізація змістового опису у вигляді, придатному для числових розрахунків (розробка алгоритму обчислювань).
3. Складання програми обчислювань мовою, відповідною обраному засобу обробки даних.
4. Введення програми та початкових даних.
5. Перевірка правильності складеної програми на контрольному прикладі та її налагодження (у випадку потреби).
6. Проведення обчислювань.
7. Аналіз та оформлення результатів.

Надалі при вивченні методів переведення чисел із однієї СЧ в іншу будемо дотримуватися наведеного плану рішення. Змістовий опис правил переведення був наведений в розділі "Основні теоретичні положення" – с. 96-115. Наступним етапом є формалізація завдання, що вирішується, тобто розробка алгоритмів переведення чисел.

1.1.2 Принцип побудови алгоритму перетворення позиційних представлень чисел. Із аналізу загальних методів перетворення виходить, що переведення чисел із однієї СЧ в іншу пов'язано з необхідністю виконання арифметичних дій над окремими розрядами чисел.

Якщо дії виконуються в новій СЧ, то кожний розряд старого представлення числа повинен множитися на основу старої СЧ у відповідному ступені. При виконанні дій в старій СЧ нове представлення числа формується із розрядів, послідовно отриманих часток внаслідок ділення на основу нової СЧ.

Таким чином, загальні методи припускають або формування нового числа із послідовно отриманих нових розрядів (робота в старій СЧ), або виконання неоднакових дій над окремими розрядами старого

представлення числа (робота в новій СЧ), але в будь-якому випадку перетворення чисел пов’язане з обробкою їх окремих розрядів.

В ПК з алгебраїчною логікою обчислювань виділення довільних розрядів числа можна виконувати, представляючи числові дані як строкові змінні. Однак при переведенні чисел окремі розряди отримуються або перетворюються послідовно один за одним, що значно спрощує поставлену задачу.

Послідовне виділення розрядів числа можна виконувати, наприклад, штучно зсуваючи кому в початковому цілому (або дробовому) числі, а потім відділяючи одержану цілу частину числа від дробової. Зсув десяткової коми на один розряд ліворуч (праворуч) виконується діленням (множенням) числа на десять.

Дробова частина числа отримується відніманням із початкового числа його цілої частини.

1.1.3 Алгоритм перетворення цілих чисел. Тому що в реальному житті всі арифметичні операції виконуються в ПК звичайно над десятковими числами, то має сенс вирішувати задачу переведення чисел, користуючись методом проміжного перетворення їх у десяткову систему числення, у якому одна із систем числення обов'язково повинна бути десятковою.

Таким чином за допомогою десяткових обчислень можна виконувати переведення чисел із СЧ з основою, в загальному випадку, відмінною від десяти, в десяткову СЧ чи із десяткової СЧ в будь-яку іншу з не десятковою основою. На відміну від десяткової умовно будемо називати інші СЧ, в загальному випадку, з не десятковими основами, довільними СЧ.

Крім того, для універсальності алгоритму будемо застосовувати звичайне ділення на основу нової системи числення замість цілочислового ділення, що використовується в другому методі переведення чисел (робота в старій СЧ). Однак у цьому випадку доводиться відновлювати остачу від ділення, для чого необхідно відокремити дробову частину частки й помножити її на ту ж основу.

Розглянемо структуру зображеного в словесному формулюванні алгоритму переведення десяткових чисел в довільну СЧ та навпаки відповідно до загальних методів і обговорених вище умов перетворення чисел:

1. Прийняти $i = 0$; $A_i = 0$; $N_s = НЦ_s = НЦ_{si}$; $s^i = s^0 = 1$.
2. Обчислити $НЦ_{si} / h$.
3. Виділити цілу $НЦ_{si+1}$ і дробову $НД_{si+1}$ частини ділення.
4. Відновити остачу від ділення $a_{i+1} = НД_{si+1} \cdot h = НЦ_{si} - НЦ_{si+1} \cdot h$.
5. Обчислити внесок розряду в результат $a_{i+1} \cdot s^i$.
6. Обчислити поточну суму $A_{i+1} = A_i + a_{i+1} \cdot s^i$.

7. Обчислити вагу наступного розряду $s^i = s \cdot s^i$.

8. Якщо ціла частина $NЦ_{si+1} = 0$, перейти до кроку 10 (інакше до кроку 9).

9. Прийняти i = i+1 та перейти до кроку 2.

10. Прийняти $N_h = NЦ_h = a_{i+1}$ і закінчити розв'язання завдання.

Тут *s* і *h* – основи старої та нової СЧ, одна з яких є десятковою;

A_i – акумулятор, де накопичується поточна сума результату переведення чисел;

a_{i+1} – відновлена остача від звичайного ділення;

$N_s = NЦ_s$ – початкове ціле число в старій СЧ;

$N_h = NЦ_h$ – результат в новій СЧ.

Приведений алгоритм є універсальним в тому розумінні, що дозволяє виробляти як перевід чисел з довільної СЧ в десяткову, так і зворотне переведення з десяткової СЧ в довільну. Інакше кажучи, десяткова СЧ може в задачах перетворення чисел бути як новою, так і старою. Залежно від цього в алгоритмі змінюється смислове значення кроків 2, 5, 6, 7.

При переведенні, наприклад, десяткових чисел в нову СЧ (коли десяткова СЧ виступає як стара) робота по даному алгоритму виконується в старій десятковій СЧ. В цьому випадку на 2-му кроці визначається чергова частка. Кроки 5 та 6 призначені для приєднання чергової остачі від ділення до числа, що формується, в новій СЧ. Кроком 7 визначається місце цифри наступного розряду в числі результату, що формується.

При переведенні в десяткову СЧ (коли вона виступає як нова) необхідні арифметичні дії виконуються в новій СЧ. Значення 2-го кроку алгоритму полягає в зсуві коми в старому числі на 1 розряд ліворуч для наступного виділення правого розряду числа. На 6-ом кроці цифра виділеного розряду множиться на його вагу, що визначає вклад даного розряду в кінцевий результат, який на 7-му кроці алгоритму додається (акумулюється) до окремої суми.

Таким чином, основа десяткової СЧ використовується в даному алгоритмі або для зсуву коми на один десятковий розряд в числі довільної СЧ з метою його наступного відділення, або для завдання знакомісця одержаної поточної цифри недесяткового розряду в сформованому числі довільної СЧ. Якщо при цьому основа довільної СЧ не більше десяти, то будь-яка її цифра розміщується в одному розряді ПК. Цифри ж систем числення, основи яких більше десяти (наприклад, шістнадцяткової СЧ), можуть займати на по два (та більше) десяткових розрядів. Практично із СЧ з основою більше десяти звичайно використовується, в основному, шістнадцяткова СЧ.

Переведення чисел з довільної СЧ в довільну по даному алгоритму виконується в два етапи з проміжним перетворенням в десяткову СЧ.

1.1.4 Блок-схема алгоритму переведення цілих чисел. Графічне зображення алгоритму перетворення в вигляді блок-схеми показано на рис. 1.1. Як видно, кількість блоків відрізняється від числа кроків алгоритму в словесному формулюванні. Це пояснюється різною мірою деталізації процесів в окремих блоках та кроках (що допустимо). Допоміжні блоки початку та кінця алгоритму не нумеруються, основні ж арифметичні (операторні) та логічні блоки позначені наскрізною нумерацією.

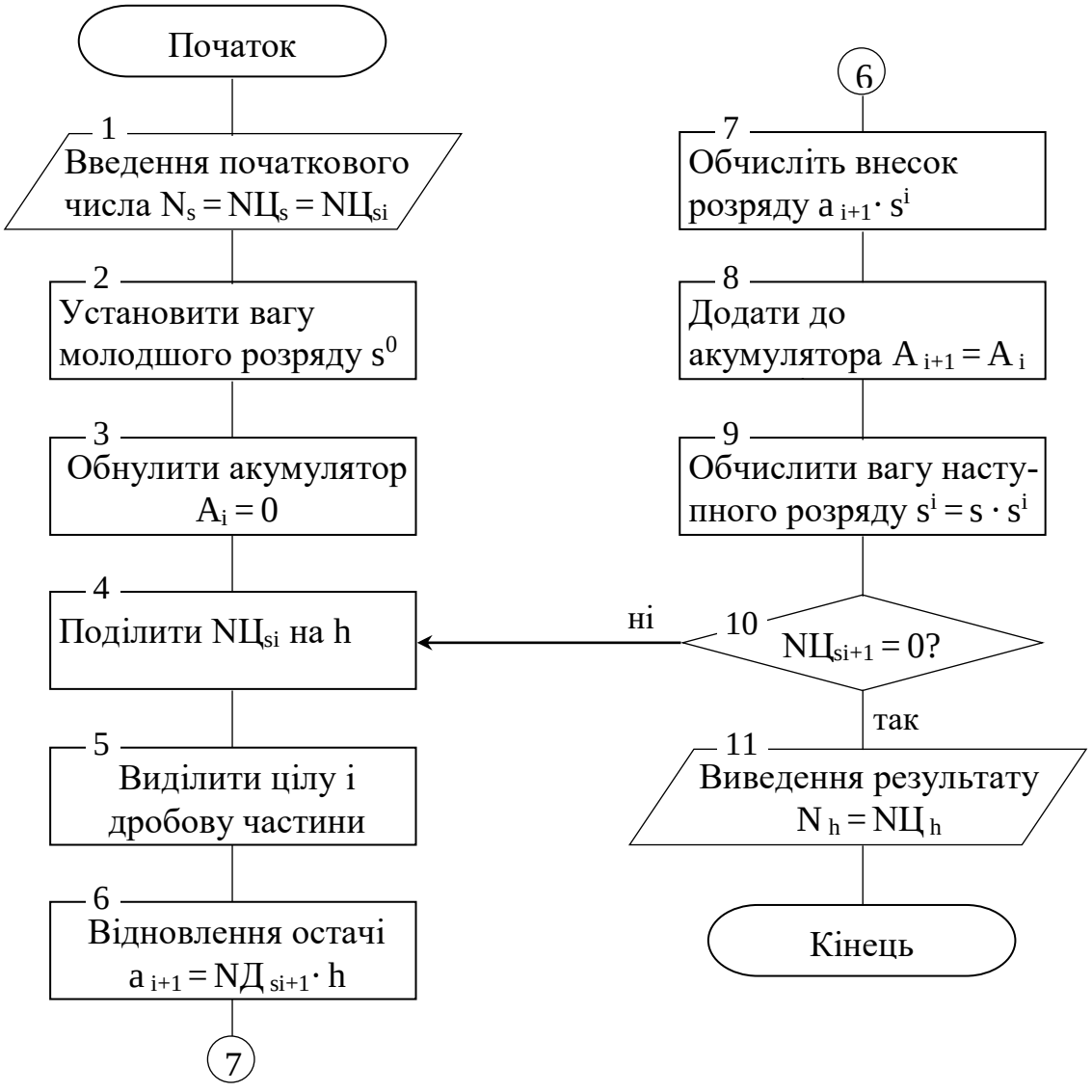


Рис. 1.1 – Блок-схема алгоритму переведення цілих чисел із довільної СЧ в десяткову та навпаки

Блоки 1, 2, 3 (рис. 1.1) відповідають першому кроку алгоритму у словесному опису. Крок 9 словесного опису реалізується в розглянутій

блок-схемі неявно тим, що наступні значення змінних запам'ятовуються в тих же місцях пам'яті, що і попередні.

1.1.5 Узагальнений алгоритм переведення чисел із цілою й дробовою частинами з довільної СЧ у довільну представлений на рис. 6.2. Алгоритм використовує метод переведення чисел з довільної СЧ в довільну з проміжним перетворенням початкового числа спочатку в десяткову СЧ. Таким чином обчислення завжди виконуються в десятковій СЧ з використанням одного з методів перекладу – в новій або старій СЧ, залежно від того, який виступає десяткова СЧ при переведенні.

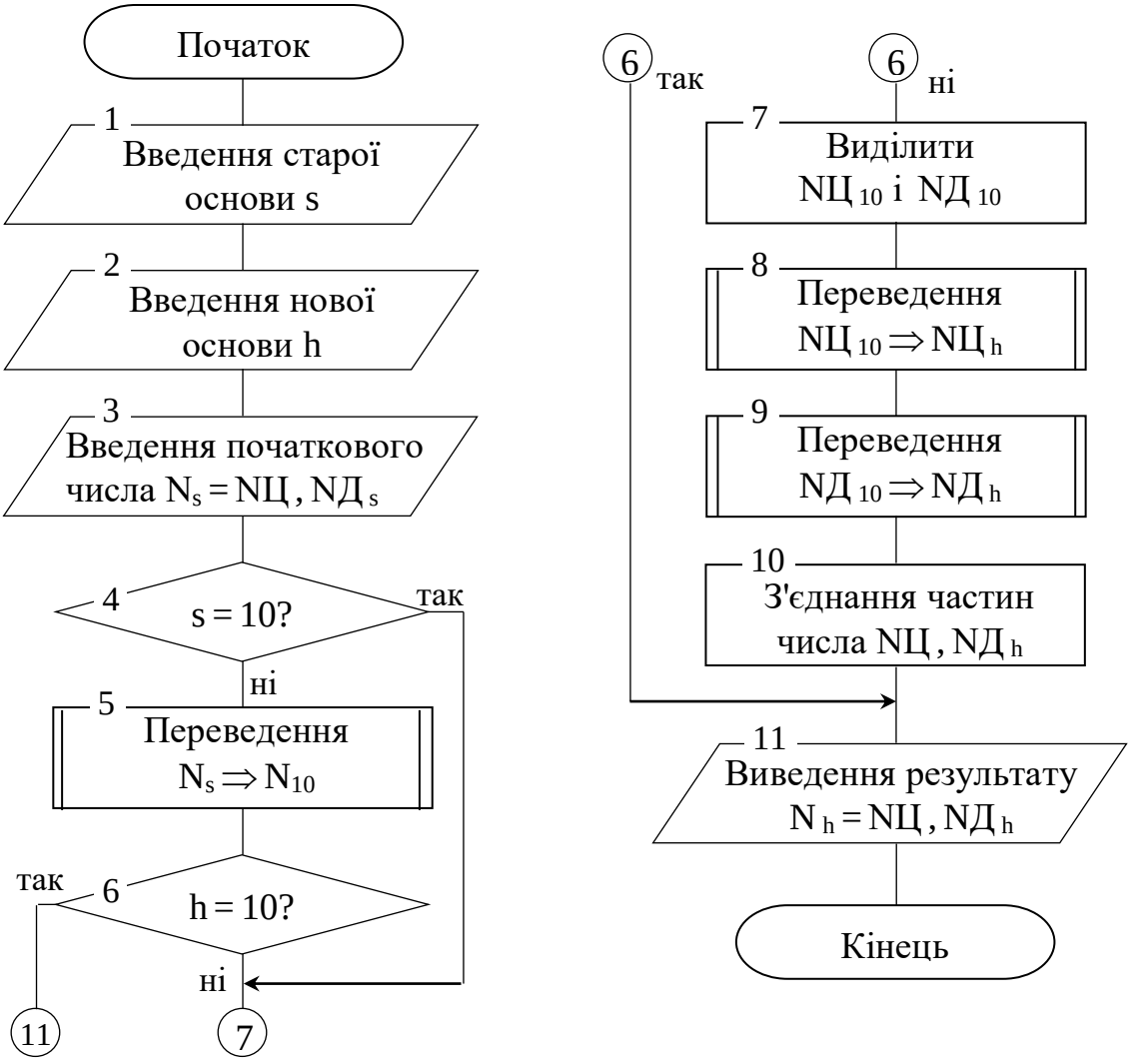


Рис. 1.2 – Блок-схема узагальненого алгоритму переведення чисел із цілою й дробовою частинами з довільної СЧ в довільну

Блок-схема узагальненого алгоритму містить три підпрограми перекладу, представлені в блоках 5, 8 і 9. Переклад чисел із старої СЧ в десяткову (блок 5) проводиться по формулі: $N_{10} = \sum_{0}^{n-1} N_{si} \cdot s^i$. Тут *Nsi* та *si* –

відповідно поточний розряд старого числа і його вага, що автономно представляються для обчислень в десятковій СЧ. Суттєвість підпрограми розкривається блок-схемою рис. 1.3. Правила дій над усіма розрядами однакові для цілої і дробової частин числа.

При перекладі чисел з десяткової системи використовується метод роботи в старій СЧ, в якому дії над цілою і дробовою частинами даних різняться. Тому в блоці 7 узагальненого алгоритму (рис. 1.2) спочатку проводиться розділення цілої і дробової частин десяткового числа.

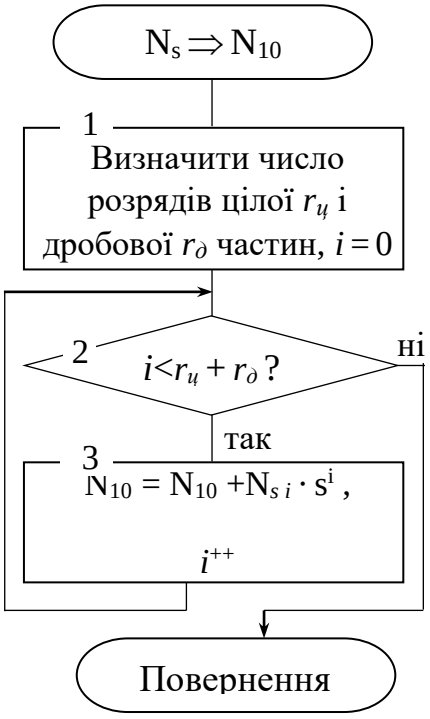


Рис. 1.3 – Підпрограма переведення чисел із довільної СЧ в десяткову

Потім обробка цілої (блок 8) і дробової (блок 9) частин числа виконується роздільно.

Для цілих чисел проводиться послідовне цілочисельне ділення числа і одержуваних часток на основу нової СЧ, приписуючи залишки від ділення в «голову» результату, поки остання частка не стане менше основи.

Для дробових чисел проводиться послідовне множення початкового дробу і дробових частин добутків на основу нової СЧ, приписуючи цілі частини добутків в «хвіст» результату.

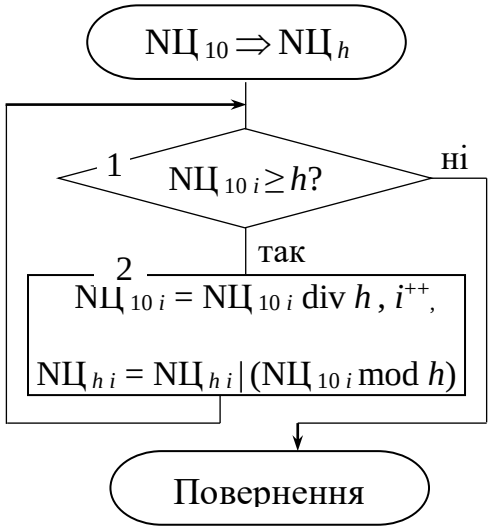


Рис. 1.4 – Підпрограма переведення цілих чисел із десяткової СЧ в довільну

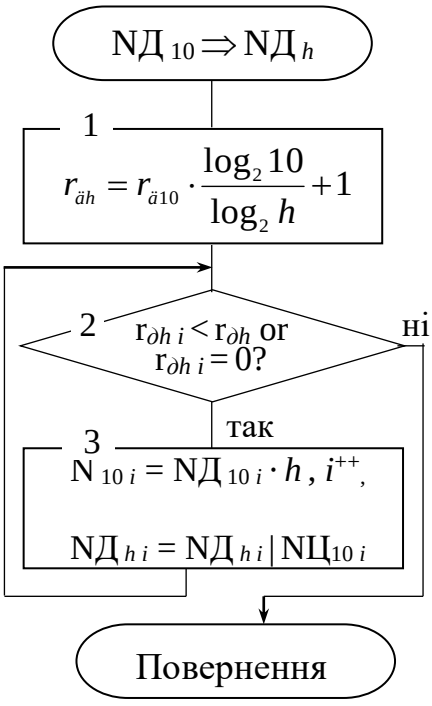


Рис. 1.5 – Підпрограма перекладу дробових чисел із десяткової СЧ в довільну

поки не буде держана необхідна кількість
розрядів дробу або дробова частина добутку не стане рівною нулю.

Процес перекладу десяткових чисел розкривається блок-схемами алгоритмів обробки цілих (рис. 1.4) і дробових (рис. 1.5) чисел. У операційному блоці 2 (рис. 1.4) відбите цілочисельне ділення (div) і приєднання до результату (символ конкатенації – |) залишків від ділення (mod).

На рис. 1.5 в блоці 1 визначається необхідна кількість розрядів дробового результату з урахуванням подальшого округлення, тобто на одиницю більше. У блоці 3 формується результат за допомогою приєднання (конкатенації) цілих частин добутків.

1.1.6 Універсальна програма Data Trans[1] перетворення цілих й дробових чисел з довільної СЧ у довільну реалізована на мові Си$^{++}$ з використанням розглянутого вище алгоритму та постачена відповідною довідкою по використанню. Інтерфейс програми з демонстрацією рішення ілюстративного прикладу переведення числа із цілою й дробовою частинами з одинадцяткової системи числення в п’ятіркову СЧ наведений на рис. 1.6.

Числа великої розмірності записуються та використовуються в програмі в природній формі (з фіксованою комою). Якщо вихідні дані задані в показовій формі (із плаваючої комою), то вони повинні бути вручну перетворені в природну форму представлення.

Програма розділяє вихідне число на цілу й дробову частини, обробляє ці частини окремо по різних алгоритмах і поєднує їх у єдиний вихідний результат, що дозволяє відразу здійснювати переведення як цілих або дробових чисел, так і чисел із цілою й дробовою частинами одночасно.

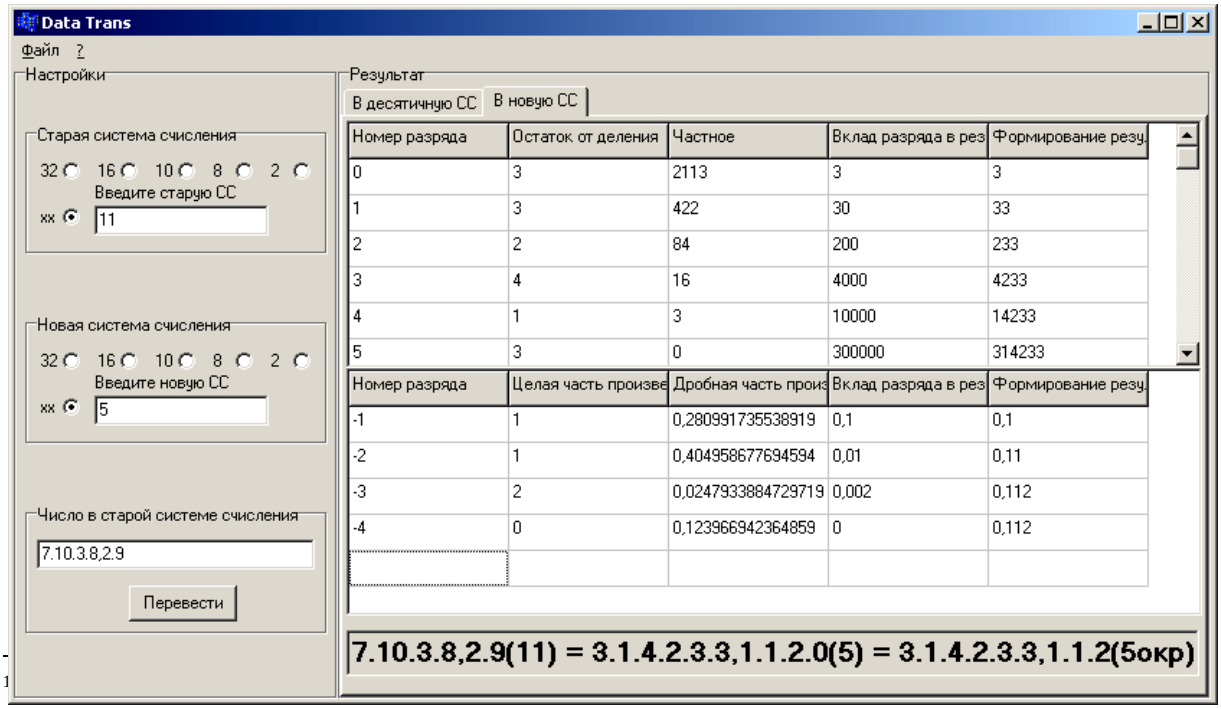


Рис. 1.6 – Інтерфейс універсальної програми переведення чисел Data Trans із довільної СЧ в довільну через десяткову

Програма розкриває процес формування результату переведення цілої й дробової частин числа (рис. 1.6). Вона містить у собі також універсальний модуль округлення результату з урахуванням точності переведення в довільній системі числення.

Звернемо увагу на те, що розряди вихідних даних, що заносяться у програму, відокремлюються друг від друга спеціальним роздільником – крапкою. Цей, на перший погляд, незручний спосіб введення насправді дозволяє працювати із числами абсолютно довільних систем числення без обмеження розмірності їхніх основ. Таким чином, програма стає дійсно універсальною, обробляючі числа будь-яких систем числення без обмежень.

При цьому ціла частка числа відокремлюється від дробової як завжди – комою, тобто без яких-небудь додаткових роздільників.

Результати деяких обчислень наведені в таблицях 1.1 й 1.2. Дані цих таблиць можуть використовуватися для контрольних прикладів при перевірці роботи застосовуваної програми і навчання роботи з нею.

Таблиця 1.1 – Приклади переведення цілих десяткових чисел в довільні СЧ

Початкове десяткове число	Нова СЧ	Результат (ручне перетворення в показову форму)
11 483	2	$1{,}0\ 110\ 011 \cdot 10^{1101}{}_{(2)} = 10\ 110\ 011\ 000\ 000_{(2)}$
11 483	5	$331\ 413_{(5)}$
$9{,}9\ 999\ 999 \cdot 10^{8}$	8	$7{,}3\ 465\ 446 \cdot 10^{11}{}_{(8)} = 7\ 346\ 544\ 600_{(8)}$
457 646	14	$B{,}CAD \cdot 10^{4} = BC\ AD0_{(14)}$
$1{,}0\ 358 \cdot 10^{9}$	16	$3{,}DBD \cdot 10^{7} = 3D\ BD0\ 000_{(16)}$
$1{,}0\ 358 \cdot 10^{9}$	32	$30{,}27\ 26\ 3 \cdot 10^{5}{}_{(32)} = 30\ 27\ 26\ \ 3\ 0\ 0_{(32)}$

Таблиця 1.2 – Приклади переведення цілих чисел з довільних СЧ в десяткову

Стара СЧ	Початкове число в довільній СЧ (ручне переведення в природну форму)	Результат в десятковій СЧ
2	$11\ 001\ 101_{(2)}$	205
5	$4{,}2\ 213\ 423 \cdot 10^{13}{}_{(5)} = 422\ 134\ 230_{(5)}$	1 755 565
8	$1{,}6\ 543\ 217 \cdot 10^{11}{}_{(8)} = 1\ 654\ 321\ 700_{(8)}$	$2{,}4\ 652\ 286 \cdot 10^{8}$
14	$A{,}BCD \cdot 10^{4}{}_{(14)}$	416 878
16	$A{,}BCD \cdot 10^{4}{}_{(16)}$	703 696
32	$30\ \ 27\ 18\ 3_{(32)}$	1 011 267

Примітка. Розряди 32-кових чисел та взагалі групи з трьох розрядів усіх чисел поділені прогалинами.

1.2 Домашнє завдання

1 Вивчити представлення та методи перетворення чисел з довільних позиційних СЧ в довільні.

2 Вивчити питання переведення чисел без втрати точності результату.

3 Ознайомитися з поняттям та класифікацією алгоритмів обчислювань.

4 Вивчити алгоритм перетворення на ПК чисел із довільної СЧ в десяткову та навпаки при роботі завжди в десятковій СЧ.

5 Розібратися з питанням універсальності машинного алгоритму переведення цілих чисел у зв'язку з можливістю його роботи як у новій, так і в старій СЧ.

6 Розібратися з роботою узагальненого алгоритму переведення чисел із цілою й дробовою частинами з довільної СЧ у довільну.

7 Розібратися з роботою універсальної програми Data Trans переведення чисел із цілою й дробовою частинами з довільної СЧ у довільну в середовищі Windows.

1.3 Контрольні запитання

1 Визначення та характеристика систем числення.

2 Загальні методи переведення чисел із довільної СЧ в довільну.

3 Переведення цілих чисел із довільної СЧ в довільну з виконанням арифметичних операцій в старій СЧ. Правило переведення. Приклади.

4 Визначення і організація циклів, класифікація по числу проходів.

5 Поняття та способи завдання алгоритмів.

6 Лінійні алгоритми.

7 Загальні етапи рішення задач на ПК.

8 Принцип послідовного виділення розрядів числа в алгоритмі перетворення чисел.

9 Алгоритм перетворення цілих чисел в словесному формулюванні.

10 Універсальність алгоритму перетворення цілих чисел.

11 Розподіл розрядів при завданні чисел в універсальній програмі переведення чисел.

12 Блок-схема алгоритму переведення цілих чисел.

13 Блок-схема узагальненого алгоритму переведення чисел із цілою й дробовою частинами з довільної СЧ у довільну.

14 Представлення цілих чисел у різних СЧ у показовій і природній формі (на прикладі початкових даних, наведених у таблицях 6.1 і 6.2).

15 Наведіть приклад переведення числа $25_{(10)}$ з десяткової системи числення у вісімкову СЧ й назад, діючи суворе по блок-схемі алгоритму рис. 6.1.

16 Наведіть приклад переведення числа $25_{(10)}$ з десяткової системи числення у шістнадцяткову СЧ й назад, діючи суворе по блок-схемі алгоритму рис. 6.1.

17 Наведіть приклад переведення числа $25_{(10)}$ з десяткової системи числення у тридцятидвійкову СЧ й назад, діючи суворе по блок-схемі алгоритму рис. 6.1.

1.4 Порядок виконання роботи

1 Запустити програму перетворення чисел.

2 Перевірити роботу впровадженої програми Data Trans по контрольним прикладам, початкові дані для яких взяти із табл. 6.1, 6.2.

3 Виконати переведення заданих чисел за допомогою програми Data Trans з оформленням результатів відповідно до табл. 6.1, 6.2.

4 Користуючись інтерфейсною таблицею програми (рис. 6.6), отримати й проаналізувати процес порозрядного формування результату переведення наданого десяткового числа у восьмеричну СЧ і зворотно.

5 Користуючись даними п. 4 записати приклади ручного переведення наданого десяткового числа у восьмеричну СЧ і зворотно у звичайному вигляді.

6 Оформити протокол лабораторної роботи с короткими висновками.

1.5 Зміст звіту

1 Мета роботи.

2 Загальні методи перетворення цілих чисел.

3 Блок-схема машинного алгоритму переведення цілих чисел з описом його універсальності.

4 Стислий опис універсальної програми переведення чисел із цілою й дробовою частинами з довільної СЧ у довільну.

5 Результати перетворення заданих чисел у форматі таблиць 6.1 і 6.2, отриманих за допомогою програми переведення Data Trans.

6 Приклади ручного порозрядного формування результатів переведення наданого десяткового числа в вісімкову СЧ та зворотно у звичайному вигляді (по результатам роботи програми переведення Data Trans).

7 Стисле обґрунтування одержаних результатів (висновки).

1.6 Варіанти завдань до роботи

Початкові дані для перетворення в СЧ з основами 10, 2, 5, 8, 14, 16, 32 наведені в табл. 1.3, в якій основи всіх СЧ умовно показані втіх же СЧ відповідно. Десяткове число послідовно перевести в усі інші із зазначених СЧ, результати надати відповідно до табл. 1.1. Інші числа перевести в десяткову СЧ, а результати надати в відповідності з табл. 1.2.

