**Волинський національний університет імені Лесі Українки**

Факультет інформаційних технологій і математики

Кафедра комп'ютерних наук та кібербезпеки

**Булатецький В. В., Булатецька Л. В., Собчук О. М.**

# АЛГЕБРА ЛОГІКИ ТА ПРОЕКТУВАННЯ ОСНОВНИХ ОПЕРАЦІЙНИХ ВУЗЛІВ

*навчальний посібник*

Луцьк 2021

**УДК 004.2(076.5)**
**Б 90**

*Рекомендовано до друку вченою радою*
*Волинського національного університету імені Лесі Українки*
*(протокол № 2 від 26.02.2021 р.)*

Електронне мережне навчальне видання

**Рецензенти:**

**Мартинюк О. С.** – доктор пед наук, професор кафедри експериментальної фізики, інформаційних та освітніх технологій Волинського національного університету імені Лесі Українки;

**Ліщина Н. М.** – кандидат технічних наук, доцент, завідувач кафедри інженерії програмного забезпечення Луцького національного технічного університету

**Б 90 Алгебра логіки та проектування основних операційних вузлів** [Електронний ресурс] : навч. посіб. / В. В. Булатецький, Л. В. Булатецька, О. М. Собчук; ВНУ ім. Лесі Українки. – Електронні текстові данні (1 файл: 3,69 Мбайт). – Луцьк : ВНУ ім. Лесі Українки, 2021. – 150 с.

Навчальний посібник розроблено для вивчення основних понять булевої алгебри при проектуванні та дослідженні вузлів і пристроїв комп'ютерних систем. Навчальне видання призначене для студентів, які вивчають архітектуру обчислювальних систем, архітектуру комп'ютерних систем та схемотехніку, що пов'язано з вивченням математичних методів побудови основних операційних вузлів обчислювальної техніки. Рекомендовано студентам, спеціальності 122 Комп'ютерні науки, 014 Середня освіта, 113 Прикладна математика та 125 Кібербезпека освітньо-професійних програм Комп'ютерні науки та інформаційні технології, Інформатика, Прикладна математика та Інформаційна безпека відповідно.

**УДК 004.2(076.5)**

**ЗМІСТ**

# ВСТУП

В даному навчальному посібнику подано матеріал практичного застосування математичної логіки та теорії автоматів при проектуванні та дослідженні основних компонентів обчислювальних систем. Даний посібник містить до кожної теми короткий виклад теоретичних відомостей, список завдань для поглибленого вивчення матеріалу та запитання для самоконтролю.

Роботи по проектуванню основних операційних вузлів пропонується виконувати, використовуючи відкритий програмний засіб Logisim. При дослідженні принципів функціонування основних вузлів цифрових пристроїв виникає проблема вибору оптимального за функціональними можливостями та доступного для широкого кола користувачів програмного забезпечення для виконання лабораторних робіт. За допомогою Logisim можна наочно ознайомити студентів із апаратною складовою обчислювальної техніки. Існує ряд комерційних продуктів, які з успіхом справляються з такими задачами, проте є або занадто складними для пересічного користувача, або вимогливими до ресурсів та дорогими в експлуатації. Logisim є навчальним інструментом для розробки і моделювання цифрових логічних схем. Завдяки простому інтерфейсу панелі інструментів та моделювання схем по ходу їх проектування, програмний засіб досить зручний. Його можна використовувати для проектування і моделювання цілих процесорів в освітніх цілях. Logisim використовують при викладанні різноманітних дисциплін, починаючи з логіки, завершуючи повними курсами з архітектури комп’ютерів. В посібнику основну увагу приділено вивченню внутрішньої будови цифрової техніки, тому знайомство з навиками роботи в програмі Logisim залишені на самостійне вивчення.

Посібник призначено для студентів закладів вищої освіти, які вивчають архітектуру обчислювальних систем. Даний предмет викладається для спеціальностей 122 Комп’ютерні науки, 014 Середня освіта (Інформатика), 113 Прикладна математика та 125 Кібербезпека.

# 1. ПРОЕКТУВАННЯ КОМБІНАЦІЙНИХ СХЕМ

При побудові й аналізі схем цифрової техніки інформація подається в двійковому вигляді. Сигнал може приймати два значення, які розглядаються як логічна «1» (сигнал високого рівня) або логічний «0» (сигнал низького рівня). Аналіз комбінаційних пристроїв і цифрових логічних схем проводять з допомогою булевої алгебри, яка оперує тільки двома поняттями: *істина* (логічна 1) і *хиба* (логічний 0). В результаті функції, які відображають інформацію приймають в кожний момент часу тільки значення 0 і 1. Таблиця в якій кожному набору вхідних сигналів ставиться у відповідність вихідний сигнал називається *таблицею істинності*.

## 1.1. Аксіоми алгебри логіки

*Логічні змінні*, які розглядаються в алгебрі логіки, можуть приймати тільки два значення – 0 або 1. В алгебрі логіки визначені: відношення *еквівалентності* (позначається знаком =) і операції: додавання (*диз'юнкція*), яка позначається знаком $\vee$(+), множення (*кон'юнкція*), яка позначається знаком & або крапкою, і *заперечення* (або інверсія), яка позначається рискою зверху або апострофом.

Алгебра логіки визначається наступною *системою аксіом*:

$$\begin{cases} x = 0, & якщо \quad x \neq 1, \\ x = 1, & якщо \quad x \neq 0; \end{cases} \quad (1.1) \qquad \begin{cases} \overline{0} = 1, \\ \overline{1} = 0; \end{cases} \quad (1.2)$$

$$\begin{cases} 1 \vee 1 = 1, \\ 0 \vee 0 = 0, \\ 0 \vee 1 = 1 \vee 0 = 1 \end{cases} \quad (1.3) \qquad \begin{cases} 0 \cdot 0 = 0, \\ 1 \cdot 1 = 1, \\ 1 \cdot 0 = 0 \cdot 1 = 0. \end{cases} \quad (1.4)$$

Аксіома (1.1) стверджує, що в алгебрі логіки розглядаються лише двійкові змінні (кожна змінна набуває лише одного з двох можливих значень), аксіома (1.2) визначає операцію заперечення, а аксіоми (1.3) і (1.4) – операції диз'юнкції та кон'юнкції відповідно.

## 1.2. Логічні вирази

*Логічним виразом* називають вираз, утворений з логічних змінних та логічних операцій кон'юнкції, диз'юнкції і заперечення з використанням логічних аксіом, в результаті обчислення якого отримують одне з *логічних значень*: істина або хиба.

Запис логічних виразів зазвичай здійснюють у *кон'юнктивній нормальній формі* (КНФ) або *диз'юнктивній нормальній формі* (ДНФ). У диз'юнктивній формі логічні вирази записуються як логічна сума логічних добутків, у кон'юнктивній формі – як логічний добуток логічних сум. Порядок дій такий же, як і у звичайних алгебраїчних виразів (в першу чергу – заперечення, потім – логічне множення і останньою дією – додавання).

З кожним логічним виразом пов'язана певна логічна функція, яка кожному є набору значень логічних змінних, що входять у вираз, ставить у відповідність його логічне значення.

## 1.3. Логічні тотожності

При перетвореннях логічних виразів використовуються л*огічні тотожності:*

$x \vee 1 = 1$; – логічне додавання до 1;

$x \cdot 1 = x$; – логічне множення на 1;

$x \vee 0 = x$; – логічне додавання до 0;

$x \cdot 0 = 0$; – логічне множення на 0;

$x \cdot \overline{x} = 0$; – закон протиріччя;

$x \vee \overline{x} = 1$; – закон виключення третього;

$x \vee x = x$; – закон ідемпотентності;

$x \cdot x = x$; – закон ідемпотентності;

$x \vee x \cdot y = x$; – закон поглинання;

$(x \vee y)x = x$; – закон поглинання;

$xy \vee x\overline{y} = x$; – закон склеювання;

$(x \vee y)(x \vee \overline{y}) = x$; – закон склеювання;

$x \vee \overline{x}y = x \vee y$; – закон заміщення;

$x(\overline{x} \vee y) = xy$; – закон заміщення;

$\overline{\overline{x}} = x$; – закон подвійного заперечення;

Закони де Моргана:

$\overline{x \vee y} = \overline{x} \cdot \overline{y}$;

$\overline{x \cdot y} = \overline{x} \vee \overline{y}$;

$x \vee y = \overline{\overline{x} \cdot \overline{y}}$;

$x \cdot y = \overline{\overline{x} \vee \overline{y}}$.

### 1.4. Логічні функції

Будь-який логічний вираз, складений з $n$ змінних $x_n$, $x_{n-1}$, …, $x_1$ за допомогою скінченної кількості операцій алгебри логіки, можна розглядати як деяку функцію $n$ змінних, аргументи якої набувають значень із множини {0, 1}, і результат також належить множині {0, 1}. Таку функцію називають *логічною* або *булевою*. Відповідно до аксіом алгебри логіки, функція може приймати, залежно від значення змінних, значення 0 або 1. Функція $n$ логічних змінних може бути визначена для $2^n$ наборів значень змінних, що відповідає всім можливим значенням $n$-розрядних двійкових чисел. Оскільки на кожному такому двійковому наборі сама функція може набувати одного із двох значень (0 або 1), то всього існує $2^{2^n}$ різних логічних функцій від $n$ аргументів. Кожні дві функції відрізняються одна від одної значенням принаймні в одному наборі. У випадку $n$=2 маємо $2^{2^n} = 2^{2^2} = 2^4 = 16$ різних логічних функцій, які подані в таблиці 1. Будемо вважати, що порядковий номер (індекс) логічної функції відповідає значенню самої функції при переведенні з двійкової системи числення в десяткову.

Як видно з табл. 1, функції $f_0$ *і* $f_{15}$ – константи, $f_3$ *і* $f_5$ – повторюють, а $f_{10}$ *і* $f_{12}$ – заперечують одну із змінних, $f_1$ *і* $f_7$ – кон'юнкція і диз'юнкція, які розглянуті раніше. Операція імплікації може бути виражена через основні логічні операції з використанням тотожності $x \to y = \overline{x} \vee y$, а еквіваленції –

$x \leftrightarrow y = x \cdot y \vee \overline{x} \cdot \overline{y}$ (тобто, змінні набувають одночасно однакових значень істинності – або обидві істинні, або хибні). Слід зауважити, що логічна функція $f_9$, яка відповідає операції еквіваленції, є інверсною до функції $f_6$ – додаванням за модулем 2.

Таблиця 1

**Значення булевих функцій двох аргументів *x* та *y***

| ***x*** | **0** | **0** | **1** | **1** | **Позначення функції** | **Назва функції** |
|---|---|---|---|---|---|---|
| ***y*** | **0** | **1** | **0** | **1** | | |
| $f_0(x,y)$ | 0 | 0 | 0 | 0 | 0 | константа 0 |
| $f_1(x,y)$ | 0 | 0 | 0 | 1 | $x \cdot y$ | кон'юнкція, логічне множення, "і" |
| $f_2(x,y)$ | 0 | 0 | 1 | 0 | $\overline{x \to y} = x \cdot \overline{y}$ | заборона за *y*, заперечення імплікації |
| $f_3(x,y)$ | 0 | 0 | 1 | 1 | $x$ | змінна *x* |
| $f_4(x,y)$ | 0 | 1 | 0 | 0 | $\overline{y \to x} = \overline{x} \cdot y$ | заборона за *x,* заперечення імплікації |
| $f_5(x,y)$ | 0 | 1 | 0 | 1 | $y$ | змінна *y* |
| $f_6(x,y)$ | 0 | 1 | 1 | 0 | $x \oplus y = x\overline{y} \vee \overline{x}y$ | сума за mod 2, логічна нерівнозначність |
| $f_7(x,y)$ | 0 | 1 | 1 | 1 | $x \vee y$ | диз'юнкція, логічне додавання, "або" |
| $f_8(x,y)$ | 1 | 0 | 0 | 0 | $\overline{x \vee y} = x \downarrow y$ | заперечення диз'юнкції, стрілка Пірса |
| $f_9(x,y)$ | 1 | 0 | 0 | 1 | $x \leftrightarrow y$ | еквіваленція |
| $f_{10}(x,y)$ | 1 | 0 | 1 | 0 | $\overline{y}$ | заперечення, інверсія *y* |
| $f_{11}(x,y)$ | 1 | 0 | 1 | 1 | $y \to x$ | імплікація від *x* до *y* |
| $f_{12}(x,y)$ | 1 | 1 | 0 | 0 | $\overline{x}$ | заперечення, інверсія *x* |
| $f_{13}(x,y)$ | 1 | 1 | 0 | 1 | $x \to y$ | імплікація від *y* до *x* |
| $f_{14}(x,y)$ | 1 | 1 | 1 | 0 | $\overline{x \wedge y} = x \mid y$ | заперечення кон'юнкції, штрих Шеффера |
| $f_{15}(x,y)$ | 1 | 1 | 1 | 1 | 1 | константа 1 |

При проектуванні логічних схем основний інтерес становлять наступні функції двох змінних *x* та *y*.

$f_1(x,y) = x \cdot y$ – логічне множення (кон'юнкція);

$f_7(x,y) = x \vee y$ – логічне додавання (диз'юнкція);

$f_{14}(x,y) = \overline{x \cdot y}$ – логічне множення з інверсією;

$f_8(x,y) = \overline{x \vee y}$ – логічне додавання з інверсією;

$f_6(x,y) = x \oplus y = x\overline{y} \vee \overline{x}y$ – підсумовування за модулем 2;

$f_9(x,y) = \overline{x \oplus y} = xy \vee \overline{x}\,\overline{y}$ – рівнозначність.

## 1.5. Аналітичне представлення булевих функцій

Як відомо, кожній формулі алгебри логіки відповідає деяка логічна функція, яка задається таблицею істинності. Розглянемо обернену задачу – опис функцій алгебри логіки довільного числа аргументів у вигляді логічного виразу.

Розроблено універсальні (канонічні) форми представлення булевих функцій, які дають можливість отримати аналітичну форму довільної функції безпосередньо із таблиці істинності. Ця форма у подальшому може бути мінімізована або спрощена. Найбільш широке розповсюдження отримали *досконала диз'юнктивна нормальна форма* (ДДНФ) і *досконала кон'юнктивна нормальна форма* (ДКНФ).

Якщо булева функція задана таблицею істинності, то вона може бути представлена в аналітичній формі за допомогою операцій кон'юнкції, диз'юнкції та інверсії за допомогою наступних правил:

- кожній одиниці в таблиці істинності ставиться у відповідність кон'юнкція рангу *n*, де *n* кількість аргументів функції;
- аргумент входить в кон'юнкцію без інверсії, якщо він у відповідному наборі приймає значення 1 і з інверсією, якщо приймає значення 0;
- всі отримані кон'юнкції об'єднуються знаками диз'юнкції.

Такий аналітичний вираз у формі диз'юнкції елементарних кон'юнкцій називають *досконалою диз'юнктивною нормальною формою* (ДДНФ). Досконалою, тому що всі кон'юнкції мають ранг *n*, нормальною тому що інверсії застосовуються тільки до окремих аргументів.

Якщо в таблиці істинності нулів набагато менше, ніж одиниць, то використовують аналітичний запис у вигляді досконалої кон'юнктивної нормальної форми (ДКНФ). Вона будується наступним чином:

- кожному нулю в таблиці істинності ставиться у відповідність диз'юнкція рангу *n*, де *n* кількість аргументів функції;
- аргумент входить в диз'юнкцію без інверсії, якщо він у відповідному наборі приймає значення 0 і з інверсією, якщо приймає значення 1;

- всі отримані диз'юнкції об'єднуються знаками кон'юнкції.

Таким чином, *досконала кон'юнктивна нормальна форма* (ДКНФ) є кон'юнкцією елементарних диз'юнкцій рангу *n*.

Розглянемо наприклад логічну функцію $f(x_1, x_2, x_3)$, що задана в табл. 2.

Таблиця 2

**Таблиця істинності функції $f(x_1, x_2, x_3)$**

| $x_1$ | $x_2$ | $x_3$ | $f(x_1,x_2,x_3)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Потрібно записати ДКНФ та ДДНФ цієї функції. Для запису ДКНФ спочатку виділимо набори аргументів, на яких функція набуває значення 0 (це трійки (0, 0, 0), (0, 1, 0), (1, 0, 1) ), та утворимо елементарні диз'юнкції так, щоб кожна з них мала значення 0. Згідно означення, диз'юнкція набуває значення 0 тоді і тільки тоді, коли кожен з її аргументів має значення 0. Тому, якщо у деякому наборі змінна набуває значення 1, включимо її в елементарну диз'юнкцію під знаком інверсії. Одержимо наступні елементарні диз'юнкції: $x_1 \vee x_2 \vee x_3$, $x_1 \vee \overline{x_2} \vee x_3$, $\overline{x_1} \vee x_2 \vee \overline{x_3}$. Шукана ДКНФ є кон'юнкцією побудованих елементарних диз'юнкцій.

Тобто, $f(x_1, x_2, x_3) = (x_1 \vee x_2 \vee x_3) \cdot (x_1 \vee \overline{x_2} \vee x_3) \cdot (\overline{x_1} \vee x_2 \vee \overline{x_3})$.

Аналогічно, для запису ДДНФ функції зафіксуємо спочатку набори аргументів (0, 0, 1), (0, 1, 1), (1, 0, 0), (1, 1, 0) та (1, 1, 1), на яких функція набуває значення 1, і побудуємо для них елементарні кон'юнкції $\overline{x_1} \cdot \overline{x_2} \cdot x_3$, $\overline{x_1} \cdot x_2 \cdot x_3$, $x_1 \cdot \overline{x_2} \cdot \overline{x_3}$, $x_1 \cdot x_2 \cdot \overline{x_3}$, $x_1 \cdot x_2 \cdot x_3$, кожна з яких набуває значення 1 на відповідному наборі аргументів. Для цього, якщо у відповідному наборі деяка змінна мала значення 0, то ми включали її у кон'юнкцію під знаком інверсії,

адже, кон'юнкція дорівнює 1 лише тоді, коли всі її члени мають значення 1. Запишемо ДДНФ функції як диз'юнкцію одержаних елементарних кон'юнкцій. Матимемо:

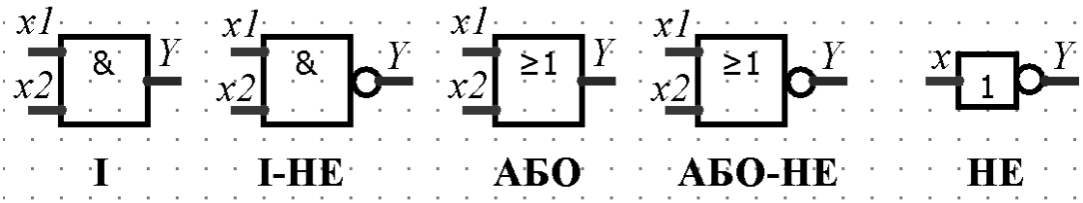
$$f(x_1, x_2, x_3) = \overline{x_1} \cdot \overline{x_2} \cdot x_3 \vee \overline{x_1} \cdot x_2 \cdot x_3 \vee x_1 \cdot \overline{x_2} \cdot \overline{x_3} \vee x_1 \cdot x_2 \cdot \overline{x_3} \vee x_1 \cdot x_2 \cdot x_3.$$

### 1.6. Логічні схеми

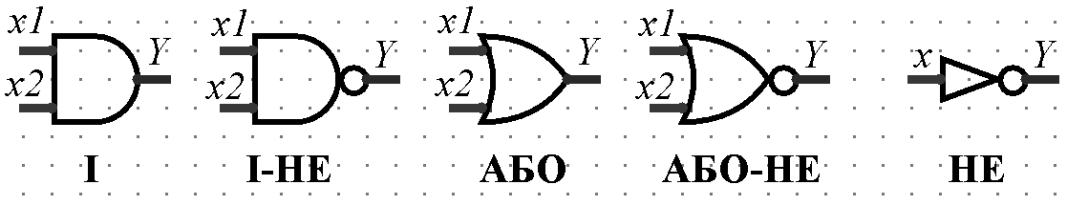
Фізичний пристрій, що реалізує одну з операцій алгебри логіки або найпростішу логічну функцію, називається *логічним елементом* (рис.1).

Позначення основних логічних елементів відповідно до англійського (BS) і американського (MIL / ANSI) стандартів показані на рис. 1.

*а)*




*б)*




*с)*

**I**

| *a* | *b* | *Y* |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**I-НЕ**

| *a* | *b* | *Y* |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**АБО**

| *a* | *b* | *Y* |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**АБО-НЕ**

| *a* | *b* | *Y* |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

**НЕ**

| *a* | *Y* |
|---|---|
| 0 | 1 |
| 1 | 0 |

Рис. 1. Основні логічні елементи цифрової техніки:

*а* – англійський (BS) стандарт, *б* – американський (MIL / ANSI) стандарт,

*с* – таблиці істинності поданих елементів

У Великобританії широко поширений американський стандарт, і лише деякі фірми використовують стандарт BS.

Існує три основних типи логічних елементів: схема І, яка реалізує логічне множення, або кон'юнкцію, схема АБО, яка реалізує логічне додавання, або диз'юнкцію, схема НЕ – інвертор, яка реалізує логічне заперечення. Всі інші логічні схеми можна отримати на основі цих трьох основних схем. Схеми І та АБО можуть мати багато входів і мають один вихід. Схема НЕ має тільки один вхід і один вихід. На рис. 1 *с*, подано таблиці істинності логічних операцій, згідно яких працюють логічні елементи.

Алгебра логіки дає можливість будувати складні функції, аргументи яких є функціями інших двійкових аргументів. Схема, складена зі скінченого числа логічних елементів за певними правилами, називається *логічною схемою*.

### 1.7. Логічне проектування

Зазвичай логічне проектування виконується в наступній послідовності:

1) складання таблиці істинності синтезованого вузла відповідно до його означення, призначення і (словесного) опису принципу роботи;

2) задання аналітичної формули для логічної функції, що описує роботу синтезуючого вузла відповідно до наявної таблиці істинності;

3) аналіз отриманої функції з метою побудови різних варіантів її логічного виразу (на підставі законів булевої алгебри) і знаходження найкращого з них відповідно до того чи іншого критерію;

4) синтез функціональної (логічної) схеми вузла із заздалегідь заданим набором логічних елементів.

Синтез комбінаційних пристроїв зазвичай починається з табулювання значень істинності всіх вхідних і вихідних величин (перший етап проектування). Табличне задання закону функціонування деякого пристрою є найбільш наочним і універсальним засобом опису його роботи. Результатом розглянутого етапу є таблиця істинності, яка зв'язує всі можливі комбінації значень аргументів і функцій.

На другому етапі потрібно задати аналітичну функцію, згідно отриманої на першому етапі таблиці істинності. Та сама функція може бути представлена різними формулами. Кожній формулі відповідає своя суперпозиція і, отже, своя схема з'єднань елементів. Очевидно, серед схем, що реалізують дану функцію, є більш проста. Пошук логічної формули, що відповідає цій схемі, представляє великий практичний інтерес, а перетворення формул булевих функцій ґрунтується на використанні законів булевої алгебри.

На другому етапі розв'язується задача *мінімізації* логічної функції, яка полягає в тому, щоб знайти найбільш компактне її подання у вигляді нормальної форми мінімальної складності: мінімальної диз'юнктивної нормальної форми (МДНФ) або мінімальної кон'юнктивної нормальної форми (МКНФ).

*Мінімальна нормальна форма* – це нормальна форма, яка містить мінімальну кількість змінних, взятих з запереченням чи без. Мінімальна диз'юнктивна нормальна форма (МДНФ) це диз'юнкція мінімальної кількості кон'юнкцій змінних, взятих з запереченням чи без. Мінімальна кон'юнктивна нормальна форма (МКНФ) це кон'юнкція мінімальної кількості диз'юнкцій змінних, взятих з запереченням чи без.

Одним з методів отримання МДНФ чи МКНФ є метод діаграм Вейча (карт Карно). Даний метод був винайдений в 1952 році Едвардом В. Вейчем і вдосконалений в 1953 році Морісом Карно.

Даний метод дозволяє швидко одержувати МДНФ або МКНФ булевої функції *f* невеликого числа змінних. В основі методу лежить задання булевих функцій діаграмами деякого спеціального виду, що одержали назву діаграм Вейча. Для булевої функції двох змінних діаграма Вейча зображена на рис. 2.

Кожна клітинка діаграми відповідає набору змінних булевої функції в її таблиці істинності. У клітинках діаграми Вейча для МДНФ записуються значення аргументів відповідної елементарної кон'юнкції, яка має значення 1, а у випадку МКНФ – елементарної диз'юнкції, яка має значення 0.

а)

| | $X1$ | $\overline{X1}$ |
|---|---|---|
| $X2$ | 11 | 01 |
| $\overline{X2}$ | 10 | 00 |

б)

| | $X1$ | $\overline{X1}$ |
|---|---|---|
| $X2$ | 00 | 10 |
| $\overline{X2}$ | 01 | 11 |

Рис. 2. Діаграма Вейча для функції двох змінних:

*а* – для МДНФ, *б* – для МКНФ

Наприклад, запис 11 у клітинці таблиці, поданій на рис. 2а, означає, що елементарна кон'юнкція $x_1 \cdot x_2$, якій вона відповідає, матиме значення 1 на наборі значень аргументів (1, 1). Запис 11 у клітинці таблиці, поданій на рис. 2 б, означає, що елементарна диз'юнкція $\overline{x_1} \vee \overline{x_2}$, якій вона відповідає, на наборі значень аргументів (1, 1) матиме значення 0.

Діаграма Вейча для булевої функції трьох змінних зображена на рис. 3. (якщо поряд з клітинкою не вказано значення аргументу, то вважається, що відповідна змінна береться з інверсією).

а)

| | $X1$ | | | |
|---|---|---|---|---|
| $X2$ | 110 | 111 | 011 | 010 |
| | 100 | 101 | 001 | 000 |

$X3$

б)

| | $X1$ | | | |
|---|---|---|---|---|
| $X2$ | 001 | 000 | 100 | 101 |
| | 011 | 010 | 110 | 111 |

$X3$

Рис. 3. Діаграма Вейча для функції трьох змінних:

*а* – для МДНФ, б – для МКНФ

Додавання до неї ще такої ж таблиці дає діаграму для функції 4-х змінних (рис.4).

Діаграми Вейча для більшого числа змінних можуть бути складені з діаграм меншого числа змінних. Тобто приписуванням ще однієї діаграми 4-х змінних до тільки що розглянутої, можна одержати діаграму для функції 5-ти

змінних. При цьому зручно приєднувати їх одна до одної однойменними стовпцями (рис. 5). Тоді сусідніми клітинами будуть також клітини стовпців, симетрично розташованих відносно лінії приєднання діаграм Вейча чотирьох змінних для однойменних рядків.

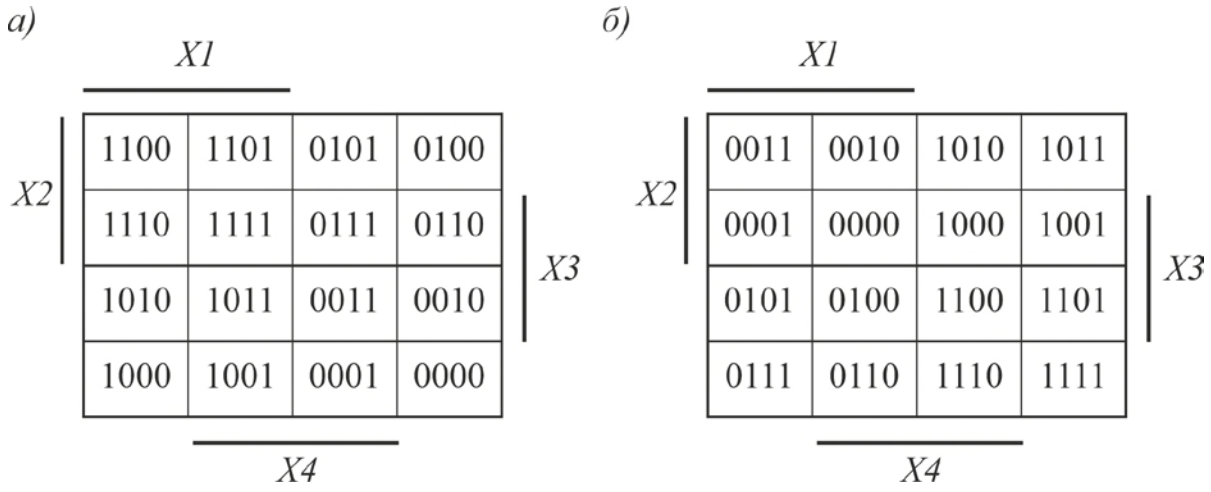



Рис. 4. Діаграма Вейча для функції чотирьох змінних:

*а* – для МДНФ, *б* – для МКНФ

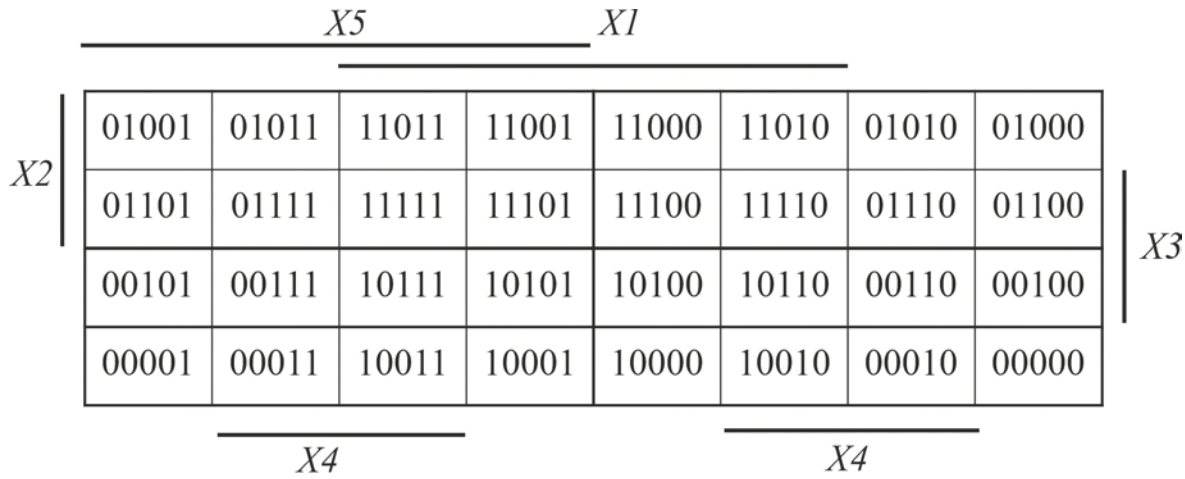



Рис. 5. Діаграма Вейча для МДНФ функції п’яти змінних

Отримання МДНФ функції відбувається на основі операцій склеювання. Операція склеювання здійснюється досконалими кон’юнкторами, або диз’юнкторами, у яких співпадають всі змінні крім однієї. В даному випадку змінні можна винести за дужки, а інші, можна склеїти. Наприклад:

$$Y = \overline{X_1} \cdot \overline{X_2} \cdot \overline{X_3} \cdot \overline{X_4} \vee \overline{X_1} \cdot \overline{X_2} \cdot \overline{X_3} \cdot X_4 = \overline{X_1} \cdot \overline{X_2} \cdot \overline{X_3}\left(\overline{X_4} \vee X_4\right) = \overline{X_1} \cdot \overline{X_2} \cdot \overline{X_3}$$

$$Y = (\overline{X_1} \vee \overline{X_2} \vee \overline{X_3} \vee \overline{X_4}) \cdot (\overline{X_1} \vee \overline{X_2} \vee \overline{X_3} \vee X_4) = \overline{X_1} \vee \overline{X_2} \vee \overline{X_3} \vee \overline{X_4} \cdot X_4 = \overline{X_1} \vee \overline{X_2} \vee \overline{X_3}$$

В діаграмі Вейча при переході із однієї комірки в іншу по вертикалі та по горизонталі змінюється значення тільки однієї змінної. В результаті чого набори значень між якими можливе склеювання отримуються згрупованими.

Розглянемо випадки склеювань на прикладі діаграми Вейча для чотирьох змінних. Склеювати (об'єднувати) можна дві, чотири, вісім, і т.д. клітин, які є між собою сусідами.

Зауважимо, що в процесі мінімізації формули для конкретної булевої функції одиницями відмічають ті клітинки діаграми Вейча, що відповідають наборам аргументів, які присутні в аналітичному записі відповідної функції. Тому для створення МДНФ на діаграмі Вейча позначимо клітинки, що відповідають наборам аргументів, на яких функція набуває значення 1. Відповідно, при створенні МКНФ у клітині діаграми Вейча ставиться одиниця, якщо булева функція приймає значення 0 для відповідного набору аргументів.

На рис. 6 зображено вироджений випадок, коли спрощення не відбувається. Це випадок може зустрічатися на будь-яких діаграмах Вейча. В даному випадку маємо функцію $Y = X_1 \cdot \overline{X_2} \cdot X_3 \cdot \overline{X_4} \vee \overline{X_1} \cdot X_2 \cdot \overline{X_3} \cdot \overline{X_4}$. До таких елементарних кон'юнкцій неможливо застосувати склеювання.

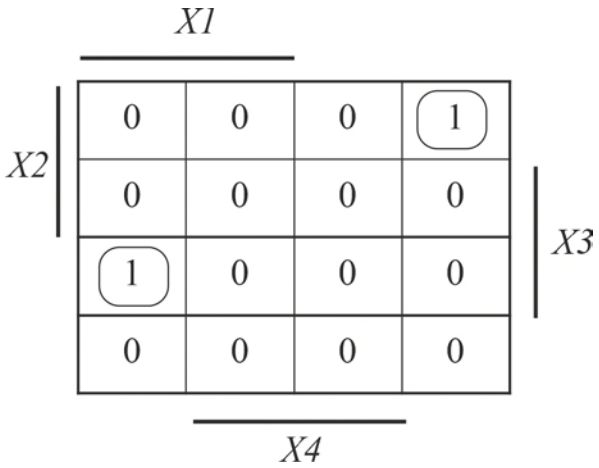



Рис. 6. Вироджений випадок склеювання в діаграмі Вейча, коли спрощення не відбувається

На рис 7 зображено випадок, коли при склеюванні спрощується одна змінна. Цей випадок теж зустрічається на діаграмах різного виду. На діаграмі Вейча для функції $Y = X_1 \cdot X_2 \cdot \overline{X_3} \cdot X_4 \vee \overline{X_1} \cdot X_2 \cdot \overline{X_3} \cdot X_4 \vee \overline{X_1} \cdot X_2 \cdot \overline{X_3} \cdot \overline{X_4} \vee X_1 \cdot \overline{X_2} \cdot X_3 \cdot \overline{X_4} \vee \overline{X_1} \cdot \overline{X_2} \cdot X_3 \cdot X_4 \vee \overline{X_1} \cdot \overline{X_2} \cdot X_3 \cdot \overline{X_4} \vee X_1 \cdot \overline{X_2} \cdot \overline{X_3} \cdot X_4 \vee \overline{X_1} \cdot \overline{X_2} \cdot \overline{X_3} \cdot X_4$, зображеної на рис 7, позначено клітинки, які можна об'єднати. Це означає, що перший і сьомий доданок можна склеїти по $X_2$, другий і третій – по $X_4$, четвертий і шостий – по $X_1$, п'ятий і восьмий – по $X_3$. В результаті отримаємо мінімізовану форму:

$$Y = X_1 \cdot \overline{X_3} \cdot X_4 \vee \overline{X_1} \cdot X_2 \cdot \overline{X_3} \vee \overline{X_2} \cdot X_3 \cdot \overline{X_4} \vee \overline{X_1} \cdot \overline{X_2} \cdot X_4.$$

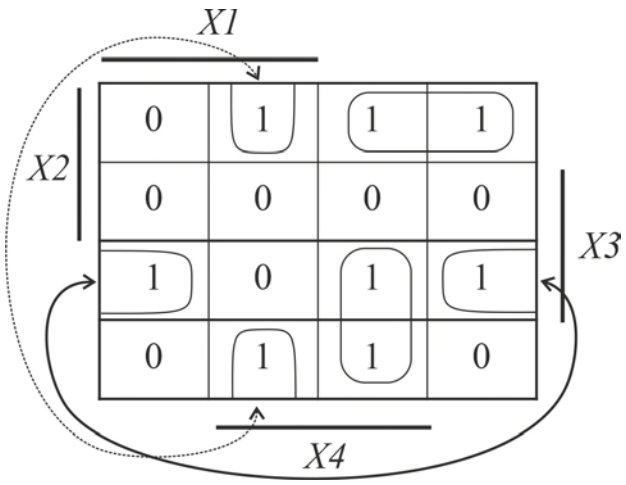



Рис. 7. Випадок склеювання в діаграмі Вейча, коли при склеюванні спрощується одна змінна

На рис 8 зображено випадок, коли при склеюванні спрощується дві змінні. Це відповідає об'єднанню чотирьох суміжних клітинок таблиці.

Такий випадок зустрічається на діаграмах Вейча для функцій, де число змінних рівне, або більше трьох. З діаграм Вейча, зображених на рис. 8 отримаємо функції:

а) $Y = X_1 \cdot \overline{X_4} \vee X_2 \cdot X_3 \vee \overline{X_1} \cdot \overline{X_3}$;

б) $Y = X_3 \cdot X_4 \vee \overline{X_3} \cdot \overline{X_4}$.

На рис. 9 зображено випадок, коли при склеюванні спрощується три змінні – об'єднуються вісім суміжних клітинок. Цей випадок зустрічається на

діаграмах Вейча для функцій, де число змінних рівне або більше чотирьох. В даному випадку з діаграм Вейча (рис. 9), отримаємо функції:

а) $Y = X_1 \vee X_3$;

б) $Y = \overline{X_3} \vee \overline{X_4}$;

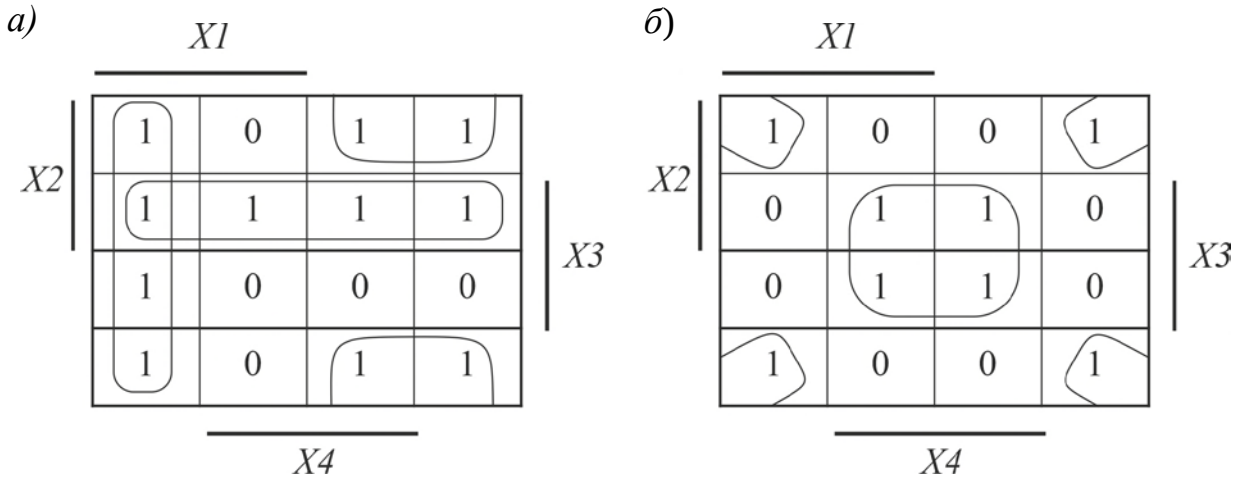



Рис. 8. Випадок склеювання в діаграмі Вейча,
коли спрощуються дві змінних

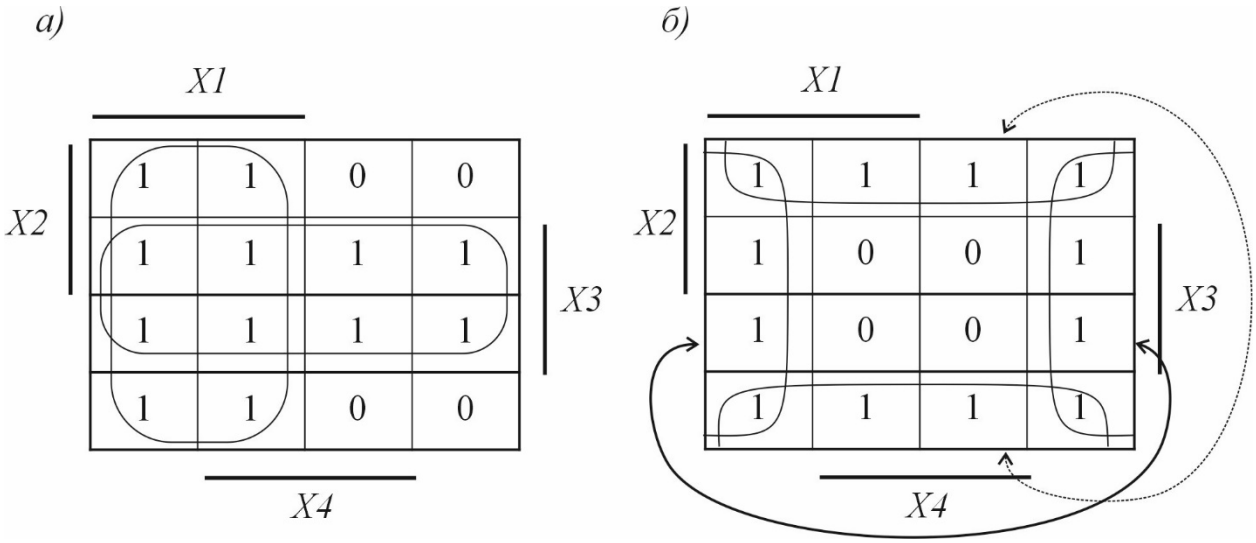



Рис. 9. Випадок склеювання в діаграмі Вейча,
коли спрощуються три змінних

На четвертому етапі відбувається синтез функціональної схеми заданої за допомогою логічної функції. При побудові схеми, аргументи функції будуть входами схеми, а значення функції – її виходом. Побудова схеми заснована на прямому заміщенні елементарних добутків, сум і заперечень відповідно кон'юнкторами, диз'юнкторами й інверторами.

Розглянемо синтез схеми на прикладі. Нехай задана таблиця істинності деякої перемикальної функції (табл. 3).

Таблиця 3

**Таблиця істинності деякого цифрового пристрою**

| *Входи* | | | | *Вихід* |
|---|---|---|---|---|
| *X*1 | *X*2 | *X*3 | *X*4 | *Y* |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 |

З таблиці істинності отримаємо вираз функції у вигляді ДДНФ:

$$Y = \overline{X_1} \cdot \overline{X_2} \cdot \overline{X_3} \cdot \overline{X_4} \vee \overline{X_1} \cdot \overline{X_2} \cdot \overline{X_3} \cdot X_4 \vee \overline{X_1} \cdot \overline{X_2} \cdot X_3 \cdot X_4 \vee \overline{X_1} \cdot X_2 \cdot \overline{X_3} \cdot X_4 \vee$$
$$\vee \overline{X_1} \cdot X_2 \cdot X_3 \cdot X_4 \vee X_1 \cdot \overline{X_2} \cdot X_3 \cdot \overline{X_4}.$$

До побудови функціональної схеми синтезованого вузла можна переходити відразу ж, як тільки стає відомим аналітичний опис його роботи. При заміщенні елементарних добутків, сум і заперечень відповідно

кон’юнкторами, диз’юнкторами й інверторами, отримаємо схему вузла в ДДНФ, який працює згідно таблиці істинності, зображеної в табл. 3 (рис. 10).

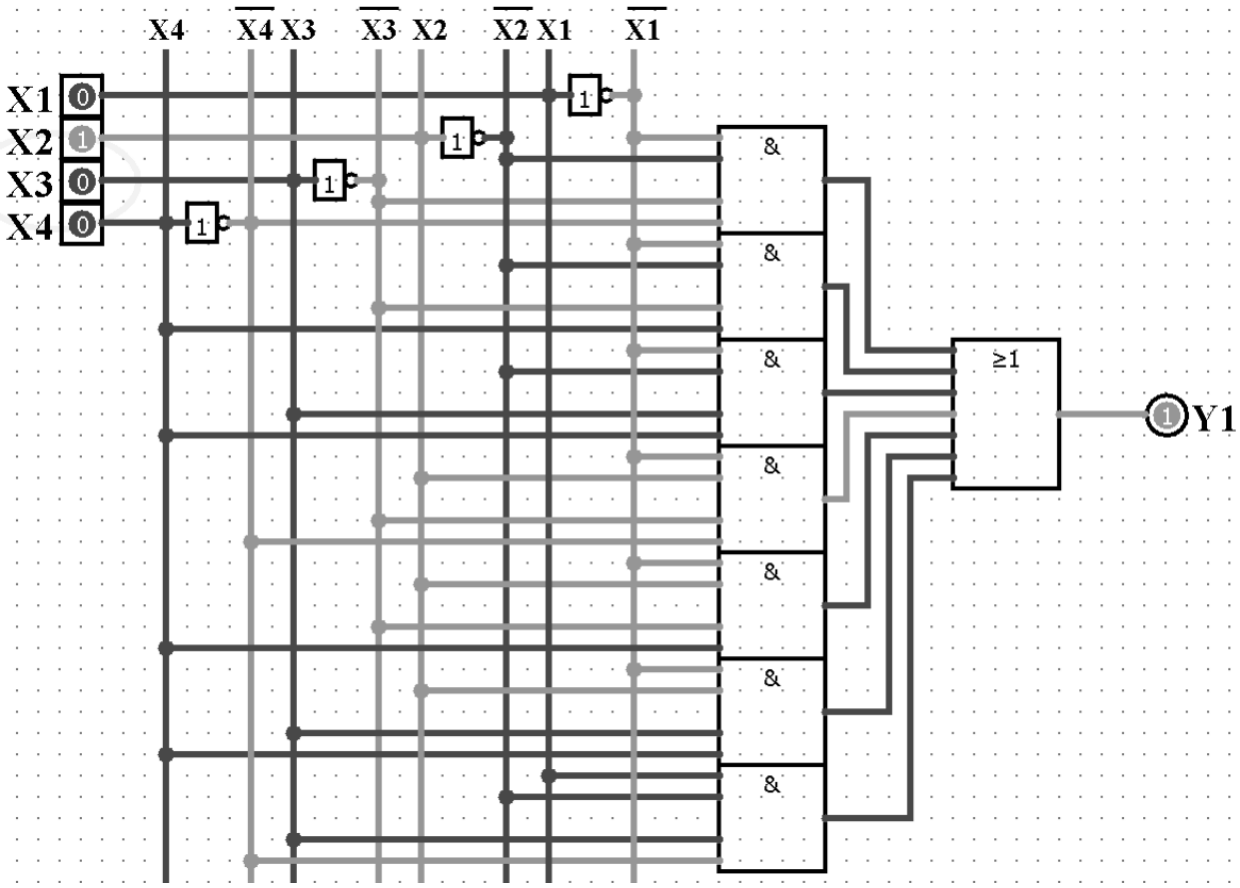



Рис. 10. Логічна схема вузла в ДДНФ, яка функціонує згідно таблиці істинності поданої в табл. 3

Тепер за допомогою діаграм Вейча знайдемо функцію у вигляді МДНФ. Для цього заповнимо діаграму Вейча (рис. 11) згідно таблиці істинності поданої в табл. 3. Після того як діаграма заповнена, робимо відповідні склеювання.

