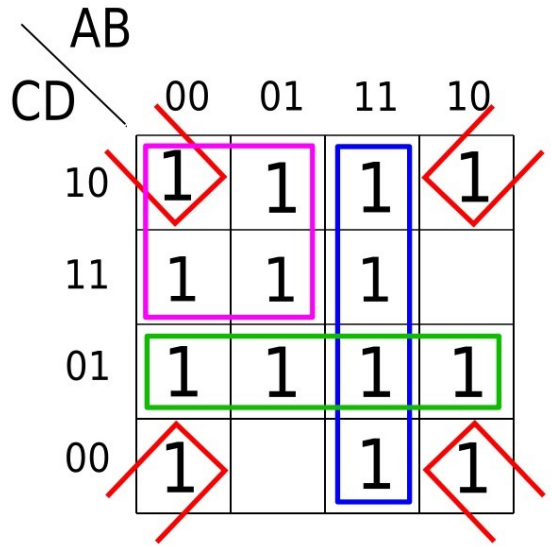
Й. Й. Білинський, П. М. Ратушний
А. О. Мельничук

ЦИФРОВА СХЕМОТЕХНІКА
ЧАСТИНА 2
ЕЛЕКТРОННІ ПРИСТРОЇ І СИСТЕМИ



ЦИФРОВА СХЕМОТЕХНІКА ЧАСТИНА 2 ЕЛЕКТРОННІ ПРИСТРОЇ І СИСТЕМИ

Міністерство освіти і науки України
Вінницький національний технічний університет

Й. Й. Білинський, П. М. Ратушний, А. О. Мельничук

ЦИФРОВА СХЕМОТЕХНІКА

ЧАСТИНА 2

Електронні пристрої і системи

Навчальний посібник

Вінниця
ВНТУ
2017

УДК 621.38.061 (075.8)
ББК 32.973.2
Б61

Рекомендовано до друку Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 8 від 27.03.2014 р.)

Рецензенти:
В. П. Манойлов, доктор технічних наук, професор
М. А. Філинюк, доктор технічних наук, професор
О. М. Шинкарук, доктор технічних наук, професор

Білинський, Й. Й.

Б61 Цифрова схемотехніка. Частина 2. Електронні пристрої і системи: навчальний посібник / Й. Й. Білинський, П. М. Ратушний, А. О. Мельничук. – Вінниця : ВНТУ, 2017. – 171 с.

У навчальному посібнику розглядаються елементи цифрових пристроїв та способи синтезу комбінаційних схем та автоматів жорсткої логіки на їх основі; різні структури пам’яті цифрових пристроїв; основи програмованих логічних схем; та принципи побудови спеціалізованих арифметико-логічних пристроїв для додавання, віднімання, множення та ділення чисел.

Метою навчального посібника є освоєння студентами методів та алгоритмів синтезу відповідних цифрових пристроїв.

УДК 621.38. 061 (075.8)
ББК 32.973.2

ЗМІСТ

ВСТУП

Курс цифрової схемотехніки є одним із основних для підготовки спеціалістів напрямків, пов’язаних з електронікою та електронними цифровими пристроями.

Всі різноманітні засоби цифрової техніки: ЕОМ, мікропроцесорні системи вимірювань та автоматизації технологічних процесів, цифровий зв'язок і телебачення будуються на елементній базі, до складу якої входять різні за рівнем складності цифрові мікросхеми та пристрої – від логічних елементів, що виконують найпростіші операції, до складних програмованих кристалів, що містять мільйони логічних елементів.

У навчальному посібнику розглядаються елементи цифрових пристроїв та способи синтезу комбінаційних схем та автоматів жорсткої логіки на їх основі; різні структури пам’яті цифрових пристроїв; основи програмованих логічних схем та принципи побудови спеціалізованих арифметико-логічних пристроїв для додавання, віднімання, множення та ділення чисел.

Метою навчального посібника є освоєння студентами методів та алгоритмів синтезу відповідних цифрових пристроїв.

В посібнику наведено багато прикладів синтезу тих чи інших цифрових схем, що допоможе студентам самостійно синтезувати необхідні цифрові схеми. Даний посібник також буде корисним при розробці курсового проекту з дисципліни «Цифрова схемотехніка».

Посібник адресований для широкого кола читачів, які займаються розробкою цифрових схем і пристроїв, зокрема розрахований на студентів бакалаврського напрямку «Мікро- та наноелектронні прилади та пристрої» і «Електронні прилади та пристрої» та може бути використаним студентами інших спеціальностей.

1 ТРИГЕРИ І ЕЛЕМЕНТИ ЦИФРОВИХ ПРИСТРОЇВ

1.1 Тригери

Тригер – це запам'ятовувальний елемент із двома стійкими станами. Зміна стійких станів відбувається під дією вхідних сигналів. Тригер – елементарна комірка пам'яті для зберігання одного біта інформації (логічний «0» або логічна «1»), він дозволяє записувати, стирати чи зчитувати двійкову інформацію. Також тригер є основою для складніших функціональних вузлів (лічильники, регістри, суматори, модулі пам'яті).

Усі тригери являють собою простий цифровий автомат, що містить елемент пам'яті та керувальну логіку.

Поточні стани тригерів визначаються значеннями виходів (Q та $\bar{Q}$). При додатному кодуванні інформації високий рівень напруги на прямому виході (Q) відображає значення логічної «1», а низький рівень – логічного «0» (Q=0). Значення на виході $\bar{Q}$ будуть відповідно протилежні.

Зміна значень вихідних станів забезпечується зміною значень вхідних керувальних сигналів, що позначаються на схемах латинськими буквами (R, S, T, D, J, K, V та ін.). Узагальнена схема тригера зображена на рис. 1.1.

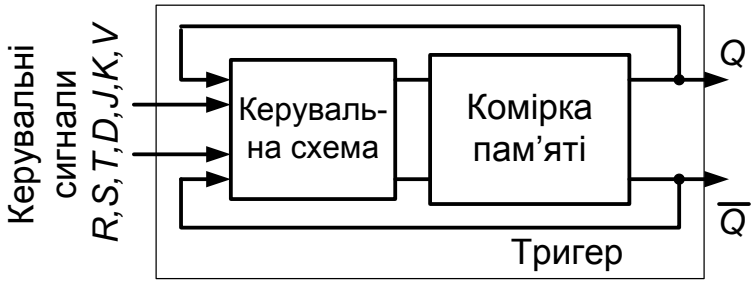


Рисунок 1.1 – Узагальнена структура тригера

Класифікація тригерів

Тригери розрізняються за такими класифікаційними ознаками:

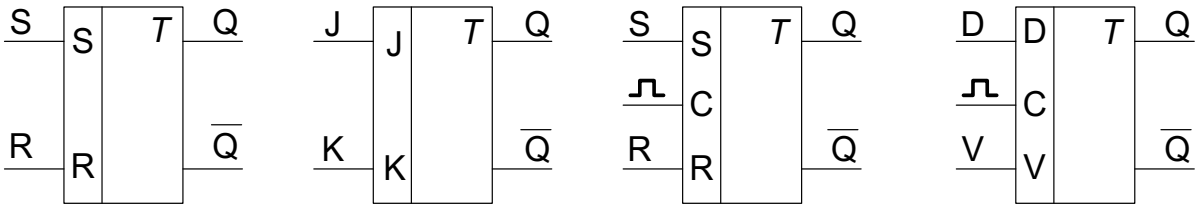
- за логікою функціонування (RS, JK, D, T та ін.);
- за способом запису інформації (синхронні та асинхронні);
- за кількістю ступенів (одно- і двоступінчасті);
- за кількістю тактів синхронізації, необхідних для спрацювання (одно-, дво- і тритактні);
- за логічними елементами, на яких вони виконані (І-НЕ,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

За логікою функціонування виділяють такі тригери:

- з розподіленим встановленням станів «0» і «1» (RS-тригер);
- з одним входом керування (D-тригер);
- із лічильним входом (T-тригер);
- універсальні тригери із розподіленим встановленням станів «0» і «1» (JK-тригери);
- комбіновані тригери (RST, TV-тригери тощо).

Входи тригерів поділяються на інформаційні (R, S, T, D, J, K) та входи керування (C, V). Призначення інформаційних входів полягає у прийманні сигналів інформації, яку необхідно записати. В тригерах зазвичай буває два види керувальних сигналів: синхронізувальний тактовий сигнал: C і сигнал дозволу V.

За способом запису інформації тригери поділяються на несинхронізовані (асинхронні) та синхронізовані (синхронні). У асинхронних тригерів запис інформації (перемикання тригера) відбувається під дією інформаційних сигналів. Ці тригери мають тільки інформаційні входи. У синхронних тригерів запис інформації відбувається під дією дозвільних сигналів синхронізації. Синхронні тригери поділяються на тригери із статичним керуванням записом, із динамічним керуванням записом та двоступінчасті. Синхронні тригери із статичним керуванням записом сприймають інформаційні сигнали увесь час, поки діє сигнал синхронізації. Отже, за час дії сигналу синхронізації тригер може перемикатися кілька разів. У цих тригерів вхід C – статичний. Умовні графічні позначення тригерів показані на рис. 1.2 та 1.3.



а б в г

Рисунок 1.2 – Умовні графічні позначення тригерів, а) – асинхронний RS-тригер, б) – асинхронний JK-тригер, в) – синхронний RS-тригер, г) – синхронний D-тригер із керувальним V-входом

Синхронні тригери із динамічним керуванням записом сприймають тільки ті інформаційні сигнали, які були на інформаційних входах до надходження синхросигналу.

Залежно від кількості тактових сигналів, необхідних для формування нового стану, розрізняють однотактні, двотактні та багатотактні тригери.

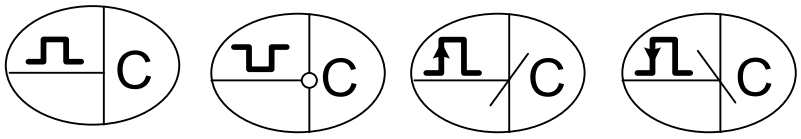


Рисунок 1.3 – Позначення керувальних входів тригерів, а) – прямий статичний, б) – інверсний статичний, в) – прямий динамічний, г) – інверсний динамічний

При керуванні фронтами, дозвіл на запис інформації дається тільки в момент переходу тактового сигналу від «0» до «1». В інші моменти часу

тригер не реагує на вхідні інформаційні сигнали, незалежно від рівня тактового імпульсу.

Динамічні характеристики тригерів

Тригери характеризуються такими динамічними характеристиками.

1. Мінімальна довжина імпульсу t_C на тактовому вході.
2. Мінімальний час попереднього установлення сигналу на інформаційному вході $t_{уст}$.
3. Час відновлення (фіксації) $t_{відн}$ – мінімальний час між наростанням синхросигналу C і спадом інформаційного сигналу D, для асинхронних тригерів $t_{відн}$ – просто тривалість вхідного сигналу.
4. Час переключення тригера $t_{п.т.}$ – часовий інтервал між фронтом вхідного перемикаючого сигналу і фронтом сигналу на виході Q. Мінімальна тривалість синхросигналу на входи тригера визначається максимальним часом переключення тригера $t_C \geq t_{п.т.}$ У двоступінчастому тригері з однотактовою синхронізацією час переключення другого ступеня визначається відповідно до спаду синхроімпульса.

На рисунку 1.4 показано часові діаграми роботи D-тригера.

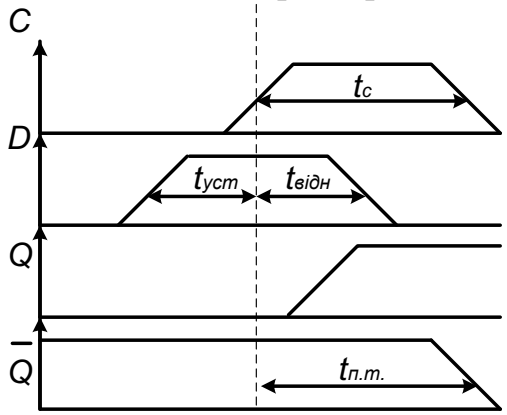


Рисунок 1.4 – Часові діаграми роботи D-тригера

RS-тригер

RS-тригером називається елемент пам’яті, що має розподілені входи керування. Сигнал виходу Q встановлюється в «0» (при високому рівні сигналу («1») на вході скиду «R») і в «1» (при високому рівні сигналу («1») на вході встановлення «S»). В таблиці 1.1 наведено переходи RS-тригера при різних значеннях вхідних сигналів. Тут R, S – керувальні входи тригера, Q – вихід. Q_{t+1} – значення на виході в наступний момент часу, X – невизначений стан (у випадку, якщо на обох вхідних керувальних входах R і S – високий рівень «1»).

Таблиці переходів відповідає карта Карно, де значення функції Q_{t+1} для мінтермів R, S і $\overline{Q}$ та R, S і Q замінені на невизначені коефіцієнти «X». Якщо припустити, що забороненої комбінації R=S=1 не існує, то отримаємо карти Карно на рис. 1.5, б і рис. 1.5, в (у випадку «1» і «0»).

Таблиця 1.1 – Переходи RS-тригера при різних значеннях вхідних сигналів

R	***S***	***Q***	Q_{t+1}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	X
1	1	1	X

Із карт Карно отримаємо логічні рівняння.

У випадку, якщо X – це «1»: $Q_{t+1} = S \vee R \cdot \overline{Q}$.

У випадку, якщо X – це «0»: $Q_{t+1} = \overline{R} \cdot (S \vee Q)$.

Дані логічні вирази визначають новий стан тригера Q_{t+1} залежно від старого стану Q_t і вхідних сигналів *R* і *S*.

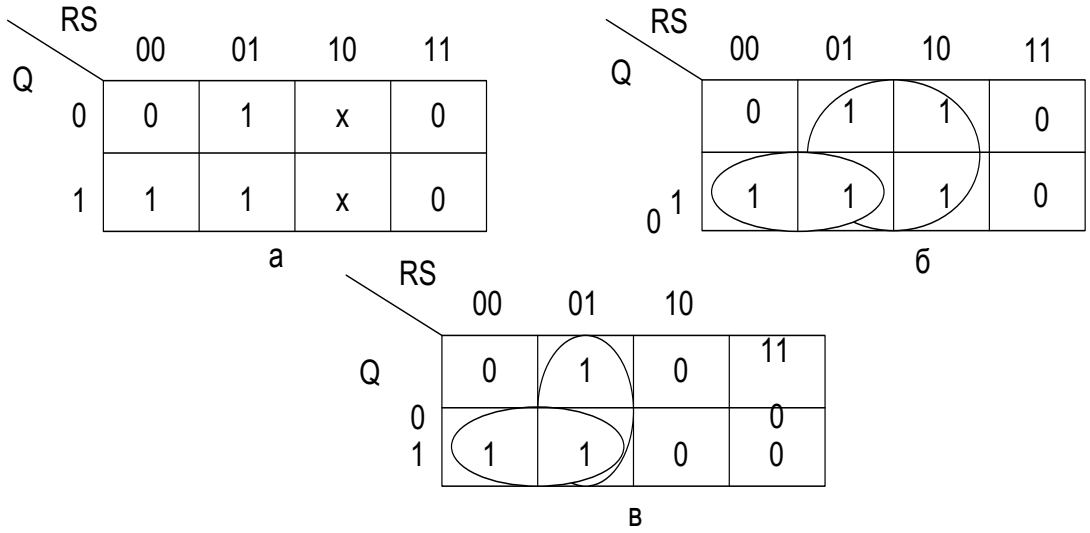


Рисунок 1.5 – Карти Карно для RS-тригера:
а) – карта Карно із невизначеними станами X, б) – карта Карно у випадку, коли X = «1», в) – карта Карно у випадку, коли X = «0»

Асинхронний RS-тригер на логічних елементах «І-НЕ» та «АБО-НЕ».

Перетворивши логічні вирази для реалізації на логічних елементах «І-НЕ», отримаємо:

$$Q_{t+1} = \overline{S \vee \overline{\overline{R} \cdot Q}} = \overline{\overline{S} \cdot \overline{\overline{R} \cdot Q}}. \qquad (1.1)$$

Схема асинхронного RS-тригера на логічних елементах «І-НЕ»

показана на рис. 1.6. Характерною особливістю даного тригера є інверсне керування по інформаційних входах. Аналіз часових діаграм роботи тригера показує, що елементи тригера в схемі не переключаються послідовно. Існує певний інтервал часу, коли на обох виходах встановлюються однакові сигнали $Q = 1$ і $\overline{Q} = 1$.

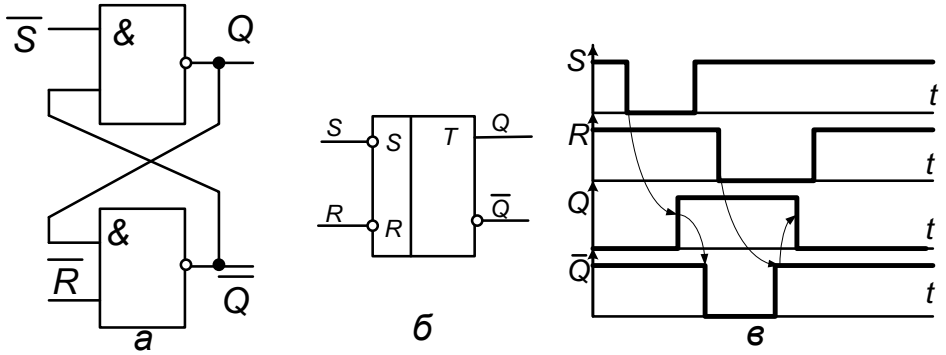


Рисунок 1.6 – Асинхронний RS тригер на логічних елементах «І-НЕ»:
а) – схема логічна принципова, б) – умовне графічне позначення,
в) – часові діаграми роботи тригера

Перетворивши логічні вирази для реалізації на логічних елементах «АБО-НЕ», маємо:

$$Q_{t+1} = \overline{\overline{\overline{R}(S \vee Q)}} = \overline{R \vee (\overline{S \vee Q})}. \qquad (1.2)$$

Схема асинхронного RS-тригера на двох елементах «АБО-НЕ» з логічними зв’язками на основі наведеного вище виразу, показана на рис. 1.7. Аналіз часових діаграм роботи тригера показує, що елементи тригера в схемі не переключаються послідовно. Існує певний інтервал часу, коли на обох виходах встановлюються однакові сигнали $Q = 0$ і $\overline{Q} = 0$.

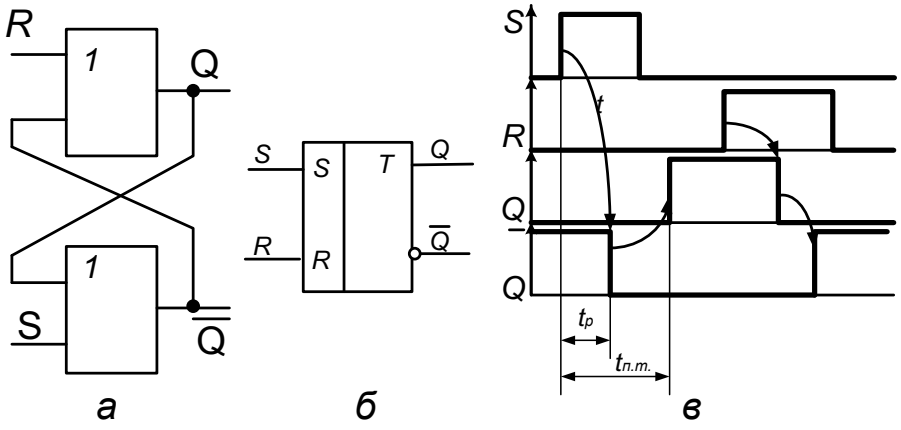


Рисунок 1.7 – Асинхронний RS тригер на логічних елементах «АБО-НЕ»:
а) – схема логічна принципова, б) – умовне графічне позначення,
в) – часові діаграми роботи тригера

Для побудови синхронного тригера на елементах «І-НЕ» потрібно

замінити в логічному виразі (1.3) змінні S і R на сукупності CS і CR, де C – синхросигнал:

$$Q_{t+1} = \overline{\overline{CS}\,\overline{CR}Q}. \qquad (1.3)$$

Схема синхронного RS-тригера на логічних елементах «І-НЕ» показана на рис. 1.8.