Таблиця 1.3 – Таблиця варіантів (до ЛР № 1)

№ п/п	Системи числення						
	10	2	5	8	14	16	32
1	$2{,}56\cdot10^{8}$	$1{,}1\cdot10^{1001}$	$4{,}321\cdot10^{14}$	$7{,}643\cdot10^{10}$	2A9D	2A9D	27 17 0 E
2	$4{,}79\cdot10^{9}$	$1\cdot10^{10000}$	$3{,}412\cdot10^{13}$	$4{,}761\cdot10^{11}$	$A\cdot10^{4}$	$4\cdot10^{4}$	25 31 7 29
3	$2{,}437\cdot10^{9}$	111101010	40231321	$7{,}777\cdot10^{10}$	BCD3	$D\cdot10^{4}$	$17{,}16\cdot10^{4}$
4	976543250	101011110	40323112	765432150	27B3	27E3	A B 19 31
5	$3{,}594\cdot10^{8}$	111100110	312410130	123577640	7AB3	7AB8	23 E 20 7
6	567989320	101010100	321342420	321765120	7DDC	7FFF	31 1 0 B
7	$2{,}345\cdot10^{8}$	110011000	123432010	44762513	3DAD	3FFF	A 17 29 3
8	99999999	101101110	102034140	457625740	1DA3	1FFF	A,B $18\cdot10^{4}$
9	909739850	110101010	23004232	535626440	DAD2	12FE	29 A 2 19
10	543219870	101101100	240420310	53562645	$5\cdot10^{4}$	EF2A	$17{,}D\cdot10^{4}$
11	102304050	111111110	204032440	44334452	ACAD	A3B2	1 A 18 3
12	$4{,}507\cdot10^{8}$	100001110	$2{,}434\cdot10^{14}$	70615243	$D\cdot10^{4}$	$C\cdot10^{4}$	7 16 21 0
13	123456780	110001110	43210123	76540123	C3B8	C3B8	$3{,}2\ 26\cdot10^{4}$
14	987654320	111001110	432101230	26150437	9A8B	9A8B	18 A 19 3
15	876543210	$1{,}1\cdot10^{1000}$	432143210	$2{,}345\cdot10^{11}$	B917	B917	3 B 20 25
16	75023481	101110110	$2{,}434\cdot10^{13}$	$7{,}171\cdot10^{10}$	45BA	FFFF	27 31 16 2
17	562324270	101111010	$4{,}321\cdot10^{14}$	15471724	BC29	BC29	19 C B 3
18	$6{,}768\cdot10^{8}$	$1{,}01\cdot10^{1000}$	$3{,}123\cdot10^{12}$	$5{,}672\cdot10^{12}$	DAB3	ED2A	20 5 C 29
19	75023481	101010101	123012103	272312335	ABCD	EDA7	25 16 A 17
20	$3{,}264\cdot10^{9}$	$1{,}11\cdot10^{1001}$	$4{,}331\cdot10^{13}$	$3{,}653\cdot10^{13}$	BCD7	DED8	5 D 29 17
21	97532148	110110110	234321012	561564373	CD89	ABCD	19 25 18 C
22	$4{,}361\cdot10^{9}$	$1{,}01\cdot10^{1010}$	$2{,}434\cdot10^{12}$	$1{,}664\cdot10^{14}$	DC98	CDEF	F 23 18 D
23	64932571	100100101	124331032	276156243	5D6B	7A8F	17 4 A 30
24	$5{,}678\cdot10^{10}$	$1{,}11\cdot10^{1011}$	$4{,}231\cdot10^{14}$	$2{,}472\cdot10^{15}$	9AAB	8B9E	A 31 21 9
25	84114392	100110101	312312321	263514375	CAD0	23DA	B C 30 21
26	102304350	110110110	214032442	45334352	AC9D	A4BD	7 28 17 3
27	$2{,}36\cdot10^{11}$	$1{,}01\cdot10^{1011}$	$2{,}134\cdot10^{13}$	$1{,}573\cdot10^{13}$	B8DC	CEDF	23 F D 17
28	563219872	101011101	243421312	54563645	$6\cdot10^{5}$	FEA2	$27{,}E\cdot10^{5}$
29	$5{,}372\cdot10^{7}$	$1{,}01\cdot10^{1101}$	$4{,}323\cdot10^{13}$	$7{,}652\cdot10^{12}$	A2D9	A2D9	27 30 3 E
30	$4{,}77\cdot10^{11}$	$1{,}1\cdot10^{10010}$	$4{,}432\cdot10^{14}$	$4{,}767\cdot10^{13}$	$A\cdot10^{6}$	$4\cdot10^{5}$	25 31 B 31
31	92998793	101101010	102134140	453625732	2DA4	3FEF	A,C $17\cdot10^{5}$

32	709738852	110101011	23233113	53562632	DAC2	52FE	19 C 7 29

Примітка. Розряди 32-кових чисел поділені прогалинами.

Лабораторна робота № 2

ВИВЧЕННЯ АЛГОРИТМУ ТА ПРОГРАМИ ПЕРЕТВОРЕННЯ ДРОБОВИХ ЧИСЕЛ ІЗ ДОВІЛЬНОЇ СЧ В ДЕСЯТКОВУ ТА НАВПАКИ

2.1 Опис алгоритму та програми

2.1.1 Алгоритм перетворення дробових чисел, приведений нижче, також як і розглянутий раніше алгоритм переведення цілих чисел, призначений для роботи як в новій, так і в старій СЧ залежно від того, якою є в умові задачі десяткова СЧ – новою чи старою. Розглянемо словесну формулювання алгоритму переведення дробових чисел із довільної СЧ в десяткову та навпаки, в основу якого покладені загальні методи перетворення чисел:

1. Взяти $i=1$; $s^i = s^1 = s$; $A = 0$; $НД_{si} = НД_s$; $j = n$.
2. Обчислити $НД_{si} \cdot h$.
3. Виділити цілу $НЦ_{si+1}$ і дробову $НД_{si+1}$ частини результату множення.
4. Обчислити вклад поточного розряду $НЦ_{si+1}/s^i$.
5. Обчислити поточну суму $A_{i+1} = A_i + НЦ_{si+1} / s^i$.
6. Обчислити вагу наступного розряду $s^i = s^i \cdot s$.
7. Якщо $A_{i+1} = 0$, то перейти до кроку 10 (інакше до кроку 8).
8. Прийняти $j = j-1$.
9. Якщо $j = 0$, то перейти до кроку 11 (інакше до кроку 10).
10. Прийняти $i = i+1$ та перейти до кроку 2.
11. Прийняти $НД_h = a_{i+1}$ та закінчити розв'язок завдання.

Тут:

s і *h* – основи відповідно старої та нової СЧ, одна з яких є десятковою;

$НД_s$ і $НД_h$ – відповідно вихідна і результуюча дроби в старій та новій СЧ;

n – необхідна кількість значущих (ненульових) розрядів результату.

На відміну від попереднього, в даному алгоритмі використати признак рівності нулю дробової частини добутку $НД_{si+1}$, як умову виходу із циклу, неможливо. Справа в тому, що початковий кінцевий дріб в новій СЧ може бути еквівалентним нескінченому результуючому дробу. А це означає, що дробова частина добутку $НД_{si+1}$ навіть при якій завгодно великій кількості циклів ніколи не стане рівній нулю.

Раніше зазначалося, що переведення дробів звичайно виконується наближеним одержанням необхідної кількості розрядів результату. Точність переведення на ПК визначається його форматом даних (розрядною сіткою) і величинами основ чисел, що перетворюються. При

цьому кількість значущих цифр результату не повинна залежати (зменшуватися) від числа його початкових нульових розрядів після коми, інакше точність переведення буде зменшуватись.

Оскільки будь-яка цифра результату – значуща або незначуща – в даному алгоритмі обробляється однаково (однократним обчисленням тіла циклу), то загальне число його повторень визначається кількістю (в загальному випадку заздалегідь невідомою) початкових нулів результуючого дробу та необхідним числом значущих розрядів результату, що задається заздалегідь. Звідси випливає, що для перетворення дробів необхідно використовувати обидва способи організації циклічних алгоритмів: ітераційний – для обробки початкових нульових розрядів та детермінований – для обробки заданої кількості значущих розрядів.

У вигляді логічної умови, яка визначає число повторів циклу з ітераційною організацією, використовуються величина накопиченої суми результату A_{i+1}, нульове значення якої є ознакою продовження, а ненульове – ознакою виходу з ітераційного циклу (кроки 7, 10 алгоритму).

Детермінований алгоритм організується за допомогою спеціального лічильника, в який заздалегідь заноситься необхідне число повторень циклу. Вміст цього лічильника декрементується після кожного обчислення тіла циклу аж до нуля, при досягненні якого здійснюється вихід із циклу (кроки 8, 9, 10). Принципово *n* може бути будь-яким великим цілим числом, однак розряди результату, отримані за межами довжини мантиси розрядної сітки, будуть загублені, хоч на їх обчислення і буде затрачено машинного часу.

Залежно від того, якою в перетвореннях виступає десяткова СЧ – новою чи старою, змінюється смислове значення кроків 2, 4, 5, 6 алгоритму.

В перетвореннях, в яких десяткова СЧ виступає старою, на 2-му кроці алгоритму визначається цифра чергового розряду результату в новій СЧ як ціла частина $NЦ_{si+1}$ добутку. Кроки 4 та 5 призначені для запису одержаної цифри по визначеному знакомісцю в ряду цифр, що зображають результуюче число. На 6-му кроці підготовлюються дані для визначення знакомісця цифри наступного розряду результату.

У випадку, коли десяткова СЧ виступає в перетвореннях як нова, на 2-му кроці алгоритму виконується зсув коми праворуч на один розряд з метою наступного виділення чергового лівого розряду початкового дробу. На 4-му кроці визначається вклад даного розряду в кінцевий результат, окреме значення якого обчислюється на 5-му кроці. На 6-му кроці алгоритму підготовлюється вага наступного розряду початкового дробу для використання його в повторному циклі.

2.1.2 Блок-схема алгоритму переведення дробових чисел показана на рис. 2.1. Підготовчі блоки 1...4 відповідають 1-му кроку, а логічні блоки

10, 11 – відповідно крокам 7, 10 або 8, 9, 10 в словесному алгоритму. Докладніше блок-схеми алгоритму розглянуті в п. 1.1.4 ЛР № 1.

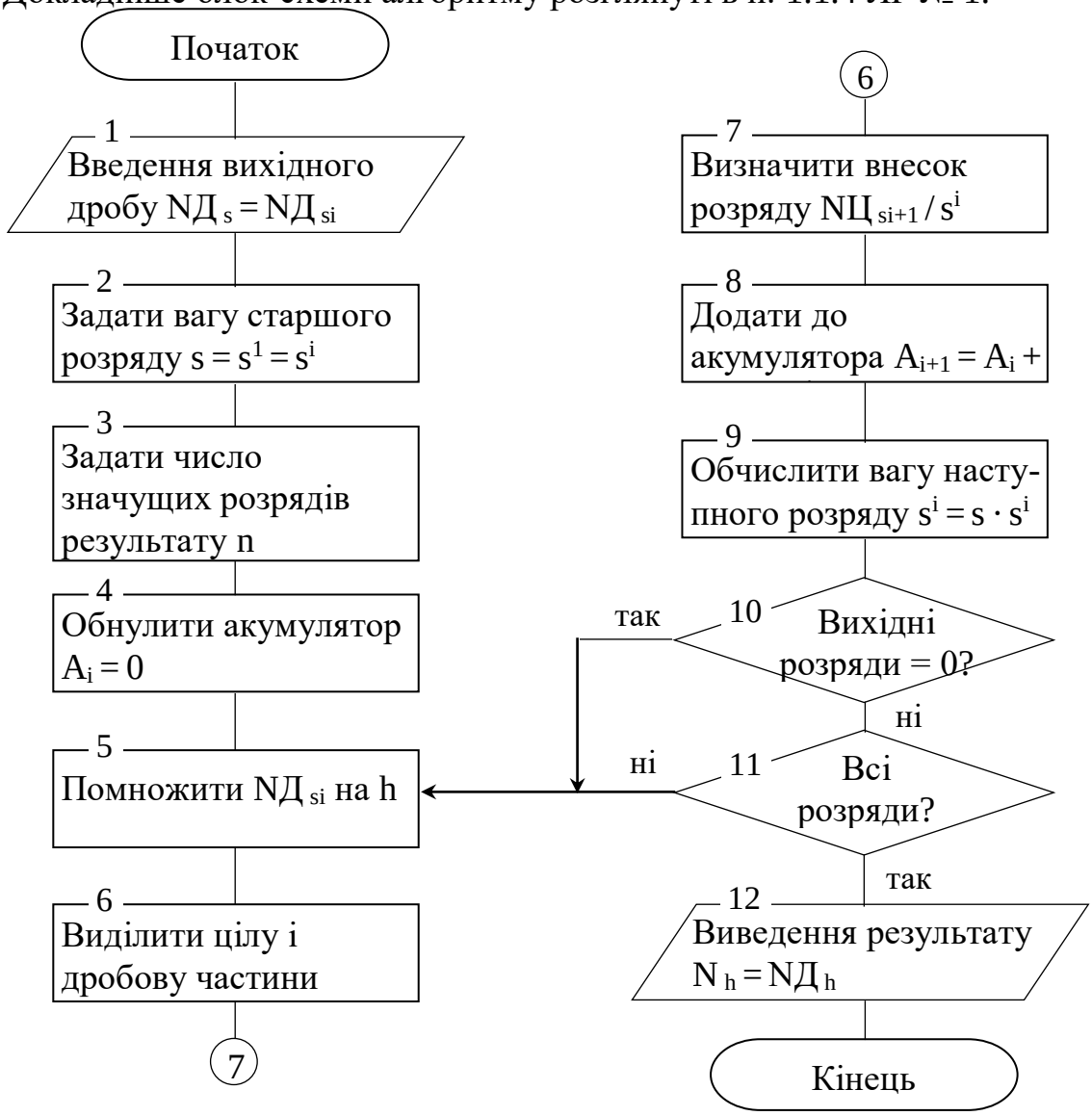


Рис. 2.1 – Блок-схема алгоритму переведення дробових чисел з довільної СЧ в десяткову та навпаки

2.1.3 Програма перетворення дробових чисел – та ж універсальна програма Data Trans, яка використовувалася і в ЛР № 1.

Приклади переведення дробових чисел, що можуть використовуватися як контрольні, наведені в табл. 2.1 та 2.2.

2.2 Домашнє завдання

Те ж саме, що і для лабораторної роботи № 1.

2.3 Контрольні запитання

1 Визначення та характеристика систем числення.

2 Системи числення, що застосовуються в цифрових пристроях.

3 Переведення чисел із довільної СЧ в довільну з виконанням арифметичних дій в новій СЧ. Правило переведення, приклади.

4 Переведення дробових чисел із довільної СЧ в довільну з виконанням арифметичних дій в старій СЧ. Правило переведення, приклади.

5 Графічне зображення алгоритму.

6 Алгоритм перетворення дробових чисел в словесному формулюванні.

7 Організація циклів в алгоритмі перетворення дробових чисел.

8 Блок-схема алгоритму перетворення дробових чисел.

9 Організація ітераційного циклу в програмі переведення чисел.

10 Організація детермінованого циклу в програмі переведення.

11 Блок-схема узагальненого алгоритму переведення чисел із цілою й дробовою частинами з довільної СЧ у довільну.

12 Представлення дробових чисел у різних СЧ у показовій і природній формі (на прикладі початкових даних, наведених у таблицях 2.1 і 2.2).

13 Наведіть приклад переведення дробового числа $0{,}37_{(10)}$ з десяткової системи числення у вісімкову СЧ й назад, діючи суворе по блок-схемі алгоритму рис. 2.1.

14 Наведіть приклад переведення числа $0{,}37_{(10)}$ з десяткової системи числення у шістнадцяткову СЧ й назад, діючи суворе по блок-схемі алгоритму рис. 2.1.

15 Наведіть приклад переведення числа $0{,}37_{(10)}$ з десяткової системи числення у тридцятидвійкову СЧ й назад, діючи суворе по блок-схемі алгоритму рис. 2.1.

Таблиця 2.1 – Приклади переведення 10-кових дробів в довільну СЧ

Вихідний десятковий дріб	Нова СЧ	Результат
0,083	2	$1{,}0101001\cdot10^{-100}{}_{(2)}$
0,85029	2	$0{,}11011001_{(2)}$
$9{,}9486925\cdot10^{-5}$	5	$1{,}2341234\cdot10^{-11}{}_{(5)}$
0,85029	8	$0{,}66326232_{(8)}$
$6{,}735475\cdot10^{2}$	14	$D{,}2B7\cdot10^{-2}{}_{(14)} = 0{,}0D2B7_{(14)}$
0,026	16	$6{,}A7E\cdot10^{-2}{}_{(16)} = 0{,}06A7E_{(16)}$
$9{,}934875\cdot10^{-1}$	32	$0{,}31\ 25\ A\ 19_{(32)}$

2.4 Порядок виконання роботи

Той же, що і для роботи № 1.

При виконанні контрольних прикладів та оформленні результатів обчислювань використати формати табл. 2.1 та 2.2.

Таблиця 2.2 – Приклади перекладу дробів з довільної СЧ в 10-кову

Вихідний дріб в довільній СЧ	Стара СЧ	Результат в десятковій СЧ
$1{,}1010111\cdot10^{-101}{}_{(2)}$	2	$5{,}2490235\cdot10^{-2}$
$3{,}4210243\cdot10^{-12}{}_{(5)}$	5	$4{,}9778361\cdot10^{-5}$
$6{,}7576456\cdot10^{-2}{}_{(8)}$	8	$1{,}0888134\cdot10^{-1}$
$B{,}A9C\cdot10^{-2}{}_{(14)} = 0{,}0BA9C_{(14)}$	14	$6{,}0023353\cdot10^{-2}$
$0{,}9BA5_{(16)}$	16	$6{,}0798645\cdot10^{-1}$
$30{,}23\ 29\ C\cdot10^{-1}{}_{(32)} = 0{,}30\ 23\ 29\ C_{(32)}$	32	$9{,}6085739\cdot10^{-1}$

2.5 Зміст звіту

1. Мета роботи.
2. Загальні методи перетворення дробових чисел.
3. Блок-схема машинного алгоритму переведення дробових чисел з описом його універсальності.
4. Результати перетворення заданих дробових чисел у форматі табл. 2.2 та 2.3, отриманих за допомогою програми переведення Data Trans..
5. Приклади ручного порозрядного формування результатів переведення наданого дробового десяткового числа в вісімкову СЧ та навпаки у звичайному вигляді (по результатам роботи програми переведення Data Trans).
6. Стисле обґрунтування одержаних результатів (висновки).

2.6 Варіанти завдань до роботи

Початкові дані для перетворення в СЧ з основами 10, 2, 8, 14, 16, 32 приведені в табл. 7.3. Десятковий дріб необхідно перевести у всі інші із зазначених СЧ з представленням результатів в відповідності з табл. 2.1. Останні початкові дроби перевести в десяткову СЧ, подавши результати в форматі табл. 2.2.

[1; 13-15]

Таблиця 2.3 – Таблиця варіантів (до ЛР № 2)

№ п/п	Системи числення						
	10	**2**	**5**	**8**	**14**	**16**	**32**
1	$4{,}65502\cdot10^{-3}$	$1{,}100001\cdot10^{-101}$	$2{,}22344\cdot10^{-11}$	$7{,}776543\cdot10^{-1}$	$C{,}0DA\cdot10^{-1}$	$A{,}CAD\cdot10^{-2}$	$9{,}19\ 0\ 17\cdot10^{-2}$
2	0,00768	$1{,}101101\cdot10^{-100}$	$1{,}23142\cdot10^{-14}$	$2{,}345\cdot10^{-11}$	$2{,}A9D\cdot10^{-3}$	$D{,}EF9\cdot10^{-2}$	$17{,}0\ E\ 27\cdot10^{-2}$
3	0,86456723	$1{,}111001\cdot10^{-110}$	$1{,}23412\cdot10^{-13}$	$7{,}1725\cdot10^{-10}$	$4{,}3AB\cdot10^{-2}$	$1{,}2E3\cdot10^{-3}$	$B{,}0\ C\ 19\cdot10^{-1}$
4	$2{,}56732\cdot10^{-4}$	$1{,}0001101\cdot10^{-10}$	$2{,}43210\cdot10^{-12}$	$1{,}547172\cdot10^{-3}$	$D{,}20B\cdot10^{-4}$	$B{,}ACE\cdot10^{-1}$	$7{,}16\ A\ 30\cdot10^{-3}$
5	0,000256457	0,0010110101	$4{,}32104\cdot10^{-11}$	$3{,}754674\cdot10^{-6}$	$2{,}7A3\cdot10^{-5}$	$C{,}EED\cdot10^{-2}$	$A{,}B\ 31\ 19\cdot10^{-2}$
6	$3{,}5789\cdot10^{-5}$	0,00011011011	$4{,}00234\cdot10^{-10}$	$1{,}257446\cdot10^{-2}$	$D{,}27A\cdot10^{-3}$	$7{,}6AB\cdot10^{-3}$	$2{,}7\ 27\ E\cdot10^{-1}$
7	$3{,}48467\cdot10^{-3}$	$1{,}0001111\cdot10^{-11}$	$2{,}043024\cdot10^{-4}$	0,76543056	$D{,}275\cdot10^{-2}$	$E{,}F2A\cdot10^{-4}$	$7{,}23\ A\ 28\cdot10^{-1}$
8	$2{,}54867\cdot10^{-2}$	$1{,}000011\cdot10^{-110}$	$3{,}210402\cdot10^{-3}$	$7{,}643201\cdot10^{-3}$	$A{,}BCD\cdot10^{-3}$	$C{,}0DA\cdot10^{-2}$	$A{,}F\ 19\ 25\cdot10^{-3}$
9	0,004758674	$1{,}010101\cdot10^{-111}$	$1{,}014123\cdot10^{-2}$	$4{,}030247\cdot10^{-5}$	$C{,}0DA\cdot10^{-2}$	$D{,}0CA\cdot10^{-1}$	$C{,}0\ 23\ 18\cdot10^{-1}$
10	0,02550292	$1{,}110011\cdot10^{-101}$	$2{,}24432\cdot10^{-12}$	$2{,}76342\cdot10^{-7}$	$C{,}AD6\cdot10^{-5}$	$F{,}FAB\cdot10^{-2}$	$F{,}20\ 1\ 17\cdot10^{-2}$
11	0,08502987	$1{,}1100011\cdot10^{-10}$	$2{,}300432\cdot10^{-4}$	$5{,}764037\cdot10^{-2}$	$C{,}3AD\cdot10^{-1}$	$B{,}03B\cdot10^{-2}$	0,0 25 D 26 E
12	0,85675432	0,010110101	$4{,}23421\cdot10^{-13}$	$7{,}443272\cdot10^{-4}$	$B{,}7CA\cdot10^{-2}$	$D{,}ED\cdot10^{-3}$	$7{,}16\ 21\ B\cdot10^{-2}$
13	$6{,}75324\cdot10^{-3}$	$1{,}000011\cdot10^{-100}$	$2{,}124142\cdot10^{-3}$	$2{,}450340\cdot10^{-5}$	$C{,}8B4\cdot10^{-2}$	$A{,}F17\cdot10^{-3}$	$2{,}5\ 25\ E\cdot10^{-2}$
14	$7{,}54335\cdot10^{-2}$	$1{,}0011001\cdot10^{-11}$	$4{,}43322\cdot10^{-10}$	$2{,}234043\cdot10^{-6}$	$C{,}0BA\cdot10^{-3}$	$F{,}AC5\cdot10^{-2}$	$16{,}21\ A\ 7\cdot10^{-1}$
15	$8{,}29345\cdot10^{-2}$	$1{,}011001\cdot10^{-10}$	$3{,}02104\cdot10^{-12}$	$7{,}654321\cdot10^{-3}$	$B{,}017\cdot10^{-4}$	$F{,}03B\cdot10^{-1}$	$3{,}1\ 31\ 19\cdot10^{-1}$
16	$4{,}75039\cdot10^{-4}$	$1{,}010101\cdot10^{-101}$	$2{,}34012\cdot10^{-10}$	$1{,}72635\cdot10^{-10}$	$9{,}ABC\cdot10^{-2}$	0,03FBA	0,0 0 1 30 D 8
17	$7{,}45204\cdot10^{-5}$	$1{,}0100101\cdot10^{-10}$	$4{,}32143\cdot10^{-14}$	$2{,}460531\cdot10^{-5}$	$7{,}6B9\cdot10^{-2}$	$6{,}7DB\cdot10^{-4}$	$7{,}25\ 19\ A\cdot10^{-2}$
18	0,067238645	$1{,}100111\cdot10^{-1010}$	$2{,}23413\cdot10^{-14}$	$6{,}21725\cdot10^{-11}$	$3{,}4CAB\cdot10^{-3}$	$2{,}12E3\cdot10^{-4}$	$B{,}0\ C\ 19\cdot10^{-2}$
19	$4{,}605302\cdot10^{-4}$	$1{,}1000101\cdot10^{-110}$	$2{,}322344\cdot10^{-13}$	$7{,}7076543\cdot10^{-11}$	$C{,}0D2A\cdot10^{-2}$	$A{,}C2AD\cdot10^{-2}$	$8{,}19\ 0\ 17\cdot10^{-2}$
20	0,00020768	$1{,}1101101\cdot10^{-101}$	$3{,}131424\cdot10^{-12}$	$2{,}304541\cdot10^{-13}$	$2{,}0A9D\cdot10^{-3}$	$D{,}EF39\cdot10^{-2}$	$27{,}0\ E\ 27\cdot10^{-2}$
21	$2{,}50789\cdot10^{-5}$	0,000101011011	$4{,}000234\cdot10^{-10}$	$1{,}2507446\cdot10^{-2}$	$D{,}207A\cdot10^{-3}$	$7{,}6A0B\cdot10^{-3}$	$9{,}7\ 27\ E\cdot10^{-3}$
22	$5{,}048467\cdot10^{-3}$	$1{,}00101111\cdot10^{-11}$	$2{,}0403024\cdot10^{-4}$	0,0076543056	$D{,}2705\cdot10^{-2}$	$E{,}F20A\cdot10^{-4}$	$7{,}23\ A\ 27\cdot10^{-4}$
23	$2{,}540867\cdot10^{-3}$	$1{,}0100011\cdot10^{-110}$	$2{,}3010402\cdot10^{-3}$	$6{,}7643201\cdot10^{-3}$	$A{,}B0CD\cdot10^{-3}$	$C{,}0D5A\cdot10^{-2}$	$A{,}E\ 19\ 25\cdot10^{-3}$
24	0,0304758674	$1{,}1010101\cdot10^{-111}$	$2{,}0104123\cdot10^{-2}$	$3{,}0340247\cdot10^{-5}$	$C{,}0D2A\cdot10^{-2}$	$D{,}0CBA\cdot10^{-1}$	$C{,}0\ 23\ 18\cdot10^{-4}$
25	$8{,}54335\cdot10^{-3}$	$1{,}10011001\cdot10^{-101}$	$2{,}43322\cdot10^{-11}$	$6{,}234043\cdot10^{-6}$	$C{,}0BCA\cdot10^{-3}$	$F{,}AC75\cdot10^{-2}$	$16{,}21\ D\ 9\cdot10^{-1}$
26	$7{,}29345\cdot10^{-4}$	$1{,}1011001\cdot10^{-110}$	$3{,}02104\cdot10^{-13}$	$7{,}654321\cdot10^{-3}$	$B{,}01C7\cdot10^{-4}$	$F{,}037B\cdot10^{-3}$	$3{,}1\ 27\ 19\cdot10^{-3}$
27	$3{,}641539\cdot10^{-4}$	$1{,}100010101\cdot10^{-101}$	$2{,}43401232\cdot10^{-12}$	$3{,}7263523\cdot10^{-10}$	$A{,}ABC\cdot10^{-2}$	$A{,}ABC\cdot10^{-2}$	0,0 0 21 30 D 8
28	$7{,}45039\cdot10^{-4}$	$1{,}01010101\cdot10^{-1011}$	$2{,}3434012\cdot10^{-13}$	$1{,}7062635\cdot10^{-10}$	$7{,}ABC\cdot10^{-4}$	0,0F3ABA	0,0 0 1 30 E 18
29	$4{,}85204\cdot10^{-5}$	$1{,}0100101\cdot101^{-101}$	$4{,}3214321\cdot10^{-14}$	$2{,}54605312\cdot10^{-5}$	$9{,}6B9\cdot10^{-3}$	$6{,}7D0B\cdot10^{-4}$	$9{,}25\ 19\ C\cdot10^{-3}$

Лабораторна робота № 3

СИНТЕЗ І ДОСЛІДЖЕННЯ КОМБІНАЦІЙНИХ СУМАТОРІВ

Мета лабораторної роботи:

– вивчити засоби підсумовування багаторозрядних слів та організації міжрозрядних переносів у паралельних суматорах;

– провести логічне та технічне проектування паралельних комбінаційних суматорів (*КСМ*) з паралельними та комбінованими переносами;

– експериментально дослідити робочу схему паралельного *КСМ* з послідовними переносами.

1.1 Домашнє завдання

Вивчити:

– засоби проектування однорозрядних комбінаційних суматорів (*ОКСМ*) у різноманітних операторних формах (*ОФ*);

– засоби комбінаційного підсумовування багаторозрядних слів;

– засоби організації міжрозрядних переносів в паралельних суматорах;

– ознайомитися з методикою проведення лабораторної роботи і описом універсального лабораторного стенду з цифрової техніки.

1.2 Лабораторне завдання

1.2.1 Вивчити опис і ознайомитися з позначеннями елементів на лицевій панелі універсального лабораторного стенда, перевірити працездатність використаних в роботі елементів стенда.

1.2.2 Реалізувати на логічних елементах (*ЛЕ*) *І-НЕ* схему однорозрядного *КСМ* (рис. 1.1) і дослідити його роботу з таблиці істинності (*ТІ*).

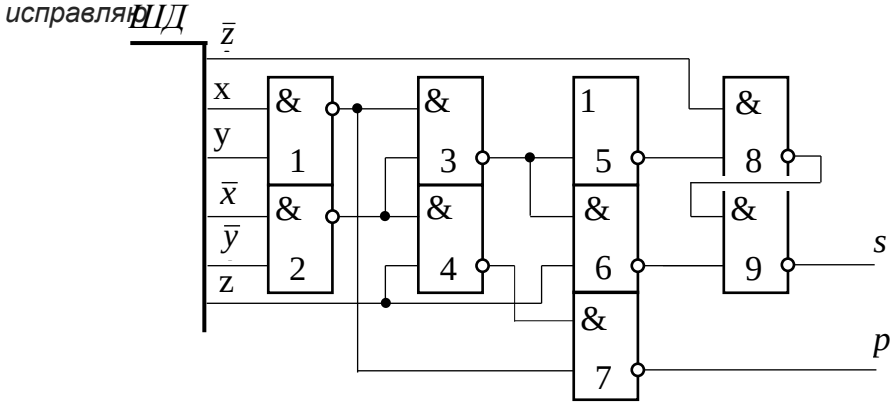


Рис. 1.1 – Принципова схема *ОКСМ*

1.2.3 Дослідити з *ТІ* роботу одного із вмонтованих однорозрядних *КСМ*, умовні графічні позначення яких наведені на лицевій панелі стенда.

1.2.4 Розробити математичну модель дворозрядної схеми паралельного переносу (*СПП*) і її принципову схему в другій *ОФ*. Представити *СПП* у вигляді функціонального елемента.

1.2.5 Розробити функціональну схему дворозрядного *КСМ* з паралельною організацією переносів.

1.2.6 Розробити функціональну схему чотирирозрядного суматора із паралельно-послідовною організацією переносів ($m = 2$), використовуючи результати п. 1.2.5. Організувати зв'язки, необхідні для роботи *СМ* в зворотному чи додатковому коді, відповідно до завдання на проектування (табл. 1.1).

1.2.7 За заданим варіантом скласти приклади алгебраїчного підсумовування кодів чисел зі знаками.

1.2.8 Розробити і зібрати робочу схему чотирирозрядного *КСМ* з послідовними переносами, яка має засоби набору вихідних даних, записи суми в регістр зберігання і виявлення переповнення розрядної сітки суматора (*ППРС*). Дослідити роботу схеми.

1.2.9 Реалізувати розроблені приклади на робочій схемі, порівняти і оцінити результати.

1.2.10 Експериментально визначити повний час підсумовування і час затримки проміжного переносу в робочій схемі *КСМ*.

1.3 Прилади, використовувані в роботі

1. Універсальний лабораторний стенд (*УЛС*).
2. Електронний осцилограф.