В результаті отримаємо запис логічної функції в МНДФ:

$Y = \overline{X_1}\,\overline{X_3} \vee \overline{X_1}\,X_4 \vee X_1\overline{X_2}X_3\,\overline{X_4}$.

За отриманою функцією побудуємо відповідну схему (рис. 12).

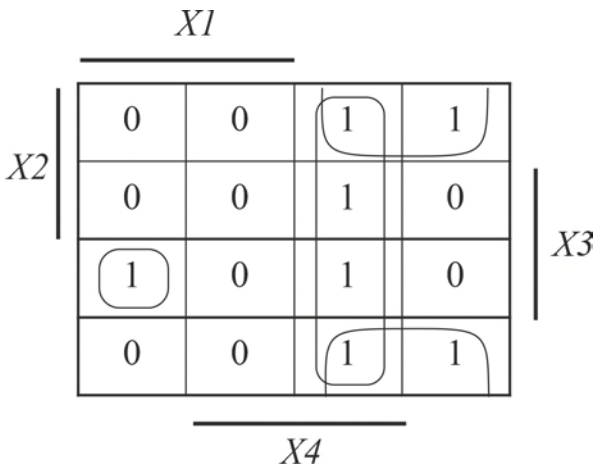



Рис. 11. Діаграма Вейча логічної функції, що працює згідно таблиці істинності поданої в табл. 3

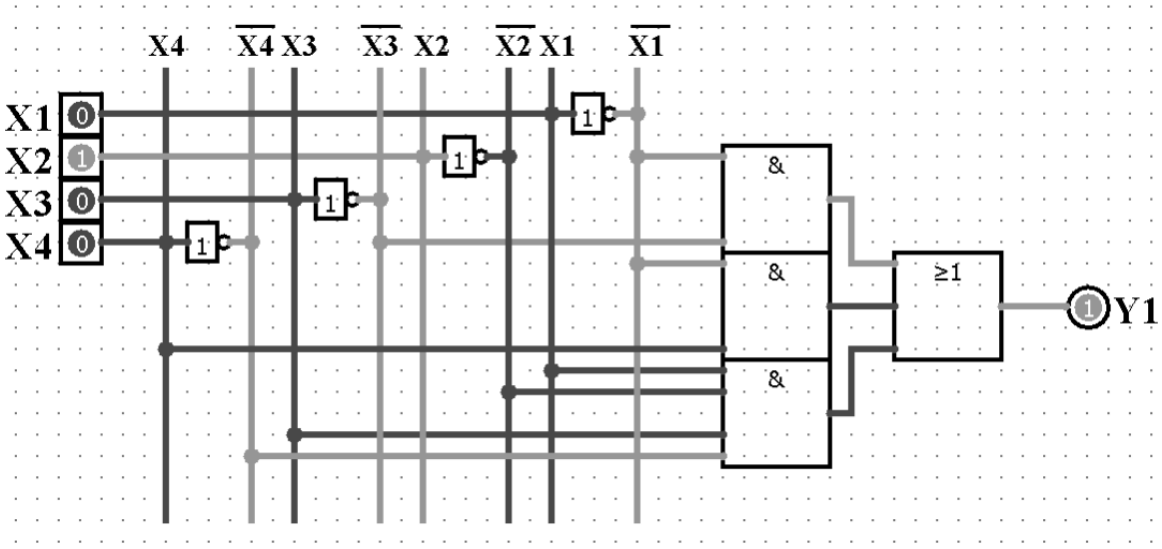



Рис. 12. Логічна схема вузла в МДНФ, яка функціонує згідно таблиці істинності поданої в табл. 3

Найбільш очевидним способом мінімізації ДДНФ є виконання перетворень, аналогічних перетворенням звичайної алгебри. Це можливо, оскільки для операцій І та АБО справедливий асоціативний і дистрибутивний закони. Мова йде про те, щоб перейти від ДДНФ до ДНФ із мінімумом доданків, при цьому кількість множників у кожному доданку повинна бути також мінімальною (позбутися від «досконалості»), тобто максимально зменшити кількість змінних і операцій у ДДНФ.

Аналіз і оптимізація (мінімізація) логічних функцій є дуже важливими задачами синтезу цифрових автоматів без пам’яті.

## Завдання

1. На елементах АБО та АБО-НЕ побудувати однотактний пристрій, який реалізує наступний алгоритм роботи:

*a)* на вхід подається 5-розрядний двійковий код (*X*1, *X*2, *X*3, *X*4, *X*5), на виході *Y* маємо 1, якщо число одиниць в коді рівно 4 чи 5, в інших випадках на виході маємо 0.

*b)* на вхід подається 5-розрядний двійковий код (*X*1, *X*2, *X*3, *X*4, *X*5), на виході *Y* маємо 1, якщо число одиниць в коді рівно 3, в інших випадках на виході маємо 0.

*c)* на вхід подається 5-розрядний двійковий код (*X*1, *X*2, *X*3, *X*4, *X*5), на виході *Y* маємо 1, якщо число одиниць в коді рівно 2, в інших випадках на виході маємо 0.

*d)* на вхід подається 5-розрядний двійковий код (*X*1, *X*2, *X*3, *X*4, *X*5), на виході *Y* маємо 1, якщо число одиниць в коді рівно 1 і 3, в інших випадках на виході маємо 0.

Для цього потрібно скласти таблицю істинності пристрою; отримати математичну формулу; на основі законів булевої алгебри перетворити формулу так, щоб операціями були тільки логічне додавання та інверсія; побудувати на логічних елементах схему, яка реалізує функцію за формулою.

2. На елементах І та І-НЕ побудувати однотактний пристрій, який реалізує наступний алгоритм роботи:

*a)* на вхід подається 5-розрядний двійковий код (*X*1, *X*2, *X*3, *X*4, *X*5), на виході *Y* маємо 1, якщо число одиниць в коді рівно 4 чи 5, в інших випадках на виході маємо 0.

*b)* на вхід подається 5-розрядний двійковий код (*X*1, *X*2, *X*3, *X*4, *X*5), на виході *Y* маємо 1, якщо число одиниць в коді рівно 3, в інших випадках на виході маємо 0.

*c)* на вхід подається 5-розрядний двійковий код (*X*1, *X*2, *X*3, *X*4, *X*5), на виході *Y* маємо 1, якщо число одиниць в коді рівно 2, в інших випадках на виході маємо 0.

*d)* на вхід подається 5-розрядний двійковий код (*X*1, *X*2, *X*3, *X*4, *X*5), на виході *Y* маємо 1, якщо число одиниць в коді рівно 1 і 3, в інших випадках на виході маємо 0.

Для цього скласти таблицю істинності пристрою; отримати математичну формулу; на основі законів булевої алгебри перетворити формулу так, щоб операціями були тільки логічне множення та інверсія; побудувати на логічних елементах схему, яка реалізує функцію за формулою.

3. Дано таблиця істинності, яка описує принцип роботи цифрового логічного пристрою:

*a)* табл. 4;

*b)* табл. 5;

*c)* табл. 6;

*d)* табл. 7;

*e)* табл. 8;

*f)* табл. 9.

За допомогою діаграм Вейча отримати операторне подання функції у вигляді МДНФ та МКНФ цифрового логічного пристрою. Побудувати схему реалізації даної функції.

Таблиця 4

| *X4* | *X3* | *X2* | *X1* | *Y* |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 |

Таблиця 5

| *X4* | *X3* | *X2* | *X1* | *Y* |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

Таблиця 6

| *X4* | *X3* | *X2* | *X1* | *Y* |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

Таблиця 7

| *X4* | *X3* | *X2* | *X1* | *Y* |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

Таблиця 8

| *X4* | *X3* | *X2* | *X1* | *Y* |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

Таблиця 9

| *X4* | *X3* | *X2* | *X1* | *Y* |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

## Питання для самоконтролю

1. Що таке логічна змінна і логічний сигнал? Які значення вони можуть приймати?
2. Що таке таблиця істинності логічного елемента чи пристрою, який виконує деяке логічне перетворення?
3. Що таке логічна функція?
4. Чи може бути логічним сигналом рівень напруги? Стан контакту?
5. Як буде поводитися схема І, якщо на одному із входів внаслідок внутрішньої несправності буде постійно присутня логічна одиниця? Логічний нуль?
6. Як буде поводитися схема АБО, якщо на одному із входів внаслідок внутрішньої несправності буде постійно присутня логічна одиниця? Логічний нуль?
7. Поясніть, чому не використані входи логічних елементів АБО, АБО-НЕ з’єднуються з корпусом (рівнем логічного «0»), а на невикористані входи логічних елементів І, І-НЕ подається напруга рівня логічної «1»?

# 2. СИНТЕЗ ЦИФРОВИХ СХЕМ В ПРОГРАМІ LOGISIM

В університетах усього світу набув поширення безкоштовний програмний інструмент Logisim, над яким з 2001 року працює Карл Барч [1]. Інструмент придатний для розробки та моделювання цифрових логічних схем. Його використовують при викладанні різноманітних дисциплін, починаючи з логіки, завершуючи повними курсами з архітектури комп'ютерів. [2]

Logisim – вільне програмне забезпечення (GNU GPL) з графічним інтерфейсом користувача, при розробці якого використано Java та бібліотеку Swing, що робить продукт кросплатформенним. Програма призначена лише для роботи з цифровими компонентами, що звужує коло її використання. Основною її перевагою є можливість редагування схем в процесі їх моделювання, що сприяє підвищенню рівня наочності при використанні Logisim в навчальному процесі [3]. Цей програмний інструмент дозволяє автоматично створювати комбінаційні схеми за заданими параметрами, використовувати зовнішні репозиторії компонент власного формату, здійснювати експорт схем в GIF-формат.

## 2.1. Робоче вікно Logisim

На рис. 13 зображено робоче вікно Logisim. Вікно складається з рядка меню, панелі інструментів, панелі провідника, таблиці атрибутів і робочої області. Панель провідника містить список схем проекту і всі інструменти завантажених бібліотек. Склад панелі інструментів налаштовується у вікні параметрів проекту, яке викликається з пункту «Параметры» меню «Проект». Таким чином, будь-які інструменти можна додатково виносити на панель інструментів. В робочій області розташовані всі компоненти редагованої схеми. Робоча область покрита сіткою, за якою вирівнюються всі компоненти й провідники. Таблиця атрибутів містить атрибути виділеного в даний момент інструменту або компонента робочої області. Logisim дозволяє моделювати поведінку цифрових схем. Якщо прапорець «Моделирование включено» з

меню «Моделирование» активовано, то Logisim прораховує процеси, що відбуваються в схемі під час її редагування: змінюються значення на входах і виходах пристроїв, оновлюється внутрішній стан пристроїв пам’яті, пристрої виведення відображають відповідну інформацію, а провідники змінюють колір залежно від значень, які по них проходять.

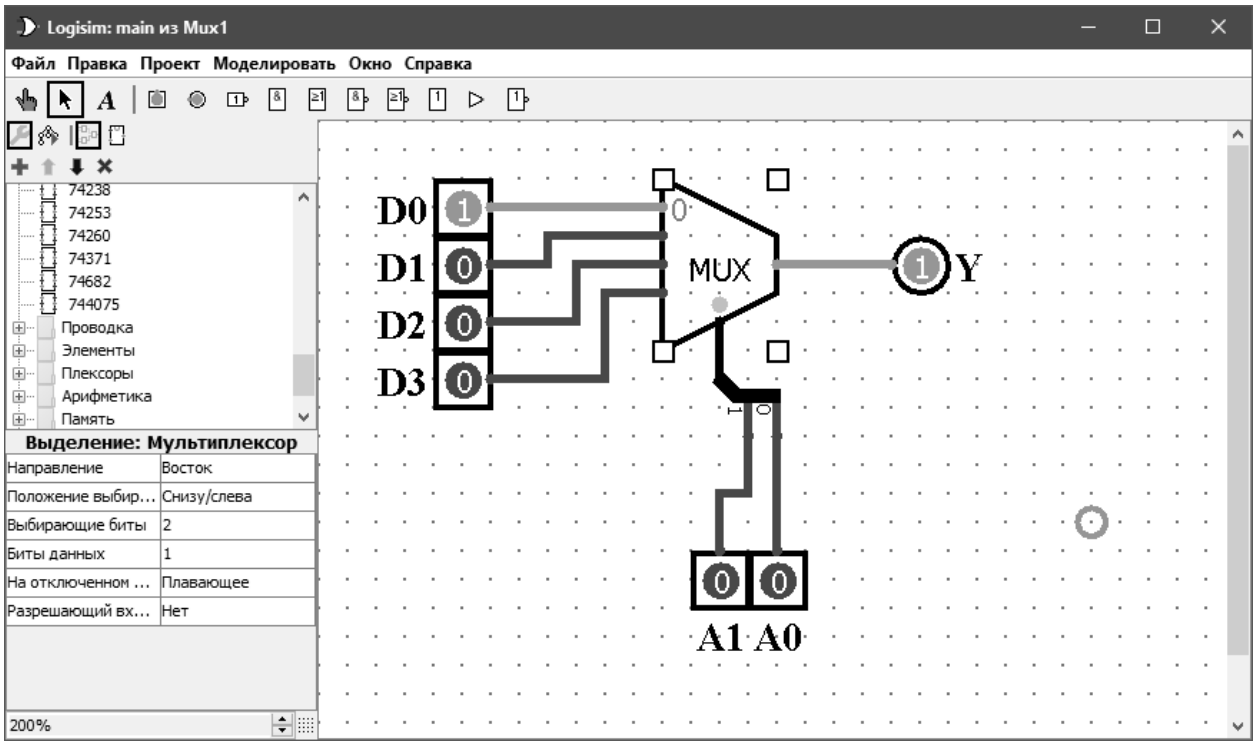



Рис. 13. Робоче вікно Logisim

Logisim підтримує дві категорії параметрів конфігурації: налаштування додатку і параметри проекту. Налаштування додатку охоплює всі відкриті проекти, а параметри проекту відносяться до одного певного проекту.

Переглядати і редагувати додаток можна через пункт «Настройки» меню «Файл» (рис. 14).

В даному вікні, ми можемо: задавати шаблон, який завантажується кожен раз, коли Logisim створює новий проект; налагоджувати параметри, що впливають на те, як поводяться вбудовані інструменти; вибрати мову та форму елементів. Форма елементів Logisim підтримує три стандарти для відтворення логічних елементів: фігурні елементи, прямокутні елементи, і елементи DIN

40700. Оскільки стиль з фігурною формою, більш популярний в США, а прямокутний стиль популярний в Європі, то стилі вибирають відповідно до цих регіонів. Logisim не відповідає якому-небудь стандарту, він дозволяє перемикатися між ними.

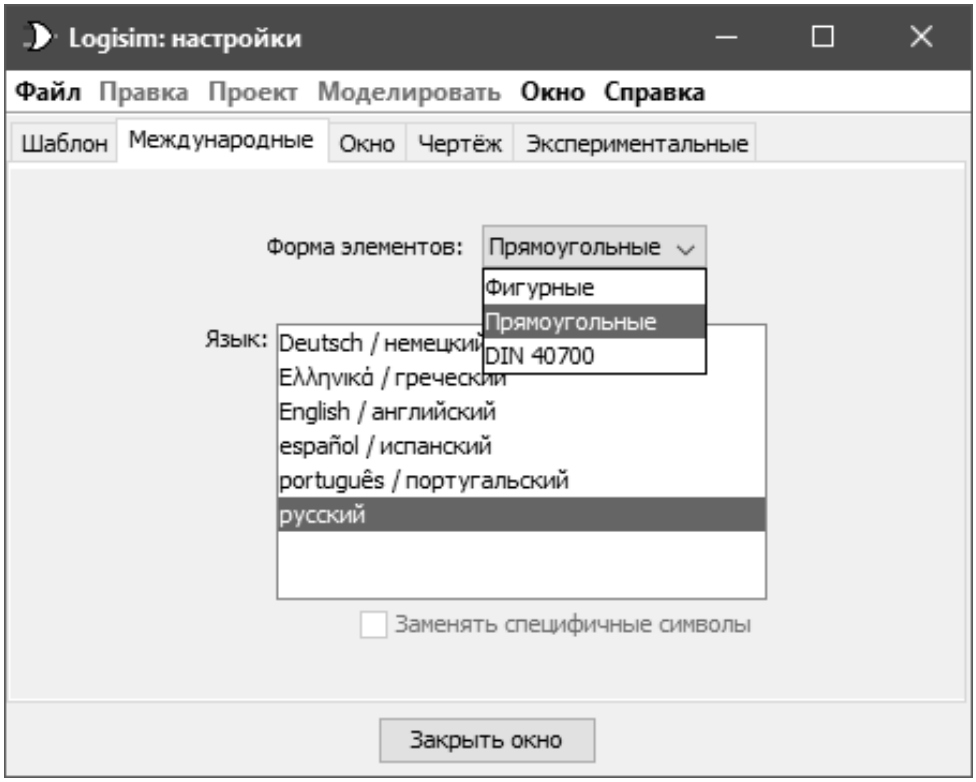



Рис. 14. Вікно налаштування програми Logisim

## 2.2. Провідники та компоненти

Розміщувати провідники в робочій області можна інструментами «Правка» або «Проводка». Компоненти схеми з'єднані між собою провідниками. Провідник й контакт елемента вважаються з'єднаними, якщо вони розташовуються строго в одній і тій же точці сітки. Те саме стосується з'єднання провідника з іншим провідником. На рис. 15 подано приклади, коли провідники з'єднані між собою через вузол (*а*) і не з'єднані (*б*). При переміщенні по робочій області вже підключених елементів, Logisim буде зберігати з'єднання, якщо це можливо. З'єднання не зберігається, якщо елемент перетягувати з натиснутою клавішею Shift.

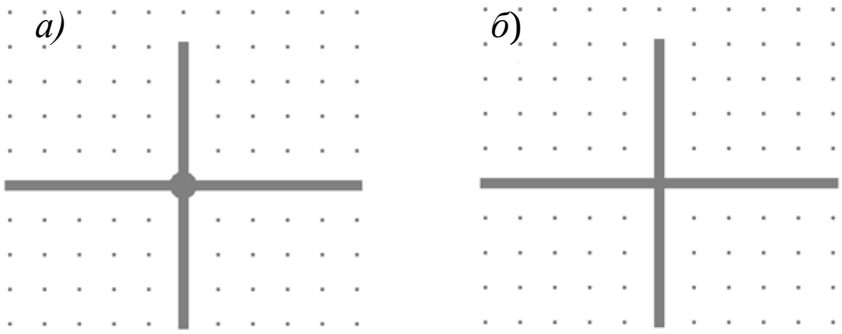



Рис. 15. Приклади реалізації з’єднання провідників (*а*)
та їх накладання (*б*)

Кожен вхід і вихід компонентів у схемі має розрядність, пов’язану з ним. Часто розрядність дорівнює 1, але багато з вбудованих компонентів Logisim мають атрибути, що дозволяють налаштувати розрядність їх входів і виходів. Logisim дозволяє створювати провідники, що зв’язують разом кілька бітів. Розрядність провідника підлаштовується під компоненти, до яких він приєднаний. Кількість бітів, що проходять по провіднику – це розрядність даного провідника. Якщо провідник з’єднує дві компоненти, що мають різну розрядність, Logisim повідомить про несумісність розрядності.

Провідники в Logisim можуть мати один з семи кольорів.

Сірий: розрядність провідника невідома. Це відбувається тому, що провідник не підключений до жодного з входів або виходів компонента.

Синій: провідник передає однобітне значення, але значення, яке він несе в даний момент, не відоме.

Темно-зелений: провідник передає однобітне значення «0».

Яскравий зелений: провідник передає однобітне значення «1».

Чорний: провідник передає багатобітне значення. Деякі або всі біти можуть бути не визначені.

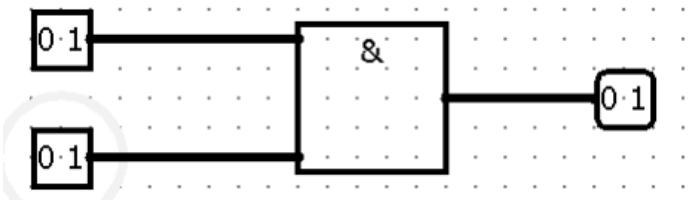
Червоний: провідник передає значення помилки. Це часто виникає через те, що елемент не може визначити правильне вихідне значення, або тому що на жоден вхід не подано певного значення. Також це часто виникає через те, що два компоненти намагаються передати провіднику два різних значення.

Багатобітні провідники стануть червоними, якщо хоча б один з провідників передає значення помилки.

Помаранчевий: компоненти, приєднані до провідника, не узгоджені за розрядністю.

На рис 16. подано схему для обчислення побітового «І» між двома двобітними входами. Також двобітний вихід є побітовим «І» для двох входів. На рис 16*а)*, всі компоненти налаштовані для роботи з двобітними даними через їх атрибут «Биты данных». На рис. 16*б*) подано приклад несумісності розрядності при підключенні.

*а)*




*б)*

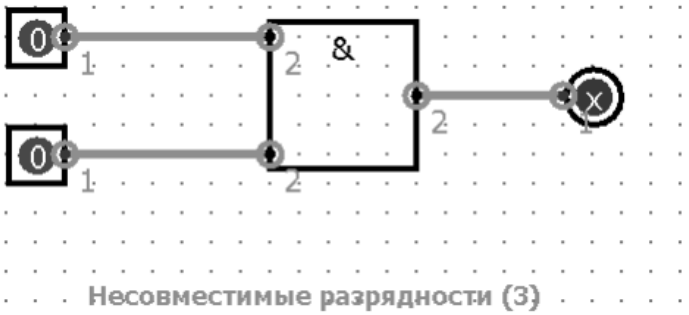



Рис. 16. Схема для обчислення побітового «І»
між двома двобітними входами

При роботі з багатобітними значеннями використовують інструмент «Разветвитель» з бібліотеки «Проводка», який дозволяє розділити пучок провідників на окремі провідники, або зібрати їх в один пучок. Цей компонент також дозволяє призначати, в якому порядку провідники входять в пучок.

## 2.3. Бібліотеки Logisim

В Logisim інструменти організовані в бібліотеки. Вони відображаються у вигляді папок в панелі провідника.

Середовище Logisim автоматично включає в себе кілька бібліотек (рис. 17):

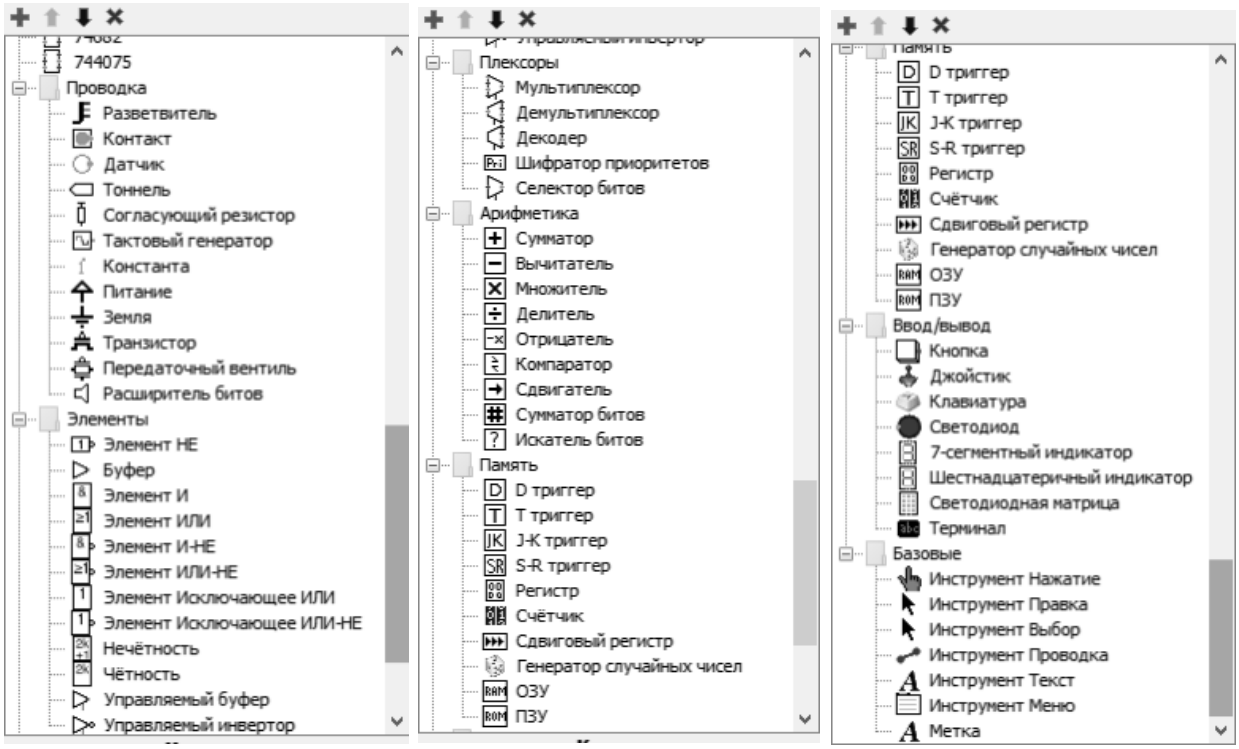



Рис. 17. Біліотеки інструментів в Logisim

«Проводка»: компоненти, які відносяться в основному до провідників і базових понять електроніки;

«Элементы»: компоненти, які виконують прості логічні функції;

«Плексоры»: більш складні комбінаційні компоненти, такі як мультиплексори і декодери;

«Арифметика»: компоненти, що виконують арифметичні дії;

«Память»: компоненти, що зберігають дані, такі як тригери, регістри, і ОЗП;

«Ввод/вывод»: компоненти для взаємодії з користувачем;

«Базовые»: інструменти, необхідні для використання Logisim.

Більш детально з інформацією по кожному компоненту бібліотеки можна ознайомитися в довідці Logisim.

Logisim дозволяє також додавати інші бібліотеки за допомогою підменю «Загрузить библиотеку…» в меню «Проект». Logisim має три категорії бібліотек. Вбудовані бібліотеки в Logisim. Бібліотеки Logisim – проекти, побудовані в Logisim і збережені на диск як проект Logisim. Також є можливість розробити набір схем в одному проекті, а потім використовувати цей набір схем у вигляді бібліотеки для інших проектів. Бібліотеки JAR – бібліотеки, які розроблені в Java, але не розповсюджуються разом з Logisim. Розробка JAR бібліотеки набагато складніша, ніж розробка бібліотеки Logisim, але компоненти можуть бути набагато більш незвичними, в тому числі в плані атрибутів і взаємодії з користувачем. Вбудовані бібліотеки (крім бібліотеки Базові) були написані з використанням того ж API, що можуть використовувати бібліотеки JAR. Таким чином, вони вдало демонструють набір функціональних можливостей, які JAR бібліотеки можуть підтримувати.

Щоб видалити бібліотеку потрібно вибрати пункт «Выгрузить библиотеки…» з меню «Проект».

Багато компонентів мають атрибути, які є властивостями для налагодження поведінки або зовнішнього вигляду компонента. Таблиця атрибутів служить для перегляду і відображення значень атрибутів компонента.

Для вибору атрибутів компонента, потрібно виділити компонент за допомогою Інструменту «Правка» . (Можна також клікнути на компоненті правою кнопкою миші (або з затиснутою клавішею Ctrl) і вибрати «Показать атрибуты» з контекстного меню. Крім того, при взаємодії з компонентом через інструмент  теж можна побачити атрибути цього компонента.

Кожен тип компонентів має свій набір атрибутів. Якщо вибрано кілька компонентів за допомогою інструменту «Правка», то таблиця атрибутів відобразить атрибути, спільні для всіх обраних компонентів (за винятком

провідників). Так можна змінити значення атрибута всіх обраних компонентів одночасно, використовуючи таблицю атрибутів.

### 2.4. Побудова схем. Підсхеми

Кожен проект Logisim – це бібліотека схем. У своїй простій формі кожен проект має тільки одну схему за замовчуванням «main». Створення підсхем можна використовувати для розбиття великої схеми на більш дрібні частини та для повторного використання вже спроектованих частин схем. Щоб додати схему потрібно вибрати «Добавить схему…» з меню «Проект», а потім задати назву для нової схеми. Щоб редагувати схему, потрібно двічі клікнути на її назві в панелі провідника. Якщо в робочій області розміщено декілька копій однієї схеми, то редагування цієї схеми буде приводити до змін у всіх копіях.

Коли підсхема розміщена всередині більшої схеми, за замовчуванням вона зображується у вигляді прямокутника з виїмкою, що позначає північний кінець креслення підсхеми. Контакти будуть розміщені на межі прямокутника в залежності від їх напрямку.

Зовнішній вигляд за замовчуванням можна змінити вибравши з меню «Проект» пункт «Редактировать внешний вид схемы», і Logisim переключиться зі звичного інтерфейсу редагування креслення на інтерфейс для малювання зовнішнього вигляду схеми.

### 2.5. Комбінаційний аналіз

В Logisim є інструмент для проведення комбінаційного аналізу. Якщо схема не містить послідовнісних логічних пристроїв, контактів розрядністю більше 1 і кількість вхідних і вихідних контактів не більш 12, то для схеми можна провести комбінаційний аналіз. Модуль «Комбинационный анализ» дозволяє переглядати таблиці істинності та логічні вирази, задавати таблицю істинності, на основі якої можна отримувати вираз, виконувати його мінімізацію та будувати відповідну схему. У вікні «Комбинационный анализ», у вкладках «Входы» і «Выходы» можна задати імена вхідних і вихідних

контактів для майбутньої схеми; ввести таблицю істинності у вкладці «Таблица» або вираз для булевих функцій (вкладка «Выражение»); провести мінімізацію виразу у вкладці «Минимизация» і натиснути кнопку «Построить схему», вказавши назву нової схеми. Logisim синтезує схему, використовуючи логічні елементи.

Це вікно можна відкрити двома способами: в меню «Окно» вибрати «Комбинационный анализ», або в меню «Проект» вибрати «Анализировать схему» (рис. 18).

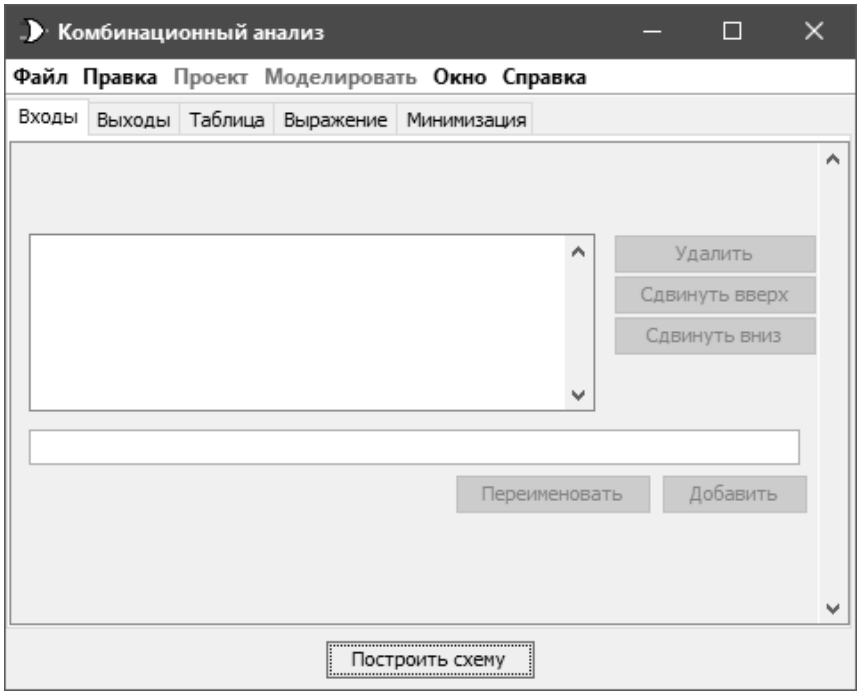



Рис.18. Вікно комбінаційного аналізу

Після аналізу схеми не залишається жодного постійного зв’язку між схемою і вікном «Комбинационный анализ». Тобто, зміни схеми не будуть відображені у вікні, і зміни в логічних виразах і/або таблиці істинності, зроблені в вікні, не будуть відображені в схемі. Звичайно, можна проаналізувати схему ще раз і замінити існуючу схему новою, яка відповідає тому, що знаходиться в вікні «Комбинационный анализ».

Більш детально про роботу з Logisim можна ознайомитися в довідці Logisim.

**Завдання**

1. В програмі Logisim, використовуючи вікно комбінаційного аналізу, побудувати пристрій, який реалізує наступний алгоритм роботи:

*a)* на вхід подається 5-розрядний двійковий код (*X*1, *X*2, *X*3, *X*4, *X*5), на виході *Y* маємо 1, якщо число одиниць в коді рівно 4 чи 5, в інших випадках на виході маємо 0.

*b)* на вхід подається 5-розрядний двійковий код (*X*1, *X*2, *X*3, *X*4, *X*5), на виході *Y* маємо 1, якщо число одиниць в коді рівно 3, в інших випадках на виході маємо 0.

*c)* на вхід подається 5-розрядний двійковий код (*X*1, *X*2, *X*3, *X*4, *X*5), на виході *Y* маємо 1, якщо число одиниць в коді рівно 2, в інших випадках на виході маємо 0.

*d)* на вхід подається 5-розрядний двійковий код (*X*1, *X*2, *X*3, *X*4, *X*5), на виході *Y* маємо 1, якщо число одиниць в коді рівно 1 і 3, в інших випадках на виході маємо 0.

*e)* на вхід подається 5-розрядний двійковий код (*X*1, *X*2, *X*3, *X*4, *X*5), на виході *Y* маємо 1, якщо дві одиниці в коді йдуть підряд (не більше), в інших випадках на виході маємо 0.

*f)* на вхід подається 5-розрядний двійковий код (*X*1, *X*2, *X*3, *X*4, *X*5), на виході *Y* маємо 1, якщо три одиниці в коді, йдуть підряд (не більше), в інших випадках на виході маємо 0.

*g)* на вхід подається 5-розрядний двійковий код (*X*1, *X*2, *X*3, *X*4, *X*5), на виході *Y* маємо 0, якщо число одиниць в коді рівно 4 чи 5, в інших випадках на виході маємо 1.

2. В програмі Logisim, використовуючи вікно комбінаційного аналізу, на логічних елементах побудувати схему, яка реалізує функцію:

*a)* $Y = \overline{\overline{X_3X_2X_1} \vee X_3\overline{\overline{X_2}X_1} \vee X_3\overline{\overline{X_1}}}$;

*b)* $Y = X_3X_2X_1 \vee \overline{\overline{X_3X_1} \vee X_3\overline{\overline{X_2}X_1} \vee X_2}$;

*c)* $Y = \overline{\overline{X_3X_2X_1 \vee \overline{X_2}X_1} \vee X_2\overline{X_1}}$;

*d)* $Y = \overline{X_3 X_1 \vee \overline{X_2} X_1 \vee X_2 \overline{X_1}};$

*e)* $Y = X_3 \overline{X_2 X_1} \vee \overline{\overline{X_3 X_1} \vee X_3 \overline{X_1 \vee X_2}};$

*f)* $Y = \overline{\overline{X_3 X_2 \vee \overline{X_2}} X_1 \vee X_2 \overline{X_1}};$

*g)* $Y = \overline{\overline{X_3 X_2} X_1 \vee X_3 \overline{\overline{X_2} X_1 \vee X_3 \overline{X_1}}};$

*h)* $Y = X_3 X_1 \vee \overline{\overline{X_3} X_1 \vee X_3 \overline{\overline{X_2} X_1 \vee X_2}};$

*i)* $Y = \overline{\overline{X_3 X_1 \vee \overline{X_2} X_1} \vee X_3 X_2 \overline{X_1}};$

*j)* $Y = \overline{X_3 \overline{X_2} X_1 \vee \overline{X_2} X_1 \vee X_2 \overline{X_1}};$

*k)* $Y = \overline{\overline{X_2 X_1} \vee \overline{X_3} X_1 \vee X_3 \vee \overline{X_2}};$

*l)* $Y = \overline{\overline{X_3 \vee \overline{X_3}}\ \overline{X_2} X_1 \vee X_3 X_2 \overline{X_1}}$

## Питання для самоконтролю

1. Які стандарти підтримує Logisim для відтворення логічних елементів?
2. Що таке розрядність входів і виходів компонентів Logisim в схемі?
3. Що таке розрядність провідника в Logisim?
4. Коли виникає несумісність розрядності при підключенні компонентів між собою?
5. Для чого потрібен інструмент «Разветвитель»?
6. Для чого потрібен є інструмент «Комбинационный анализ» в Logisim? Які функції він виконує?

# 3. ПРОЕКТУВАННЯ ДЕШИФРАТОРІВ ТА ШИФРАТОРІВ

Дешифратори і шифратори (як і елементи І, АБО, НЕ, І-НЕ, АБО-НЕ) є комбінаційними елементами: сигнали на їх виходах залежать від стану входів. Такі елементи не зберігають попередній стан після зміни сигналу на входах, тобто не мають пам’яті.

Дешифратори можуть бути повними й неповними. Повні дешифратори реагують на всі вхідні коди, неповні – на коди, величина яких не перевищує деякого, заздалегідь встановленого, значення. Виходи дешифраторів можуть бути прямими і інверсними.

## 3.1. Дешифратори

*Дешифратор* призначений для перетворення двійкового коду на вході в керуючий сигнал на одному з виходів. Дешифратор призначений для дешифрації числа, яке подане позиційним $n$-розрядним двійковим кодом.

Двійковий код, який містить завжди тільки одну одиницю, називається *унітарним*. Тобто, дешифратор є перетворювачем вхідного позиційного коду в унітарний вихідний код.

Якщо дешифратор має $n$ входів то для повного дешифратора вихідних шин повинно бути $N=2^n$, оскільки кожному із $2^n$ значень вхідного коду повинен відповідати одиничний сигнал на одному із виходів дешифратора. Якщо дешифратор не повний, то число виходів буде $N<2^n$

Повний дешифратор з трьома входами працює відповідно до табл. 10. Основу структури дешифратора можуть становити елементи І; вихід кожного з них є виходом дешифратора. Якщо цей вихід повинен бути збуджений, то на входах елемента І повинні збиратися логічні одиниці. При цьому розряди вхідного коду, в якому присутні логічні одиниці, повинні надходити на входи елемента І безпосередньо, а нульові розряди повинні інвертуватися.

Таблиця 10

**Таблиця істинності повного дешифратора з трьома входами**

| Десяткове число | Входи | | | Виходи | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | X2 | X1 | X0 | Y0 | Y 1 | Y 2 | Y 3 | Y 4 | Y 5 | Y 6 | Y 7 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 4 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 5 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 6 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 7 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Функціонування повного дешифратора з трьома входами подається системою логічних виразів виду:

$$Y_0 = \overline{X_2} \cdot \overline{X_1} \cdot \overline{X_0} = \overline{X_2 \vee X_1 \vee X_0},$$

$$Y_1 = \overline{X_2} \cdot \overline{X_1} \cdot X_0 = \overline{X_2 \vee X_1 \vee \overline{X_0}},$$

$$Y_2 = \overline{X_2} \cdot X_1 \cdot \overline{X_0} = \overline{X_2 \vee \overline{X_1} \vee X_0},$$

……

$$Y_7 = X_2 \cdot X_1 \cdot X_0 = \overline{\overline{X_2} \vee \overline{X_1} \vee \overline{X_0}}.$$

Схема дешифратора, побудованого за допомогою даної системи рівнянь на елементах І зображена на рис.19.

Логічна 1 на виході $Y_0$ повинна з'явитися, коли на входах $X_2$, $X_1$, $X_0$ присутній двійковий код 000 десяткового числа 0. Тому входи верхнього (за схемою) кон'юнктора повинні бути з'єднані з лініями $\overline{X_2}$, $\overline{X_1}$, $\overline{X_0}$ на кожній з яких є логічна 1, коли на входах $X_2 = X_1 = X_0 = 0$. Логічна 1, наприклад, на виході $Y_2$ повинна з'явитися, тоді коли на входах $X_2$, $X_1$, $X_0$ встановлюється код 010 десяткового числа 2, тому входи відповідного кон'юнктора повинні бути

з’єднані з лініями $\overline{X_2}$, $X_1$, $\overline{X_0}$, на кожній з яких є логічна 1, коли $X_2$=0, $X_1$=1, $X_1$=0. Аналогічно з'єднуються з лініями входи інших кон’юнкторів.

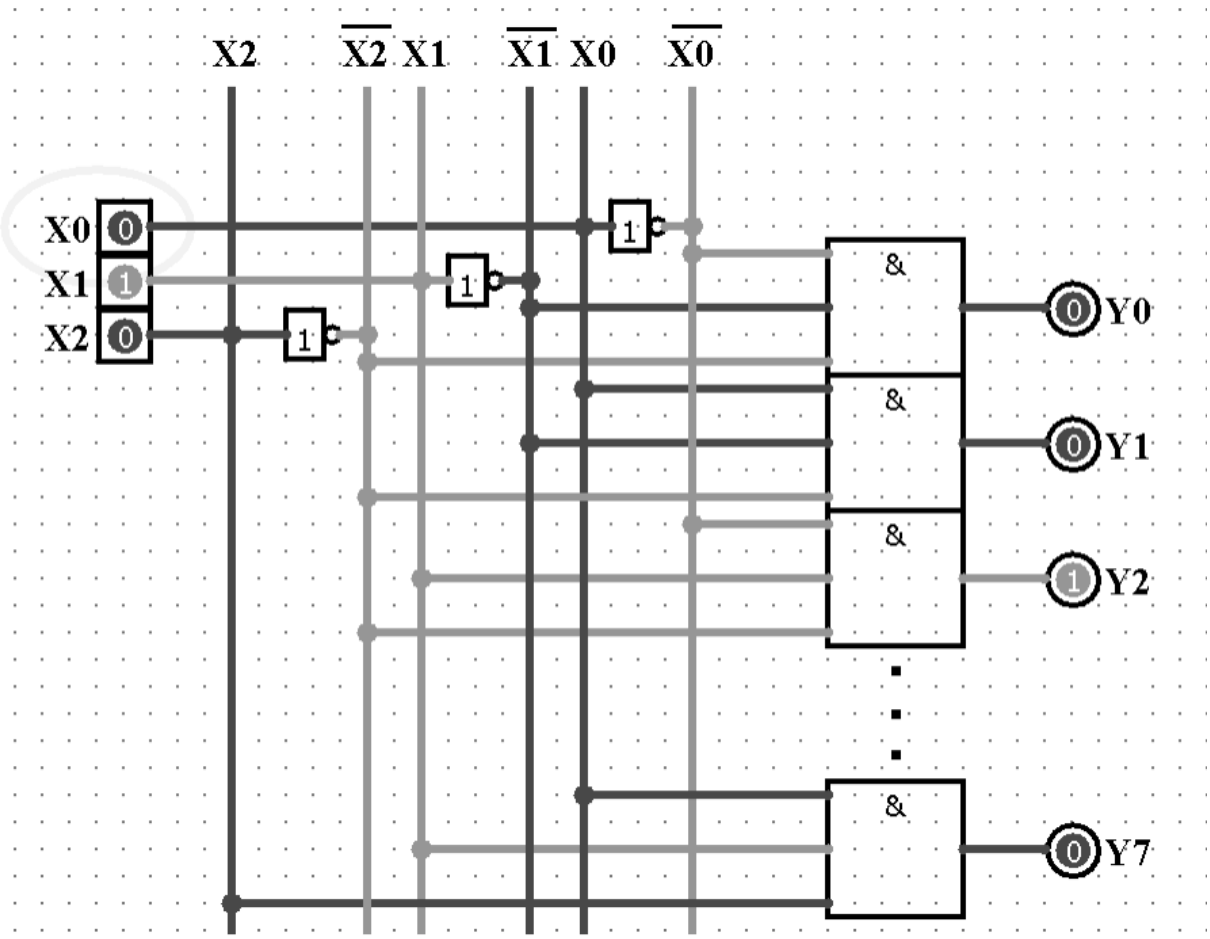



Рис.19. Схема повного дешифратора з трьома входами

Деякі типи дешифраторів мають інверсні виходи: на збудженому (активізованому) виході присутній логічний 0, в той час як на всіх інших – логічна 1. Такі дешифратори зручно використовувати, коли активним сигналом для вибору пристрою з виходу дешифратора є логічний 0.

Наприклад, розглянемо принцип роботи мікросхеми 74154 (рис. 20).

Мікросхема 74154 перетворює 4-разрядний двійковий код у сигнал низького рівня на одному з 16 виходів. Коли 4-разрядний двійковий код поступає на адресні входи мікросхеми 74154 (A, B, C, D), то на відповідному виході встановлюється напруга низького рівня, а на інших виходах – високого.

Однак, це відбувається в тому випадку, якщо на обидва дозволяючих

входи мікросхеми 74154 G1 та G2, подається напруга низького рівня. Якщо на один або на обидва дозволяючі входи подається напруга високого рівня, така напруга встановлюється і на всіх виходах.

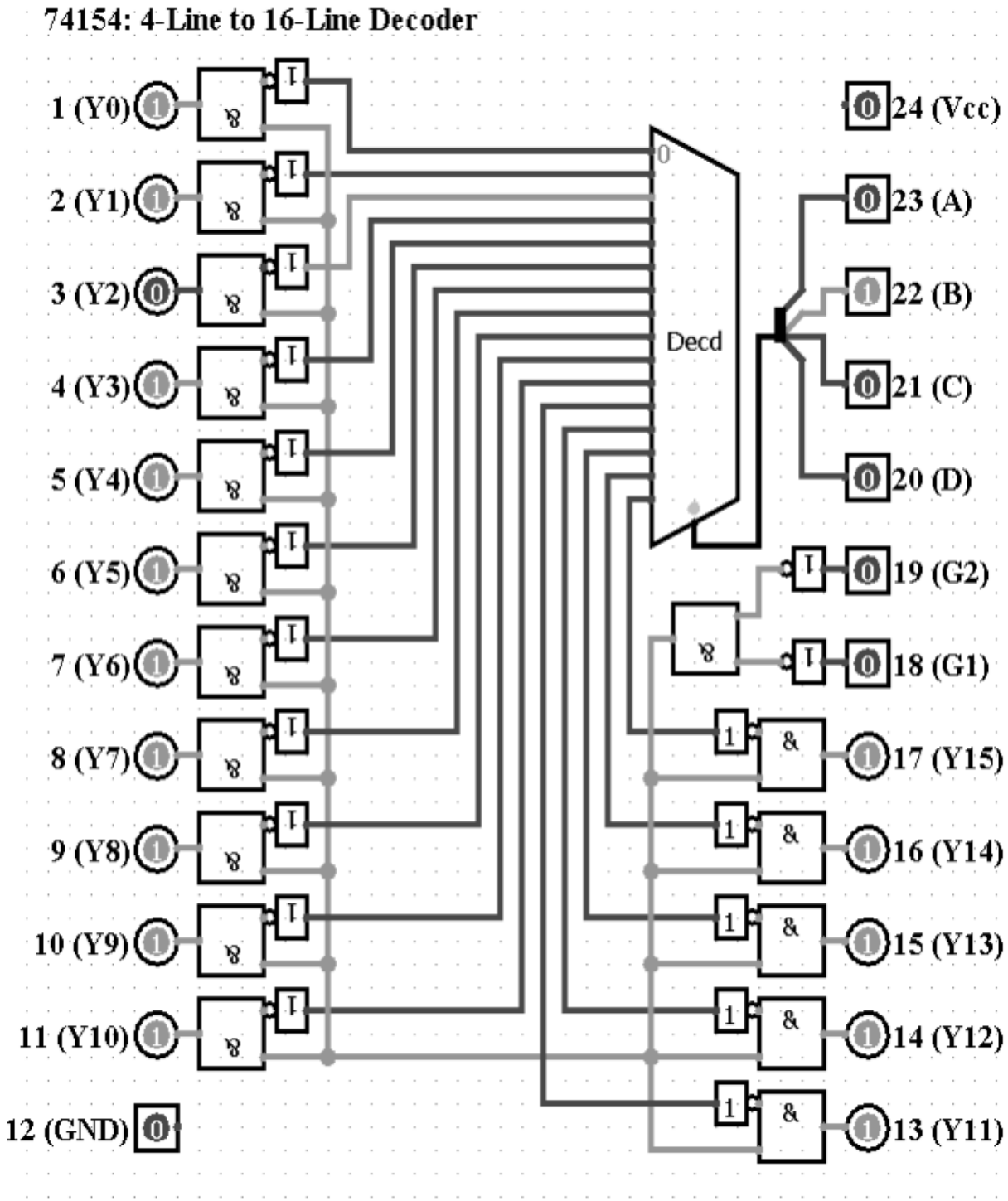



Рис. 20. Детальна схема мікросхеми 74154

Якщо на одному з дозволяючих входів мікросхеми 74154 подається напруга низького рівня, а інший розглядається як інформаційний вхід, то вибраний через адресні входи вихід, буде мати такий же логічний рівень, що і другий дозволяючий вхід. Таким чином, мікросхема 74154 використовується в якості демультиплексора або багатоканального комутатора даних.