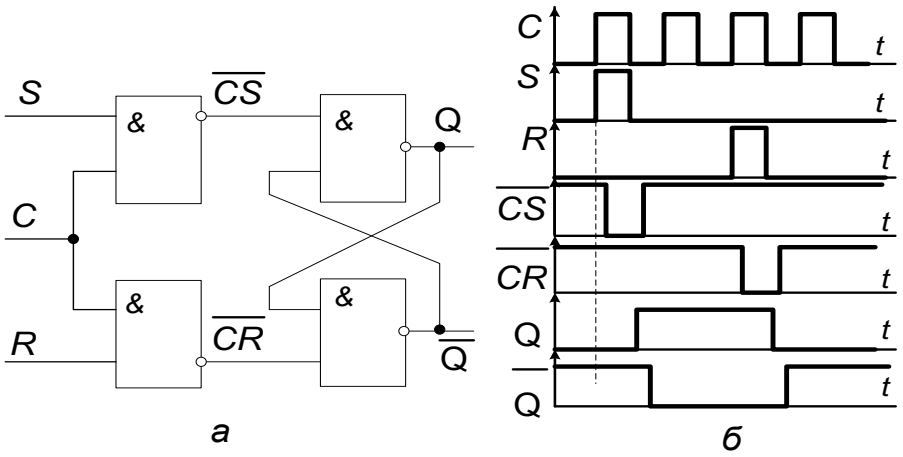


Рисунок 1.8 – Синхронний RS-тригер на логічних елементах «І-НЕ», а) – схема логічна принципова, б) – часові діаграми роботи тригера

Для побудови синхронного RS-тригера на елементах «АБО-НЕ» потрібно замінити в логічному виразі змінні S і R на сполучення $\overline{CS}$ *і* $\overline{CR}$.

$$Q_{t+1} = \overline{\overline{CR} \vee (\overline{\overline{CS}} \vee Q)} = \overline{\overline{\overline{C} \vee \overline{R}} \vee (\overline{\overline{C} \vee \overline{S}} \vee Q)}. \qquad (1.4)$$

Схема синхронного RS-тригера на чотирьох елементах «АБО-НЕ» і його часові діаграми показані на рис. 1.9.

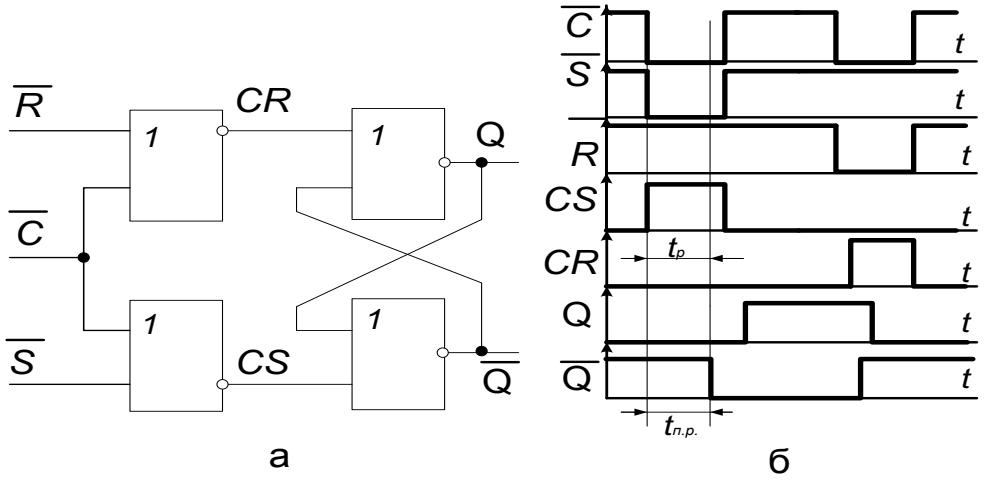


Рисунок 1.9 – а) – синхронний RS-тригер на логічних елементах «АБО-НЕ», б) – його часові діаграми

Двоступінчасті RS-тригери

Двоступінчасті тригери будуються за способом «M-S» і забезпечують сукупність двох процесів – одночасного запису нової інформації і зчитування старої. Під час дії синхроімпульсу C перший ступінь «M» (Master – основний) приймає нову вхідну інформацію, і інший ступінь – «S» (Slave – допоміжний) в цей же час передає у внутрішнє коло стару інформацію. Після закінчення синхроімпульсу C інформація із першого ступеня записується у другий. На рис. 1.10 показано схему двоступінчастого MS-тригера.

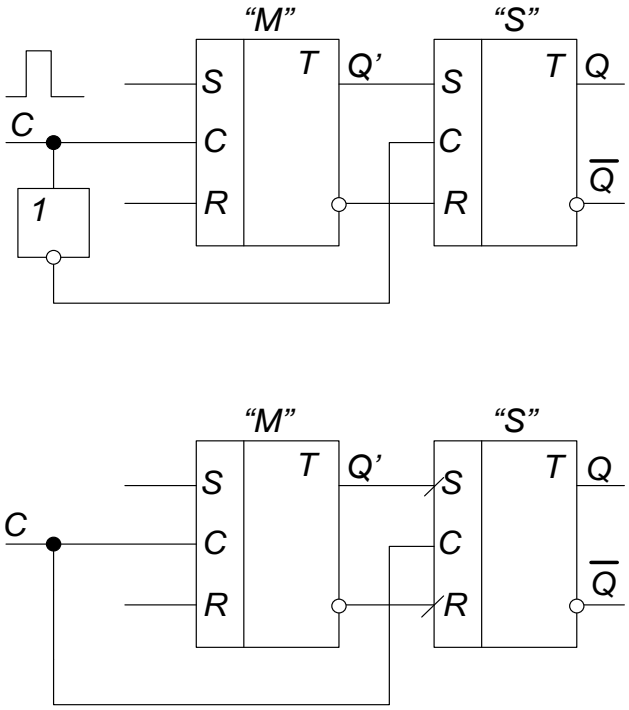


Рисунок 1.10 – Організація зв'язку між ступенями тригера M і S

JK-тригер

Тригером типу JK називається елемент пам'яті із двома стійкими станами та інформаційними входами J (аналог S) і K (аналог R), що забезпечують відповідно роздільне встановлення станів «1» і «0». Він функціонує аналогічно RS-тригеру, але на відміну від останнього, при збіганні сигналів JK=1 переключається в протилежний стан, тобто виконує логічну функцію «сума за модулем 2». Таким чином, JK-тригер не має заборонених комбінацій вхідних сигналів. Тригер типу JK є універсальним, оскільки може виконувати функції RS-тригера (при окремому надходженні сигналів J і K). Нижче наведено рівняння та таблиця переходів JK-тригера:

$$Q_{t+1} = \overline{K_t} Q_t \vee J_t \overline{Q_t}. \qquad (1.5)$$

K_t	J_t	Q_t	Q_{t+1}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

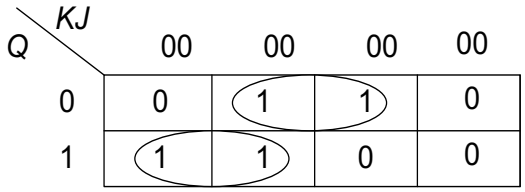


Рисунок 1.11 – Таблиця переходів JK-тригера та карта Карно для мінімізації вихідної функції

Для побудови одноступінчастого синхронного JK-тригера на елементах І-НЕ потрібно замінити в рівнянні (1.6) змінні K і J на сукупності CJ і CK, після чого виконати спрощення:

$$Q_{t+1} = \overline{\overline{C \cdot K} \cdot Q} \vee C \cdot J \cdot \overline{Q} = \overline{\overline{C \cdot K} \cdot Q} \cdot \overline{C \cdot J \cdot \overline{Q}}. \qquad (1.6)$$

Схема двоступінчастого JK-тригера із логічними зв’язками на основі рівняння (1.6) показана на рис. 1.12.

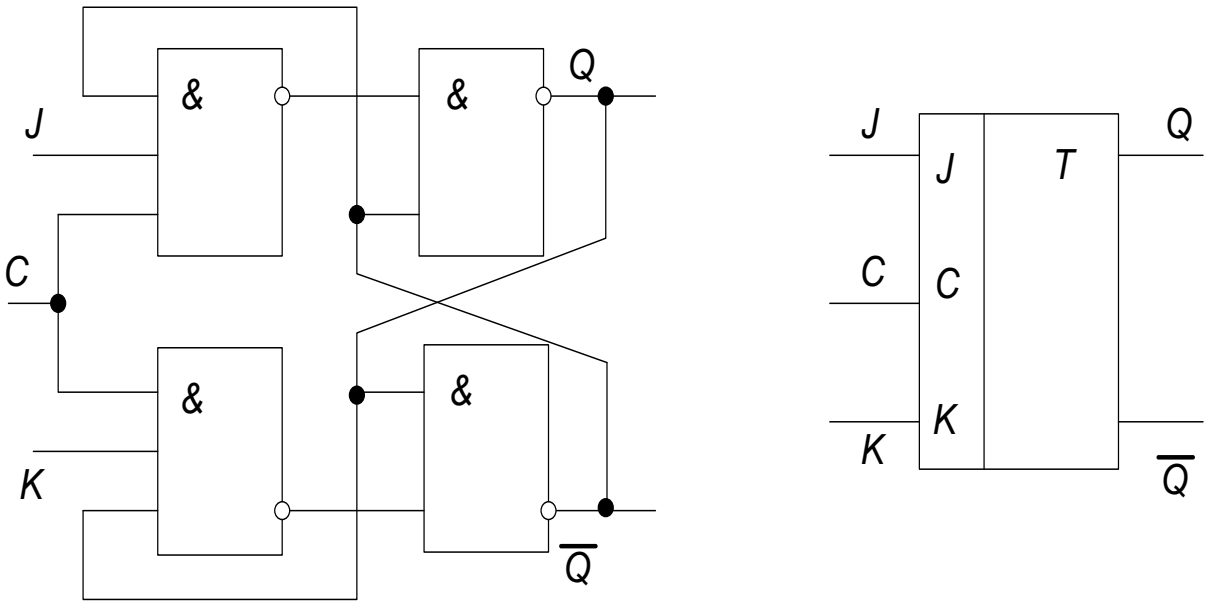


Рисунок 1.12 – Схема та умовне позначення двоступінчастого JK-тригера

T-тригер

Тригером типу Т називається запам'ятовувальний елемент із двома стійкими станами і одним інформаційним Т-входом. Стан Т-тригера змінюється на протилежний після кожного приходу лічильного сигналу на вхід Т. Логіка функціонування асинхронного лічильного тригера подана таблицею переходів і описується логічним рівнянням.

$$Q_{t+1} = \overline{T}_t \cdot Q_t \vee T_t \cdot \overline{Q}_t. \tag{1.7}$$

Для побудови асинхронного RS-тригера на елементах І-НЕ рівняння набуде вигляду, зручного для реалізації в заданому елементному базисі:

$$Q_{t+1} = \overline{\overline{\overline{T} \cdot Q} \vee \overline{T \cdot \overline{Q}}} = \overline{\overline{\overline{T} \cdot Q} \cdot Q \cdot \overline{T \cdot \overline{Q}}}. \tag{1.8}$$

Схема двоступінчастого асинхронного Т-тригера на елементах І-НЕ наведена на рис. 1.13.

Асинхронний Т-тригер містить два синхронних RS-тригера, при цьому на вхід Т основного ступеня подається зчитуваний сигнал, а входи S і R з'єднані відповідно із виходами Q і $\overline{Q}$ допоміжного ступеня.

T_t	Q_t	Q_{t+1}
0	0	0
0	1	1
1	0	1
1	1	1

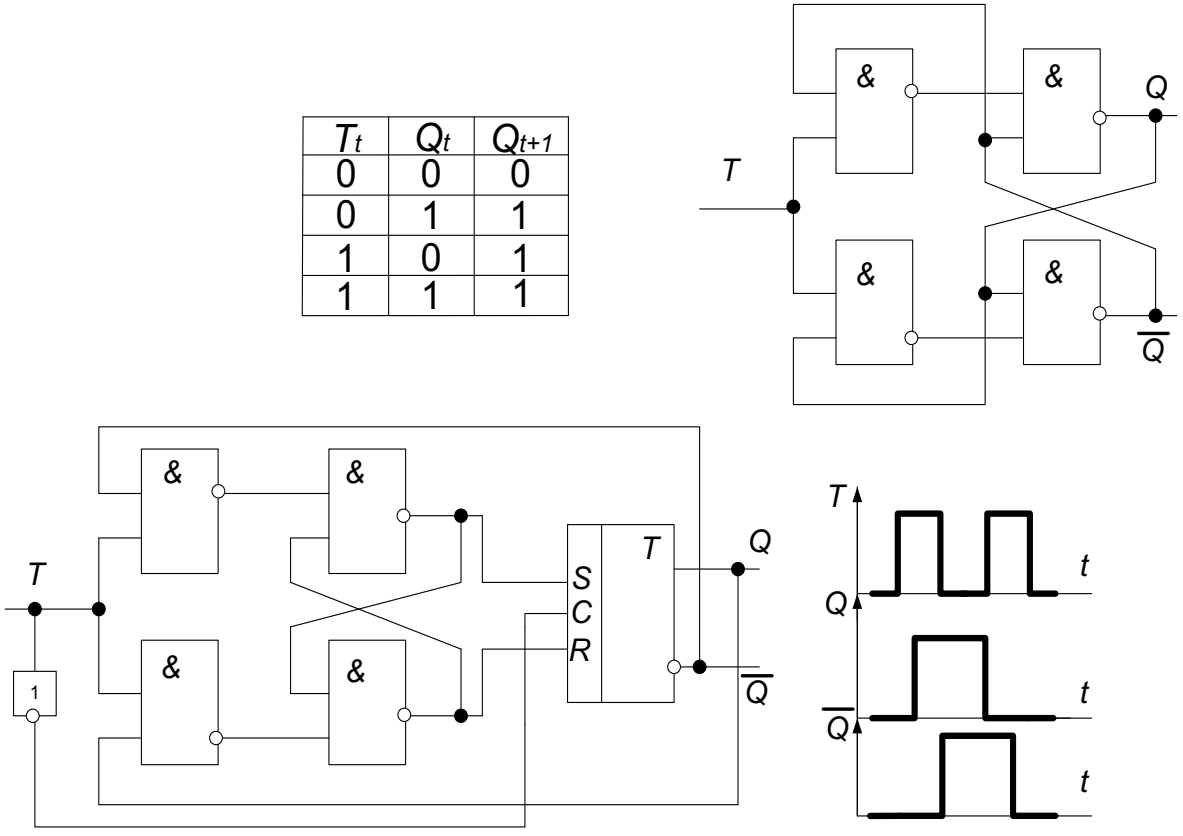


Рисунок 1.13 – Таблиця переходів, схеми синхронного, асинхронного Т-тригерів та часова діаграма роботи

D-тригер

Тригером типу D називається синхронний запам'ятовувальний елемент із двома стійкими станами і одним інформаційним входом. Закон функціонування D-тригера описується таким виразом:

$$Q_{t+1} = C_t D_t . \tag{1.9}$$

Це рівняння показує, що після переключення стану D-тригера повторюється значення сигналу на D-вході в тактові моменти часу. Тому такий тригер часто ще називають тригером затримки.

Схему D-тригера можна побудувати на основі синхронного RS-тригера, якщо сигнал по входу S одночасно подавати через інвертор на вхід R. Схеми D-тригера також будуються на основі самостійного логічного рівняння. Замінивши сигнал S на D і сигнал R на $\overline{D}$, маємо:

$$Q_{t+1} = \overline{\overline{C \cdot S} \cdot \overline{C \cdot R \cdot Q}} = \overline{\overline{C \cdot D} \cdot \overline{C \cdot \overline{D} \cdot Q}}. \tag{1.10}$$

Схеми D-тригера на елементах І-НЕ та на RS-тригері із логічними зв'язками показано на рис. 1.14.

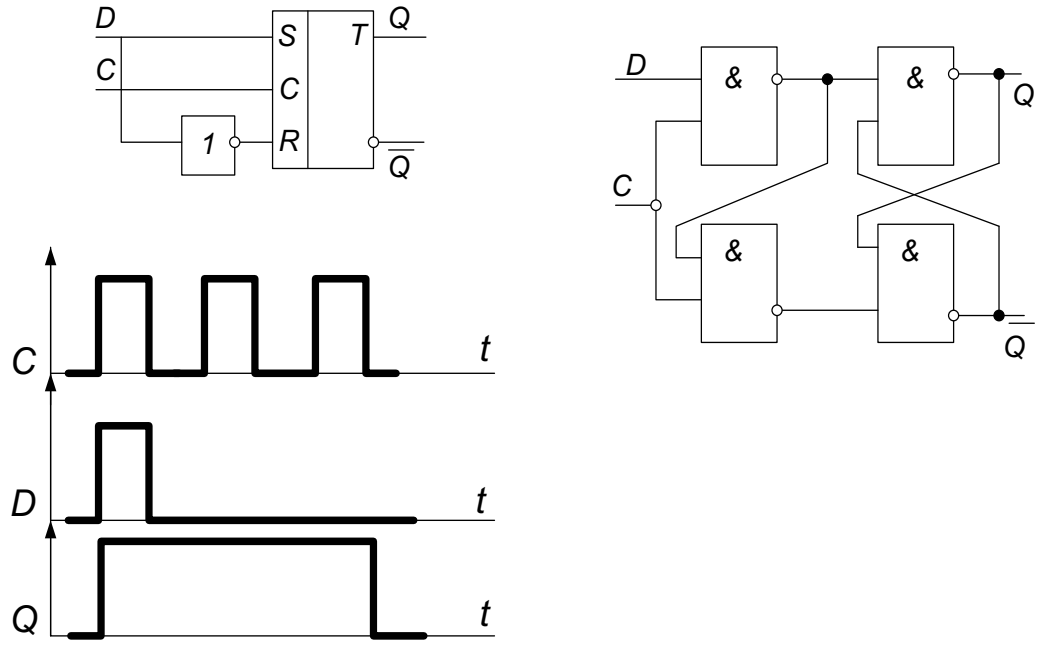


Рисунок 1.14 – Схеми синхронного D-тригера на RS-тригері, D-тригера на логічних елементах І-НЕ та часова діаграма роботи

Синтез довільного тригера на основі тригерів RS, JK, D, T

Для побудови тригера із заздалегідь визначеними параметрами на базі відомих тригерів використовують функції переходів даних тригерів (рис. 1.15).

Q	Q^{t+1}	
0	0	0
1	1	1
0	1	↑
1	0	↓

} ф-ції переходів

Рисунок 1.15 – Функції переходів тригерів

Таблиця 1.2 – Функції переходів відомих тригерів (х – «1» або «0»)

F	RS		JK		T	D
	R	S	J	K	T	D
0	0	x	0	x	0	0
1	x	0	x	0	0	1
↑	1	0	1	x	1	1
↓	0	1	x	1	1	0

На рис. 1.16 наведені карти Карно для мінімізації функції входів R і S, а також схема синтезу тригера із функцією, заданою в табл. 1.2.

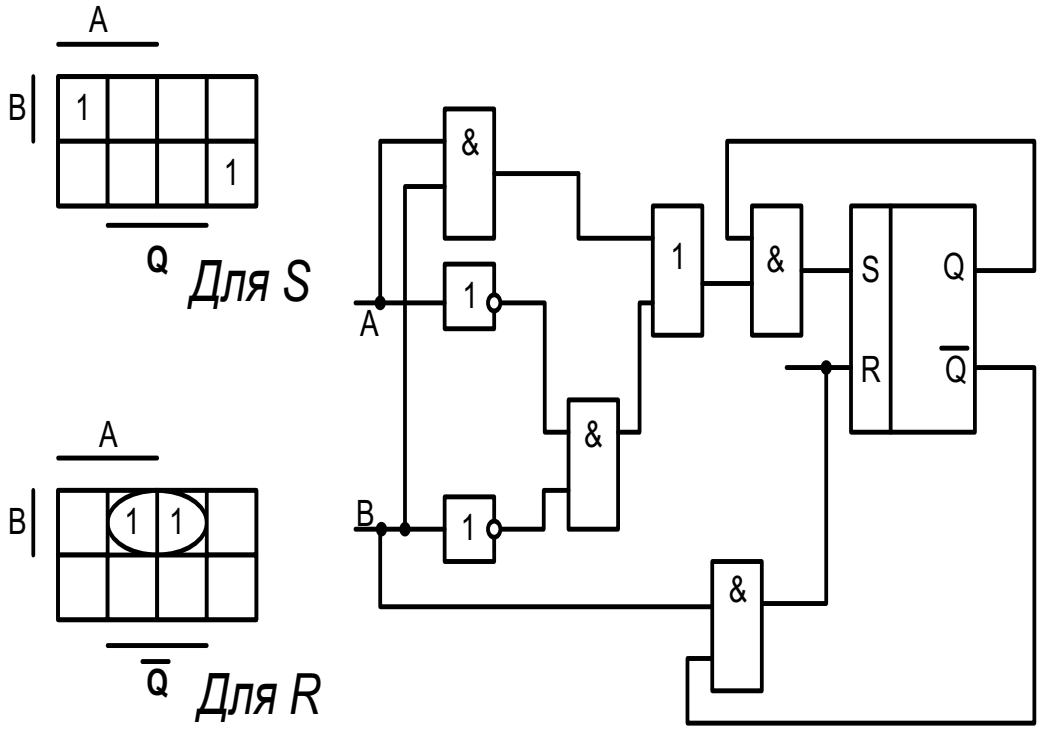


Рисунок 1.16 – Карти Карно для мінімізації функції входів R і S, а також схема синтезу тригера із функцією, заданою в табл. 1.2

Приклад

Синтезувати тригер, заданий таблицею переходів.

A	B	Q
0	0	1
0	1	0
1	0	Q
1	1	$\overline{Q}$

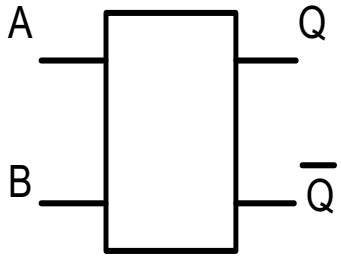


Рисунок 1.17 – Графічне позначення АВ-тригера

Побудуємо розширену таблицю переходів АВ-тригера.