Таблиця 1.1 – Таблиця варіантів

Номер варіанта	Цифрові розряди слів (±X, ±Y)	Коди слів	Номер варіанта	Цифрові розряди слів (±X, ±Y)	Коди слів
1	001; 111	ЗК	13	101; 011	ДК
2	111; 110	ДК	14	100; 111	ЗК
3	010; 111	ЗК	15	111; 001	ДК
4	110; 101	ДК	16	110; 111	ЗК
5	010; 110	ЗК	17	111; 010	ДК
6	111; 101	ДК	18	101; 110	ЗК
7	011; 111	ЗК	19	110; 010	ДК
8	101; 100	ДК	20	100; 101	ЗК
9	011; 110	ЗК	21	111; 011	ДК
10	110; 100	ДК	22	100; 110	ЗК
11	011; 101	ЗК	23	110; 011	ДК
12	111; 100	ДК	24	101; 111	ЗК

1.4 Опис універсального лабораторного стенда

УЛС призначений для вивчення роботи основних елементів цифрової техніки. На стенді можуть бути реалізовані різноманітні комбінаційні схеми (наприклад, суматори, схеми порівняння, дешифратори, мультиплексори та ін.) і цифрові автомати (тригери різноманітних типів, регістри, лічильники та ін.). На лицьовій панелі розташовані умовні графічні позначення використовуваних елементів (*ЛЕ*, тригери, однорозрядні *КСМ*, генератори імпульсів, тумблерні регістри для набору кодів), до входів і виходів яких підключені виводи реальних елементів мікроелектронної інтегральної серії К155. Виводи елементів з'єднуються між собою комутаційними шнурами згідно з досліджуваною схемою.

На лицьовій панелі також розташовані органи керування генераторами імпульсів (*ГІ*) і поодиноких імпульсів (ГПІ), набірними регістрами *X* і *Y*. Генератор імпульсів видає періодичні прямокутні сигнали однієї із трьох частот – 15 Гц, 5 кГц, 500 кГц. Генератор поодиноких імпульсів видає поодинокий прямокутний імпульс певної тривалості при натискуванні на клавішу. Про рівень сигналу (0 або 1) на виходах елементів можна судити з стану світлодіодів, розміщених в графічних позначеннях елементів.

Набірні регістри *X* і *Y* дозволяють задати довільне двійкове число і передати його прямим, зворотним (інверсним) або парафазним кодом на входи будь-якої набраної схеми. Включений тумблер (або клавіша) створює одиничний (високий додатний) рівень сигналу на прямому виході відповідного розряду регістру. Регістри мають керуючу лінію видавання коду (*ВК*), що дозволяє логічно вимикати їх виходи від входів наступних схем. При нульовому рівні керуючого сигналу *ВК* на усіх виходах регістрів (прямих і інверсних) створюється одиничний рівень, що відповідає відсутності інформації, оскільки одиниця для *ЛЕ І-НІ* – неактивний рівень сигналу. За вимкненої лінії *ВК* або подачі на неї одиничного (розв'язувального) рівня керуючого сигналу коди чисел, що набрані на регістрах, передаються на входи наступних схем.

1.5 Порядок виконання основних етапів лабораторної роботи

Увага! Для запобігання виведення з ладу забороняється з'єднувати ***виходи*** будь-яких елементів стенда між собою!

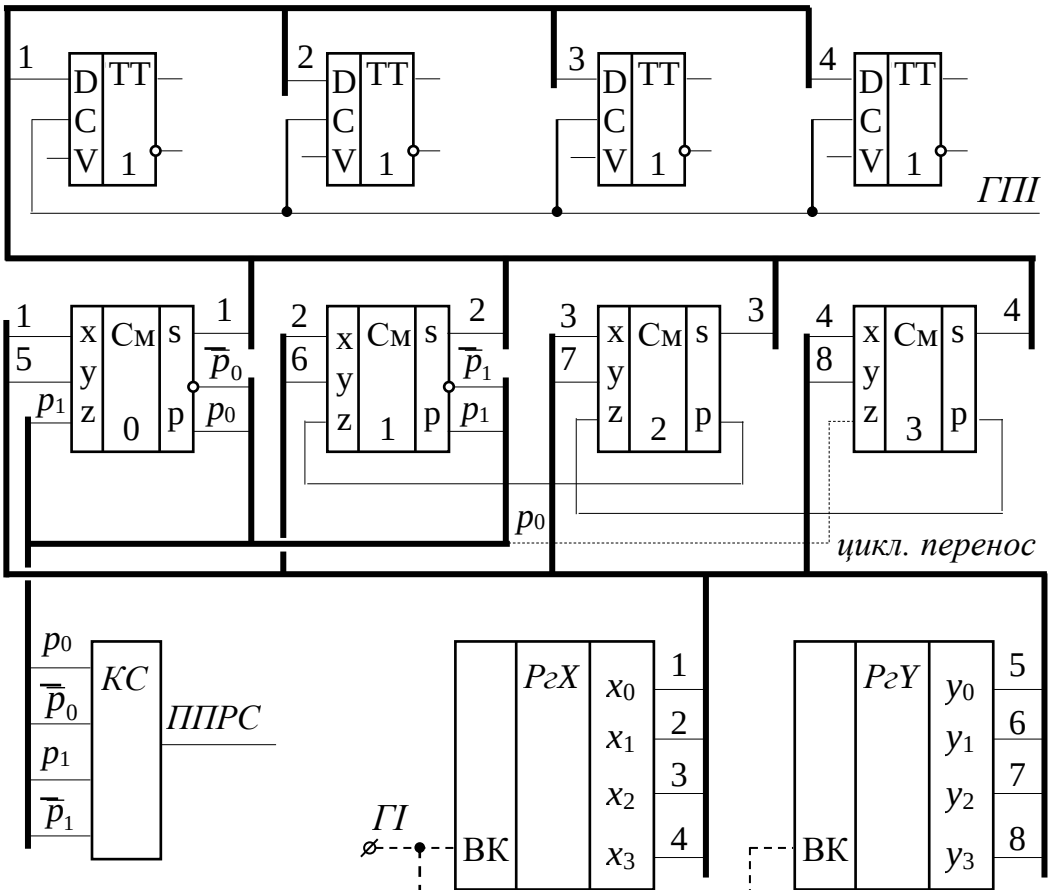
1.5.1 *Перевірка працездатності ЛЕ*, обраних для реалізації схеми (див. п. 1.2.1). Перед набором на стенді необхідної схеми треба переконатися в працездатності обраних елементів за всіма входами і виходами. ЛЕ, всі входи якого не використані (вільні), видає нульовий рівень вихідної напруги, що означає наявність на входах одиничних (неактивних) рівнів сигналу. Таким чином, згідно із законами алгебри логіки, невикористані (зайві) входи *ЛЕ* в набраній схемі не впливають на її роботу.

Перевірка працездатності *ЛЕ* в статичному режимі виконується почерговою подачею на його входи сигналу нульового рівня. При цьому на виході справного елемента кожен раз повинен створюватись одиничний рівень сигналу, який підтверджується світлодіодом що світить. Перевірка *ЛЕ* без світлодіода виконується підключенням його виходу до одного із входів *ЛЕ* зі світлодіодом. Сигнали нульового рівня можна брати з виходу будь-якого невикористаного справного *ЛЕ*.

1.5.2 *Дослідження однорозрядного КСМ (див. п. 1.2.3).* Ввімкнути прямі входи *x*, *y* та *z* однорозрядного суматора до прямих виходів регістра *X*. Подаючи різноманітні набори на вхід суматора, перевірити стан виходів *s* і *p* з його *ТІ*.

1.5.3 *Дослідження схеми багаторозрядного КСМ (див.п.1.2.6).* Зібрати на лабораторному стенді робочу схему чотирирозрядного *КСМ* (рис. 1.2) з послідовною організацією переносів. Перевірити правильність роботи схеми, набравши на регістрах *X* і *Y* відповідно коди 0000 і 1111, після цього – навпаки. Далі перевірити ланцюги розповсюдження міжрозрядних переносів, послідовно набираючи одночасно на обох регістрах *X* і *Y* коди 0001, 0010, 0100, 1000, і ланцюг крізного переносу, установивши на регістрах коди 1111 і 0001 (та навпаки). При помилці в будь-якому тестовому прикладі виявити і усунути несправність в схемі суматора.

1.5.4 *Експериментальне визначення часу підсумовування (див. п. 1.2.9).* На регістрах *X* і *Y* установити коди 1111 і 0001. Початок операції підсумовування ініціюється сигналом видавання кодів *ВК* доданків (імпульс опитування тумблерних регістрів), яким синхронізується осцилограф. На рис. 1.2 лінія *ВК* показана пунктиром. Зафіксувати з екрана осцилографа час від початку розгортки до моменту встановлення остаточної суми (на виході s_0) у старшому розряді суматора.



Час затримки розповсюдження сигналу наскрізного переносу визначити за тих самих кодів як різницю моментів початку розгортки і появи сигналу переносу на виході p_0 старшого розряду суматора.

1.6 Зміст звіту

1. Стислий опис мети роботи і основні теоретичні положення.
2. Принципова схема і таблиця істинності однорозрядного *КСМ*.
3. Математична модель, принципова і функціональна схеми дворозрядного *КСМ* з паралельним переносом.
4. Функціональна схема чотирирозрядного *КСМ* з комбінованим переносом.
5. Робоча схема *КСМ* для дослідження на лабораторному стенді (див. рис. 1.2).
6. Приклади алгебраїчного підсумовування кодів чисел зі знаками.
7. Осцилограми і обчислення з п. 1.2.9.
8. Висновки з результатів роботи.

1.7 Контрольні запитання і завдання

1. Одержати досконалі форми перемикальних функцій суми і переносу *ОКСМ*.
2. Оцінити складність комбінаційних схем за Квайном.
3. Описати *ОКСМ* в різноманітних *ОФ* з урахуванням обмеження за входами логічних операторів.
4. Спроектувати *ОКСМ*, використовуючи винесення логічних змінних за дужки.
5. Позначення *КСМ* на функціональних і структурних схемах.
6. Способи підсумовування багаторозрядних слів; приклади схем суматорів і їх характеристики.
7. Способи організації міжрозрядних переносів в паралельних *КСМ*; функціональна схема *КСМ* з послідовними переносами.
8. Математичний опис паралельної організації міжрозрядних переносів в багаторозрядних *КСМ*.
9. Схема організації паралельних переносів (*СПП*).
10. Функціональна схема трирозрядного паралельного *КСМ* з паралельною організацією міжрозрядних переносів.
11. Суть комбінованого способу організації міжрозрядних переносів.

12. Функціональна схема багаторозрядного *КСМ* з комбінованими переносами.

13. Засоби виявлення переповнення розрядної сітки (*ППРС*) суматора.

14. Коди двійкових чисел і їх характеристики.

1.8 Теоретичні відомості

Однім із вузлів цифрових побудов (*ЦП*), чималою мірою визначаючих швидкість його роботи, є суматор. На сьогодні в *ЦП* найбільш розповсюджені суматори з використанням комбінаційного засобу обробки інформації, яких називають комбінаційними суматорами (*КСМ*). Суматори з послідовним способом обробки інформації, основним елементом яких є тригер з лічильним входом, називають нагромаджуючими.

Двійковим суматором називають схему, яка реалізує алгебраїчне підсумовування двох *n*-розрядних двійкових чисел зі знаками, представленими в зворотному чи додатковому кодах.

Синтез однорозрядного суматора (ОКСМ). Звичайно суматор становить композицію із *n* однорозрядних суматорів. При додаванні двох чисел, незалежно від системи числення, в кожному розряді виробляється додавання трьох цифр: цифри наданого *i*-го розряду першого доданку x_i, цифри *i*-го розряду другого доданку y_i і цифри (0 або 1) переносу z_i із сусіднього молодшого (*i+1*)-го розряду. Внаслідок додавання в кожному розряді отримується цифра суми s_i для цього розряду і цифра (0 або 1) переносу p_i в наступний старший (*i-1*)-ий розряд.

Перемикальні функції s_i і p_i, що реалізуються однорозрядним двійковим суматором згідно з правилами двійкової арифметики, наведені в табл. 1.2.

Таблиця 1.2 – Таблиця істинності *ОКСМ*

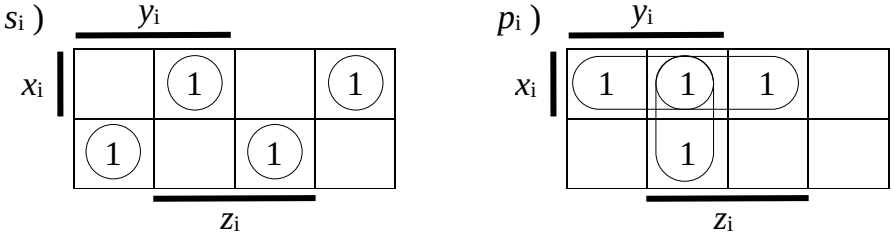
x_i	0	0	0	0	1	1	1	1
y_i	0	0	1	1	0	0	1	1
z_i	0	1	0	1	0	1	0	1
s_i	0	1	1	0	1	0	0	1
p_i	0	0	0	1	0	1	1	1

Досконалі диз'юнктивні нормальні форми (*ДДНФ*) цих функцій, одержані із табл. 1.2 по "одиницях", мають вигляд:

$$s_i = \bar{x}_i \bar{y}_i z_i \vee \bar{x}_i y_i \bar{z}_i \vee x_i \bar{y}_i \bar{z}_i \vee x_i y_i z_i = \vee(1,2,4,7),$$

$$p_i = \bar{x}_i y_i z_i \vee x_i \bar{y}_i z_i \vee x_i y_i \bar{z}_i \vee x_i y_i z_i = \vee(3,5,6,7).$$

Мінімальні диз'юнктивні форми функцій (*МДНФ*) визначимо методом сумісної мінімізації системи функцій із застосуванням діаграм Вейча. В основі методу лежить ідея використання однієї функції чи її частини для одержання інших функцій:



Як видно із діаграм Вейча, сумісна мінімізація наданих функцій неможлива через відсутність у них спільних груп конституєнт 1. *МДНФ* функцій дорівнює:

$$s_i = \bar{x}_i \bar{y}_i z_i \vee \bar{x}_i y_i \bar{z}_i \vee x_i \bar{y}_i \bar{z}_i \vee x_i y_i z_i ,$$
$$p_i = x_i y_i \vee x_i z_i \vee y_i z_i .$$

Для використання *ЛЕ І-НІ* остання система функцій перетворюється в другу операторну форму (*ОФ*):

$$s_i = \overline{\overline{\bar{x}_i \bar{y}_i z_i} \cdot \overline{\bar{x}_i y_i \bar{z}_i} \cdot \overline{x_i \bar{y}_i \bar{z}_i} \cdot \overline{x_i y_i z_i}} ,$$
$$p_i = \overline{\overline{x_i y_i} \cdot \overline{x_i z_i} \cdot \overline{y_i z_i}} .$$

Для оцінки апаратних витрат на реалізацію комбінаційної схеми користуються поняттям її складності за Квайном. Складність (ціна) схеми з Квайну визначається сумарним числом входів *ЛЕ* в складі схеми. За такої оцінки одиниця складності – один вхід *ЛЕ*. Тоді складність схеми *КСМ*, реалізованого за останнім виразом, дорівнює 25 одиницям за Квайном.

Для двовходових логічних операторів (*ЛО*) одержані функції приймають вид (з урахуванням обмеження по входах *ЛО*):

$$s_i = \overline{\overline{\bar{x}_i \bar{y}_i z_i} \cdot \overline{\bar{x}_i y_i \bar{z}_i} \cdot \overline{x_i \bar{y}_i \bar{z}_i} \cdot \overline{x_i y_i z_i}} ,$$
$$p_i = \overline{\overline{x_i y_i} \cdot \overline{x_i z_i} \cdot \overline{y_i z_i}} .$$

Для реалізації схеми слід використати 23 *ЛЕ*. Складність схеми дорівнює 39 одиницям з Квайну (з урахуванням 7 інверторів).

Надто ефективною для мінімізації операторних форм функцій є операція винесення змінних за дужки. Справді,

$$s_i = \bar{x}_i \bar{y}_i z_i \vee \bar{x}_i y_i \bar{z}_i \vee x_i \bar{y}_i \bar{z}_i \vee x_i y_i z_i =$$
$$= (\bar{x}_i \bar{y}_i \vee x_i y_i) z_i \vee (\bar{x}_i y_i \vee x_i \bar{y}_i) \bar{z}_i .$$

Враховуючи, що $\bar{x}_i y_i \vee x_i \bar{y}_i = \overline{\bar{x}_i \bar{y}_i \vee x_i y_i} = \overline{\bar{x}_i \bar{y}_i} \cdot \overline{x_i y_i}$ (одержіть заперечення функції за допомогою діаграми Вейча), функцію суми s_i в базисі "штрих Шефера" можна подати у вигляді:

$$s_i = \overline{\overline{\overline{\bar{x}_i \bar{y}_i} \cdot \overline{x_i y_i} \cdot z_i} \cdot \overline{\overline{\bar{x}_i \bar{y}_i} \cdot \overline{x_i y_i} \cdot \bar{z}_i}} .$$

Застосовуючи винесення за дужки для функції переносу, аналогічно одержуємо

$$p_i = x_i y_i \vee x_i z_i \vee y_i z_i = x_i y_i \vee (x_i \vee y_i) z_i =$$
$$= x_i y_i \vee \overline{\bar{x}_i \bar{y}_i} \cdot z_i = \overline{\overline{x_i y_i \vee \bar{x}_i \bar{y}_i} \cdot z_i} = \overline{\overline{x_i y_i} \cdot \overline{\bar{x}_i \bar{y}_i} \cdot z_i} .$$

Ці вирази реалізуються комбінаційною схемою, що містить вісім двовходових ЛЕ І-НЕ і один інвертор (див. рис. 1.1). Складність такої схеми дорівнює всього 17 одиницям Квайну.

Умовне позначення однорозрядного комбінаційного суматора на функціональних схемах показано на рис. 1.3,а.

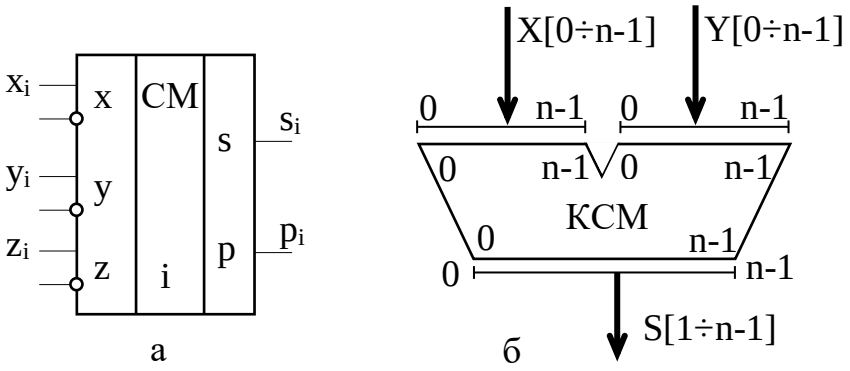


Рис. 1.3 – Функціональна (а) і структурна (б) схеми КСМ

Способи організації підсумовування багаторозрядних слів. Існує три основних способи організації операції підсумовування *n*-розрядних двійкових кодів: паралельний, послідовний і паралельно-послідовний.

В паралельному КСМ підсумовування всіх розрядів числа здійснюється одночасно (рис. 1.4,а). Кількість однорозрядних суматорів в такій схемі дорівнює кількості розрядів чисел, що складаються. На вхід z_{n-1} при підсумовуванні в додатковому коді подається нуль, а при підсумовуванні в зворотному коді для створювання циклічного переносу вхід z_{n-1} підключається до виходу p_0 цього ж суматора. Позначення паралельного *n*-розрядного суматора на структурних схемах показано на рис. 1.3,б.

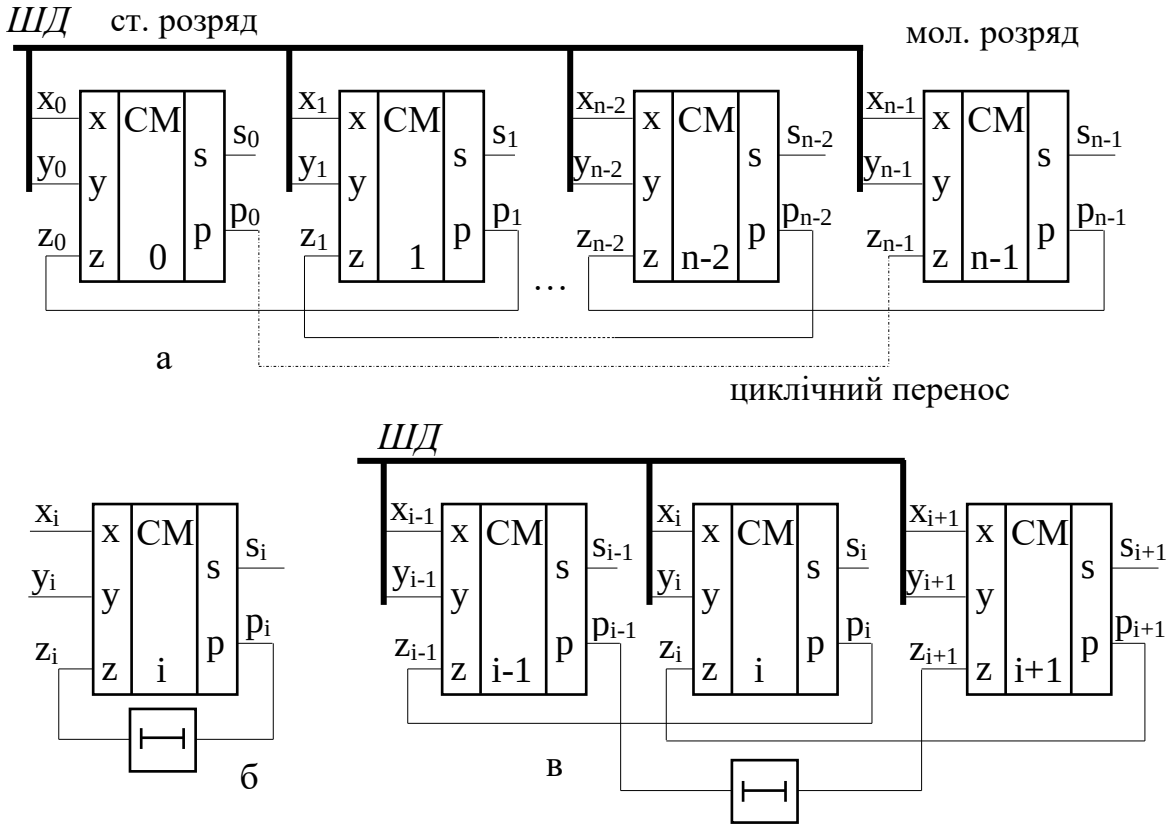


Рис. 1.4 – Паралельний (а), послідовний (б)
і паралельно-послідовний (в) суматори

Послідовний суматор будується за схемою рис. 1.4,б. Порозрядні значення доданків x_i та y_i надходять на входи суматора по черзі – спочатку молодший (n-1)-й розряд, після цього наступний по вазі (n-2)-й розряд і т.д. Значення переносу p_i, що з'являється у наданому такті, за допомогою елемента затримки запам'ятовується на час одного такту і на початку наступного такту подається на вхід z одночасно з подачею x_{i-1}, y_{i-1}.

Побудова паралельно-послідовного суматора ілюструється схемою на рис. 1.4,в. В цьому випадку n-розрядні слова X і Y розбиваються на декілька підслів з m розрядів (на рис.1.4,в $m = 3$). Підслова подаються на суматор послідовно, проте підсумовування кожного з них відбувається паралельно.

Організація переносів в паралельних суматорах. За характером розповсюдження переносів розрізняють три виду паралельних суматорів: з послідовними переносами, з паралельними переносами, з паралельно-послідовними (комбінованими) переносами.

Суматори з послідовними переносами (рис. 1.4,а) передають перенос послідовно від молодшого розряду до старшого в міру його утворення у кожному окремому розряді. Тому час розповсюдження переносу в них $T_n = L \cdot \tau_n$, де L – максимальна довжина переносу, що визначається паралельним числом розрядів, крізь які відбувається перенос; τ_n – затримка переносу в одному розряді суматора. При найбільш несприятливих умовах для розповсюдження переносу, наприклад, при додаванні кодів 11...11 і 00...01, одиниця переносу буде пробігати крізь весь суматор від молодшого розряду до старшого. Оскільки в наданому суматорі фактична довжина переносу для кожної конкретної нагоди додавання не визначається, зате для операції повинен виділятися проміжок часу, який відповідає максимальній довжині переносу, тобто $L = n$, де n – кількість розрядів *КСМ*.

Наданий тип суматора вимагає мінімальних апаратних витрат на організацію переносів, але має низьку швидкодію.

Суматори з паралельними переносами формують значення переносу в кожному розряді по значеннях всіх молодших розрядів доданків, включаючи наданий. Переноси в кожному розряді такого суматора відбуваються одночасно і незалежно один від одного додатковими схемами, що значно підвищує їх швидкодію. Будуються *КСМ* з паралельними переносами таким чином. Маючи на увазі, що $z_i = p_{i+1}$, функцію переносу можна представити у вигляді

$$p_i = x_i y_i \vee x_i z_i \vee y_i z_i = x_i y_i \vee (x_i \vee y_i) z_i = \alpha_i \vee \beta_i z_i = \alpha_i \vee \beta_i p_{i+1},$$

де $\alpha_i = x_i \cdot y_i$ – умова виникнення переносу в розряді i;

$\beta_i = x_i \vee y_i$ – умова проходження переносу із молодшого розряду крізь наданий розряд.

Тоді, узагальнено позначивши перенос в молодший розряд p, функції переносів трирозрядного суматора можна описати так:

$$\begin{cases} p_2 = \alpha_2 \vee \beta_2 p\,; \\ p_1 = \alpha_1 \vee \beta_1 p_2\,; \\ p_0 = \alpha_0 \vee \beta_0 p_1\,. \end{cases} \qquad (1.1)$$

Вираз (1.1) відбиває послідовне проходження переносу від молодших розрядів до старших, тобто описують суматор з послідовною організацією переносів.

Підставляючи функцію p_2 в p_1, після цього знов утворену функцію p_1 в p_0, одержуємо наступне подання отих самих функцій, що є математичною моделлю суматора з паралельною організацією переносів:

$$\begin{cases} p_2 = \alpha_2 \vee \beta_2 p\,; \\ p_1 = \alpha_1 \vee \beta_1 (\alpha_2 \vee \beta_2 p) = \alpha_1 \vee \alpha_2 \beta_1 \vee \beta_2 \beta_1 p\,; \\ p_0 = \alpha_0 \vee \beta_0 (\alpha_1 \vee \alpha_2 \beta_1 \vee \beta_2 \beta_1 p) = \alpha_0 \vee \alpha_1 \beta_0 \vee \alpha_2 \beta_1 \beta_0 \vee \beta_2 \beta_1 \beta_0 p\,. \end{cases}$$

Тут значення переносів утворяться на основі аналізу всіх молодших розрядів доданків – паралельно. Час обчислювання суми в таких суматорах не залежить від числа їх розрядів.

Як приклад розглянемо математичну модель трирозрядної ($n = 3$) схеми організації паралельних переносів (*СПП*), що описується такими функціями переносу:

$$\begin{cases} p_2 = \alpha_2 \text{ (} \beta 2 \text{ p ;} \\ \text{p1 = } \alpha 1 \text{ (} \alpha 2 \text{ } \beta 1 \text{ (} \beta 2 \text{ } \beta 1 \text{ p ;} \\ \text{p0 = } \alpha 0 \text{ (}\square\alpha 1 \text{ } \beta 0 \text{ (} \alpha 2 \text{ } \beta 1 \text{ } \beta 0 \text{ (} \beta 2 \text{ } \beta 1 \text{ } \beta 0 \text{ p ;} \\ \square \text{ EMBED Equation.3 } \square\square\square \\ \square \text{ EMBED Equation.3 } \square \alpha_1 = x_1 y_1\,, \quad \beta_1 = x_1 \vee y_1\,; \\ \alpha_0 = x_0 y_0\,, \quad \beta_0 = x_0 \vee y_0\,, \end{cases}$$

де p – узагальнене позначення переносу в молодший розряд суматора.

Перетворюючи одержані вирази в другу *ОФ*, одержуємо:

$$\begin{cases} p_2 = \overline{\overline{\alpha}_2 \cdot \overline{\beta_2 p}}\,; \\ p_1 = \overline{\overline{\alpha}_1 \cdot \overline{\alpha_2 \beta_1} \cdot \overline{\beta_2 \beta_1 p}}\,; \\ p_0 = \overline{\overline{\alpha}_0 \cdot \overline{\alpha_1 \beta_0} \cdot \overline{\alpha_2 \beta_1 \beta_0} \cdot \overline{\beta_2 \beta_1 \beta_0 p}}\,; \\ \alpha_2 = \overline{\overline{x_2 y_2}}\,, \quad \beta_2 = \overline{\overline{x}_2 \overline{y}_2}\,; \\ \alpha_1 = \overline{\overline{x_1 y_1}}\,, \quad \beta_1 = \overline{\overline{x}_1 \overline{y}_1}\,; \\ \alpha_0 = \overline{\overline{x_0 y_0}}\,, \quad \beta_0 = \overline{\overline{x}_0 \overline{y}_0}\,. \end{cases} \qquad (1.2)$$

Очевидно, що умови α_1 і α_2 в базисі “штрих Шефера” реалізуються дещо складніше, ніж в першій *ОФ* (має бути додатковий інвертор), що, проте, виправдано, бо при цьому попутно отримуються заперечення зазначених функцій, необхідні надалі. Умова виникнення переносу із старшого розряду реалізується одним *ЛЕ І-НІ*, оскільки вона використовується далі тільки в інверсному вигляді в функції p_0.

Принципова схема організації паралельних переносів (*СПП*), реалізована за виразом (1.2), показана на рис. 1.5,а, на рис. 1.5,б *СПП* позначена у вигляді функціонального елемента (*ФЕ*), у якого вхід з позначкою *D* в додатковому полі – вхід шини наданих (розряди доданків *X* та *Y*).

Функціональну схему трирозрядного *КСМ* з паралельними переносами зображено на рис. 1.6. При алгебраїчному додаванні у зворотному коді циклічний перенос утвориться подачею на вхід *p СПП* значення переносу із старшого розряду p_0 (показаний пунктиром). Якщо дії виконуються у додатковому коді, то на вхід *p СПП* подається нульовий рівень. Схеми безпосередньо *ОКСМ* тут спрощені, бо реалізують тільки функції суми s_i.