На рис. 20 подано детальну схему мікросхеми 74154. Якщо подати на дозволяючі входи G1 та G2 «0», а на адресні входи код числа 2 (0010), тоді на виході Y2 буде «0», а на всіх інших виходах «1» .

### 3.2. Розширення розрядності дешифратора

Розглянемо розширення розрядності дешифратора на прикладі дешифратора на 2 входи (рис. 21). Дешифратор має два адресних входи і чотири виходи. Для побудови схеми для розширення розрядності дешифратора, він повинен мати дозволяючий вхід *V*.

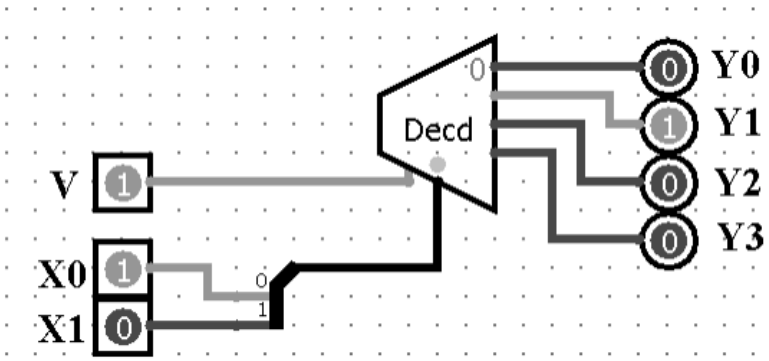



Рис. 21. Функціонування дешифратора на 2 входи

Розширення розрядності подано на рис. 22. На даному рисунку лівий (за схемою) дешифратор постійно активізований логічною 1 на вході *V*. Кодами на його адресних входах *X*3, *X*2 може бути активізований (обраний) кожен з дешифраторів *DC*0…*DC*3 Вибір одного з виходів 0...4 кожного з них визначається кодом на об’єднаних входах *X*1, *X*0. Таким чином, кожен з 16 виходів може бути активізований восьмирозрядним кодом, два розряди якого вибирають номер дешифратора, а два – номер його виходу.

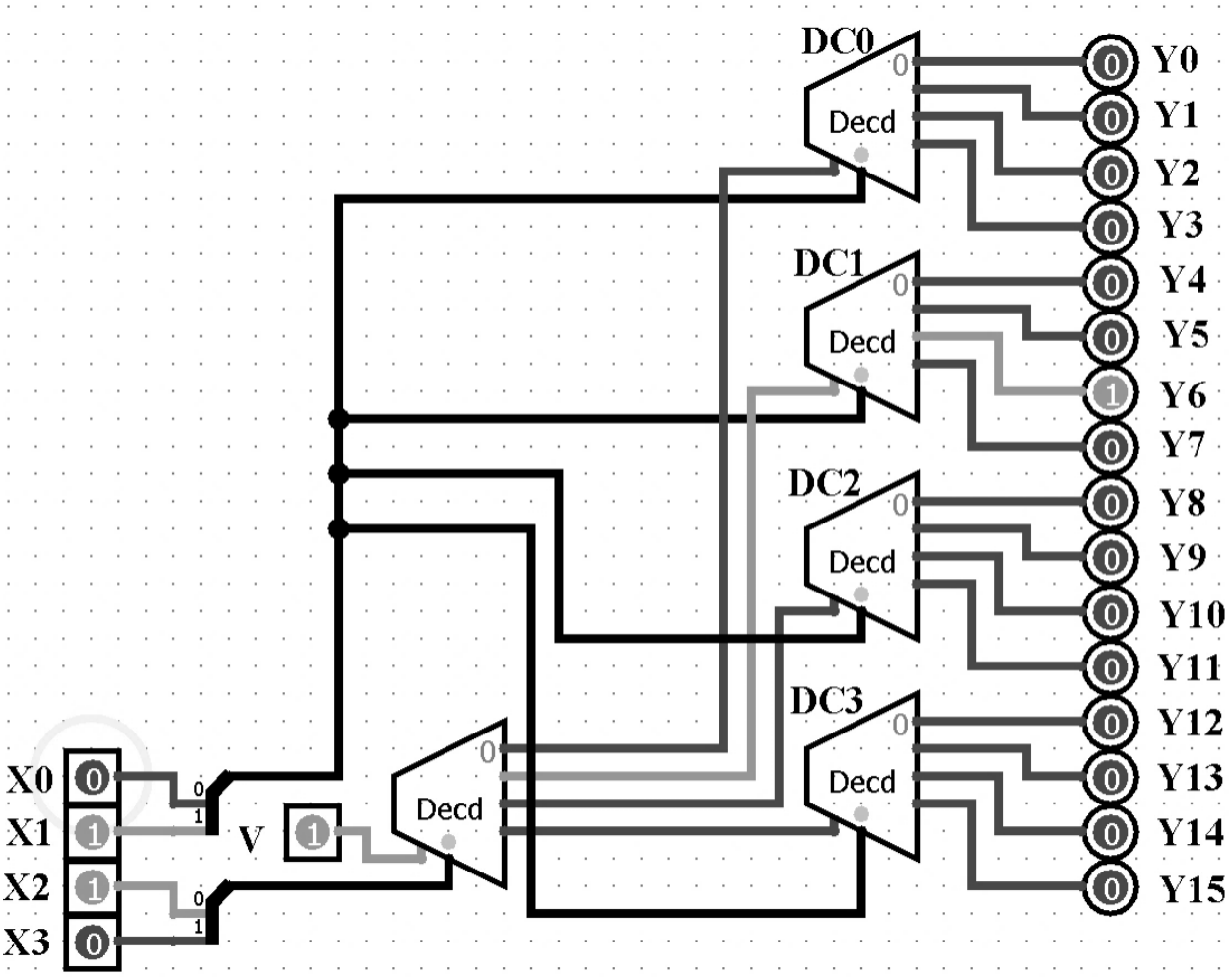



Рис. 22. Розширення розрядності дешифратора
на прикладі дешифратора на 2 входи

### 3.3. Застосування дешифраторів

Основне призначення дешифратора полягає в тому, щоб вибрати (адресувати, ініціалізувати) один об’єкт із багатьох. Кожному об’єкту присвоюють конкретну адресу (номер). Коли на входи дешифратора надходить двійковий код адреси, відповідний елемент активується на зв’язаному з ним виході дешифратора, а інші елементи залишаються заблокованими.

Можна передбачити, щоб з одного з виходів дешифратора на визначений блок надходив керуючий сигнал у випадку, коли на входах дешифратора з’являється визначений код, що відповідає, наприклад, перевищенню якогось параметра (температури, напруги і т.д.), який повинен бути приведений до нормального рівня зазначеним блоком.

В комп’ютерній техніці дешифратори використовуються для виконання наступних операцій:

- дешифрування коду операції, записаного в регістр команд процесора, що забезпечує вибір потрібної мікропрограми;
- перетворення коду адреси операнда в команді в керуючі сигнали вибору заданої комірки пам’яті в процесі записування або читання інформації;
- забезпечення візуалізації на зовнішніх пристроях;
- реалізація логічних операцій та побудови мультиплексорів і демультиплексорів.

Дешифратори використовуються для дешифрування коду операції і адреси операнда, розташованих в регістрі команд процесора. Дешифрування коду операції в пристрої керування визначає тип машинної команди. Дешифрування адреси операнда в оперативній пам’яті забезпечує доступ до вказаної комірки пам’яті для записування або зчитування даних.

На дешифраторі можуть бути реалізовані логічні функції.

*Наприклад* $Y = \overline{X_2} \cdot X_1 \vee \overline{X_2} \cdot X_0 \vee X_2 \cdot \overline{X_1} \cdot X_0$.

Виконаємо для даної функції перетворення:

$$Y = \overline{X_2} \cdot X_1 \vee \overline{X_2} \cdot X_0 \vee X_2 \cdot \overline{X_1} \cdot X_0 = \overline{X_2} \cdot X_1 \cdot (X_0 \vee \overline{X_0}) \vee \overline{X_2} \cdot X_0 \cdot (X_1 \vee \overline{X_1}) \vee$$
$$X_2 \cdot \overline{X_1} \cdot X_0 = \overline{X_2} \cdot X_1 \cdot X_0 \vee \overline{X_2} \cdot X_1 \cdot \overline{X_0} \vee \overline{X_2} \cdot X_0 \cdot X_1 \vee \overline{X_2} \cdot X_0 \cdot \overline{X_1} \vee X_2 \cdot$$
$$\overline{X_1} \cdot X_0 = \overline{X_2} \cdot X_1 \cdot \overline{X_0} \vee \overline{X_2} \cdot X_1 \cdot X_0 \vee \overline{X_2} \cdot \overline{X_1} \cdot X_0 \vee X_2 \cdot \overline{X_1} \cdot X_0$$

Логічна функція від трьох змінних. Тому дешифратор потрібно брати на 3 входи. Логічні змінні подаються на адресні входи дешифратора ($X_2$, $X_1$, $X_0$). Перша кон’юнкція функції *y* збуджує вихід №2 , друга – вихід №3, третя – вихід №1 і четверта – вихід № 5. Так як умова $Y = 1$ повинна мати місце при наявності кожної з цих кон’юнкцій, то виходи 1, 2, 3 і 5 треба об’єднати диз’юнкцією (рис. 23).

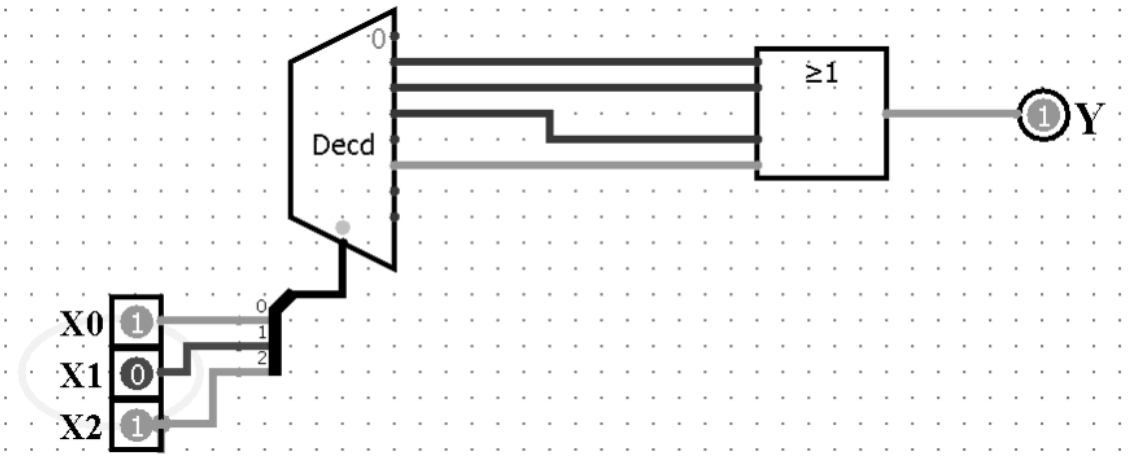



Рис. 23**.** Реалізація логічної функції $Y = \overline{X_2} \cdot X_1 \vee \overline{X_2} \cdot X_0 \vee X_2 \cdot \overline{X_1} \cdot X_0$.
за допомогою дешифратора

### 3.4. Шифратори

*Шифратор* (кодер) перетворює одиничний сигнал на одному із входів в *n*-розрядний двійковий код.

Шифратор розв'язує задачу, обернену дешифратору: зокрема, на його виходах встановлюється двійковий код, що відповідає десятковому номеру збудженого інформаційного входу. Повний двійковий шифратор має $2^n$ входів і $n$ виходів. Для неповного шифратора входів менше ніж $2^n$.

Шифратори, випускаються пріоритетними і не пріоритетними. У пріоритетного шифратора входи мають різний пріоритет. Збуджений вхід з більшим пріоритетом придушує дію раніше збудженого і встановлює на виходах код, який відповідає його значенню.

Повний непріоритетний дешифратор з трьома виходами працює відповідно до табл. 11. В даному випадку вважається, що збудженим може бути тільки один з входів.

Функціонування повного шифратора непріоритетного з трьома виходами подається системою логічних виразів виду:

$$Y0 = X1 \vee X3 \vee X5 \vee X7 = \overline{\overline{X1} \cdot \overline{X3} \cdot \overline{X5} \cdot \overline{X7}},$$

$$Y1 = X2 \vee X3 \vee X6 \vee X7 = \overline{\overline{X2} \cdot \overline{X3} \cdot \overline{X6} \cdot \overline{X7}},$$

$$Y2=X4 \vee X5 \vee X6 \vee X7 = \overline{\overline{X4} \cdot \overline{X5} \cdot \overline{X6} \cdot \overline{X7}}.$$

Таблиця 11

**Таблиця істинності повного непріоритетного шифратора на 3 виходи**

| Десяткове число | Входи | | | | | | | | Виходи | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | *X*0 | *X*1 | *X*2 | *X*3 | *X*4 | *X*5 | *X*6 | *X*7 | *Y*2 | *Y*1 | *Y*0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 4 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

При побудові шифратора на елементах АБО (рис. 24) для одержання на виході натурального двійкового коду враховують, що одиницю в молодшому розряді такого коду мають непарні десяткові цифри 1, 3, 5, 7, ... , тобто на виході молодшого розряду повинна бути 1, якщо вона є на вході № 1 або на вході № 3 і т.д. Тому входи під вказаними номерами через елемент АБО з’єднуються з виходом молодшого розряду. Одиницю в другому розряді двійкового коду мають десяткові цифри 2, 3, 6, 7, . . .; входи з цими номерами через елемент АБО повинні підключатися до виходу шифратора, на якому встановлюється другий розряд коду. Аналогічно, входи 4, 5, 6, 7,... через елемент АБО повинні бути з’єднані з виходом, на якому встановлюється третій розряд, тому що їхні коди мають в цьому розряді одиницю, і т.д. В непріоритетному дешифраторі якщо на два і більше входи одночасно подається 1, то на виході вихідні сигнали будуть накладатися.

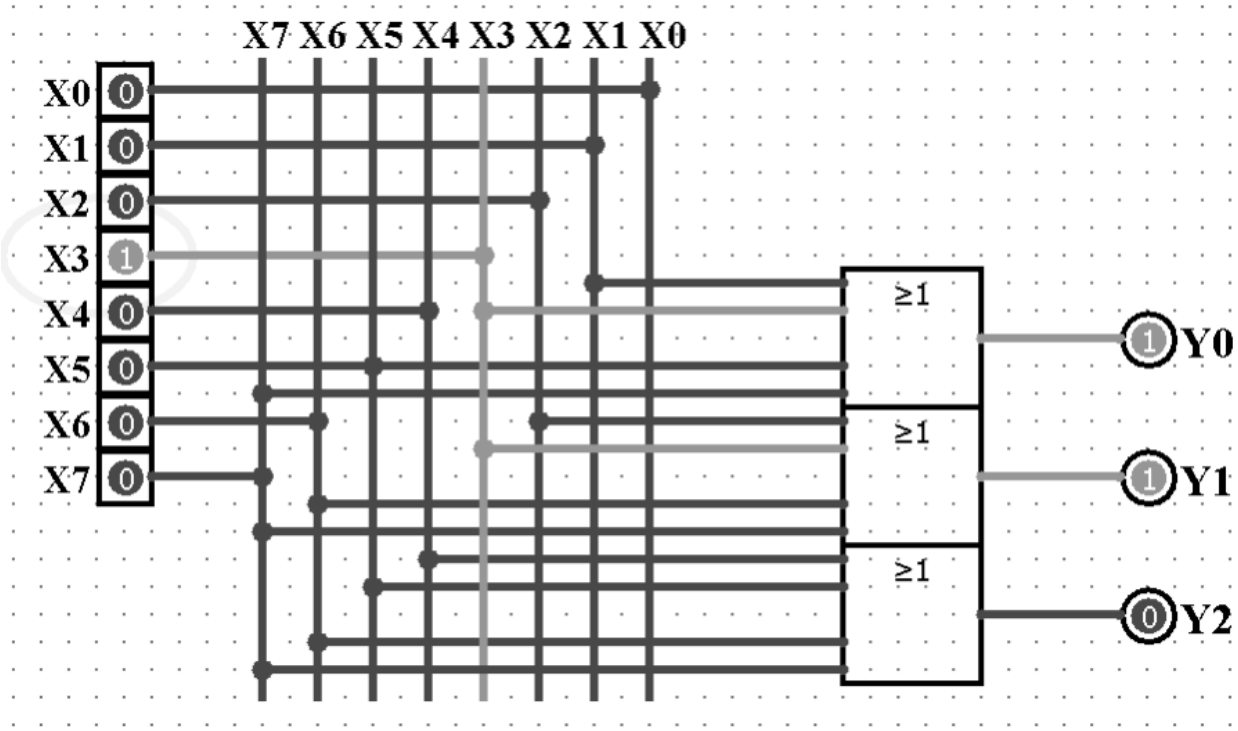



Рис. 24. Схема непріоритетного шифратора на 8 входів

Для побудови пріоритетного шифратора, в якому може бути збуджений більше як один з входів потрібно в таблиці істинності врахувати всі можливі значення на входах. Побудову такого дешифратора розглянемо на прикладі дешифратора на 4 входи і 2 виходи. Пріоритетний шифратор означає, що на його виходах буде ідентифікуватись номер найстаршого із збуджених розрядів вхідного коду. В табл. 12 подано таблицю істинності пріоритетного шифратора на 4 входи. Крім того додамо вихід Enable Output, який буде визначати відсутність збуджених входів.

Побудуємо діаграми Вейча для синтезу пріоритетного шифратора (рис. 25). З Діаграм Вейча отримаємо логічні вирази виду:

$$Y0=X3 \vee X1 \cdot \overline{X2} = \overline{\overline{X3} \cdot \overline{X1 \cdot \overline{X2}}},$$

$$Y1=X3 \vee X2 = \overline{\overline{X3} \cdot \overline{X2}}\ .$$

Вихід Enable Output буде реалізовувати наступна функція (рис. 26):

$$\text{Enable Output} = \overline{X3} \cdot \overline{X2} \cdot \overline{X1} \cdot \overline{X0}.$$

Таблиця 12

**Таблиця істинності повного пріоритетного шифратора на 4 входи**

| Входи | | | | Виходи | | |
|---|---|---|---|---|---|---|
| *X*3 | *X*2 | *X*1 | *X*0 | *Y*1 | *Y*0 | Enable Output |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 |

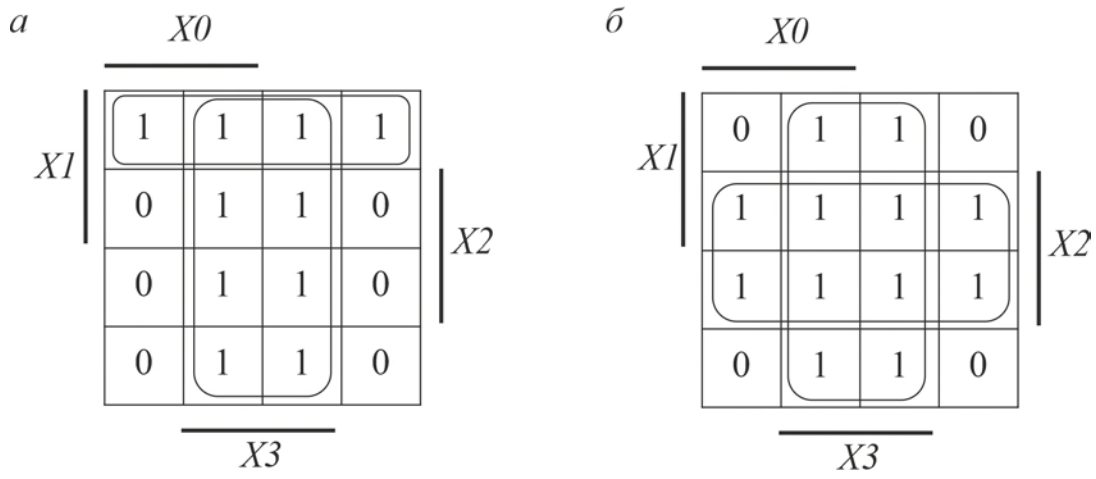



Рис. 25. Діаграми Вейча для синтезу пріоритетного шифратора на 4 входи і 2 виходи згідно табл. 10:

*а*) діаграма Вейча для функції *Y*0, *б*) діаграма Вейча для функції *Y*1

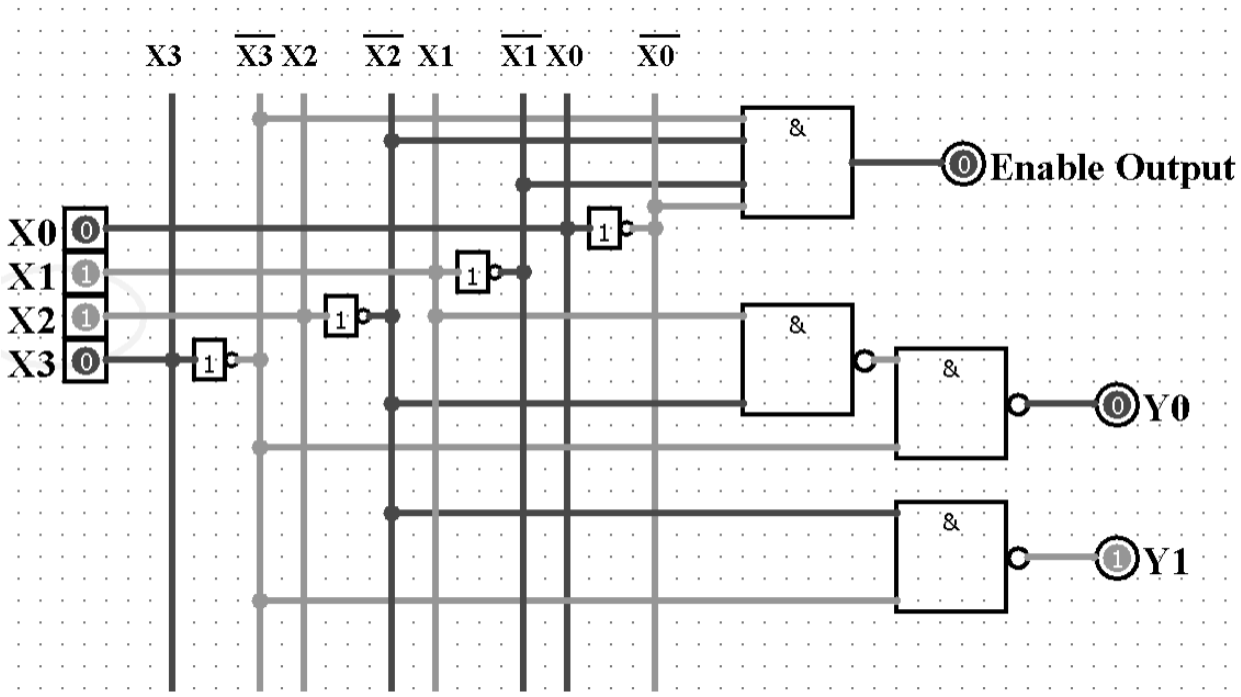



Рис. 26. Схема пріоритетного шифратора на 4 входи

Принцип роботи пріоритетного шифратора можна розглянути на прикладі мікросхеми 74147. У мікросхемі 74147 дев’ять входів (з 1 по 9) і чотири виходи двійкового коду.

Активним в мікросхемі 74147 є низький рівень вхідного сигналу. Якщо на входи не подається напруга низького рівня, то на всіх виходах встановлюється напруга високого рівня (відповідає десятковому числу 0). Якщо напруга низького рівня подається на один із входів, то на виходах формується двійковий код, який відповідає цьому входу (в інверсному вигляді).

Наприклад, якщо напруга низького рівня подається на вхід 6, то на виходах встановлюються наступні значення: $A0=1$, $A1=0$, $A2=0$, $A3=1$ (цифра 6 в двійковій-десятковому коді має вигляд 0110, а в інверсному вигляді при активному низькому рівні сигналу вона дорівнює 1001). Якщо на два і більше входи мікросхеми 74147 одночасно подається напруга низького рівня, формується код, відповідний входу з найбільшим порядковим номером (найвищим пріоритетом), а стан інших входів ігнорується. Наприклад, коли на входи 4 і 6 мікросхеми 74147 одночасно подається напруга низького рівня,

вихідний сигнал дорівнює 1001; якщо ж на входи 4 і 7 подається одночасно напруга низького рівня, на виходах формується двійковий код 1000.

### 3.5. Застосування шифраторів

Найбільше застосування вони знаходить у пристроях введення інформації (пультах управління) для перетворення десяткових чисел в двійкову систему числення. Припустимо, на пульті десять клавіш з гравіюванням від 0 до 9. При натисненні будь-якої із них на вхід шифратора подається одиничний сигнал (*X*0, ..., *X*9). На виході шифратора повинен з'явитися двійковий код (*Y*0, ..., *Y*3) цього десяткового числа.

Шифратор може бути організований не тільки для подання (кодування) десяткового числа двійковим кодом, але і для видачі певного коду (його значення заздалегідь вибирається), наприклад, при натисненні клавіші з відповідним символом. З появою даного коду система сповіщається про те, що натиснено певну клавішу клавіатури.

Шифратори застосовуються в пристроях, що перетворюють один вид коду в інший. При цьому спочатку дешифрується комбінація вихідного коду, у результаті чого на відповідному виході дешифратора з'являється логічна 1. Це відображення вхідного коду, значення якого визначено номером збудженого виходу дешифратора, подається на шифратор, організований таким чином, щоб кожний вхідний код викликав появу заданого вихідного коду.

**Завдання**

**1.** Побудувати дешифратор на базі логічних елементів:

*a)* на 2 входи на елементах АБО та НЕ;

*b)* на 2 входи на елементах І та НЕ;

*c)* на 3 входи на елементах АБО та НЕ;

*d)* на 3 входи на елементах І та НЕ;

*e)* на 4 входи на елементах АБО та НЕ;

*f)* на 4 входи на елементах І та НЕ.

**2.** На основі дешифратора на 3 входи (дешифратор взяти з бібліотеки Logisim) продемонструвати розширення розрядності до:

*a)* 24 виходів;

*b)* 20 виходів;

*c)* 18 виходів;

*d)* 16 виходів;

*e)* 12 виходів;

*f)* 10 виходів.

**3.** На основі дешифратора на 4 входи (дешифратор взяти з бібліотеки Logisim) продемонструвати розширення розрядності до:

*a)* 34 виходів;

*b)* 32 виходів;

*c)* 30 виходів;

*d)* 28 виходів;

*e)* 20 виходів;

*f)* 18 виходів.

**4.** На основі дешифратора (дешифратор взяти з бібліотеки Logisim) побудувати логічну функцію:

*a)* $Y = \overline{X_3}X_2\overline{X_1} + X_3\overline{X_2}\,\overline{X_1} + X_3\overline{X_2}X_1;$

*b)* $Y = X_3X_2\overline{X_1} + X_3\overline{X_2}\,\overline{X_1} + X_3X_2X_1;$

*c)* $Y = \overline{X_3}\,\overline{X_2}\,\overline{X_1} + \overline{X_3}X_2X_1 + X_3X_2\overline{X_1};$

*d)* $Y = X_3X_2\overline{X_1} + X_3\overline{X_2}\,\overline{X_1} + X_3X_2X_1 + \overline{X_3}\,\overline{X_2}\,\overline{X_1};$

*e)* $Y = X_3X_2\overline{X_1} + X_3\overline{X_2}X_1 + X_3X_2X_1 + \overline{X_3}X_2X_1;$

*f)* $Y = \overline{X_3}X_2\overline{X_1} + X_3\overline{X_2}\,\overline{X_1} + X_3\overline{X_2}X_1 + \overline{X_3}X_2\overline{X_1}.$

**5.** Розглянути принцип дії дешифратора реалізованого на мікросхемі 74154.

**6.** Побудувати непріоритетний шифратор на базі логічних елементів:

*a)* на 4 входи;

*b)* на 8 входів;

*c)* на 16 входів;

**7.** Побудувати пріоритетний шифратор на базі логічних елементів:

*a)* на 4 входи;

*b)* на 8 входів;

*c)* на 16 входів;

**8.** Розглянути принцип дії пріоритетного шифратора реалізованого на мікросхемі 74147.

**9.** Розглянути принцип дії пріоритетного шифратора реалізованого на мікросхемі 74148.

## Питання для самоконтролю

1. Як працює дешифратор?
2. Яку мінімальну розрядність повинен мати дешифратор для адресації одинадцяти пристроїв?
3. Скільки входів повинен мати неповний дешифратор, що має 10 виходів?
4. В яких пристроях використовується дешифратор?
5. Як працює шифратор? При розв’язуванні яких задач він використовується?
6. Який номер збудженого входу шифратора, якщо на виході встановився код 0110 і керуючий сигнал високого рівня?
7. Який номер збудженого входу шифратора, якщо на виході встановився код 0110 і керуючий сигнал низького рівня?
8. В чому полягає зміст слова “пріоритетний” в назві шифратора типу 74148?
9. За яким принципом виконується розширення розрядності дешифратора?
10. Як на базі дешифратора можуть бути реалізовані логічні функції?

# 4. ПРОЕКТУВАННЯ МУЛЬТИПЛЕКСОРІВ ТА ДЕМУЛЬТИПЛЕКСОРІВ

*Мультиплексор* – цифровий пристрій комбінаційного типу, який є комутатором сигналів з одного із декількох інформаційних входів на єдиний вихід. Вибір того або іншого входу визначається кодом, який встановлюється на адресних входах мультиплексора. Це дозволяє при зміні кодів передавати на вихід цифрову інформацію то з одного, то з іншого каналу. *Демультиплексор* є комутатором єдиного інформаційного входу на один із декількох виходів. Вибір того або іншого виходу визначається кодом на адресних входах демультиплексора. Таким чином, демультиплексор вирішує задачу, обернену мультиплексору: при зміні кодів він може передавати цифрову інформацію то в один, то в інший канал з одного входу.

## 4.1. Мультиплексори. Структура мультиплексора

*Мультиплексор* комутує на вихід *Y* сигнал з одного з інформаційних входів, один із входів який вибирається (адресується) двійковим кодом на адресних входах. Мультиплексор має $2^n$ інформаційних входів (*D*0, *D*1, … $D2^{n-1}$), *n* – адресних входів (*A*1, *A*2,… *An*) та один вихід. Крім того мультиплексор може мати входи дозволу мультиплексування.

На рис. 27 наведена функціональна схема мультиплексора. Він має вісім інформаційних (*D*0...*D*7) і три адресні (*A*1...*A*3) входи.

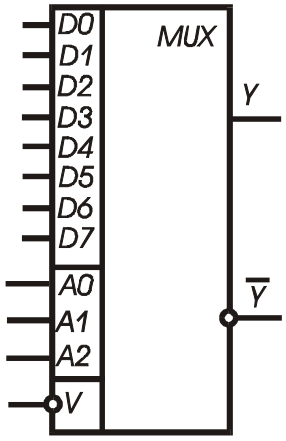



Рис. 27. Функціональна схема мультиплексора

На адресні входи надходить трирозрядний цифровий код, повне число комбінацій якого рівне восьми. Код 111 повинен забезпечити з'єднання виходу *Y* з входом *D*7, код 110 із входом *D*6 і т. д., код 000 – з входом *D*0.

Як і в дешифраторі, для розблокування кон'юнктора його входи треба з'єднати безпосередньо з тими адресними входами, на яких при даному коді присутні одиниці, і через інвертори – з тими входами, на яких знаходяться нулі. Так, наприклад, на другий зверху кон'юнктор (рис. 28) безпосередньо надходить сигнал з входу *A*0 і через інвертори – з входів *A*2 і *A*1, що при коді адреси 001 забезпечить на цьому кон'юнкторі три логічні 1, тобто підключення до виходу мультиплексора входу *D*1. Входи нижнього кон'юнктора сполучені з адресними входами безпосередньо, що при коді адреси 111 забезпечить підключення до виходу мультиплексора інформаційного входу D7.

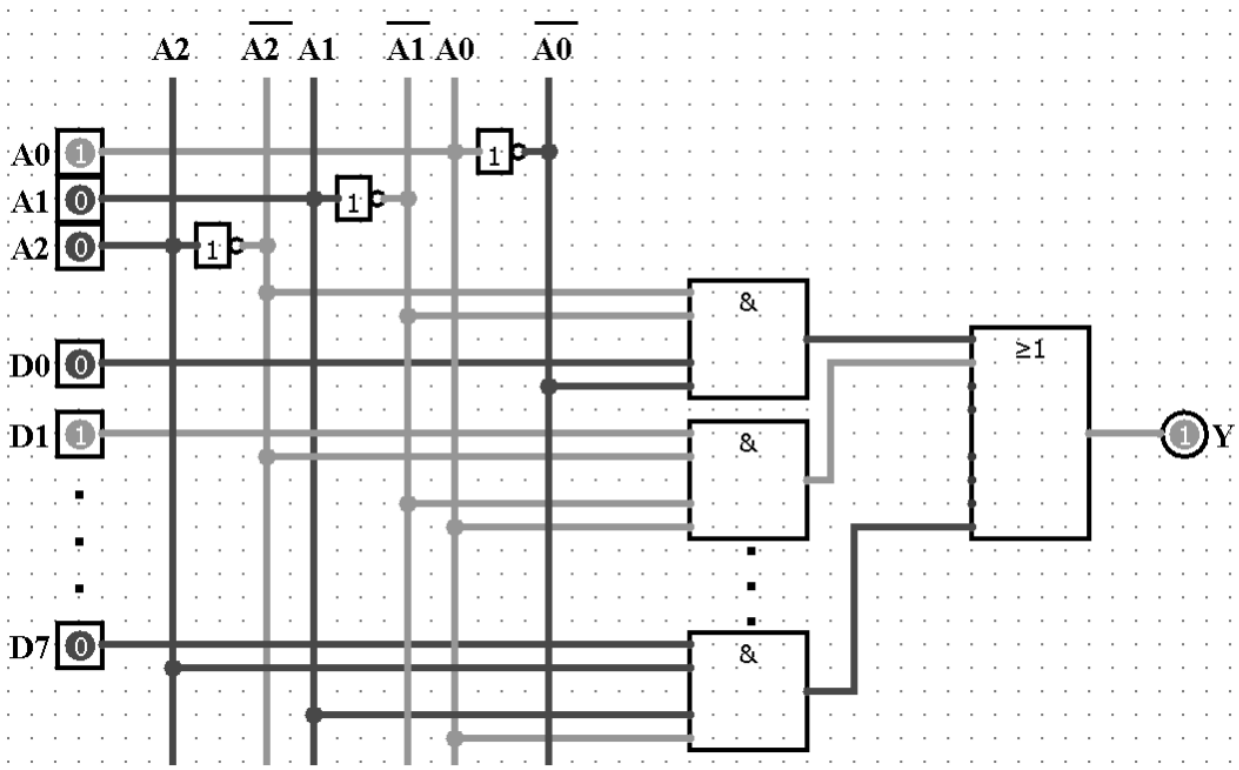



Рис. 28. Схема мультиплексора, побудованого на логічних елементах

Функціонування мультиплексора, який має три адресних входи і вісім інформаційних, подається логічним виразом:

$$Y = D0 \cdot \overline{A2} \cdot \overline{A1} \cdot \overline{A0} \vee D1 \cdot \overline{A2} \cdot \overline{A1} \cdot A0 \vee D2 \cdot \overline{A2} \cdot A1 \cdot \overline{A0} \vee D3 \cdot \overline{A2} \cdot A1 \cdot A0 \vee D4 \cdot A2 \cdot \overline{A1} \cdot \overline{A0} \vee D5 \cdot A2 \cdot \overline{A1} \cdot A0 \vee D6 \cdot A2 \cdot A1 \cdot \overline{A0} \vee D7 \cdot A2 \cdot A1 \cdot A0.$$

На рис. 29 зображена схема мультиплексора з інверсним входом $V$ дозволу мультиплексування. За наявності на вході V логічної 1 мультиплексор блокується: на прямому виході встановлюється логічний 0 незалежно від потенціалів на інформаційних входах. Функціонування такого мультиплексора можна описати наступним логічним виразом:

$$Y = D0 \cdot \overline{A2} \cdot \overline{A1} \cdot \overline{A0} \cdot \overline{V} \vee D1 \cdot \overline{A2} \cdot \overline{A1} \cdot A0 \cdot \overline{V} \vee D2 \cdot \overline{A2} \cdot A1 \cdot \overline{A0} \cdot \overline{V} \vee D3 \cdot \overline{A2} \cdot A1 \cdot A0 \cdot \overline{V} \vee D4 \cdot A2 \cdot \overline{A1} \cdot \overline{A0} \cdot \overline{V} \vee D5 \cdot A2 \cdot \overline{A1} \cdot A0 \cdot \overline{V} \vee D6 \cdot A2 \cdot A1 \cdot \overline{A0} \cdot \overline{V} \vee D7 \cdot A2 \cdot A1 \cdot A0 \cdot \overline{V}$$

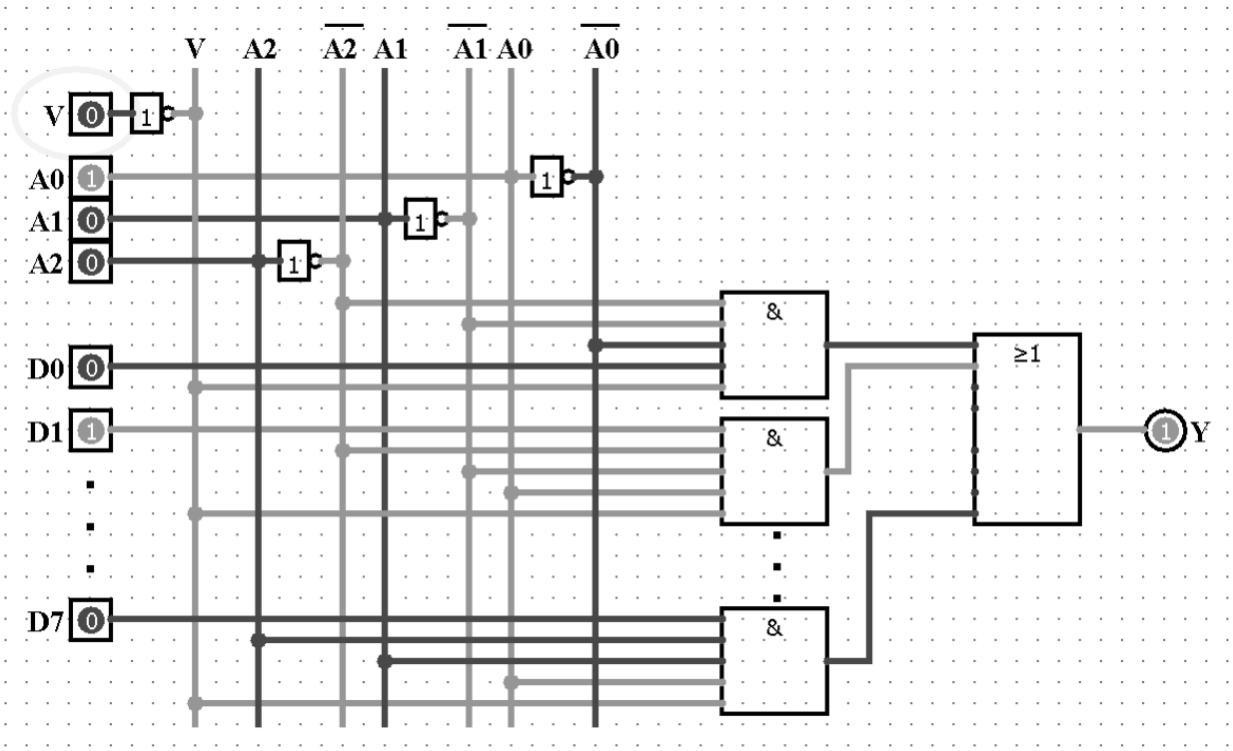



Рис. 29 Схема мультиплексора, побудованого на логічних елементах з інверсним дозволяючим входом

### 4.2. Розширення розрядності мультиплексора

Розширення розрядності мультиплексорів в загальному випадку ілюструє рис. 30. Тут «мультиплексорне дерево» містить чотири чотиривхідних мультиплексори MUX1...MUX4 із під'єднаними паралельно адресними входами *A*0, *A*1, якими одночасно вибирається один з входів *D*0…*D*3 всіх чотирьох елементів, а вихідний мультиплексор кодом на адресних входах *A*2, *A*3 вибирає один з виходів *Y*0, ..., *Y*3. Таким чином, чотирьохрозрядний код на входах *A*0, ..., *A*3 сполучає з виходом *Y* тільки один з 16 входів *D*0, ..., *D*15.

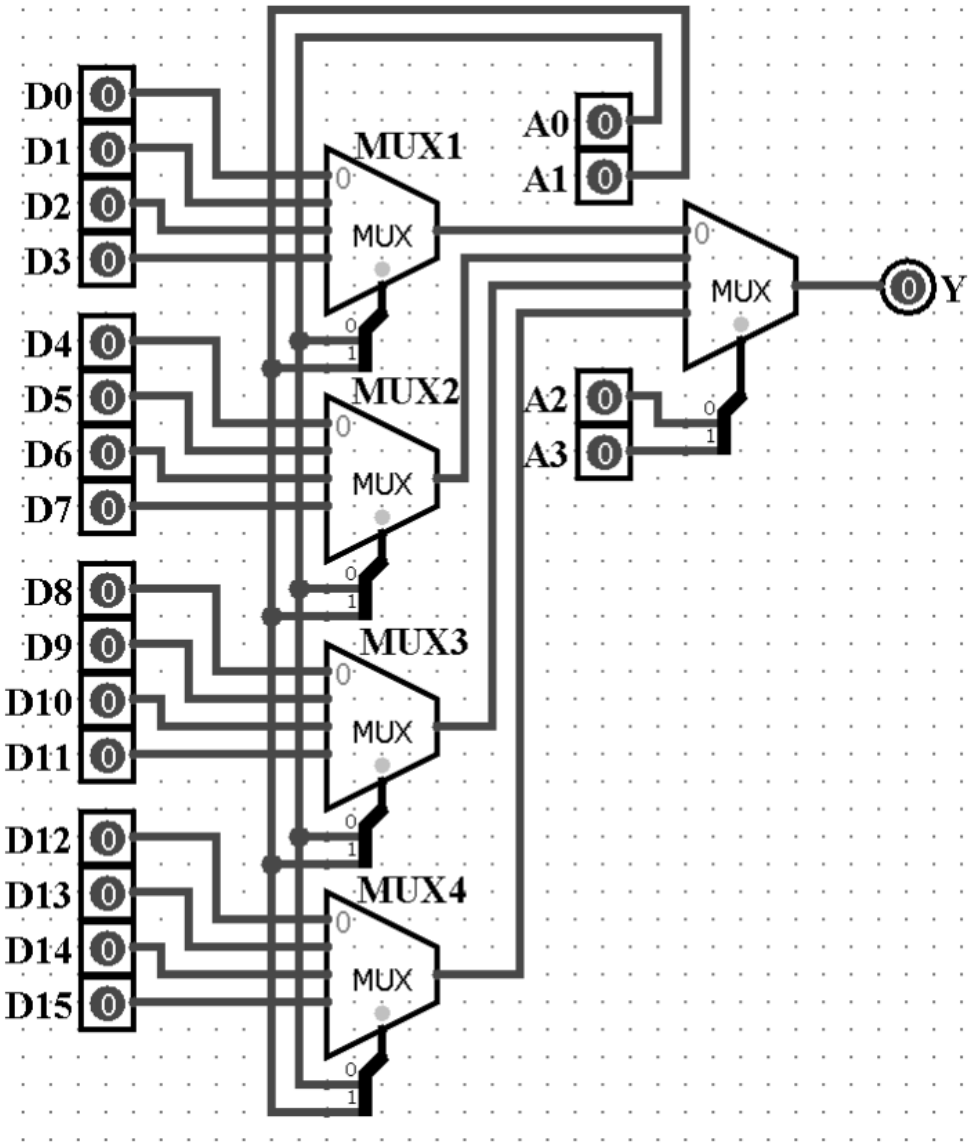



Рис. 30. Розширення розрядності мультиплексора

### 4.3. Застосування мультиплексорів

На мультиплексорі можуть бути реалізований логічні функції. Нехай, наприклад, задана функція: $Y = \overline{X_3}X_2X_1 + X_3\overline{X_2}X_1 + X_3X_2\overline{X_1}$

Логічні змінні $X_3$, $X_2$, $X_1$ подають на адресні входи $A0$, $A1$, $A2$. Коли набір $X_3$, $X_2$, $X_1$ складатиме кожну з приведених кон'юнкцій, на вихід $Y$ по черзі комутуватимуться відповідно входи $D3$, $D5$, $D6$. Оскільки кожна з них повинна забезпечити $Y$ =1, то на вказані входи слід подати логічні одиниці, а на ті, що залишилися – логічні нулі (рис. 31).

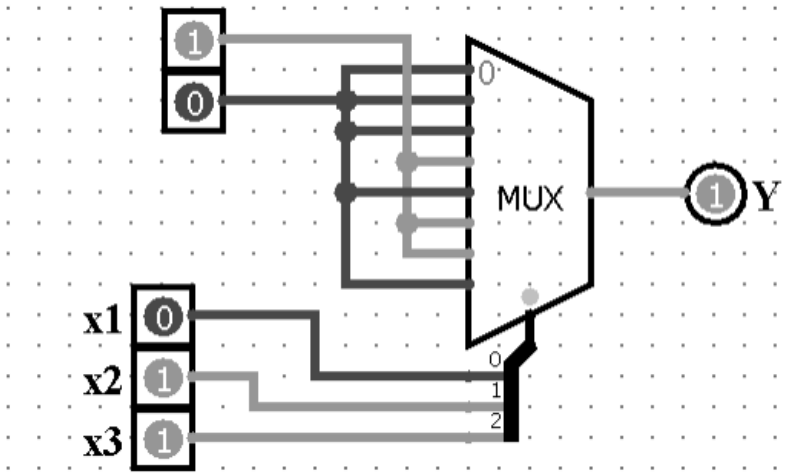



Рис. 31. Реалізація логічної функції $y = \overline{x_3}x_2x_1 + x_3\overline{x_2}x_1 + x_3x_2\overline{x_1}$ за допомогою мультиплексора

Сукупність мультиплексорів із з'єднаними адресними входами може бути використана для послідовної передачі на їх виходи декількох багаторозрядних кодів. Таку можливість забезпечить схема на рис. 30, якщо з неї виключити крайовий мультиплексор. При цьому розряди першого коду подають на входи $D0$ всіх мультиплексорів, розряди другого коду – на входи $D1$ і т.д. При зміні адреси на входах $A0$, $A1$ на виходи мультиплексорів передаватимуться розряди то першого, то другого, то третього коду. Таке часове мультиплексування використовують при необхідності передавати на одні і ті ж входи наступного пристрою то один, то інший код. Окрім комутації

з одного з *n* входів на один вихід, мультиплексор може використовуватися для перетворення паралельного коду, розряди якого подаються на входи *D*0, *D*1, ..., в послідовний код на виході *Y*. Для цього код на адресних входах повинен циклічно змінюватися, приймаючи всі послідовні значення. Таку зміну можна забезпечити, приєднавши до адресних входів виходи лічильника, які послідовно змінюють свій стан під дією імпульсів генератора.