Таблиця 1.3 – Розширена таблиця переходів АВ-тригера з відповідними функціями переходів для RS, JK, D і T тригерів

A	B	Q	Q^{t+1}	F	RS		JK		D	T
					S	R	J	K		
0	0	0	1	▲	1	0	1	x	0	1
0	0	1	1	1	x	0	x	0	1	0
0	1	0	0	0	0	x	0	x	0	0
0	1	1	0	▼	0	1	x	1	1	1
1	0	0	0	0	0	x	0	x	0	0
1	0	1	1	1	x	0	x	0	1	0
1	1	0	1	▲	1	0	1	x	0	1
1	1	1	0	▼	0	1	x	1	1	1

Синтез на базі RS-тригера

Використаємо з таблиці істинності ф-ції переходів. При підйомі «1» подається сигнал на S, при спуску «1» – на R.

$$S = \overline{A}\,\overline{B}\,\overline{Q} \vee AB\overline{Q} = \overline{Q}(\overline{A}\,\overline{B} \vee AB),$$
$$R = \overline{A}BQ \vee ABQ = BQ. \qquad (1.11)$$

Таким чином за отриманими рівняннями побудуємо схему АВ-тригера на основі RS-тригера (рис. 1.18).

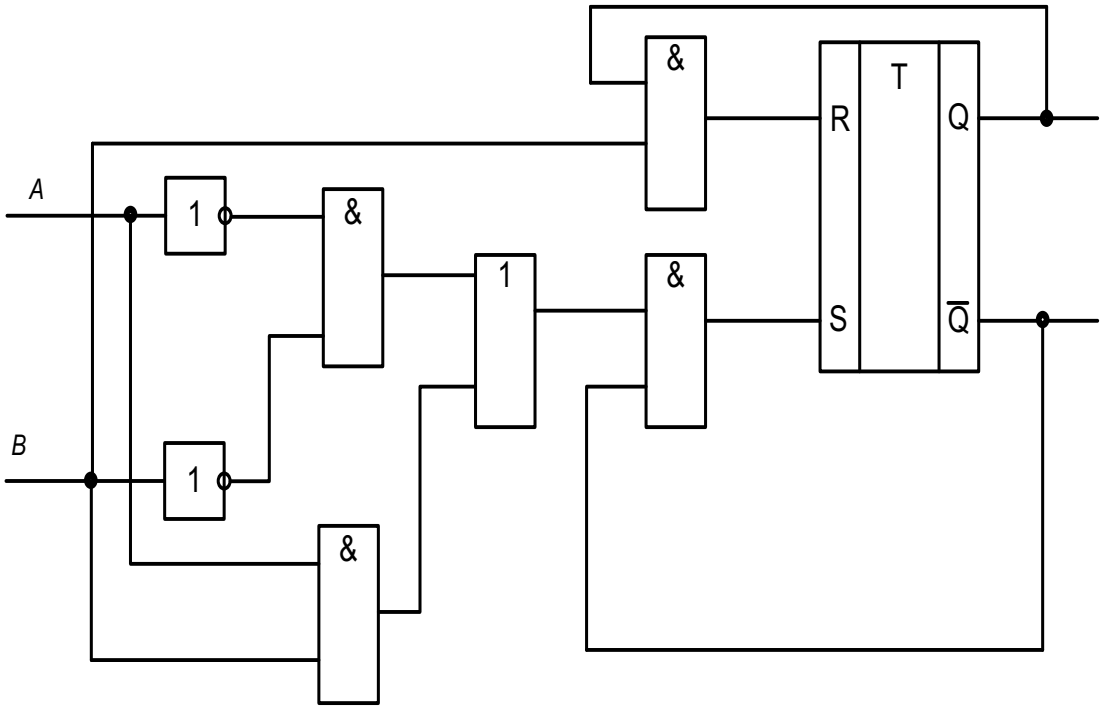


Рисунок 1.18 – Схема синтезованого АВ-тригера на базі RS-тригера

Синтез на базі JK-тригера

Для тригера JK функції переходів подібні до RS, замість S – J, замість R – K. Для входу J використовуються переходи 0 – 1 і 1 – 1, а для входу К – переходи 1 – 0 і 0 – 0. Таким чином отримаємо рівняння для входів тригера та відповідні спрощені рівняння за допомогою карт Карно (рис. 1.19).

$$\begin{aligned} \mathrm{J} &= \overline{A}\,\overline{\mathrm{B}}\,\overline{\mathrm{Q}} \vee \overline{\mathrm{A}}\,\overline{\mathrm{B}}\mathrm{Q} \vee \mathrm{A}\overline{\mathrm{B}}\mathrm{Q} \vee \mathrm{AB}\overline{\mathrm{Q}}; \\ \mathrm{K} &= \overline{\mathrm{A}}\mathrm{B}\overline{\mathrm{Q}} \vee \mathrm{A}\overline{\mathrm{B}}\,\overline{\mathrm{Q}} \vee \overline{\mathrm{A}}\mathrm{BQ} \vee \mathrm{ABQ}; \\ \mathrm{J} &= AB\overline{\mathrm{Q}} \vee \overline{\mathrm{B}}\mathrm{Q} \vee \overline{\mathrm{A}}\,\overline{\mathrm{B}}; \\ \mathrm{K} &= \mathrm{A}\overline{\mathrm{B}}\,\overline{\mathrm{Q}} \vee \mathrm{BQ} \vee \overline{\mathrm{A}}\mathrm{B}. \end{aligned} \qquad (1.12)$$

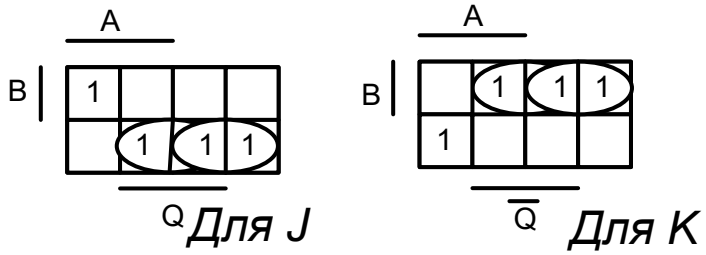


Рисунок 1.19 – Карти Карно для мінімізації функції входів A і B (для JK-тригерів)

Відповідно до цих рівнянь побудуємо схему AB-тригера на базі JK-тригера (рис. 1.20).

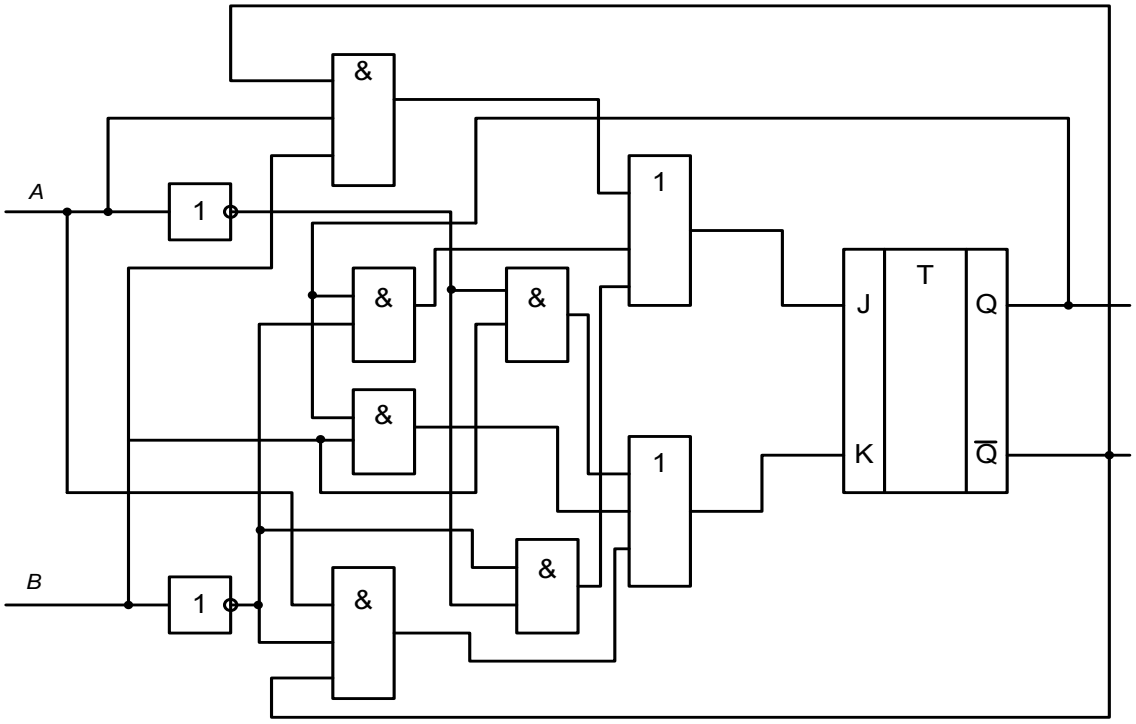


Рисунок 1.20 – Схема синтезованого AB-тригера на базі JK-тригера

Синтез на базі D-тригера

Для синтезу AB-тригера на базі D-тригера в таблиці переходів використовуються переходи 0 – 1 і 1 – 1 для подачі на вхід D. Таким чином складемо рівняння для входу D та спрощене рівняння за допомогою карти Карно (рис. 1.21).

$$\mathrm{D} = \overline{A}\,\mathrm{B}\overline{\mathrm{Q}} \vee \overline{\mathrm{A}}\,\mathrm{B}\overline{\mathrm{B}}\mathrm{Q} \vee \mathrm{A}\overline{\mathrm{B}}\mathrm{Q} \vee \mathrm{AB}\overline{\mathrm{Q}};$$

$$\mathrm{D} = AB\overline{\mathrm{Q}} \vee \overline{\mathrm{B}}\mathrm{Q} \vee \overline{\mathrm{A}}\,\overline{\mathrm{B}}. \qquad (1.13)$$

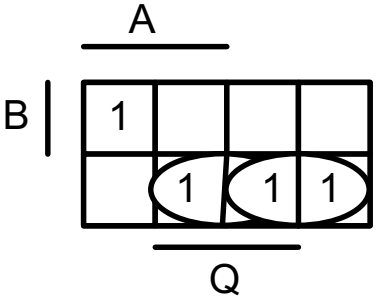


Рисунок 1.21 – Карта Карно для мінімізації функції входів A і B (для D-тригера)

Відповідно до цих рівнянь побудуємо схему АВ-тригера на базі D-тригера (рис. 1.22).

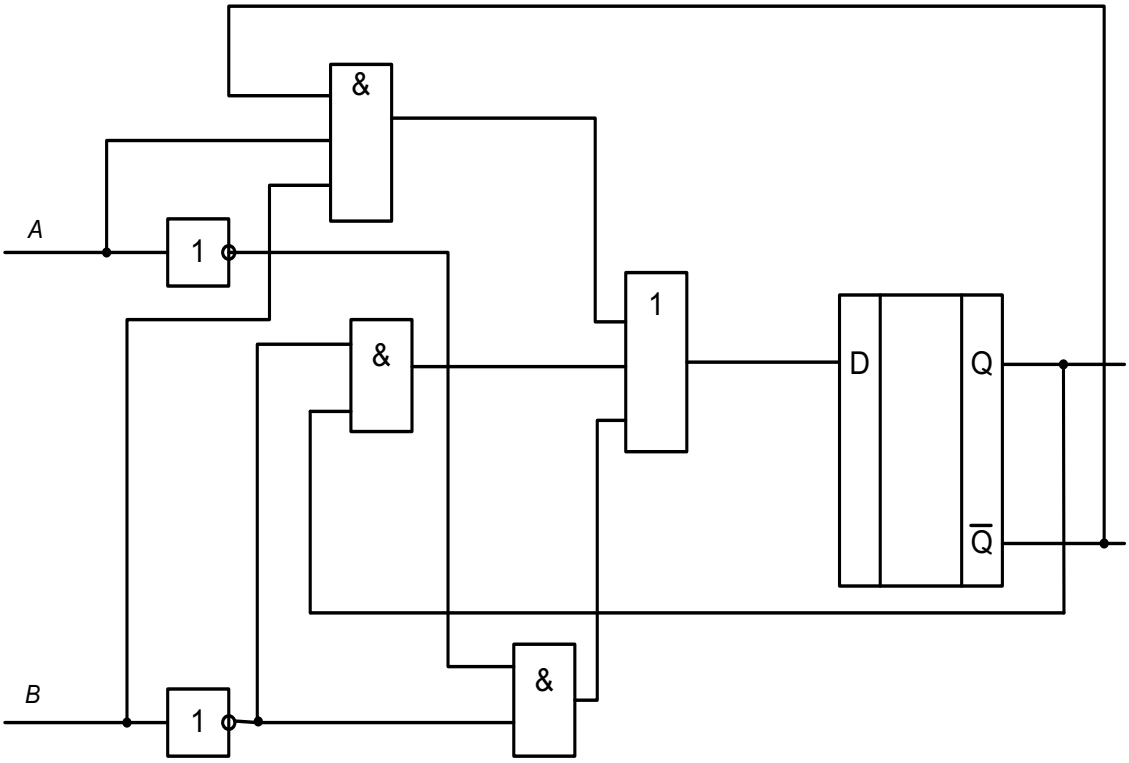


Рисунок 1.22 – Схема синтезованого АВ-тригера на базі D-тригера

Синтез на базі Т-тригера

Для синтезу АВ-тригера на базі Т-тригера в таблиці переходів використовуються переходи 0 – 1 і 1 – 0 для подачі на вхід Т. Таким чином складемо рівняння для входу Т та спрощене рівняння за допомогою карти Карно (рис. 1.23).

$$\mathrm{T} = \overline{A}\,\overline{\mathrm{B}}\,\overline{\mathrm{Q}} \vee \overline{\mathrm{A}}\mathrm{BQ} \vee \mathrm{AB}\overline{\mathrm{Q}} \vee \mathrm{ABQ};$$
$$\mathrm{T} = \overline{A}\,\overline{\mathrm{B}}\,\overline{\mathrm{Q}} \vee \mathrm{BQ} \vee \mathrm{AB}. \qquad (1.14)$$

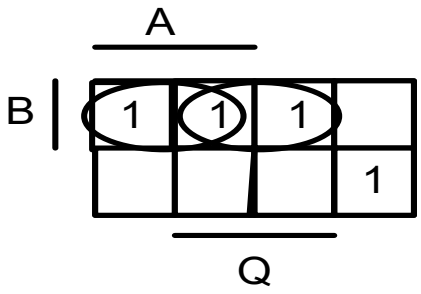


Рисунок 1.23 – Карта Карно для мінімізації функції входів A і B (для T-тригера)

Відповідно до цих рівнянь побудуємо схему AB-тригера на баз T-тригера (рис. 1.24).

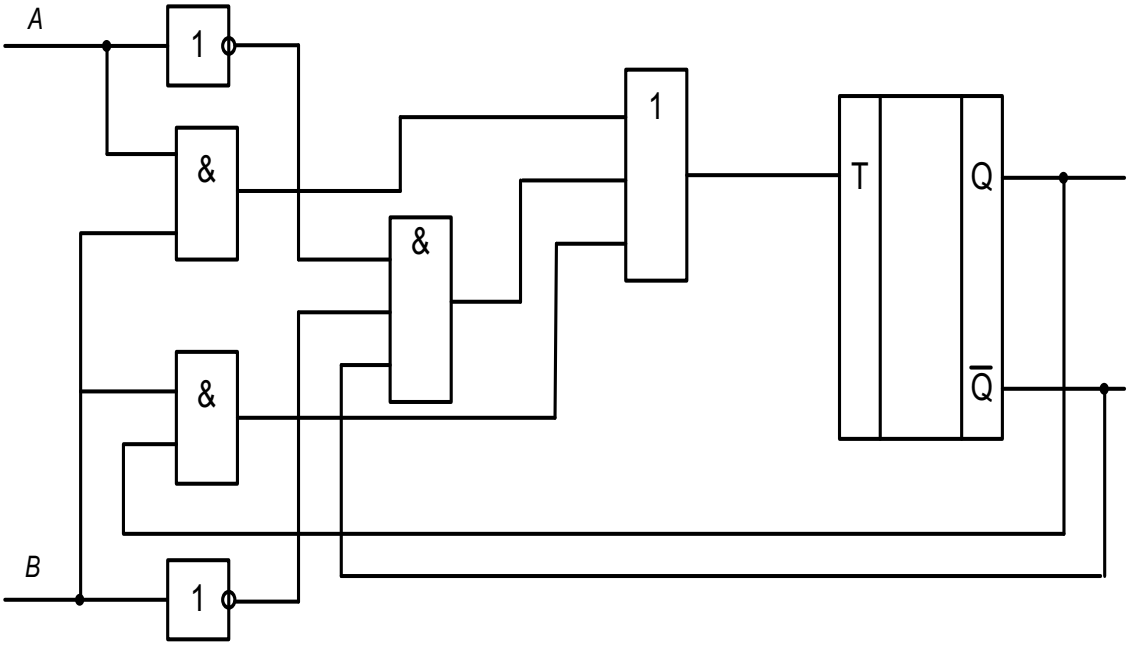


Рисунок 1.24 – Схема синтезованого AB-тригера на базі T-тригера

1.2 Регістри

Регістр являє собою набір двійкових ланок (тригерів з керувальними елементами), головним призначенням якого є зберігання інформації у вигляді багаторозрядних двійкових чисел (двійкового коду). На відміну від пристроїв довготривалої пам'яті в регістрах інформація запам'ятовується короткочасно, тобто на період одного або кількох циклів роботи всієї системи. Регістри призначені для запису, зберігання і читання одного двійкового числа або іншої кодової комбінації. Крім цих основних операцій регістри виконують додаткові операції: інвертування коду,

скидання в нульовий стан, перетворення послідовного коду в паралельний і навпаки.

В загальному випадку регістри забезпечують виконання таких мікрооперацій:

- установлення регістра в нуль (скидання);
- приймання слова з іншого регістра, лічильника і т. п.;
- передавання слів на інший регістр, лічильник і т. п.;
- перетворення кодів збережених слів в інверсні коди;
- зсув слова вліво або вправо на необхідне число розрядів;
- перетворення послідовного коду в паралельний і навпаки.

Схеми конкретних регістрів у деяких випадках можуть реалізувати лише деякі з перелічених мікрооперацій.

Запам'ятовувальні елементи регістра за кількістю розрядів двійкового числа виготовляють на основі RS-, D-, JK-тригерів. Для допоміжних операцій (введення до регістра або виведення з нього числа, яке зберігається, перетворення коду двійкового числа, зсуву числа на певне число розрядів вліво або вправо) застосовують комбінаційні схеми на основі логічних елементів.

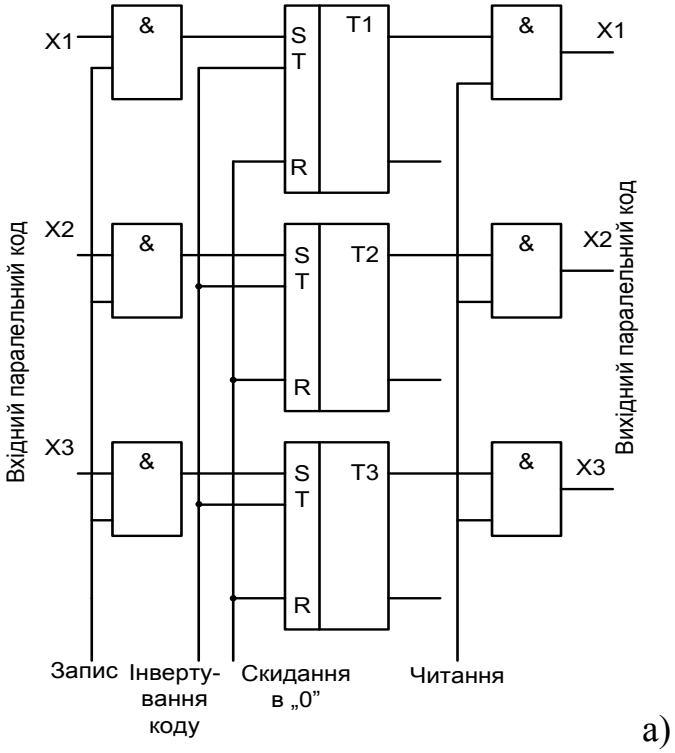
Залежно від способу запису інформації регістри розділяють на 3 типи:

1) регістри паралельного типу (без зсуву);
2) регістри послідовного типу (із зсувом);
3) комбіновані регістри.