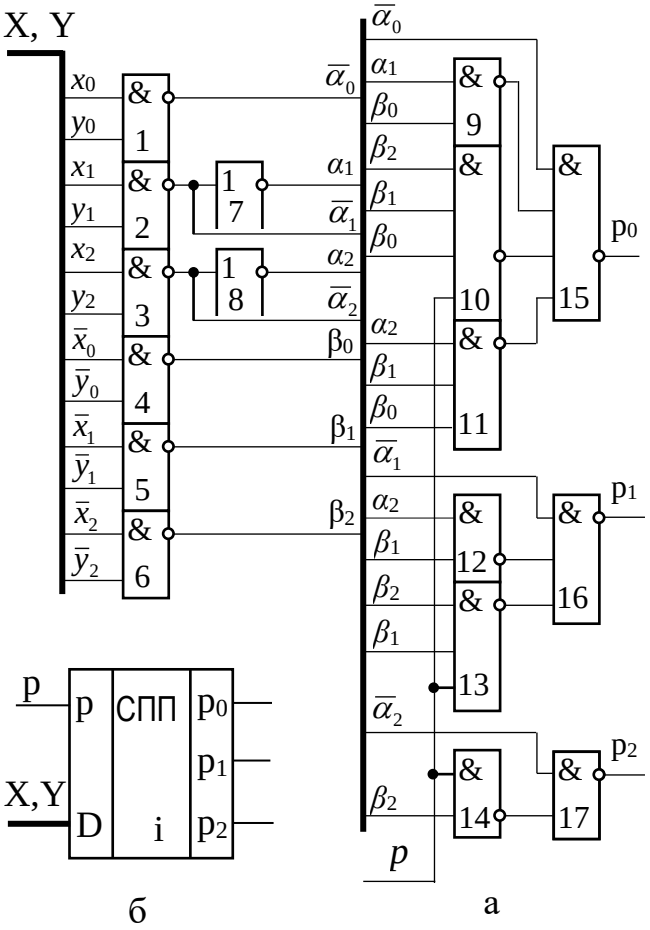


Рис. 1.5 – *СПП* для трирозрядного *КСМ* (а)
та її позначення у вигляді *ФЕ* (б)

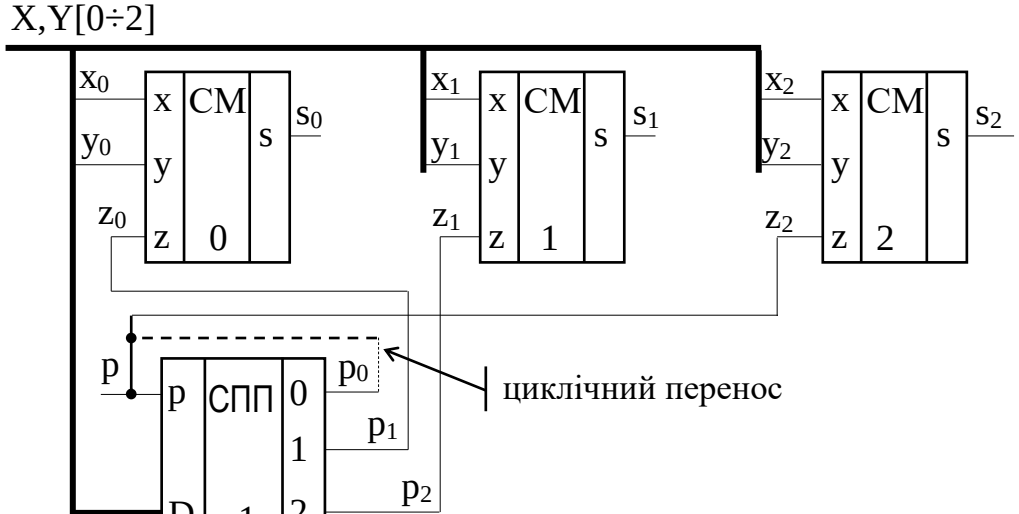


Рис. 1.6 – Функціональна схема *КСМ* із паралельними переносами

Складність схеми паралельного *КСМ* з паралельними переносами зростає зі збільшенням розрядності настільки швидко, що у чистому вигляді він практично не застосовується. шість

Суматори із паралельно-послідовними (комбінованими) переносами об’єднують в собі переваги двох попередніх типів суматорів, забезпечуючи достатньо високу швидкодію при помірних апаратних витратах. При цьому *n*-розрядний суматор поділяється на $k = n/m$ груп *m*-розрядних суматорів із паралельним переносом. Схема шістирозрядного суматора для $m = 3$ показана на рис. 1.7. Переноси всередині груп здійснюються паралельно, а між групами – послідовно. Максимальна довжина переносу $L = k$, час його розповсюдження $T_n = k\ \tau_n$. Таким чином, принцип паралельно-послідовної організації переносів дозволяє зменшити час їх формування в *m* раз у порівнянні з послідовним способом.

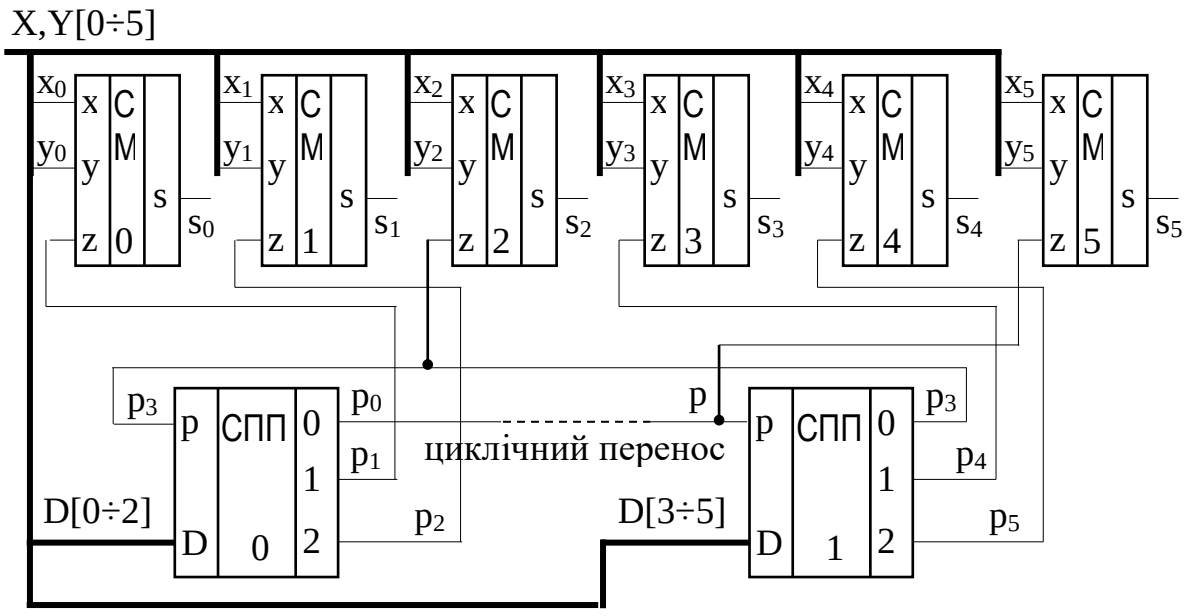


Рис. 1.7 – Функціональна схема КСМ із комбінованими переносами

Суматори з пам'яттю. При підсумовуванні обидва операнда подаються на входи *КСМ* одночасно (паралельно). Крім того *КСМ* дозволяє одночасно з підсумовуванням просто реалізувати додатковий код одного із операндів, якщо на вхід *КСМ* поданий його зворотний код. Для цього під час підсумовування значення переносу в молодший розряд z_{n-1} встановлюється таким, що дорівнює одиниці. Ці якості визначили переважне використання

комбінаційних *СМ* в сучасних цифрових побудовах на *ЛЕ* потенційного типу, витиснувши суматори на тригерах.

Проте *КСМ* не володіють пам'яттю, тобто після зняття вхідних даних сигнали суми і переносу на його виходах також знімаються. Для термінового запам'ятовування результату підсумовування використовується спеціальний регістр зберігання, ввімкнений до виходу *КСМ* (рис. 1.8). Запис результату в *Рг3* відбувається тільки при наявності керуючого сигналу прийому в регістр зберігання *Прг3*.

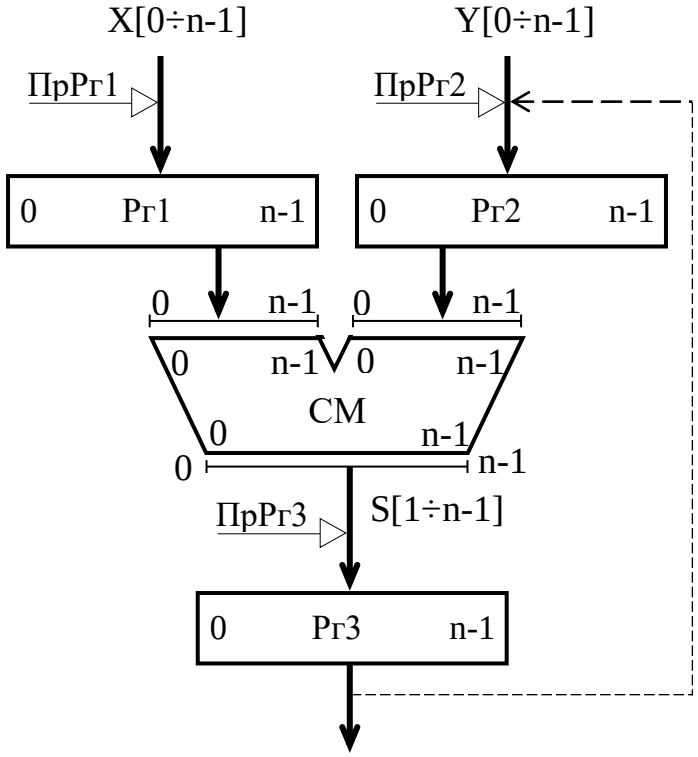


Рис. 1.8 – Структурна схема *КСМ* з пам'яттю

Якщо вихід *Рг3* підключити до входу *Рг2* (пунктирна лінія на рис. 1.8), то на базі *КСМ* можна одержати нагромаджуючий *СМ*. В цьому випадку вхідні дані послідовно подаються на *Рг1* та складаються з частковими сумами, що створюються на *Рг3*.

Засоби виявлення переповнення суматорів. При використанні зворотного і додаткового кодів алгебраїчне додавання чисел з різними знаками замінюється арифметичним додаванням кодів (включаючи розряди знаків), при цьому автоматично отримується код знаку результату. Знакові розряди кодів розглядаються як старші розряди чисел. Перенос із знакового розряду відкидається, якщо числа представляються в додатковому коді, чи виробляється циклічний перенос в молодший розряд, якщо числа зображаються зворотним кодом.

Проте при алгебраїчному додаванні двох чисел, що поміщуються в розрядну сітку суматора, може виникнути його переповнення (*ППРС*), тобто утворюється сума, яка вимагає для свого подання на один цифровий розряд більше наявного числа розрядів *СМ*.

Для виявлення *ППРС* використовують два основних способи, що відрізняються складом аналізованих змінних.

У першому способі аналізуються значення переносів в знаковий та із знакового розряду при виконанні підсумовування кодів.

Ознакою *ППРС* суматора є наявність переносу у знаковий розряд суми при одночасній відсутності переносу із її знакового розряду (позитивне переповнення) або наявність переносу із знакового розряду суми за відсутністю переносу у її знаковий розряд (від'ємне переповнення). Якщо і у знаковий і із знакового розрядів суми є переноси чи обох цих переносів немає, то переповнення відсутнє. Позначимо перенос із знакового розряду p_0, а перенос у знаковий розряд із старшого цифрового – $p_1 = z_0$. Тоді функція переповнення розрядної сітки суматора дорівнює

$$ППРС = p_0 \bar{p}_1 \vee \bar{p}_0 p_1 .$$

В другому способі для подання знаку коду використовують два розряди, аналіз значень яких дозволяє зробити висновок про *ППРС* суматора.

Коди чисел в цьому випадку називають модифікованими. Додатний знак коду представляється двома нулями – 00, а від'ємний знак – двома одиницями – 11. Ознакою переповнення є незбіг цифр у знакових розрядах суми. Набору 01 у знакових розрядах відповідає додатне переповнення, а набору 10 – від'ємне. Таким чином, функція переповнення

$$ППРС = s_0 \bar{s}_1 \vee \bar{s}_0 s_1 ,$$

де $\tilde{s}_0 , \tilde{s}_1$ – знакові розряди модифікованого коду суми (у прямому чи інверсному значенні).

[3; 9-12]

Лабораторна робота № 4

СИНТЕЗ ТА ДОСЛІДЖЕННЯ ЦИФРОВИХ СХЕМ ПОРІВНЯННЯ

Мета лабораторної роботи:

– вивчити принципи роботи та структури комбінаційних схем порівняння (*КСП*) на рівність, більше, менше слів із константами, абсолютних величин слів та слів зі знаками;

– з'ясувати етапи проектування різноманітних *КСП* з заданими технічними характеристиками;

– реалізувати та експериментально дослідити *КСП* слова із константами, повну *КСП* слів та схему порівняння на базі комбінаційних суматорів (*КСМ*).

1 Домашнє завдання

Вивчити:

– методику синтезу *КСП* слів із константами, що реалізують різноманітні логічні умови (рівність, більше, менше);

– методику проектування багаторозрядних неповних *КСП* слів на рівність-нерівність, більше-менше;

– методику синтезу повної *КСП* слів з вихідним дешифратором умов;

– спосіб побудови *КСП* на базі *КСМ*;

– організацію порівняння слів зі знаками та структуру відповідної *КСП*;

– ознайомитися з методикою проведення лабораторної роботи на універсальному лабораторному стенді.

2 Лабораторне завдання

2.2.1 Розробити принципову *КСП* із константою для заданої логічної умови (табл. 2.1), реалізувати схему на *ЛЕ І-НІ* та дослідити її роботу.

2.2.2 Розробити принципову схему для повного порівняння дворозрядних слів згідно з кодами сигналів переносу (*СП*), заданих табл. 2.1. Для одержання мінімальної системи функцій переносу $p_i^{'}$ та $p_i^{''}$ використати операцію винесення змінних $z_i^{'}$ та $z_i^{''}$ за дужки. Мінімізувати структуру молодшої *ОКСП*. Провести синтез неповного вихідного дешифратора умов.

2.2.3 Реалізувати *КСП* повного порівняння дворозрядних слів за 2-ю *ОФ* функцій, одержаних у п. 2.2.2. Дослідити її роботу в заздалегідь розробленій таблиці для аналізу роботи *КСП* (згідно з табл. 2.7 для 1-го варіанту завдань).

2.2.4 Розробити за заданим варіантом функціональну схему неповного порівняння чотирирозрядних слів на базі *КСМ*. Реалізувати схему на стенді та дослідити її роботу.

Таблиця 2.1 – Таблиця варіантів

Номер варіанта	*КСП* слів с константою		Коди сигналів переносу повної *КСП*			Порівняння слів на *КСМ*	
	ЛУ	$К_{(10)}$	$x_i = y_i$	$x_i < y_i$	$x_i > y_i$	*ЛУ*	p_0
1.	X=K	12	00	01	10	X<Y	0
2.	X≠K	12	00	10	01	X<Y	1
3.	X<K	10	01	00	10	X≤Y	0
4.	X≤K	10	01	10	00	X≤Y	1
5.	X>K	11	10	00	01	X>Y	0
6.	X≥K	11	10	01	00	X>Y	1
7.	X=K	13	00	01	11	X≥Y	0
8.	X≠K	13	00	11	01	X≥Y	1
9.	X<K	11	01	00	11	X<Y	0
10.	X≤K	11	01	11	00	X<Y	1
11.	X>K	12	11	00	01	X≤Y	0
12.	X≥K	12	11	01	00	X≤Y	1
13.	X=K	14	00	10	11	X>Y	0
14.	X≠K	14	00	11	10	X>Y	1
15.	X<K	12	10	00	11	X≥Y	0
16.	X≤K	12	10	11	00	X≥Y	1
17.	X>K	13	11	00	10	X<Y	0
18.	X≥K	13	11	10	00	X<Y	1
19.	X=K	11	01	10	11	X≤Y	0
20.	X≠K	11	01	11	10	X≤Y	1
21.	X<K	8	10	01	11	X>Y	0
22.	X≤K	8	10	11	01	X>Y	1
23.	X>K	9	11	01	10	X≥Y	0
24.	X≥K	9	11	10	01	X≥Y	1
25.	X=K	10	01	00	11	X<Y	0
26.	X≠K	10	01	10	00	X<Y	1
27.	X<K	13	10	01	00	X≤Y	0
28.	X≤K	14	10	00	01	X≤Y	1
29.	X>K	14	11	01	10	X>Y	0
30.	X≥K	13	11	10	01	X>Y	1

2.2.5 З метою виключення помилок у наборі схем, що підлягають реалізації, логічні елементи повинні бути визначені номерами обраних

елементів лабораторного стенда. Коди порівняльних слів набирати та подавати з виходів тумблерних регістрів стенда.

В разі виявлення збоїв у роботі будь-який з реалізованих схем проаналізувати та усунути причини несправностей (наприклад, помилки на етапі проектування чи при наборі схеми, несправність елементів стенду, прихований обрив комутуючих провідників).

2.3 Прилади, використані у роботі

Універсальний лабораторний стенд (*УЛС*), опис якого наведено у п. 1.4 лабораторної роботи № 1.

2.4 Зміст звіту

1. Стислий опис мети роботи та основних теоретичних положень.
2. Етапи синтезу та принципова схема *КСП* слів з константою (за заданим варіантом).
3. Результати проектування повної *КСП* дворозрядних слів (*ТКСП, ТІ ОКСП*, діаграми Вейча та *МДНФ* функцій *ОКСП*, функції переносу з молодшого розряду, *ТІ* і *МДНФ* функцій вихідного дешифратора, загальна система функцій схеми порівняння в 2-ій *ОФ*).
4. Принципова схема *КСП* дворозрядних слів на *ЛЕ І-НІ*.
5. Таблиця для аналізу роботи *КСП* за заданим варіантом.
6. Функціональна схема *КСП* чотирирозрядних слів на базі *КСМ* з підключенням тумблерних регістрів.
7. Висновки за результатами роботи.

2.5 Контрольні запитання та завдання

1. Математичний опис повних і неповних *КСП* слів у загальному випадку.
2. Порівняння слів із константами на рівність-нерівність. Наведіть приклади *КСП* на рівність та нерівність для $K = 7_{(10)}$; $K = 13_{(10)}$.
3. Як спроектувати *КСП* слів із константами на більше-менше? Наведіть приклади *КСП* для $X \le 7_{(10)}$; $X < 8_{(10)}$; $X > 11_{(10)}$; $X \ge 10_{(10)}$.
4. Розробіть *КСП* слів з константами для логічної умови $2_{(10)} \le X \le 12_{(10)}$.
5. Розробіть *КСП* слів з константами, що реалізує логічну умову $11_{(10)} < X \le 3_{(10)}$.
6. Порівняння слів на рівність-нерівність. Наведіть приклад дворозрядної *КСП*, збудованої за 7-ою операторною формою функції.
7. Доведіть тотожність виразів (2.9) та (2.10).

8. Перетворіть вираз (2.11) у 7-у *ОФ* та розробіть однорозрядну *КСП* (*ОКСП*) на логічних елементах *І-АБО-НІ*.

9. Спроектуйте неповну *ОКСП* за 2-ою *ОФ* функції для виконання логічної умови $X \leq Y$.

10. Спроектуйте неповну *ОКСП* за 2-ою *ОФ* функції для виконання логічної умови $X > Y$.

11. Спроектуйте неповну *ОКСП* за 2-ою *ОФ* функції для виконання логічної умови $X \geq Y$.

12. Спростіть функцію переносу із молодшого розряду неповної *ОКСП* слів на менше, якщо умова $X < Y$ кодується нулем.

13. Поясніть методику проектування повної *ОКСП* слів.

14. Розробіть дворозрядні *КСП* слів на базі *КСМ* для різноманітних варіантів кодування логічної умови X>Y.

15. Спростіть вираз (2.16), використовуючи закони алгебри логіки.

16. Розробіть принципову схему *СПЗ* (рис. 2.11) за 2-ою *ОФ* виразу (2.17).

17. Структура та робота повної *КСП* слів зі знаками.

2.6 Варіанти завдань

В табл. 2.1 наведені варіанти кодування логічних умов сигналами переносу (*СП*) для повної дворозрядної *КСП* слів ($p_i^{'}$ та $p_i^{''}$) і чотирирозрядної *КСП* на базі *КСМ* (p_0). Виконання логічних умов у *КСП* слів з константами кодується одиницею.

2.7 Теоретичні відомості

В цифрових пристроях (*ЦП*) використовується декілька різновидів операції порівняння слів: за модулем, з урахуванням знаків операндів, порівняння порядків слів, заданих у напівлогарифмічній формі. Тут під словом слід розуміти групу двійкових розрядів, що спільно піддаються обробці при арифметичній чи логічній операції (у окремому випадку, багаторозрядне двійкове число). Найбільш повною є операція порівняння, за якою встановлюється факт виконання однієї із умов $X = Y$, $X < Y$, $X > Y$, де X, Y – цілі чи дробові двійкові числа зі знаками.

Найбільш розповсюджені два способи реалізації операції порівняння. Перший полягає в тому, що із одного слова віднімають друге та за знаком і різницею судять про виконання наведених умов. Реалізується така операція на суматорі, до якого додатково підключається комбінаційна схема для фіксації нульового результату.

З метою скорочування часу реалізації операції порівняння та розвантаження суматора *ЦП* для виконання інших операцій

використовується інший спосіб виконання порівняння слів – побудова спеціальної комбінаційної схеми порівняння (*КСП*).

Повною схемою порівняння *n*-розрядних слів у загальному випадку називають схему з (2*n*+2) входами та трьома виходами, що реалізує такі перемикальні функції:

$$f_{x=y} = \begin{cases} 1(0) & \text{при } X=Y, \\ 0(1) & \text{при } X\neq Y; \end{cases}$$
$$f_{x<y} = \begin{cases} 1(0) & \text{при } X<Y, \\ 0(1) & \text{при } X\geq Y; \end{cases} \qquad (2.1)$$
$$f_{x>y} = \begin{cases} 1(0) & \text{при } X>Y, \\ 0(1) & \text{при } X\leq Y. \end{cases}$$

Цифри у дужках означають, що виконання відповідної логічної умови може кодуватися одиницею чи нулем згідно з вимогами завдання, що розв'язується.

Кожну функцію у повній схемі порівняння можна реалізувати незалежно. Однак, якщо схеми для $f_{x=y}$ та $f_{x<y}$ вже збудовані, то функцію $f_{x>y}$ простіше реалізувати за формулою

$$f_{x>y} = \overline{f_{x=y}} \cdot \overline{f_{x<y}} = \overline{f_{x=y} \vee f_{x<y}}.$$

Аналогічно можна одержати значення будь-якої із функцій, виконаних схемою порівняння, при відомих двох інших:

$$f_{x=y} = \overline{f_{x<y}} \cdot \overline{f_{x>y}} = \overline{f_{x<y} \vee f_{x>y}},$$
$$f_{x<y} = \overline{f_{x=y}} \cdot \overline{f_{x>y}} = \overline{f_{x=y} \vee f_{x>y}}.$$

У окремому випадку схема порівняння може бути неповною, тобто реалізувати тільки одне чи два із наведених умов.

Порівняння слів з константами

Порівняння на рівність. Нехай $X = x_0\ x_1 \ldots x_{n-1}$ та $K = k_0 k_1 \ldots k_{n-1}$ відповідно *n*-розрядні слово та константа, що підлягають порівнянню. Необхідно спроектувати *КСП*, що реалізує деяку перемикальну функцію $f_{x=k}$, що набирає значення 1 при рівності слова та константи і значення 0 при їх нерівності:

$$f_{x=k} = \begin{cases} 1 & \text{при } X=K, \\ 0 & \text{при } X\neq K. \end{cases} \qquad (2.2)$$

З умови завдання випливає, що *КСП* повинна описуватися конституєнтою одиниці, що відповідає двійковому набору, який визначається константою *K* (за визначенням, тільки на цьому наборі конституєнта одиниці буде дорівнювати 1). Отже, $f_{x=k} = \tilde{x}_0 \tilde{x}_1 \ldots \tilde{x}_{n-1}$, де $\tilde{x}_i = 1$, якщо $k_i = 1$, та $\tilde{x}_i = 0$, якщо $k_i = 0$.

Наприклад, при порівнянні на рівність константам $K_0 = 0000_{(2)}$ та $K_{10}=1010_{(2)}$ відповідають такі функції: $f_{x=0} = \overline{x}_0\overline{x}_1\overline{x}_2\overline{x}_3$ та $f_{x=10} = x_0\overline{x}_1x_2\overline{x}_3$, за якими і реалізуються необхідні схеми порівняння.

Порівняння на нерівність. Якщо умова рівності кодується нулем (а нерівності – одиницею), то *КСП* описується перемикальною функцією

$$f_{x\neq k} = \begin{cases} 0 & \text{при } X=K, \\ 1 & \text{при } X\neq K, \end{cases} \qquad (2.3)$$

із якої випливає, що надана функція дорівнює нулю на одному єдиному наборі змінних x_i ($i = 0, 1, \ldots, n\text{-}1$) та одиниці – на усіх інших наборах.

Така функція уявляє собою конституєнту нуля, що відповідає двійковому набору, який задається константою K, тобто $f_{x\neq k} = \tilde{x}_0 \vee \tilde{x}_1 \vee \ldots \vee \tilde{x}_{n-1}$, де $\tilde{x}_i = 1$, коли $k_i = 0$ та $\tilde{x}_i = 0$, коли $k_i = 1$.

Наприклад, тим самим константам $K_0 = 0000_{(2)}$ та $K_{10} = 1010_{(2)}$ при кодуванні умови рівності нулем відповідають перемикальні функції, за якими реалізовані *КСП* на рис. 2.1:

$$f_{x\neq 0} = x_0 \vee x_1 \vee x_2 \vee x_3 = \overline{\overline{x}_0\overline{x}_1\overline{x}_2\overline{x}_3};$$

$$f_{x\neq 10} = \overline{x}_0 \vee x_1 \vee \overline{x}_2 \vee x_3 = \overline{x_0\overline{x}_1x_2\overline{x}_3}.$$

Розглянуті *КСП* можна інтерпретувати як неповні дешифратори з одним виходом. Такі схеми широко застосовують у мікропроцесорних пристроях у вигляді дешифраторів адрес зовнішніх пристроїв (*ЗП*). Кожному *ЗП* привласнюється конкретний код адреси із адресного простору мікропроцесора (*МП*). Пристрій включається у роботу тільки в тому випадку, якщо на його спеціальному розв'язуючому вході вибірки кристалу (*ВК*), ввімкнутому до виходу дешифратора адреси, встановлюється активний нульовий рівень сигналу. Дешифратор адреси, входи якого з'єднані з адресною шиною *МП*, безупинно декодує адреси, що генеруються МП, та, при впізнанні коду "своєї" адреси, видає нульовий рівень вихідного сигналу, активізуючи тим самим *ЗП*.

Рис. 2.1 – *КСП* з константами на нерівність

Інколи необхідно порівняти на рівність слово з деякими константами, тобто реалізувати наступну перемикальну функцію

$$f_{x=k} = \begin{cases} 1 & \text{при } X = K_0 \vee K_1 \vee \ldots \vee K_m, \\ 0 & \text{при } X \neq K_0 \vee K_1 \vee \ldots \vee K_m, \end{cases} \qquad (2.4)$$
$$m = 0, 1, \ldots .$$

Очевидно, що досконала диз'юнктивна нормальна форма (*ДДНФ*) функції в даному випадку являє собою диз'юнкцію конституєнт одиниці, відповідних наборам K_0, K_1, ..., K_m.. Ця функція мінімізується відомими засобами.

Наприклад, логічна умова $X = K_8 \vee K_9 \vee K_{12} \vee K_{13}$, виконання якої кодується одиницею, описується *ДДНФ* функції $f_{x=k} = \vee(8, 9, 12, 13)$. Мінімальна форма (*МДНФ*), одержана за допомогою діаграми Вейча чотирьох змінних, $f_{x=k} = x_0\overline{x}_2$.

Порівняння з константами на більше-менше. Нехай необхідно спроектувати *КСП*, що виявляє вхідні слова, величина яких менша за наперед задане число (набору). Таку схему описують такою перемикальною функцією:

$$f_{x<k} = \begin{cases} 1 & \text{при } X < K_m, \\ 0 & \text{при } X \geq K_m, \end{cases}$$
$$m = 0, 1, \ldots, 2^n\text{-}1,$$

де m – десятковий номер набору; n – число розрядів у слові. Нагадаємо, що число різноманітних наборів від n змінних дорівнює 2^n.

Логічна умова $X < K_m$ означає, що функція $f_{x<k}$ набирає одиничного значення на вхідних наборах, що дорівнюють константам K_0, K_1,..., K_{m-1}, та нульового значення на всіх інших наборах, що дорівнюють константам K_m, K_{m+1}, ..., K^{2^n-1}. Отже, у досконалій формі функція являє собою диз'юнкцію конституєнт 1, відповідних двійковим наборам констант K_0, K_1,..., K_{m-1}.

Наприклад, логічній умові $X < K_{12}$ ($K_{12} = 1100_{(2)}$) у наданому завданні відповідає *ДДНФ* функції $f_{x<12} = \vee(0, 1, \ldots, 9, 10, 11)$. Використовуючи діаграму Вейча чотирьох змінних (рис. 2.2,а), отримуємо *МДНФ* цієї функції $f_{x<12} = \overline{x}_0 \vee \overline{x}_1 = \overline{x_0 x_1}$, що реалізується одним логічним елементом (*ЛЕ*) *І-НІ* (рис. 2.2,б).

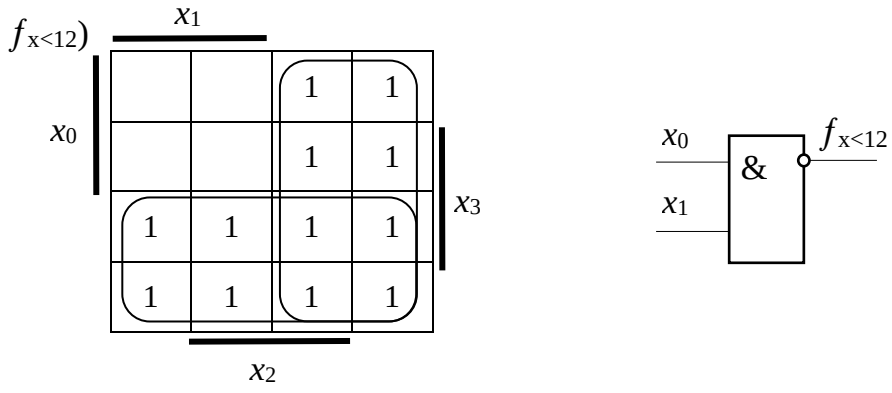


Рис. 2.2 – Синтез *КСП* з константою на менше

КСП, що реалізують логічні умови $X \le K_m$, $X > K_m$, $X \ge K_m$, проектуються аналогічно. Помітимо тільки, якщо нерівність нежорстка ($X \le K_m$ або $X \ge K_m$), то *ДДНФ* функцій включають до себе і конституєнту одиниці, відповідну набору K_m. Наприклад, логічній умові $X \le K_{12}$ відповідає функція досконалої форми

$$f_{x<12} = \vee (0, 1, ..., 9, 10, 11, 12).$$

Порівняння слів на рівність і нерівність

Комбінаційну схему, що реалізує перемикальну функцію вигляду

$$f_{x=y} = \begin{cases} 1 & \text{при } X = Y, \\ 0 & \text{при } X \ne Y, \end{cases} \qquad (2.6)$$

називають схемою порівняння багаторозрядних слів X та Y на рівність (схемою рівності слів), що проектується таким чином. Багаторозрядні слова X та Y рівні тільки в тому випадку, якщо одночасно будуть дорівнювати всі їх однойменні розряди x_i та y_i ($i = 0, 1, ..., n\text{-}1$), кожний з яких порівнюється незалежно від інших. Тому достатньо провести синтез однієї однорозрядної *КСП* (*ОКСП*), щоб розробити структуру схеми рівності слів довільної розрядності, що є однорідною. Робота *ОКСП* описується таблицею істинності (див. табл. 2.2), з якої отримуємо функцію рівності (еквівалентності) однойменних розрядів x_i і y_i у 1-ій та 2-ій *ОФ*:

$$r_i = x_i \equiv y_i = \bar{x}_i \bar{y}_i \vee x_i y_i = \overline{\overline{\bar{x}_i \bar{y}_i} \cdot \overline{x_i y_i}}\ .$$

Функцію $f_{x=y}$ рівності багаторозрядних слів X та Y одержимо з обліком того, що останнє можливо, якщо дотримується рівність всіх однойменних розрядів слів, тобто $f_{x=y} = r_0 \cdot r_1 \cdot ... \cdot r_{n\text{-}1}$.

Перемикальна функція

$$f_{x\ne y} = \begin{cases} 0 & \text{при } X = Y \\ 1 & \text{при } X \ne Y \end{cases} \qquad (2.7)$$

у якій умова рівності кодується нулем, описує КСП багаторозрядних слів на нерівність. Функція нерівності g_i однойменних розрядів x_i та y_i, одержана із табл. 2.2, визначається наступним виразом

$$g_i = \bar{r}_i = x_i \oplus y_i = \bar{x}_i y_i \vee x_i \bar{y}_i = \overline{\overline{\bar{x}_i y_i} \cdot \overline{x_i \bar{y}_i}}\ ,$$

де $\oplus$ – знак операції нерівнозначності або *додавання за модулем два*. Багаторозрядні слова X і Y не рівні, якщо хоч би у одному їх однойменному розряді спостерігається нерівність, тобто функція $f_{x\ne y}$ нерівності багаторозрядних слів має вид $f_{x\ne y} = g_0 \vee g_1 \vee ... \vee g_{n\text{-}1}$. Помітимо, що функції $f_{x=y}$ та $f_{x\ne y}$ інверсні, тобто $f_{x=y} = \overline{f_{x\ne y}}$, тому вони обидві легко

реалізуються на практиці одночасно простим інвертуванням однієї з них, вже збудованої.

На рис. 2.3 показано *КСП* слів на рівність та нерівність дворозрядних слів *X* і *Y*, реалізована за виразом

$$f_{x=y} = \overline{g_0 \vee g_1} = \overline{\overline{g}_0 \cdot \overline{g}_1} = \overline{\overline{r_0 \cdot r_1}}.$$

Складність комбінаційних схем за Квайном визначають сумарним числом входів усіх *ЛЕ*. Ціна по Квайну багаторозрядних *КСП* слів на рівність-нерівність, реалізованих за різноманітними операторними формами функцій, дорівнює $7n$ (рис. 2.3).