### 4.4. Демультиплексори. Структура демультиплексора

*Демультиплексор* (рис. 32) виконує задачу, обернену мультиплексору: він комутує єдиний інформаційний вхід *D* на один з виходів, який адресується двійковим кодом на адресних входах *A*0, *A*1. Демультиплексор має *n* адресних входів і $2^n$ виходів. Схему демультиплексора будують так, щоб при подачі будь-якої логічної комбінації на адресні входи демультиплексора, тільки на один з елементів «І» буде подана логічна «1» і, відповідно, пройде інформаційний сигнал на його вихід. Всі інші логічні елементи «І» будуть заблокованими і сигнал з них не передається, оскільки на входи елементів «І», що керують цими входами подано логічний «0».

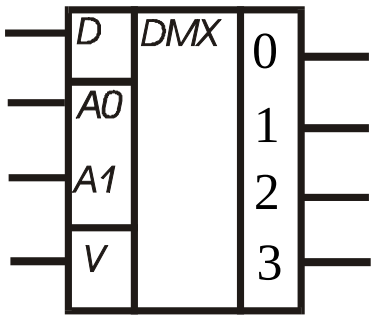



Рис. 32**.** Функціональна схема демультиплексора.

Функціонування демультиплексора, який має три адресних входи, один інформаційний вхід і вісім виходів, подається системою логічних виразів:

$$Y0 = D \cdot \overline{A2} \cdot \overline{A1} \cdot \overline{A0};$$

$$Y1 = D \cdot \overline{A2} \cdot \overline{A1} \cdot A0;$$

$$Y2 = D \cdot \overline{A2} \cdot A1 \cdot \overline{A0};$$

$$Y3 = D \cdot \overline{A2} \cdot A1 \cdot A0;$$

$$Y4 = D \cdot A2 \cdot \overline{A1} \cdot \overline{A0};$$

$$Y5 = D \cdot A2 \cdot \overline{A1} \cdot A0;$$

$$Y6 = D \cdot A2 \cdot A1 \cdot \overline{A0};$$

$$Y7 = D \cdot A2 \cdot A1 \cdot A0.$$

На рис. 33 подано логічну схему восьмиканального демультиплексора, побудованого за отриманою системою логічних виразів.

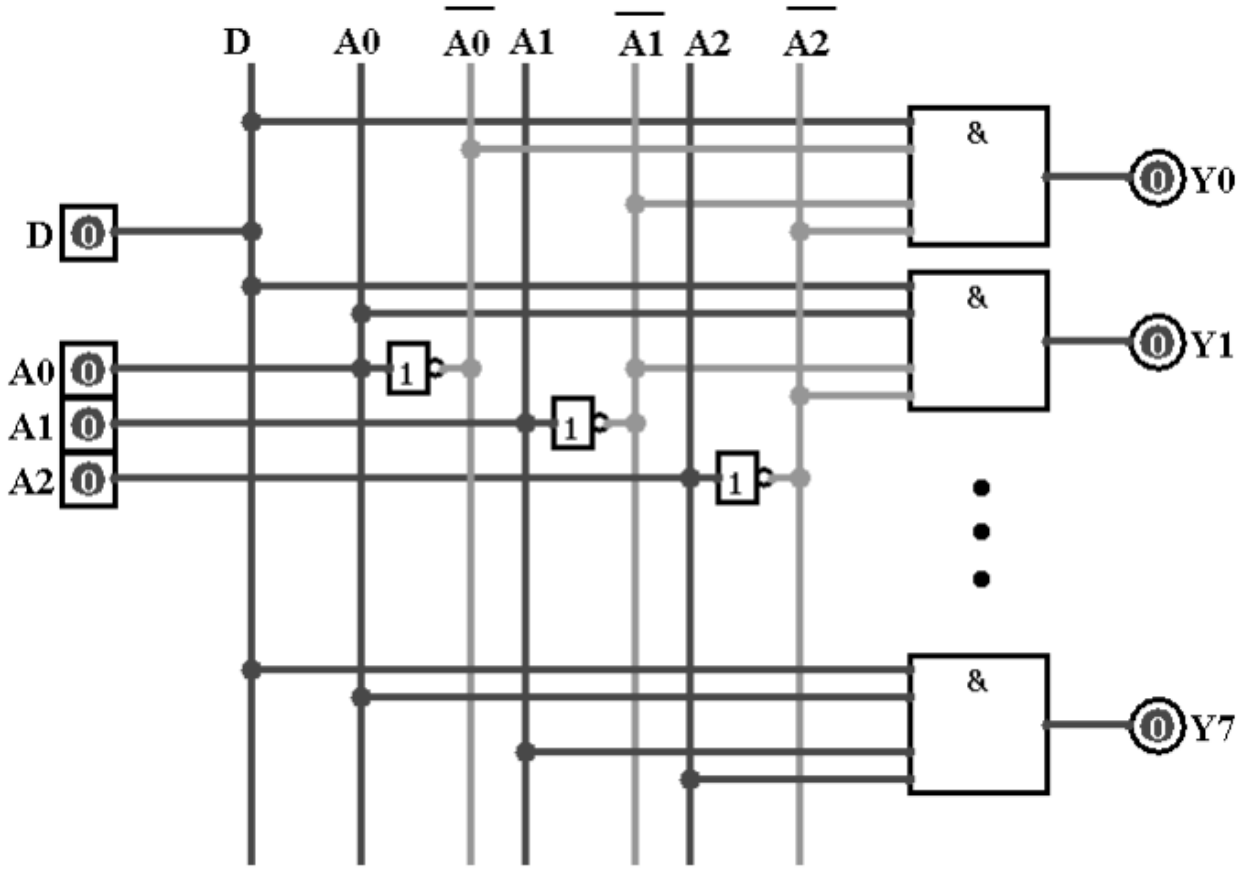



Рис. 33. Логічна схема восьмиканального демультиплексора

Мультиплексор може працювати, як дештфратор, якщо вхід *D* вважати за дозволяючий вхід дешифратора, а адресні входи демультиплексора будуть адресними входами дешифратора.

Демультиплексор також можна побудувати на базі дешифраторів. Для цього потрібно додати додатковий вхід *D*, який під'єднується до всіх виходів дешифратора через елемент «І» (рис. 34)

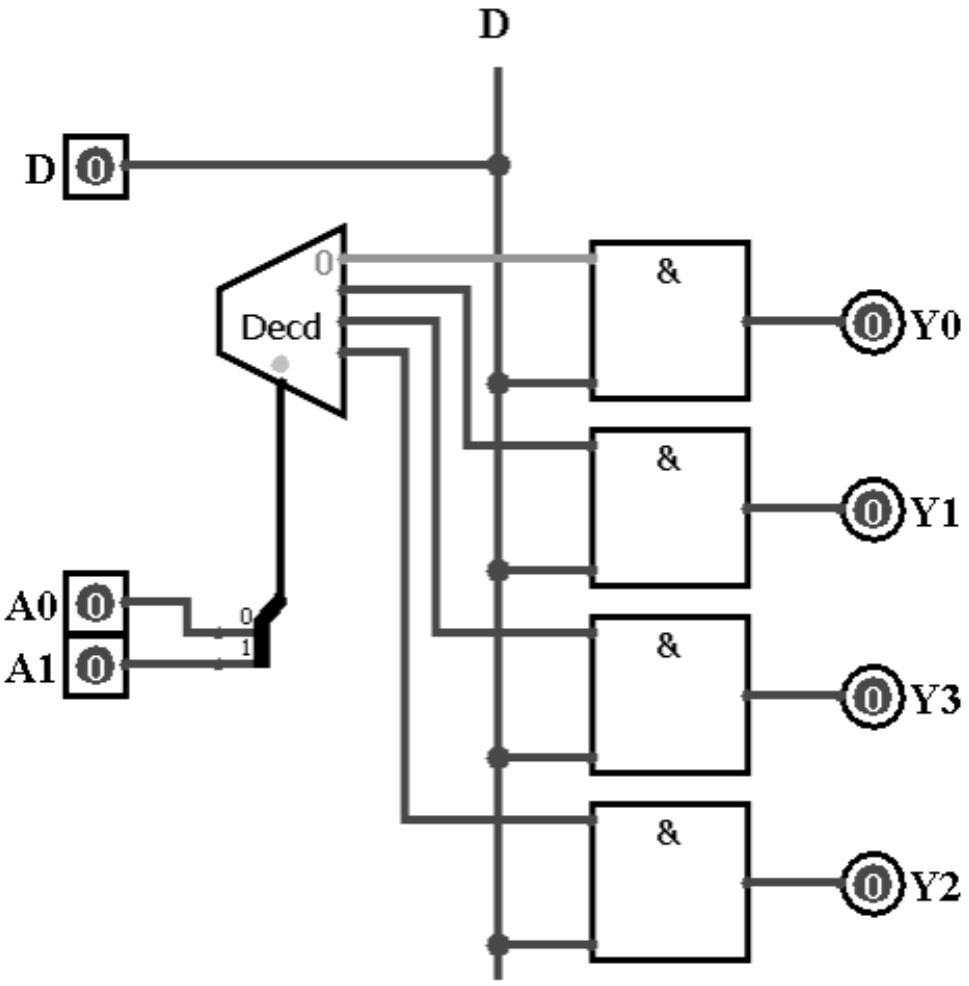



Рис. 34. Демультиплексор, побудований на основі дешифратора

### 4.5. Розширення розрядності демультиплексора

На рис. 35 показаний загальний випадок нарощування розрядності демультиплексорів. На інформаційний вхід D надходять логічні 1 і 0. Кодом на адресних входах *A*0, *A*1 вибирається один з виходів DMX0, а кодом на *A*2, *A*3 одночасно вибираються чотири однойменні виходи всіх чотирьох демультиплексорів DMX1, DMX2, DMX3, DMX4. В результаті кодом *A*3, *A*2, *A*1, *A*0 вибирається один з 16 виходів.

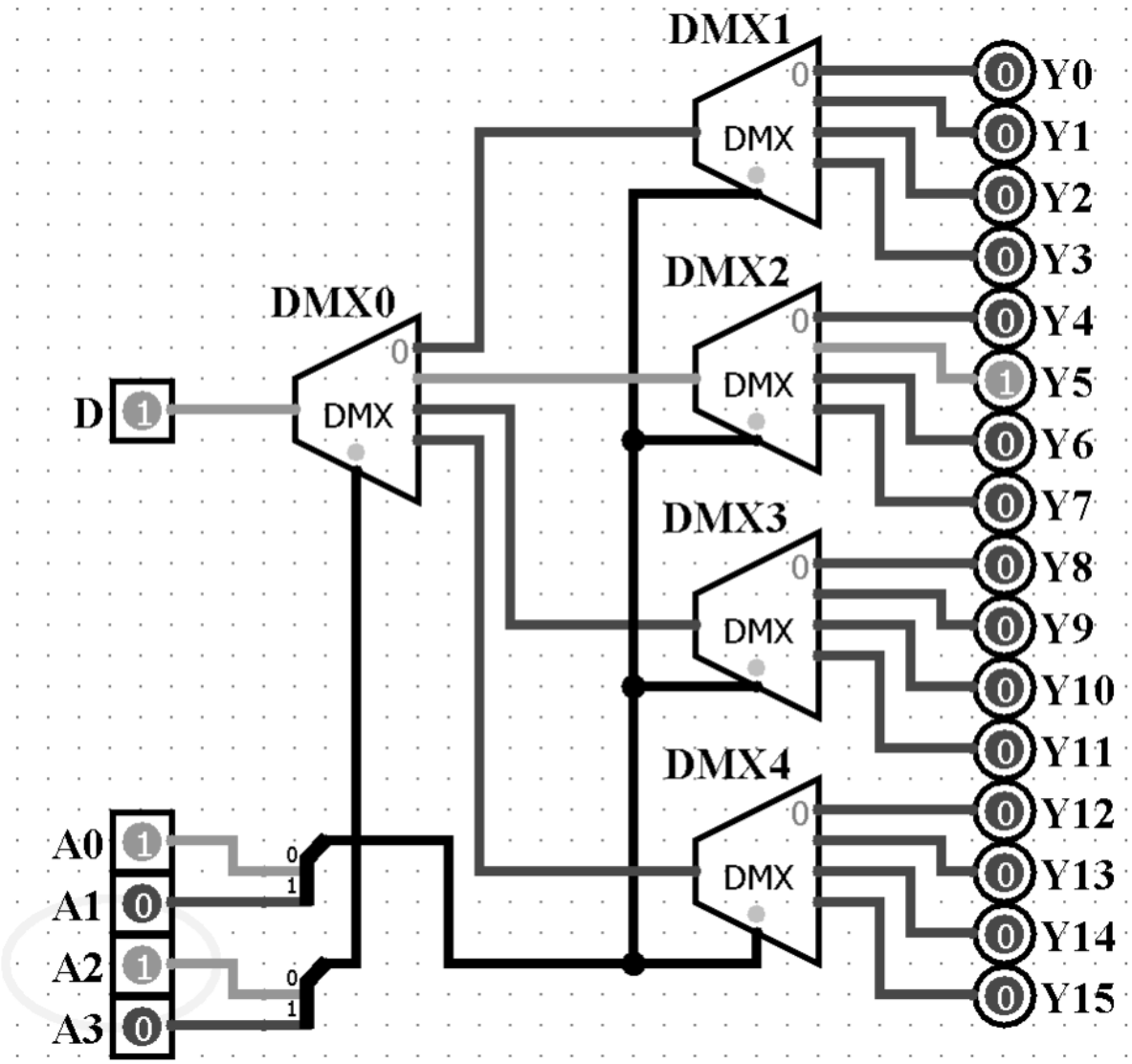



Рис. 35. Розширення розрядності демультиплексора

### 4.6. Застосування демультиплексорів

Окрім прямого призначення, демультиплексор в сукупності з мультиплексором дозволяє комутувати будь-який вхід мультиплексора з будь-яким виходом демультиплексора. Для цього вихід мультиплексора треба з’єднати з інформаційним входом демультиплексора.

Демультиплексори використовують для реалізації складних пристроїв пам’яті з простіших. В даному випадку на вхід демультиплексора подається сигнал для запису, його виходи підключаються до входів даних регістрів, а на керуючий вхід подається номер регістра, в який проводиться запис.

**Завдання**

**1.** Побудувати 4-канальний мультиплексор:

*a)* без дозволяючого входу;

*b)* з інверсним дозволяючим входом;

*c)* з прямим дозволяючим входом.

**2.** Побудувати 8-канальний мультиплексор:

*a)* без дозволяючого входу;

*b)* з інверсним дозволяючим входом;

*c)* з прямим дозволяючим входом.

**3.** Побудувати 16-канальний мультиплексор:

*a)* без дозволяючого входу;

*b)* з інверсним дозволяючим входом;

*c)* з прямим дозволяючим входом.

**4.** На базі мультиплексора складіть схему, яка реалізує функцію:

*a)* $Y = X_2X_1 + \overline{X_2}X_1 + X_2\overline{X_1} + \overline{X_2}\,\overline{X_1}$

*b)* $Y = X_3X_2X_1 + X_3\overline{X_2}X_1 + \overline{X_3}X_2\overline{X_1} + \overline{X_3}X_2X_1$

*c)* $Y = X_3X_2X_1 + X_3\overline{X_2}X_1 + \overline{X_3}X_2\overline{X_1} + \overline{X_3}\,\overline{X_2}X_1 + X_3\overline{X_2}\,\overline{X_1}$

*d)* $Y = \overline{X_4}X_3X_2X_1 + X_4X_3\overline{X_2}X_1 + X_4\overline{X_3}X_2\overline{X_1} + \overline{X_4}\,\overline{X_3}X_2X_1$

*e)* $Y = \overline{X_4}X_3\overline{X_2}X_1 + X_4\overline{X_3}X_2X_1 + X_4\overline{X_3}X_2\overline{X_1} + \overline{X_4}X_3X_2\overline{X_1}$

*f)* $Y = X_4X_3X_2X_1 + X_4\overline{X_3}X_2X_1 + X_4\overline{X_3}X_2\overline{X_1} + \overline{X_4}\,\overline{X}\,\overline{X_2}\,\overline{X_1}$

**5.** На основі мультиплексора на 3 адресні входи (мультиплексор взяти з бібліотеки Logisim) продемонструйте розширення розрядності мультиплексора до:

*a)* 24 інформаційних входів;

*b)* 20 інформаційних входів;

*c)* 18 інформаційних входів;

*d)* 16 інформаційних входів;

*e)* 12 інформаційних входів;

*f)* 10 інформаційних входів.

**6.** На основі мультиплексора на 4 адресні входи (мультиплексор взяти з бібліотеки Logisim) продемонструйте розширення розрядності мультиплексора до:

*a)* 34 інформаційних входів;

*b)* 32 інформаційних входів;

*c)* 30 інформаційних входів;

*d)* 28 інформаційних входів;

*e)* 20 інформаційних входів;

*f)* 18 інформаційних входів.

**7.** Побудувати демультиплксор на логічних елементах І та НЕ:

*a)* на 4 виходи;

*b)* на 8 виходів;

*c)* на 16 виходів.

*d)* на 32 виходи.

**8.** Побудувати демультиплксор на логічних елементах АБО та НЕ:

*a)* на 4 виходи;

*b)* на 8 виходів;

*c)* на 16 виходів.

*d)* на 32 виходи.

**9.** Побудувати демультиплксор на основі дешифратора (дешифратор взяти з бібліотеки Logisim):

*a)* на 4 виходи;

*b)* на 8 виходів;

*c)* на 16 виходів.

*d)* на 32 виходи.

**10.** На основі демультиплексора на 8 виходів (демультиплексор взяти з бібліотеки Logisim) продемонструйте розширення розрядності демультиплексора до:

*a)* 24 виходів

*b)* 20 виходів;

*c)* 18 виходів;

*d)* 16 виходів;

*e)* 12 виходів;

*f)* 10 виходів.

**11.** На основі демультиплексора на 16 виходів (демультиплексор взяти з бібліотеки Logisim) продемонструйте розширення розрядності демультиплексора до:

*a)* 34 виходів;

*b)* 32 виходів;

*c)* 30 виходів;

*d)* 28 виходів;

*e)* 20 виходів;

*f)* 18 виходів.

## Питання для самоконтролю

**1.** Що таке мультиплексор?

**2.** Скільки адресних входів повинен мати мультиплексор, якщо кількість інформаційних входів рівна вісім?

**3.** Для чого використовується мультиплексор?

**4.** Що таке демультиплексор і для розв'язку яких задач він використовується?

**5.** Яка особливість демультиплексора відрізняє його від дешифратора?

**6.** Скільки адресних входів повинен мати демультиплексор, якщо кількість виходів рівна вісім?

**7.** Скільки виходів має демультиплексор, якщо кількість адресних входів рівна 4?

**8.** Як на базі мультиплексора можуть бути реалізовані логічні функції?

**9.** Як на базі демультиплексора отримати схему дешифратора?

**10.** Як на базі дешифратора можна побудувати демультиплексор?

# 5. ПЕРЕТВОРЮВАЧІ КОДІВ

*Комбінаційні перетворювачі кодів* призначені для перетворення *m*-елементного паралельного коду на вході в *n*-елементний паралельний код на виході. Тобто, перетворювач коду це функціональний вузол, призначений для перетворення двійкового коду з однієї форми в іншу. Наприклад, при введенні інформації в ЕОМ необхідно перетворювати десяткові числа у двійкові, а при виводі інформації на індикатори або друкувальний пристрій – двійкові або двійково-десяткові коди в коди керування знакогенератором, світлодіодними або рідкокристалічними індикаторними панелями, механізмом друку.

До перетворювачів коду, також відносяться шифратори, дешифратори, мультиплексори та демультиплексори, однак ці функціональні вузли виділені в окремі самостійні класи.

## 5.1. Види перетворювачів кодів та їх синтез

Зв’язок між вхідними і вихідними сигналами можна задати таблицями істинності або логічними функціями. Синтез перетворювача коду зводиться до знаходження для кожного розряду вихідного коду булевої функції, що встановлює зв’язок даного розряду із вхідними наборами двійкових змінних. Знаходження такого зв’язку й мінімізація булевого виразу здійснюються за допомогою діаграм Вейча. На заключному етапі отримана функція перетвориться до виду, зручного для реалізації в заданому (обраному) елементному базисі. У табл. 13 наведені найпоширеніші в цифровій схемотехніці двійкові коди. У позначеннях кодів 8421, 7421, 5421, 2421. Код Грея утворений послідовністю двійкових чисел, у якій два будь-яких сусідніх числа (перше й останнє число також вважаються сусідніми) відрізняються тільки одним розрядом. У коді Джонсона перехід до наступного числа здійснюється послідовною заміною 0 на 1, починаючи справа, а після установки у всіх розрядах 1 – заміною 1 на 0.

Таблиця 13

**Найпоширеніші в цифровий схемотехніці двійкові перетворювачі кодів**

| Десят-кове число | Код 8421 | Код 7421 | Код 5421 | Код Айкена 2421 | Код Грея | Доповне-ння до 9: 9-$N$ | Доповне-ння до 10: 10- $N$ | Код Джонсона |
|---|---|---|---|---|---|---|---|---|
| 0 | 0000 | 0000 | 0000 | 0000 | 0000 | 1001 | 1010 | 00000 |
| 1 | 0001 | 0001 | 0001 | 0001 | 0001 | 1000 | 1001 | 00001 |
| 2 | 0010 | 0010 | 0010 | 0010 | 0011 | 0111 | 1000 | 00011 |
| 3 | 0011 | 0011 | 0011 | 0011 | 0010 | 0110 | 0111 | 00111 |
| 4 | 0100 | 0100 | 0100 | 0100 | 0110 | 0101 | 0110 | 01111 |
| 5 | 0101 | 0101 | 1000 | 1011 | 0111 | 0100 | 0101 | 11111 |
| 6 | 0110 | 0110 | 1001 | 1100 | 0101 | 0011 | 0100 | 11110 |
| 7 | 0111 | 1000 | 1010 | 1101 | 0100 | 0010 | 0011 | 11100 |
| 8 | 1000 | 1001 | 1011 | 1110 | 1100 | 0001 | 0010 | 11000 |
| 9 | 1001 | 1010 | 1100 | 1111 | 1101 | 0000 | 0001 | 10000 |

Дуже поширеними є прямий, обернений і доповнювальний коди, які забезпечують подання знаку числа і заміну операції віднімання додаванням, а також визначення переповнення розрядної сітки. Для подання знаку числа у цих перетворювачах відводиться знаковий розряд, який розташовується зліва від числа і відділяється комою. У знаковий розряд записується нуль – для додатного числа і одиниця – для від’ємного.

### 5.2. Перетворювачі на дешифраторі та шифраторі

Перетворювач кодів описується системою логічних рівнянь, де аргументами є елементи перетворюваного коду, а функцією – кожен елемент перетвореного. Принцип перетворення на дешифраторі і шифраторі полягає в тому, що кожному вхідному коду дешифратора ставиться у відповідність збуджений вихід, якому шифратор ставить у відповідність вихідний код.

Перетворювач кодів на основі дешифратора, який працює згідно таблиці істинності (табл. 14.) подано на рис. 36.

Таблиця 14

**Таблиця істинності перетворювача кодів**

| Інформація | | | | | |
|---|---|---|---|---|---|
| на вході | | | на виході | | |
| *X*2 | *X*1 | *X*0 | *Y*2 | *Y*1 | *Y*0 |
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 |

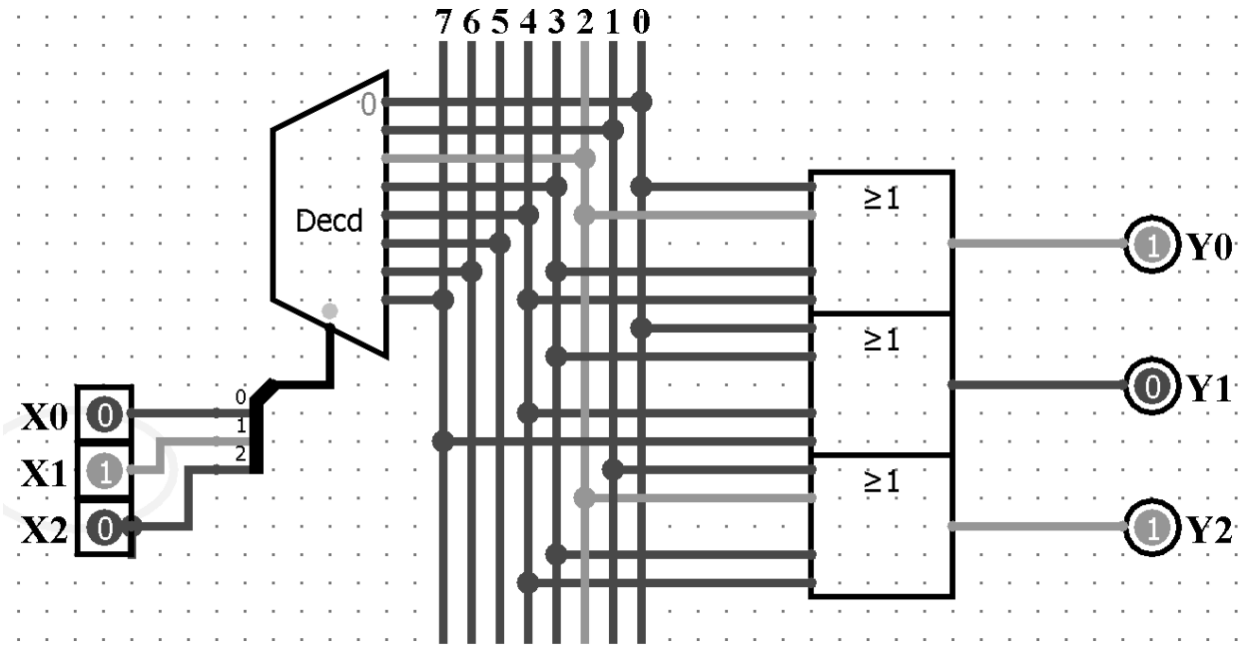



Рис. 36. Перетворювач коду, який працює згідно таблиці істинності, поданої в табл. 14

### 5.3. Перетворювач двійково-десяткового коду в код семисегментного індикатора

Індикація чисел на табло і пультах проводиться, як правило, в десятковому вигляді. Ми знаємо, що для цього можна використовувати семисегментні світлодіодні або рідкокристалічні індикатори. Подаючи ккеруючу напругу на окремі елементи індикатора і викликаючи його світіння (світлодіодні індикатори) або змінюючи його забарвлення (рідкокристалічні індикатори), ми можемо одержувати зображення десяткових цифр 0, 1, …, 9 (рис. 37.).

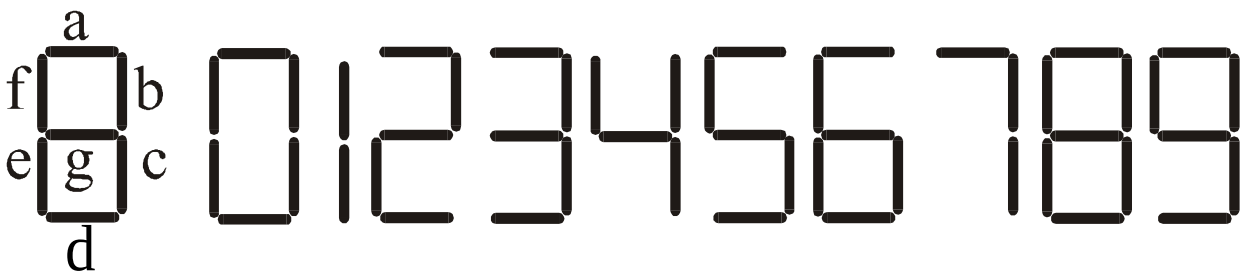



Рис. 37. Зображенні цифр на семисегментних індикаторах

Для зручності перекладу двійкової інформації в десятковий вигляд часто використовують двійково-десятковий код (або код 8421), тобто представлення десяткових чисел у вигляді чотирьохрозрядних двійкових чисел.

Наприклад, $9_{(10)} = 1 \cdot 8 + 0 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 = 1001_{(2)}$

$75_{(10)} = 0111_{(2)}\ \ 0101_{(2)}$;

$910(10) = 1001_{(2)}\ \ 0001_{(2)}\ \ 0000_{(2)}$.

Закон функціонування перетворювача двійково-десяткового коду в код семисегментного індикатора працює відповідно до табл. 15:

Побудуємо перетворювач двійково-десяткового коду в код семисегментного індикатора на основі дешифратора. Дешифратор потрібно взяти на 4 входи. Для побудови використовуємо тільки 10 виходів. В якості індикатора, для зручності, візьмемо компонент «7-сисегментный индикатор» з бібліотеки «Ввод/вывод» (рис. 38).

Таблиця 15

**Таблиця істинності перетворювача двійково-десяткового коду в код семисегментного індикатора**

| цифра | Двійково-десятковий код "8421" | | | | Семисегментний код | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | *X*3 | *X*2 | *X*1 | *X*0 | *a* | *b* | *c* | *d* | *e* | *f* | *g* |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 7 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 8 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 9 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |

Для того, щоб побудувати перетворювач двійково-десяткового коду в код семисегментного індикатора на основі логічних елементів, потрібно для кожного виходу отримати логічну функцію. Для цього побудуємо діаграми Вейча (рис. 39) та отримаємо значення функцій у вигляді МДНФ.

Функціонування перетворювача двійково-десяткового коду в код семисегментного індикатора, можна подати системою логічних виразів: функцій *a*, *b*, *c*, *d*, *e*, *f*, *g*.

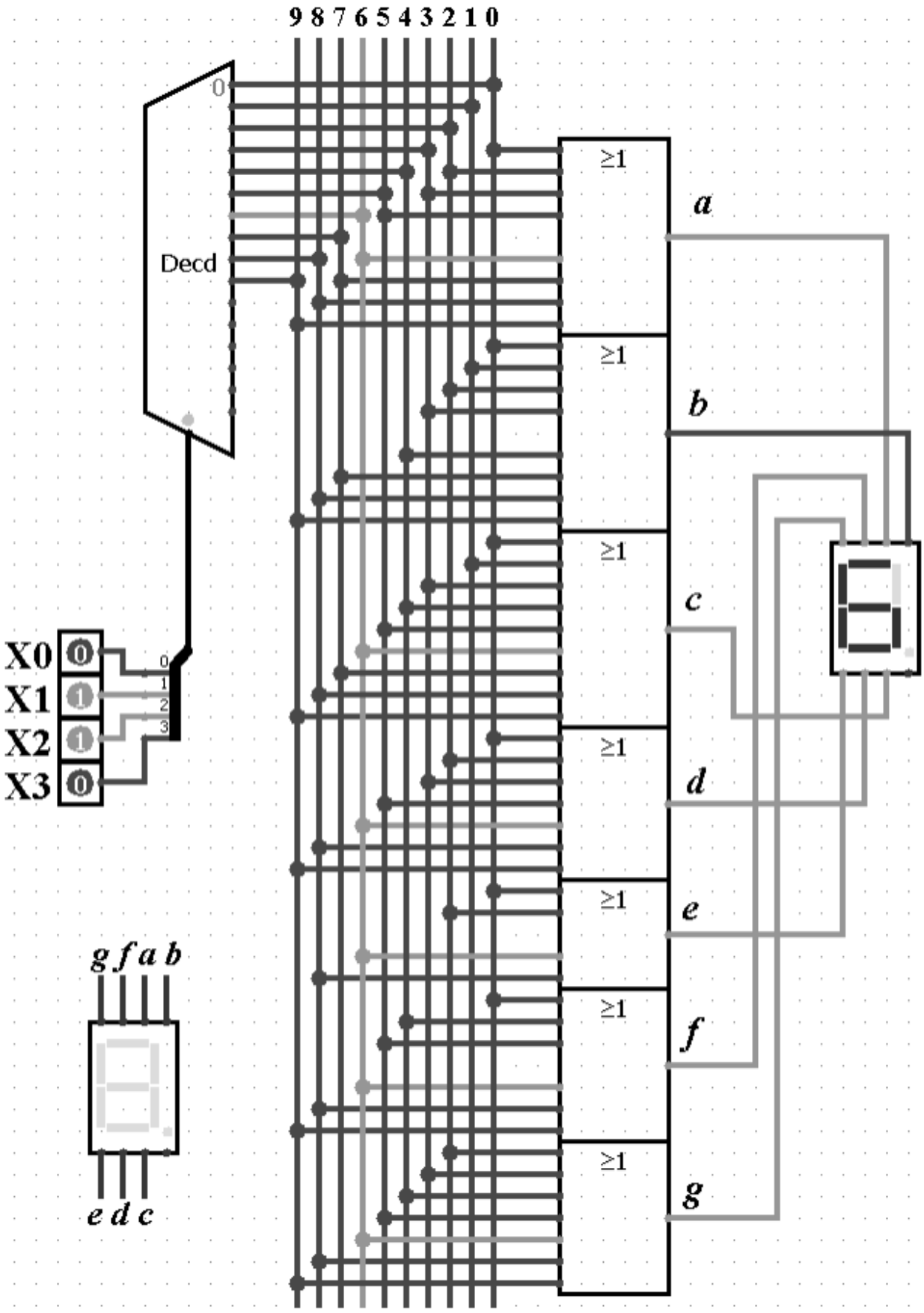



Рис. 38. Перетворювач двійково-десяткового коду в код семисегментного індикатора на основі дешифратора

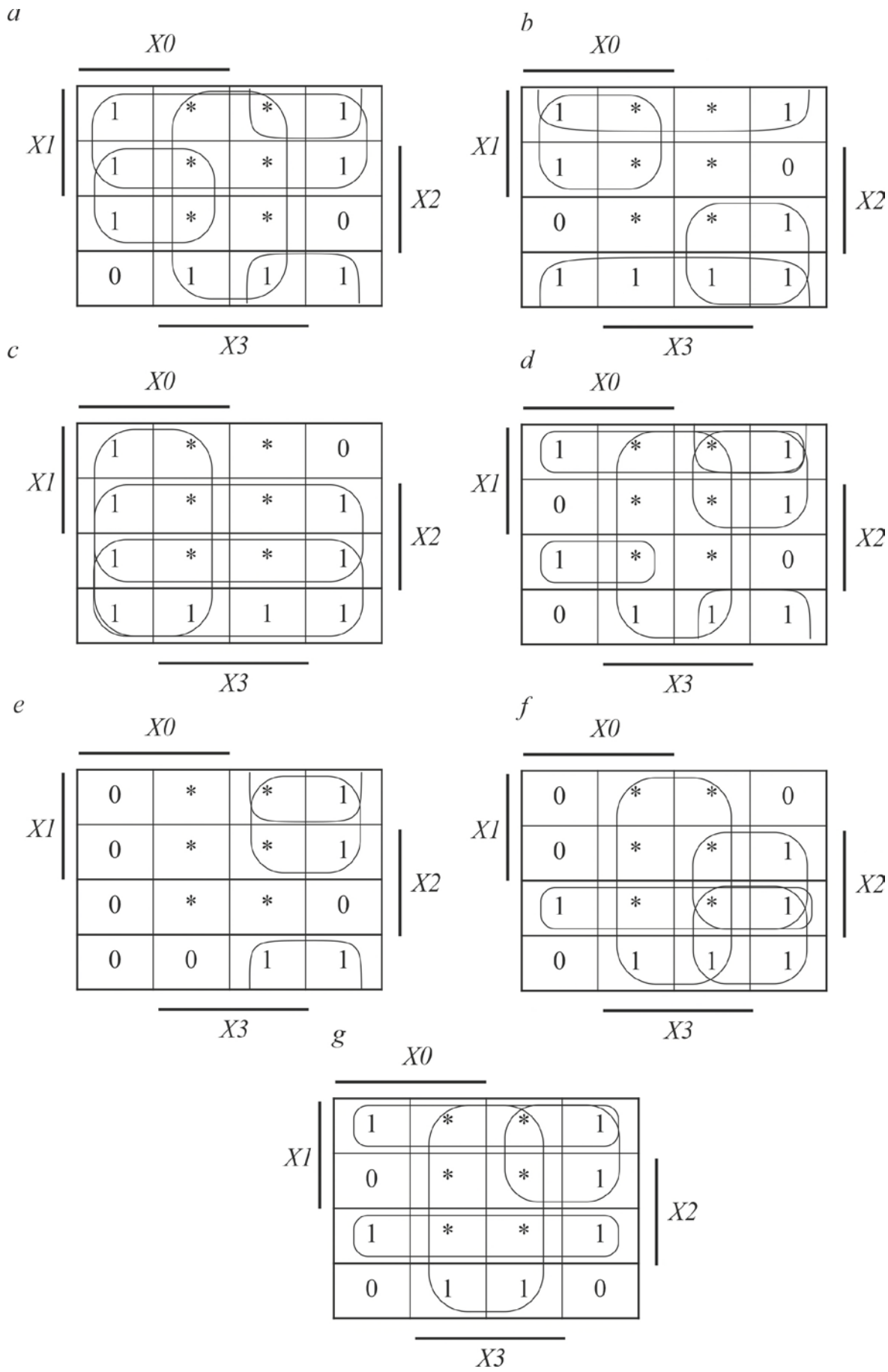



Рис. 39. Діаграми Вейча для функцій *a*, *b*, *c*, *d*, *e*, *f*, *g* перетворювача двійково-десяткового коду в код семисегментного індикатора

$a=X1 \vee X0 \cdot X2 \vee X3 \vee \overline{X0} \cdot \overline{X2}$,

$b=X0 \cdot X1 \vee \overline{X2} \vee \overline{X0} \cdot \overline{X1}$,

$c=X0 \vee \overline{X1} \vee X2$,

$d=X1 \cdot \overline{X2} \vee \overline{X3} \vee \overline{X0} \cdot X1 \vee X0 \cdot \overline{X1} \cdot X2 \vee \overline{X0} \cdot \overline{X2}$ ,

$e=\overline{X0} \cdot X1 \vee \overline{X0} \cdot \overline{X2}$,

$f=X3 \vee \overline{X0} \cdot X2 \vee \overline{X0} \cdot \overline{X1} \vee \overline{X1} \cdot X2$,

$g=X3 \vee X1 \cdot \overline{X2} \vee \overline{X1} \cdot X2 \vee \overline{X0} \cdot X1$.

## Завдання

**1.** Використовуючи чотиривхідний дешифратор, побудувати перетворювач кодів з коду 8421 в:

*а*) в код 7421;

*б*) в код 5421;

*в*) в код 2421;

*г*) в код Грея;

*д*) в код доповнення до 9;

*е*) в код доповнення до 10;

*є*) в код Джонсона.

**2.** На логічних елементах І, НЕ та І-НЕ побудувати перетворювач кодів з коду 8421 в:

*а*) в код 7421;

*б*) в код 5421;

*в*) в код 2421;

*г*) в код Грея;

*д*) в код доповнення до 9;

*е*) в код доповнення до 10;

*є*) в код Джонсона.

**3.** На логічних елементах АБО, НЕ та АБО-НЕ побудувати перетворювач кодів з коду 8421 в:

*а*) в код 7421;

*б*) в код 5421;

*в*) в код 2421;

*г*) в код Грея;

*д*) в код доповнення до 9;

*е*) в код доповнення до 10;

*є*) в код Джонсона.

**4.** Використовуючи чотиривхідний дешифратор, побудувати перетворювач кодів з коду Грея в:

*а*) в код 7421;

*б*) в код 5421;

*в*) в код 2421;

*д*) в код доповнення до 9;

*е*) в код доповнення до 10;

*є*) в код Джонсона.

**5.** Побудувати перетворювач двійково-десяткового коду в код семисегментного індикатора:

*а*) на логічних елементах І, НЕ;

*б*) на логічних елементах АБО, НЕ;

*в*) використовуючи чотиривхідний дешифратор.

## Питання для самоконтролю

**1.** Які є види перетворювачів кодів?

**2.** Для чого потрібні перетворювачів кодів?

**3.** В чому полягає принцип роботи перетворювача коду на дешифраторі та шифраторі?

**4.** В якому вигляді проводиться індикація чисел на табло?

**5.** Які є види індикаторів?

# 6. ЦИФРОВИЙ КОМПАРАТОР

*Цифровий компаратор* призначений для порівняння двох двійкових чисел.

Він має дві групи входів. На одну з них надходять розряди першого числа ($a$), на другу групу – розряди другого числа ($b$). Три виходи компаратора появою логічної 1 фіксують результат порівняння. На одному виході вона встановлюється при рівності чисел ($a$=$b$), на іншому – при $a$>$b$, на третьому – при $a$<$b$.

Цифровий компаратор може використовуватися, наприклад, у системах автоматичного контролю і регулювання. При цьому число $a$ є параметром деякого процесу, а число $b$ – порогом, якого (відповідно до умов задачі) цей параметр не повинен перевищувати чи опускатися нижче нього.

У табл. 16 подається зв'язок між сигналами на виходах і входах компаратора при порівнянні однорозрядних чисел $a$ і $b$, що можуть бути рівні одиниці чи нулю. На відповідному виході з'являється логічна одиниця, коли коди на входах знаходяться в належному співвідношенні. Так, якщо $a_0$=1, $b_0$=1 (числа однакові), то функція, яка характеризує рівність чисел, $F_{a=b} = 1$, а функції, що характеризують їхню нерівність, $F_{a>b} = 0$ і $F_{a<b} = 0$. Аналогічно заповнюються інші рядки таблиці.

Таблиця 16

**Таблиця істинності функціонування однорозрядного компаратора.**

| Входи | | Виходи | | |
|---|---|---|---|---|
| $a_0$ | $b_0$ | $F_{a>b}$ | $F_{a=b}$ | $F_{a<b}$ |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 |

З табл. 16 можна записати наступні логічні функції, що характеризують співвідношення однорозрядних чисел:

$$F_{a>b} = a_0\overline{b_0},$$

$$F_{a=b} = a_0 b_0 + \overline{a_0}\,\overline{b_0},$$

$$F_{a<b} = \overline{a_0} b_0.$$

Якщо значення $a_0$ і $b_0$ такі, що права частина функції дорівнює 1, то співвідношення, зазначене в лівій частині, виконується. Якщо права частина функції дорівнює 0, то співвідношення між $a_0$ і $b_0$ протилежні зазначеному.

Схема однорозрядного компаратора, що реалізує наведені функції, зображена на рис. 40.

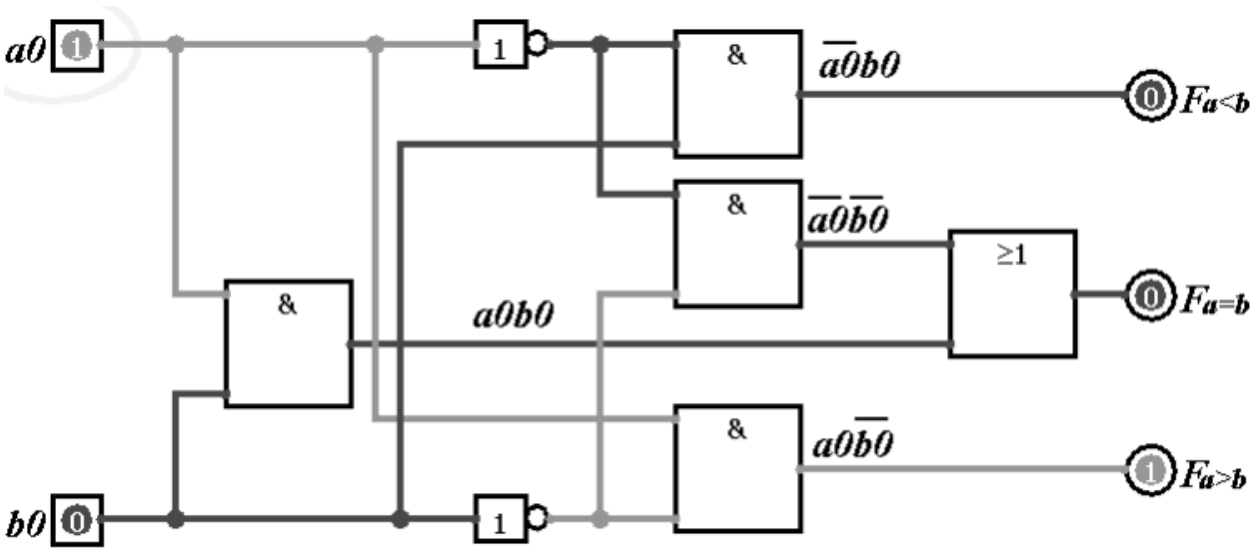



Рис. 40. Схема однорозрядного компаратора

Зупинимося докладніше на рівності чисел. Зазначимо, що функція $F_{a=b}$ – функція рівнозначність. За змістом вона протилежна функції нерівнозначність (виключаюче АБО):

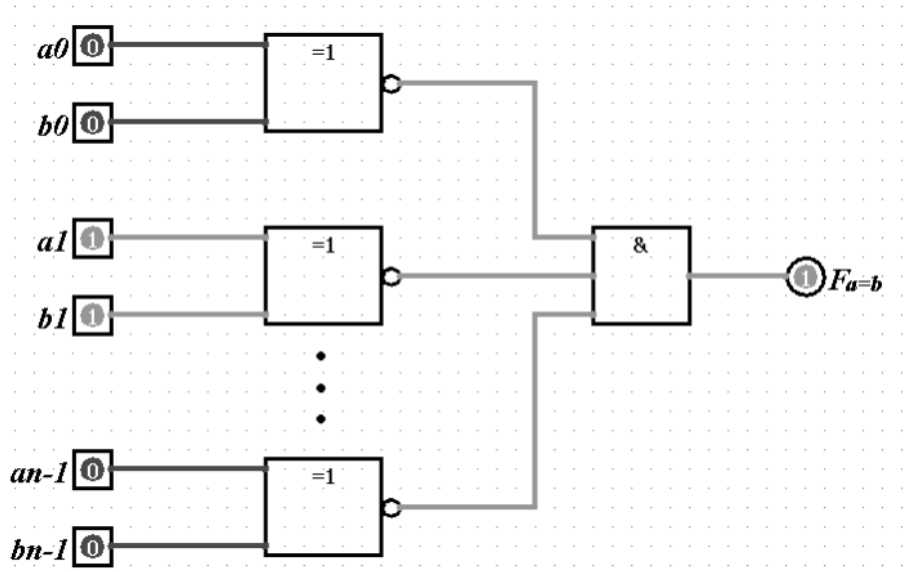
$$F_{a\neq b} = a_0\overline{b}_0 + \overline{a}_0 b_0 = a_0 \otimes b_0,$$

тобто

$$F_{a=b} = \overline{F}_{a\neq b} = \overline{a_0\overline{b}_0 + \overline{a}_0 b_0} = \overline{a_0 \otimes b_0}$$

Тому перевірку рівності пари однойменних розрядів двох чисел можна здійснити, використовуючи елемент рівнозначність («Элемент исключающее ИЛИ» бібліотеки «Элементы») (рис. 41, *а*) чи елемент нерівнозначність («Элемент исключающее ИЛИ-НЕ» бібліотеки «Элементы»), доповнений інвертором (рис. 41,*б*).