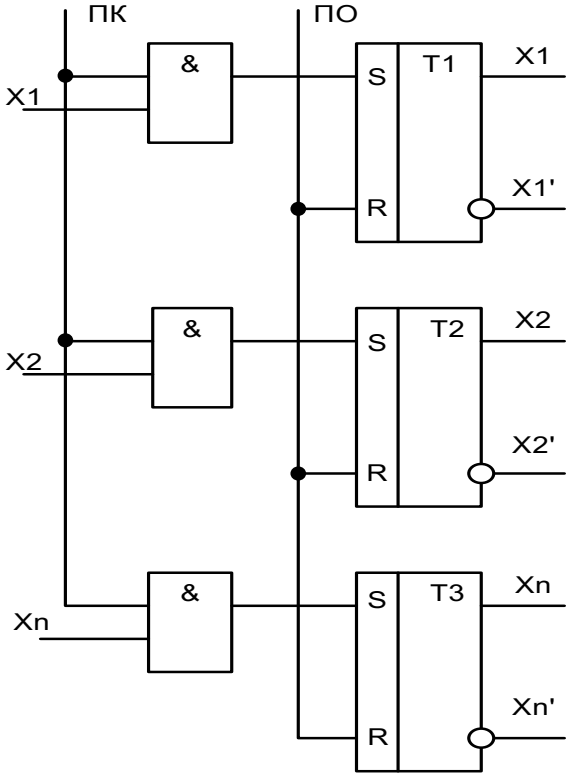
Регістри паралельного типу (без зсуву)

В паралельних регістрах інформація (двійкові числа – слова) записується одночасно до всіх розрядів (паралельний код). Паралельним регістром називається такий регістр, що реалізовує всі перераховані мікрооперації крім зсуву й перетворення послідовного коду в паралельний і навпаки. Якщо в паралельному регістрі на вхід кожного каскаду інформація надходить по двох каналах у парафазному коді, то такий регістр називають парафазним. При наявності тільки одного каналу (прямого або інверсного) надходження інформації в кожному розряді регістра називають однофазним. Загальна схема паралельного регістра зображена на рис. 1.20. Схема паралельного однофазного регістра, що виконує перші дві мікрооперації з наведеного вище списку, показана на рис. 1.25, б.

На нульові входи всіх тригерів подається сигнал установлення нульового стану ПО. Після подання цього сигналу всі тригери регістра будуть переведені в стан «0» до появи на вхідних шинах записуваного слова й сигналу приймання ПК у тих розрядах, де $X_i = 1$, відбудеться установлення тригерів в одиничний стан. Там, де $X_i = 0$, стан тригерів не змінюється. Видача інформації з регістра може відбуватися в прямому, інверсному й парафазному кодах.



а)



б)

Рисунок 1.25 Загальна схема паралельного регістра – (а),
схема паралельного однофазного регістра – (б)

Схема видачі інформації в прямому й інверсному кодах показана на рис. 1.26. *B1* – сигнал видачі прямого коду, *B2* – сигнал видачі інверсного коду. Одночасна поява сигналів *B1* й *B2* заборонена. Схема регістра з видачею пара фазного коду наведена на рис. 1.26, б.

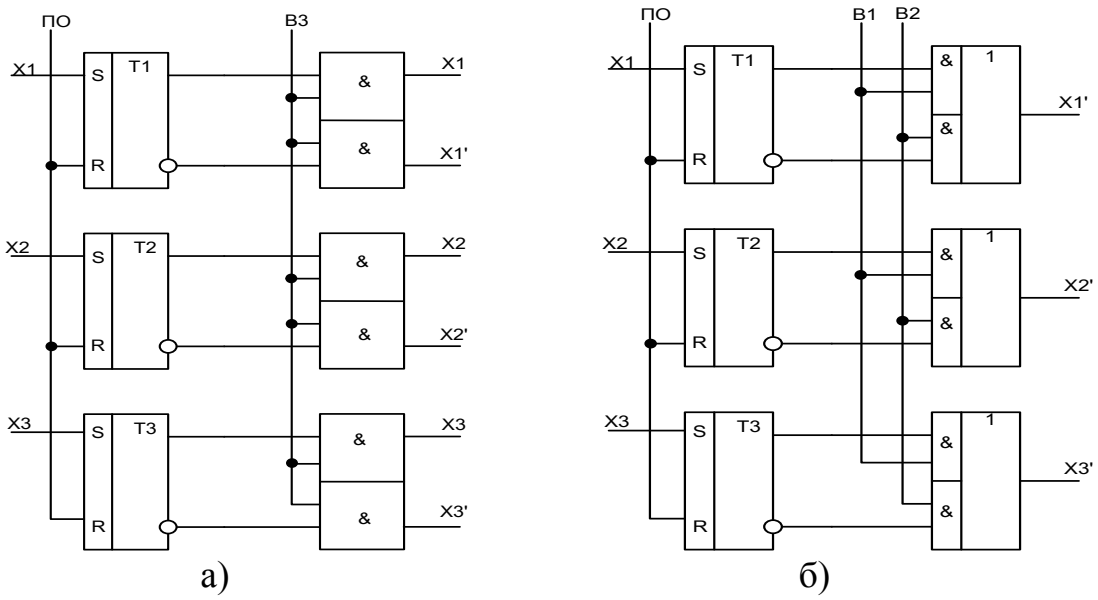


Рисунок 1.26 Схема видачі інформації в прямому й інверсному кодах – (а), схема регістра з видачею парафазного коду – (б)

Використання парафазного коду забезпечує установлення тригерів у необхідний стан незалежно від тієї інформації, що була записана в них раніше.

Регістри послідовного типу (із зсувом)

Зсув – це одночасне просторове переміщення двійкового слова в розрядній сітці із збереженням порядку проходження нулів і одиниць. Регістри, призначені для виконання мікрооперацій зсуву, називаються регістрами зсуву.

Мікрооперації зсуву використовують у процесі виконання команд множення, ділення і нормалізації. Крім того, за допомогою зсуву здійснюється перетворення паралельного коду в послідовний або навпаки (наприклад, при обміні інформацією з магнітними стрічками і дисками).

Зсув слова може виконуватися вправо (у бік молодших розрядів) або вліво (у бік старших розрядів).

Загальна структурна схема регістра послідовного типу (із зсувом) зображена на рис. 1.27

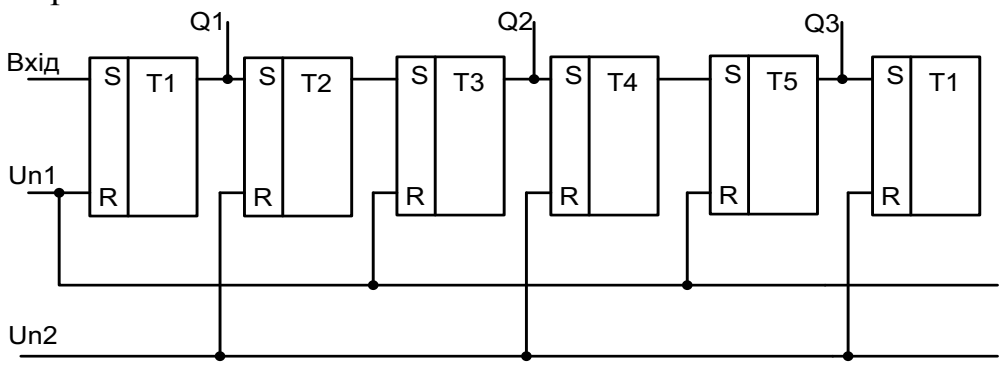


Рисунок 1.27 – Загальна структурна схема регістра послідовного типу

Схема складається з попарно включених тригерів – основного T_i і допоміжного T'_i. Є дві шини просування імпульсів U_{n1}, U_{n2}. Нехай $Q_1 = 1$, а на виході $Q_n = 0$, тоді під дією сигналів U_{n1}, U_{n2} ця логічна одиниця буде просуватись вправо. Так само можна за допомогою U_{n1}, U_{n2} послідовно записати код в тригери $T_1...T_n$, тобто послідовний код перетворити в паралельний. Для записування n-розрядного числа потрібно 2n імпульсів просування. Можна також записаний паралельний код за допомогою U_{n1}, U_{n2} просувати вправо і на виході Q_n з'явиться послідовний код.

В послідовних регістрах здійснюється зсув коду числа за однотактною або багатотактною схемою. Однотактними регістрами керує одна послідовність, за якої зсув коду на один розряд здійснюється кожним зсувним імпульсом. Коли регістрами керує кілька послідовностей (дві, три і т. д.) імпульсів, вони називаються багатотактними (двотактними, тритактними і т. д.).

Найбільш широко відомий тип регістра зсуву на JK-тригерах наведено на рис. 1.28. Припустимо, що всі тригери спочатку перебувають у нульовому стані. Таку ситуацію можна забезпечити, подавши імпульси на лінію, безпосередньо з'єднану з виводом скидання кожного тригера (якщо такі передбачені) або зсуваючи ланцюжок нулів.

Введення даних здійснюється в такий спосіб. Припустимо, що спочатку вхідна лінія W перебуває в стані W_1 (1 або 0). Оскільки $J = W$, а $K = \overline{W}$, після першого синхроімпульсу $A = W_1$. Тепер стан на вході стає рівним W_2. Після другого синхроімпульсу $A = W_2$, а $B = W_1$.

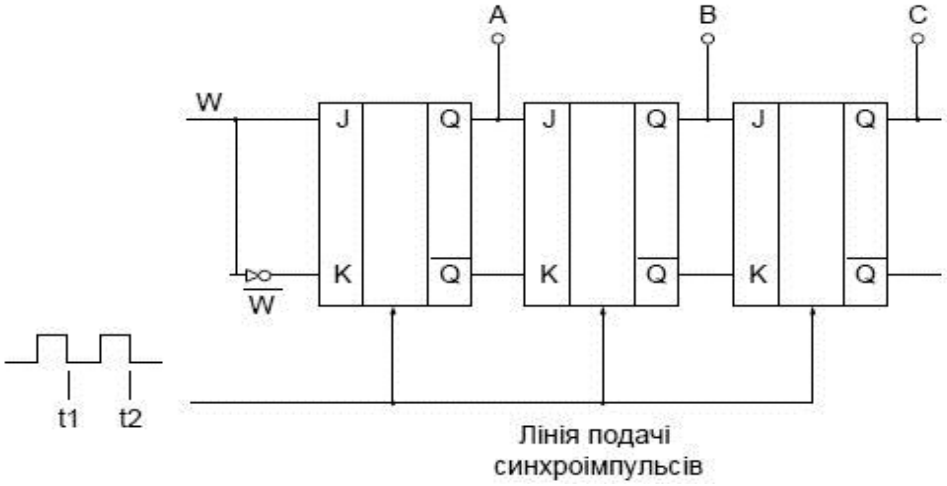


Рисунок 1.28 – Регістр зсуву на JK-тригерах

У табл. 1.4 показано, як інформація зсувається по рядку. Залежно від конструктивного рішення стан на виході пристрою може змінюватися під час проходження переднього або заднього фронту синхроімпульсу або відразу після його проходження.

Таблиця 1.4 – Зсув інформації по регістру

Час	Виходи		
	A	B	C
0	0	0	0
t1	W_1	0	0
t2	W_2	W_1	0
t3	W_3	W_2	W_1
t4	W_4	W_3	W_2

Перетворення послідовного коду в паралельний і навпаки. Схема чотирирозрядного регістра зсуву вправо на JK-тригерах, яка забезпечує перетворення кодів, показана на рис. 1.29, а. Старший розряд регістра за допомогою інвертора на К-вході працює в режимі D-тригера.

Нехай від накопичувачів на магнітних дисках або стрічках на вхід регістра по лінії D надходить послідовний код слова A=1101 в напрямку від молодших розрядів до старших. Значення розрядів слова надходить одночасно із синхроімпульсами, які забезпечують як приймання коду в старший розряд, так і одночасний зсув вмісту регістра вправо (рис. 1.29, б).

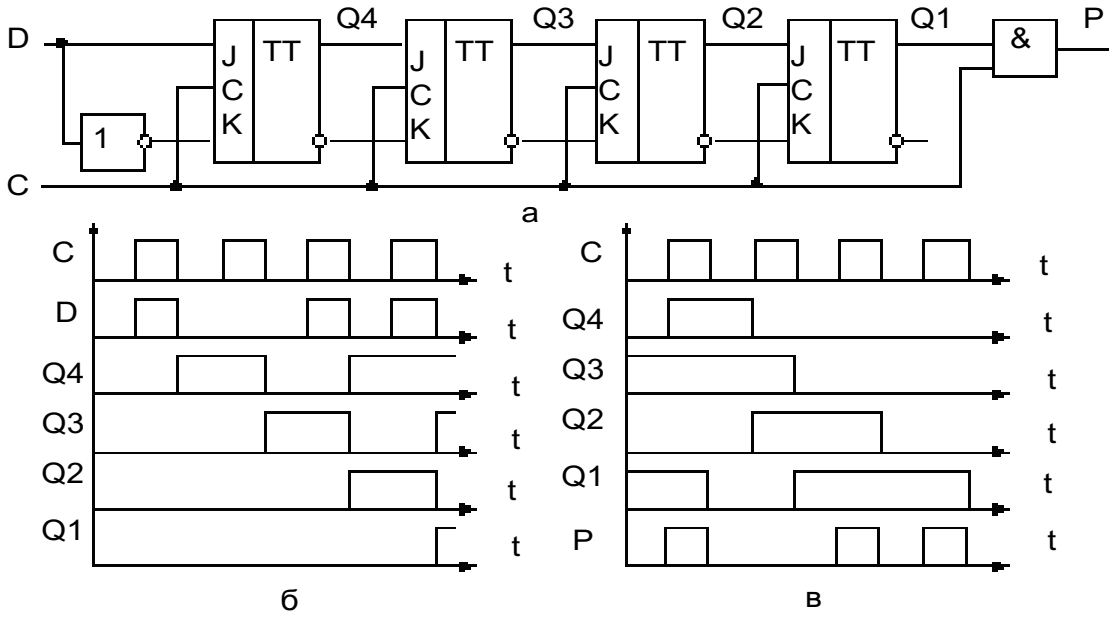


Рисунок 1.29 – Регістр зсуву: а) – схема; б), в) – перетворення послідовного коду в паралельний і навпаки

Після надходження чотирьох синхроімпульсів на виходах регістра $Q_4 – Q_1$ встановлюється код 1101. Таким чином здійснюється перетворення

послідовного коду в паралельний, яке часто називають послідовним введенням слова в регістр.

Перетворення паралельного коду в послідовний також відбувається зсувом слова, яке зберігається. Процес перетворення паралельного коду слова A=1101 в послідовний в напрямку від молодших розрядів до старших за допомогою зсуву вправо показано на рис. 1.29, в.

Комбіновані регістри

Комбінований регістр – це такий регістр, що дозволяє проводити паралельний запис і послідовний зсув інформації. Його ще називаються універсальним. Схема комбінованого регістра подана на рис. 1.30.

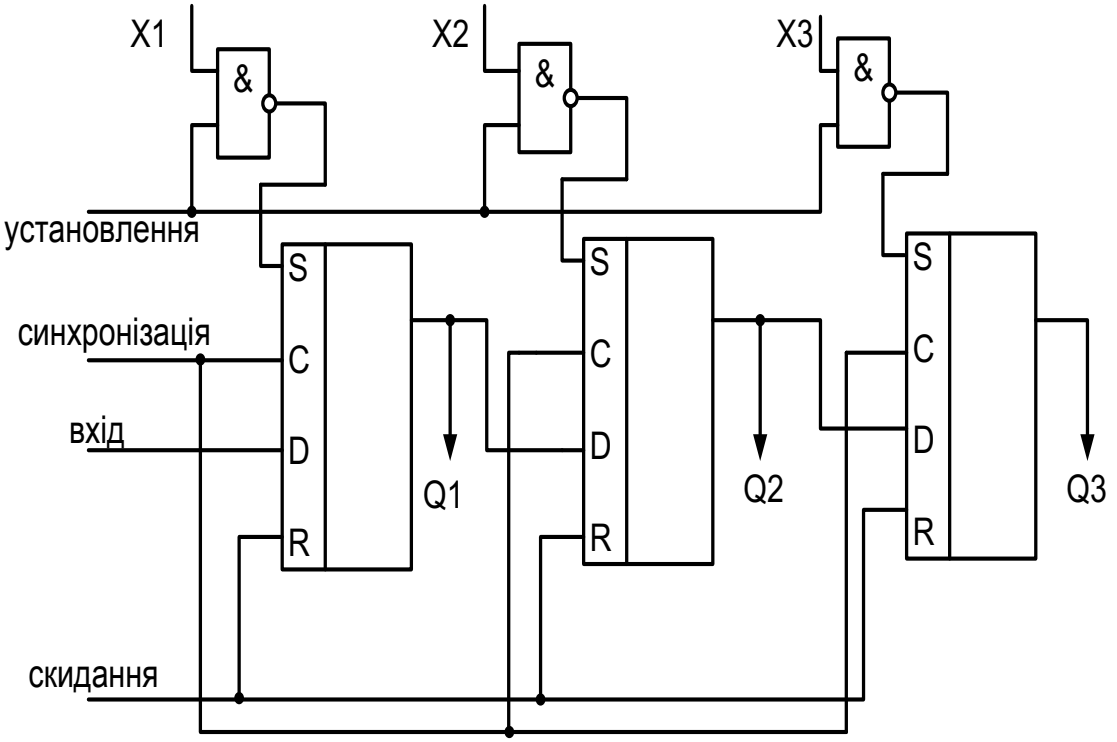


Рисунок 1.30 – Комбінований регістр

Даний регістр має входи керування: синхронізація, скидання та установлення, які дозволяють використовувати його як для паралельного записування, так і для зсуву. «0» на каналі «установлення» – зсув, «1» – паралельний запис інформації.

Синтез регістрів

Інколи існує необхідність синтезу регістра із довільною логікою роботи. Наприклад необхідно синтезувати регістр на базі D-тригера, двонаправлений, послідовного зсуву. Тут Qn – вихідне значення даної комірки, Qn-1 – сигнал попередньої комірки, Qn+1 – сигнал наступної комірки. Складемо таблицю станів регістра.

За умовою пристрій повинен виконувати дві функції та мати один сигнал синхронізації. Для побудови регістра вибираємо синхронний D-тригер, розглянемо функції переходів даного тригера.

К = 0 – зсув вправо.

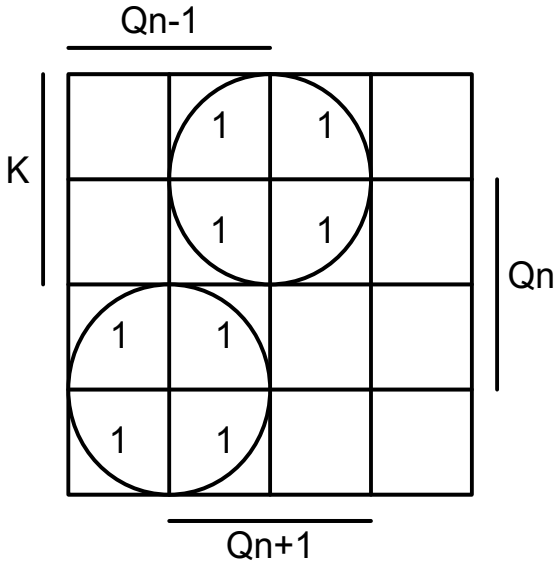
К = 1 – зсув вліво.

Від Qn-1 – до Qn – зсув вправо.

Від Qn+1 – до Qn – зсув вліво.