Таблиця 2.2 – *ТІ КСП*

x_i	0	0	1	1
y_i	0	1	0	1
r_i	1	0	0	1
g_i	0	1	1	0

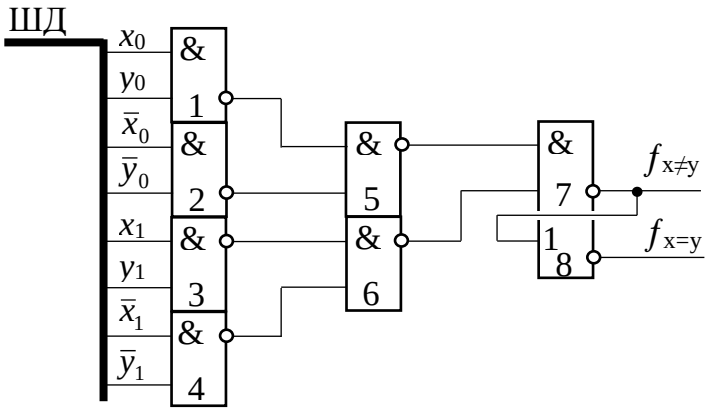


Рис. 2.3 – *КСП* слів на рівність та нерівність

Порівняння слів на більше-менше

Порівнянням слів на більше-менше називається знаходження значень логічних умов $X < Y$, $X \le Y$, $X > Y$, $X \ge Y$. Схемна реалізація таких умов здійснюється складніше, ніж схем порівняння на рівність, які будуються із сукупності *ОКСП*, не зв'язаних між собою, і реалізують функцію логічної рівнозначності.

Розглянемо порівняння багаторозрядних слів на менше, що описується перемикальною функцією

$$f_{x<y} = \begin{cases} 1 & \text{при } X < Y, \\ 0 & \text{при } X \ge Y. \end{cases} \qquad (2.8)$$

Так як вага старшого розряду слова у будь-якій позиційній системі числення на одиницю більше за суму ваг усіх його молодших розрядів, то в разі нерівності старших розрядів слів, що порівнюються, достатньо оцінити співвідношення цих розрядів, щоб з'ясувати співвідношення слів у цілому. Однак старші розряди слів можуть опинитися рівними і тоді для вияву співвідношення слів необхідно аналізувати сусідні молодші розряди. У загальному випадку необхідний аналіз усіх розрядів слів, що порівнюються. сумма весов

Справді, функція $f_{x<y} = 1$ в тому випадку, коли $x_0 < y_0$, або коли $x_0 = y_0$, але $x_1 < y_1$, або ж коли $x_0 = y_0$, $x_1 = y_1$, але $x_2 < y_2$ і так далі. Звідси, наприклад, випливає, що функція $f_{x<y}$ для чотирирозрядних слів може бути записана у такому вигляді (виконання логічної умови $X < Y$ кодується одиницею):

$$f_{x<y} = \bar{x}_0 y_0 \vee (x_0 \equiv y_0)\bar{x}_1 y_1 \vee (x_0 \equiv y_0)(x_1 \equiv y_1)\bar{x}_2 y_2 \vee$$
$$\vee (x_0 \equiv y_0)(x_1 \equiv y_1)(x_2 \equiv y_2)\bar{x}_3 y_3 = \qquad (2.9)$$
$$= \bar{x}_0 y_0 \vee (x_0 \equiv y_0)\{\bar{x}_1 y_1 \vee (x_1 \equiv y_1)[\bar{x}_2 y_2 \vee (x_2 \equiv y_2)\bar{x}_3 y_3]\},$$

де $x_i \equiv y_i = x_i y_i \vee \bar{x}_i \bar{y}_i \quad (i = 0, 1)$ – функція рівнозначності (рівності) *i*-х розрядів слів.

Чотирирозрядна *КСП*, що реалізує функцію $f_{x<y}$ в 1-ій *ОФ*, зображена на рис. 2.4. Таким чином, для вияву співвідношень більше-менше *n*-розрядних слів необхідно *n*-1 разів виконати операцію рівнозначності. Використовуючи закон алгебри логіки

$$x \vee \bar{x}B = x \vee B \text{ (або інакше } xA \vee \bar{x}AB = xA \vee AB),$$

де *A, B* – довільні логічні вирази, функцію (2.9) можна спростити заміною операції рівнозначності $(x_i y_i \vee \bar{x}_i \bar{y}_i)$ на імплікацію від *x* до *y* (в диз'юнктивній формі – $\bar{x}_i \vee y_i$) та звести її до вигляду (*докажіть самостійно*):

$$f_{x<y} = \bar{x}_0 y_0 \vee (\bar{x}_0 \vee y_0)(\bar{x}_1 y_1 \vee (\bar{x}_1 \vee y_1)(\bar{x}_2 y_2 \vee (\bar{x}_2 \vee y_2)\bar{x}_3 y_3)). \qquad (2.10)$$

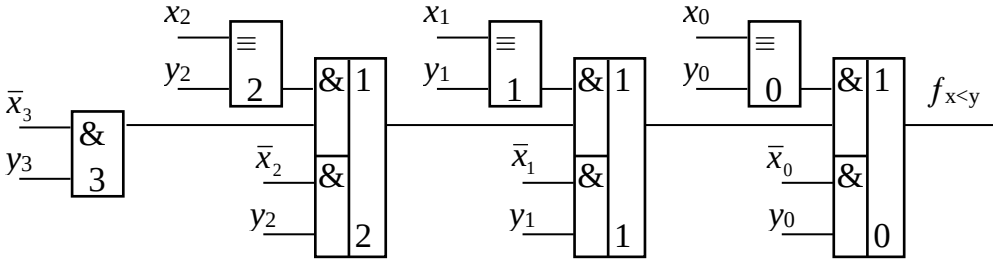


Рис. 2.4 – *КСП* слів на менше

Аналіз виразів (2.9) та (2.10) дозволяє зробити вивід про те, що багаторозрядна *КСП* на більше-менше може складатися з однотипних ОКСП, зв'язаних між собою ланцюгами передачі інформації про співвідношення молодших розрядів слів (рис. 2.5).

Слова у таких схемах аналізуються порозрядно послідовно, починаючи з молодших розрядів. На входи *i*-ої *ОКСП* має подаватися, крім значень *i*-х розрядів слів *X* та *Y*, також сигнал з виходу сусідньої молодшої (*i+1*)-ої *ОКСП*, який за аналогією з суматорами назвемо сигналом переносу $z_i = p_{i+1}$. Значення цього сигналу характеризує співвідношення

молодших частин слів, що порівнюються і однозначно визначає результат порівняння *i*-х розрядів, а отже, і частин слів від молодшого розряду до наданого. Виходом схеми є сигнал переносу із старшої *ОКСП*.

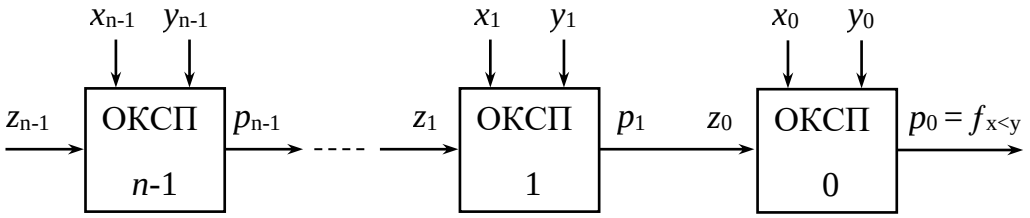


Рис. 2.5 – Узагальнена структурна схема багаторозрядної *КСП*

Для проектування багаторозрядної *КСП* однорідної структури достатньо розробити всього лише одну *ОКСП*. Для цього спершу виконується етап кодування сигналів переносу, результати якого заносяться у таблицю кодів сигналів переносу – *ТКСП* (табл. 2.3). Сигнали переносу для функції (2.8) повинні відбивати тільки дві логічні умови $x_i < y_i$ та $x_i \geq y_i$, що можуть бути закодовані значеннями однієї двійкової змінної – вхідної z_i та вихідної p_i.

Таблиця 2.3 – *ТКСП*

ЛУ \ *СП*	z_i, p_i
$x_i < y_i$	1
$x_i \geq y_i$	0

У *ТКСП* виконання логічної умови $X < Y$ закодовано одиницею, проте можливий інверсний варіант кодування. При цьому сигнали z_i та p_i кодуються завжди однаково.

Збудуємо таблицю істинності згідно з принципом роботи *ОКСП* і кодуванням сигналів переносу (табл. 2.4).

Порядок заповнення таблиці істинності наступний: якщо на визначеному вхідному наборі розряди x_i та y_i дорівнюють, то вхідний сигнал переносу передається без змін на вихід *ОКСП*, тобто $p_i = z_i$ (див., наприклад, набори 1, 6); при нерівності наданих розрядів значення функції p_i визначається тільки їх закодованим співвідношенням – без обліку сигналу переносу z_i (див., наприклад, набори 3, 5).

ДДНФ, витягнута з *ТІ*, має вигляд

$$p_i = \bar{x}_i \bar{y}_i z_i \vee \bar{x}_i y_i \bar{z}_i \vee \bar{x}_i y_i z_i \vee x_i y_i z_i = \vee(1, 2, 3, 7)$$

Таблиця 2.4 – *ТІ ОКСП*

x_i	0	0	0	0	1	1	1	1
y_i	0	0	1	1	0	0	1	1
z_i	0	1	0	1	0	1	0	1
p_i	0	1	1	1	0	0	0	1

Мінімізуючи функцію за допомогою діаграми Вейча трьох змінних, отримуємо

$$p_i = \bar{x}_i y_i \vee \bar{x}_i z_i \vee y_i z_i = \bar{x}_i y_i \vee (\bar{x}_i \vee y_i) z_i \,. \qquad (2.11)$$

Значення переносів z_i у кожну *ОКСП*, крім молодшої, визначають конкретним співвідношенням слів, що порівнюються. Перенос у молодший розряд z_{n-1} задається кодом рівності (див. табл. 2.3) і від величин слів, що порівнюються, не залежить. Відоме значення переносу у молодший розряд, визначене на етапі кодування, дозволяє спростити структуру молодшої *ОКСП*. Для наданого прикладу значення переносів при рівності кодується нулем (див. табл. 2.3), тобто $z_{n-1} = 0$. Підставивши це значення у (2.11), спростимо функцію переносу p_{n-1} у молодший розряд:

$$p_{n-1} = \bar{x}_{n-1} y_{n-1} \vee (\bar{x}_{n-1} \vee y_{n-1}) z_{n-1} = \bar{x}_{n-1} y_{n-1} \vee (\bar{x}_{n-1} \vee y_{n-1}) \cdot 0 = \bar{x}_{n-1} y_{n-1} \,.$$

Отже, трирозрядна *КСП* на менше, у якої виконання логічної умови кодується одиницею, описується наступною системою перемикальних функцій, що перетворені у другу *ОФ*:

$$\begin{aligned}
&p_2 = \bar{x}_2 y_2 = \overline{\overline{\bar{x}_2 y_2}} \,, \\
&p_1 = \bar{x}_1 y_1 \vee (\bar{x}_1 \vee y_1) p_2 = \overline{\overline{\bar{x}_1 y_1 \vee \overline{\bar{x}_1 \vee y_1} \cdot p_2}} = \overline{\overline{\bar{x}_1 y_1} \cdot \overline{x_1 \bar{y}_1} \cdot p_2} \,, \\
&p_0 = \bar{x}_0 y_0 \vee (\bar{x}_0 \vee y_0) p_1 = \overline{\overline{\bar{x}_0 y_0 \vee \overline{\bar{x}_0 \vee y_0} \cdot p_1}} = \overline{\overline{\bar{x}_0 y_0} \cdot \overline{x_0 \bar{y}_0} \cdot p_1} \,.
\end{aligned} \qquad (2.12)$$

Принципова схема трирозрядної *КСП*, що реалізована за виразами (2.12), показана на рис. 2.6.

Багаторозрядні *КСП*, що виконують інші логічні умови ($X \le Y$, $X > Y$, $X \ge Y$), проектуються аналогічно розглянутій вище методиці.

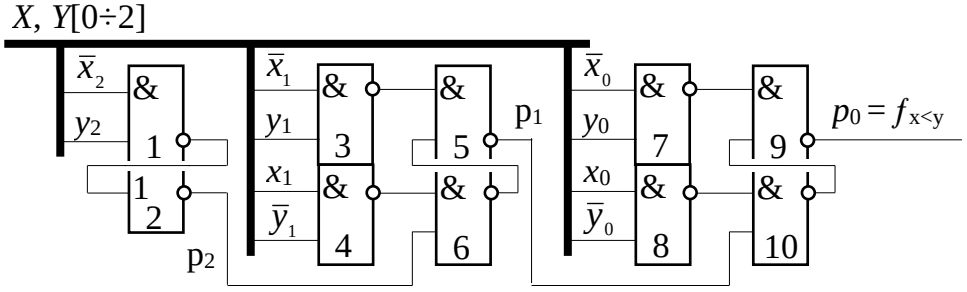


Рис. 2.6 – Принципова *КСП* трирозрядних слів

Синтез повної схеми порівняння слів

Найбільш складною є повна схема порівняння, що реалізує одночасно всю систему функцій (див. формулу (2.1)). Кожна *ОКСП* в ній має приймати і видавати одну з трьох умов (рівно, менше, більше), кодування яких провадиться двома двійковими змінними – вхідними $z_i^{'}$, $z_i^{''}$ та вихідними $p_i^{'}$, $p_i^{''}$ (рис. 2.7).

Можливе порозрядне кодування логічних умов наведено у табл. 2.5 (див. перший варіант табл. 2.1). Видно, що набір 11 не використовується для кодування. Це має призвести до появи заборонених наборів, вільне довизначення яких допоможе спростити неповністю визначені функції $p_i^{'}$ і $p_i^{''}$ при їх мінімізації.

Згідно з кодами сигналів переносу та принципом роботи *ОКСП* збудуємо таблицю істинності (табл. 2.6), заповнення якої проводять за правилом, сформульованим вище для неповної схеми порівняння. Набори (3, 7, 11, 15), відповідні конституєнтам одиниці $\tilde{x}_i\tilde{y}_i z_i^{'} z_i^{''}$, є забороненими. Досконалі форми неповністю визначених функцій $p_i^{'}$ та $p_i^{''}$, що витягнуті з ТІ, мають вигляд

$$p_i^{'} = \vee (2, 8, 9, 10, 14) = \& (0, 1, 4, 5, 6, 12, 13),$$
$$p_i^{''} = \vee (1, 4, 5, 6, 13) = \& (0, 2, 8, 9, 10, 12, 14).$$

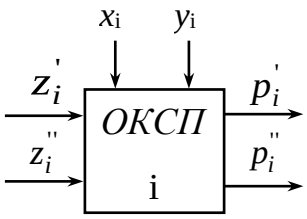


Рис. 2.7 – Повна *ОКСП* слів на більше-менше

Таблиця 2.5 – *ТКСП* повної *ОКСП*

ЛУ \ *СП*	$z_i^{'}$, $p_i^{'}$	$z_i^{''}$, $p_i^{''}$
$x_i = y_i$	0	0
$x_i < y_i$	0	1
$x_i > y_i$	1	0

Таблиця 2.6 – *ТІ ОКСП*

x_i	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
y_i	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
$z_i^{'}$	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
$z_i^{''}$	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
$p_i^{'}$	0	0	1	–	0	0	0	–	1	1	1	–	0	0	1	–
$p_i^{''}$	0	1	0	–	1	1	1	–	0	0	0	–	0	1	0	–

Проведемо сумісну мінімізацію одержаних виразів за допомогою діаграм Вейча чотирьох змінних (рис. 2.8), довизначаючи значення

функцій на заборонених наборах та виносячи змінні за дужки. Внаслідок одержимо мінімальні форми функцій

$$p_i^{'} = x_i\bar{y}_i \vee x_i z_i^{'} \vee \bar{y}_i z_i^{'} = x_i\bar{y}_i \vee (x_i \vee \bar{y}_i)z_i^{'},$$
$$p_i^{''} = \bar{x}_i y_i \vee \bar{x}_i z_i^{''} \vee y_i z_i^{''} = \bar{x}_i y_i \vee (\bar{x}_i \vee y_i)z_i^{''}.$$

Враховуючи, що умова рівності закодована нульовими значеннями сигналів переносу, тобто $z_{n-1}^{'} = z_{n-1}^{''} = 0$ (див. табл. 2.5), спростимо структуру молодшої *ОКСП*:

$$p_{n-1}^{'} = x_{n-1}\bar{y}_{n-1} \vee (x_{n-1} \vee \bar{y}_{n-1})z_{n-1}^{'} = x_{n-1}\bar{y}_{n-1} \vee (x_{n-1} \vee \bar{y}_{n-1}) \cdot 0 = x_{n-1}\bar{y}_{n-1},$$
$$p_{n-1}^{''} = \bar{x}_{n-1}y_{n-1} \vee (\bar{x}_{n-1} \vee y_{n-1})z_{n-1}^{''} = \bar{x}_{n-1}y_{n-1} \vee (\bar{x}_{n-1} \vee y_{n-1}) \cdot 0 = \bar{x}_{n-1}y_{n-1}.$$

Закодовані значення функцій системи (2.1) у вигляді рівнів змінних $p_0^{'}$, $p_0^{''}$ з'являються на виходах старшої *ОКСП*. Для декодування результату порівняння слів розробимо схему неповного дешифратора, виходи якого будуть одночасно і виходами багаторозрядної *КСП* слів. Використовуючи коди логічних умов із *ТКСП* (табл. 2.5), одержуємо досконалі форми функцій вихідного дешифратора:

$$f_{x=y} = \vee(0) = \&(1, 2) = \bar{p}_0^{'} \cdot \bar{p}_0^{''},$$
$$f_{x<y} = \vee(1) = \&(0, 2) = \bar{p}_0^{'} \cdot p_0^{''},$$
$$f_{x>y} = \vee(2) = \&(0, 1) = p_0^{'} \cdot \bar{p}_0^{''},$$

що зведемо до *МДНФ* за допомогою діаграми Вейча двох змінних (рис. 2.9):

$$f_{x=y} = \bar{p}_0^{'}\bar{p}_0^{''}, \quad f_{x<y} = p_0^{''}, \quad f_{x>y} = p_0^{'}.$$

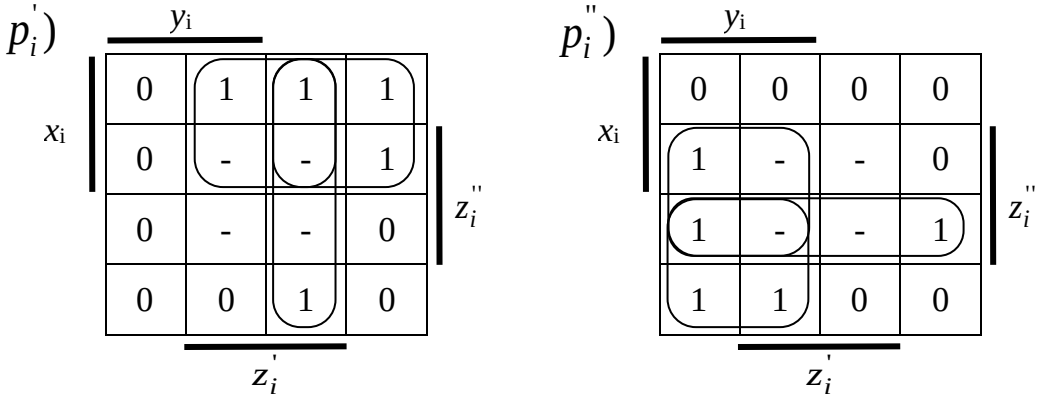


Рис. 2.8 – Мінімізація функцій повної *ОКСП*

Таким чином, повна *КСП* трирозрядних слів з вихідним дешифратором описується такою

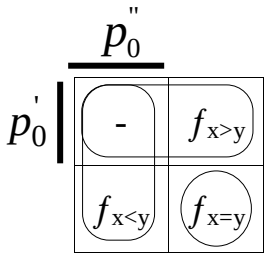


Рис. 2.9 – Синтез вихідного дешифратора

системою перемикальних функцій, що перетворені в другу операторну форму (враховуючи, що $p_{i+1} = z_i$):

$$
\begin{cases}
p_2' = x_2\bar{y}_2 = \overline{\overline{x_2\bar{y}_2}}, \\
p_2'' = \bar{x}_2 y_2 = \overline{\overline{\bar{x}_2 y_2}}, \\
p_1' = x_1\bar{y}_1 \vee (x_1 \vee \bar{y}_1)p_2' = \overline{\overline{x_1\bar{y}_1} \cdot \overline{\overline{\bar{x}_1 y_1} \cdot p_2'}}, \\
p_1'' = \bar{x}_1 y_1 \vee (\bar{x}_1 \vee y_1)p_2'' = \overline{\overline{\bar{x}_1 y_1} \cdot \overline{\overline{x_1\bar{y}_1} \cdot p_2''}}, \\
p_0' = f_{x>y} = x_0\bar{y}_0 \vee (x_0 \vee \bar{y}_0)p_1' = \overline{\overline{x_0\bar{y}_0} \cdot \overline{\overline{\bar{x}_0 y_0} \cdot p_1'}}, \\
p_0'' = f_{x<y} = \bar{x}_0 y_0 \vee (\bar{x}_0 \vee y_0)p_1'' = \overline{\overline{\bar{x}_0 y_0} \cdot \overline{\overline{x_0\bar{y}_0} \cdot p_1''}}, \\
f_{x=y} = \overline{\overline{p_0' p_0''}} = \overline{\overline{\bar{f}_{x<y} \cdot \bar{f}_{x>y}}}\ .
\end{cases}
\qquad (2.13)
$$

Помітимо, що логічні оператори $\overline{\bar{x}_i y_i}$, $\overline{x_i\bar{y}_i}$ $(i = 0, 1)$ у одержаній системі використовуються двічі, що відбито у структурі повної принципової *КСП* слів, що показана на рис. 2.10. *ЛЕ*, відмічені знаком ⊗, при реалізації та аналізі роботи схеми на лабораторному стенді мають бути обрані з індикаторами станів виходів (світлодіодами). Дослідження реалізованої схеми проводять за табл. 2.7, збудованої заздалегідь у відповідності з логікою роботи *КСП*.

Для того ж варіанта кодування вислови p_i' та p_i'' повної *ОКСП* у 7-й *ОФ*, що реалізується на *ЛЕ І-АБО-НІ*, мають вигляд

$$
\begin{aligned}
p_i' &= \overline{\bar{x}_i y_i \vee (\bar{x}_i \vee y_i)\bar{z}_i'} = \overline{\bar{x}_i y_i \vee \overline{x_i\bar{y}_i} \cdot \bar{z}_i'}, \\
p_i'' &= \overline{x_i\bar{y}_i \vee (x_i \vee \bar{y}_i)\bar{z}_i''} = \overline{x_i\bar{y}_i \vee \overline{\bar{x}_i y_i} \cdot \bar{z}_i''}\ .
\end{aligned}
\qquad (2.14)
$$

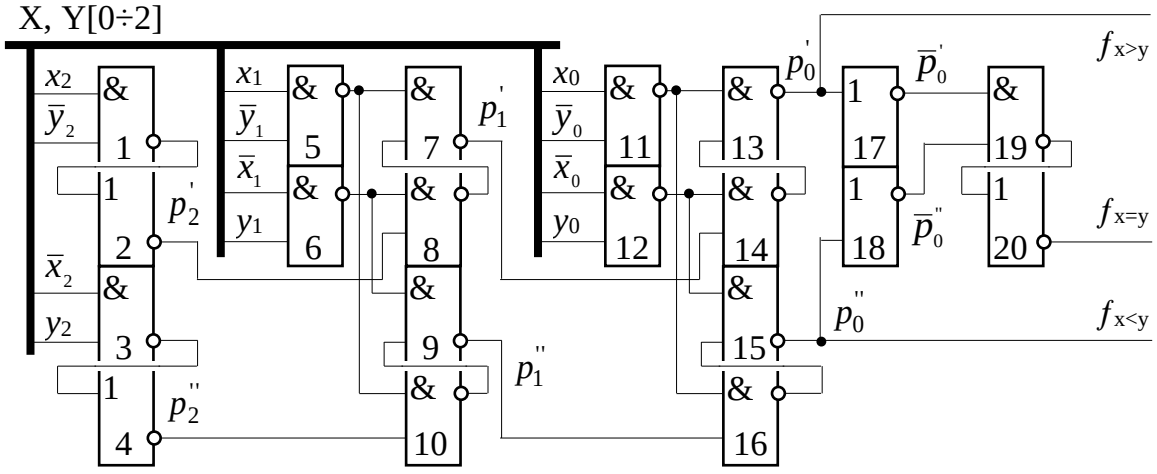


Рис. 2.10 – Повна принципова *КСП* трирозрядних слів з вихідним дешифратором

Сигнали переносу в усіх розглянутих *КСП* виробляються та передаються послідовно від молодших розрядів до старших.

Швидкодіючі *КСП* будуються з паралельною та комбінованою організацією переносів, аналогічно схемам *КСМ*.

Таблиця 2.7 – Дані для аналізу *КСП*

X	Y	$p_0^{'}\ p_0^{''}$	$f_{x<y}$	$f_{x=y}$	$f_{x>y}$
00	00	00	0	1	0
	01	01	1	0	0
	10	01	1	0	0
	11	01	1	0	0
01	00	10	0	0	1
	01	00	0	1	0
	10	01	1	0	0
	11	01	1	0	0
10	00	10	0	0	1
	01	10	0	0	1
	10	00	0	1	0
	11	01	1	0	0
11	00	10	0	0	1
	01	10	0	0	1
	10	10	0	0	1
	11	00	0	1	0

Порівняння слів за допомогою суматора

Порівняємо вирази сигналів переносу p_i неповної *ОКСП*, синтез якої був проведений раніше, і однорозрядного суматора:

ОКСП $\quad p_i = \bar{x}_i y_i \vee (\bar{x}_i \vee y_i) \cdot z_i\ , \quad z_{n-1} = 0\ ,$

ОКСМ $\quad p_i = x_i y_i \vee (x_i \vee y_i) \cdot z_i\ .$

Ці вирази відрізняються тільки кодами слів, що порівнюються (знаком інверсії над змінною x_i). Видно, що для порівняння на менше слів X і Y на входи *ОКСП* необхідно подавати прямий код слова Y та інверсний код слова X, причому на вхід переносу молодшого розряду має подаватися нульовий рівень, що у сукупності відповідає зворотному коду слова X. Тут під зворотним кодом будемо розуміти інверсію усіх розрядів числа без знаку. Отже, *ОКСП* виробляє сигнал переносу аналогічно суматору при арифметичному підсумовуванні прямого коду слова Y та зворотного коду слова X, що дозволяє використовувати схему суматора, тобто її частина,

що виробляє переноси, для неповного порівняння на більше-менше. На етапі технічного проектування використання (навіть часткове) схеми *КСМ* замість *КСП* виправдано, оскільки суматори є стандартними елементами більшості серій інтегральних мікросхем.

Аналіз роботи неповних *ОКСП*, що реалізують інші логічні умови типу нерівностей ($X \le Y$, $X > Y$, $X \ge Y$), показує, що всі вони описуються виразами двох типів (табл. 2.8), які відрізняються тільки кодами порівняльних слів. Реалізація жорсткої та нежорсткої нерівностей здійснюється *КСП* однакової структури та відрізняється лише значенням сигналу переносу z_{n-1} в молодший розряд багаторозрядної схеми. Таким чином, на схемах *КСМ* можна реалізувати будь-яке неповне порівняння слів, варіюючи коди цих слів та сигнали переносу у молодший розряд.

У табл. 2.9 наведені коди, в яких мають подаватися на *КСМ* слова, що порівнюються, для реалізації будь-яких логічних умов типу нерівностей з урахуванням їх кодування.

Таблиця 2.8 – Структури неповної *ОКСП*

Логічна умова	Логічна умова кодується одиницею ($p_0 = 1$)		Логічна умова кодується нулем ($p_0 = 0$)	
$X<Y$	$p_i = \bar{x}_i y_i \vee (\bar{x}_i \vee y_i) z_i$	$z_{n-1} = 0$	$p_i = x_i \bar{y}_i \vee (x_i \vee \bar{y}_i) z_i$	$z_{n-1} = 1$
$X \le Y$		$z_{n-1} = 1$		$z_{n-1} = 0$
$X>Y$	$p_i = x_i \bar{y}_i \vee (x_i \vee \bar{y}_i) z_i$	$z_{n-1} = 0$	$p_i = \bar{x}_i y_i \vee (\bar{x}_i \vee y_i) z_i$	$z_{n-1} = 1$
$X \ge Y$		$z_{n-1} = 1$		$z_{n-1} = 0$

Таблиця 2.9 – Порівняння слів на *КСМ*

Коди слів	Реалізуємі *ЛУ*	
	$p_0 = 1$	$p_0 = 0$
$X_{зв}$, $Y_{пр}$	$X<Y$	$X \ge Y$
$X_{дод}$, $Y_{пр}$	$X \le Y$	$X>Y$
$X_{пр}$, $Y_{зв}$	$X>Y$	$X \le Y$
$X_{пр}$, $Y_{дод}$	$X \ge Y$	$X<Y$

Порівняння слів зі знаками

Розглянуті вище *КСП* виконують порівняння абсолютних величин слів. Порівняння слів зі знаками реалізуються складніше. Словесно функцію такої *КСП* можна описати наступним чином. Допустимо, треба порівняти слова зі знаками на менше. Слово *X* буде завжди менше слова *Y* в тому випадку, якщо перше з них від'ємне, а друге – позитивне. В разі збігу знаків задана логічна умова буде виконуватися, якщо при позитивних

числах модуль першого менше модуля другого, а при від'ємних, навпаки, модуль першого більше модуля другого числа. Звідси випливає, що багаторозрядні КСП слів зі знаками на менше описуються такою перемикальною функцією $f_{x<y}$, у якій знаки від'ємних слів та задана логічна умова кодується одиницею:

$$f_{x<y} = x_0\bar{y}_0 \vee \bar{x}_0(x_0 \equiv y_0)\cdot f_{|x|<|y|} \vee x_0(x_0 \equiv y_0)\cdot f_{|x|>|y|}\,. \qquad (2.15)$$

Тут x_0, y_0 – знакові розряди слів, що порівнюються;

$f_{|x|<|y|}$, $f_{|x|>|y|}$ – функції порівняння на менше і більше абсолютних величин слів X та Y відповідно;

$x_0 \equiv y_0 = x_0y_0 \vee \bar{x}_0\bar{y}_0$ – функція рівності знакових розрядів слів, що порівнюються.

Перший член виразу (2.15) дорівнює одиниці тільки при від'ємному X, коли $x_0 = 1$, і позитивному Y, коли $y_0 = 0$, а $\bar{y}_0 = 1$. Логічний добуток $\bar{x}_0(x_0 \equiv y_0)$ відбиває факт позитивності та рівності знаків слів, що порівнюються.

Аналогічно можна одержати перемикальні функції порівняння слів зі знаками на більше та рівність:

$$f_{x>y} = \bar{x}_0y_0 \vee x_0(x_0 \equiv y_0)\cdot f_{|x|<|y|} \vee \bar{x}_0(x_0 \equiv y_0)\cdot f_{|x|>|y|} = \bar{f}_{x=y}\cdot\bar{f}_{x<y}, \qquad (2.16)$$

$$f_{x=y} = (x_0 \equiv y_0)\cdot f_{|x|=|y|} = \bar{f}_{x<y}\cdot\bar{f}_{x>y} = \overline{f_{x<y} \vee f_{x>y}}\quad .$$

Спростимо вираз (2.15), використовуючи закони алгебри логіки:

$$\begin{aligned}
f_{x<y} &= x_0\bar{y}_0 \vee \bar{x}_0(x_0 \equiv y_0)\,f_{|x|<|y|} \vee x_0(x_0 \equiv y_0)\,f_{|x|>|y|} = \\
&= x_0\bar{y}_0 \vee \bar{x}_0(x_0y_0 \vee \bar{x}_0\bar{y}_0)\,f_{|x|<|y|} \vee x_0(x_0y_0 \vee \bar{x}_0\bar{y}_0)\,f_{|x|>|y|} = \\
&= x_0\bar{y}_0 \vee \bar{x}_0\bar{y}_0\cdot f_{|x|<|y|} \vee x_0y_0\cdot f_{|x|>|y|} = \\
&= x_0\bar{y}_0 \vee x_0\bar{y}_0 \vee \bar{x}_0\bar{y}_0\cdot f_{|x|<|y|} \vee x_0y_0\cdot f_{|x|>|y|} = \\
&= (x_0\bar{y}_0 \vee \bar{x}_0\bar{y}_0\cdot f_{|x|<|y|}) \vee (x_0\bar{y}_0 \vee x_0y_0\cdot f_{|x|>|y|}) = \\
&= \bar{y}_0(x_0 \vee \bar{x}_0\,f_{|x|<|y|}) \vee x_0(\bar{y}_0 \vee y_0\,f_{|x|>|y|}) = \\
&= \bar{y}_0(x_0 \vee f_{|x|<|y|}) \vee x_0(\bar{y}_0 \vee f_{|x|>|y|}) = \\
&= x_0\bar{y}_0 \vee \bar{y}_0\cdot f_{|x|<|y|}) \vee x_0\cdot f_{|x|>|y|}\,.
\end{aligned} \qquad (2.17)$$

Із (2.17) випливає, що умова $X<Y$ виконується, якщо перше слово від'ємне, а друге – додатне, чи, коли друге слово додатне, а по модулю воно більше першого, чи, коли перше слово від'ємне і за абсолютною величиною більше за друге.