*а)*




*b)*

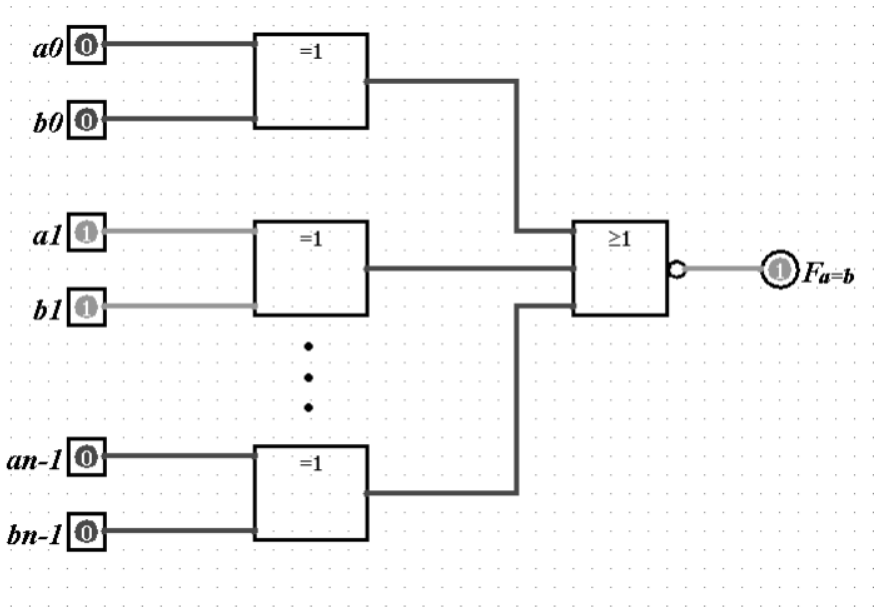



Рис. 41. Схема цифрового компаратора для функції $F_{a=b}$, на порівняння двох трирозрядних чисел

Два числа $a$ і $b$ рівні, якщо їхні однойменні розряди містять однакові значення ($a_0$=$b_0$ і $a_1$=$b_1$ і … $a_{n-1}$=$b_{n-1}$), тобто функція, яка характеризує співвідношення чисел, повинна бути кон'юнкцією функцій, які характеризують співвідношення цифр в їхніх однойменних розрядах:

$$F_{a=b} = F_{a_0=b_0} \cdot F_{a_1=b_1} \cdot \ldots \cdot F_{a_{n-1}=b_{n-1}}$$

Коли цифри в однойменних розрядах чисел $a$ і $b$ однакові, то на виходах всіх елементів рівнозначність (рис. 41) будуть логічні одиниці і $F_{a=b}$=1. Якщо хоча б в одній парі розрядів знаходяться різні цифри, то на виході відповідного елемента рівнозначність буде логічний 0 і функція $F_{a=b}$=0, що вказує на нерівність чисел $a$ і $b$.

Розглянемо випадки нерівності чисел. Виявлення більшого з двох багаторозрядних чисел $a$ і $b$ починається із старших розрядів: якщо вони рівні, то порівнюється наступна пара однойменних розрядів і т.д.

Так, у випадку трирозрядних чисел $a > b$ , якщо мають місце :

- нерівність старших розрядів ($a_2 > b_2$, $F_{a_2>b_2} = a_2\overline{b}_2 = 1$);
- при рівності старших розрядів ($a_2 = b_2$, $F_{a_2=b_2} = a_2b_2 + \overline{a}_2\overline{b}_2 = 1$); існує нерівність розрядів $a_1$, $b_2$ ($a_2 > b_2$, $F_{a_1>b_1} = a_1\overline{b}_1 = 1$);
- при рівності ($a_2 = b_2$, $F_{a_2=b_2} = a_2b_2 + \overline{a}_2\overline{b}_2 = 1$) і ($a_1 = b_1$, $F_{a_1=b_1} = a_1b_1 + \overline{a}_1\overline{b}_1 = 1$); існує нерівність розрядів $a_0$, $b_0$ ($a_0 > b_0$, $F_{a_0>b_0} = a_0\overline{b}_0 = 1$);

Позначивши для зручності:

$$F_{a_2=b_2} = a_2b_2 + \overline{a}_2\overline{b}_2 = F_2;$$

$$F_{a_1=b_1} = a_1b_1 + \overline{a}_1\overline{b}_1 = F_1 .$$

Використовуючи нові позначення подамо приведені умови у вигляді:

$$F_{a>b} = a_2\overline{b}_2 + a_1\overline{b}_1F_2 + a_0\overline{b}_0F_2F_1.$$

При $a_2 > b_2$ ($a_2 = 1$, $b_2 = 0$) кон'юнкція $a_2\overline{b}_2 = 1$ – функція $F_{a>b} = 1$, що вказує на справедливість нерівності $a > b$.

Коли $a_2 = b_2$ ($F_2 = 1$, $a_2\overline{b}_2 = 0$), але $a_1 > b_1$ ($a_1 = 1$, $b_1 = 0$, $a_1\overline{b}_1 = 1$), то на справедливість нерівності $a > b$ вказує другий член записаної функції: $a_1\overline{b}_1F_2 = 1$.

Якщо $a_2 = b_2$ ($F_2 = 1$, $a_2\overline{b}_2 = 0$) і $a_1 = b_1$ ($F_1 = 1$, $a_1\overline{b}_1 = 0$), але $a_0 > b_0$ ($a_0 = 1$, $b_0 = 0$, $a_0\overline{b}_0 = 1$), то $a_0\overline{b}_0F_2F_1 = 1$ і $F_{a>b} = 1$.

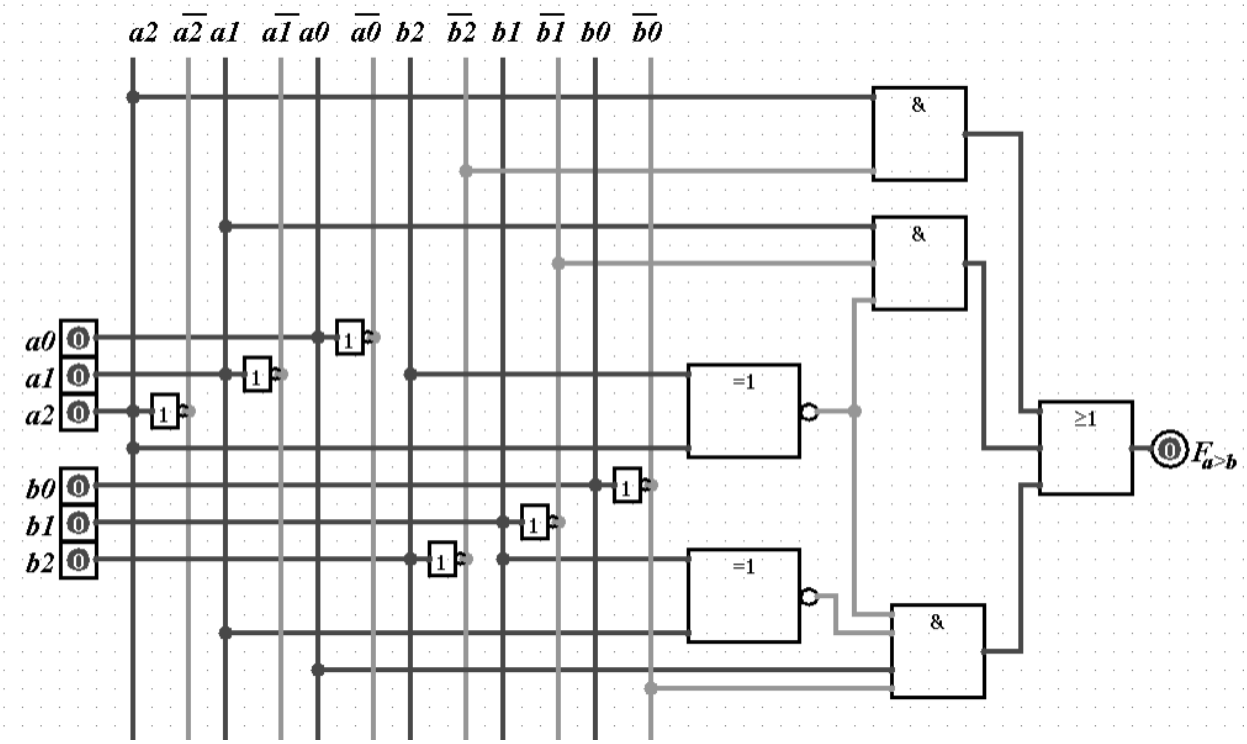



Рис. 42. Схема цифрового компаратора для функції $F_{a>b}$, на порівняння двох трирозрядних чисел

Функція $F_{a>b}$ реалізується схемою рис. 42. На рис. 43 вона доповнена елементом рівнозначність, на входи якого подаються розряди $a_0$, $b_0$, кон'юнктором, на виході якого формується функція $F_{a=b}$. Коли $a_2 = b_2$ ($F_2 = 1$), $a_1 = b_1$ ($F_1 = 1$) і $a_0 = b_0$ ($F_0 = 1$), то $F_{a=b} = F_2 \cdot F_1 \cdot F_0 = 1$, тобто $a = b$. Дана схема також доповнена елементом АБО-НЕ, на виході якого формується функція $F_{a<b}$. Якщо в результаті порівняння чисел $F_{a>b}$=0 і $F_{a=b}$=0, то на виході АБО-НЕ буде логічна 1 ($F_{a<b} = 1$), тобто $a < b$.

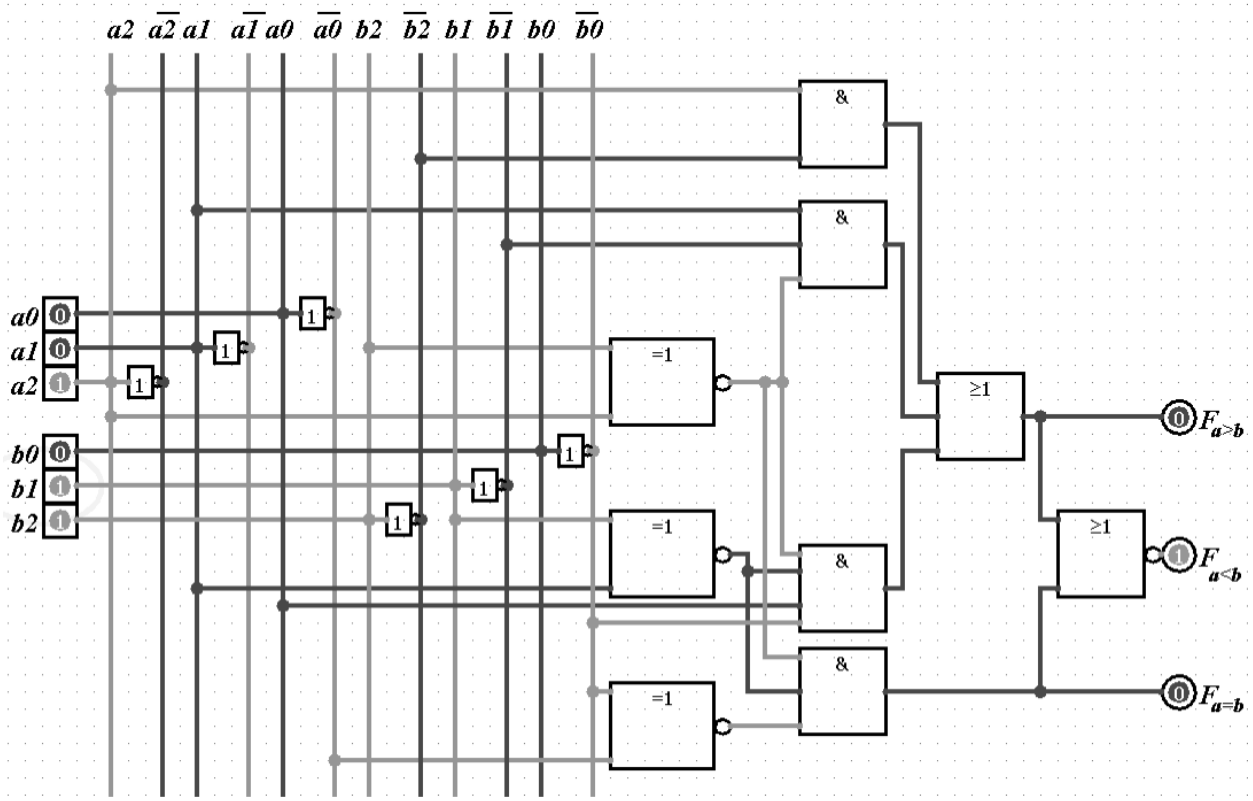



Рис.43. Схема цифрового компаратора на порівняння двох трирозрядних чисел

Аналогічно (рис. 43) будуються компаратори для порівняння чисел з більшою розрядністю.

## Завдання

**1.** Скласти схему цифрового компаратора для порівняння на рівність двох:

*а)* дворозрядних кодів;

*б)* трирозрядних кодів;

*в)* чотирирозрядних кодів;

**2.** Скласти схему цифрового компаратора, що виконує дію «більше»:

*а)* дворозрядних кодів;

*б)* трирозрядних кодів;

*в)* чотирирозрядних кодів;

**3.** Скласти схему цифрового компаратора, що виконує дію «менше»:

*а)* дворозрядних кодів;

*б)* трирозрядних кодів;

*в)* чотирирозрядних кодів;

**4.** Скласти схему цифрового компаратора для порівняння двох трирозрядних чисел. На основі побудованого компаратора знайти:

*а)* найбільше з трьох чисел;

*б)* найменше з трьох чисел;

*в)* найбільше з чотирьох чисел;

*г)* найменше з чотирьох чисел;

*д)* рівних два числа з трьох чисел;

*е)* рівних два числа з чотирьох чисел.

**5.** Скласти схему цифрового компаратора для порівняння двох дворозрядних чисел. На основі побудованого компаратора знайти:

*а)* найбільше з чотирьох чисел;

*б)* найменше з чотирьох чисел;

*в)* найбільше з п’яти чисел;

*г)* найменше з п’яти чисел;

*д)* рівних три числа з чотирьох чисел;

*е)* рівних три числа з п’яти чисел.

## Питання для самоконтролю

**1.** Для чого призначений компаратор?

**2.** Які функції виконує цифровий компаратор, в яких пристроях він може бути використаний?

**3.** Як функціонує елемент рівнозначність.

**4.** Як функціонує елемент нерівнозначність.

# 7. ПРОЕКТУВАННЯ СУМАТОРІВ.

Арифметичні дії над двійковими числами виконуються за тими ж правилами, що і над десятковими. Дії над числами виконуються порозрядно, починаючи з молодшого розряду. Необхідно тільки враховувати, що додавання двох одиниць дає нуль в даному розряді і одиницю переносу до сусіднього старшого розряду, а при відніманні, якщо цифра від'ємника більша за цифру відповідного розряду зменшуваного, то необхідно позичити одиницю у сусіднього старшого розряду.

Основною дією над двійковими числами є додавання. Воно використовується як саме по собі, так і в операціях віднімання, а також лежить в основі множення і ділення чисел. *Суматор* – комбінаційний логічний пристрій призначений для арифметичного додавання двох двійкових чисел. На схемах суматори позначаються літерами SM. За числом виходів розрізняють напівсуматори, одно- та багаторозрядні суматори.

## 7.1. Проектування однорозрядних суматорів

*Напівсуматор* – пристрій, призначений для додавання двох однорозрядних кодів, який має два входи, два виходи та який формує із вхідних сигналів сигнал суми та сигнал перенесення у старший розряд. В табл. 17. Подано таблицю істинності напівсуматора.

Таблиця 17

**Таблиця істинності напівсуматора**

| Вхід | | Вихід | |
|---|---|---|---|
| Доданки | | Сума | Перенос |
| $x$ | $y$ | $s$ | $p$ |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 |

Склавши діаграми Вейча (рис. 44), отримавши операторні подання функцій $s$ і $p$, можна побудувати схему напівсуматора (рис. 45).

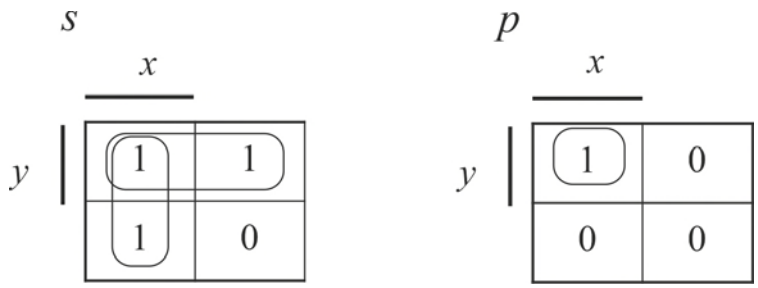



Рис. 44. Діаграми Вейча для функцій $s$ і $p$ напівсуматора

$s = x \vee y;$

$p = x \cdot y.$

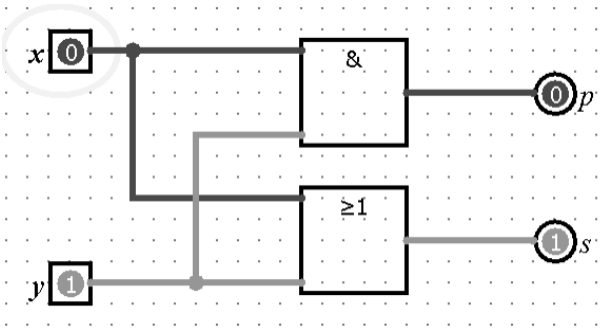



Рис. 45. Схема напівсуматора

*Однорозрядний суматор* – пристрій, призначений для додавання двох однорозрядних кодів, що має три входи, два виходи та формує із сигналів вхідних доданків і сигналу перенесення з молодших розрядів сигнали суми та сигнали перенесення у старший розряд.

В табл. 18 подано таблицю істинності однорозрядного суматора.

Оскільки цей суматор буде виконувати додавання чисел в двійковій системі числення, то коефіцієнт $k = 2$. Виходячи з таблиці, складемо діаграму Вейча для функцій $s_i$ і $p_i$ (рис. 46).

Виконавши відповідні склеювання, одержимо мінімальні диз'юнктивні нормальні форми (МДНФ) функцій $s_i$ і $p_i$.

Таблиця 18

**Таблиця істинності однорозрядного повного суматора**

| Вхід | | | Вихід | |
|---|---|---|---|---|
| Доданки | | Перенос | Сума | Перенос |
| $x_i$ | $y_i$ | $z_i$ | $s_i$ | $p_i$ |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

$s_i = x_i\overline{y}_i\overline{z}_i \vee \overline{x}_i\overline{y}_i z_i \vee \overline{x}_i y_i\overline{z}_i \vee x_i y_i z_i;$

$p_i = x_i y_i \vee x_i z_i \vee y_i z_i$ .

Перепишемо $s_i$ у вигляді:

$s_i = x_i y_i z_i \vee x_i\overline{p}_i \vee y_i\overline{p}_i \vee z_i\overline{p}_i.$

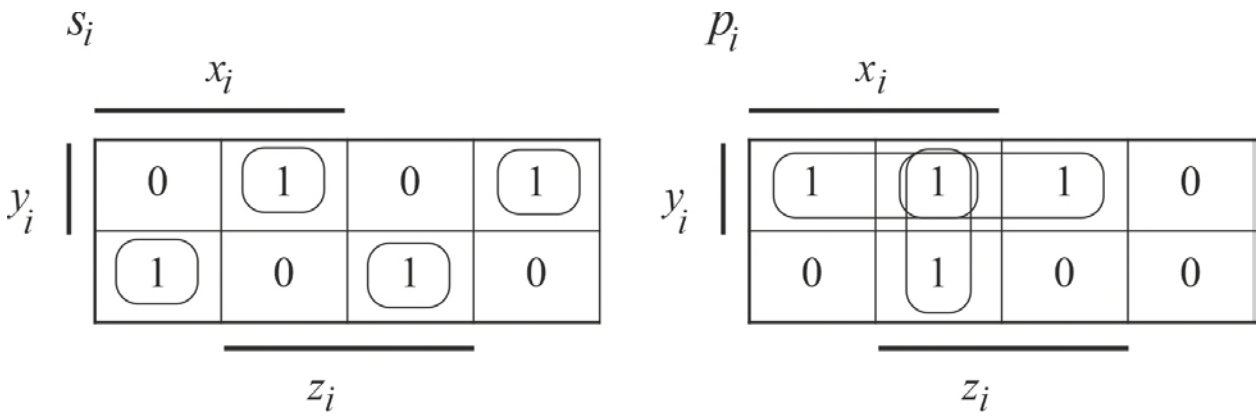



Рис. 46. Діаграми Вейча для функцій $s_i$ і $p_i$

На рис. 47 подано схему однорозрядного суматора.

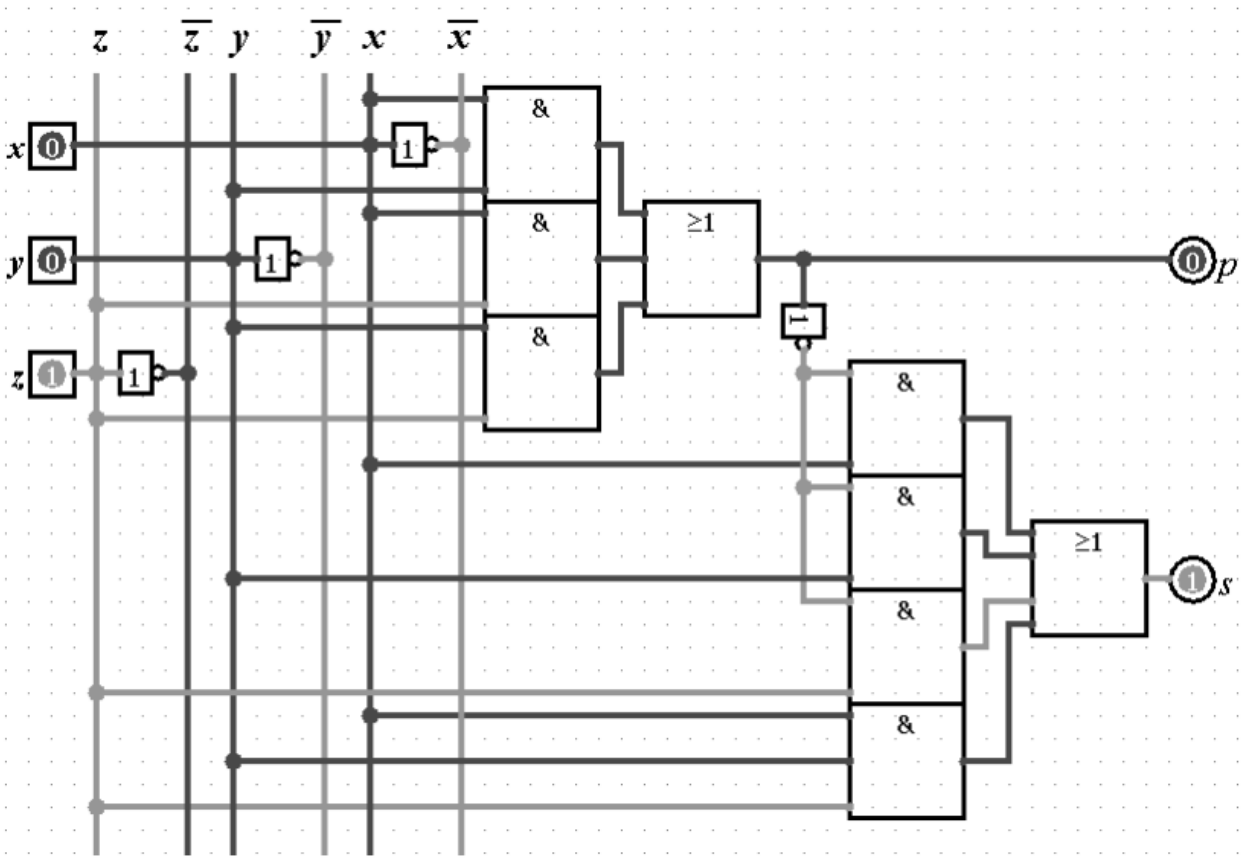



Рис. 47. Схема однорозрядного суматора.

### 7.2. Багаторозрядні суматори з послідовним переносом

Багаторозрядні суматори поділяють на послідовні, паралельні та паралельно-послідовні. Послідовний суматор виконує додавання чисел порозрядно, починаючи з молодшого розряду. В паралельних суматорах усі розряди вхідних кодів підраховуються одночасно.

При побудові багаторозрядного суматора можна подавати сигнал переносу з виходу кожного повного однорозрядного суматора на один з входів суматора наступного розряду. В результаті отримаємо багаторозрядний суматор з послідовним переносом. Схема трирозрядного двійкового суматора з послідовним переносом.  показаний на рис. 48.

Такий багаторозрядний суматор простий в реалізації, але він дуже повільний: кожен однорозрядний суматор повинен дочекатися сигналу перенесення з попереднього розряду, тобто суматори з’єднані послідовно, а отже, затримки елементів однорозрядних суматорів підсумовуються. В

результаті загальна затримка багаторозрядного суматора виходить величезною.

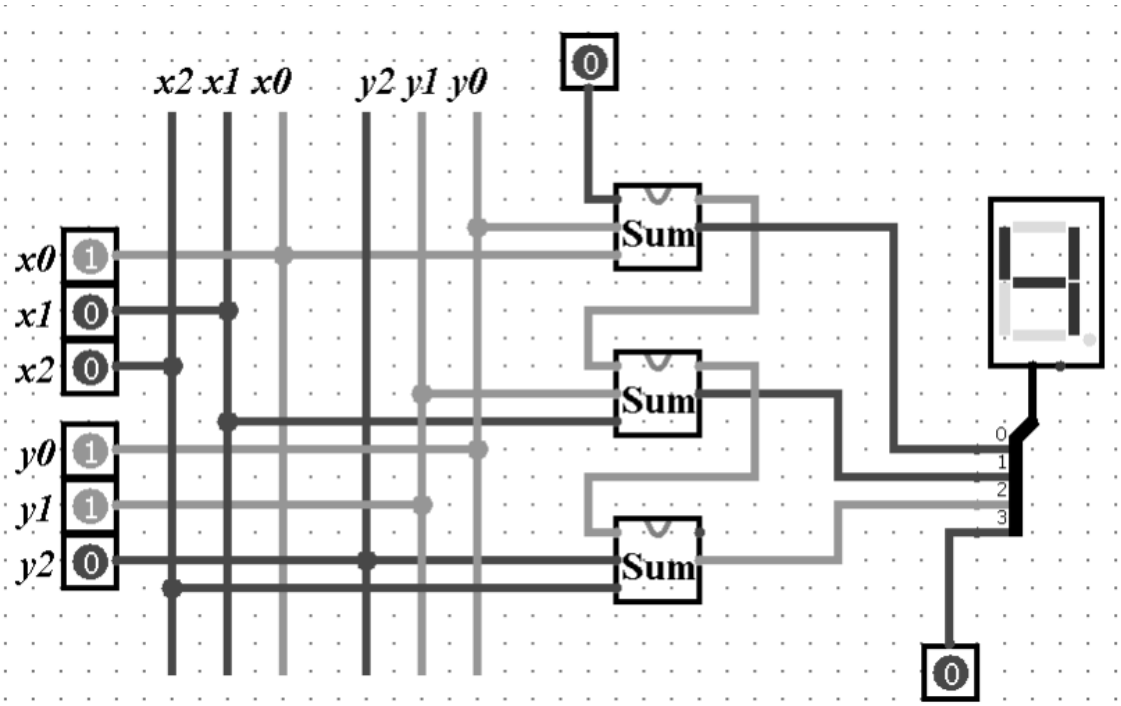



Рис. 48. Схема трирозрядного суматора з послідовним переносом.

### 7.3. Багаторозрядні суматори з паралельним переносом

Виключити затримку поширення переносу дозволяє суматор з паралельним переносом. Ідея полягає в тому, щоб сигнали перенесення для всіх розрядів формувалися чисто логічно на основі доданків $x$ і $y$. Розглянемо, як це можна зробити.

Сигнал перенесення в 1-й розряд $p_1 = x_0 \cdot y_0$. Сигнал перенесення в другий розряд можна визначити за наступною формулою:

$$p_2 = x_1 \cdot y_1 \vee p_1 \cdot x_1 \vee p_1 \cdot y_1 = x_1 \cdot y_1 \vee p_1 \cdot (x_1 \vee y_1) = x_1 \cdot y_1 \vee x_0 \cdot y_0 \cdot (x_1 \vee y_1)$$

Як видно, сигнал перенесення $p_2$ може бути отриманий за значеннями попередніх розрядів чисел $x$ і $y$ без сигналу переносу $p_1$, формованого сумматором. Аналогічно, можна отримати логічний вираз для сигналу переносу $p_2$ третього розряду, в якому також будуть значення тільки попередніх розрядів чисел $x$ і $y$.

$p_3 = x_2 \cdot y_2 \vee p_2 \cdot x_2 \vee p_2 \cdot y_2 = x_2 \cdot y_2 \vee p_2 \cdot (x_2 \vee y_2) = x_2 \cdot y_2 \vee x_1 \cdot y_1 \vee x_0 \cdot y_0 \cdot (x_1 \vee y_1) \cdot (x_2 \vee y_2)$.

Таким чином, сигнали переносу для будь-якого розряду можуть бути сформовані чисто логічним шляхом за значеннями доданків. Значить, немає необхідності очікувати, поки буде сформовано сигнали переносу однорозрядними суматорами. Але чим більша розрядність чисел, тим вища складність булевих функцій сигналів переносу, особливо для самого старшого розряду. Тому суматори з паралельним переносом роблять для чисел невеликої розрядності.

Схема суматора для сумування двох трирозрядних чисел з паралельним переносом наведена на рис. 49.

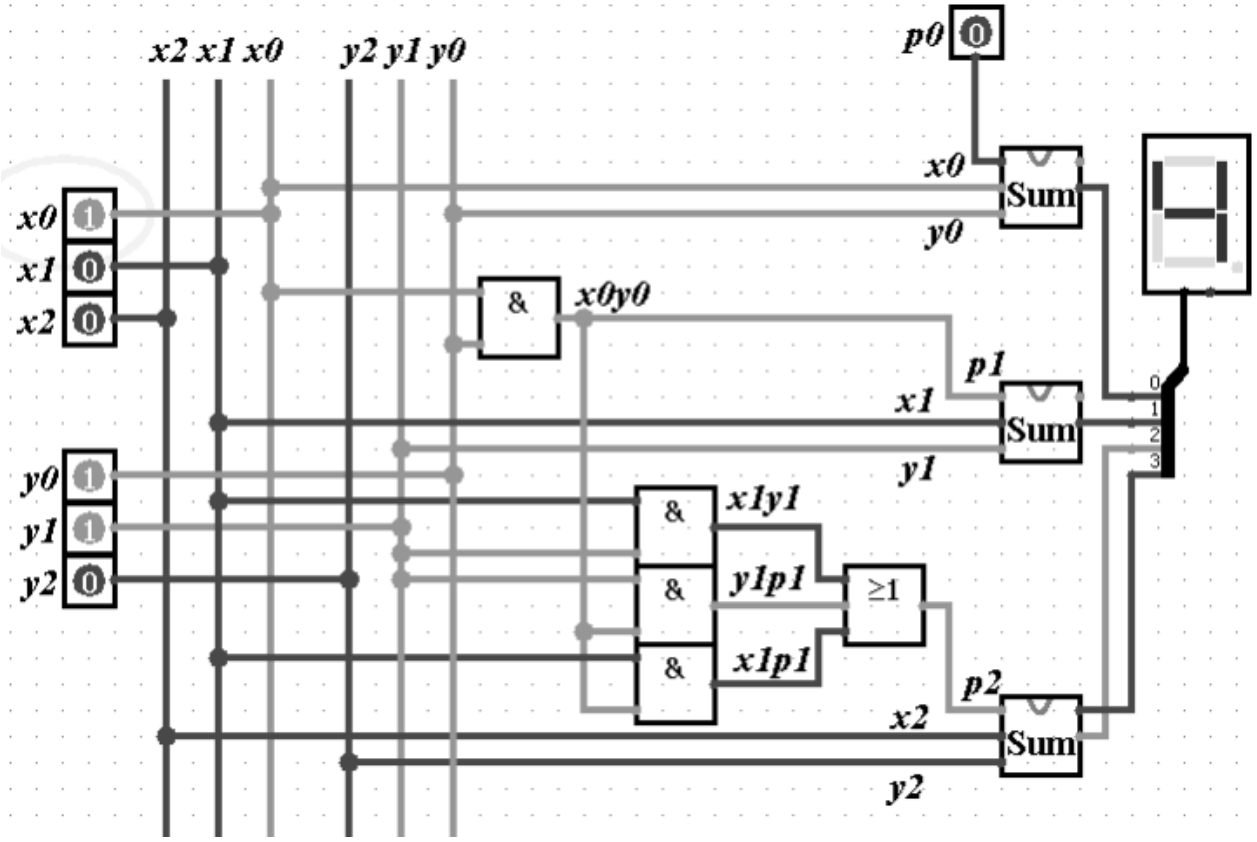



Рис. 49. Схема трирозрядного суматора з паралельним переносом.

### 7.4. Багаторозрядні суматори з груповим переносом

Для чисел великої розрядності застосовують суматори з груповим переносом. Схема такого суматора розбивається на *l* груп розрядності *m:*

наприклад, чотири групи по вісім розрядів для додавання 32-розрядних чисел. Кожна група представляє собою паралельний суматор. На вхід переносу надходить сигнал переносу від старшого розряду попередньої групи до молодшого розряду наступної групи. Цей сигнал формується блоком переносу, що аналізує $m$ розрядів доданків $x$ і $y$ «своєї» групи, «не чекаючи», коли в ній відбудеться додавання всіх розрядів. Блоки переносу (БП) різних груп включені послідовно.

Структура суматора з груповим переносом аналогічна структурі суматора з послідовним переносом, де замість однорозрядних суматорів включаються групові. Така структура отримала назву групового суматора з ланцюговим переносом (рис. 50).

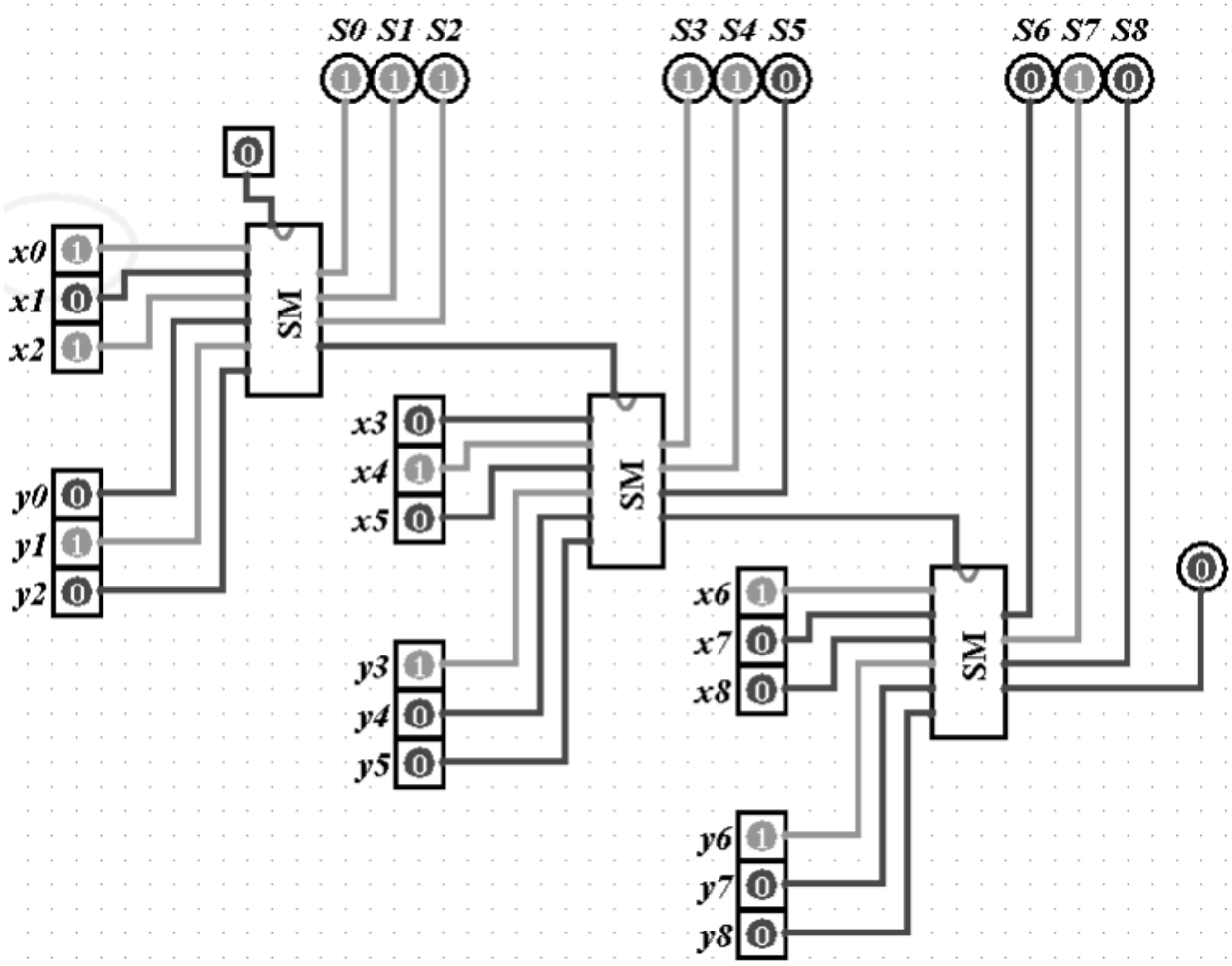



Рис. 50. Груповий суматор з ланцюговим переносом

Крім структури групового суматора з ланцюговим переносом можлива структура з паралельними міжгруповими переносами. Структура цього суматора аналогічна структурі суматора з паралельним переносом, в якому роль однорозрядних суматорів відіграють групи. Апаратна складність суматорів з паралельними міжгруповими переносами вища, ніж складність попереднього варіанту, але при великих розрядах вони дають переваги по швидкодії.

### 7.5. Проектування відніматора

Для того, щоб не ускладнювати конструкцію АЛП (арифметично логічного пристрою), операцію віднімання замінюють додаванням (що виконується суматором) зменшуваного з від'ємником, представленим в спеціальному коді. Розглянемо це більш детально.

Віднімання двійкових чисел, записаних в прямому коді, подібне до віднімання в десятковій системі (рис. 51):

| | | | |
|---|---|---|---|
| | *A* | 10111 | 53 |
| − | *B* | − 1110 | − 14 |
| | *F* | 1001 | 39 |

Рис. 51. Демонстрація віднімання двох чисел

Стрілками показана операція «позичання», яка виконується для тих розрядів, в яких від'ємник більший за зменшуване. В десятковій системі позичається одиниця старшого розряду, яка дорівнює десяти одиницям сусіднього молодшого розряду, а в двійковій – двом одиницям молодшого розряду.

Для заміни операції віднімання операцією додавання потрібно подати від'ємник *B* в додатковому коді. Додатковий код утворюється з оберненого (інверсного) коду, додаванням до нього одиниці. Так, чотирьохрозрядний від'ємник *B*, представлений в прямому коді $B_{пр}$=*B*3 *B*2 *B*1 *B*0 , може бути

поданий і в оберненому коді $B_{обер}=\overline{B3}\ \overline{B2}\ \overline{B1}\,\overline{B0}$, і в додатковому коді $B_{дод}=B_{обер}+1$. Очевидно, для чотирирозрядних чисел, записаних в цих кодах, справедливі рівності:

$B_{пр}+B_{обер}=1111$;

$B_{пр}+B_{дод}=B_{пр}+B_{обер}+1=1111+1=10000$;

$B_{пр}=10000-B_{дод}=10000-B_{обер}-1$.

А тому, операцію віднімання можна подати у вигляді:

$A_{пр}-B_{пр}=A_{пр}+B_{дод}-10000$.

Таким чином, в АЛП при виконанні операції віднімання, вхідний операнд $B$ перетворюється в додатковий код, а віднімання числа 10000 виконується без допомоги спеціальних схем, тільки з використанням сигналу переносу в старший (п’ятий) розряд. Але, при цьому результат арифметичних дій на виході АЛП буде також поданий в оберненому коді.

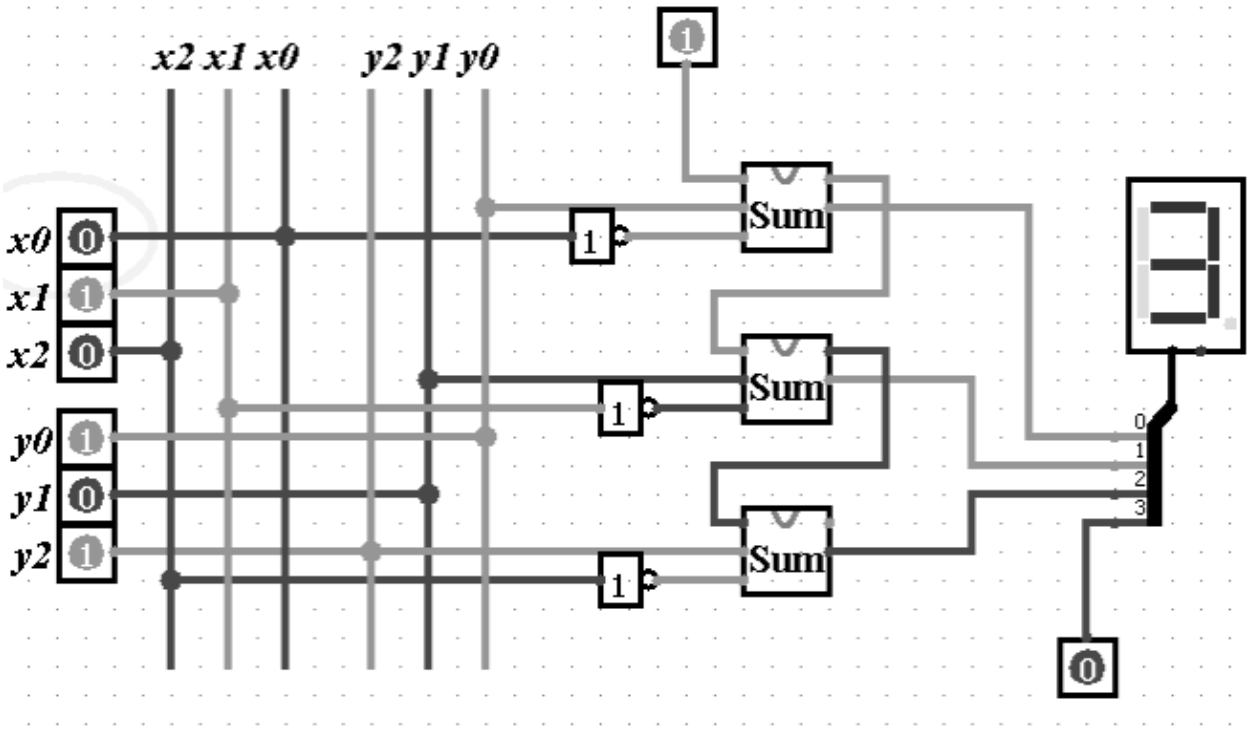



Рис. 52. Схема віднімання двох трирозрядних двійкових чисел за допомогою суматора з послідовним переносом

### 7.6. Проектування помножувача двійкових чисел

При побудові пристрою для множення чисел потрібно пам’ятати, що розрядність результату в два рази більша, ніж розрядність множників. Один з найбільш простих та швидких методів множення заснований на використанні комбінаційної логічної схеми. Нехай необхідно знайти добуток двох трирозрядних двійкових чисел $x_2x_1x_0$ та $y_2y_1y_0$. Схема виконання операції множення відповідає звичайному «множенню стовпчиком». Оскільки в операції використовують двійкові числа, то арифметичне множення замінює кон’юнкція. Для трирозрядних чисел алгоритм множення має вигляд, зображений на рис. 51.

$$
\begin{array}{cccccc}
 & & & x2 & x1 & x0 \\
 & & \times & y2 & y1 & y0 \\
\hline
 & & & x2y0 & x1y0 & x0y0 \\
+ & & x2y1 & x1y1 & x0y1 & \\
 & x2y2 & x1y2 & x0y2 & & \\
\hline
P5 & P4 & P3 & P2 & P1 & P0
\end{array}
$$

Рис. 53. Алгоритм множення двох трирозрядних двійкових чисел.

Всі кон’юнкції виконуються паралельно схемами І, а для операцій додавання використовуються однорозрядні суматори. Також потрібно врахувати те, що при отриманні функції $P_3$ ми використовуємо два суматори для додавання трьох чисел і перенос з попереднього суматора при отриманні функції $P_2$. Для отримання функції $P_4$ теж потрібно використати два суматори, оскільки ми додаємо два числа і два переноси з двох суматорів, які використовували для отримання функції $P_3$. Згідно наведеного алгоритму, можна побудувати схему помножувача для множення двох трирозрядних чисел. (рис. 54).

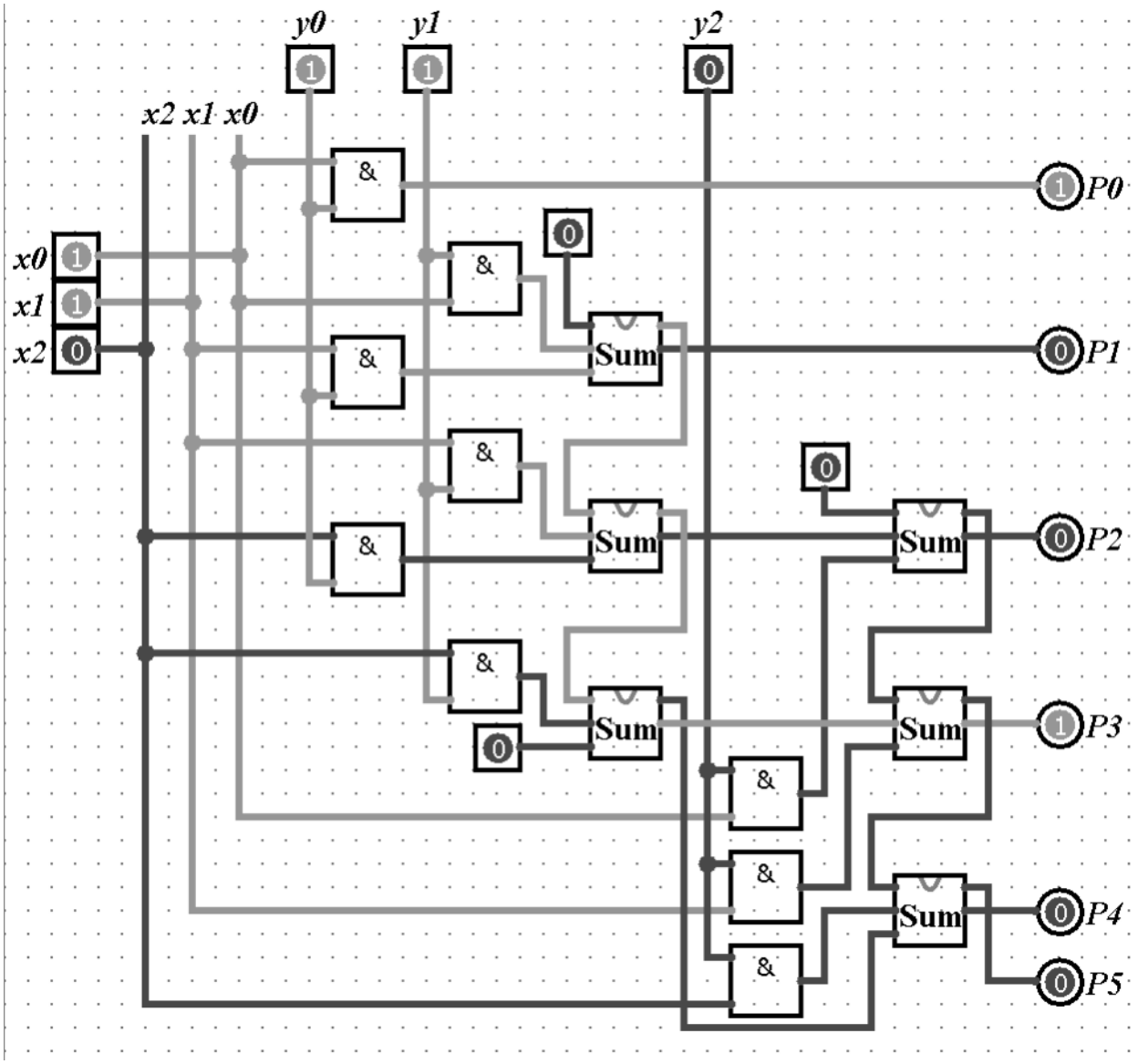



Рис. 54. Схема для множення двох трирозрядних чисел

**Завдання**

**1.** Побудувати однорозрядний напівсуматор, використовуючи логічні елементи:

*a)* І, НЕ;

*b)* АБО, НЕ.