Таблиця 1.5 – Таблиця станів регістра на базі D-тригера, двонаправленого, послідовного зсуву

C	K	Qn-1	Qn	Qn+1	Qn^{t+1}
⎍	0	0	0	0	0
⎍	0	0	0	1	0
⎍	0	0	1	0	0
⎍	0	0	1	1	0
⎍	0	1	0	0	1 ↑
⎍	0	1	0	1	1 ↑
⎍	0	1	1	0	1
⎍	1	0	0	0	0
⎍	1	0	0	1	1 ↑
⎍	1	0	1	0	0
⎍	1	0	1	1	1 ↑
⎍	1	1	0	0	0
⎍	1	1	0	1	1 ↑
⎍	1	1	1	0	0
⎍	1	1	1	1	1 ↑



$$Q_n^{t+1} = Q_{n-1}\overline{K} + Q_{n+1}K,$$
$$Q_n^{t+1}C = C(Q_{n-1}\overline{K} + Q_{n+1}K).$$

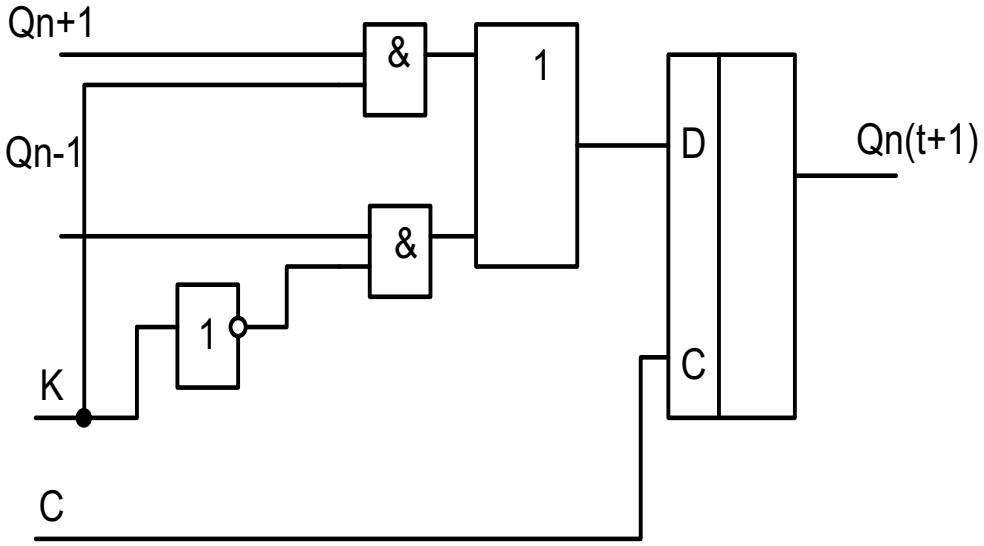


Рисунок 1.31 – Двонаправлений регістр послідовного зсуву

Приклад

Синтез генератора псевдовипадкових чисел на регістрі

Нехай необхідно створити генератор послідовності значень 0→1→3→7→6→5→2→4. Для реалізації такого пристрою достатньо використати трирозрядний регістр. Граф переходів регістра:

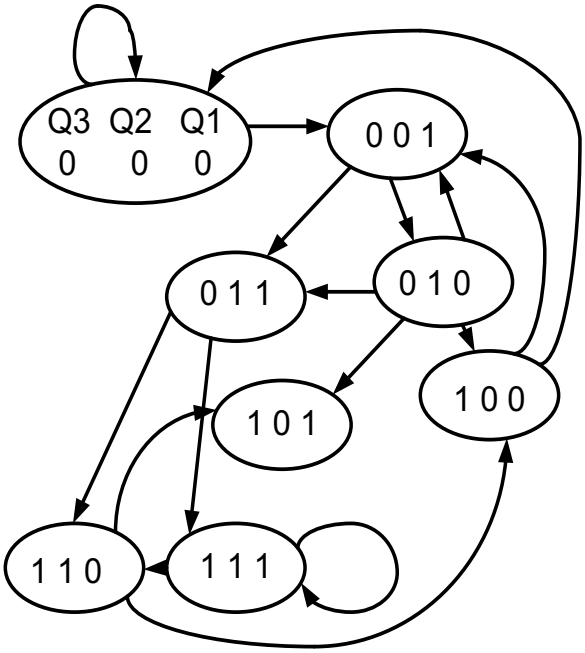


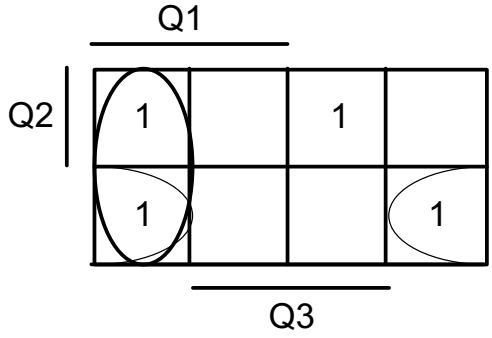
Рисунок 1.32 – Граф переходів стану регістра

Таблиця 1.6 – Таблиця переходів регістра

Q3	Q2	Q1	$Q3^{t}$	$Q2^{t+1}$	${}^{1}Q1^{t+1}$	Перехід
0	0	0	0	0	1	↑
0	0	1	0	1	1	1
0	1	1	1	1	1	1
1	1	1	1	1	0	↓
1	1	0	1	0	1	↑
1	0	1	0	1	0	↓
0	1	0	1	0	0	0
1	0	0	0	0	0	0

Вихідна функція регістра:

$$F = \overline{Q}1\overline{Q}2\overline{Q}3 \vee Q1\overline{Q}2\overline{Q}3 \vee Q1Q2\overline{Q}3 \vee \overline{Q}1Q2Q3. \qquad (1.15)$$



$$D = Q1\overline{Q}3 \vee \overline{Q}2\overline{Q}3 \vee \overline{Q}1Q2Q3. \qquad (1.16)$$

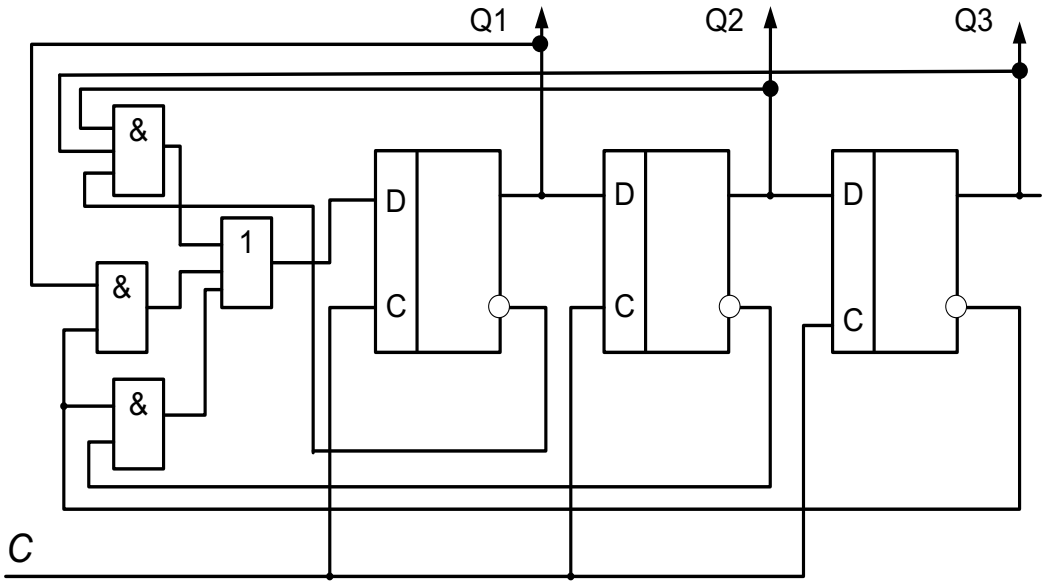


Рисунок 1.33 – Генератор псевдовипадкових чисел на базі регістра

1.3 Лічильники

Лічильником називається типовий функціональний вузол комп'ютера, призначений для лічіння вхідних імпульсів. Лічильник являє собою зв'язане коло тригерів, які утворюють пам'ять із заданим числом сталих станів (рис. 1.34).

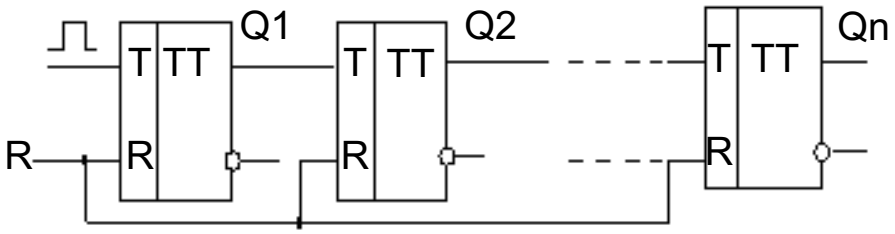


Рисунок 1.34 – Логічна структура лічильника

Розрядність лічильника n дорівнює числу тригерів. Кожний вхідний імпульс змінює стан лічильника, який зберігається до надходження наступного сигналу. Значення виходів тригерів лічильника *Qn, Qn–1, ..., Q1* відображають результат лічби в прийнятій системі числення. Логічна функція лічильника позначається буквами СТ (counter). Список мікрооперацій лічильника вміщує попереднє встановлення в початковий стан, інкремент або декремент слова, яке зберігається, видачу слів паралельним кодом та ін.

Вхідні імпульси можуть надходити на лічильник як періодично, так і довільно розподіленими у часі. Амплітуда і тривалість лічильних імпульсів мають задовольняти технічні вимоги для серій мікросхем, які використовуються.

Лічильник є одним з основних функціональних вузлів комп'ютера, а також різних цифрових керувальних та інформаційно-вимірювальних систем.

Основне застосування лічильників:

- утворення послідовності адрес команд програми (лічильник команд або програмний лічильник);
- підрахунок числа циклів при виконанні операцій ділення, множення, зсуву (лічильник циклів);
- одержання сигналів мікрооперацій і синхронізації;
- аналого-цифрові перетворення і побудова електронних таймерів (годинників реального часу).

Лічильник характеризується модулем і ємністю лічби. Модуль лічби КЛЧ визначає число станів лічильника. Модуль двійкового *n*-розрядного лічильника визначається цілим степенем двійки $M=2n$. Після лічіння числа імпульсів NBX=КЛЧ лічильник повертається в початковий стан. Таким чином, модуль лічби, який часто називають коефіцієнтом перерахунку, визначає цикл роботи лічильника, після чого його стан повторюється. Тому число вхідних імпульсів і стан лічильника однозначно визначені тільки для першого циклу.

Ємність лічби N_{max} визначає максимальну кількість вхідних імпульсів, яку може зафіксувати лічильник при одному циклі роботи. Ємність лічби $N_{max}=КЛЧ-1$ за умови, що робота лічильника починається з нульового початкового стану.

У лічильниках використовуються три режими роботи: керування, накопичення і ділення. У режимі керування зчитування інформації виконується після кожного вхідного лічильного імпульсу, наприклад, в лічильнику адреси команд. У режимі накопичення головним є підрахунок заданого числа імпульсів або лічіння протягом певного часу. У режимі ділення (перерахунку) основним є зменшення частоти надходження імпульсів в КЛЧ разів. Більшість лічильників може працювати в усіх режимах, проте в спеціальних лічильниках-подільниках стани в процесі лічби можуть змінюватися в довільному порядку, що дозволяє спростити схему вузла.

Лічильники класифікують за такими ознаками:

- способом кодування – позиційні та непозиційні;
- модулем лічби – двійкові, десяткові, з довільним постійним або змінним (програмованим) модулем;
- напрямком лічби – прості (підсумовувальні, віднімальні) і реверсивні;
- способом організації міжрозрядних зв'язків – з послідовним, наскрізним, паралельним і комбінованим переносами (позикою);

- типом використовуваних тригерів – T, JK, D в лічильному режимі;

- елементним базисом – потенціальні, імпульсні та потенціально-імпульсні.

У лічильниках з позиційним кодуванням числовий вираз поточного стану лічильника визначається за формулою:

$$N = \sum_{i=1}^{n} r_i, \tag{1.17}$$

$$Q_i = r_n\, Q_n + r_{n-1}\, Q_{n-1} + \ldots + r_1\, Q_1, \tag{1.18}$$

де r_i – вага і-го розряду;

Q_i – значення виходу і-го розряду;

n – число розрядів.

Нульове значення всіх розрядів звичайно беруть як початковий стан лічильника. Всі інші стани нумерують за числом вхідних імпульсів, що надійшли.

У лічильниках з непозиційним кодуванням (наприклад, у кодах Грея) розряди не мають постійних ваг і кожному набору станів $Q_n, Q_{n-1}, \ldots, Q_1$ приписується певна кількість вхідних імпульсів. У комп'ютерах переважно використовують лічильники з позиційним кодуванням.

За видом переходів прості лічильники (Лч) розподіляються на підсумовувальні (прямої лічби) і віднімання (зворотної лічби). У підсумовувальних лічильниках кожний доданий імпульс $U+$ збільшує стан на одиницю, тобто реалізовується мікрооперація інкремента $Лч: = Лч + 1$.

До часових характеристик лічильників відносяться роздільна здатність, швидкодія і час встановлення (перемикання) коду.

Роздільна здатність $t_{рз}$ визначається мінімальним інтервалом часу між двома вхідними імпульсами, при якому ще зберігається роботоздатність лічильника. Параметр $t_{рз}$ задають часом перемикання t_T першого (молодшого) тригера лічильника, тобто $t_{рз} = t_T$, оскільки він перемикається під дією кожного вхідного імпульсу.

Швидкодія лічильника визначається максимальною частотою F_m надходження вхідних імпульсів в режимі ділення й обчислюється за формулою $F_m = 1/t_T$. Час встановлення коду $t_{вст}$ відраховується від початку вхідного імпульсу до моменту отримання нового стану. Даний параметр дозволяє обчислювати швидкодію лічильника в режимі керування. Міжрозрядні зв'язки забезпечують вироблення сигналів перенесення в старші розряди при додаванні імпульсів і сигналів позики – при відніманні. Від виду реалізації міжрозрядних зв'язків суттєво залежать параметри $t_{вст}$ і $F_{т.к}$.

У лічильниках з послідовними перенесеннями тригери перемикаються почергово після кожного вхідного імпульсу в напрямку від молодших

розрядів до старших. Такі лічильники називаються послідовними або асинхронними. У лічильниках з паралельними перенесеннями тригери перемикаються одночасно після кожного вхідного імпульсу, такі лічильники називаються паралельними або синхронними.

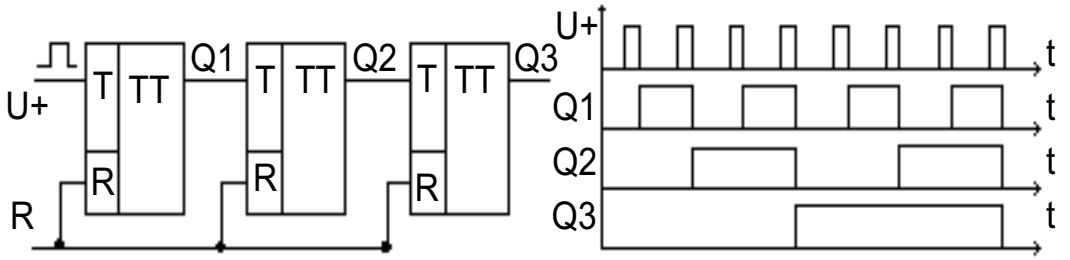
Двійкові підсумовувальні лічильники та лічильники віднімання

Двійкові лічильники реалізовують лічбу вхідних імпульсів у двійковій системі числення.

Число розрядів *n* двійкового підсумовувального лічильника для заданого модуля *M* знаходять із виразу $n = log2M$.

У двійковому підсумовувальному лічильнику перенесення P_i в сусідній старший розряд Q_{i+1} виникає в тому випадку, коли в момент надходження чергового лічильного імпульсу *U+* всі молодші розряди знаходяться в одиничному стані, тобто $P_i=U+Q_iQ_{i-1}...Q_1=1$. Після вироблення перенесення старший розряд перемикається в стан «1», а всі молодші розряди – в стан «0».

Асинхронні підсумовувальні лічильники на двоступеневих *T*-тригерах будуються так, щоб вхідні імпульси *U+* надходили на лічильний вхід тільки першого (молодшого) розряду. Сигнали перенесення передаються асинхронно (послідовно в часі) з прямих виходів молодших розрядів на *T*-входи сусідніх старших, як показано на рис. 1.35 для трирозрядного лічильника.



а) б)

Рисунок 1.35 – Асинхронний підсумовувальний лічильник на двоступеневих *T*-тригерах: а) – схема; б) – часові діаграми роботи

Зміна станів тригерів відбувається за спадом лічильного імпульсу для першого розряду, а для останніх – за спадом сигналу перенесення (рис. 1.30, б). Після підрахунку семи імпульсів на виході трирозрядного лічильника установлюється двійковий код *Q3Q2Q1=111* (тобто максимальне значення або ємність лічби). Після надходження восьмого вхідного імпульсу *U+* трирозрядний підсумовувальний лічильник перемикається у початковий нульовий стан послідовно (асинхронно) в часі: спочатку спадає напруга на виході *Q1*, потім – на виході *Q2* і т. д.

За допомогою імпульсу за входом скидання *R* лічильник повертається в нульовий стан у будь-який момент часу.

Схема трирозрядного асинхронного двійкового підсумовувального лічильника на *T*-тригерах з динамічним керуванням по фронту показана на рис. 1.36. Лічильні імпульси *U*+ надходять на *T*-вхід тільки першого (молодшого) розряду; наступні тригери перемикаються асинхронно від сигналів перенесення з інверсних виходів сусідніх молодших розрядів.

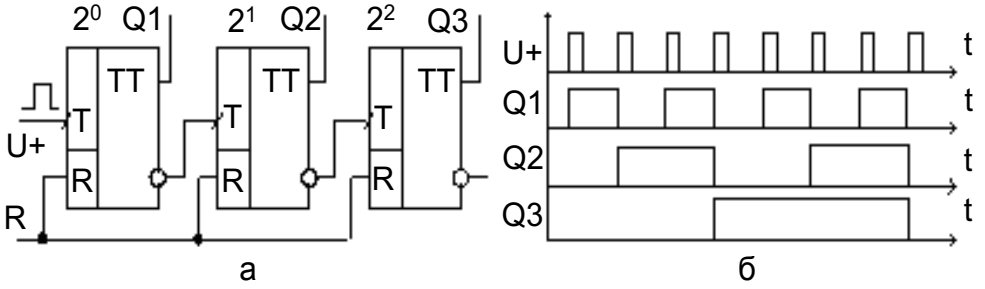


Рисунок 1.36 – Асинхронний підсумовувальний лічильник на тригерах з динамічним керуванням по фронту: а) – схема; б) – часові діаграми роботи

Перевагою асинхронних лічильників є простота схеми: збільшення розрядності виконується підключенням необхідного числа тригерів. До недоліків асинхронних лічильників відносяться порівняно низька швидкодія в режимі керування та її залежність від числа розрядів, а також поява проміжних вихідних двійкових кодів у процесі послідовного перемикання тригерів у новий стан.

Для одержання мінімального часу перемикання лічильника використовують паралельні перенесення (рис. 1.37). Для цього в кожному розряді синхронного лічильника є схема збігу, за допомогою якої аналізуються стани всіх попередніх молодших тригерів і виробляються функції перенесення згідно з такими логічними співвідношеннями:

$$\begin{aligned} P1 &= U+Q_1; \\ P2 &= U+Q_2Q_1; \\ P3 &= U+ Q_3Q_2Q_1; \\ P4 &= U+ Q_4Q_3Q_2Q_1. \end{aligned} \qquad (1.19)$$

При надходженні чергового лічильного імпульсу *U*+ перемикаються тільки ті тригери, для яких усі попередні (молодші) розряди знаходяться в цей момент в одиничному стані.

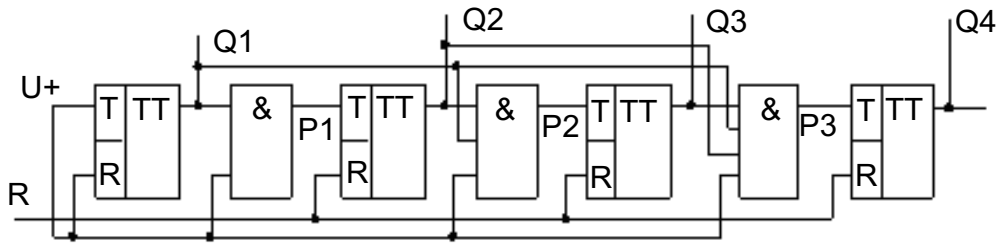


Рисунок 1.37 – Схема підсумовувального лічильника з паралельними перенесеннями

При побудові багаторозрядних синхронних лічильників з'являються труднощі – зростання числа входів вентилів у колі перенесення і збільшення навантаження на виходи тригерів.

У двійковому лічильнику віднімання кожний імпульс *U*- зменшує стан на одиницю.

У лічильниках віднімання сигнали міжрозрядного зв'язку називаються позиками. За правилом двійкового віднімання в момент надходження лічильного імпульсу *U*-позика із старшого розряду з одиничним значенням виникає за умови, що всі молодші тригери знаходяться в нульовому стані. Після цього всі вони перемикаються в стан «1», а старші – в стан «0». Сигнали позики утворюються на інверсних виходах двоступеневих тригерів або на прямих виходах тригерів з динамічним керуванням по фронту.

Параметри лічильника віднімання (модуль і ємність лічби, швидкодія) збігаються з аналогічними характеристиками підсумовувальних лічильників.

Схема трирозрядного двійкового асинхронного лічильника віднімання на двоступеневих тригерах показана на рис. 1.38, а.

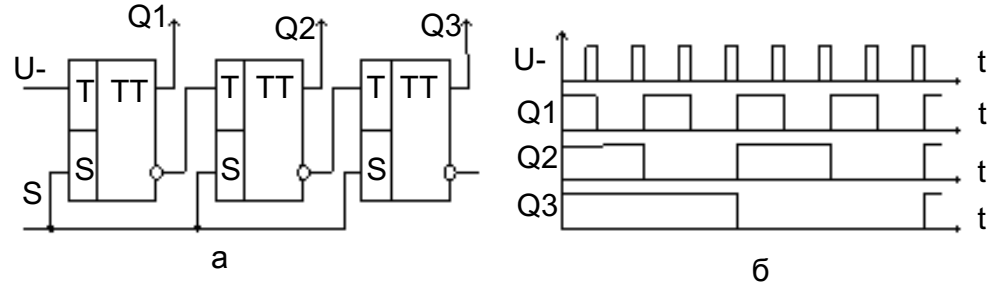


Рисунок 1.38 – Асинхронний лічильник віднімання на двоступеневих тригерах: а) – схема; б) – часові діаграми роботи

Перед початком роботи за допомогою сигналу на спільному вході *S* всі тригери лічильника встановлюються в стан «1», утворюючи вихідний код 111. Імпульс віднімання *U*- надходить на лічильний вхід лише першого молодшого розряду, міжрозрядні сигнали позики знімаються асинхронно з інверсних виходів тригера.

Після надходження семи імпульсів віднімання усі тригери лічильника установлюються в стан «0», утворюючи вихідний код 000. Восьмий імпульс віднімання перемикає лічильник в стан 111 (за умови, що розглядається трирозрядна схема).