Розглянемо структуру *КСП* слів зі знаками (рис. 2.11), яка реалізує вираз (2.17), що містить функції порівняння модулів слів на більше-менше. Порівняння абсолютних величин слів на менше провадиться схемою *КСМ*,

на входи якого подаються цифрові розряди прямого коду Y і зворотного коду X (знак "¬" на рис. 2.11). Сигнал переносу із старшого цифрового розряду *КСМ* $p_1 = f_{|x|<|y|}$, одиничне значення якого посвідчує про виконання умови $|X|<|Y|$, є виходом схеми порівняння модулів. Функцію $f_{|x|>|y|}$ при готовій схемі для $f_{|x|<|y|}$ технічно простіше реалізувати у вигляді

$$f_{|x|>|y|} = \bar{f}_{|x|=|y|} \cdot \bar{f}_{|x|<|y|} = \overline{f_{|x|=|y|} \vee f_{|x|<|y|}} .$$

Це пояснюється тим, що необхідну функцію рівності модулів можна частково реалізувати на тому ж *КСМ*, використовуючи результат додавання модулів. Справді, якщо слова дорівнюють за абсолютною величиною, то модуль суми зворотного коду X і прямого коду Y містить одиниці в усіх розрядах (*наведіть приклад*). Отже, операцію порівняння слів на рівність можна звести до більш простого порівняння на рівність результату на виході *КСМ* з константою $K = (2^{n-1}-1)_{(10)}$, тобто $f_{|x|=|y|} = f_{|s|=|k|} = s_1 \cdot s_2 \cdot \ldots \cdot s_{n-1}$, де s_i ($i = 1, 2, \ldots, n-1$) – цифрові розряди суми. Надана функція реалізується комбінаційною схемою порівняння з константою (*СПК*) – логічним елементом I із n-1 входами.

Безпосередньо функцію $f_{x<y}$ (див. формулу 2.17) реалізує схема порівняння слів зі знаками (*СПЗ*), для якої вхідними змінними є функції порівняння модулів та знаки слів X і Y.

Повна *КСП* слів із знаками повинна реалізувати і спрощені вирази (2.16).

[3; 9-12]

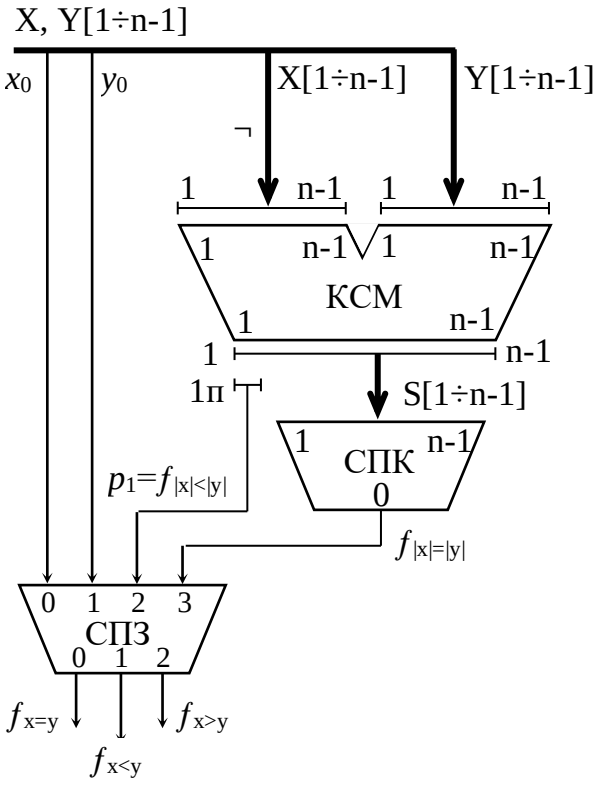


Рис. 2.11 – *КСП* слів зі знаками

Лабораторна робота № 5

СИНТЕЗ ТА ДОСЛІДЖЕННЯ ТРИГЕРНИХ СХЕМ ДОВІЛЬНИХ ТИПІВ

1 Мета лабораторної роботи

Метою лабораторної роботи є вивчення принципів функціонування, логічного проектування та експериментального дослідження схем різноманітних типів тригерів інтегральних систем елементів. Звичайно тригери розглядаються як елементарні автомати, що використовуються у вигляді запам'ятовуючих елементів при структурному синтезі дискретних автоматів. У цій роботі синтезуються різноманітні типи тригерів як структурні автомати на основі найпростішого асинхронного RS-тригера.

2 Практична частина роботи

2.1 Домашнє завдання

1. Вивчити основні теоретичні положення до наданої лабораторної роботи, підготуватися до відповідей на поставлені нижче контрольні запитання та завдання.
2. Ознайомитися з порядком проведення лабораторної роботи, описом лабораторного стенда та налагодженням схеми.

2.2 Контрольні запитання до роботи

1. Який рівень вхідного сигналу називають активним?
2. У чому відмінність між синхронними та асинхронними тригерами?
3. У чому відмінність між асинхронними та синхронними входами синхронних тригерів?
4. Які умовні графічні позначення тригерів різноманітних типів та їх входів?
5. У чому особливість синтезу тригерних схем різноманітних типів порівняно із загальним методом синтезу структурних автоматів?
6. Які вихідні дані необхідні для синтезу тригерних схем різноманітних типів?
7. Як визначити за таблицею переходів синхронного RS-тригера активні рівні вхідних сигналів t, R, S?
8. Як визначити по таблицях переходів універсальних JK- та DV-тригерів активні рівні всіх їх вхідних сигналів?
9. Як визначити по таблицях переходів синхронних T- та D-тригерів активні рівні вхідних сигналів t, T та t, D?
10. Скільки варіантів СТП може бути побудовано для синхронного RS-тригера? Побудуйте їх.

11. Скільки варіантів СТП може бути побудовано для універсального JK-тригера? Побудуйте їх.

12. Скільки варіантів СТП може бути побудовано для універсального DV-тригера? Побудуйте їх.

13. Скільки варіантів СТП може бути побудовано для синхронного та ТП для асинхронного Т-тригера? Побудуйте їх.

14. Скільки варіантів СТП може бути побудовано для синхронного D-тригера? Побудуйте їх.

15. Поясніть структуру та роботу двотактного тригера.

16. З якою метою у схему двотактного тригера будь-якого типу вводиться синхронний RS-тригер?

17. Як проводиться синтез двотактних тригерів? Чим він відрізняється від синтезу однотактних тригерів?

18. Наведіть приклади перетворень універсального JK-тригера у тригери інших типів.

19. Наведіть приклади перетворень універсального DV-тригера у тригери інших типів.

20. Наведіть приклади перетворень RS- та D-тригерів.

2. 3 Порядок виконання роботи

1. Синтезувати за заданим варіантом (див. табл. 3.1) структуру необхідного типу тригера. Схема тригера повинна мати входи асинхронної установки у 0 та 1. Зображати ЛЕ тригери (І-НЕ) треба відповідно до вимог ЄСКД та обов’язково зазначати їх порядковий номер на комутаційному полі лабораторного стенда. Для індикації станів тригера вибираються ЛЕ зі вмонтованими у них світлодіодами.

Для тригерів типу RS, D, DV синтезуються та реалізуються синхронні однотактні і двотактні схеми. Для тригерів типу Т, JK синтезуються і досліджуються асинхронні та синхронні варіанти двотактних схем.

2. Побудувати часові діаграми вихідних сигналів всіх ЛЕ, що складають схему, для наборів 0 та 1 (D- і Т-тригери) та 00, 01, 10, 11 (для RS-, JK-, DV-тригерів). Вихідний стан тригера дорівнює 0. Часову діаграму будувати без урахування часових затримок сигналів ЛЕ. Побудова закінчується, як тільки одержано такий же стан входів та виходів тригера, що вже зустрічався раніше. Часові діаграми будуються для двох варіантів тригерів (однотактних та двотактних чи синхронних та асинхронних).

3. Зібрати схему дослідження тригера (рис. 3.1) і перевірити її роботу.

4. Підключити розроблену схему тригера заданого типу до схеми дослідження тригера (спочатку одну, потім, після її дослідження, другу).

5. Дослідити тригер у статичному режимі.

6. Дослідити тригер у динамічному режимі.

7. Оформити звіт по лабораторній роботі.

2.4 Прилади, використані у роботі

Універсальний лабораторний стенд (УЛС), опис якого наведено у підрозділі 1.4 лабораторної роботи № 1.

Таблиця 3.1 – Таблиця варіантів

Варіант	СТП			Варіант	СТП			Варіант	СТП			Варіант	СТП			Варіант	СТП		
	Q tRS	0	1		Q tJK	0	1		Q tDV	0	1		Q tD	0	1		Q tT	0	1
1	0 - -	0	1	5	000	1	0	9	000	0	1	13	00	1	1	17	00	1	0
	100	0	1		001	1	1		001	1	1		01	0	0		01	0	1
	101	1	1		010	0	0		010	0	1		10	0	1		10	0	1
	110	0	0		011	0	1		011	0	0		11	0	1		11	0	1
	111	-	-		1 - -	0	1		1 - -	0	1								
2	000	0	1	6	0 - -	0	1	10	0 - -	0	1	14	00	0	0	18	00	0	1
	001	1	1		100	1	1		100	0	1		01	1	1		01	1	0
	010	0	0		101	1	0		101	1	1		10	0	1		10	0	1
	011	-	-		110	0	1		110	0	1		11	0	1		11	0	1
	1 - -	0	1		111	0	0		111	0	0								
3	0 - -	0	1	7	0 - -	0	1	11	0 - -	0	1	15	00	0	1	19	00	0	1
	100	0	0		100	0	0		100	1	1		01	0	1		01	0	1
	101	-	-		101	0	1		101	0	1		10	1	1		10	1	0
	110	0	1		110	1	0		110	0	0		11	0	0		11	0	1
	111	1	1		111	1	1		111	0	1								
4	0 - -	0	1	8	0 - -	0	1	12	0 - -	0	1	16	00	0	1	20	00	0	1
	100	-	-		100	1	0		100	0	1		01	0	1		01	0	1
	101	0	0		101	1	1		101	0	0		10	0	0		10	0	1
	110	1	1		110	0	0		110	0	1		11	1	1		11	1	0
	111	0	1		111	0	1		111	1	1								

2.5 Експериментальне дослідження тригера

Експериментальне дослідження тригера проводиться на універсальному лабораторному стенді.

Для дослідження роботи тригера необхідно подати на його входи послідовність різноманітних наборів сигналів. Це реалізується за допомогою двійкового дворозрядного лічильника (рис. 3.1). Кодові послідовності (00, 01, 10, 11) знімаються із прямих виходів тригерів лічильника та подаються на інформаційні входи досліджуваного тригера. Для тригерів з одним інформаційним входом достатньо використовувати однорозрядний двійковий лічильник (Т-тригер), знімаючи сигнали з його прямого виходу.

Працездатність заданого типу тригера перевіряють у статичному та динамічному режимах. У статичному режимі сигнали на синхронізуючий вхід досліджуваного тригера, подаються із виходу ГПІ. В цьому режимі за СТП та АТП перевіряють правильність виконання переходів тригера. Схема, яку необхідно зібрати для дослідження, зображена на рис. 3.1.

Послідовність дослідження у статичному режимі така. У вихідному стані вхід опитування регістра “ВК” підключають до “землі”. В цьому випадку виходи клавішного регістра закриті (на обох парафазних виходах x_i кожного розряду буде рівень 1) і код, що набирається на регістрі, не передається на схему задання кодової послідовності (дворозрядний чи однорозрядний лічильник). На клавішному регістрі набирають початковий код, який необхідно занести у лічильник та досліджуваний тригер. Натиснута клавіша відповідає логічній одиниці. Відмиканням входу “ВК” від “землі” набраний код заносять у лічильник та досліджуваний тригер. Потім вхід “ВК” знову вмикають до “землі” та, натискуючи кнопку ГПІ, перевіряють правильність переходу тригера з одного стану в інший.

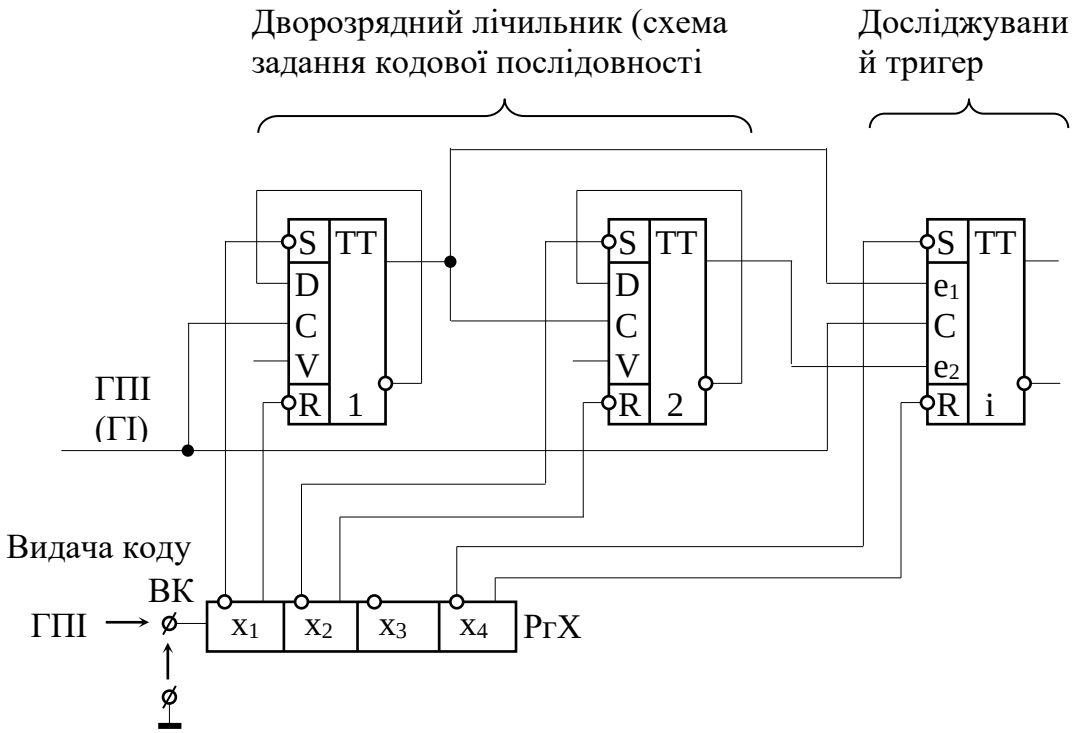


Рис. 3.1 – Схема дослідження тригера

При дослідженні схеми у динамічному режимі на її синхронізуючий вхід подають імпульси з виходу ГІ (ГПІ у цьому випадку від’єднують). Динамічний режим дозволяє визначити часові параметри та працездатність тригера за допомогою осцилографа на різноманітних частотах ГІ звірянням часової діаграми, побудованої заздалегідь, з одержаними осцилограмами.

Наладка схеми тригера пов’язана з пошуком та усуненням несправностей. Хибна робота тригера може бути зумовлена тим, що:

– допущена помилка при комутації схеми;
– у схемі тригера опинилися зламані (відмовні) ЛЕ;
– відсутні контакти у з'єднаннях;
– допущена помилка на етапі синтезу логічної структури тригера.

Налагоджувати схему доцільно у статичному режимі. При виявленні збоїв у роботі тригера, необхідно конкретизувати та локалізувати несправність з тим, щоб після цього її усунути.

2.6 Зміст звіту

Оформлення протоколу повинно починатися при домашній підготовці до лабораторної роботи, до початку занять. У протокол вносять:

1. Титульний список (назва роботи, дата, прізвища членів бригади та ін.).
2. Стислий опис роботи і таблицю переходів заданого типу тригера.
3. Результати синтезу схем тригера (таблиці переходів та істинності, діаграми Вейча, функції збудження елементарного автомата та ін.).
4. Схеми тригера, побудовані на ЛЕ І-НЕ.
5. Часові діаграми роботи схем тригера.
6. Схему дослідження тригера.
7. Стислі висновки за результатами роботи.

При оформленні протоколу необхідно внести зміни у схеми, що заздалегідь проектувалися, якщо у ході виконання роботи вони змінилися, з стислим описом причин змін.

3 Основні теоретичні положення

3.1 Вступ

В цифрових пристроях як елементарні автомати використовують тригери різноманітних типів. Тригером називають елементарний автомат Мура з двома внутрішніми станами, яким відповідають два різноманітних вихідних сигнали. Властивість повноти системи переходів (ПСП) та виходів (ПСВ) дає можливість позначати внутрішні стани та вихідні сигнали тригерів одним і тим самим символом Q та кодувати їх літерами двозначного структурного алфавіту (0 та 1). Вхідний структурний алфавіт таких елементарних автоматів також двозначний.

Стан тригера звичайно представляється двома парафазними вихідними сигналами Q та $\overline{Q}$: у стані 0 сигнали Q = 0 і $\overline{Q} = 1$, у стані 1 сигнали Q = 1, а $\overline{Q} = 0$. Коли стан тригера відбивається парою сигналів Q та $\overline{Q}$, кажуть, що його стан Q представляється у парафазному коді. Вихід тригера Q називається прямим, а вихід $\overline{Q}$ – інверсним.

3.2 Активний рівень (діюче значення) вхідного сигналу

Основою для синтезу тригерних схем у даній роботі є логічні елементи (ЛЕ) І-НЕ. Активним рівнем вхідного сигналу (0 або 1) для конкретної схеми називають таке його значення, яке, будучи поданим хоч би на один вхід схеми, однозначно визначає значення вихідного сигналу незалежно від рівнів сигналів на інших її входах.

Для ЛЕ І та І-НЕ активним рівнем вхідних сигналів є нульовий. Справді, при подачі на один з входів цих ЛЕ значення 0 на виході ЛЕ І установиться 0, на виході ЛЕ І-НЕ установиться 1, що визначається логічною функцією, яку виконує кожний з цих елементів.

Відповідно, для ЛЕ АБО та АБО-НЕ діючим значенням вхідних сигналів є одиниця. В цьому випадку на виході ЛЕ АБО з'явиться сигнал 1, на виході ЛЕ АБО-НЕ – сигнал 0, якщо хоч би на один з входів буде поданий сигнал 1.

Звідси випливає, що на невикористані (“зайві”) входи ЛЕ для нормальної роботи схем необхідно подавати тільки неактивні рівні сигналів: для ЛЕ І, І-НЕ – значення 1, для ЛЕ АБО, АБО-НЕ – значення 0. Відзначимо, що в інтегральних системах логічних елементів на невикористані входи ЛЕ автоматично подається потенціал неактивного рівня сигналів. Однак для надійної роботи схем в інженерних розробках на невикористані входи елементів зовнішнім монтажем виконується додаткова подача неактивних рівнів вхідних сигналів.

3.3 Синхронні та асинхронні тригери

Тригери поділяються на два класи: синхронні та асинхронні. Синхронний тригер має синхронізуючий вхід С для подачі тактового сигналу *t*, який не несе логічної функції, а тільки відзначає хід реального часу та дозволяє перемикання тригера лише у визначені його моменти. Якщо рівень синхронізуючого (тактового) сигналу неактивний, тригер зберігає свій поточний стан незалежно від значень сигналів на його інформаційних входах. Тригер перемикається у стан, відповідний значенням вхідних інформаційних сигналів, тільки в момент надходження активного рівня тактового сигналу *t*. Тригер, що не має синхронізуючого входу, називається асинхронним. Такий тригер відразу перемикається в момент зміни значень інформаційних сигналів на його входах.

Умовні графічні позначення тригерів на функціональних схемах показані на рис. 3.2,в; 3.3,в; 3.5,б. В основному полі ставиться символ Т або ТТ для позначення однотактного чи двотактного (див. далі) тригера. Ліве додаткове поле може бути поділено на дві частини: асинхронну та синхронну. В асинхронній частині проставляються символи S та R несинхронізованої установки тригера у 1 та 0, в синхронній – символи входів відповідно до типу тригера. При цьому використовують такі позначення для входів:

S – (*set* – установка) – вхід роздільної установки тригера в 1;

R – (*reset* – скидання) – вхід роздільної установки в 0;

T – (*toggle* – перекидатися) – вхід тригера із лічильним входом;

D – (*delay* – затримка) – вхід D-тригера;

J – вхід синхронізованої установки у стан 1 (або інверсний стан) в універсальному JK-тригерові;

K – вхід синхронізованої установки у стан 0 (або інверсний стан) в універсальному JK-тригерові;

V – вхід, за яким тригер настроюється на виконання функцій, що визначаються значенням інших інформаційних входів (“клямка”);

C – вхід синхронізації (тактовий вхід);

 C – прямий статичний вхід, прийом вхідної інформації виконується при одиничному значенні синхронізуючого сигналу (активний рівень сигналу дорівнює одиниці);

 C – інверсний статичний вхід, прийом вхідної інформації виконується при нульовому значенні синхронізуючого (тактового) сигналу (активний рівень сигналу дорівнює 0);

 C – прямий динамічний вхід, прийом інформації у тригер виконується в момент зміни значення синхронізуючого сигналу з 0 на 1 (перепад 0/1);

 C – інверсний динамічний вхід, прийом інформації у тригер виконується в момент зміни значення синхронізуючого сигналу з 1 на 0 (перепад 1/0).

3.4 Асинхронний RS-тригер

Асинхронний (несинхронізований) RS-тригер (ARST) на інтегральних ЛЕ АБО-НЕ показано на рис. 3.2,а. Тригер утворений з двох логічних елементів АБО-НЕ, сполучених так, що виникають позитивні зворотні зв’язки, завдяки яким у одному з двох стійких станів вихідний транзистор одного елемента закрито, а другого – відкрито. Закон функціонування цього тригера визначається асинхронною кодованою таблицею переходів АТП (табл. 3.2) або умовною таблицею переходів УТП (рис. 3.2,б).

Забороненим вхідним набором є така комбінація, при якій на обох входах виявляються активні рівні сигналів. В цьому випадку тригерне кільце розривається та тригер перетворюється у два незалежних інвертора. Для тригера на ЛЕ АБО-НЕ забороненим вхідним набором є набір 11.

З’ясувати, якій з входів (верхній чи нижній на рис. 3.2,а) є входом S та який – R, можна таким чином. Якщо подати на входи набір сигналів 01 (0 – на верхній вхід, 1 – на нижній), то на виході Q тригера установиться сигнал 1, на виходи $\bar{Q}$ – сигнал 0, тобто тригер установиться в одиничній стан. Вхід, на який був поданий активний сигнал, при цьому має бути входом S (нижній вхід),

Таблиця 3.2 – АТП ARST на ЛЕ АБО-НЕ

Q / RS	0	1	Режими роботи
00	0	1	Збереження
01	1	1	Установка 1
10	0	0	Скидання
11	-	-	Заборона

інший (верхній) – входом R.

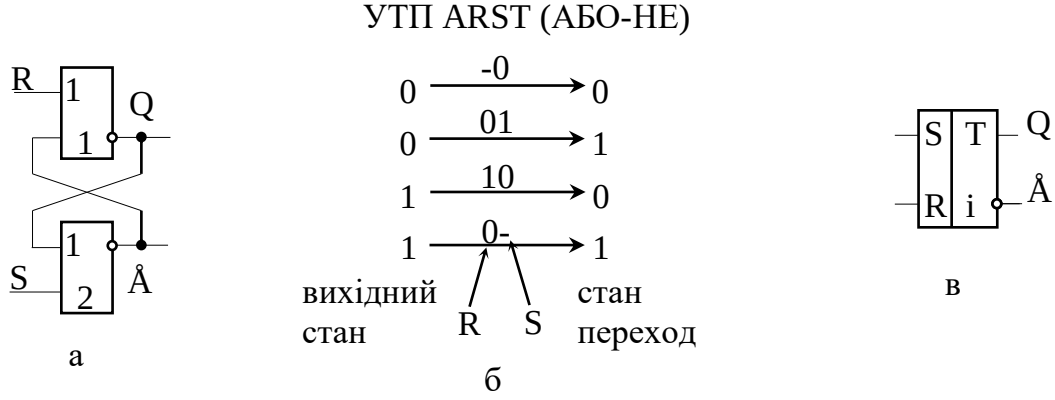


Рис. 3.2 – Принципова (а) та функціональна (в) схеми асинхронного RS-тригера на ЛЕ АБО-НЕ та його умовна таблиця переходів – УТП (б)

Асинхронний RS-тригер на ЛЕ І-НЕ показано на рис. 3.3,а. Закон функціонування цього тригера задається АТП (табл. 3.3) або УТП (рис. 3.3,б). Оскільки активний рівень вхідного сигналу тут нульовий, то забороненим буде вхідний набір 00. Входи S та R визначаються на схемі (рис. 3.3,а) аналогічно шляхом подачі вхідних наборів 01 та 10 і вияву станів, у яких опиниться тригер (при цьому заздалегідь визначають прямий та інверсний виходи). Умовне графічне позначення даного тригера показано на рис. 3.3,в.

Таблиця 3.3 – АТП ARST на ЛЕ І-НЕ

Q / $\overline{R}\overline{S}$	0	1	Режими роботи
00	-	-	Заборона
01	0	0	Скидання
10	1	1	Установка 1
11	0	1	Зберігання

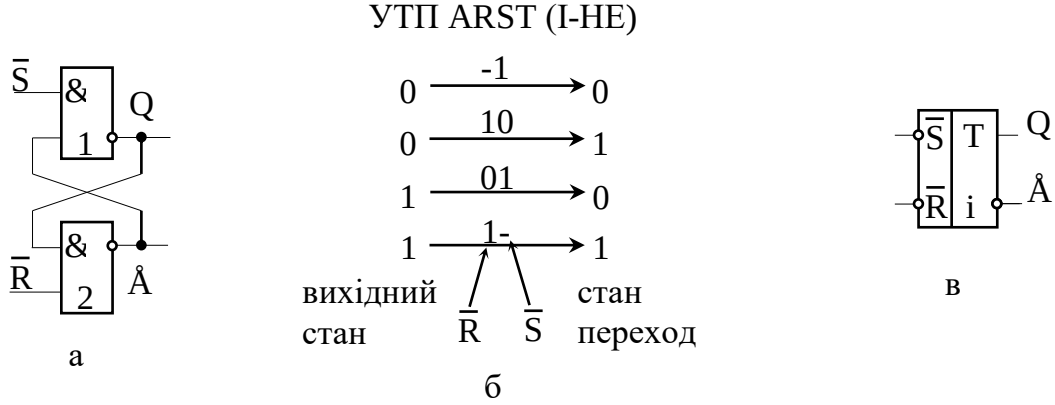


Рис. 3.3 – Принципова (а) та функціональна (в) схеми асинхронного RS-тригера на ЛЕ І-НЕ та його умовна таблиця переходів – УТП (б)

3.5 Синтез тригерних схем різноманітних типів

Синтез тригерів різноманітних типів проводять на основі теорії структурного синтезу цифрових автоматів. У вигляді структурного автомата (СА), схему якого треба спроектувати, тут виступає той чи інший тип тригера, заданий таблицею переходів. Елементарним автоматом, на базі якого будується схема СА, служить асинхронний RS тригер на ЛЕ І-НЕ чи АБО-НЕ.

Однак синтез схем різноманітних тригерів проводиться простіше, ніж загальний структурний синтез автоматів, з таких причин:

– таблиця переходів СА (тригера, що синтезується) звичайно вже задана у закодованому вигляді, в зв'язку з чим відпадає етап кодування абстрактних змін;

– число внутрішніх станів СА, що синтезується (тригера потрібного типу), дорівнює числу станів елементарного автомата (ARST), тобто функції кодованих виходів вже визначені вихідними сигналами (Q і $\overline{Q}$) ARST.

Таким чином, синтез тригерних схем різноманітних типів зводиться тільки до відшукання функцій збудження ARST на основі таблиці переходів заданого типу тригера та умовної таблиці переходів використаного елементарного автомата.

3.6 Приклад синтезу універсального DV-тригера

Синтез тригерних схем розглянемо на прикладі синтезу схеми універсального DV-тригера. Цей тип тригера має синхронізуючий вхід С, на який подається сигнал синхронізації *t*, тому закон його функціонування задається синхронною таблицею переходів СТП (табл. 3.4). Залежно від кодування (0 або 1) активних рівнів вхідних сигналів для кожного типу тригера можна побудувати 2^n варіантів синхронних таблиць переходів, де n – число вхідних змін. Так, наприклад, для універсального JK-тригера можна представити 8 варіантів СТП, для синхронного Т-тригера – 4 різноманітних СТП і т.д.

Таблиця 3.4 – СТП *DV*-тригера

Q *tDV*	0	1	Режим роботи
000	0	1	Зберігання
001	0	1	Зберігання
010	0	1	Зберігання
011	0	1	Зберігання
100	0	1	Зберігання
101	0	0	Скидання
110	0	1	Зберігання

Таблиця 3.5 – СТП *DV*-тригера (спакована)

Q *tDV*	0	1	Режим роботи
0 - -	0	1	Зберігання
100	0	1	Зберігання
101	0	0	Скидання
110	0	1	Зберігання
111	1	1	Установка 1

111	1	1	Установка 1	

Таблиця 3.6 – АТП для асинхронного скидання

Q / $\overline{R}_a$	0	1	Режим роботи
0	0	0	Скидання
1	~	~	до СТП

Таблиця 3.7 – АТП для асинхронної установки1

Q / $\overline{S}_a$	0	1	Режим роботи
0	1	1	Установка 1
1	~	~	до СТП

Таблиця 3.8 – АТП (повна)

Q / $\overline{R}_a\overline{S}_a$	0	1	Режим роботи
00	-	-	Заборона
01	0	0	Скидання
10	1	1	Установка 1
11	~	~	до СТП

Для компактного запису СТП можна представити у вигляді табл. 3.5, у якій всі набори інформаційних вхідних сигналів при неактивному рівні сигналу синхронізації зведені у один рядок. Для синтезу схеми тригера проте необхідно використовувати тільки розпаковану СТП.

Крім синхронних входів D, V, C даний тригер може мати асинхронні входи установки у 0 або 1 ($\overline{R}_a$ та $\overline{S}_a$ або тільки $\overline{R}_a$ чи $\overline{S}_a$). Тоді функціонування тригера, крім СТП, описується ще конкретною асинхронною таблицею переходів АТП (табл. 3.6, 3.7, або 3.8). Знак "~" в АТП означає, що за наявності неактивних рівнів сигналів на асинхронних входах, тригер функціонує згідно з СТП, тобто асинхронні входи логічно відключені. Узагальнена структурна схема DV-тригера, аналогічна схемі будь-якого однотактного тригера, показана на рис. 3.4.