**2.** Побудувати однорозрядний повний суматор, використовуючи логічні елементи:

*a)* І, НЕ;

*b)* АБО, НЕ.

**3.** На основі однорозрядного повного суматора побудувати схему з послідовним переносом для демонстрації додавання двох двійкових чисел:

*a)* трирозрядних;

*b)* чотирирозрядних;

*c)* п’ятирозрядних;

*d)* шестирозрядних.

**3.** На основі однорозрядного повного суматора побудувати схему з послідовним переносом для демонстрації віднімання двох двійкових чисел:

*a)* трирозрядних;

*b)* чотирирозрядних;

*c)* п’ятирозрядних;

*d)* шестирозрядних.

**4.** На основі побудованого багаторозрядного суматора з послідовним переносом побудувати схему для демонстрації обчислення виразу:

*a)* $A+B+C$;

*b)* $A-B+C$;

*c)* $A+B-C$;

*d)* $A-B-C$;

*e)* $A+B+C+D$;

*f)* $A-B+C+D$;

*g)* $A+B-C+D$;

*h)* $A-B-C+D$;

*i)* $A+B+C-D$;

*j)* $A-B+C-D$;

*k)* $A+B-C-D$;

*l)* $A-B-C-D$.

**5.** На основі однорозрядного повного суматора побудувати схему з паралельним переносом для демонстрації додавання двох двійкових чисел:

*a)* дворозрядних;

*b)* трирозрядних;

*c)* чотирирозрядних;

*d)* п’ятирозрядних;

*e)* шестирозрядних.

**6**. На основі однорозрядного повного суматора побудувати схему з паралельним переносом для демонстрації віднімання двох двійкових чисел:

*a)* дворозрядних;

*b)* трирозрядних;

*c)* чотирирозрядних;

*d)* п’ятирозрядних;

*e)* шестирозрядних.

**7.** На основі однорозрядного повного суматора побудувати схему помножувача, для множення двох двійкових чисел:

*a)* дворозрядних;

*b)* трирозрядних;

*c)* чотирирозрядних;

## Питання для самоконтролю

1. Для чого призначений суматор?
2. Що таке напівсуматор?
3. Скільки входів та виходів має напівсуматор суматор?
4. Що таке повний однорозрядний суматор?
5. Скільки входів та виходів має повний однорозрядний суматор?
6. Як утворюється сигнал переносу?
7. Яке призначення сигналу переносу?
8. Які Ви знаєте види суматорів.
9. Як утворюється додатковий код?
10. Як множення двох двійкових чисел можна замінити додаванням?
11. Записати алгоритм множення двох дворозрядних чисел.

# 8. АРИФМЕТИКО-ЛОГІЧНІ ПРИСТРОЇ

Пристрої для арифметичних операцій досить рідко використовуються в схемах безпосередньо. Як правило, вони включаються до складу арифметико-логічного пристрою (АЛП) (англ. *arithmetic and logic unit*, ALU).

АЛП – блок процесора, який під керування пристрою керування використовується для виконання арифметичних і логічних перетворень над даними, що подаються у вигляді багаторозрядних значень (машинних слів), які називаються операндами. У більшості випадків АЛП – комбінаційний пристрій.

## 8.1. Арифметико-логічна операція зсуву

Розглянемо, як працює операція зсуву. На рис. 55 подана схема зсуву, що містить 8 входів і 8 виходів. Вісім вхідних бітів подаються на лінії *D*7, *D*6, *D*5, *D*4, *D*3, *D*2, *D*1 *D*0. Вихідні дані, які представляють собою вхідні дані, зміщені на 1 біт, надходять на лінії *S*7, *S*6, *S*5, *S*4, *S*3, *S*2, *S*1 *S*0. Лінія управління *C* визначає напрямок зсуву: 0 – вліво, 1 – вправо.

Щоб зрозуміти, як працює така схема, розглянемо пари елементів І (крім крайніх). Якщо *C*=1, правий член кожної пари включається, пропускаючи через себе відповідний біт. Оскільки правий елемент І з'єднаний з входом елемента АБО, який розташований праворуч від цього елемента І, відбувається зсув вправо. Якщо *C*=0, включається лівий елемент І з пари, і тоді відбувається зсув вліво.

В Logisim є компонент зсув («Сдвигатель» бібліотеки «Арифметика»). Цей компонент включає два входи: дані *D* і дистанція *C*, і має один вихід *S*, значення на якому – результат зсуву даних на кількість позицій заданих входом *C* дистанція (рис. 56). І дані, і вихід мають однакову кількість бітів. Компонент підтримує такі типи зсуву: логічний лівий, логічний правий: арифметичний правий, циклічний лівий, циклічний правий.

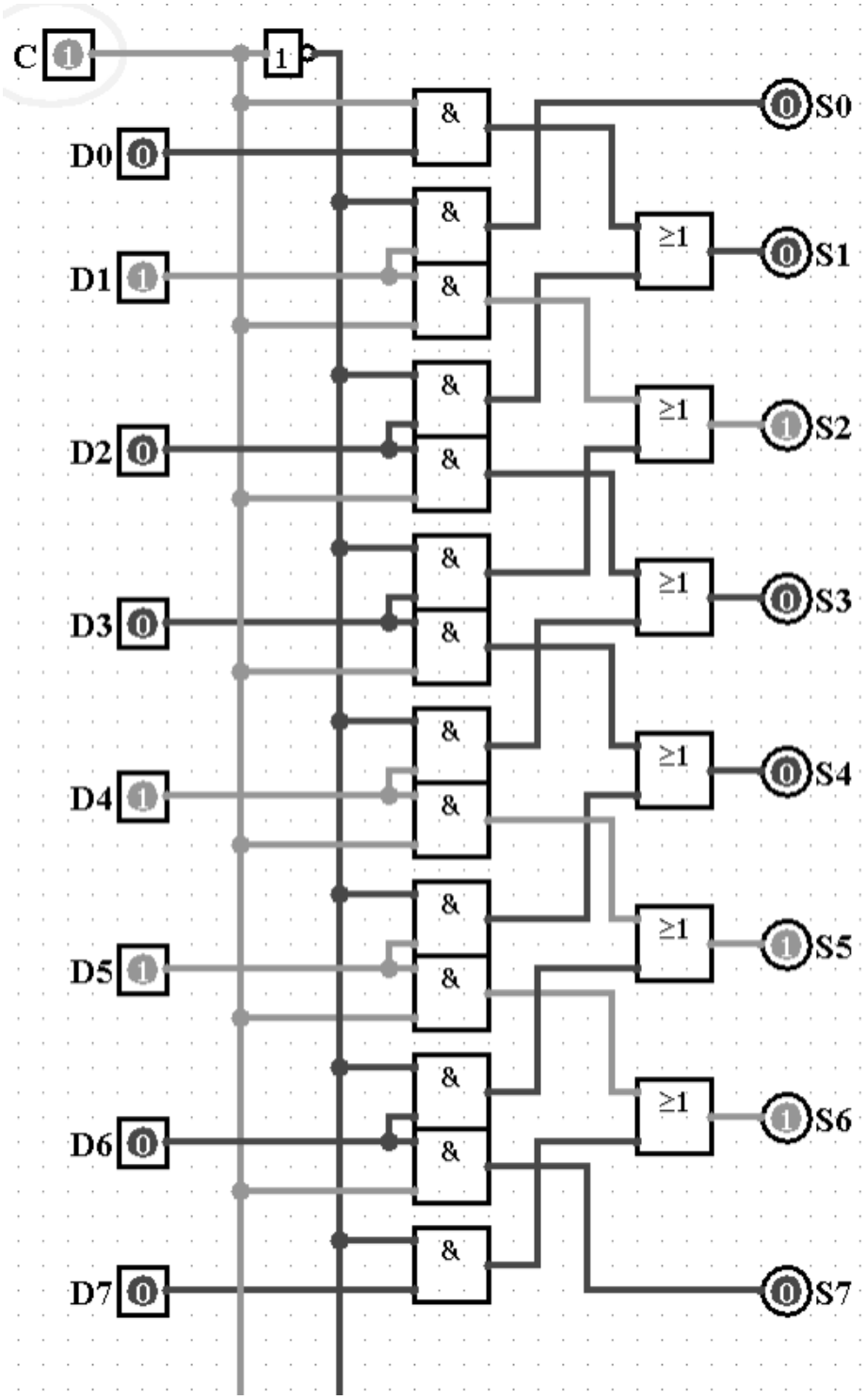



Рис. 55. Схема зсуву

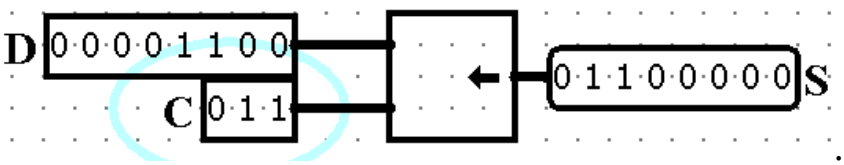



Рис. 56. Робота елемента зсув

## 8.2. Проектування восьмибітного АЛП

Примітивний АЛП має два входи для операндів, один або два (якщо реалізовані множення і/або ділення з остачею) виходів для результату, а також керуючий вхід, на який подається код операції, яку необхідно виконати над операндами. АЛП може приймати і видавати біт переносу, видавати сигнал переповнення і ділення на нуль, і т.д. Типове умовно-графічне позначення АЛП наведено на рис. 57.

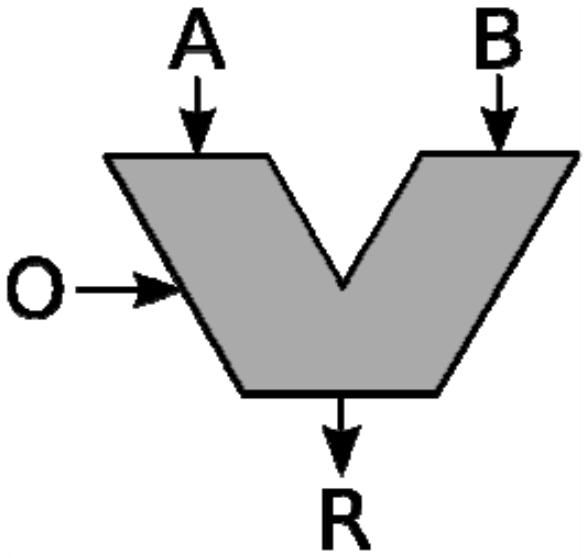



Рис. 57. Умовно-графічне позначення АЛП:
A і B – операнди, R – результат, O – код операції.

На рис 58 подана схема примітивного восьмибітного АЛП. Він може виконувати чотири операції: побітове АБО, побітове І, додавання і логічне ліве зміщення. Двобітний код операції надходить на керуючий вхід мультиплексора, і на вихід надходить результат обраної операції. Ліве зміщення вимагає в якості другого операнда трибітне значення, тому з операнда B попередньо виділено три молодших біта за допомогою компонентів розгалуджувача «Разветвитель».

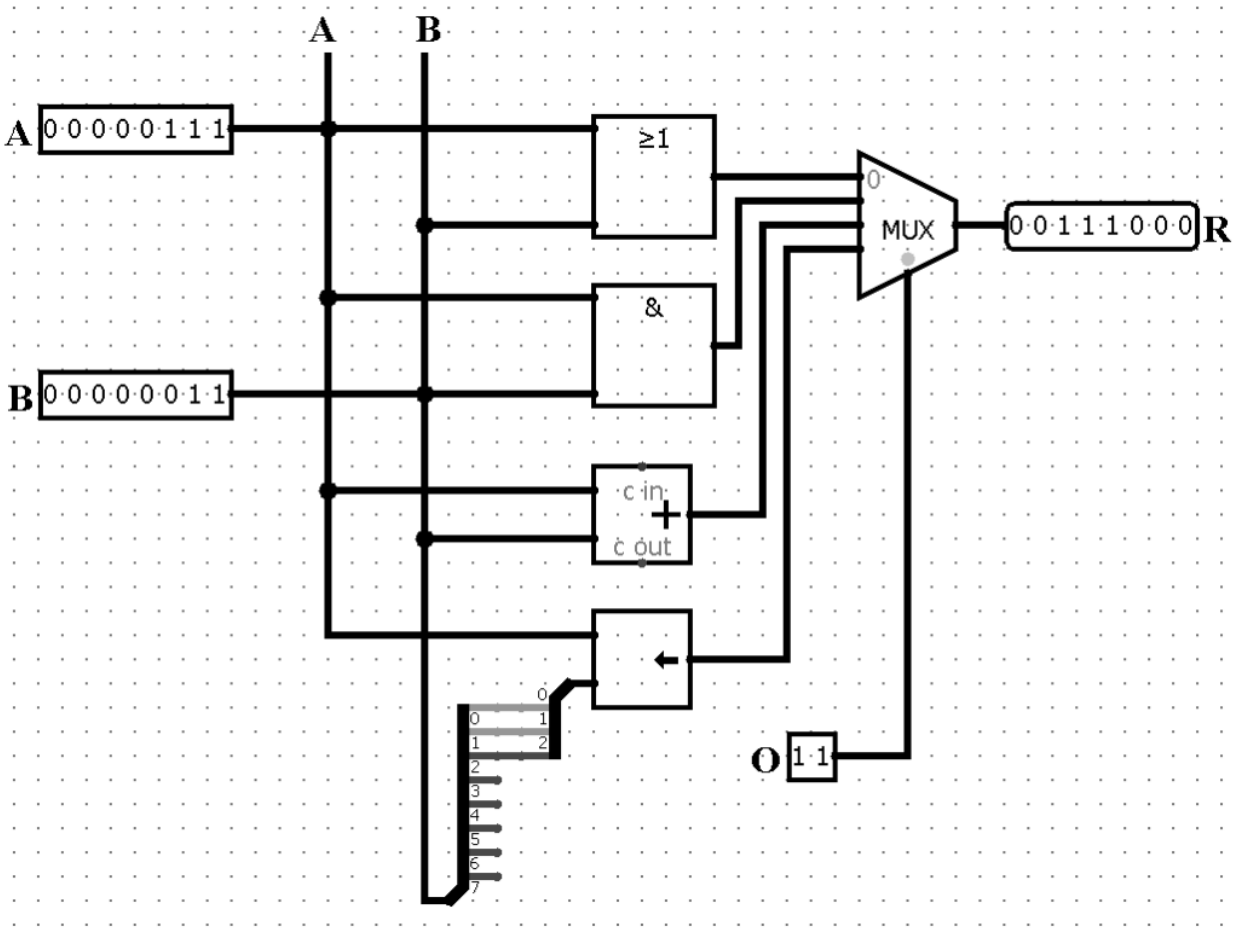



Рис. 58. Схема примітивного восьмибітного АЛП.

Вище були розглянуті арифметичні операції тільки над цілими числами. Звичайно, існують цифрові пристрої для виконання операцій над дійсними числами (числами з плаваючою комою).

### 8.3. Проектування схеми АЛП 74181

В кінці 1960-х років у складі серії мікросхем 7400 у вигляді окремої мікросхеми було випущено 4-бітовий АЛП 74181 (вітчизняний аналог – 155ІП3). Він дозволяє виконувати сім логічних операцій, додавання і віднімання операндів, а також деякі інші комбінації логічних і арифметичних операцій. Множення і ділення в цьому АЛП не реалізовані через їх складність. Об’єднавши кілька таких АЛП, які виконують операції над 4-бітними операндами, можна отримати АЛП більшої розрядності. Біти переносу і позичання можна передавати від одного 4-бітного АЛП до іншого як

безпосередньо (отримуючи суматор і від'ємник з послідовним переносом), так і через додаткову схему прискореного переносу (74181 також генерує біти P і G). Схема АЛП 74181 наведена на рис. 59.

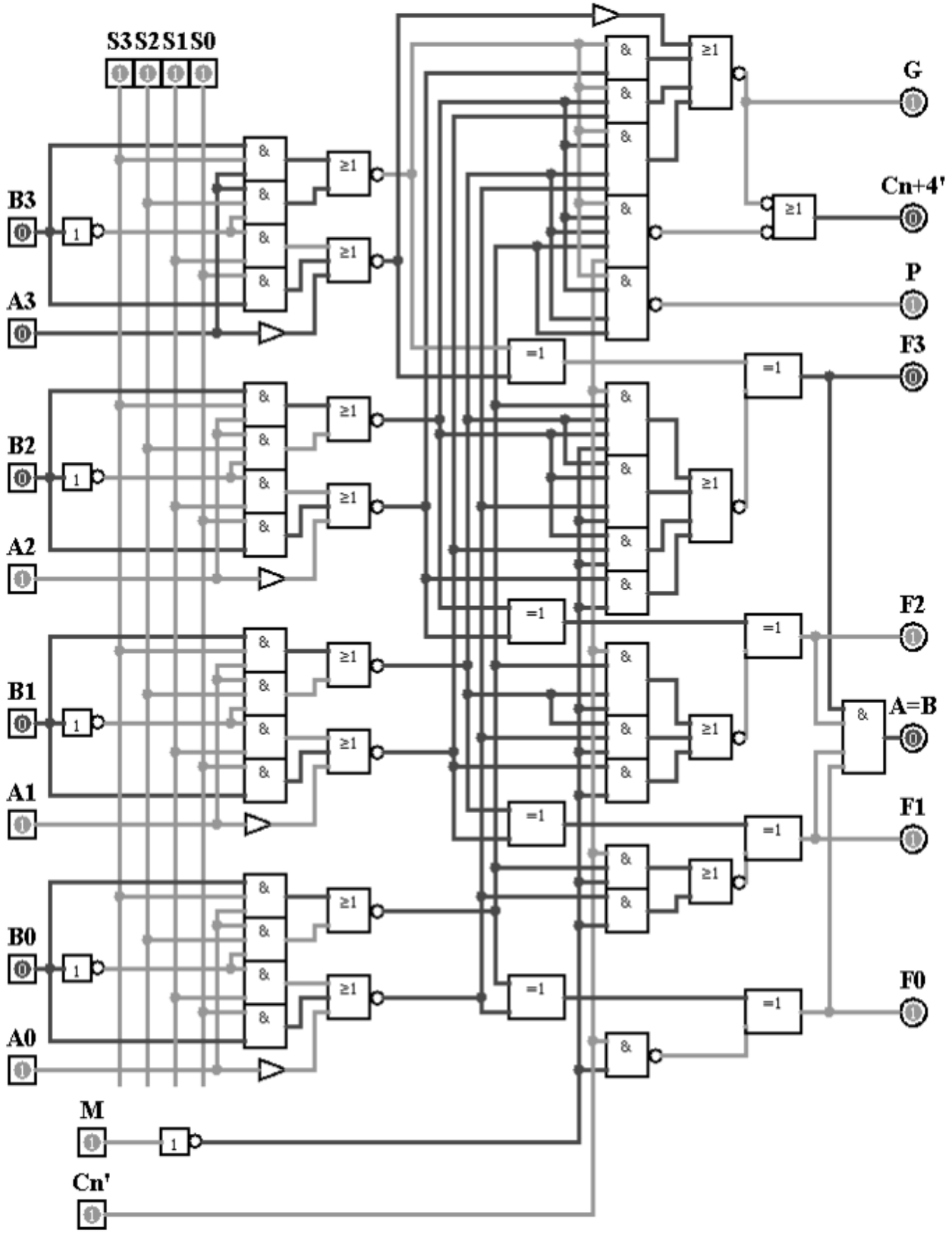



Рис. 59. Схема АЛП 74181

Схема цього АЛП – приклад того, наскільки істотно можна зменшити складність внутрішньої будови при аналізі і проектуванні. Вона містить всього близько семидесяти логічних елементів, а об'єднавши два таких пристрої, можна отримати АЛП, який еквівалентний за функціональністю (і навіть трохи кращий, за винятком операції множення).

На цій схемі входи *A*0, *A*1, *A*2, *A*3 і *B*0, *B*1, *B*2, *B*3 – операнди, *S*0, *S*1, *S*2, *S*3 – код операції, *Cn*' – біт перенесення (або запозичення), який надходить з попереднього розряду, *M* – ознака того, що код операції задає логічні операції (коли на цьому вході 0, виконуються арифметичні). Виходи *F*0, *F*1, *F*2, *F*3 – результат виконання операції, *A*=*B* – ознака рівності операндів при виконанні віднімання, *Cn*+4' – вихід біту переносу (або запозичення) в наступний розряд, *P* і *G* – біти *propagate* і *generate* для схеми прискореного переносу. Вхід *Cn*' і вихід *Cn*+4' – інверсні, тобто наявність переносу (або позичання) позначається логічним нулем.

Відповідність кодів і операцій (логічних і арифметичних) наведено в таблиці 19.

Словами «*plus*» і «*minus*» позначені відповідні арифметичні операції щоб уникнути плутанини з логічним додаванням (тобто з операцією АБО). Символ «<<» означає логічний лівий зсув.

При *M*=0 (тобто при виконанні арифметичних операцій) для всіх операцій «*plus*» і «*minus*» враховується і генерується біт переносу (або запозичення), а також біти *P* і *G*; при *M*=1 виконуються виключно порозрядні операції, тобто біти переносу не враховуються і не генеруються.

Таблиця 19

**Операції АЛП 74181**

| *№* | *S*3 | *S*2 | *S*1 | *S0* | Логічні ($M$=1) | Арифметичні (M=0, $Cn$'=1) |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | $\overline{A}$ | $A$ |
| 2 | 0 | 0 | 0 | 1 | $\overline{A+B}$ | $A+B$ |
| 3 | 0 | 0 | 1 | 0 | $\overline{A}\cdot B$ | $A+\overline{B}$ |
| 4 | 0 | 0 | 1 | 1 | 0 | $minus\ 1$ |
| 5 | 0 | 1 | 0 | 0 | $\overline{A\cdot B}$ | $A\ plus\ A\cdot\overline{B}$ |
| 6 | 0 | 1 | 0 | 1 | $\overline{B}$ | $(A+B)\ plus\ A\cdot\overline{B}$ |
| 7 | 0 | 1 | 1 | 0 | $A\oplus B$ | $A\ minus\ Bminus\ 1$ |
| 8 | 0 | 1 | 1 | 1 | $A\cdot\overline{B}$ | $A\cdot B\ minus\ 1$ |
| 9 | 1 | 0 | 0 | 0 | $\overline{A}+B$ | $A\ plus\ A\cdot B$ |
| 10 | 1 | 0 | 0 | 1 | $\overline{A\oplus B}$ | $A\ plus\ B$ |
| 11 | 1 | 0 | 1 | 0 | $B$ | $\left(A+\overline{B}\right)\ plus\ A\cdot B$ |
| 12 | 1 | 0 | 1 | 1 | $A\cdot B$ | $A\cdot Bminus\ 1$ |
| 13 | 1 | 1 | 0 | 0 | 1 | $A\ plus\ (A\ll 1)$ |
| 14 | 1 | 1 | 0 | 1 | $A+\overline{B}$ | $(A+B)\ plus\ A$ |
| 15 | 1 | 1 | 1 | 0 | $A+B$ | $\left(A+\overline{B}\right)\ plus\ A$ |
| 16 | 1 | 1 | 1 | 1 | $A$ | $A\ minus\ 1$ |

## Завдання

**1.** Побудувати схему, яка реалізує арифметико-логічну операцію зсуву восьмирозрядного числа вліво на:

*a)* одну позицію;

*b)* дві позиції;

*c)* три позиції;

*d)* чотири позиції.

**2.** Побудувати схему, яка реалізує арифметико-логічну операція зсуву восьмирозрядного числа вправо на:

*a)* одну позицію;

*b)* дві позиції;

*c)* три позиції.

*d)* чотири позиції.

**3**. Побудувати схему восьмибітного АЛП, який виконує операції: побітове АБО, побітове І, додавання і логічне ліве зміщення. Розглянути принцип його роботи.

**4.** Побудувати чотирибітний АЛП 74181. За поданою схемою продемонструвати виконання операцій логічних та арифметичних:

*a)* № 1, № 2;

*b)* № 3, № 4;

*c)* № 5, № 6;

*d)* № 7, № 8;

*e)* № 9, № 10;

*f)* № 11, № 12;

*g)* № 13, № 14;

*h)* № 15, № 16;

## Питання для самоконтролю

1. Де використовується АЛП?
2. Які операції може виконувати АЛП?
3. Чи виконує операцію множення АЛП 74181?
4. Як подаються від’ємні числа в АЛП?

# 9. ПРОЕКТУВАННЯ ТА ДОСЛІДЖЕННЯ ТРИГЕРІВ

*Тригер* – найпростіша цифрова схема послідовного типу. У тригерів стан виходу *Y* в будь-який момент часу визначається поточним станом входу *X* та внутрішнім станом схеми *Q*.

$Y=F(X,Q)$

Усі тригери являють собою простий цифровий автомат, що містить елемент пам'яті та керуючу логіку. Іншими словами, цифровий автомат є не тільки перетворювачем, але й зберігає попередню й поточну інформацію (стани). Ця властивість забезпечується наявністю в схемах зворотних зв'язків. Також тригер є основою для складніших функціональних вузлів (лічильники, регістри, суматори, модулі пам'яті).

## 9.1. Класифікація тригерів.

Поточні стани тригерів визначаються значеннями виходів ($Q$ та $\overline{Q}$). При позитивному кодуванні інформації високий рівень напруги на прямому виході ($Q$) відображає значення логічної «1», а низький рівень – логічного «0» ($Q$=0). Значення на виході $\overline{Q}$ будуть відповідно протилежні. Зміна значень вихідних станів забезпечується зміною значень вхідних керуючих сигналів, що позначаються на схемах латинськими літерами (*R, S, T, D, J, K, V* та ін.).

Тригери розрізняються за такими класифікаційними ознаками:

- за логікою функціонування (RS, JK, D, T та ін.);
- за способом запису інформації (синхронні та асинхронні);
- за кількістю ступенів (одно- і двоступінчаті);
- за кількістю тактів синхронізації, необхідних для спрацювання (однотактні, двотактні і тритактні);
- за логічними елементами, на яких вони виконані (І-НЕ, АБО-НЕ, і т.д.).

Входи тригерів поділяються на інформаційні (*R, S, T, D, J, K*) та входи керування (*C, V*). Призначення інформаційних входів полягає у прийомі

сигналів інформації, яку необхідно записати. В тригерах, зазвичай, буває два види керуючих сигналів: синхронізуючий тактовий сигнал *C*, і сигнал дозволу *V*.

### 9.2. Асинхронний RS – тригер.

RS-тригер має два входи *S* і *R*, основний і інверсний виходи. Стан тригера визначається по сигналу на основному вході. Тригер має два керуючих входи: установки *S* (*set* – установка) і скидання *R* (*reset* – скидання), на які подаються вхідні сигнали від зовнішніх джерел. При подачі керуючого сигналу на вхід *S* на основному виході встановлюється логічна одиниця або ця одиниця підтверджується, якщо вона там була (табл. 20). При подачі керуючого сигналу на вхід *R* на основному виході з'являється логічний нуль, тобто, тригер скидається. Якщо тригер був уже скинутий, то скидання підтверджується. Подача керуючих сигналів одночасно на входи *S* і *R* заборонена. У відсутності керуючих сигналів стан тригера змінитися не може, тригер знаходиться в режимі збереження інформації. В залежності від типу логічних елементів, на яких зібраний тригер, керуючими сигналами можуть бути, як нулі, так і одиниці.

У табл. 20, $Q^t$ – це значення вихідного сигналу до моменту подачі керуючих сигналів $S^t$ і $R^t$, або його вихідний стан. $Q^{t+1}$ – новий стан тригера після подачі керуючих сигналів, якими є логічні одиниці.

На рис. 60 показана процедура мінімізації функції $Q^{t+1}$ з використанням діаграм Вейча. Отримана формула описує роботу RS-тригера.

З діаграми Вейча отримаємо ДДНФ функцію $Q^{t+1}$:

$$Q^{t+1} = S^t + Q^t \cdot \overline{R^t}.$$

Схеми тригерів будують після перетворення цієї формули, замінюючи операцію логічного множення на логічне додавання, або логічне додавання замінюють логічним множенням. Після перетворення можна одержати наступну формулу

$$Q^{t+1} = S^t + \overline{\overline{Q^t} + R^t}.$$

Таблиця 20

**Таблиця істинності роботи асинхронного RS-тригера з прямими входами**

| $Q^t$ | $S^t$ | $R^t$ | $Q^{t+1}$ | Стан тригера |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Збереження інформації |
| 0 | 0 | 1 | 0 | Підтвердження 0 |
| 0 | 1 | 0 | 1 | Установка в 1 |
| 0 | 1 | 1 | X | Заборона |
| 1 | 0 | 0 | 1 | Збереження інформації |
| 1 | 0 | 1 | 0 | Скидання в 0 |
| 1 | 1 | 0 | 1 | Підтвердження 1 |
| 1 | 1 | 1 | X | Заборона |

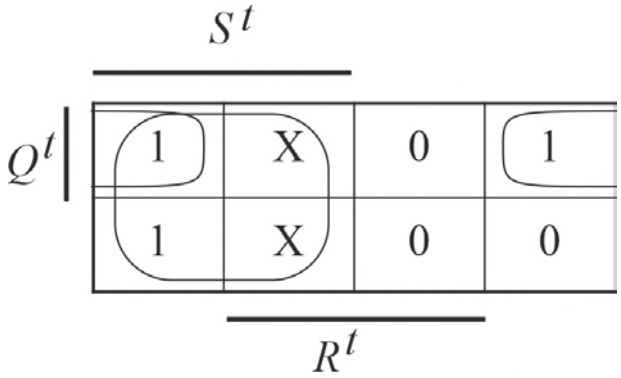



Рис. 60. Діаграма Вейча для побудови функції $Q^{t+1}$ RS-тригера

Якщо замінити додавання на множення, то одержимо:

$$Q^{t+1} = \overline{\overline{S^t} \cdot \overline{Q^t \cdot \overline{R^t}}}.$$

Схеми тригерів, побудовані за цими формулами показані на рис. 61. Перша зі схем (рис. 61, *а*) побудована на елементах АБО-НЕ. Цей тригер керується логічними одиницями. Таблиця його функціонування наведена в табл. 20. Схема тригера побудована за другою формулою на елементах І-НЕ найпоширеніша. Цей тригер керується логічними нулями, тобто має інверсні входи. Табл. 21 – таблиця функціонування даного RS – тригера.

*а)* *б)*

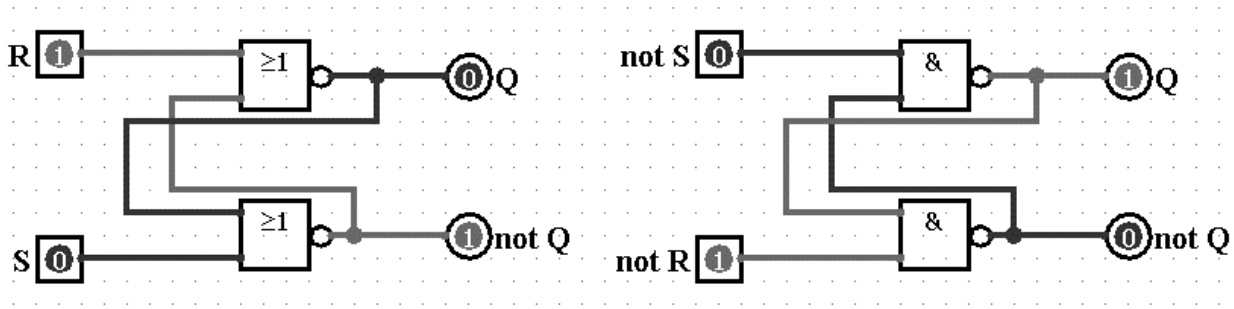



Рис. 61. Схема асинхронного RS – тригера :

*а)* побудованого на елементах АБО-НЕ, *б)* побудованого на елементах І-НЕ

Таблиця 21

**Таблиця істинності роботи асинхронного RS-тригера з інверсними входами**

| $Q^t$ | $S^t$ | $R^t$ | $Q^{t+1}$ | Стан тригера |
|---|---|---|---|---|
| 0 | 0 | 0 | X | Заборона |
| 0 | 0 | 1 | 1 | Установка в 1 |
| 0 | 1 | 0 | 0 | Підтвердження 0 |
| 0 | 1 | 1 | 0 | Збереження інформації |
| 1 | 0 | 0 | X | Заборона |
| 1 | 0 | 1 | 1 | Підтвердження 1 |
| 1 | 1 | 0 | 0 | Скидання |
| 1 | 1 | 1 | 1 | Збереження інформації |

Широкому використанню асинхронного RS-тригера в якості самостійного пристрою заважають його серйозні недоліки: наявність заборонених комбінацій вхідних сигналів, подача інформації по двох окремих входах.

### 9.3. Синхронний RS-тригер

RS-тригер може бути синхронним. У цьому випадку, крім двох інформаційних входів *S* і *R*, тригер має ще вхід синхронізації. Сигнали на входах *S* і *R* лише готують тригер до потрібного переключення, а саме переключення відбувається тільки в момент подачі синхронізуючого імпульсу. Для побудови синхронного тригера на елементах І-НЕ потрібно замінити у логічному виразі функції роботи RS-тригера змінні *R* і *S* на *CR* і *CS*:

$$Q^{t+1} = \overline{\overline{CS^t} \cdot \overline{Q^t \cdot \overline{CR^t}}}.$$

Тобто синхронізація організовується за допомогою двох додаткових елементів І-НЕ. При відсутності сигналу синхронізації (*C*=0) на входах асинхронного RS-тригера встановлюються дві одиниці, що забезпечує в ньому збереження інформації. При подачі синхронізуючого сигналу (*C*=1) тригер перемикається відповідно до інформації, поданої на входи *S* і *R*. На рис. 62. наведено схему одноступінчастого синхронного тригера.

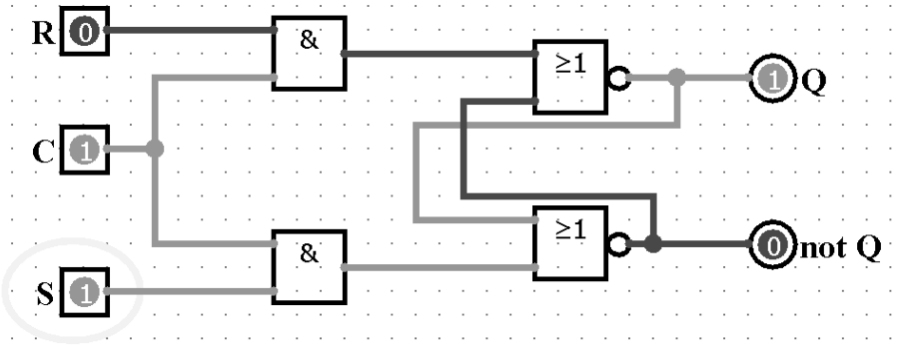



Рис. 62. Схема одноступінчастого синхронного RS – тригера

Двоступінчастий синхронний RS-тригер складається з двох одноступінчастих синхронних RS-тригерів, об'єднаних за принципом «ведучий-ведений» (рис. 63). При подачі синхроімпульсу *C* перемикається перший тригер відповідно до таблиці функціонування одноступінчастого синхронного RS-тригера. На вхід *C*2 синхроімпульс *C* подається в інверсному вигляді і таким чином, поки на вході першого тригера синхроімпульс дорівнює «1», на вході другого він дорівнює «0». І другий тригер перемикається після

того як переключився перший. Інформація з першого тригера (першого ступеня) переписується у другій тригер (другий ступінь). Таким чином, перший ступінь перемикається завжди, поки *C* = 1; другий ступінь перемикається по задньому фронту сигналу *C*, переписуючи інформацію з першого ступеня.

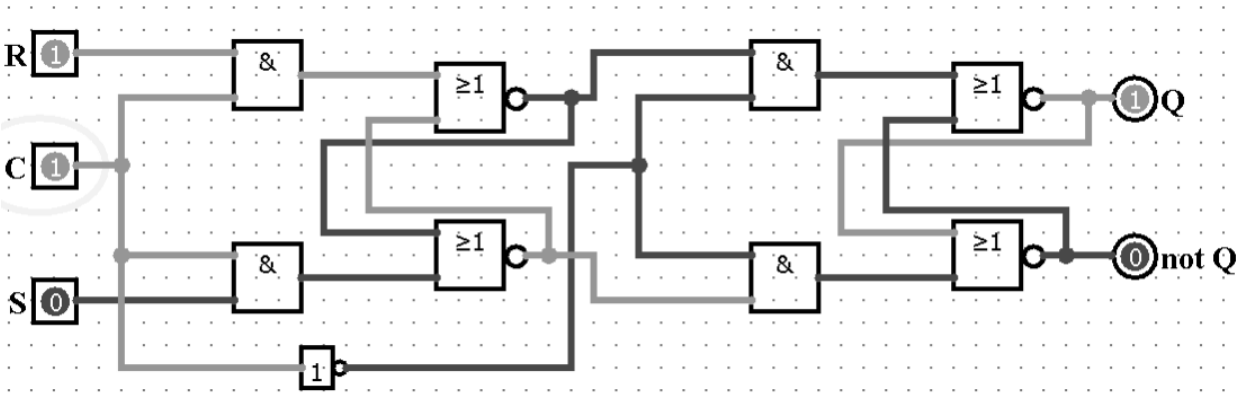



Рис. 63. Схема двоступінчастого синхронного RS – тригера

### 9.4. Статичний синхронний D-тригер

D-тригер, називається ще тригером затримки, може бути асинхронним і синхронним. Проте, асинхронний D-тригер змісту не має, тому що має один інформаційний вхід *D* та основний і інверсний виходи. Опис роботи тригера при різних комбінаціях вхідних сигналів представлений в табл. 22. Сигнал (інформація) на виході завжди збігається з інформацією на вході, тобто $Q^{t+1} = D^t$. Зміст має тільки синхронний D-тригер, у якого крім інформаційного входу *D* є вхід синхронізації *C*. Інформація з входу *D* передається на основний вихід (записується в тригер) у момент надходження синхронізуючого імпульсу. Структурна формула, що описує роботу синхронного D-тригера наступна: $Q^{t+1} = Q^t \cdot \overline{C^t} + C^t \cdot D^t$. З формули видно, що при *C*=0 стан тригера не міняється ($Q^{t+1} = Q^t$), а при *C*=1 стан тригера збігається із значенням інформації на вході *D* ($Q^{t+1} = D^t$). У такий спосіб при відсутності синхронізуючого імпульсу стан тригера не міняється, інформація записана в тригер зберігається (затримується) на період проходження синхронізуючих

імпульсів. На рис. 64 показаний один з варіантів схеми D-тригера. При $C$ = 0, на входах асинхронного RS-тригера, що входить до складу D-тригера, встановлюються дві одиниці, що означає збереження інформації. Можна простежити за схемою, що при $C$=1 тригер встановиться в 1, якщо на його вході $D$ була 1, і скинеться в 0, якщо на вході $D$ був логічний 0.

Таблиця 22

**Таблиця істинності роботи синхронного D-тригера з інверсними входами**

| $C^t$ | $D^t$ | $Q^{t+1}$ | Стан тригера |
|---|---|---|---|
| 0 | 0 | $Q^t$ | Збереження інформації |
| 0 | 1 | $Q^t$ | Збереження інформації |
| 1 | 0 | 0 | Запис 0 |
| 1 | 1 | 1 | Запис 1 |

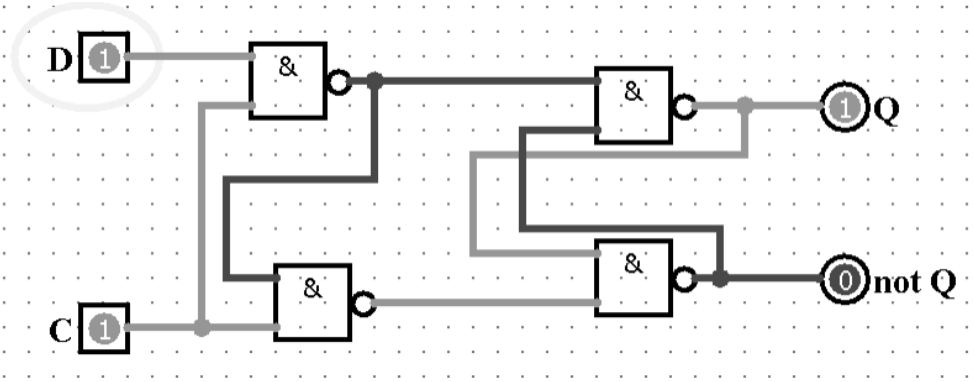



Рис. 64. Схема синхронного статичного D-тригера

Якщо сигнал зміниться під час дії синхроімпульсу, то в тригері виявиться записаною та інформація, яка була перед закінченням синхроімпульсу. Завдяки цій властивості (зміни інформації впродовж усього часу, поки $C$=1) даний тригер отримав назву статичного синхронного D-тригера. Для нормальної роботи статичного D-тригера необхідно, щоб зміна інформації на вході $D$ відбувалась тільки при $C$=0.

### 9.5. Динамічний синхронний D – тригер

Динамічний синхронний D-тригер виключає наскрізну передачу сигналу з *D* входу на вихід тригера в момент дії синхроімпульсу. В тригері з динамічним керуванням інформація записується тільки в момент перепаду напруги на вході сигналу синхронізації. Схема динамічного синхронного D-тригера показана на рис. 65. Умовне графічне позначення рис. 66.*а.*

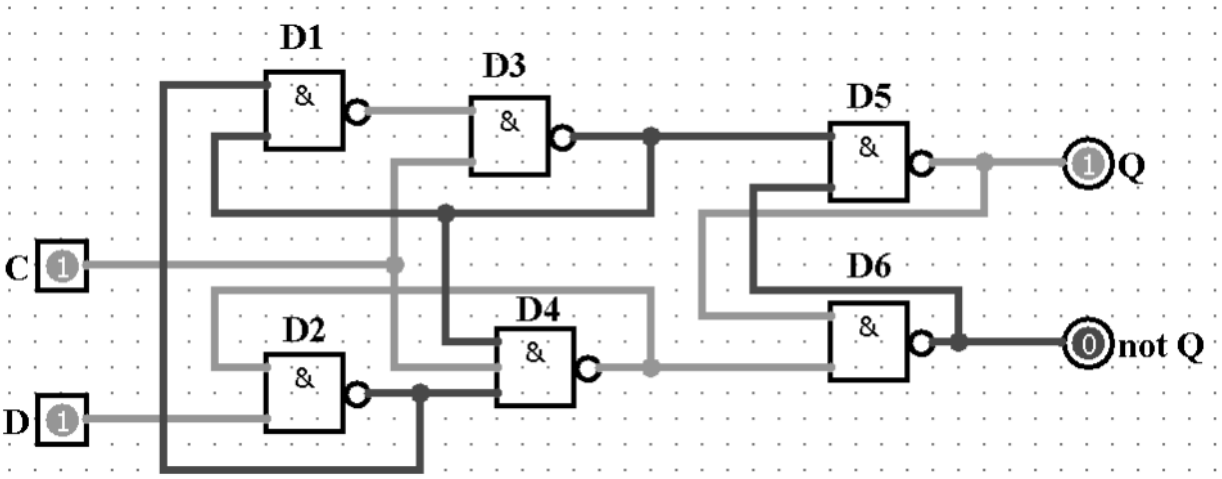



Рис. 65. Схема синхронного динамічного D-тригера

*а)* *б)*

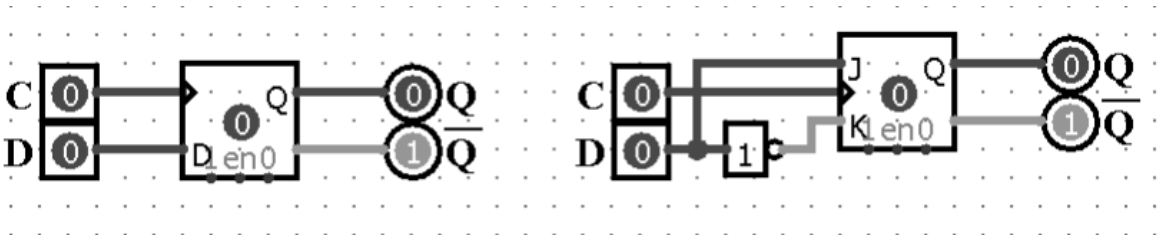



Рис. 66. Схема D-тригера з бібліотеки Logisim:

*а)* динамічний D-тригер, *б)* D-тригер побудований на основі JK-тригера

Динамічний D-тригер складається з трьох статичних RS-тригерів. Перших два (складених з елементів D1, D3 та D2, D4) готують інформацію. Третій тригер (D5, D6) записує попередньо оброблену інформацію. Саме така двоступінчаста побудова динамічного тригера і дозволяє позбутися від прямого проходження сигналу з входу *D* на вихід тригера в період дії синхроімпульсу.

Ускладнення схеми зображеної на рис. 65, дозволяє отримати універсальний D-тригер (рис. 67), який виконує функції як RS-тригера так і динамічного D-тригера. Модернізація полягає в заміні всіх двовхідних елементів І-НЕ на тривхідні елементи І-НЕ. Додаткові входи елементів D1, D5 будуть входами сигналу *not S*, а входи D2, D3, D6 – входами сигналу *not* R. Поки сигнал на *not S* і *not R* рівний 1, універсальний тригер працює як динамічний D-тригер. Як тільки на один з входів *not S* або *not R* поступає сигнал, рівний 0, то тригер зразу перестає реагувати на сигнали *C* і *D* і приймає стан який визначається сигналами *not S* або *not R*. Стан *not S*= *not R*=0 вважається забороненим.