У лічильнику віднімання на тригерах з динамічним керуванням по фронту сигнали позики знімаються з прямих виходів тригерів.

Двійкові реверсивні лічильники

Двійкові реверсивні лічильники мають переходи у двох напрямках: в прямому (при лічінні підсумовувальних сигналів *U*+) і в зворотному (при перерахуванні сигналів віднімання *U*-).

Розрізняють одноканальні та двоканальні реверсивні лічильники. В одноканальних реверсивних лічильниках підсумовувальні сигнали *U*+ і сигнали віднімання *U*- почергово надходять на спільний лічильний вхід, а напрямок лічби задається напрямком ланцюгів міжрозрядних перенесень або позик. Для перемикання міжрозрядних зв'язків у одноканальному реверсивному лічильнику потрібні додаткові сигнали керування.

Двоканальні реверсивні лічильники мають два лічильних входи: один для підсумовувальних імпульсів *U*+, другий – для імпульсів віднімання *U*-. Перемикання кіл міжрозрядних зв'язків здійснюється автоматично лічильними сигналами: для перенесень – імпульсами *U*+, для позики – імпульсами *U*-. Схема одноканального трирозрядного двійкового реверсивного лічильника показана на рис. 1.39. Міжрозрядні зв'язки комутуються за допомогою логічних елементів І–АБО.

Для задання напрямку лічіння використовують додатковий *RS*-тригер: з його прямого виходу знімається сигнал керування додаванням $Y_Д$ (вмикає кола перенесення), а з інверсного виходу – сигнал керування відніманням Y_B (вмикає кола позики). На виходах елементів І–АБО (які називаються «схеми реверса») виробляється сигнал T_i для лічильних входів старших розрядів:

$$T_i = Y_Д Q_i \vee Y_B \overline{Q_i}, \ \ i = 1, 2, 3, ..., n. \qquad (1.20)$$

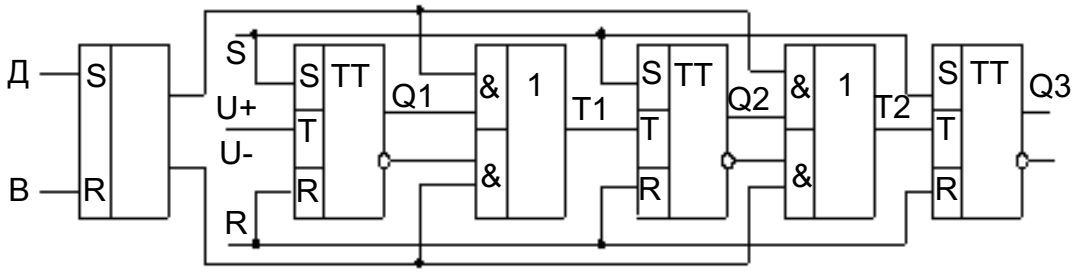


Рисунок 1.39 – Схема одноканального реверсивного лічильника

Таким чином, якщо керувальний *RS*-тригер знаходиться в стані «1», то лічильник реалізовує режим прямого лічіння вхідних імпульсів (тобто підсумовування), в іншому випадку – забезпечує режим зворотного лічіння (віднімання). В обох режимах роботи тригери перемикаються асинхронно.

Схема двоканального чотирирозрядного двійкового реверсивного лічильника показана на рис. 1.40. Лічильні *T*-входи в тригерах внутрішньо пов'язані схемою АБО.

Підсумовувальні імпульси *U*+ надходять на лічильний вхід першого (молодшого) розряду лічильника і одночасно – на входи всіх вентилів у колі паралельного перенесення. При цьому формуються імпульси міжрозрядних перенесень на основі логічних виразів:

$$\begin{aligned} P1 &= U{+}Q1; \\ P2 &= U{+}Q2Q1; \\ P3 &= U{+}Q3Q2Q1; \\ P4 &= U{+}Q_4Q_3Q_2Q_1. \end{aligned} \qquad (1.21)$$

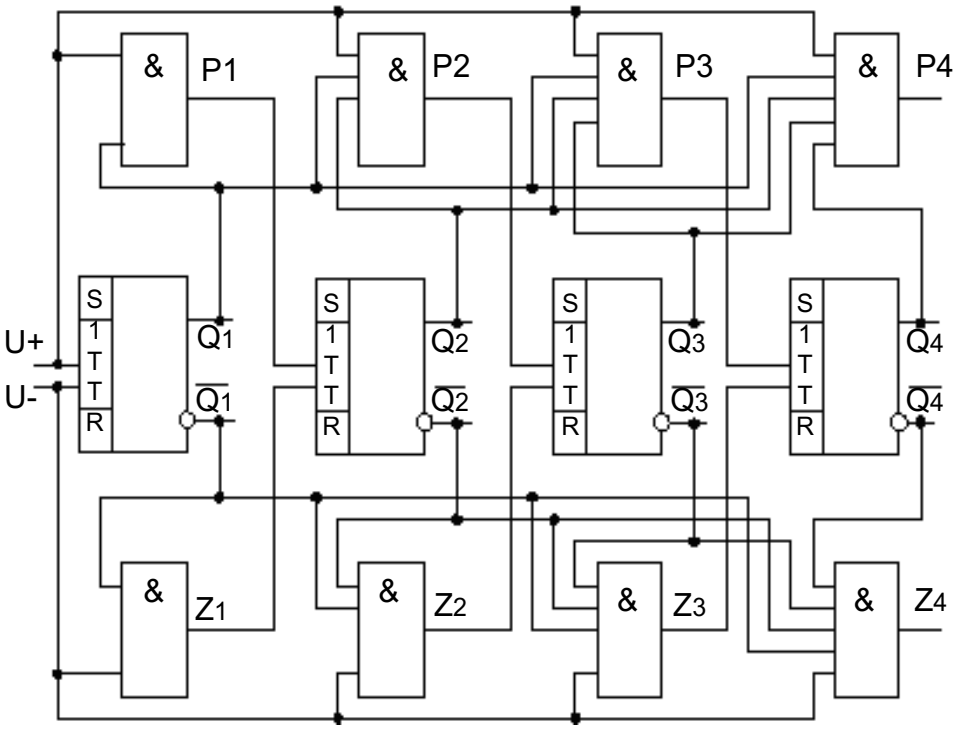


Рисунок 1.40 – Схема двоканального реверсивного лічильника

Імпульси віднімання *U*- надходять на лічильний вхід першого розряду лічильника і одночасно на входи всіх вентилів у колі паралельних позик. При цьому формуються імпульси міжрозрядних позик на основі таких логічних виразів:

$$
\begin{aligned}
Z_1 &= U - \overline{Q}_1;\\
Z_2 &= U - \overline{Q}_1 \cdot \overline{Q}_2;\\
Z_3 &= U - \overline{Q}_1 \cdot \overline{Q}_2 \cdot \overline{Q}_3;\\
Z_4 &= U - \overline{Q}_1 \cdot \overline{Q}_2 \cdot \overline{Q}_3 \cdot \overline{Q}_4.
\end{aligned}
\qquad (1.22)
$$

Таким чином, у двоканальних реверсивних лічильниках напрямок лічби безпосередньо задається підсумовувальними *U+* або віднімальними *U-* імпульсами. Забороняється одночасне надходження на входи двоканального реверсивного лічильника підсумовувальних і віднімальних імпульсів.

На практиці з урахуванням схемотехнічних можливостей мікросхем середнього ступеня інтеграції багаторозрядні реверсивні лічильники будують у вигляді групової структури. При цьому кожна група подається, наприклад, мікросхемою чотирирозрядного реверсивного лічильника з паралельними перенесеннями і позиками. Між групами можуть бути утворені послідовні або паралельні зв'язки.

Двійково-десяткові лічильники

Двійково-десяткові лічильники реалізовують лічбу імпульсів у десятковій системі числення, причому кожна десяткова цифра від нуля до дев'яти кодується чотирирозрядним двійковим кодом (тетрадою). Ці лічильники часто називають десятковими або декадними, оскільки вони працюють з модулем лічби, кратним десяти (10, 100, 1000 і т. д.).

Багаторозрядний двійково-десятковий лічильник будується на основі регулярного кола декад, при цьому перша (молодша) декада має вагу 100, друга – 101, третя – 102 і т. д.

Декада будується на основі чотирирозрядного двійкового лічильника, в якому вилучається надлишкове число станів. Вилучення зайвих шести станів у декаді досягається багатьма способами:

- попереднім записуванням числа 6 (двійковий код 0110);
- після лічіння дев'ятого імпульсу вихідний код дорівнює 1111 і десятковий сигнал повертає лічильник у початковий стан 0110, отже, тут результат лічби фіксується двійковим кодом з надлишком 6;
- блокуванням перенесень: лічба імпульсів до дев'яти здійснюється у двійковому коді, після чого вмикаються логічні зв'язки блокування перенесень; з надходженням десятого імпульсу лічильник закінчує цикл роботи і повертається в початковий нульовий стан;
- введенням обернених зв'язків, які забезпечують лічбу в двійковому коді, примусовим перемиканням лічильника в нульовий початковий стан після надходження десятого імпульсу.

Схема синхронного десяткового лічильника з блокуванням перенесень показана на рис. 1.41. У цій схемі *C*-входи використовуються як лічильні. З надходженням десятого імпульсу на *C*-вхід молодшого розряду *JK*-тригера

обнуляються перший і четвертий розряди та сигналом з виходу Q_4 блокують перемикання другого і третього розряду.

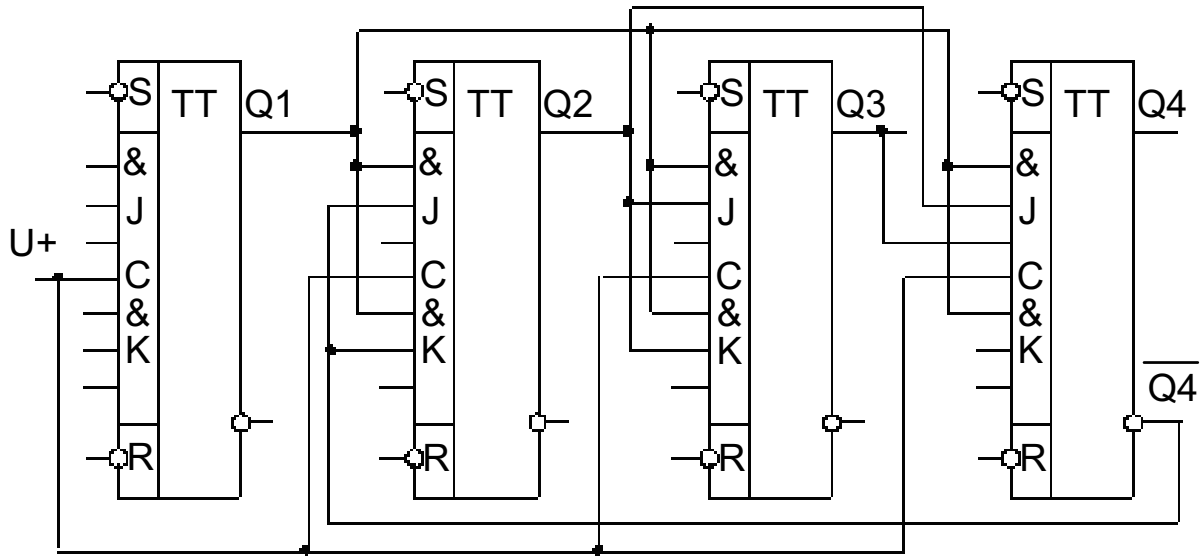


Рисунок 1.41 – Схема десяткового лічильника на *JK*-тригерах

Схема п’ятирозрядного підсумовувального двійково-десяткового лічильника показана на рис. 1.42.

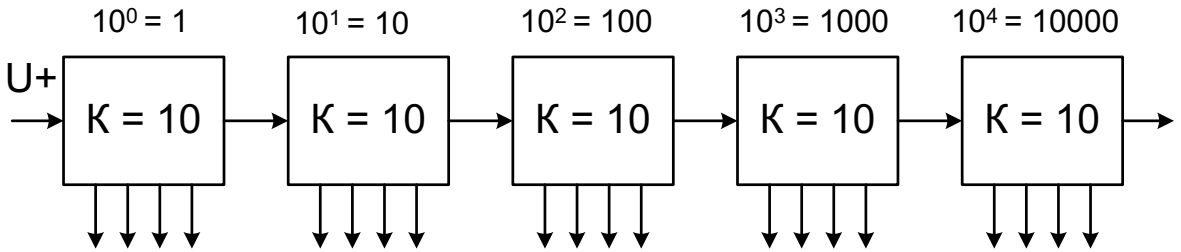


Рисунок 1.42 – Схема п’ятирозрядного підсумовувального двійково-десяткового лічильника

Модуль даного лічильника становить $K_{лч}=105=100000$, ємність лічби $N_{max}=K_{лч}-1=99999$.

Виходи тригерів кожної декади підключаються до входів дешифраторів, які забезпечують візуальну індикацію стану лічильника за допомогою різного роду світлових табло.

Лічильники з одиничним кодуванням

При одиничному (унітарному) кодуванні стани *n*-розрядного лічильника розрізняються лише місцеположенням однієї одиниці, яка називається маркувальним кодом; в інших розрядах записані нулі. В

окремих випадках маркувальний код складається з двох одиниць і називається парно-одиничним.

Лічильник з одиничним кодуванням – це коло тригерів, в якому забезпечується зсув попередньо записаного маркувального коду по «кільцю» в напрямку старших розрядів (прямий підрахунок) або молодших (обернений підрахунок). Такі лічильники часто називають кільцевими (за аналогією з кільцевими регістрами зсуву).

Лічильник з одиничним кодуванням характеризується модулем $КСЧ = n$ і ємністю лічби $N_{max} = n - 1$. Таким чином, число станів кільцевого лічильника дорівнює його розрядності і є значно меншим порівняно з іншими типами лічильників. В кільцевих лічильниках кожний розряд має вагу, яка дорівнює номеру стану *0, 1, 2, ..., (n – 1)*. Із стану (*n – 1*) після надходження чергового імпульсу лічильник утворює на виході сигнал закінчення циклу (переповнення) і повертається в початковий стан за допомогою кола оберненого зв'язку з виходу старшого розряду Q_n на вхід молодшого розряду Q_1. Схема чотирирозрядного кільцевого лічильника показана на рис. 1.43, а.

Перед початком роботи за входом *D* схеми АБО в молодший розряд лічильника записується одиниця і встановлюється початковий код $Q_4Q_3Q_2Q_1=0001$. З надходженням кожного лічильного імпульсу за входом синхронізації одиничний код послідовно зсувається в бік старших розрядів; при цьому молодші розряди, виконані на *D*-тригерах з динамічним керуванням, обнуляються.

Після надходження четвертого імпульсу лічильник повертається в початковий стан за допомогою сигналу з виходу Q_4 на вхід схеми АБО.

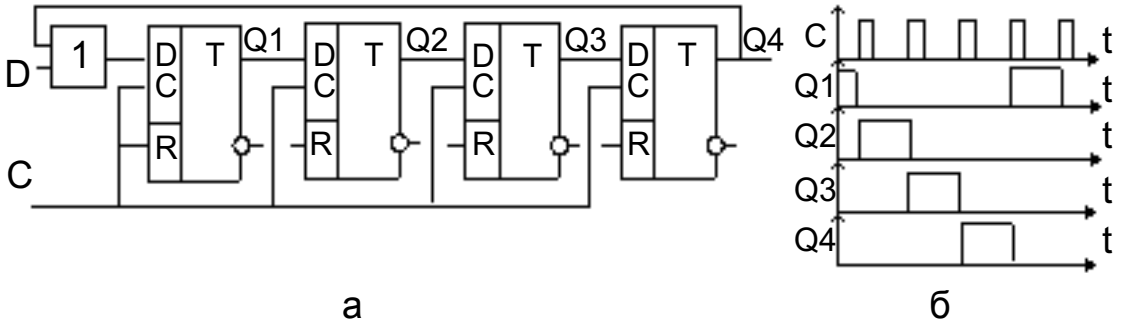


Рисунок 1.43 – Кільцевий лічильник: а – схема; б – часові діаграми

Практичне використання кільцевих лічильників пояснюється такими їх перевагами:

- не потребує вихідного дешифратора, оскільки всі стани відрізняються наявністю одиниці лише в одному якому-небудь тригері;

- в процесі лічіння завжди переключається в одиничний стан лише один тригер, що забезпечує мінімальне значення $t_{вст}$;

- спрощується будова схеми контролю лічильника.

Приклади

Синтез лічильників

Синтезувати лічильник із заданим модулем лічби. Нехай необхідно синтезувати лічильник за модулем лічби K=5/7.

$$5 \rightarrow 0101 \rightarrow Q0\overline{Q1}Q2\overline{Q3},$$
$$7 \rightarrow 0111 \rightarrow \overline{Q0}Q1Q2Q3, \quad (1.23)$$

$$m = 5 \; y = 0, \quad (1.24)$$
$$m = 7 \; y = 1.$$

$$K5 = \overline{Y}Q0\overline{Q1}Q2\overline{Q3}, \quad (1.25)$$
$$K7 = Y\overline{Q0}Q1Q2Q3.$$

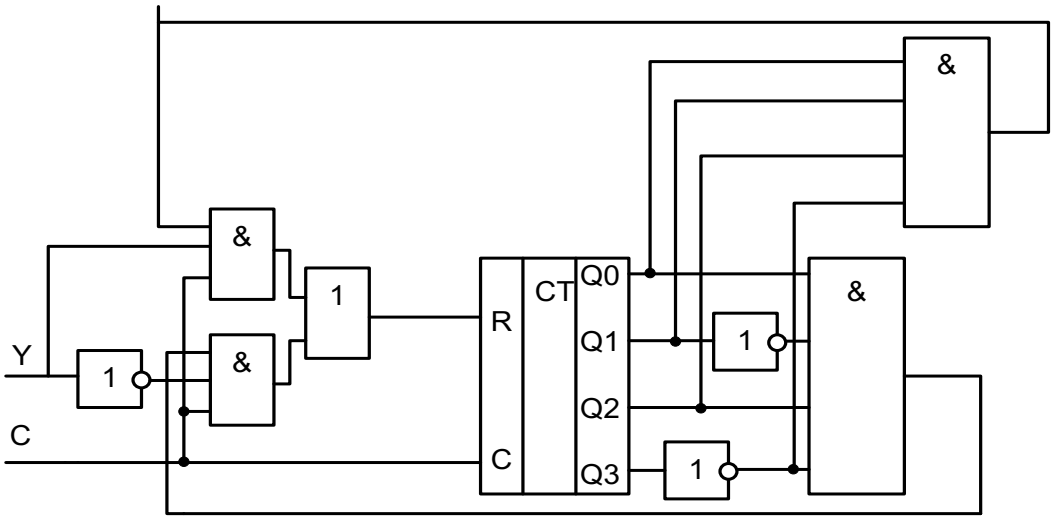


Рисунок 1.44 – Лічильник за mod 5/7

Генератор випадкових чисел на основі лічильника

Нехай необхідно синтезувати генератор випадкових чисел на основі лічильника 3→2→12→8. Для побудови генератора потрібно вибрати мінімальну кількість розрядів. У даному випадку мінімальна кількість розрядів – 4.

Для побудови такого генератора достатньо вибрати 2 тригери, оскільки вони можуть кодувати 4 стани.

Таблиця 1.7 – Таблиця істинності генератора випадкових чисел:

Q1	Q2	C3	C2	C1	C0
0	0	0	0	1	1
0	1	0	0	1	0
1	0	1	1	0	0
1	1	1	0	0	0

Для створення функції вибираємо стани тільки ті, які мають одиничні значення.

$$
\begin{aligned}
&C3 = Q1\overline{Q2} \vee Q1Q2 = Q1;\\
&C2 = Q1Q2;\\
&C1 = \overline{Q1}\,\overline{Q2} \vee \overline{Q1}Q2 = \overline{Q1};\\
&C0 = \overline{Q1}\,\overline{Q2}.
\end{aligned}
\qquad (1.26)
$$

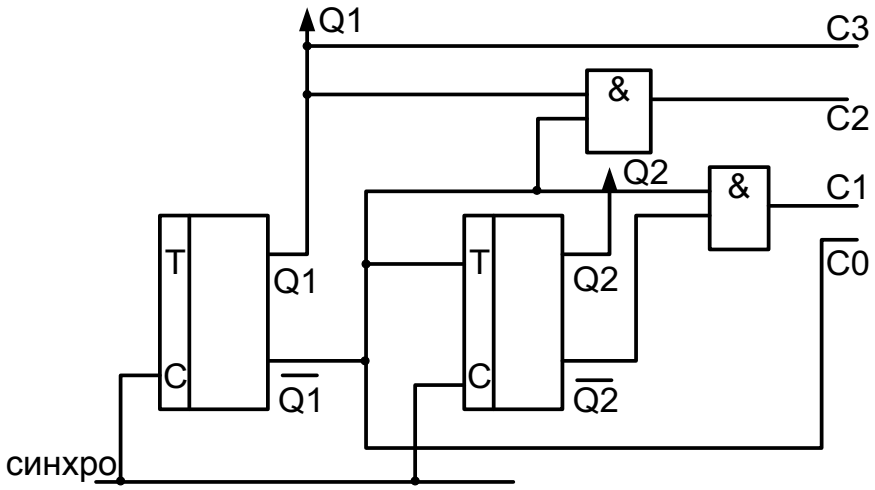


Рисунок 1.45 – Генератор випадкових чисел на регістрі

Синтез двійково-десяткового лічильника із довільною вагою розрядів.