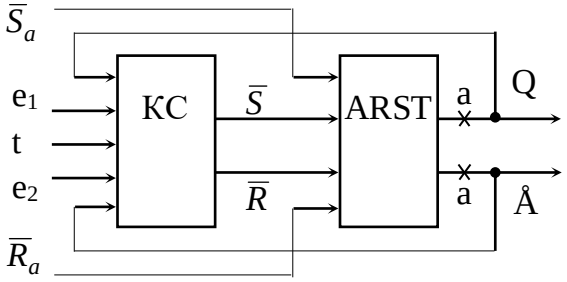


Рис. 3.4 – Узагальнена структурна схема однотактного тригера

У вигляді елементарного автомата вибираємо ARST на ЛЕ І-НЕ, як найбільш розповсюджений у інтегральних системах логічних елементів. Результатом синтезу схеми DV-тригера будуть функції збудження $\overline{R}$ та $\overline{S}$ ARST. Сигнали асинхронної установки DV-тригера у 0 (вхід $\overline{R}_a$) та 1 (вхід $\overline{S}_a$) подаються безпосередньо на додаткові входи ARST (рис. 3.4). Якщо один з них приймає нульове значення, отже DV-тригер встановлюється у відповідний стан, незалежно від значень сигналів на входах D, V, C. Таким чином пріоритет асинхронних входів вище ніж синхронних. Для керування тригером по синхронних входах на асинхронні входи $\overline{R}_a$, $\overline{S}_a$ мають бути подані неактивні рівні сигналів (в даному випадку – одиничні).

Для визначення функцій збудження $\overline{R}$, $\overline{S}$ будується таблиця функцій збудження ТФЗ (табл. 3.9) по СТП та УТП асинхронного RS-тригера на ЛЕ І-НЕ (рис. 3.3,б). ДДНФ функцій збудження ARST, записані у числовій формі, такі:

$\overline{R} = \vee(14, 1, 3, 5, 7, 9, 13, 15) = \&(11)$,

$\overline{S} = \vee(0, 2, 4, 6, 8, 10, 12, 11) = \&(14)$.

МДНФ функцій збудження одержимо методом сумісної мінімізації за допомогою діаграм Вейча (рис. 3.5):

$$\overline{R} = \overline{t} \vee \overline{V} \vee D = \overline{tV\overline{D}}; \quad \overline{S} = \overline{t} \vee \overline{V} \vee \overline{D} = \overline{tVD} \quad .$$

За цими виразами вже можна побудувати схему універсального DV-тригера. Однак, щоб не застосовувати інвертор для інвертування змінної D у виразі $\overline{R}$, перетворюємо цей вираз таким чином:

$$\overline{R} = \overline{tV\overline{D}} = \overline{tV\overline{D}} \vee tV \cdot \overline{t\overline{V}} = \overline{tV(\overline{D} \vee \overline{tV})} = tV \cdot \overline{tVD} = \overline{tV\overline{S}} \, .$$

Таблиця 3.9 – ТФЗ

Q / tDV	0	1
000	-1	1-
001	-1	1-
010	-1	1-
011	-1	1-
100	-1	1-
101	-1	01
110	-1	1-
111	10	1-

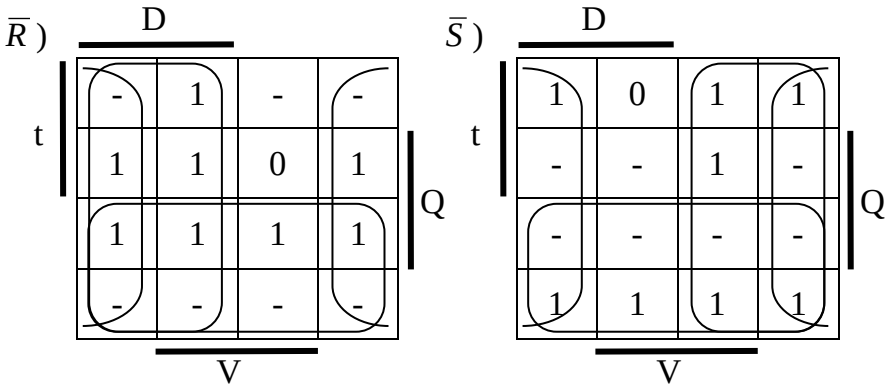
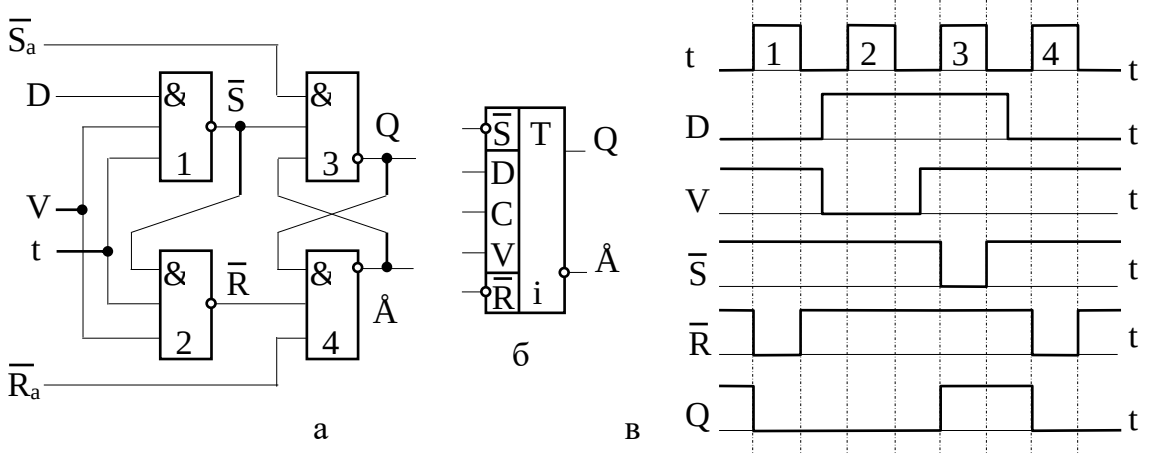


Рис. 3.5 – Діаграми Вейча для сумісної мінімізації функцій збудження ARST

Схема універсального DV-тригера з асинхронними входами $\overline{S}_a$, $\overline{R}_a$ (табл. 3.8) та його умовне графічне позначення показані на рис. 3.6а,б. Послідовність перемикання в часі ЛЕ схеми DV-тригера зображується часовою діаграмою (рис. 3.6,в), з якої випливає, що даний однотактний тригер затримує розповсюдження вхідного сигналу максимум на півтакту. Для затримки на один такт використовується двотактна схема тригера (див. далі). Часова діаграма роботи DV-тригера побудована у припущенні, що фронти та затримка сигналів логічними елементами дорівнюють нулю. Вхідні сигнали перемикають тригер з приходом синхронізуючого імпульсу. Зміна сигналів на інформаційних входах D та V дозволена тільки у проміжках між синхронізуючими імпульсами.



а б в

Рис. 3.6 – Принципова (а), функціональна (б) схеми
однотактного DV-тригера та часова діаграма його роботи (в)

Всі інші типи тригерів синтезуються за розглянутою методикою. Однак однотактні синхронні Т- та JK-тригери (рис. 3.4), у яких є загальний зворотний зв’язок з виходу на вхід, будуть стійко працювати тільки у тому випадку, якщо синхронізуючий імпульс за тривалістю менше ніж час спрацьовування асинхронного RS-тригера. Інакше на виходах цих тригерів буде спостерігатися генерація. Оскільки тривалість синхронного сигналу в реальних схемах звичайно більше за час перемикання тригера, отже схеми синхронних однотактних Т- та JK-тригерів практично не мають застосування.

3.7 Двотактні тригери

Всі тригерні схеми можуть бути побудовані за узагальненою схемою (рис. 3.4). Однак для стійкої роботи у схемах деяких типів тригерів у точках "а" мають бути включені елементи затримки, які затримували б у даному такті передачу на вхід КС за зворотним зв'язком нового стану. Для реалізації затримки в інтегральних схемах звичайно використовують запам'ятовуючі елементи на основі асинхронних RS-тригерів.

Узагальнена структурна схема таких (двотактних) тригерів (рис. 3.7) утримує додаткову комбінаційну схему (ДКС) та додатковий асинхронний RS-тригер (DARST). Вхідні сигнали e_1 та e_2 основної КС визначаються типом заданого тригера. У двотактному тригері за синхронним сигналом t значення функцій збудження $\overline{R}$ та $\overline{S}$ запам’ятовуються у ARST, а після цього за зміщеним в часі синхронним сигналом f передаються у DARST. Бо такий тригер керується двома серіями зсунутих синхронізуючих сигналів t та f (які знаходяться, звичайно, у протифазі), він одержав назву двотактного.

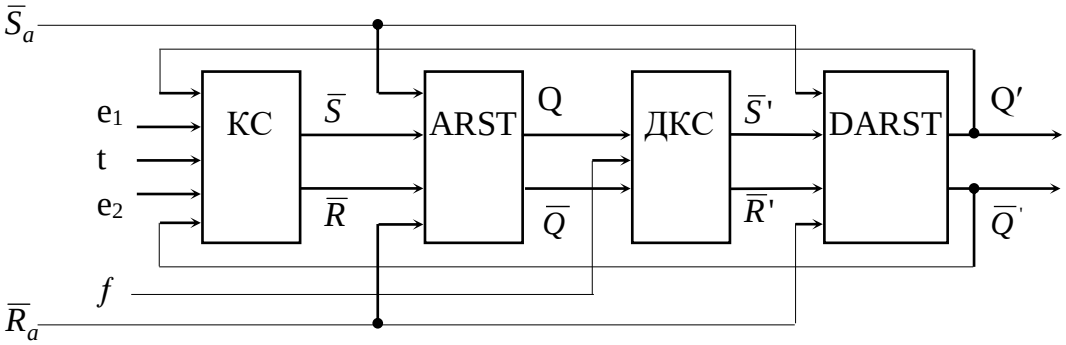


Рис. 3.7 – Узагальнена структурна схема двотактного тригера

Умови роботи ДКС можна словесно (вербально) сформулювати таким чином:

– за відсутності синхронного сигналу f інформація не повинна передаватися з основного ARST у DARST, тобто вихідні рівні сигналів ДКС $\overline{R}'$ та $\overline{S}'$ мають бути неактивними (для ЛЕ І-НЕ – одиничними);

– по наявності синхронного сигналу f DARST повинен встановлюватися у стан, який відповідає стану ARST, тобто значення $\overline{R}'$ і $\overline{S}'$ мають збігатися зі значеннями сигналів $\overline{R}$, $\overline{S}$;

– синхронні сигнали t та f не повинні перетинатися в часі, інакше інформація буде передаватися у тому самому такті у DARST.

На підставі вербального опису складається таблиця істинності (табл. 3.10), з якої витягаються та мінімізуються функції $\overline{R}'$ та $\overline{S}'$ ДКС:

$$\overline{R}' = \overline{f} \vee Q = \overline{f \cdot \overline{Q}}; \quad \overline{S}' = \overline{f} \vee \overline{Q} = \overline{f \cdot Q}.$$

Таким чином, структуру синхронного двотактного тригера будь-якого типу можна представити у вигляді незмінної частини, яка складається з синхронного однотактного RS-тригера, що тактується синхросигналом $f = \overline{t}$, і змінної частини, яка визначається заданим типом синхронного чи асинхронного тригера.

На рис. 3.8,а зображена схема синхронного двотактного DV-тригера, де ліва частина являє собою синхронний однотактний DV-тригер, а права – синхронний RS-тригер. Умовне графічне позначення двотактного тригера містить у верхній частині основного поля дві літери ТТ. З часової діаграми (рис. 3.8,б) видно, що двотактний тригер затримує вхідний сигнал D максимум на один такт.

Таблиця 3.10 – ТІ ДКС

f	0	0	1	1
Q	0	1	0	1
$\overline{R}'$	1	1	0	1
$\overline{S}'$	1	1	1	0

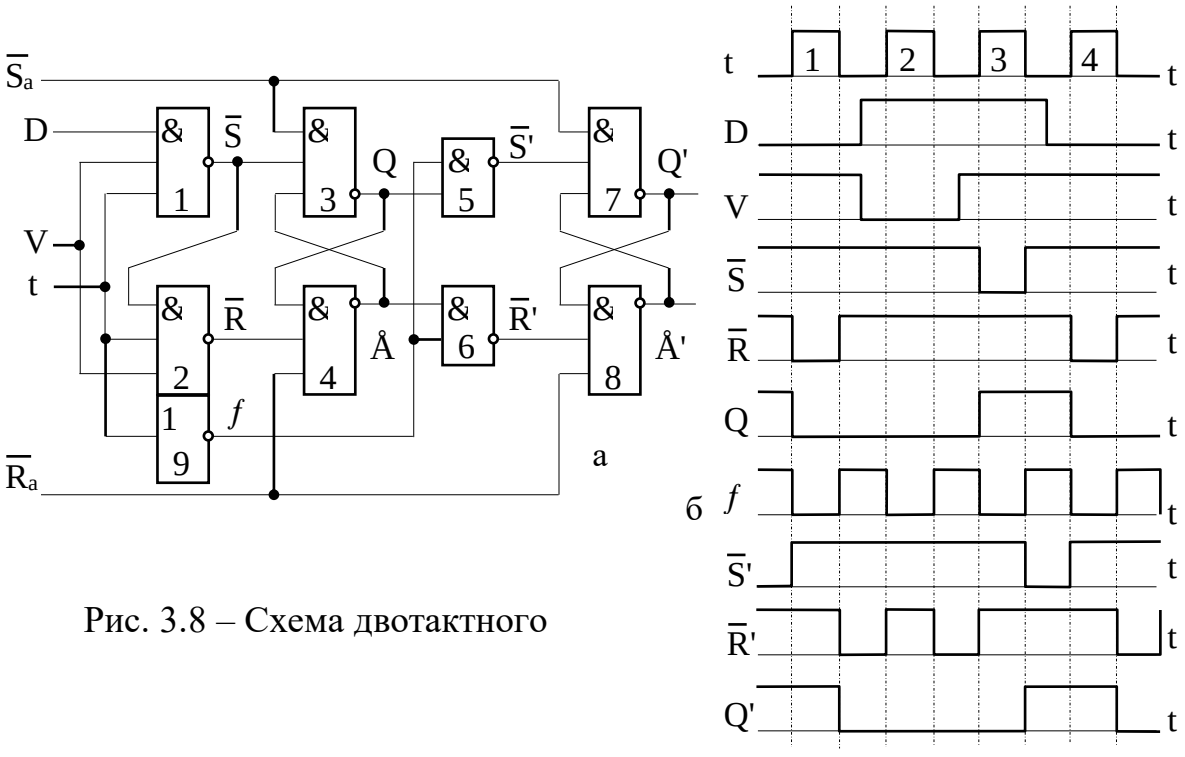


Рис. 3.8 – Схема двотактного

DV-тригера (а) і часова
діаграма його роботи (б)

Для синтезу двотактного тригера спочатку за наведеною вище методикою синтезується відповідний однотактний тригер, до виходу якого підключається синхронний RS-тригер, тактований синхросигналом $\bar{t}$. При цьому у двотактних Т- та JK-тригерах зворотні зв'язки організуються з виходів синхронного RS-тригера (Q' та $\overline{Q}'$).

3.8 Перетворення (метаморфози) тригерних схем

Розглянемо, як можна, маючи один тип тригера, збудувати на його основі інший. У загальному випадку це можна зробити так само, як вище будувались різноманітні типи тригерів на основі асинхронного RS-тригера, тобто скласти СТП тригери, що синтезуються, і УТП заданого та за ними визначити і реалізувати функції збудження відомого елементарного автомата. Для важливіших практичних випадків приклади перетворень показані на рис. 3.9.

Закони функціонування та принципи побудови структурних автоматів – тригерів RS-, JK-, DV-, D-, T-типів – розглянуті на прикладі асинхронного RS-тригера на ЛЕ І-НЕ. При використанні інших ЛЕ (наприклад, АБО-НЕ, І-АБО-НЕ та ін.) схемна реалізація тригерів, природно, буде відрізнятися від наведених. Конфігурація схеми тригера також залежить від конкретного варіанта СТП і АТП, способу довизначення функцій збудження, від числа входів ЛЕ, від використаної технічної системи ЛЕ та ін.

[3; 5; 7; 11-17]

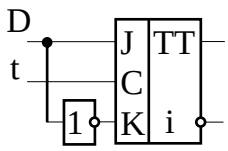
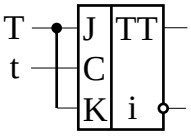
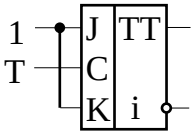
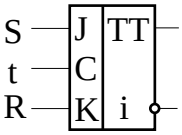
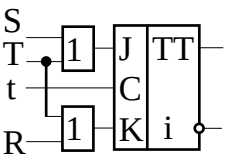
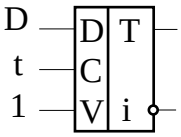
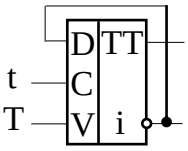
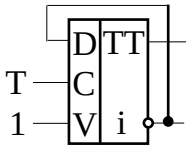
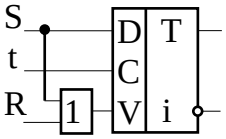
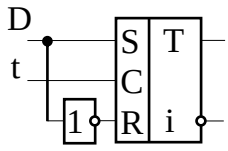
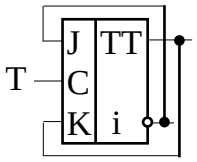
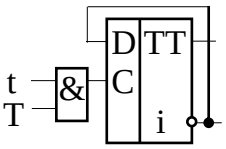
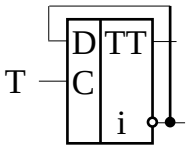
D-тригер	Синхронний Т-тригер	Асинхронний Т-тригер	RS-тригер	RST-тригер
Перетворення JK-тригера				
				
Перетворення DV-тригера				
				
Перетворення RS-тригера				
				
Перетворення D-тригера				
				

Рис. 3.9 – Перетворення тригерних схем

Лабораторна робота № 6

СИНТЕЗ ТА ДОСЛІДЖЕННЯ РЕГІСТРІВ ЗБЕРІГАННЯ І ЗСУВУ

1 Мета лабораторної роботи:

– вивчити основні вузли цифрової техніки – регістри зберігання та зсуву (тільки в один бік та реверсивних);

– оволодіти методом синтезу багатофункціональних регістрів зсуву;

– набути практичних навичок щодо набору схеми на лабораторному стенді, наладки та експериментального дослідження регістрів.

2 Практична частина роботи

2.1 Домашнє завдання та підготовка до виконання лабораторної роботи

1. Вивчити основні теоретичні положення до лабораторної роботи, підготуватися до відповідей на контрольні запитання та завдання.

2. Ознайомитися з порядком проведення лабораторної роботи, описом лабораторного стенду та наладкою схеми.

3. Провести синтез схеми заданого варіанта (табл. 4.1) чотирирозрядного багатофункціонального регістра зсуву. Оскільки ЛЕ серії 155 (а також серій 133, 555, тощо) реалізують функції І-НЕ (або І-АБО-НЕ), то синтез комбінаційної схеми треба проводити для зазначених базисів. Синтез реалізується табличним або графічним способом. Схема регістра має бути доповнена ланцюгами паралельної установки довільного коду з використанням тумблерного регістра та асинхронних установочних входів тригерів.

4. Побудувати схему замкненого у кільце регістра зсуву, тобто вихід регістра закомутувати на його вхід. В цьому випадку серія імпульсів, що подаються на шину зсуву, буде здійснювати режим циркуляції інформації, що зсовується, всередині регістру.

2.3 Порядок виконання роботи

1. Провести синтез схеми багатофункціонального регістра зсуву відповідно до заданого варіанта завдання.

2. Провести комутацію схеми реверсивного регістра зсуву на знеструмленому лабораторному стенді.

3. Перевірити роботу регістра у статичному режимі. Провести наладку, пошук та усунення несправностей у схемі регістра.

Таблиця 4.1 – Таблиця варіантів завдань

Варіант	Мікро-операції	Тип тригера	Початковий код	Варіант	Мікро-операції	Тип тригера	Початковий код
1	П31, Л31	JK	1001	23	П33, Л32	T	0101
2	П31, Л31	T	1001	24	П33, Л33	DV	1101
3	П31, Л31	DV	1011	25	П33, Л33	JK	1101
4	П31, Л31	JK	1011	26	П33, Л33	T	1101
5	П31, Л31	T	1011	27	П31, П32	DV	0011
6	П31, Л31	DV	1110	28	П31, П32	JK	0011
7	П31, Л31	JK	1110	29	П31, П32	T	0011
8	П31, Л31	T	1110	30	П32, П33	DV	0111
9	П32, Л31	DV	1010	31	П32, П33	JK	0111
10	П32, Л31	JK	1010	32	П32, П33	T	0111
11	П32, Л31	T	1010	33	П31, П33	DV	0100
12	П32, Л32	DV	1100	34	П31, П33	JK	0100
13	П32, Л32	JK	1100	35	П31, П33	T	0100
14	П32, Л32	T	1100	36	Л31, Л32	DV	1101
15	П32, Л33	DV	0110	37	Л31, Л32	JK	1101
16	П32, Л33	JK	0110	38	Л31, Л32	T	1101
17	П32, Л33	T	0110	39	Л32, Л33	DV	0101
18	П33, Л31	DV	0010	40	Л32, Л33	JK	0101
19	П33, Л31	JK	0010	41	Л32, Л33	T	0101
20	П33, Л31	T	0010	42	Л31, Л33	DV	1101
21	П33, Л32	DV	0101	43	Л31, Л33	JK	1101
22	П33, Л32	JK	0101	44	Л31, Л33	T	1101

4. Перевірити роботу регістра у динамічному режимі. Збудувати часову діаграму роботи закільцьованого регістра для заданого (або скоректованого) коду.

2.4 Зміст протоколу

Оформлення протоколу повинно розпочинатися при домашній підготовки до лабораторної роботи. У протокол вносять:

1. Титульний список (назва роботи, дата, прізвища членів бригади та ін.).
2. Мета та стислий опис роботи і завдання.
3. Кодовану таблицю переходів *i*-го розряду регістра.
4. Діаграми Вейча функцій збудження заданого типу тригера.
5. Схему реверсивного регістра зсуву.
6. Часові діаграми роботи регістра.

2.4 Контрольні запитання та завдання

1. Які функції (мікрооперації) можуть виконувати регістри?
2. Функціональна класифікація регістрів.
3. Покажіть принципову схему тригера на ЛЕ І-НЕ з синхронним S-входом та асинхронним інверсним $\overline{R}$-входом.
4. Дайте визначення парафазного представлення інформації.
5. Наведіть схеми регістрів зберігання з паралельним прийомом інформації. Поясніть їх роботу.
6. Приведіть схеми регістрів зберігання з парафазним прийомом інформації. Поясніть їх роботу.
7. Приведіть схеми регістрів зберігання з видаванням прямого чи зворотного коду, з парафазним видаванням коду. Поясніть їх роботу.
8. Наведіть схему регістра зберігання на синхронних RS-тригерах для почергового паралельного прийому слів з двох (або більше) регістрів. Поясніть її роботу.
9. Як, використовуючи синхронні та асинхронні входи RS-тригерів, можна виконати почерговий прийом слів у даний регістр зберігання з двох інших? Наведіть схему.
10. Як реалізувати почерговий прийом слів з різних місць цифрового пристрою на один і той же самий регістр, використовуючи мультиплексор?
11. В чому відміна замкненого в кільце регістра зсуву від розімкненого?
12. Наведіть схеми та поясніть роботу чотири та двотактних регістрів зсуву на однотактних тригерах.
13. Як здійснюється зсув за допомогою “навскісної передачі” слів? Нарисуйте відповідну структурну схему.
14. Наведіть схеми та поясніть роботу нереверсивних регістрів зсуву на двотактних тригерах різноманітних типів.
15. Методологія та алгоритм синтезу реверсивних багатофункціональних регістрів зсуву.
16. Як будується кодована таблиця переходів для і-го розряду реверсивного регістра зсуву?
17. Основна особливість синтезу цифрових автоматів на DV-тригерах.
18. Графи і графічний синтез реверсивних регістрів зсуву.
19. Як реалізується на регістрах зсуву перетворення паралельного коду в послідовний і навпаки?

2.5 Прилади, використані у роботі

Універсальний лабораторний стенд (УЛС), опис якого наведено у підрозділі 1.4 лабораторної роботи № 1.

2.6 Наладка регістра та усунення несправностей

Наладка схеми регістра проводиться в статичному режимі за допомогою ГПІ. Для цього необхідно зібрати схему, показану на рис. 4.1.

Послідовність досліджень така. В початковому стані вивід “ВК” (видавання коду тумблерного регістра) підключено до одного з ГПІ. На тумблерному регістрі набирають код, який необхідно занести в досліджуваний регістр. Натиснута клавіша генератора відповідає логічній одиниці. Натисканням клавіші ГПІ набраний код заноситься в регістр зсуву, який перевіряється, за асинхронними установочними входами його тригерів. Натискаючи після цього клавішу другого ГПІ, ввімкнутого до лінії зсуву, за світлодіодами тригерів перевіряють правильність зсуву. Сигнали управління x та $\overline{x}$ подаються з одного із розрядів вільного клавішного регістра.

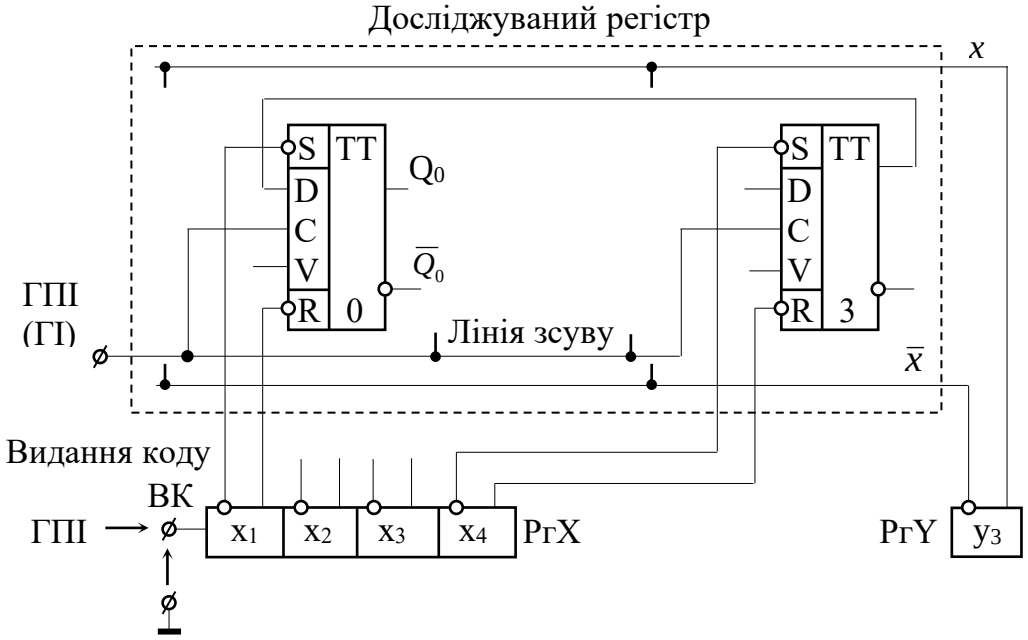


Рис. 4.1 – Схема дослідження регістра

Треба звернути увагу на те, що не будь-який код дозволяє провести повну перевірку регістра. Код при зсуві повинен перемикати всі тригери регістра як з 1 в 0, так і із 0 в 1. Наприклад, початковий код 1010 для чотирирозрядного кільцевого регістра зсуву на два розряди праворуч або ліворуч не забезпечує повної перевірки. При виявленні коду, який не забезпечує повну перевірку регістра, слід самостійно підібрати необхідний код та узгодити його з викладачем.

Якщо у статичному режимі регістр зсуву функціонує нормально, то переходять до дослідження схеми у динамічному режимі. Для цього, після занесення коду в досліджуваний регістр по імпульсах ГПІ, до шини зсуву підключають вихід генераторів імпульсів ГІ. За осцилографом перевіряють правильність зсуву на всіх частотах генератора ГІ. Дослідження закінчується побудовою часової діаграми роботи кільцевого регістра для таких точок схеми: шини зсуву, шини управління зсувом (x та $\overline{x}$), виходи всіх ЛЕ та тригерів регістра.

Якщо у статичному режимі виявлена помилкова робота регістра, то спочатку необхідно переконатися в правильному синтезі схеми, потім, якщо

помилки у синтезі немає, локалізувати та конкретизувати несправність у набраній схемі із тим, щоб її усунути.

3 Основні теоретичні положення

3.1 Введення

Регістром (РГ) називають композицію (упорядковану послідовність) елементарних автоматів та допоміжних комбінаційних схем, призначених для прийому, зберігання і видавання інформації, а також для виконання деяких операцій її перетворення.

В загальному випадку регістри виконують такі функції:

– установка регістра в нуль (скидання регістру);

– прийом слова з інших цифрових пристроїв (регістрів, лічильників, суматорів);

– передача слова у другий регістр, лічильник, суматор;

– перетворення прямого коду слова в зворотний та навпаки;

– зсув ліворуч чи праворуч на потрібну кількість розрядів;

– перетворення послідовного коду слова в паралельний та навпаки.

У окремому випадку схеми конкретних регістрів можуть реалізувати лише деякі з перелічених функцій. Для побудови регістрів використовують різноманітні типи елементарних автоматів (RS-, D-, T-, DV-, JK-тригери).

Залежно від функцій, що виконуються, регістри класифікуються таким чином:

– за видом реалізації основної функції – накопичувані (регістри зберігання) та зсувові (регістри зсуву);

– за введенням-виведенням інформації – паралельні, послідовні та комбіновані (паралельно-послідовні);

– за напрямом передачі інформації – нереверсивні (однонапрямлені чи прості) та реверсивні.

Для однозначного розуміння прийнято зсув інформації в бік старших розрядів називати зсувом ліворуч, а зсув в бік молодших розрядів зсувом праворуч.

Регістром *зберігання* (нагромаджуючим) називають такий регістр, що реалізує перші чотири з перелічених функцій Основне призначення регістрів зберігання – паралельний прийом багаторозрядних слів інформації та збереження їх протягом необхідного часу. Функціональний склад таких регістрів – це звичайно набір найпростіших асинхронних чи синхронних RS-тригерів, що мають загальну шину скидання. Як правило, інформація вводиться в RS-тригери через систему логічних елементів (ЛЕ) (І, І-НЕ), на яки одночасно подається сигнал прийому інформації (сигнал управління). Записана в регістрі інформація знімається також крізь додаткові ЛЕ.

Регістри *зсуву* призначені для більш складної обробки інформації за рахунок її зсуву за сигналами синхронізації. Наявність в регістрах зсуву можливості як паралельного, так і послідовного прийому інформації дозволяє

виконувати на них послідовно-паралельне перетворення кодів слів. Ця функція є однієї з основних в мережних картах, що являються апаратнім забезпеченням локальних комп'ютерних мереж (ЛКМ).

Функціональна структура регістрів зсуву значно складніше структури регістрів зберігання.

3.2 Регістри зберігання (накопичувані)

Регістри зберігання – досить прості пристрої і їх структура достатньо прозора. Тому для синтезу регістрів зберігання цілком достатньо словесного опису, за яким будується схема.

На рис. 4.2, а показано схему регістра зберігання РгА з паралельним прийомом інформації. Слово $X = x_0 x_1$, що підлягає прийому в регістр, паралельно подається на один з двох входів усіх логічних елементів І. На входи скидання R усіх тригерів спочатку подається керуючий сигнал установки нуля У0. Після подачі сигналу скидання всі тригери регістру будуть знаходитися у нульовому стані до моменту появи керуючого сигналу прийому слова в регістр ПрРгА. В тих розрядах, де $x_i = 1$, відбудеться установка тригерів в одиничний стан. Там само, де $x_i = 0$, нульовий стан тригерів збережеться.

Для регістрів із тригерів з інверсними статичними входами використовують схему, показану на рис. 4.2,б. Як видно, для реалізації регістра зберігання з асинхронних RS-тригерів потрібно включити у кожний розряд регістра по одному ЛЕ І або І-НЕ. Застосування синхронних тригерів дозволяє обійтися без них. В цьому випадку керуючий сигнал скидання регістра подається на асинхронні входи $\overline{R}$ усіх тригерів, а сигнал прийому слова ПрРгА подається на їх входи синхронізації C (рис. 4.2,в).

Всі розглянуті вище схеми регістрів зберігання здійснюють прийом інформації за два такти. Скоротити час прийому до одного такту дозволяє так зване парафазне подання інформації.