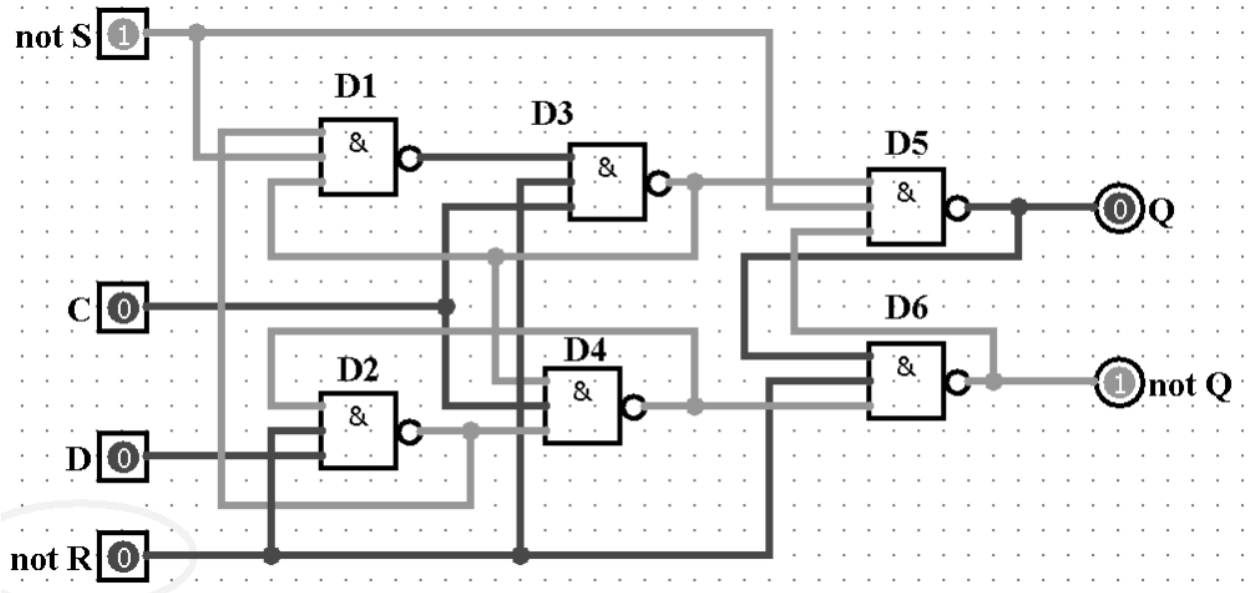



Рис. 67. Схема універсального D-тригера

### 9.6. JK-тригер

Несинхронний JK-тригер має два входи *J* і *K*, основний і інверсний виходи. Вхід *J* аналогічний входові *S* RS-тригера. По цьому входу тригер встановлюється в стан «1». По входу *K* тригер скидається в «0», як і RS-тригері по входу *R*. Відмінність від RS-тригера полягає в тому, що цей тригер не має заборонених комбінацій сигналів на входах, а при подачі керуючих сигналів

одночасно на обидва входи тригер переключається в протилежний стан (табл. 23). Синхронний JK-тригер має ще один вхід *C* – вхід синхронізації і переключається тільки при подачі імпульсу на цей вхід.

Таблиця 23

**Таблиця істинності роботи синхронного JK-тригера з інверсними входами**

| $Q^t$ | $J^t$ | $K^t$ | $Q^{t+1}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

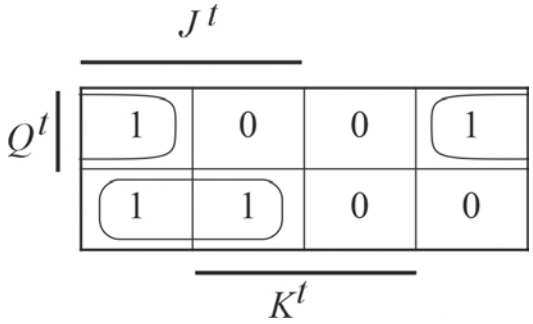



Рис. 68. Діаграма Вейча для побудови функції $Q^{t+1}$ JK -тригера

Структурна формула, отримана з діаграм Вейча (рис. 68), що описує роботу несинхронного JK-тригера має такий вигляд:

$$Q^{t+1} = J^t\overline{Q^t} + \overline{K^t}Q^t.$$

Синхронний JK-тригер можна побудувати на основі синхронного динамічного D-тригера (рис. 69).

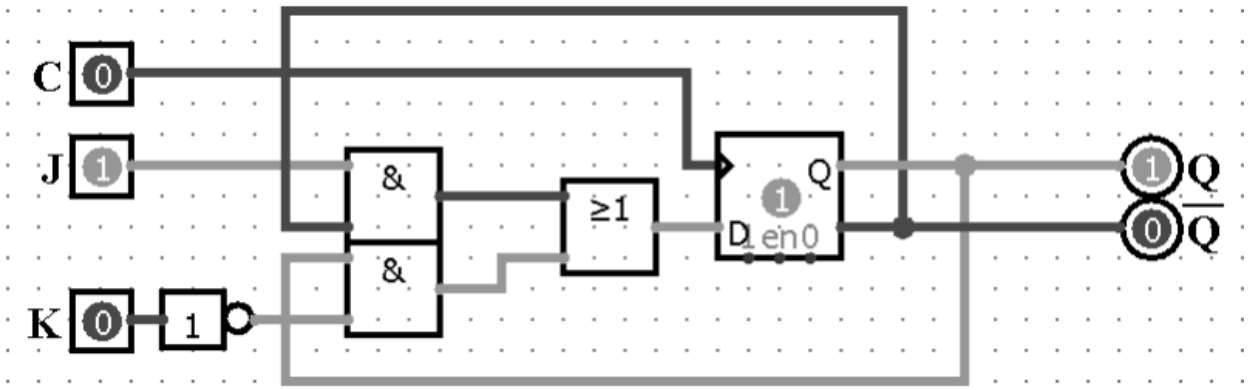



Рис. 69. Схема синхронного JK-тригера, побудована
на основі динамічного D-тригера

JK-тригер називають універсальним, тому що з нього можна зробити будь-який тип тригера. RS-тригер одержується з JK-тригера, коли входи *J* і *K* використовуються як входи *S* і *R* відповідно, а заборонена комбінація не подається.

Якщо у формулі несинхронного JK-тригера *J* назвати входом *D*, а на вхід *K* подати $\overline{D}$, то одержимо: $Q^{t+1} = D(\overline{Q^t} + Q^t) = D$, що відповідає несинхронному D-тригерові. Проте, оскільки несинхронний D-тригер змісту не має, то для одержання синхронного D-тригера потрібно використовувати синхронний JK-тригер. Для одержання T-тригера досить об'єднати входи *J* і *K* і назвати цей вхід входом *T* по якому тригер буде переключатися в протилежний стан, як це повинен робити T-тригер.

### 9.7. T-тригер (лічильний тригер)

Несинхронний T-тригер має один вхід *T*, основний і інверсний виходи. Інформація на виходах міняється на протилежну при кожному позитивному перепаді напруги на вході *T*. Тому цей тригер використовують в якості подільника частоти вхідного сигналу. Тригер такого типу може бути

створений з D-тригера з динамічним управлінням, якщо його інверсний вихід з'єднати з інформаційним входом (рис. 70). При цьому, якщо в початковий момент часу на виході *Q* був нульовий рівень, то на вході *D* був рівень *not Q*=1. По фронту першого синхроімпульсу одиниця з входу *D* перепишеться (з запізненням, рівним затримці одного логічного елемента) на вихід *Q*. Відповідно на виході і вході *D* з'явиться нульовий рівень (з запізненням, рівним затримці двох логічних елементів). В наступному такті на вихід *Q* буде переписано нульове значення з входу *D* і т. д.

Побудувати лічильний тригер на базі статичного D-тригера таким чином (зворотнім зв'язком з виходу *not Q* на вхід *D*) не можливо. Оскільки статичний тригер має потенційне управління, то при *C*=1 напруга на виході за рахунок впливу зворотного зв'язку буде постійно змінюватися на протилежну, тобто виникне високочастотне коливання.

Схему T-тригера, також можна побудувати на основі універсального JK-тригера (рис. 70).

*а)* *б)*

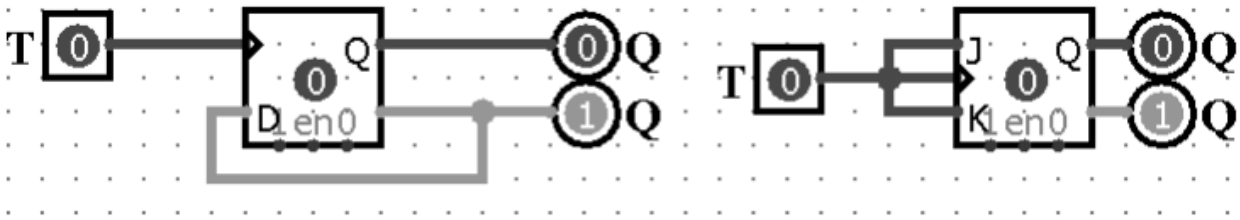



Рис. 70. Схема T-тригера, побудована на основі D-тригера (а) та універсального JK-тригера (б)

**Завдання**

1. Побудувати та розглянути принцип роботи RS-тригера:
   - *a)* асинхронного на елементах АБО;
   - *b)* асинхронного на елементах І;
   - *c)* синхронного на елементах АБО;
   - *d)* синхронного на елементах І;

*e)* двоступінчастого синхронного на елементах АБО;

*f)* двоступінчастого синхронного на елементах І.

2. Побудувати та розглянути принцип роботи D-тригера:

*a)* асинхронного статичного на елементах АБО, НЕ;

*b)* асинхронного статичного на елементах І, НЕ;

*c)* синхронного статичного на елементах АБО, НЕ;

*d)* синхронного статичного на елементах І, НЕ;

*e)* синхронного динамічного на елементах АБО, НЕ;

*f)* синхронного динамічного на елементах І, НЕ;

*g)* універсального, який виконує функції як RS-тригера, так і динамічного D-тригера;

*h)* використовуючи RS-тригер;

*i)* використовуючи JK- тригер.

3. Побудувати JK-тригер, та розглянути принцип його роботи:

*a)* універсальний на елементах АБО, НЕ;

*b)* універсальний на елементах І НЕ;

*c)* на основі D-тригера (D-тригер взяти з бібліотеки Logisim);

*d)* на основі динамічного D-тригера побудованого на логічних елементах.

3. Побудувати та розглянути принцип роботи Т-тригера:

*a)* на елементах АБО, НЕ;

*b)* на елементах І НЕ;

*c)* на основі D-тригера (D-тригер взяти з бібліотеки Logisim);

*d)* на основі JK -тригер (JK -тригер взяти з бібліотеки Logisim).

**Питання для самоконтролю**

1. Дати визначення тригера. Принципи його роботи.
2. Навести класифікацію тригерів за способом прийому інформації.
3. Дати визначення таблиці переходів.
4. Навести класифікацію тригерів за принципом побудови.
5. Навести класифікацію тригерів за функціональним можливостям

6. Як властивість запам’ятовування відображається в характеристичних рівняннях тригерів?
7. В чому принципова різниця роботи синхронних тригерів від асинхронних?
8. Який пріоритет інформаційних і установочних входів в синхронних тригерах?
9. Що собою представляє RS – тригер, його умовне позначення?
10. Чому комбінація сигналів 11 на входах RS-тригера називається «забороненою»?
11. Які ви знаєте різновиди RS – тригера?
12. Чим відрізняється робота RS-тригера з прямими входами від роботи з інверсними входами?
13. Що являє собою JK – тригер?
14. Що являє собою D – тригер?
15. Що являє собою T – тригер?
16. Чому T- тригер отримав назву лічильного ? Яке число імпульсів він може порахувати?
17. Як працює D-тригер, якщо D=Q?

# 10. ПРОЕКТУВАННЯ ТА ДОСЛІДЖЕННЯ РЕГІСТРІВ

*Регістром* називається пристрій, що здійснює прийом, збереження перетворення і видачу чисел у вигляді *n*-розрядного двійкового коду. Регістри будують на тригерах з входами установки, які реалізують запам'ятовування двійкових чисел. Інші операції виконуються за допомогою підключення до входів і виходів тригерів логічних елементів та за рахунок організації відповідних зв'язків між тригерами. Кількість тригерів відповідає числу розрядів двійкового коду. Регістри виконують ряд мікрооперацій над словами:

- прийом слова в регістр у прямому і оберненому коді, дані зберігаються в регістрі поки не з'явиться команда на їхню зміну;
- видача слова з регістра в прямому і оберненому коді;
- виконання порозрядних логічних операцій над декількома словами;
- зсув коду вправо чи вліво на необхідне число розрядів, перетворення паралельного коду в послідовний і навпаки.

Існують також регістри, які виконують порозрядні операції кон'юнкції, диз'юнкції та додавання за mod 2. За способом запису і зчитування коду числа в регістрі розрізняють: паралельні регістри, послідовні регістри та комбіновані регістри.

## 10.1. Паралельні регістри

У паралельному регістрі інформація надходить у вигляді паралельного коду. Тобто код числа, що запам'ятовується, подається на інформаційні входи всіх тригерів і записується в регістр із приходом тактового імпульсу. Вихідна інформація змінюється з подачею нового вхідного слова і приходом наступного імпульсу запису. Такі регістри використовують у системах оперативної пам'яті. Число тригерів в них дорівнює максимальній розрядності збережених слів. На рис. 71 зображено паралельний чотирирозрядний регістр на тактуючих D-тригерах. На тактові входи всіх тригерів одночасно подається логічний сигнал *C*. Під час подачі імпульсу *C*

спрацьовують усі тригери. Інформація зберігається в паралельному регістрі у вигляді паралельного коду і може бути зчитана з виходів тригерів.

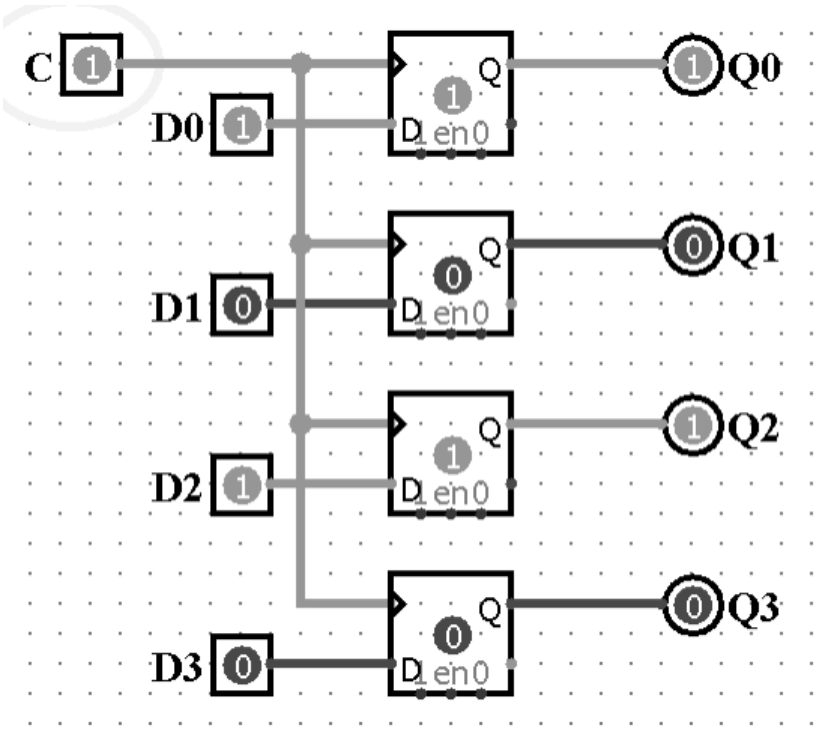



Рис. 71. Паралельний регістр на тактуючих D-тригерах

В Logisim в бібліотеці «Память» є елемент регістр «Регистр» (рис. 72), який зберігає одне багатобітне значення, і видає його на вихід Q при подачі синхронізуючого імпульсу.

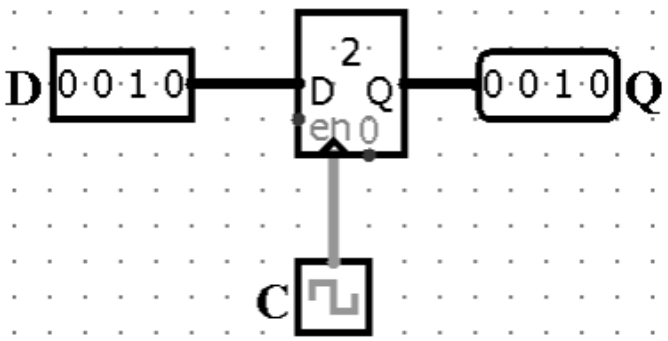



Рис. 72. Елемент регістр з бібліотеки «Память» в Logisim

## 10.2. Послідовні регістри

Послідовні регістри, складаються з послідовно з’єднаних тригерів. Послідовні регістри, ще називають регістрами зсуву. Регістри зсуву здійснюють зсув інформації в одному з напрямків: в сторону молодших розрядів, або старших. Тому розрізняють регістри зсуву вправо та зсуву вліво. Є ще реверсивні регістри, які виконують зсув інформації в обидва боки, в залежності від значення на керуючому вході.

У регістрі зсуву запис інформації, яка надходить у послідовному коді, відбувається побітно, у відповідності до її надходження. Схема послідовного регістра зсуву вправо на D-тригерах з динамічним керуванням, наведена на рис. 73.

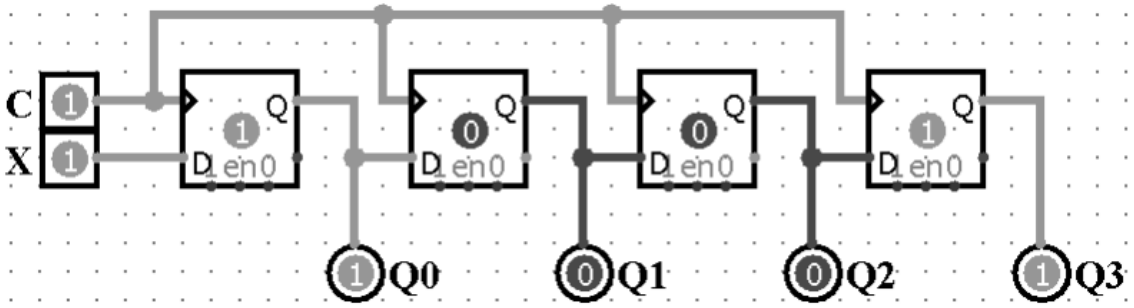



Рис. 73. Послідовний регістр вправо вліво на тактуючих D-тригерах

При надходженні тактового імпульсу *C* перший тригер записує код *X* (0 або 1), що перебуває в цей момент на його вході *D*, а кожний наступний тригер перемикається в стан, у якому до цього перебував попередній. Так відбувається тому, що записуваний сигнал проходить із входу D-тригера до виходу *Q* із затримкою, більшою тривалості переднього фронту тактового імпульсу (протягом якого й відбувається запис). Кожний тактовий імпульс послідовно зсуває код числа в регістрі на один розряд. Тому для запису *n*-розрядного коду потрібно *n* тактових імпульсів. Чотиризначне число 1011 було записано у відповідні розряди регістра (1 – *Q*3, 0 – *Q*2, 1 – *Q*1, 1 – *Q*0) після приходу четвертого тактового імпульсу. До наступного тактового імпульсу це число зберігається в регістрі у вигляді паралельного коду на

виходах *Q*3, …, *Q*0. Якщо необхідно одержати збережену інформацію в послідовному коді, то її знімають із виходу *Q*4 у моменти приходу наступних чотирьох імпульсів (5–8). Такий режим називається режимом послідовного зчитування.

На рис. 74 зображено схему послідовного регістра зсуву вліво. В даному регістрі заповнення його даними відбувається справа на ліво.

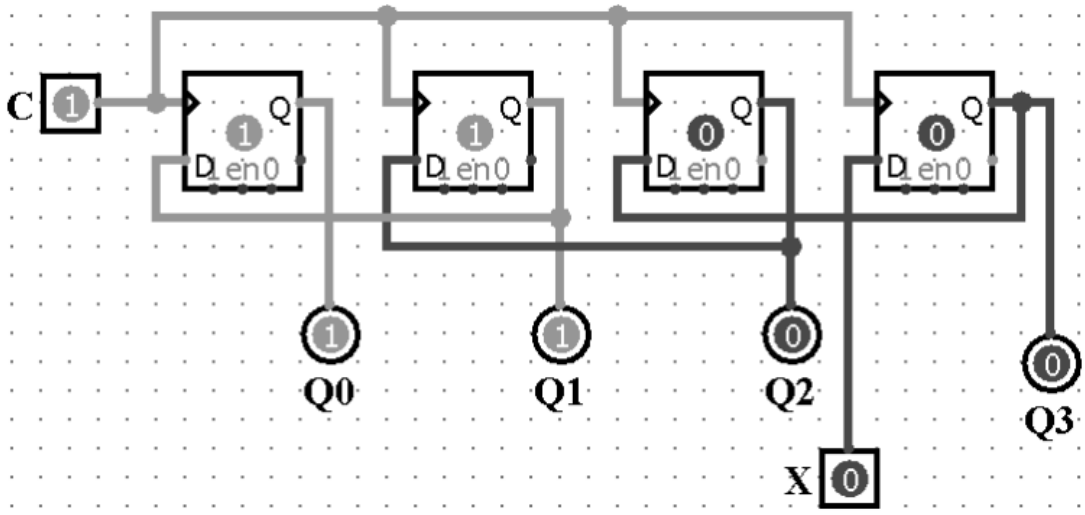



Рис. 74. Послідовний регістр зсуву вліво на тактуючих D-тригерах

В Logisim в бібліотеці «Память» є елемент зсувний регістр «Сдвиговой регистр» (рис. 75). Цей регістр складається з декількох ступенів; кожне спрацьовування тактового входу може привести до того, що кожен ступінь отримає значення попереднього ступеня, а нове значення завантажиться в перший ступінь. Компонент також підтримує паралельне читання і запис значень усіх ступенів.

В цифровій техніці також використовуються регістри, в яких можливо змінювати напрям зсуву у процесі роботи. Такі регістри називаються реверсивними. В них з'єднання розрядів між собою виконується за допомогою логічних схем, які здійснюють комутацію відповідно до вхідних сигналів керування напрямом зсуву.

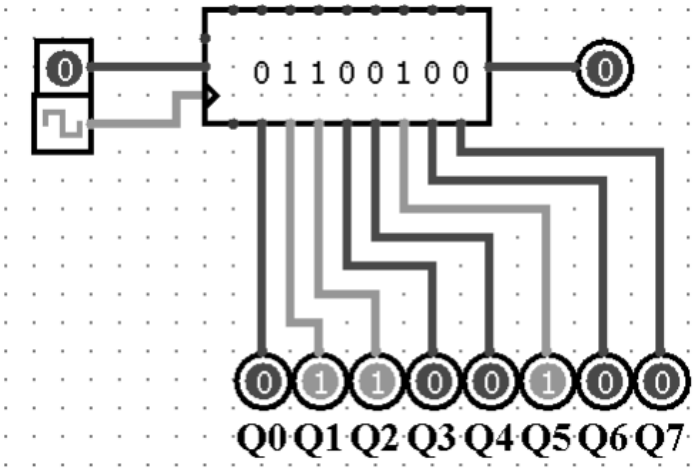



Рис. 75. Елемент зсувний регістр з бібліотеки «Память» в Logisim

Приклад побудови такого регістру показано на рис. 76. Коли на вхід *V* подано «1», то регістр передає дані з входу *Xl* і працює як регістр зсуву вліво. Коли на вхід *V* подано «0», то регістр передає дані з входу *Xr* і працює як регістр зсуву вправо.

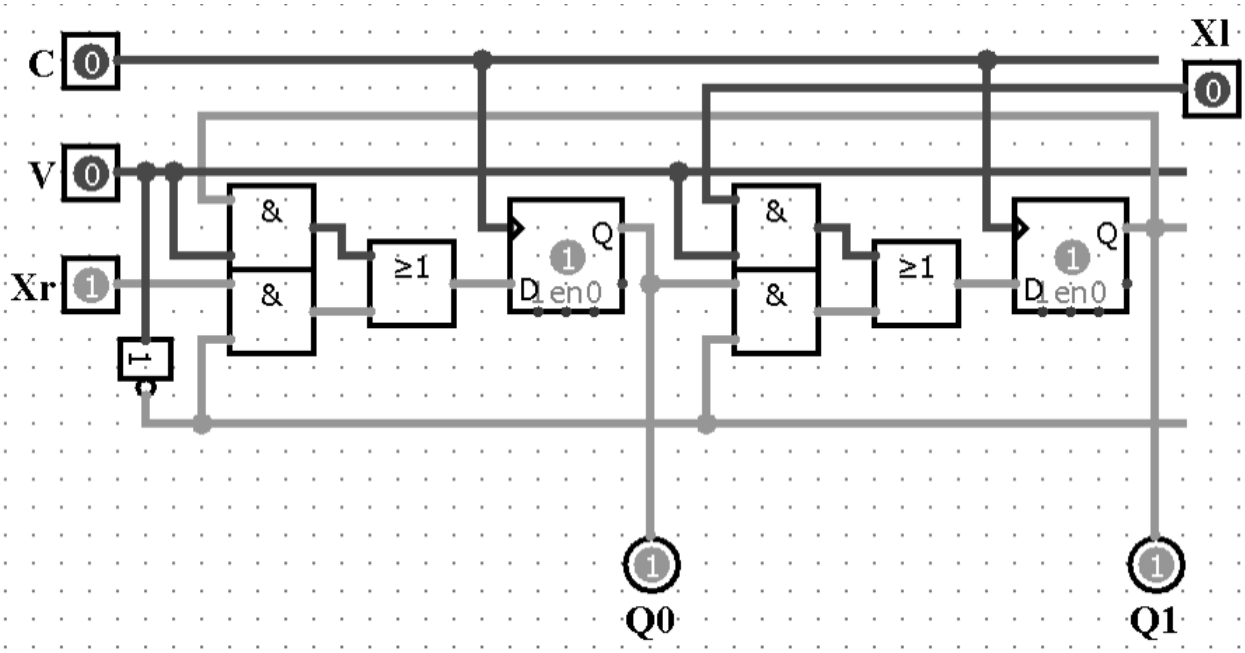



Рис. 76. Послідовний реверсивний регістр на тактуючих D-тригерах

### 10.3. Послідовно паралельний регістр

У послідовно-паралельний регістр інформація записується в послідовному багаторозрядному коді, а потім паралельно зчитується з усіх розрядів сигналом управління. На рис. 77 подана схема послідовно паралельного регістра в режимі запису і зчитування. Послідовно паралельний регістр має один інформаційний вхід, вхід запису $C_з$ вхід читання $C_ч$ і чотири виходи. Спочатку за чотири такти на вході $C_з$ здійснюється запис даних в регістр з входу $X$, а потім за один такт на вході $C_ч$ здійснюється зчитування даних на чотири виходи ($Q3$, $Q2$, $Q1$, $Q0$).

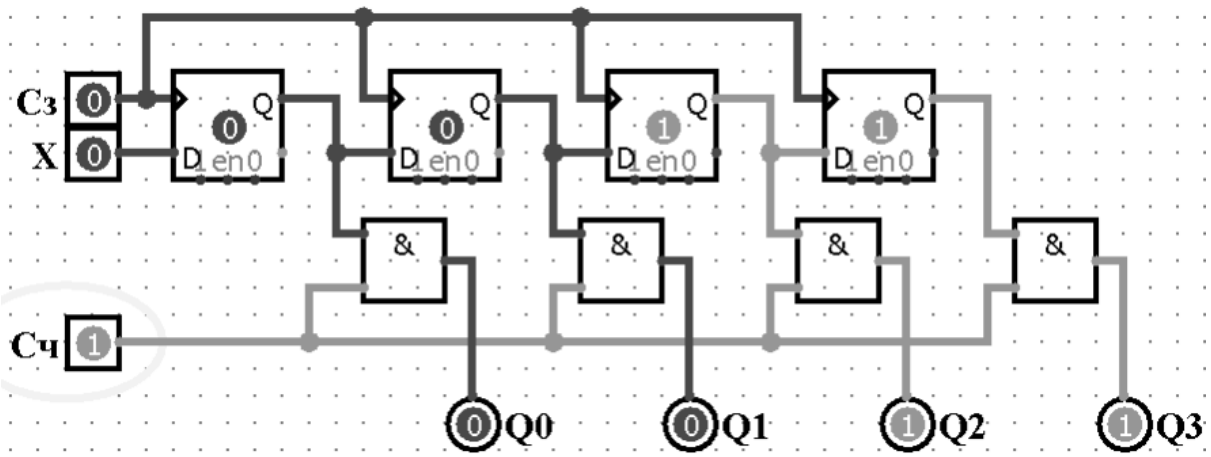



Рис. 77. Схема послідовно паралельного регістра в режимі послідовного запису і паралельного зчитування

### 10.4. Паралельно послідовний регістр

У паралельно-послідовний регістр дані записуються паралельно в усі розряди, а зчитування інформації здійснюється у послідовній формі розряд за розрядом. На рис. 78 подана схема паралельно послідовного регістра в режимі запису і зчитування. Паралельно послідовний регістр має чотири інформаційних входи, вхід запису $C_з$ вхід читання $C_ч$ і один вихід. Спочатку за один такт на вході $C_з$ здійснюється запис даних в регістр з входів D0, D1, D2, D3. За допомогою входу V дані передаються другий блок тригерів і за один такт на вході $C_ч$ здійснюється запис вхідних даних в другий блок тригерів. В режимі V=0 за чотири такти $C_ч$ відбувається зчитування даних на вихід $Q$.

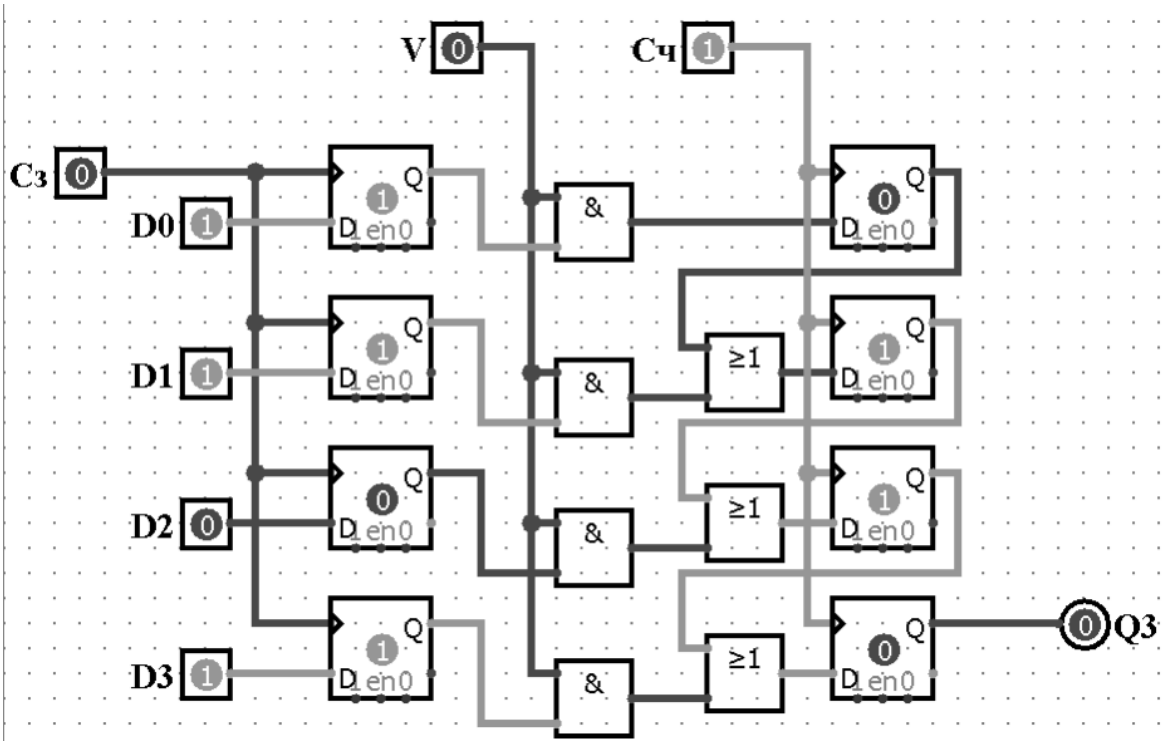



Рис. 78. Схема паралельно послідовного регістра в режимі паралельного запису і послідовного зчитування.

Найбільш універсальними вважаються регістри, які мають у своєму складі одночасно послідовні і паралельні входи й виходи. Такі регістри називають регістрами з послідовно-паралельним прийманням інформації та послідовно-паралельним передаванням.

**Завдання**

1. Побудувати та розглянути принцип роботи трирозрядного паралельного регістра:
   - *a)* на D-тригерах (тригер взяти з бібліотеки Logisim);
   - *b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);
   - *c)* на RS- тригерах (тригер взяти з бібліотеки Logisim);
   - *d)* на D-тригерах з видачею інформації в оберненому коді (тригер взяти з бібліотеки Logisim);
   - *e)* на JK-тригерах з видачею інформації в оберненому коді (тригер взяти з бібліотеки Logisim);

*f)* на RS- тригерах з видачею інформації в оберненому коді (тригер взяти з бібліотеки Logisim).

2. Побудувати та розглянути принцип роботи чотирирозрядного паралельного регістра:
   *a)* на D-тригерах (тригер взяти з бібліотеки Logisim);
   *b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);
   *c)* на RS- тригерах (тригер взяти з бібліотеки Logisim);
   *d)* на D-тригерах з видачею інформації в оберненому коді (тригер взяти з бібліотеки Logisim);
   *e)* на JK-тригерах з видачею інформації в оберненому коді (тригер взяти з бібліотеки Logisim);
   *f)* на RS- тригерах з видачею інформації в оберненому коді (тригер взяти з бібліотеки Logisim).
3. Побудувати та розглянути принцип роботи трирозрядного послідовного регістра:
   *a)* на D-тригерах (тригер взяти з бібліотеки Logisim);
   *b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);
   *c)* на RS- тригерах (тригер взяти з бібліотеки Logisim);
   *d)* на D-тригерах з видачею інформації в оберненому коді (тригер взяти з бібліотеки Logisim);
   *e)* на JK-тригерах з видачею інформації в оберненому коді (тригер взяти з бібліотеки Logisim);
   *f)* на RS- тригерах з видачею інформації в оберненому коді (тригер взяти з бібліотеки Logisim).
4. Побудувати та розглянути принцип роботи чотирирозрядного послідовного регістра:
   *a)* на D-тригерах (тригер взяти з бібліотеки Logisim);
   *b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);
   *c)* на RS- тригерах (тригер взяти з бібліотеки Logisim);
   *d)* на D-тригерах з видачею інформації в оберненому коді (тригер взяти з бібліотеки Logisim);

*e)* на JK-тригерах з видачею інформації в оберненому коді (тригер взяти з бібліотеки Logisim);

*f)* на RS- тригерах з видачею інформації в оберненому коді (тригер взяти з бібліотеки Logisim).

5. Побудувати та розглянути принцип роботи трирозрядного послідовного реверсивного регістра:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на RS- тригерах (тригер взяти з бібліотеки Logisim);

*d)* на D-тригерах з видачею інформації в оберненому коді (тригер взяти з бібліотеки Logisim);

*e)* на JK-тригерах з видачею інформації в оберненому коді (тригер взяти з бібліотеки Logisim);

*f)* на RS- тригерах з видачею інформації в оберненому коді (тригер взяти з бібліотеки Logisim).

6. Побудувати та розглянути принцип роботи чотирирозрядного послідовного реверсивного регістра:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на RS- тригерах (тригер взяти з бібліотеки Logisim);

*d)* на D-тригерах з видачею інформації в оберненому коді (тригер взяти з бібліотеки Logisim);

*e)* на JK-тригерах з видачею інформації в оберненому коді (тригер взяти з бібліотеки Logisim);

*f)* на RS- тригерах з видачею інформації в оберненому коді (тригер взяти з бібліотеки Logisim).

7. Побудувати та розглянути принцип роботи трирозрядного регістра з послідовним завантаженням даних та паралельною видачею інформації:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на RS- тригерах (тригер взяти з бібліотеки Logisim).

8. Побудувати та розглянути принцип роботи чотирирозрядного регістра з послідовним завантаженням даних та паралельною видачею інформації:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на RS- тригерах (тригер взяти з бібліотеки Logisim).

9. Побудувати та розглянути принцип роботи трирозрядного регістра з паралельним завантаженням даних та послідовною видачею інформації:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на RS- тригерах (тригер взяти з бібліотеки Logisim).

10. Побудувати та розглянути принцип роботи чотирирозрядного регістра з паралельним завантаженням даних та послідовною видачею інформації:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на RS- тригерах (тригер взяти з бібліотеки Logisim).

11. Побудувати та розглянути принцип роботи п’ятирозрядного регістра з паралельним завантаженням даних та послідовною видачею інформації:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на RS- тригерах (тригер взяти з бібліотеки Logisim).

12. Складіть принципову схему трирозрядного регістра із двома керуючими входами, один із яких встановлює паралельне введення числа, а інший послідовне:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на RS- тригерах (тригер взяти з бібліотеки Logisim).

13. Складіть принципову схему чотирирозрядного регістра із двома керуючими входами, один із яких встановлює паралельне введення числа, а інший послідовне:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на RS- тригерах (тригер взяти з бібліотеки Logisim).

14. Складіть принципову схему п’ятирозрядного регістра із двома керуючими входами, один із яких встановлює паралельне введення числа, а інший послідовне:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на RS- тригерах (тригер взяти з бібліотеки Logisim).

**Питання для самоконтролю**

1. Що таке регістр? Для чого необхідні регістри?
2. Що таке розряд регістру?
3. Які регістри бувають, в залежності від способу запису інформації?
4. Які регістри називають реверсивними?
5. На основі яких елементів можна побудувати запам’ятовуючий елемент регістру?
6. Скільки імпульсів на вході запису, потрібно для запису *n*-розрядного числа в регістр послідовного типу?
7. Скільки імпульсів на вході запису, потрібно для запису *n*-розрядного числа в регістр паралельного типу?
8. Який код буде записаний в *Q*0, *Q*1, *Q*2, *Q*3 Якщо *Q*0=0, *Q*1=1,*Q*2=0, *Q*3=1 і на вхід подати 2 імпульси просування (при відсутності коду на вході)?
9. Який код буде записаний в *Q*0, *Q*1, *Q*2, *Q*3 Якщо *Q*0=1, *Q*1=1,*Q*2=0, *Q*3=0 і на вхід подати 2 імпульси просування (при відсутності коду на вході)?
10. Який код буде записаний в *Q*0, *Q*1, *Q*2, *Q*3 Якщо *Q*0=0, *Q*1=1,*Q*2=0, *Q*3=1 і на вхід подати 3 імпульси просування (при присутності одиниці на вході)?
11. Якою повинна бути мінімальна розрядність регістра для запису чисел, десятковий еквівалент найбільшого з яких рівний 45.

# 11. ПРОЕКТУВАННЯ ЛІЧИЛЬНИКІВ

*Лічильником* називають операційний елемент послідовної дії, що здійснює рахунок імпульсів, які надходять на його вхід. Результат рахунку зберігається лічильником до приходу наступного імпульсу. Зчитування результату рахунку може відбуватися в проміжках між рахунковими імпульсами. Тобто, лічильник перетворює число – імпульсний код в певний двійковий код.

Лічильники, як і зсувні регістри, складаються з ланцюжка послідовно включених тригерів. Розрядність лічильника, а отже, і кількість тригерів $N$ визначаються максимальним числом, до якого він повинен рахувати. Це число називається коефіцієнтом (модулем) рахунку – $K_p$. Якщо число вхідних імпульсів $n>K_p$, то через кожні $K_p$ імпульсів лічильник повертається у вихідний стан і починає рахувати імпульси спочатку. До важливих характеристик лічильника належать *час реєстрації* – інтервал часу між моментом надходження вхідного сигналу і закінченням перехідних процесів в схемі та *розрізняльна здатність* – мінімально допустимий період проходження вхідних сигналів, при якому лічильник працює стабільно.

## 11.1. Види лічильників

Велика різноманітність типів лічильників зумовлена їх широким використанням як в обчислювальній техніці, так і в різних пристроях автоматики. Лічильники застосовуються для утворення послідовностей адрес команд, для рахунку числа циклів виконання операцій, для запам’ятовування коду в аналого-цифрових перетворювачах і т.д..

Лічильники *класифікуються* за такими параметрами:

- по розрядності;
- підсумовуючі, віднімаючі, реверсивні, з довільним порядком перерахунку;
- синхронні, асинхронні

- за типом формування переносу всередині лічильника: з послідовним, з паралельним, з комбінованим;
- з функцією установки довільного числа, з установкою в нуль.

### 11.2. Асинхронний послідовний двійковий лічильник

Асинхронний двійковий лічильник утворений ланцюжком послідовно включених лічильних тригерів. Асинхронний лічильник може бути створений на лічильному тригері будь якого типу. В більшості випадків використовують D- та JK- тригери, що працюють в режимі Т-тригера.

Результат рахунку відображається на виходах лічильника $Q(n–1)$, .... $Q0$ у вигляді паралельного двійкового коду числа порахованих імпульсів. Оскільки кількість вихідних змінних дорівнює числу тригерів $n$ і кожна змінна може приймати лише два значення, то число можливих станів (коефіцієнт рахунку) дорівнює: $K_p=2^n$. Тому що з $2^n$ станів лічильник переключається на нульовий стан, а максимальне число, при якому лічильник повністю заповнюється одиницями, дорівнює $(2^n-1)$.

Найпростішим однорозрядним лічильником з $K_p=2$ є Т-тригер, що змінює свій стан на протилежний під дією кожного вхідного сигналу. У результаті перепад напруги на виході тригера має удвічі меншу частоту, ніж на вході. По цих перепадах запускається наступний тригер, і на його виході зміни стану відбуваються вже в чотири рази рідше, ніж на вході першого тригера.

З приходом кожного наступного імпульсу паралельний двійковий код на виході лічильника буде збільшуватися на одиницю, поки не настане переповнення лічильника, при якому всі тригери скинуться в нульовий стан.

Істинна інформація на виходах лічильника встановлюється тільки через час $T=n\cdot t$, що пройшов після надходження тактового імпульсу. Де $t$ – затримка поширення імпульсу в кожному тригері. При збільшенні розрядності сумарна затримка збільшується і новий сигнал не можна подавати до того часу, доки не встановиться лічильник, оскільки це може привести до спотворювання

інформації в лічильнику. При збільшенні розрядності сумарна частота понижується $F_{max} \leq \frac{1}{T}$, тому багаторозрядні лічильники з послідовним переносом рахункових імпульсів від тригера до тригера можуть працювати тільки на знижених частотах, при досить великих періодах проходження імпульсів.

На рис. 79 подано схему сумуючого лічильника з коефіцієнтом рахунку 16 побудованого на D-тригерах, кожен з яких працює в режимі Т-тригера. Такий лічильник отримується при подачі інверсних сигналів на тактові входи.

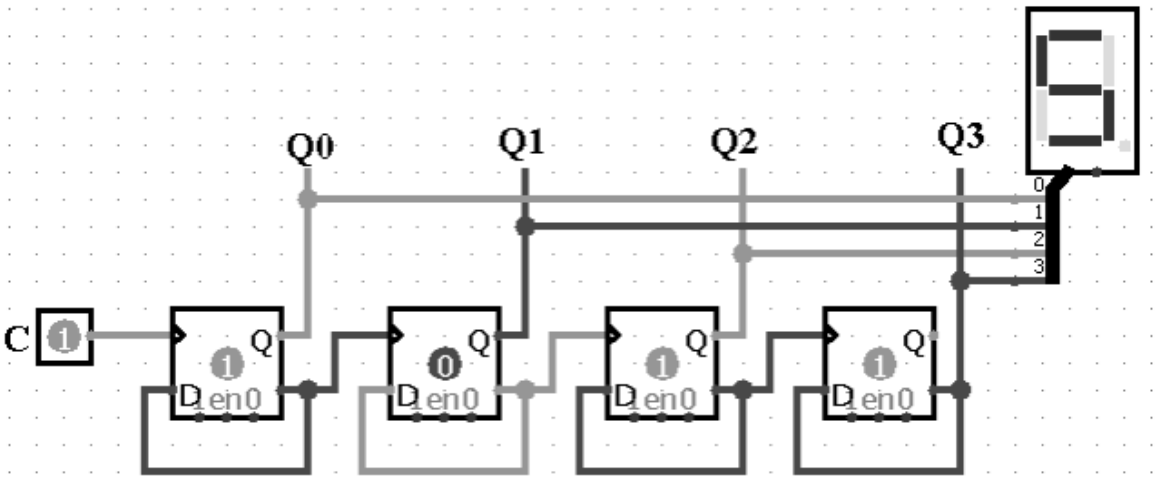



Рис. 79. Сумуючий асинхронний лічильник з послідовним переносом на D-тригерах

Крім розглянутого підсумовуючого лічильника, є й лічильники, що віднімають, у яких вихідний код зменшується на 1 із приходом кожного рахункового імпульсу (рис.80).

Такий лічильник отримується при подачі прямих сигналів на тактові входи. Для цього необхідно тактові входи тригерів підключити до прямих виходів *Q* попередніх тригерів.

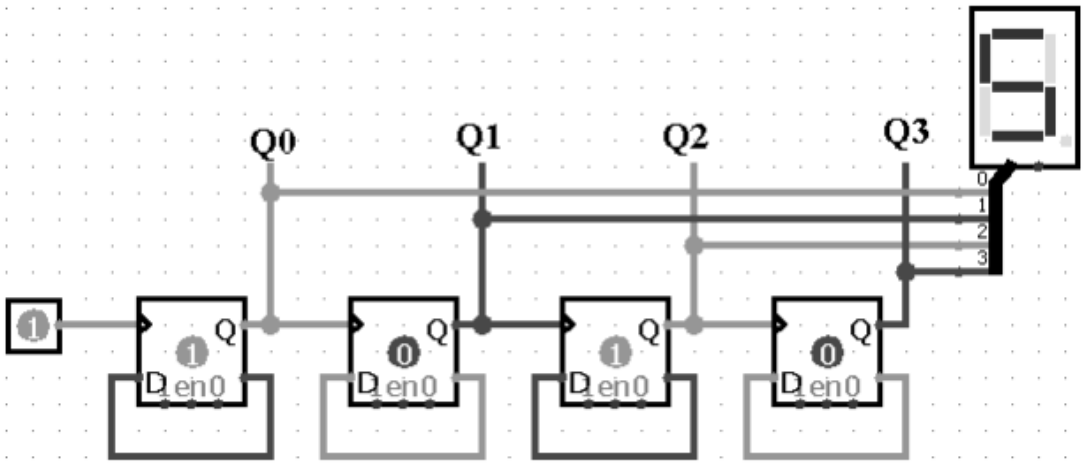



Рис. 80. Віднімаючий асинхронний лічильник з послідовним переносом на D-тригерах

### 11.3. Синхронний (паралельний) двійковий лічильник

Паралельні лічильники (синхронні з паралельним переносом) мають максимальну швидкодію, оскільки в них усі розряди переключаються одночасно.

Паралельний лічильник містить розрядні тригери з кон'юнкторами (рис. 81), що аналізують стан попередніх розрядів. При надходженні вхідного сигналу переключаються тільки ті тригери, для яких усі попередні були в одиничному стані, що і потрібно.