Розглянемо приклад синтезу синхронного двійково-десяткового лічильника на основі тригерів D-типу, який працює в коді з вагою розрядів 5-3-2-1 (замість класичних 8-4-2-1).

Синхронний лічильник на D-тригерах – це лічильник, на тактові входи тригерів якого надходять вхідні імпульси, а на D-входи – сигнали, що керують переходом тригерів до нового стану відповідно до закону функціонування лічильника. Закон функціонування лічильника задамо табл. 1.8, в якій в кожному з десяти станів лічильника поставимо у відповідність значення станів тригерів лічильника, враховуючи, що вага розрядів тригерів відповідно дорівнює 5, 3, 2 і 1. Оскільки зображення десяткових цифр в коді 5-3-2-1 на деяких наборах неоднозначне, слід прагнути до вибору таких варіантів, реалізація яких приводить до менших затрат обладнання, якщо послідовність станів не обумовлена.

В таблиці відмітимо значення сигналів на D-входах тригерів, які необхідно сформувати, щоб забезпечити роботу лічильника згідно з законом функціонування. З логіки роботи D-тригера випливає, що вказані сигнали будуть відповідати значенням тригерів в наступному такті.

Згідно з табл. 1.8 запишемо функції D-входів тригерів залежно від змінних T та проведемо їх мінімізацію за допомогою діаграм Вейча.

Таблиця 1.8 – Таблиця переходів лічильника

Десяткова цифра	Вага розрядів				Функції D-входів			
	5 T_4	3 T_3	2 T_2	1 T_1	D_4	D_3	D_2	D_1
0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	1	0
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	1	0	1
4	0	1	0	1	1	0	0	0
5	1	0	0	0	1	0	0	1
6	1	0	0	1	1	0	1	0
7	1	0	1	0	1	0	1	1
8	1	0	1	1	1	1	0	1
9	1	1	0	1	0	0	0	0

Діаграма станів лічильника відповідно до заданої ваги розрядів і діаграми, які відповідають функціям $D_1 – D_4$, показані відповідно на рис. 1.41. Стани, відмічені знаком X, є надлишковими. Вони можуть використовуватись для спрощення (мінімізації) функцій $D_1 – D_4$, оскільки вважається, що відповідні їм коди ніколи не будуть з'являтися при поданні двійково-десяткових цифр. На наборах, відмічених знаком X, функції $D_1 – D_4$ визначають таким чином, щоб вони були подані в мінімальній формі. Це дозволить скоротити апаратні витрати на реалізацію лічильника.

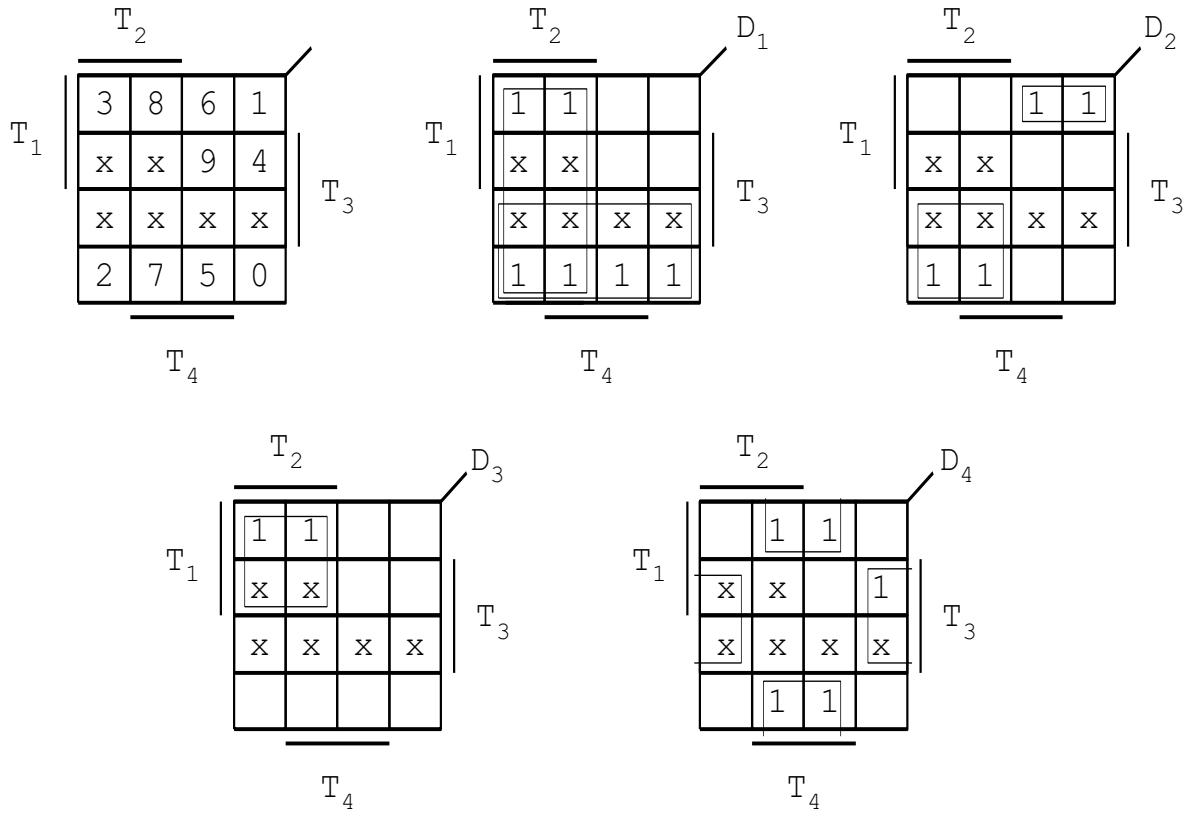


Рисунок 1.46 – Діаграми Вейча для функцій збудження тригерів

Згідно з діаграмами Вейча запишемо значення функцій на D-входах тригерів і перетворимо їх до вигляду, зручного для реалізації на елементах «І-НЕ», «І-АБО-НЕ».

$$D_1 = \overline{T_1} + T_2 = \overline{\overline{\overline{T_1} + T_2}} = \overline{T_1 * \overline{T_2}};$$

$$D_2 = \overline{T}_1 T_2 + T_1 \overline{T}_2 \overline{T}_3 = \overline{\overline{\overline{T}_1 T_2 + T_1 \overline{T}_2 \overline{T}_3}} = \overline{\overline{T}_1 T_2 * T_1 \overline{T}_2 \overline{T}_3};$$

$$D_3 = T_1 * T_2 = \overline{\overline{T_1 * T_2}};$$

$$D_4 = T_3 \overline{T}_4 + \overline{T}_3 T_4 = \overline{\overline{T_3 \overline{T}_4 + \overline{T}_3 T_4}} = \overline{T_3 \overline{T}_4 * \overline{T}_3 T_4} = (\overline{T_3} + T_4) * (T_3 + \overline{T_4}) =$$

$$= \overline{0 + \overline{T}_3 \overline{T}_4 + T_3 T_4 + 0} = \overline{\overline{T}_3 \overline{T}_4 + T_3 T_4}.$$

Реалізуючи вирази функції для D-входів тригерів, отримаємо схему, показану на рис. 1.47. Аналіз цієї схеми показує, що лічильник є синхронним (всі тригери можуть змінювати свій стан одночасно під дією вхідного сигналу), що, згідно з критерієм проектування, забезпечує високу швидкодію. Мінімізація функцій D-входів тригерів забезпечує мінімум апаратних витрат на реалізацію комбінаційної частини лічильника. Наявність обох факторів дозволяє задовольнити критерій проектування.

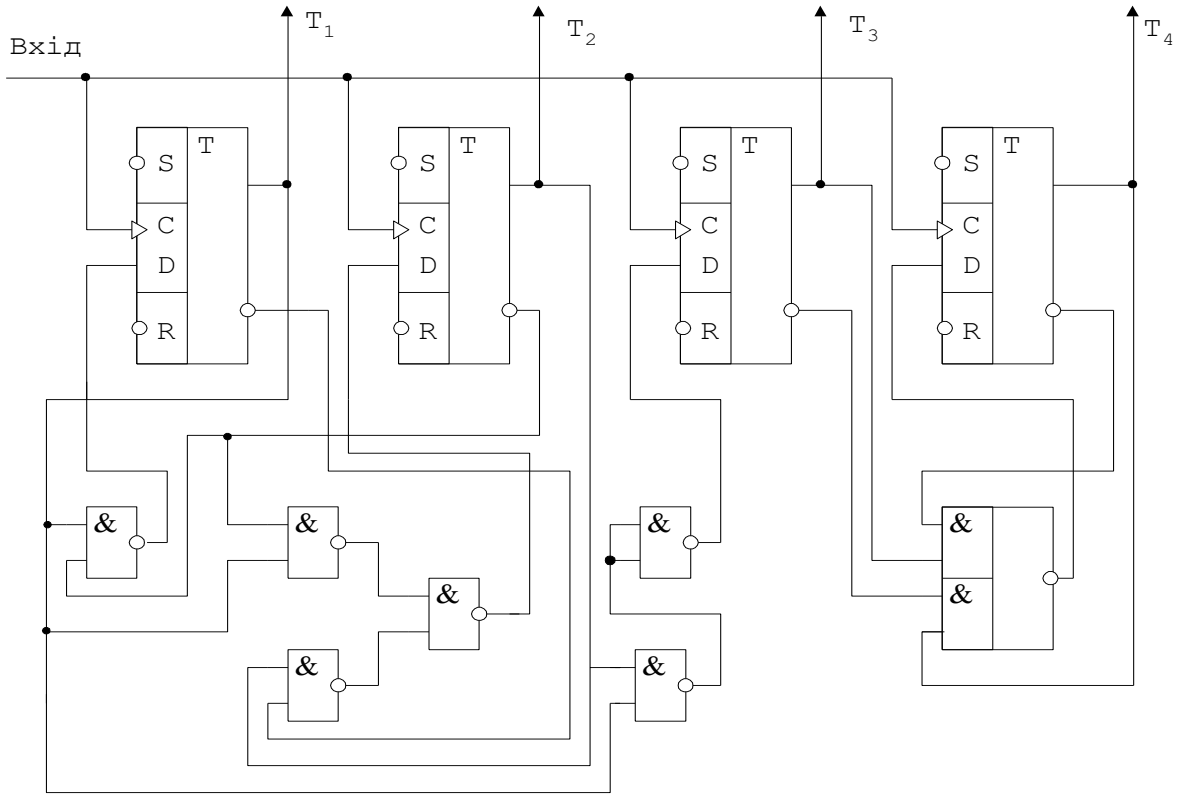


Рисунок 1.47 – Функціональна схема двійково-десяткового лічильника

Ідеалізована часова діаграма роботи синтезованого лічильника, який функціонує відповідно табл. 1.8, показана на рис. 1.48.

Аналіз результатів моделювання полягає у встановленні відповідності одержаних результатів закону функціонування лічильника.

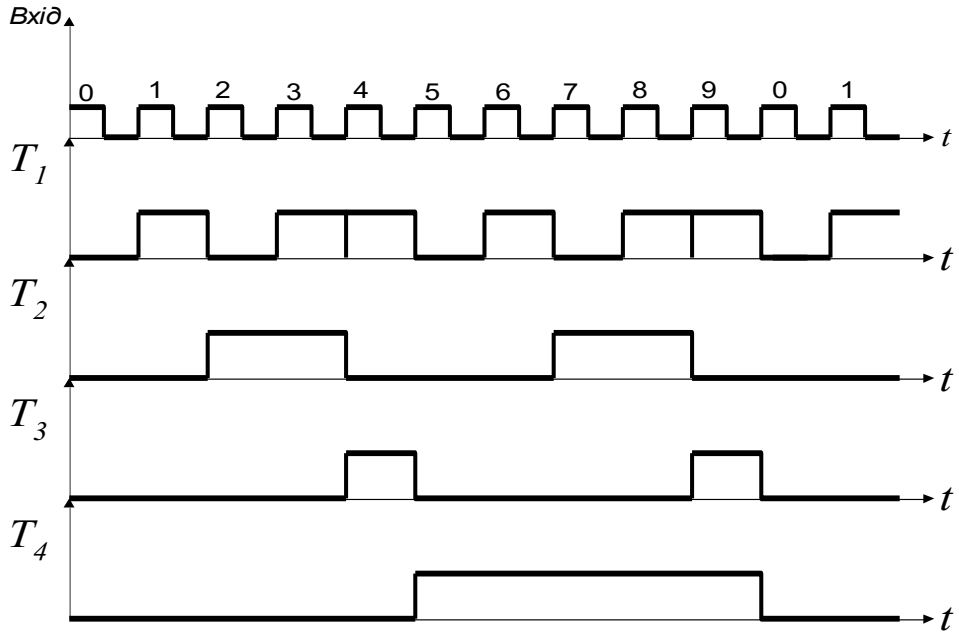


Рисунок 1.48 – Часова діаграма роботи двійково-десяткового лічильника

Контрольні запитання

1. *Тригери, призначення й принцип роботи.*
2. *Наведіть класифікацію тригерів.*
3. *Опишіть динамічні характеристики тригерів.*
4. *Наведіть схему і таблицю станів RS-тригера.*
5. *Наведіть схему і таблицю станів D-тригера.*
6. *Наведіть схему і таблицю станів T-тригера.*
7. *Наведіть схему і таблицю станів JK-тригера.*
8. *В чому різниця між RS- і JK-тригерами?*
9. *Синтезуйте D-тригер на основі SR-тригера.*
10. *Синтезуйте T-тригер на основі RS-тригера.*
11. *Що являють собою регістри?*
12. *Які типи регістрів використовуються в цифровій схемотехніці?*
13. *Які операції виконуються за допомогою регістрів?*
14. *Як відрізняються паралельні, послідовні та комбіновані регістри?*
15. *Наведіть загальну структурну схему паралельного регістра.*
16. *Наведіть загальну структурну схему послідовного регістра.*
17. *Синтезуйте 4-розрядний паралельний регістр.*
18. *Синтезуйте 4-розрядний послідовний регістр.*
19. *Що таке комбінований регістр?*

20. *Що таке лічильник?*
21. *Наведіть основні галузі застосування лічильників.*
22. *Опишіть основні параметри лічильників.*
23. *Охарактеризуйте розрядність лічильника.*
24. *Охарактеризуйте ємність лічильника.*
25. *Охарактеризуйте режими роботи лічильників.*
26. *Наведіть класифікацію лічильників.*
27. *Що являють собою двійкові лічильники?*
28. *Що являють собою реверсивні лічильники?*
29. *Що являють собою двійково-десяткові лічильники?*
30. *Що являють собою лічильники з одиничним кодуванням?*
31. *Опишіть переваги кільцевих лічильників.*
32. *Яким чином відбувається підрахунок в реверсивних лічильниках?*
33. *Синтезуйте 4-розрядний двійковий лічильник.*

2 КОМБІНАЦІЙНІ СХЕМИ ТА ЇХ СИНТЕЗ

2.1 Дешифратори

Дешифратором називається функціональний вузол комп'ютера, призначений для перетворення кожної комбінації вхідного двійкового коду в керувальний сигнал лише на одному із своїх виходів. У загальному випадку дешифратор має n однофазних входів (іноді $2n$ парафазних) і $m = 2^n$ виходів, де n – розрядність (довжина) коду, який дешифрується. Дешифратор з максимально можливим числом виходів $m = 2^n$ називається повним. Функціонування повного дешифратора описується системою логічних виразів вигляду:

$$\begin{aligned} F_0 &= \overline{X_n}\,\overline{X_{n-1}}\ldots\overline{X_2}\,\overline{X_1}; \\ F_1 &= \overline{X_n}\,\overline{X_{n-1}}\ldots\overline{X_2}X_1; \\ &\ldots. \\ F_{m-1} &= X_n X_{n-1}\ldots X_2 X_1, \end{aligned} \tag{2.1}$$

де $X_1 \ldots X_n$ – вхідні двійкові змінні;

$F_0 \ldots F_{m-1}$ – вихідні логічні функції, що являють собою мінтерми (конституенти 1) n змінних.

Індекс функції F_i визначає номер обраного виходу і відповідає десятковому еквіваленту вхідного коду. Вихід, на якому з'являється керувальний сигнал, називається активним. Якщо значення сигналу на активному виході відображається лог. «1», то на решті пасивних виходів встановлюється лог. «0». Двійковий код, який вміщує завжди тільки одну одиницю, а інші – нулі, називається унітарним. Тому дешифратор є перетворювачем вхідного позиційного коду в унітарний вихідний код.

У дешифраторах в інтегральному виконанні стан активного виходу часто відображається значенням лог. «0», а на інших пасивних виходах установлюється лог. «1». Функціонування повного дешифратора з інверсними виходами подається такою системою:

$$\begin{aligned} L_0 &= X_n + X_{n-1} + \ldots X_2 + X_1; \\ L_1 &= X_n + X_{n-1} + \ldots X_2 + \overline{X_1}; \\ &\ldots \\ L_{m-1} &= \overline{X_n} + \overline{X_{n-1}} + \ldots \overline{X_2} + \overline{X_1}, \end{aligned} \tag{2.2}$$

де L_0, L_1, ..., L_{m-1} – вихідні логічні функції, що є макстермами (конституенти 0) n змінних. Індекс функції Li визначає номер вибраного виходу і відповідає десятковому еквіваленту вхідного коду. Між двома видами вихідних функцій існує простий зв'язок: $F_i = \overline{L}_i$. Дешифратори класифікуються за такими ознаками:

- способом структурної організації – одноступеневі (лінійні) і багатоступеневі, в тому числі пірамідальні та прямокутні (матричні);

- форматом вхідного коду – двійкові, двійково-десяткові;

- розрядністю коду, який дешифрується – *2, 3, ..., n*;

- формою подання вхідного коду – з однофазними і парафазними входами;

- кількістю виходів – повні і неповні дешифратори;

- видом вхідних стробувальних сигналів – в прямому або інверсному значеннях;

- типом використовуваних логічних елементів – І-НЕ, АБО-НЕ І, НЕ, АБО і т. д. До основних характеристик дешифраторів відносять: число ступенів (каскадів) дешифрації, кількість використаних логічних елементів або мікросхем, загальне число входів логічних елементів, час дешифрування і споживану потужність. Умовні графічні позначення дешифраторів на електричних схемах показані на рис. 2.1.

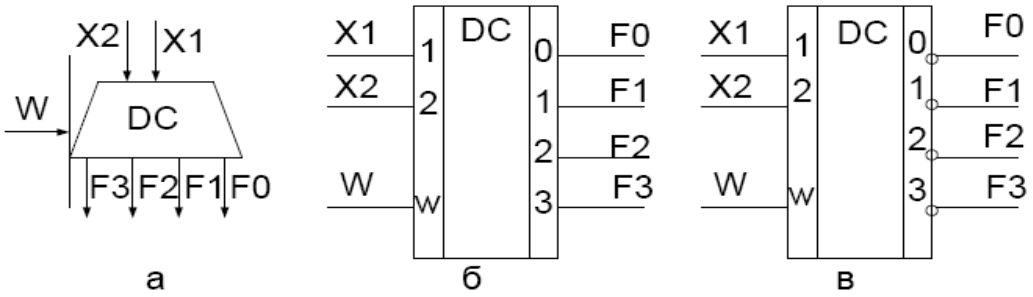


Рисунок 2.1 – Умовні графічні позначення дешифратора:
а) – на функціональних схемах; б), в) – на принципових схемах

Логічна функція дешифратора позначається буквами DC (decoder). Мітки лівого додаткового поля в умовному позначенні відображають десяткові ваги вхідних змінних, а мітки правого додаткового поля відповідають десятковим еквівалентам вхідних комбінацій двійкових змінних. У схему дешифраторів вбудовуються один або два стробувальних (сигнали дозволу) входи, наприклад, *W* (рис. 2.1, б). За допомогою сигналу на вході *W* визначається момент спрацювання дешифратора; крім того, вхід *W* використовується для нарощування розрядності вхідного коду. На практиці повний дешифратор на *n* входів і *m* виходів для стислості називають дешифратором «з *n* в *m*», наприклад, дешифратор «з 3 у 8» – активізується одна з восьми вихідних ліній.

В комп’ютерах дешифратори використовують для виконання таких операцій:

- дешифрування коду операції, записаного в регістр команд процесора, що забезпечує вибір потрібної мікропрограми;

- перетворення коду адреси операнда в команді на керувальні сигнали вибору заданої комірки пам’яті в процесі записування або читання інформації;

- забезпечення візуалізації на зовнішніх пристроях;

- реалізації логічних операцій та побудови мультиплексорів і демультиплексорів. Використання дешифраторів для дешифрування коду операції і адреси операнда, розташованих в регістрі команд процесора, показано на рис. 2.2.