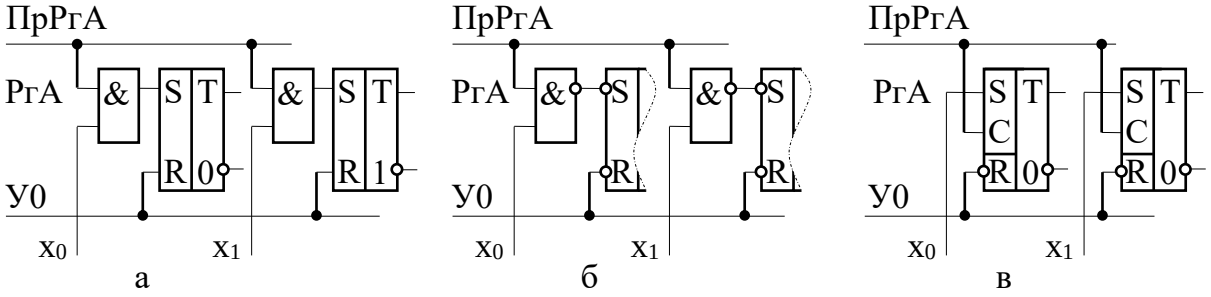


Рис. 4.2 – Паралельний прийом слів в регістри з асинхронних (а, б) та синхронних (в) тригерів

Слово X представляється парафазним кодом коли кожний його i-й розряд відображається прямим x_i та інверсним $\overline{x}_i$ значеннями. В парафазному

коді обов'язково одне із значень (x_i або $\bar{x}_i$) дорівнює одиниці, друге – нулю. Подані на установні входи тригера ці сигнали спільно встановлюють його у потрібний стан незалежно від стану, в якому він знаходився раніше. Схеми регістрів зберігання з парафазним прийомом інформації показані на рис. 4.3.

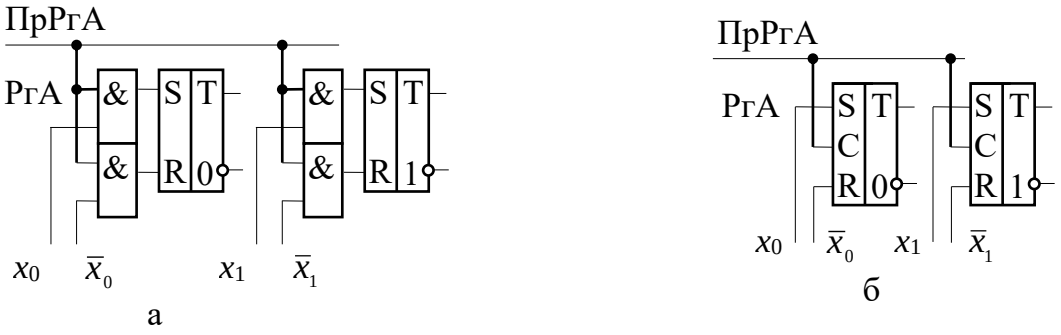


Рис. 4.3 – Парафазний прийом слів в регістри з асинхронних (а) та синхронних (б) тригерів

Схеми видавання інформації в прямому чи зворотному коді показана на рис. 4.4,а. Тут ВидРгАП – керуючий сигнал видавання з регістра РгА прямого коду, ВидРгАЗ – сигнал видавання зворотного коду. Очевидно, що одночасна поява цих двох керуючих сигналів має бути заборонена, тобто повинна виконуватися логічна умова ВидРгАП & ВидРгАЗ = 0. Схему регістра зберігання з парафазним видаванням коду показано на рис. 4.4,б. Звичайно операція видавання коду з регістра суміщається з прийомом коду на інший регістр (або інший цифровий пристрій), тому всі керуючі сигнали передачі можна позначати як сигнали прийому інформації (рис.4.5).

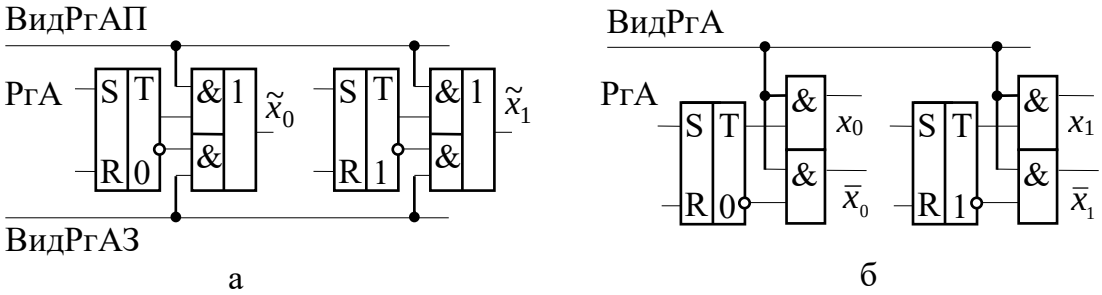


Рис. 4.4 – Способи видавання кодів слів з регістрів: а – прямого чи зворотного, б – парафазного

Часто в цифрових пристроях виникає необхідність прийняти інформацію в даний регістр почергово з двох (або більше) інших регістрів. В цих випадках перед кожним тригером даного регістра встановлюють логічну схему, що виявляє за конкретним керуючим сигналом, з якого регістра повинна вводитися інформація в даний. На рис. 4.5 показані схеми почергового прийому інформації з двох регістрів в регістр зберігання Рг3. Функцію збудження D_{i3} *i*-го розряду регістра Рг3, що складається з D-тригерів (рис. 4.5,а), визначають за таким виразом:

$$D_{i3} = Q_{i1} \cdot \text{ПрРг31} \vee Q_{i2} \cdot \text{ПрРг32},$$

де Q_{i1}, Q_{i2} – вихідні сигнали *i*-х тригерів регістрів Рг1 та Рг2 відповідно;
ПрРг31 – керуючий сигнал прийому інформації на Рг3 з Рг1;
ПрРг32 – керуючий сигнал прийому інформації на Рг3 з Рг2.

Схему почергового прийому з використанням як синхронних, так і асинхронних входів тригерів приймального регістра Рг3 показано на рис. 4.5,б. Парафазне представлення інформації та додаткові ЛЕ І-НЕ тут використовують для асинхронної частини тригерів. Синхронна частина у D-тригерів (так само як і у Т-тригерів) має тільки один інформаційний вхід. Однак D-тригер, в силу своїх властивостей, дозволяє приймати інформацію без попереднього скидання в нуль, що значно спрощує структуру регістра при збереженні його швидкодії.

На рис. 4.5,в показана організація почергового прийому слів із двох регістрів Рг1 та Рг2 в регістр Рг3 з використанням комутуючих схем – мультиплексорів. Оскільки для сигналів прийому має дотримуватися логічна умова ПрРг31 & ПрРг32 = 0, керувати комутацією можна тільки одним з зазначених сигналів, наприклад, сигналом прийому ПрРг32, як показано на рисунку. За відсутністю активного рівня сигналу ПрРг32 на входи регістра Рг3 будуть комутуватися виходи регістра Рг1, за наявності його – виходи регістра Рг2.

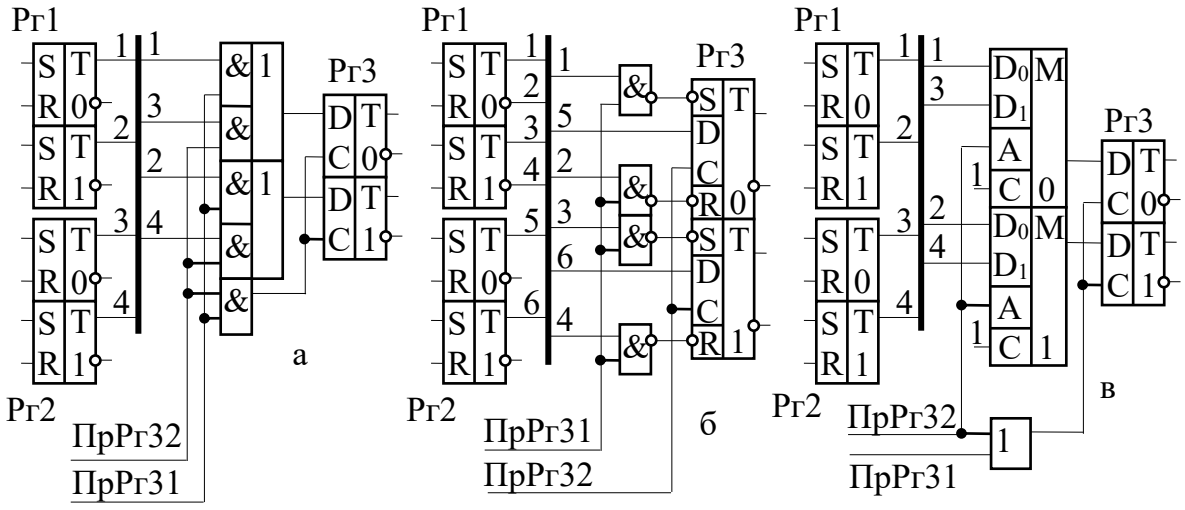


Рис. 4.5 – Почерговий прийом слів з двох регістрів: а) – на регістр з D-тригерів; б) – з використанням асинхронних входів тригерів; в) – за допомогою мультиплексорів.

Умовні графічні позначення (УГП) регістрів на функціональних та структурних схемах показані відповідно на рис. 4.6 та 4.7,в.

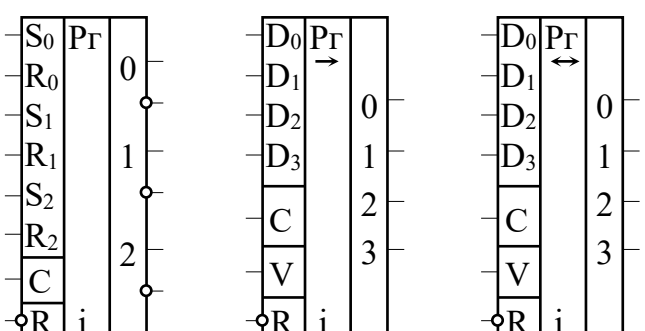


Рис. 4.6 – Умовні графічні позначення (УГП) регістрів на функціональних схемах: а – регістр зберігання; б – зсувовий праворуч; в – реверсивний

3.3 Зсувові регістри

При виконанні різноманітних математичних операцій в арифметико-логічному (та інших) пристроях комп'ютера широке застосування знаходять різноманітні види зсувів інформації, використані в якості мікрооперацій при виконанні складних операцій, таких, наприклад, як множення, ділення та ін. Операція зсуву виконується на регістрах зсуву. Регістр зсуву являє собою схему на тригерах, з'єднання між якими, що називаються ланцюгами зсуву, забезпечують передачу двійковій інформації від одних тригерів регістру до інших.

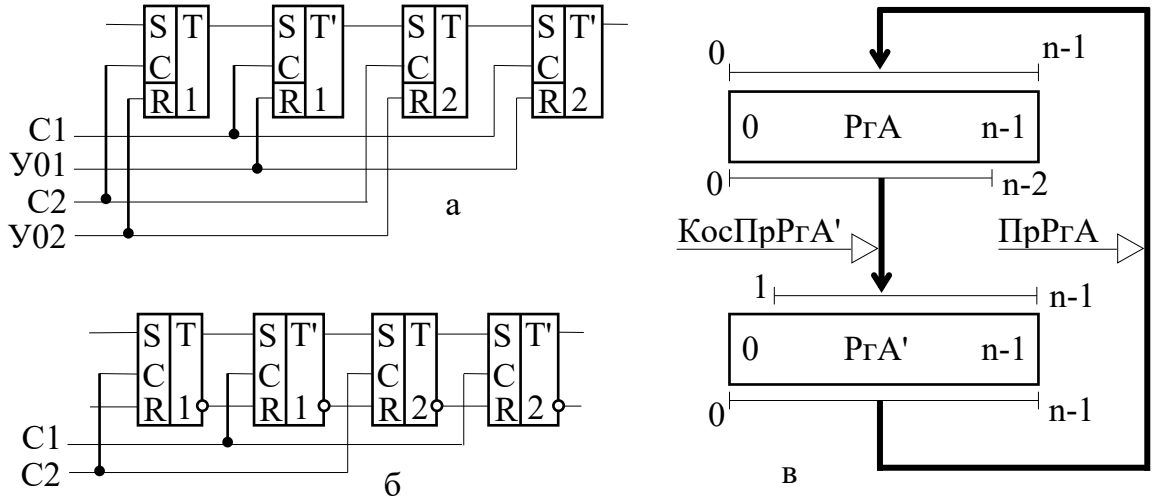


Рис. 4.7 – Регістри зсуву на однотактних тригерах: а) – зсув за чотири такта; б) – двотактний зсув; в) – "навскісний" прийом слова

Зсуви можуть виконуватися на один чи декілька розрядів праворуч чи ліворуч. В загальному випадку операція зсуву коду в регістрі на *i* розрядів праворуч означає, що кожний тригер цього регістра T_j повинен передати інформацію, що зберігається в ньому, тригеру T_{j+i} та після цього прийняти інформацію від тригера T_{j-i}. Звичайно старші (молодші) розряди регістра, що звільняються при зсуві, можуть зберігати колишні значення, чи заповнятися двійковою константою, чи приймати інформацію ззовні. При цьому дані, що висуваються з молодших (старших) розрядів регістра, втрачаються або

передаються на зовнішні ланцюги. В деяких випадках регістр може бути замкнений в кільце так, що інформація, яка зсувається, циркулює всередині регістру, такий регістр називається *кільцевим*.

Основна трудність, що виникає при побудові регістрів зсуву, полягає в тому, що при виконанні зсуву кожний розрядний тригер повинен одночасно видавати свій поточний стан в наступний розряд та приймати дані з попереднього розряду. Для правильного обміну сигнали видавання та прийому інформації повинні бути рознесені в часу, інакше використання однотактних тригерів стає неможливим із-за їх невизначеного перемикання.

З метою рознесення в часі цих сигналів в регістрах на інтегральних елементах застосовується багатотактний спосіб обміну інформацією. Правильний обмін інформацією при цьому забезпечується введенням допоміжних (додаткових) тригерів. Кожний розряд регістру утримує два тригера – основний Т та допоміжний Т '. На рис. 4.7,а показано схему регістра зсуву праворуч на однотактних тригерах. Зсув праворуч на один розряд здійснюється у такій схемі за чотири такти. В першому такті допоміжні тригери Т ' скидаються у 0 сигналом У01, потім в другому такті відбувається прийом інформації з основних тригерів в допоміжні по сигналу С1. В третьому такті сигналом У02 скидаються основні тригери та в четвертому, по сигналу С2, зсунута інформація знов записується в основні тригери Т регістра.

Однак з метою підвищення швидкодії та надійності роботи регістрів зсуву звичайно в них використовують парафазні коди слів. В цьому випадку підвищення швидкодії досягається за рахунок виключення мікрооперацій скидання регістру, а підвищення надійності – за рахунок надмірності в кількості каналів передачі інформації. На рис. 4.7,б показано схему регістра зсуву із парафазним представленням інформації. Тут аналогічна операція зсуву на один розряд праворуч здійснюється за два такти: по сигналу С1 інформація заноситься в допоміжні тригери регістра, а по сигналу С2 зсунута інформація знов приймається в основні тригери.

Якщо виділити основні та допоміжні тригери у окремі регістри РгА та РгА' (рис. 4.7,в), отже, зсув інформації можна уявити за допомогою так званої “навскісної передачі” слова з РгА в РгА' за керуючим сигналом КосПрРгА' (аналогічний сигнал С1 на рис. 4.7,б) та прямим поверненням слова з РгА' у РгА по сигналу ПрРгА (аналогічний сигнал С2 на рис. 4.7,б).

Реальні двотактні тригери в інтегральному виконанні являють собою послідовне з’єднання двох однотактних синхронних тригерів. Внутрішня структура таких тригерних схем вже передбачає розділення в часі етапів прийому вхідної інформації та зміни вихідного сигналу тригера. По фронту синхронізуючого імпульсу приймається інформація, а за його спадом – зміна вихідного сигналу. Це дозволяє обійтися при побудові регістрів зсуву одним двотактним тригером на кожний розряд.

На рис. 4.8,а показано схему регістра зсуву на один розряд праворуч на JK-тригерах. Для реалізації зсуву праворуч необхідно з’єднати виходи тригера *i*-го розряду з входами J та K тригера *i+1*-го розряду. Синхронізуючі

входи всіх тригерів об’єднують та утворюють лінію правого зсуву (ПЗ). Зсув на один розряд виконується подачею одного імпульсу на лінію зсуву. В момент надходження цього імпульсу значення сигналів на входах J та K кожного тригера визначають його перехід в новий стан, яке кожний тригер приймає по закінченні імпульсу зсуву. Схеми регістрів зсуву на двотактних тригерах інших типів показані на рис. 4.8,б,в,г.

Якщо регістр виконує мікрооперацію зсуву на один розряд, то зсув слова на *k* розрядів здійснюється послідовним виконанням *k* мікрооперацій. Для зменшення часу, необхідного для *k*-розрядного зсуву, можна реалізувати у регістрі ланцюги зсуву слова відразу на *k* розрядів.

Регістр зсуву реалізує перетворення послідовного коду в паралельний та навпаки. При першому перетворенні запис коду у регістр проводиться синхронно зі зсувом його вмісту праворуч, якщо послідовний код надходить зі старших розрядів (див. рис.4.7,а), або ліворуч, якщо код надходить з молодших розрядів. Після заповнення всіх розрядів регістра відбувається паралельне видавання коду через елементи І, підключені до виходів тригерів регістра (див., наприклад, рис. 4.4).

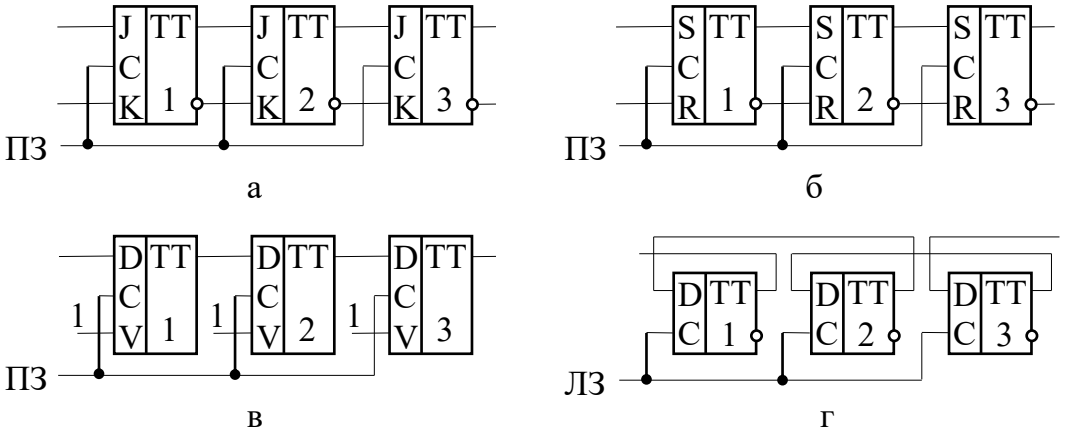


Рис. 4.8 – Схеми нереверсивних регістрів зсуву на двотактних тригерах різноманітних типів

При перетворенні паралельного коду в послідовний інформація заноситься в регістр паралельним кодом, а після цього йде серія з *n* сигналів зсуву, де *n* – число розрядів в слові. Послідовний код зчитується з тригера молодшого розряду при зсуві праворуч чи з тригера старшого розряду при зсуві ліворуч.

Відзначимо, що занесення паралельного коду в регістр зсуву легко реалізується при використанні асинхронних входів тригерів R_a, S_a. Введення паралельного коду по синхронним входам тригерів призвело б до ускладнення схеми регістру зсуву, бо при занесенні коду необхідно логічно від’єднувати ланцюги зсуву.

В цифрових пристроях часто використовують зсувові регістри, на яких реалізується як зсув ліворуч, так і зсув праворуч. Такі регістри називаються

реверсивними регістрами зсуву. Очевидно, на одному і тому ж регістрі можливо організувати виконання деяких зсувів як ліворуч, так і праворуч.

3.4 Синтез регістрів зсуву

Роботу багатофункціональних регістрів зсуву, що виконують декілька зсувів як ліворуч, так і праворуч, можна описати, користуючись апаратом теорії цифрових автоматів.

В загальному випадку реверсивний багатофункціональний регістр містить набір тригерів якого-небудь типу (звичайно, двотактних); шину зсуву, по якій на входи синхронізації всіх тригерів подаються імпульси зсуву; шину фізичних входів, по яких подаються сигнали управління x_i, які визначають потрібну мікрооперацію з набору мікрооперацій зсуву, що виконуються регістром; логічні схеми, що складають ланцюги зсуву між тригерами. Сигнали управління настроюють позначений ланцюг зсуву від одних тригерів до інших та забороняють роботу інших ланцюгів. Задача синтезу схеми регістра зсуву при заданому типі тригера полягає в складанні функцій збудження кожного тригера на основі кодованої таблиці переходів (КТП) регістра та умовної таблиці переходів (УТП) обраного типу елементарного автомату.

Розглянемо метод синтезу на конкретному прикладі. Нехай потрібно спроектувати реверсивний регістр зсуву праворуч на два розряди та ліворуч на один розряд. У вигляді елементарного автомата оберемо універсальний DV-тригер.

Визначимо число фізичних входів з формули $\ell = log_2 m$, де m – кількість мікрооперацій зсуву, які виконуються регістром, що в даному разі дорівнює двом. Тоді число фізичних входів $\ell = log_2 2 = 1$. Отже, управління регістром можна здійснювати однією логічною змінною x.

Закодуємо мікрооперації зсуву значеннями цієї змінної (табл. 4.2). Тут ПЗ2, ЛЗ1 – мікрооперації зсуву відповідно праворуч на два розряди та ліворуч на один розряд.

Таблиця 4.2 – ТКВхС

МО	x
ПЗ2	0
ЛЗ1	1

Оскільки структура будь-якого регістру регулярна, то достатньо отримати вирази функцій збудження тільки для одного тригера та розповсюдити отримані вирази на всі n тригерів регістра. Однак в синтезі у даному випадку мають брати участь ще два тригери, стани яких, відповідно мікрооперація регістра, передаються в даний i-й тригер. Це старший тригер T_{i-2}, стан якого передається в даний тригер при зсуві праворуч на два розряду, та молодший тригер T_{i+1}, стан якого передається в T_i при лівому зсуві на один розряд.

Кодована таблиця переходів КТП проектованого реверсивного регістра наведена у (табл. 4.3). Стрілками показано як формально відзначати стан переходу i-го тригера при зсуві праворуч (див. рядок 2) та ліворуч (див. рядок 3).

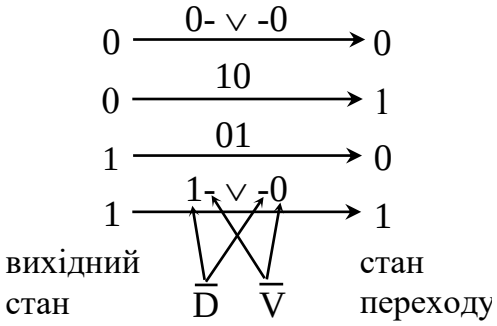
Таблиця 4.3 – КТП Рг 1↔2

Q_{i-2}, Q_i, Q_{i+1} x		000	001	010	011	100	101	110	111
ПЗ2	0	0	0	0	0	1	1	1	1
ЛЗ1	1	0	1	0	1	0	1	0	1

На основі кодованої таблиці переходів регістра (табл. 4.3) та умовної таблиці переходів DV-тригера (табл. 4.4) будується таблиця функцій збудження *i*-го елементарного автомата (табл. 4.5).

Синтез дискретних автоматів на DV-тригерах специфічний, оскільки деякі переходи DV- тригера мають по два варіанта функцій збудження. Їх необхідно обидва заносити в таблицю функцій збудження (див. табл. 4.4 та 4.5) для подальшого використання при мінімізації.

Таблиця 4.4 – УТП DV-тригера



Таблиця 4.5 – Таблиця функцій збудження – ТФЗ

Q_{i-2}, Q_i, Q_{i+1} x		000	001	010	011	100	101	110	111
ПЗ2	0	0--0	0--0	01	01	11	11	1--0	1--0
ЛЗ1	1	0--0	11	01	1--0	0--0	11	01	1--0

D V D V

Функції збудження *i*-го тригера D_i та V_i з можливими варіантами витягаються з ТФЗ (табл. 4.5) та заносяться відразу в наступні діаграми Вейча, що представлені на рис. 4.9.

При мінімізації функцій D_i та V_i треба врахувати, що вибираючи конкретні значення функцій D_i (з двох можливих) в якому-небудь полі D_i-діаграми, тим самим визначаємо значення функції у відповідному полі V_i-діаграми та навпаки. Наприклад, якщо у полі для набору з десятковим номером 12 D_i-діаграми вибирається значення 0, то в тому ж полі V_i-діаграми має стояти риска (заборонений набір). Таким чином, D_i- та V_i-діаграми є

зв’язаними, що приводить до необхідності сумісного розгляду цих діаграм при мінімізації функцій збудження DV-тригера. Сумісний вибір конкретних значень в варіантних полях визначається з умови одержання мінімальних виразів функцій D_i та V_i.

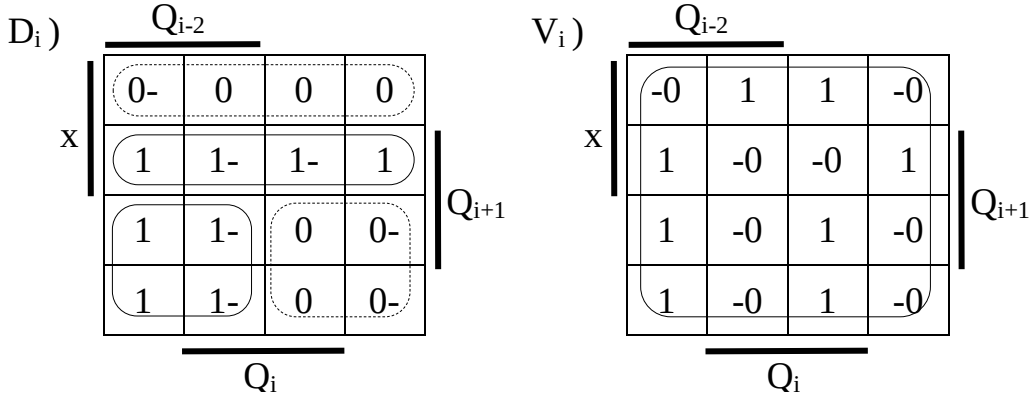


Рис. 4.9 – Мінімізація функцій збудження реверсивного регістру зсуву

МДНФ даних функцій отримуються, якщо взяти функцію $V_i = 1$, тобто в усіх варіантних полях V_i-діаграми залишаються заборонені набори (риски):

$$\begin{cases} D_i = x \cdot Q_{i+1} \vee \overline{x} \cdot Q_{i-2} = \overline{\overline{x \cdot Q_{i+1}} \cdot \overline{\overline{x} \cdot Q_{i-2}}} = \overline{x \cdot \overline{Q_{i+1}} \vee \overline{x} \cdot \overline{Q_{i-2}}}, \\ V_i = 1. \end{cases}$$

В виразі D_i ліва форма – диз’юнктивна чи перша операторна, призначена для реалізації на ЛЕ І та АБО; середня – друга операторна форма – для ЛЕ І-НЕ; права – сьома операторна форма – витягається з пунктирних петель діаграми Вейча та реалізується на логічних елементах І-АБО-НЕ.

На рис. 4.10 показано схему реверсивного регістра зсуву, що синтезувалася. Суцільними лініями позначені зв’язки для *i*-го тригера, одержані внаслідок синтезу, пунктирними лініями – з’єднання, що проводять за аналогією для інших тригерів на підставі регулярності структури регістра.

3.5 Графічний синтез регістрів зсуву

Синтез зсувового регістра можна проводити також за його графом. На рис. 4.11 показано граф розглянутого реверсивного регістра зсуву, збудованого за табл. 4.3. Тут кожний вузол графа відзначений станами регістра, *x* та $\overline{x}$ – коди сигналів відповідно лівого та правого зсувів. Оберемо у вигляді елементарного автомата D-тригер.

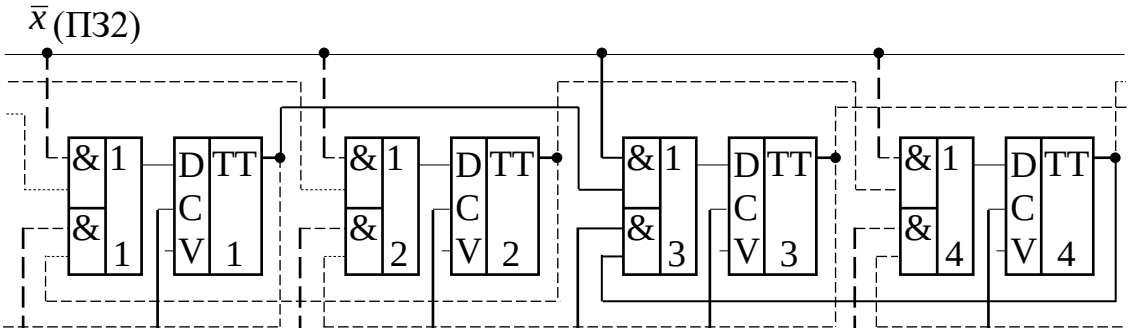


Рис. 4.10 – Схема реверсивного регістру зсуву на DV-тригерах

Графічний синтез розпочинається з позначення дуг графа символами приватних функцій збудження D-тригера. Згідно з властивістю D-тригера, символом D відзначаються дуги, що входять до вузлів, у яких $Q_i = 1$ (тут це вузли, у яких є одиниця посередині коду стану).

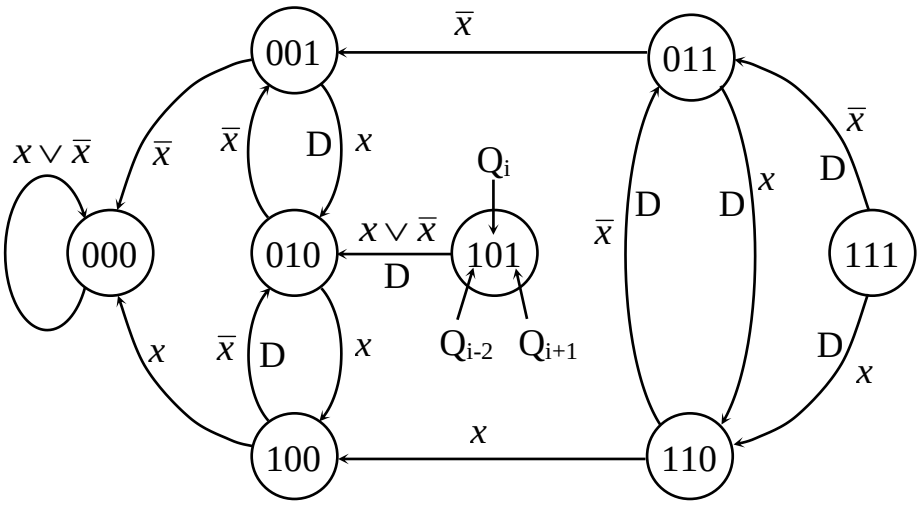


Рис. 4.11 – Граф реверсивного регістру зсуву, відмічений функціями збудження D-тригера

Після цього з відміченого графа витягається повна функція збудження D_i як диз'юнкція окремих функцій збудження. Приватні функції збудження, відповідні відміченим символом D дугам графа, являють собою набори, що складаються з коду вхідного сигналу, приписаного до даної дуги, та коду стану, з якого ця дуга виходить. ДДНФ функції збудження D_i у числовій формі для даного прикладу дорівнює

$$D_i = \vee (4, 9, 13, 5, 6, 11, 15, 7).$$

МДНФ даної функції, одержана за діаграмою Вейча, має наступний вигляд

$$D_i = x\, Q_{i+1} \vee \bar{x}\, Q_{i-2}\,.$$

Оскільки функції збудження D_i для реверсивних регістрів зсуву на D- та DV-тригерах отримались однаковими, отже й їх схеми (за вилученням типу тригера) також збігаються.

[3; 5; 11; 16; 17]

Перелік посилань

1. Препелиця Г.П. Комп’ютерна схемотехніка. Практикум: Навчальний посібник/ М-во освіти і науки; Одес. держ. екологічний ун-т. – Вид. 2-е, перероб. і доп. – Одеса: Екологія, 2008. – 252 с.
2. Препелиця Г.П. Схемотехніка ЕОМ. Практикум: Навчальний посібник – Одеса, 2002. – 208 с.
3. Препелиця Г.П. Комп’ютерна схемотехніка. Конспект лекцій. Одеса: Екологія, 2008. – 340 с.
4. Великий В.І., Препелиця Г.П. Мікропроцесорні системи обробки даних та управління в гідрометеорології: Навчальний посібник. – Одеса, Вид-во “ТЭС”, 2004. – 212 с.