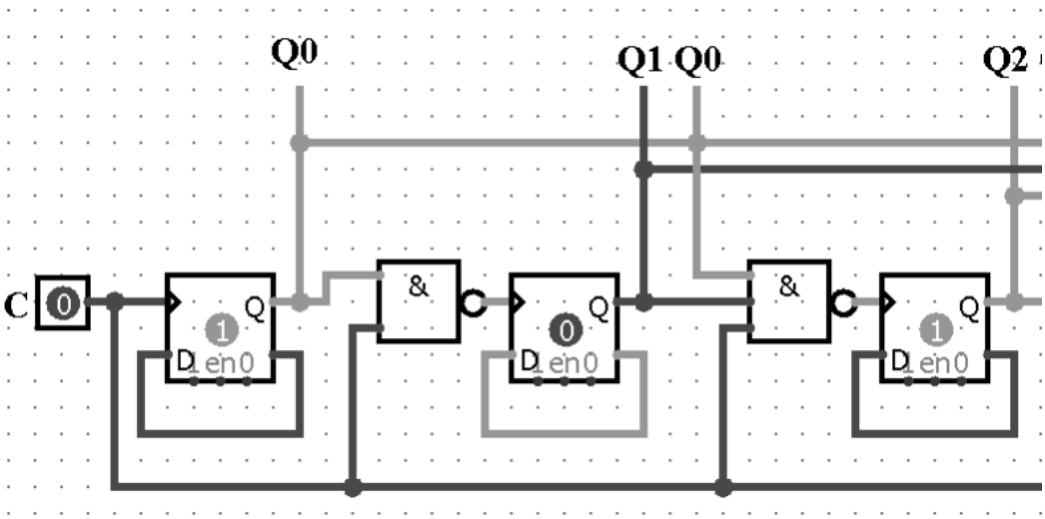



Рис. 81. Схема лічильника з паралельним переносом на D-тригерах

Час установки лічильника не залежить від розрядності і дорівнює

$t_{з.к} + t$, де $t_{з.к}$ – затримка сигналу кон'юнктором,

$t$ – затримка поширення імпульсу в тригері.

Максимальна частота в такому тригері рівна:

$F_{max} \leq \frac{1}{t_{з.к}+t}$.

На час установки кон'юнктор не впливає, але впливає на частоту, оскільки повинен пройти час після установки тригера для його перемикання в новий стан. Лічильник працює швидше, і всі значення виходів змінюються одночасно (синхронно), тому такий лічильник називають синхронним.

Труднощі реалізації багаторозрядних паралельних лічильників пов'язані з ростом числа входів у кон'юнкторів по мірі збільшення розрядності лічильника. Другий обмежуючий фактор – ріст навантаження на виходи тригерів по мірі збільшення числа розрядів лічильника. Застосування різних схем буферного типу для подолання зазначених обмежень, звичайно, небажано, оскільки знижується швидкодія лічильника.

У паралельних лічильниках всі розряди перемикаються одночасно, тому їхній структурі властиві часові змагання сигналів. При використанні двоступінчастих тригерів ці недоліки виключаються.

### 11.4. Реверсивні лічильники

Реверсивні лічильники змінюють напрямок рахунку під впливом керуючого сигналу чи при зміні точки подачі лічильних сигналів. У першому випадку схема має керуючий і лічильний входи, у другому – два лічильних входи.

Найбільш розповсюджений спосіб побудови реверсивних лічильників – переключення міжрозрядних зв'язків. Лічильники прямого і зворотного рахунку відрізняються лише точкою знімання сигналу, який подається з попереднього розряду на наступний. Якщо керуючий сигнал перебудовує міжрозрядні зв'язки, переносячи точку знімання сигналу з одного виходу тригера на інший, то реалізується схема реверсивного лічильника.

Послідовний лічильник перетвориться в реверсивний шляхом введення в його структуру елементів реверса (рис. 82). У зв'язку з появою додаткових затримок введення реверса знижується швидкодія лічильника.

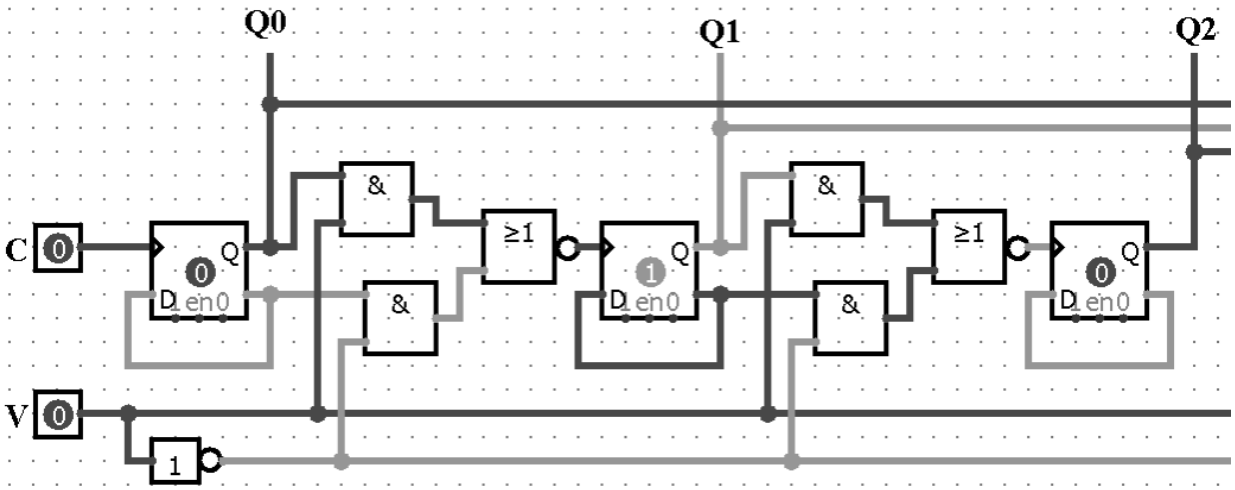



Рис. 82. Схема реверсивного лічильника на D-тригерах

### 11.5. Лічильник з довільним коефіцієнтом рахунку

Часто потрібні лічильники із числом стійких станів, відмінних від $2^n$. Наприклад, в електронних годинниках є мікросхеми з коефіцієнтами рахунку 6 (десятки хвилин), 10 (одиниці хвилин), 7 (дні тижня), 24 (години). Для побудови лічильника з $K_p \neq 2^n$ можна використовувати пристрій з $n$-тригерів, для якого виконується умова $2^n > K_p$. Очевидно, такий лічильник має зайві стійкі стани ($2^n - K_p$). Виключити ці непотрібні стани можна використанням зворотних зв'язків, по ланцюгах яких лічильник перемикається в нульовий стан у тому такті роботи, у якому він дораховує до числа $K_p$.

Для лічильника з $K_p=10$ потрібні чотири тригери (тому що $2^3<10<2^4$). Лічильник повинен мати десять стійких станів (0, ..., 9). У тому такті, коли він повинен був би перейти в одинадцятий стійкий стан (число 10), його необхідно встановити у вихідний нульовий стан. Для такого лічильника можна використовувати будь-який чотирирозрядний лічильник зі зворотнім зв'язком з виходів, що відповідають числу 10 (тобто 2 і 8), на входи установки лічильника в 0. На самому початку одинадцятого стану (число 10) на обох

входах елемента І мікросхеми з’являються логічні 1, що виробляють сигнал скидання всіх тригерів лічильника в нульовий стан (рис. 83).

Розглянутий лічильник є двійковим еквівалентом рахункової декади, що представляє будь-яку десяткову цифру її двійковим кодом. Тому такий лічильник називають двійково-десятковим, а його вихідний код – двійково-десятковим кодом (або кодом 8421).

Якщо двійково-десятковий лічильник призначений для роботи в системах, де потрібна візуальна інформація про число підрахованих імпульсів (наприклад, усілякі цифрові вимірювальні прилади), то після лічильника ставиться перетворювач двійково-десяткового коду в код семисегментного індикатора.

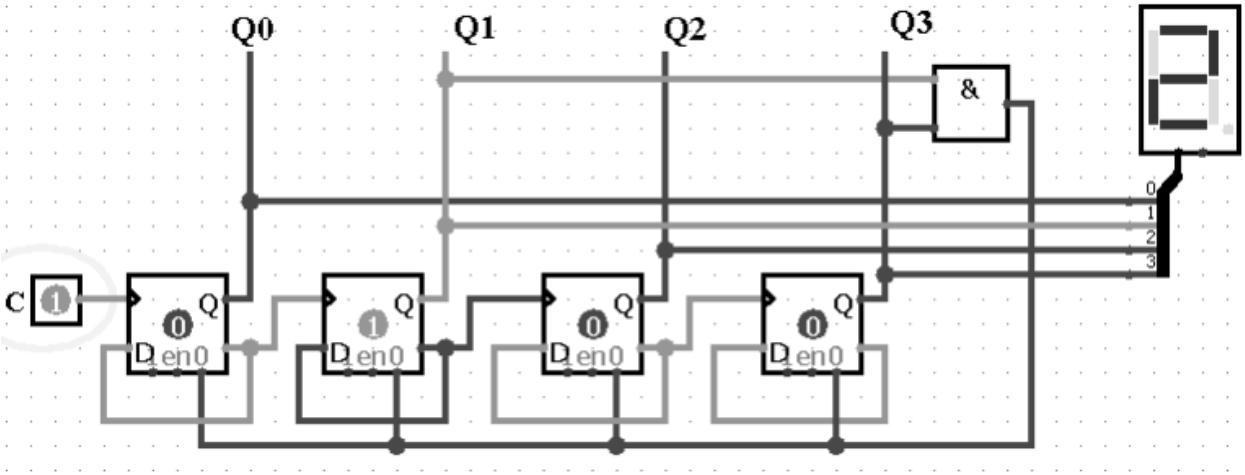



Рис. 83. Схема сумуючого лічильника з послідовним переносом з коефіцієнтом рахунку 10 на D-тригерах

### 11.6. Лічильник з попередньою установкою

Лічильник з попередньою установкою може встановлюватися в початковий стан, рівний будь-якому числу від 0 до $K_p$-1. Ця операція здійснюється паралельним записом у лічильник коду необхідного числа. Рахунок (додавання або віднімання) буде починатися вже не з нуля, а із встановленого числа (рис. 84). Такий режим роботи лічильника необхідний,

наприклад, у керуючому пристрої ЕОМ при утворенні послідовності адрес команд із заданої початкової адреси.

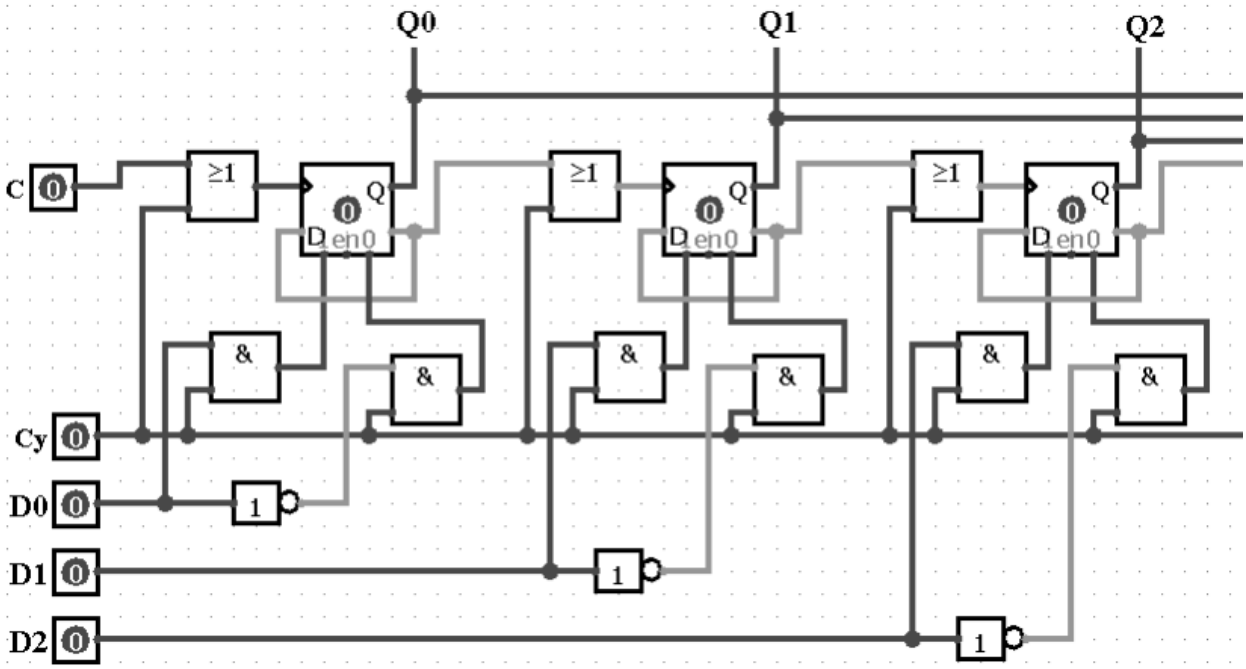



Рис. 84. Схема лічильника з послідовним переносом з попередньою установкою на D-тригерах

### 11.7. Кільцеві лічильники

Розглянуті вище лічильники належать до виду лічильників з позиційним кодуванням. Крім того, для побудови лічильників $K \neq 2^n$ часто використовуються, так звані, кільцеві лічильники імпульсів. Ці лічильники будуються на основі регістру зсуву, в якому сигнал з виходу регістра подається на його ж вхід. Вони мають назву – *кільцеві лічильники імпульсів*.

Якщо в один з розрядів ввести логічну одиницю, при тому що інші розряди встановлені в 0, то ця одиниця при надходженні кожного тактового імпульсу буде пересуватися від розряду до розряду у циклі, довжина якого буде дорівнювати кількості тригерів. При цьому, місцезнаходження цієї 1 у лічильнику однозначно визначає кількість імпульсів, які надійшли до схеми. Це надає можливість використовувати такий лічильник без дешифраторів, що відповідно зменшує апаратні витрати і збільшує швидкодію системи.

Недоліком таких лічильників є те, що для таких схем коефіцієнт лічби $K=n$, тобто кількості тригерів. Схема такого лічильника подана на рис. 85.

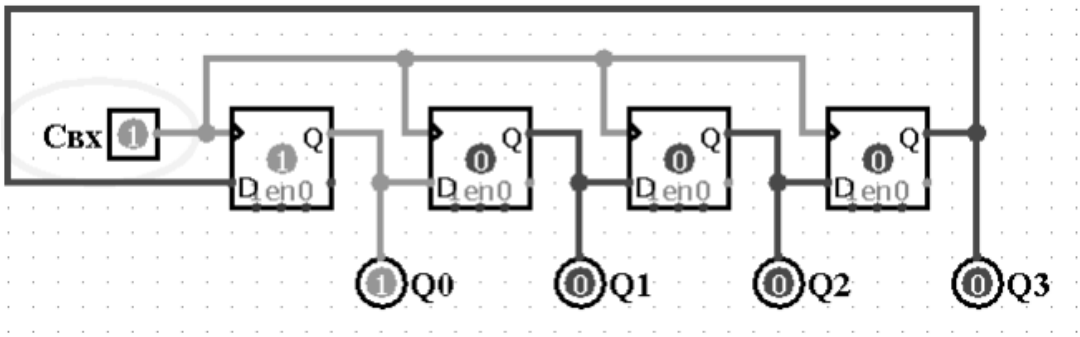



Рис. 85. Схема кільцевого лічильника на D-тригерах

Ще однією перевагою кільцевого лічильника є те, що в процесі рахунку завжди перемикається в одиничний стан лише один тригер, що забезпечує мінімальне значення затримки поширення імпульсу та спрощується побудова схеми контролю лічильника.

Коефіцієнт рахунку кільцевого лічильника можна збільшити до значення $K=2n$, якщо вхід одного з тригерів з'єднати з інверсним виходом попереднього. Оскільки лічильник є кільцевим, то не має значення між якими розрядами організовано цей зв'язок. Такі пристрої мають назву – лічильники Джонсона (рис. 86).

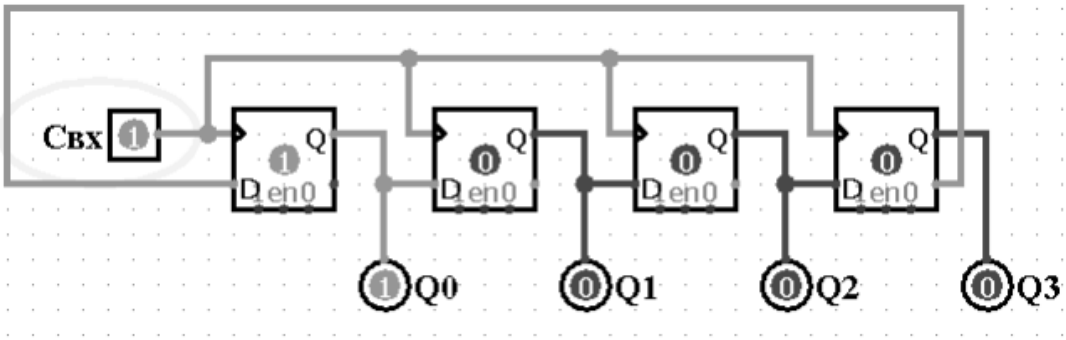



Рис. 86. Схема лічильника Джонса на D-тригерах

В лічильниках Джонса, в процесі рахунку спочатку рухається «хвиля» одиниць, а потім – «хвиля» нулів. Дешифрація станів лічильника Джонсона здійснюється простіше, порівняно з двійковими позиційними лічильниками.

### 11.8. Дільники частоти

Лічильники можуть виконувати функції дільників частоти, тобто пристроїв, що формують із імпульсної послідовності із частотою $f_{вх}$ імпульсну послідовність на виході останнього тригера із частотою $f_{вих}$, в $K_p$ раз меншу вхідної. При такому використанні лічильників немає необхідності знати, яке число в ньому записано в даний момент; тому дільники можуть не мати всіх проміжних виходів. Це значно спрощує їхню схему й конструкцію.

**Завдання**

1. Побудувати схему сумуючого лічильника з послідовним переносом з дійсним порядком рахунку 8:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на T- тригерах (тригер взяти з бібліотеки Logisim).

2. Побудувати схему сумуючого лічильника з послідовним переносом з дійсним порядком рахунку 16:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на T- тригерах (тригер взяти з бібліотеки Logisim).

3. Побудувати схему віднімаючого лічильника з послідовним переносом з дійсним порядком рахунку 8:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на T- тригерах (тригер взяти з бібліотеки Logisim).

4. Побудувати схему віднімаючого лічильника з послідовним переносом з дійсним порядком рахунку 16:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на T- тригерах (тригер взяти з бібліотеки Logisim).

5. Побудувати схему сумуючого лічильника з паралельним переносом з дійсним порядком рахунку 8:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на T- тригерах (тригер взяти з бібліотеки Logisim).

6. Побудувати схему сумуючого лічильника з паралельним переносом з дійсним порядком рахунку 16:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на T- тригерах (тригер взяти з бібліотеки Logisim).

7. Побудувати схему віднімаючого лічильника з паралельним переносом з дійсним порядком рахунку 8:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на T- тригерах (тригер взяти з бібліотеки Logisim).

8. Побудувати схему віднімаючого лічильника з паралельним переносом з дійсним порядком рахунку 16:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на T- тригерах (тригер взяти з бібліотеки Logisim).

9. Побудувати схему реверсивного лічильника який змінює порядок рахунку під впливом керуючого сигналу з дійсним порядком рахунку 8:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на T- тригерах (тригер взяти з бібліотеки Logisim).

10. Побудувати схему реверсивного лічильника який змінює порядок рахунку під впливом керуючого сигналу з дійсним порядком рахунку 16:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на T- тригерах (тригер взяти з бібліотеки Logisim).

11. Побудувати схему сумуючого лічильника з послідовним переносом з коефіцієнтом рахунку 10:

*a)* на D-тригерах(тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на T- тригерах (тригер взяти з бібліотеки Logisim).

12. Побудувати схему віднімаючого лічильника з послідовним переносом з коефіцієнтом рахунку 10:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на T- тригерах (тригер взяти з бібліотеки Logisim).

13. Побудувати схему сумуючого лічильника з послідовним переносом з коефіцієнтом рахунку 18:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на T- тригерах (тригер взяти з бібліотеки Logisim).

14. Побудувати схему віднімаючого лічильника з послідовним переносом з коефіцієнтом рахунку 18:

*a)* на D-тригерах(тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах(тригер взяти з бібліотеки Logisim);

*c)* на T- тригерах (тригер взяти з бібліотеки Logisim).

15. Побудувати схему сумуючого лічильника з послідовним переносом з попередньою установкою з коефіцієнтом рахунку 16:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на T- тригерах (тригер взяти з бібліотеки Logisim).

16. Побудувати схему віднімаючого лічильника з послідовним переносом з попередньою установкою з коефіцієнтом рахунку 16:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на T- тригерах (тригер взяти з бібліотеки Logisim).

17. Побудувати схему сумуючого лічильника з послідовним переносом з коефіцієнтом рахунку 18 з попередньою установкою в значення 3:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на T- тригерах (тригер взяти з бібліотеки Logisim).

18. Побудувати схему віднімаючого лічильника з послідовним переносом з коефіцієнтом рахунку 18 з попередньою установкою в значення 3:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на T- тригерах (тригер взяти з бібліотеки Logisim).

20. Побудувати схему кільцевого лічильника з коефіцієнтом рахунку 10:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на T- тригерах (тригер взяти з бібліотеки Logisim).

21. Побудувати схему лічильника Джонса з коефіцієнтом рахунку 14:

*a)* на D-тригерах (тригер взяти з бібліотеки Logisim);

*b)* на JK-тригерах (тригер взяти з бібліотеки Logisim);

*c)* на T- тригерах (тригер взяти з бібліотеки Logisim).

## Питання для самоконтролю

1. Які види лічильників ви знаєте?
2. Де використовуються лічильники?
3. Скільки JK-тригерів потрібно використати для побудови лічильника з дійсним порядком рахунку 16? Чому?
4. Скільки потрібно тригерів для побудови лічильника з коефіцієнтом рахунку рівним 10? Як працює такий лічильник?
5. Що таке кільцевий лічильник? Як він працює?
6. Як працює подільник частоти? Для чого він призначений?

# 12. РОБОТА З ПАМ’ЯТТЮ

Для зберігання великих обсягів даних в складних цифрових пристроях використовують ОЗП (оперативно запамятовуючий пристрій, англ. RAM, *random access memory*) і ПЗП (постійний запам’ятовуючий пристрій, англ. ROM, *read-only memory*). Головна відмінність між ними – ОЗП дозволяє динамічно (оперативно) змінювати своє значення, а при відключенні живлення збережена в ньому інформація втрачається, а ПЗП зберігає інформацію і після відключення живлення, але ця інформація статична (записується один раз і не може бути оперативно перезаписана).

## 12.1. Пам’ять ОЗП

Фізично комірка пам’яті призначена для зберігання одного біта у складі ОЗП може бути тригером на логічних елементах (пам’ять статичного типу, англ. SRAM ), але частіше – елементарним пристроєм з одного транзистора і одного конденсатора (пам’ять динамічного типу, англ. DRAM).

Пам’ять динамічного типу компактніша, але повільніша через те, що час заряджання або розряджання конденсатора більший, ніж переривання логічних елементів у складі тригера.

Логічна структура обох видів пам’яті являє собою лінійну послідовність окремих комірок пам’яті фіксованої розрядності. Як правило, розрядність комірки (цей параметр пам’яті називається «розрядність даних» ) є степенем двійки (8 біт, 16 бітів, 32 біта, і т.д.), однак можуть бути винатки. Довжина послідовності комірок визначається розрядністю адреси запам’ятовуючого пристрою: якщо розрядність адреси дорівнює $n$, то кількість комірок пам’яті в пристрої рівна $2^n$.

Найпростіший ОЗП має багаторозрядний вхід адреси ($A$), значення на якому визначає номер комірки, до якої ми звертаємося в даний момент; багаторозрядний вхід даних ($D$), на який подається значення для запису у вибрану комірку; багаторозрядний вихід даних ($D$), на який видається

значення, збережене в поточній комірці; і у випадку синхронного ОЗП – тактовий вхід (в Logisim позначається трикутником), при спрацьовуванні якого у вибрану комірку записується нове значення. ОЗП можна подати у вигляді ланцюга регістрів, доступ до яких відбувається за допомогою демультиплексора і мультиплексора (DMX і MUX) (рис. 87).

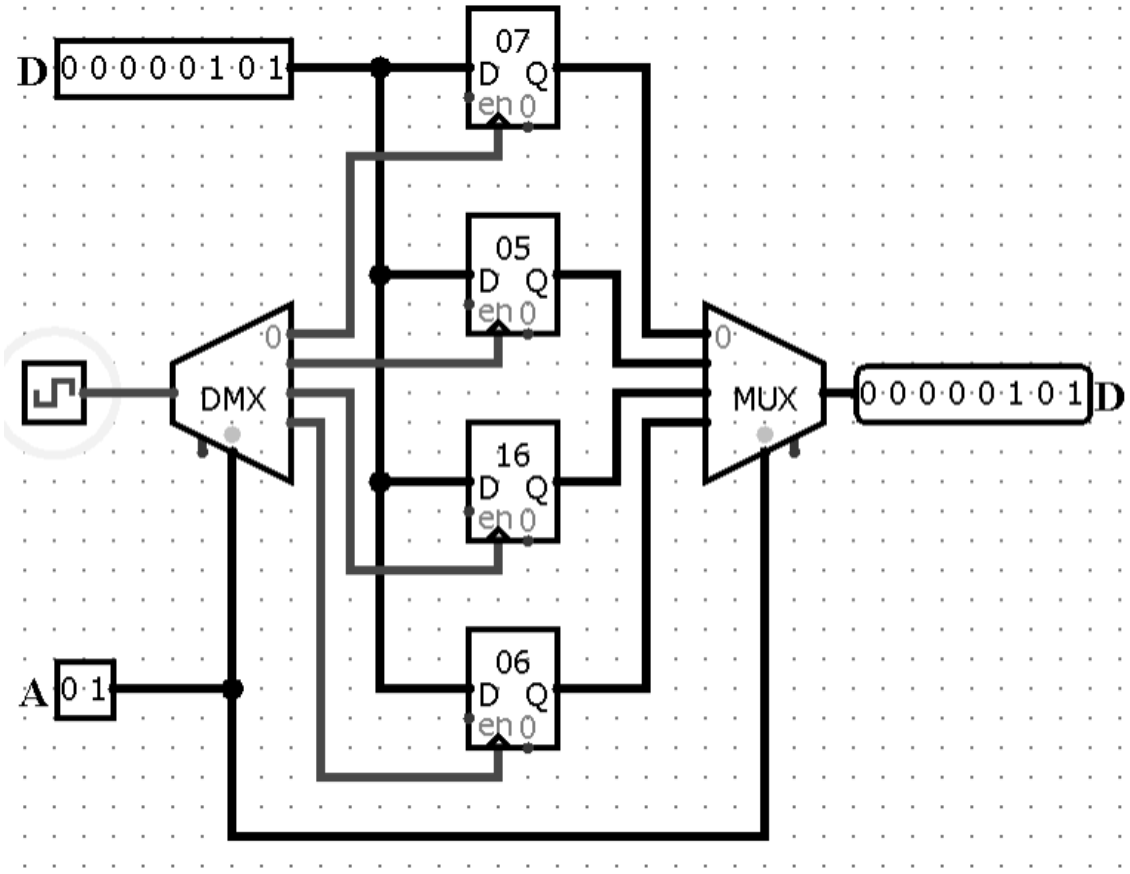



Рис. 87. Найпростіший пристрій ОЗП реалізований на регістрах

У Logisim є спеціальний компонент ОЗП (рис. 88). Для даного компонента передбачено особливий інтерфейс даних, у випадку використання якого ОЗП має єдиний порт для запису і читання даних ( тобто він є одночасно входом і виходом). Для роботи з таким інтерфейсом даних потрібно використовувати компонент буфер керування («Буфер» бібліотеки «Элементы» ).

У випадку використання одного порту для читання і запису, в Logisim можна вибирати між синхронним і асинхронним ОЗП. Значна частина промислових модулів пам’яті – асинхронні, з єдиним портом.

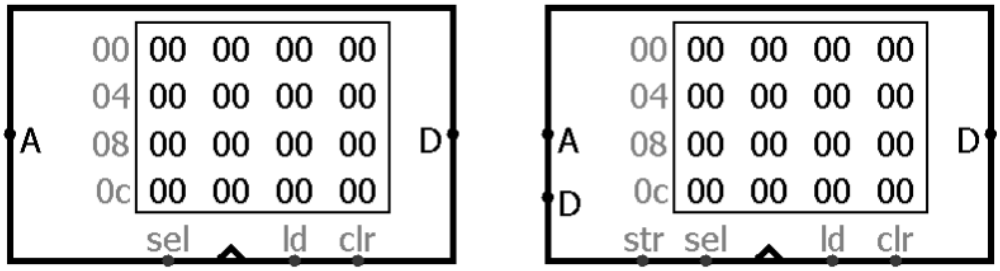



Рис. 88. Пристрій ОЗП в Logisim

Компонент ОЗП, найскладніший компонент вбудованої бібліотеки Logisim. Він зберігає до 16777216 значень (задається в атрибуті «Розрядность адреса»), кожен з яких може включати до 32 бітів (задається в атрибуті «Розрядность данных»). Схема може зчитувати і записувати значення в ОЗП. Крім того, користувач може інтерактивно змінити окремі значення за допомогою інструменту «Натиснення», або змінити весь вміст через контекстне меню.

Поточні значення відображаються в компонентах. Їх адреси подані сірим кольором зліва від області відображення. Значення всередині подані в шістнадцятковому виді.

*Основні контакти*

*A* – вхід , розрядність відповідає атрибуту розрядності адреси.

*D* – вхід/вихід або вихід, розрядність відповідає атрибуту розрядності даних.

Якщо на входах *sel* і *ld* «1» або плаваюче значення, то компонент ОЗП надсилає на цей порт значення за обраною в даний момент адресою.

*str* – вхід, розрядність дорівнює «1». Цей вхід представлений, якщо для атрибута «Интерфейс данных» вибрано «Раздельные порты чтения і записи». Коли на ньому «1» або плаваюче значення, тактовий імпульс призведе до

запису в пам’ять даних, на вході *A* (за умови, що на вході *sel* теж «1» або плаваюче значення).

*sel* – вхід, розрядність якого дорівнює «1». Цей вхід вмикає або вимикає весь модуль ОЗП, залежно від того, 1/плаваюче значення на ньому чи 0. Це вхід призначений в першу чергу для ситуацій, коли є кілька модулів пам’яті тільки один з яких може бути включений в якийсь момент.

Використовуючи компонент ОЗП («ОЗУ» з бібліотеки «Память»), рисунок 87 можемо зобразити більш компактніше (рис. 89).

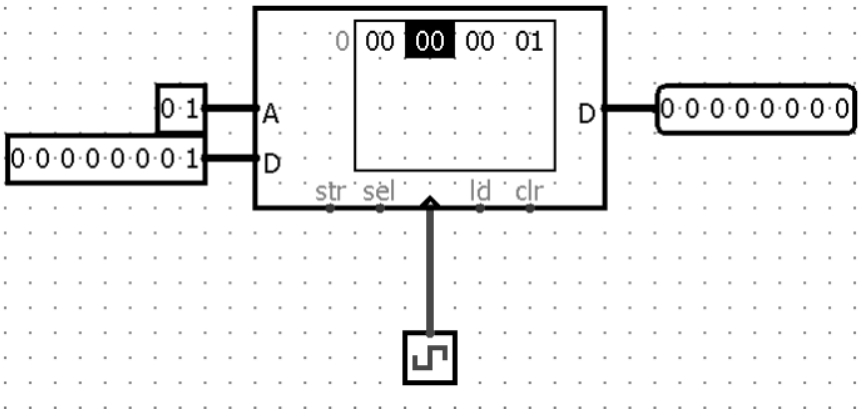



Рис. 89. Логічна схема, яка демонструє роботу ОЗП з двома входами (*A*, *D*)

### 12.2. Гарвардська архітектура

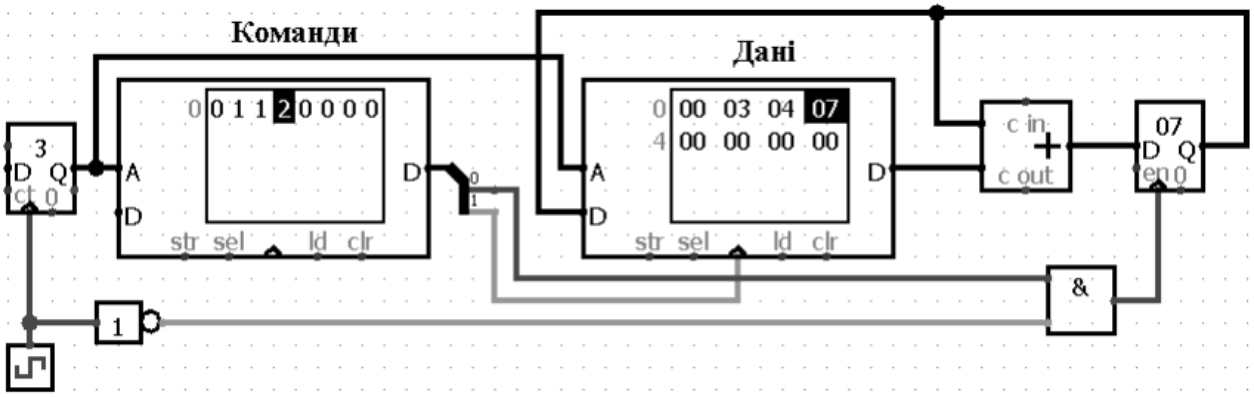
Нехай, в ОЗП зберігається набір чисел, а нам потрібно додати всі числа і зберегти результат (у вільну комірку). Будемо зберігати команди («додати» і «зберегти») в одному ОЗП, а оброблювані числа – в іншому (рис. 90, *а*). Такий спосіб зберігання є ознакою Гарвардської архітектури.

Команда «1» буде записувати число з суматора в акумулятор, команда «2» буде записувати число з акумулятора в другий ОЗП.

Припустимо, в ОЗП зберігаються кілька масивів чисел. Тобто нам необхідно завантажувати числа з ОЗУ в акумулятор, завантажувати числа з суматора в акумулятор, зберігати числа з акумулятора в ОЗП. Тоді потрібно використовувати для вибірки даних з ОЗП мультиплексор MUX. Нова

інструкція «5» по передньому фронту тактового генератора перемикає мультиплексор MUX, а по задньому фронту (по спаду) проводить запис в акумулятор (рис. 90, *б*).

*а)*




*б)*

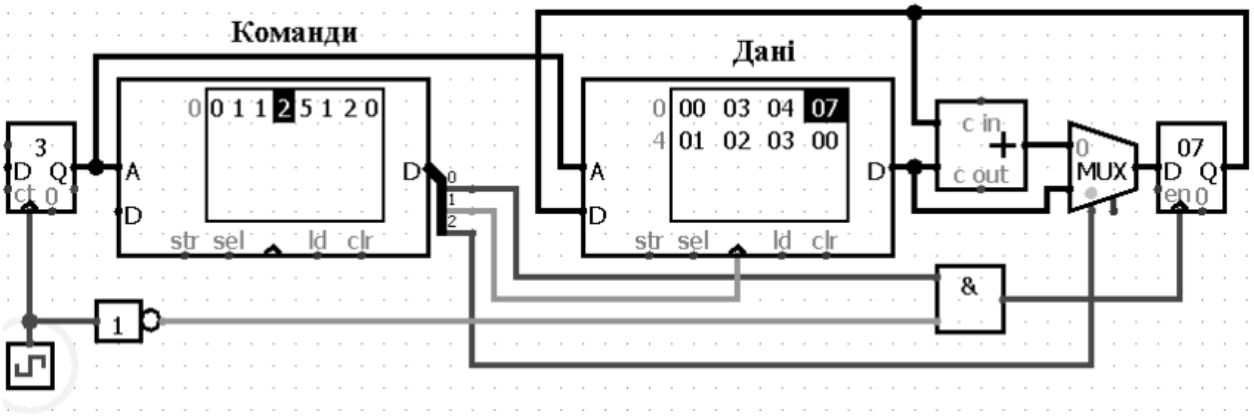



Рис.90. Організація додавання двох чисел, які збережені в ОЗП, використовуючи Гарвардську архітектуру

### 12.3. Архітектура фон Неймана

Далі, будемо зберігати інструкції і дані в одному ОЗП. Такий спосіб зберігання є ознакою архітектури фон Неймана (рис. 91).

У перших чотирьох розрядах восьмирозрядних комірок пам’яті зберігатиметься команда, у других чотирьох розрядах буде зберігатися адреса (пряма адресація). Будемо завантажувати адресу і команду в окремі регістри, а потім за допомогою мультиплексора переходити за збереженою адресою.

Для запису даних в ОЗП будемо спочатку завантажувати дані в тимчасовий регістр Temp (інакше в ОЗП запишуться інструкції, а не дані), а потім в акумулятор Acc. Запис даних в Temp і Acc будемо виконуватися по задньому фронту.

Будемо додавати, наприклад, числа 18 і 19, які лежать в комірках 8 і 9, а зберігати результат в комірку з адресою 2*a*.

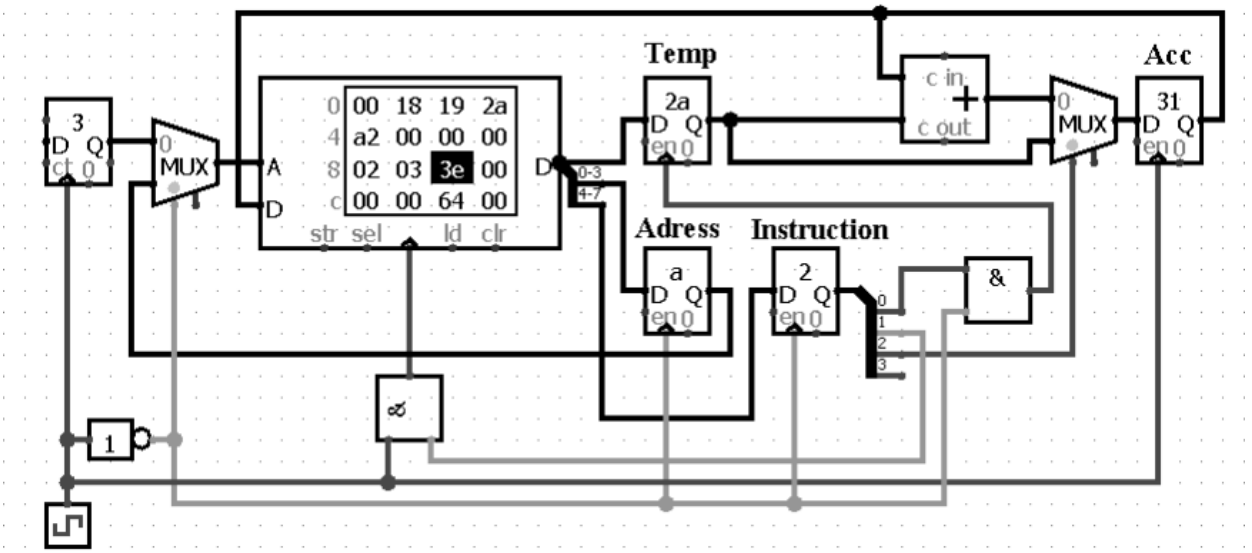



Рис.91. Організація додавання двох чисел, які збережені в ОЗП, використовуючи архітектуру фон Неймана

Додамо також можливість здійснювати безумовні переходи. Реалізуємо пристрій, в якому буде виконуватися лише одна команда – команда безумовного переходу. Для цього на лічильник, що генерує адресу, будемо відправляти чотири молодших розряди (власне, адресу), а на порт завантаження подаватимемо команду, що зберігається в старших розрядах комірки пам’яті (рис. 92). Наприклад, для стрибка в комірку з адресою 2 використовується команда 82.

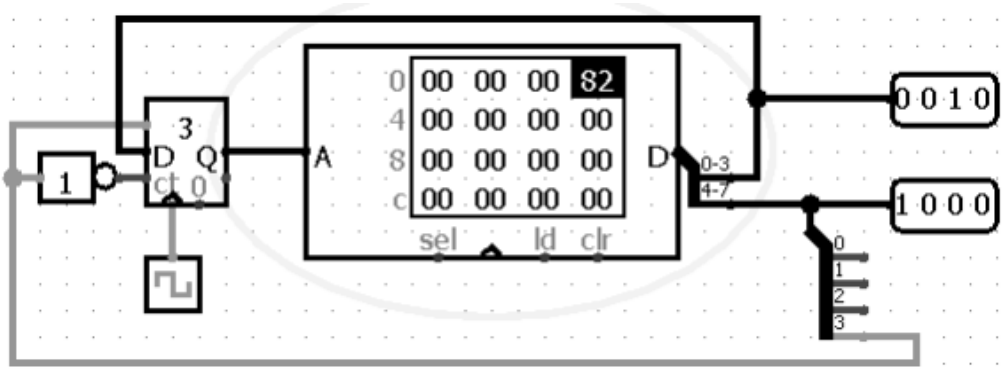



Рис. 92. Команда безумовного переходу

Додамо команду безумовного переходу в попередню схему (рис. 93).

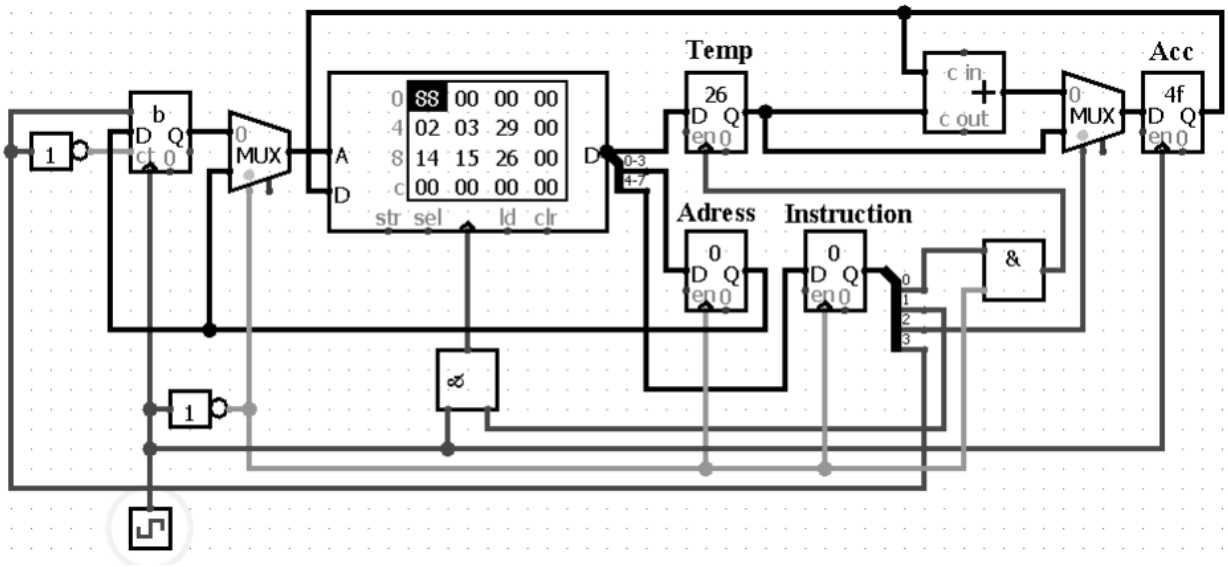



Рис.93. Організація додавання двох чисел, які збережені в ОЗП, використовуючи архітектуру фон Неймана з командою безумовного переходу

**Завдання**

1. Побудувати пристрій ОЗП реалізований на регістрах, для зберігання:

*a)* восьми восьмибітних слів;

*b)* шести восьмибітних слів;

*c)* чотирьох восьмибітних слів;

*d)* восьми шестибітних слів;

*e)* шести шестибітних слів;

*f)* чотирьох шестибітних слів.

2. Побудувати логічну схему з використанням елемента ОЗП з бібліотеки «Память» в Logisim з двома входами (*A*, *D*) та пояснити принцип роботи.
3. Побудувати логічну схему з використанням елемента ОЗП з бібліотеки «Память» в Logisim з одним входом (*A*) та пояснити принцип роботи.
4. Нехай дані збережені в одному ОЗП, а команди збережені в іншому ОЗП. Побудувати логічну схему, яка виконує над двома числами арифметичну операцію:
   *a)* додавання;
   *b)* віднімання;
   *c)* множення.
5. Нехай дані і а команди збережені в одному ОЗП. Побудувати логічну схему, яка виконує над двома числами арифметичну операцію:
   *a)* додавання;
   *b)* віднімання;
   *c)* множення;

**Питання для самоконтролю**

1. Поясніть принцип роботи пристрою ОЗП в Logisim?
2. Які існують види пам’яті. Розшифруйте абревіатуру RAM?
3. Що таке буфер управління?
4. Яка різниця між синхронним і асинхронним входом і виходом в логічному пристрої ОЗП?

## СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ


1. Булатецький В.В. Логічні елементи та основні операційні вузли: Лабораторний практикум / В. В. Булатецький, Л. В. Булатецька. – Луцьк., Вежа, 2007. – 68 с. [Електронний ресурс] — Режим доступу : http://evnuir.vnu.edu.ua/handle/123456789/14550
2. Logisim [Електронний ресурс] — Режим доступу : http://www.cburch.com/logisim/index_ru.html
3. Карлащук В.И. Электронная лаборатория на IBM PC. –М.: Солон-Р, 1999.- 510 с.
4. Дмитрова М. Пунджев В. 33 схемы с логическими элементами И-НЕ. – М.: Радио и связь. 1987, - 110 с.
5. Самофалов К.Г., Корнейчук В.И., Тарасенко В.П. Цифровые ЭВМ. - К: Вища школа., 1989. - 420 с.
6. Самофалов К.Г., Корнейчук В.И., Тарасенко В.П. Цифровые ЭВМ (Практикум). - К: Вища школа., 1989. - 420 с.
7. Мальцева Л.А. Фромберг Э.М. Ямпольский В.С. Основы цифровой техники. – Радио и связь, 1986. – 125 с.
8. Микросхемы и их применение. Справочное пособие. // Батушев В.А., Вениаминов В. Н., Ковалев В.Г. и др. – М.: Радио и связь, 1984, - 271 с.
9. Ямпольський В.С. Основы автоматики и электронно-вычислительной техники. – М., Просвещение, 1991. – 223с.
10. Дебновецкий С.В. “Основы автоматизированного проектирования электронных приборов.” Киев, Вища школа, 1987г.
11. Dr. Karl Burch. [Електронний ресурс] — Режим доступу : http://ozark.hendrix.edu/~burch/
12. Logisim (Русский). [Електронний ресурс] — Режим доступу : http://ozark.hendrix.edu/~burch/logisim/ru/index.html
13. Burch K. Logisim: A graphical system for logic circuit design and simulation. / Carl Burch // Jornal on Educational resources in computing (ACM). — 2002. — Vol 2, № 1. — P. 5–16.
14. Проектирование процессора (CPU Design) Logisim [Електронний ресурс] — Режим доступу : https://www.pvsm.ru/diy-ili-sdelaj-sam/262720
15. Проектирование процессора Logisim [Електронний ресурс] — Режим доступу : https://habr.com/ru/post/406157/

**УДК 004.655**

*Електронне мережне навчальне видання*

В. В. Булатецький, Л. В. Булатецька, Собчук О. М.

# АЛГЕБРА ЛОГІКИ ТА ПРОЕКТУВАННЯ ОСНОВНИХ ОПЕРАЦІЙНИХ ВУЗЛІВ

Навчальний посібник

Видання друкується в авторській редакції
Гарн. Times. Обсяг 2,64 ум. друк. арк.