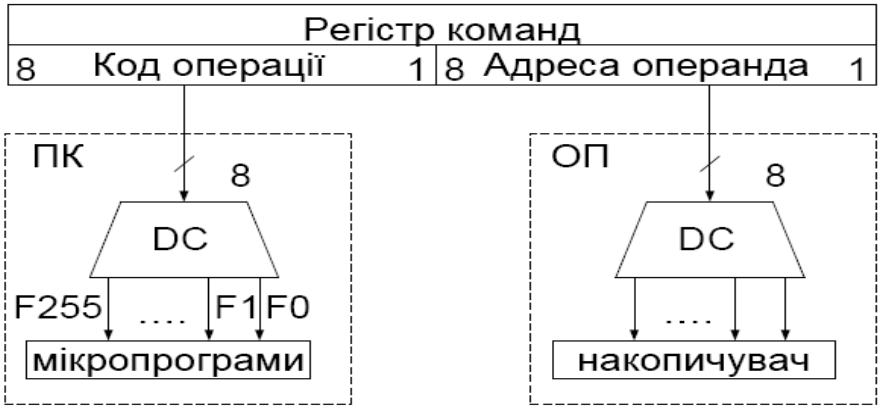


Рисунок 2.2 – Використання дешифраторів

Дешифрування коду операції в пристрої керування (ПК) визначає тип машинної команди. Дешифрування адреси операнда в оперативній пам'яті (ОП) забезпечує доступ до вказаної комірки пам'яті для записування або зчитування даних.

Таблиця 2.1 – Лінійні дешифратори на два входи і чотири виходи

X_2	X_1	F_0	F_1	F_2	F_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

У лінійному дешифраторі «з n в m» кожна вихідна функція Fi реалізовується повністю окремим n-вхідним логічним елементом при використанні парафазного вхідного коду. Логіка роботи повних дешифраторів на два входи X_1, X_2 і чотири прямих виходи F_0, F_1, F_2, F_3 та чотири інверсних виходи L_0, L_1, L_2, L_3 наведена в табл. 2.2

Таблиця 2.2 – Логіка роботи повного дешифратора

X_2	X_1	L_0	L_1	L_2	L_3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

За даними табл. 2.2 отримують систему логічних функцій в ДДНФ:

$$F_0 = \overline{X_2}\,\overline{X_1};\ F_1 = \overline{X_2}X_1;\ F_2 = X_2\overline{X_1};\ F_3 = X_2X_1. \qquad (2.3)$$

Для лінійного дешифратора зі стробувальним входом W система рівнянь (2.3) набуває вигляду:

$$F_0 = \overline{X_2}\,\overline{X_1}W;\ F_1 = \overline{X_2}X_1W;\ F_2 = X_2\overline{X_1}W;\ F_3 = X_2X_1W. \qquad (2.4)$$

Схема лінійних дешифраторів на основі рівнянь (2.3) і (2.4) показана на рис. 2.3.

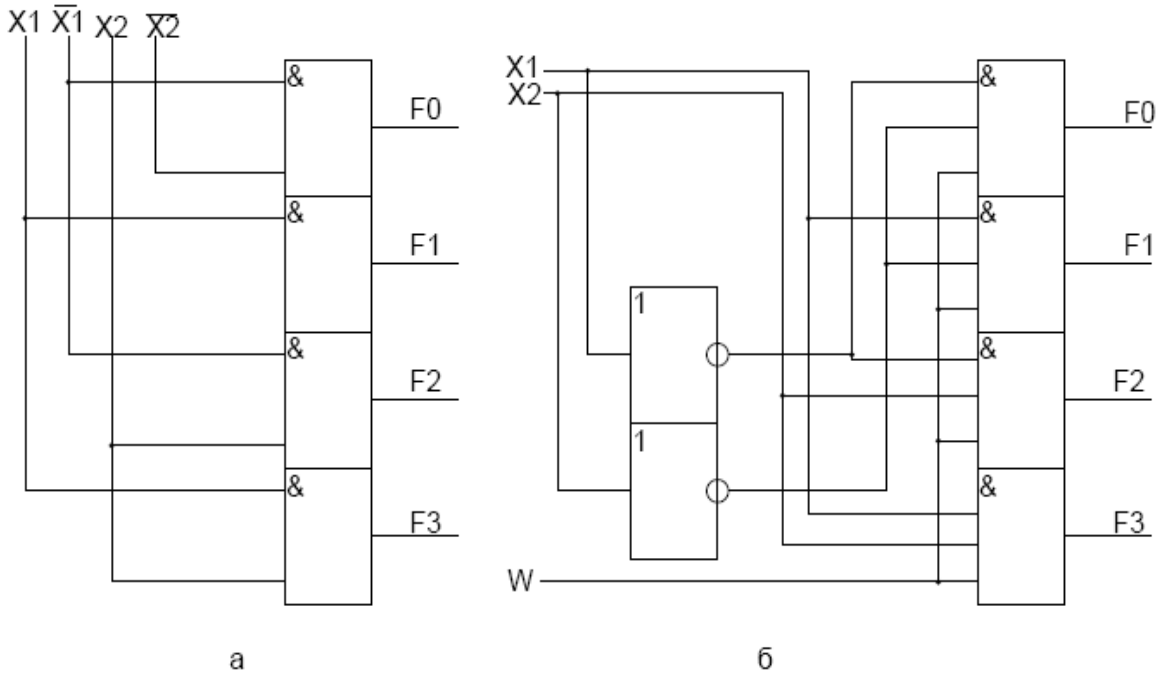


Рисунок 2.3 – Схема лінійних дешифраторів на елементах І:
а) – з парафазними входами; б) – з однофазними входами і стробуванням

Прямокутні дешифратори

Прямокутний дешифратор будується за двоступеневою схемою. При цьому вхідний код розбивається на дві групи по *n/2* розрядів при парному *n;* при непарній розрядності в групах міститься нерівне число змінних. Дві групи змінних декодуються на першому ступені двома повними лінійними дешифраторами, а на другому ступені формуються вихідні функції.

Умовно вважають, що один з дешифраторів першого ступеня формує адреси матриці, а другий – адреси стовпців матриці. На перетині ліній рядків і стовпців підключається $m = 2^n$ двовходових схем збігу, які утворюють другий, вихідний ступінь дешифратора. При парному n матриця вентилів квадратна, при непарному n – прямокутна. Тому такі дешифратори називають матричними або прямокутними.

Запишемо систему вихідних функцій повного дешифратора «з 4 в 16» у вигляді таких скорочених значень:

$$F_0=a_0b_0;\ F_4=a_1b_0;\ F_8=a_2b_0;\ F_{12}=a_3b_0;$$
$$F_1=a_0b_1;\ F_5=a_1b_1;\ F_9=a_2b_1;\ F_{13}=a_3b_1. \quad (2.5)$$

$$F_2=a_0b_2;\ F_6=a_1b_2;\ F_{10}=a_2b_2;\ F_{14}=a_3b_2;$$
$$F_3=a_0b_3;\ F_7=a_1b_3;\ F_{11}=a_2b_3;\ F_{15}=a_3b_3, \quad (2.6)$$

де введені дворозрядні функції a_i і b_i, які реалізуються дешифраторами рядків і стовпців, відповідно:

$$b_0=\overline{X_2}\,\overline{X_1};\ b_1=\overline{X_2}X_1;\ b_2=X_2\overline{X_1};\ b_3=X_2X_1;$$
$$a_0=\overline{X_4}\,\overline{X_3};\ a_1=\overline{X_4}X_3;\ a_2=X_4\overline{X_3};\ a_3=X_4X_3. \quad (2.7)$$

Схема прямокутного дешифратора на основі рівнянь 2.5 … 2.7 показана на рис. 2.4.

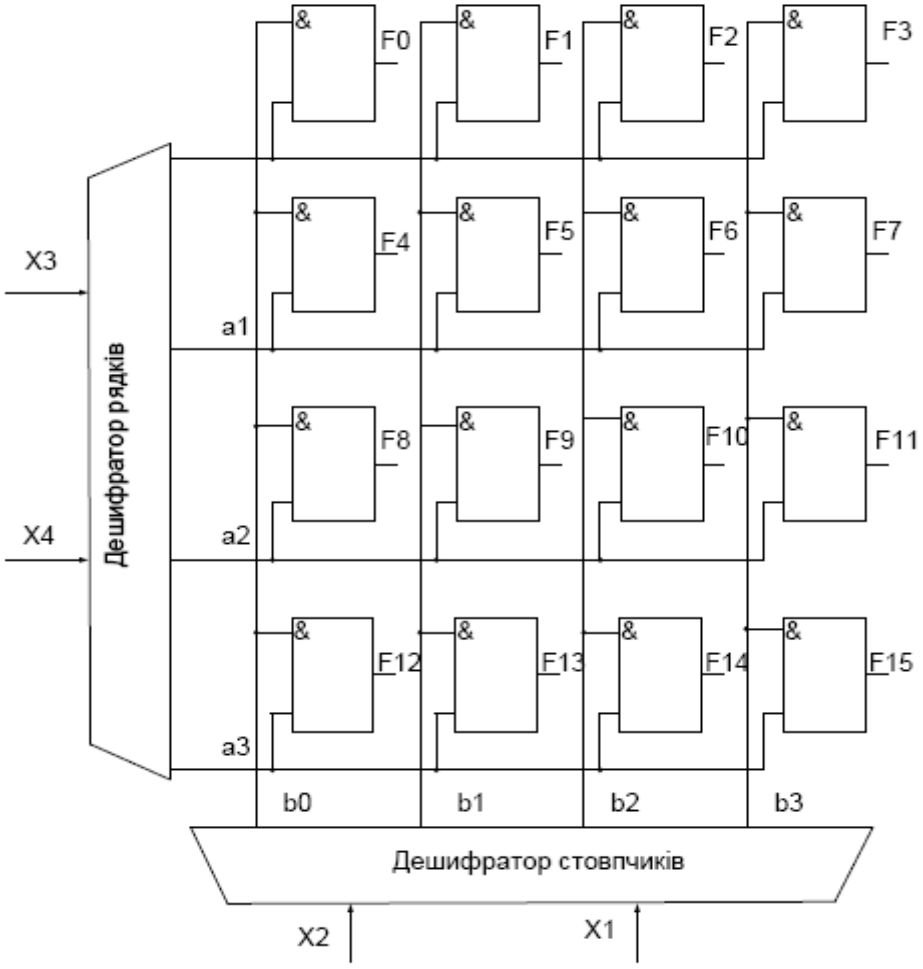


Рисунок 2.4 – Схема прямокутного дешифратора

При великому числі розрядів прямокутний дешифратор майже у *n/2* рази економічніший лінійного.

Багатоступеневі дешифратори. Каскадування дешифраторів

Принцип побудови багатоступеневих дешифраторів полягає у послідовному розбитті вхідного багаторозрядного коду до отримання у кожній групі двох – трьох розрядів. Як приклад на рис. 2.5 показано розбиття коду, який дешифрується, для $n = 10$ і $n = 13$. Після цього багатоступенева схема дешифратора зображається у вигляді з'єднання ряду лінійних схем.

Під каскадуванням (нарощуванням) розуміють спосіб з'єднання дешифраторів у вигляді мікросхем середнього ступеня інтеграції для одержання більшої розрядності вхідного коду. З'єднання двох трирозрядних дешифраторів типу К5551ДЗ для декодування чотирирозрядного коду показано на рис. 2.5.

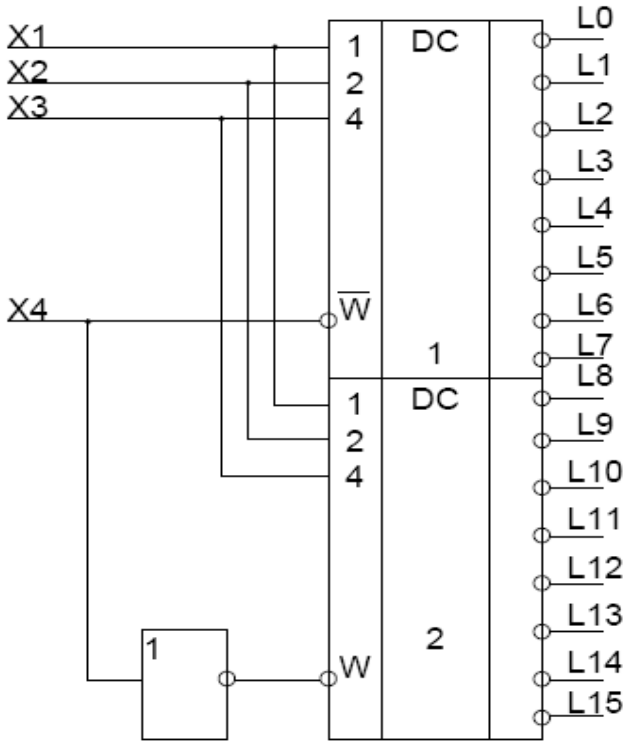


Рисунок 2.5 – Каскадування дешифраторів

Вхідні змінні *X1, X2, X3* подаються паралельно на входи обох дешифраторів: змінна *X4* подається безпосередньо на вхід стропування *W* першого дешифратора, через інвертор – на вхід стробування другого дешифратора. Ця каскадна схема працює так. Якщо значення старшого розряду вхідного коду $X4 = 0$, то в роботу включається перший дешифратор з інверсними вісьмома виходами $L_0, ..., L_7$, при цьому другий дешифратор блокований (вимкнений) і на його виходах $L_8,..., L_{15}$ встановлюються високі рівні. При $X4 = 1$ блокується перший дешифратор і включається в роботу друга мікросхема.

Таким чином, через наявність стробувального входу два трирозрядні дешифратори утворюють схему дешифрування чотирирозрядного коду.

Приклад

Синтезувати дешифратор на 7 входів і 128 виходів з використанням дешифратора К1533ИД7 на 3 входи і 8 виходів

Даний дешифратор має 3 входи та відповідно 8 виходів, а також вхід для стробувального сигналу. Вихідний сигнал з'являється тільки при наявності логічної одиниці на стробувальному вході. Нижче наведена таблиця істинності для даного дешифратора.

Таблиця 2.3 – Таблиця істинності дешифратора К1533ИД7

Стробуваль-ний сигнал	Вхідний код			Вихідний код							
En	X2	X1	X0	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0

Синтезуємо пірамідальний дешифратор $7 - 2^7$ на базі дешифратора $3 - 2^3$ (тобто n = 7, p = 3). Дешифратор $7 - 2^7$ буде містити у вихідному каскаді $2^{n-p} = 2^{7-3} = 16$ дешифраторів $3 - 2^3$. Тому попередній каскад буде містити $2^{n-2p} = 2^{7-6} = 2$ дешифратори $3 - 2^3$. Кожен вихід дешифраторів $3 - 2^3$ цього каскаду підключений до входу розширення En одного із дешифраторів вихідного каскаду. На адресні входи вихідного каскаду дешифратора подаються розряди A0, A1, A2 вхідної адреси, а на адресні входи попереднього каскаду – розряди A3, A4, A5. Вибірка одного з двох дешифраторів попереднього каскаду виконується старшим розрядом A6 вхідної адреси.

Дешифратор DC17 керується за входом En сигналом $\overline{A6}$, а дешифратор DC18 – сигналом A6.

На наступному рисунку зображена схемна реалізація дешифратора на 7 входів та 128 виходів.

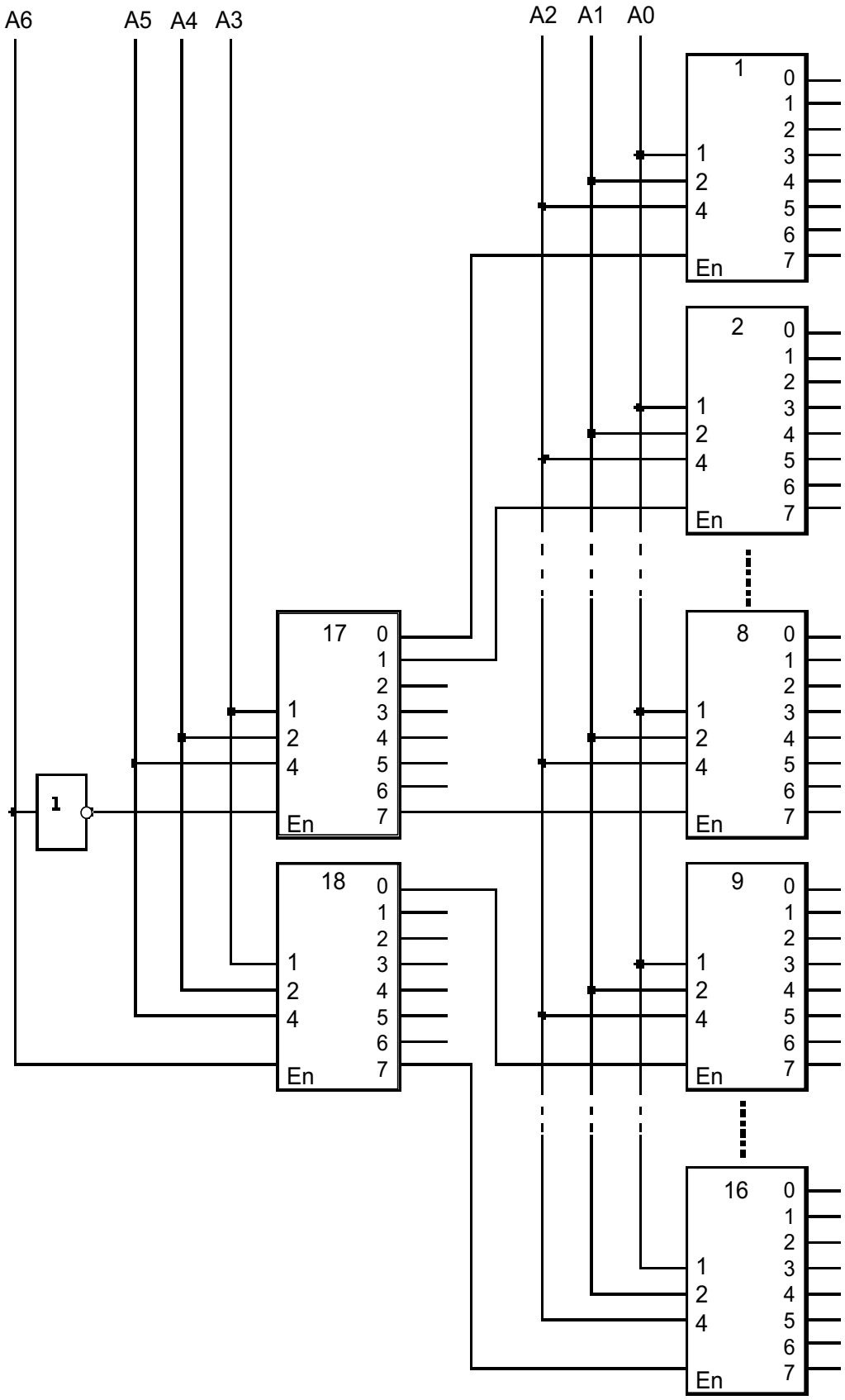


Рисунок 2.6 – Схемна реалізація дешифратора 7 – 128

2.2 Шифратори

Шифратором називається функціональний вузол комп'ютера, призначений для перетворення вхідного *m* – розрядного унітарного коду у вихідний *n* – розрядний двійковий позиційний код. Двійкові шифратори виконують функцію, обернену функції дешифратора. При активізації однієї з вхідних ліній дешифратора на його виходах формується код, який відображає номер активного входу. Умовні графічні позначення шифраторів на схемах показані на рис. 2.7.

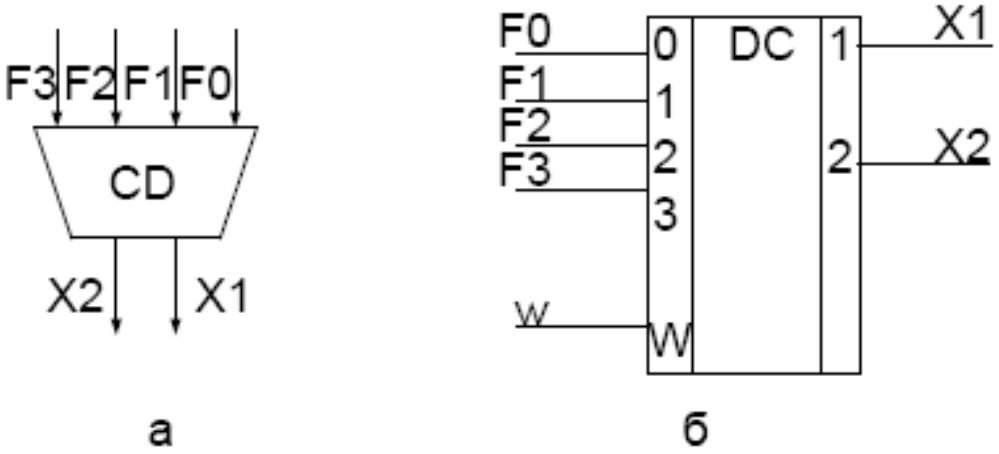


Рисунок 2.7 – Умовні позначення шифратора: а) – на функціональних схемах; б) – на принципових схемах

Функція шифратора позначається буквами *CD* (coder). Входи шифратора нумеруються послідовними десятковими цифрами *0, 1, ..., m-1*, а позначки виходів відображають ваги вихідних двійкових змінних *1, ..., 2^{n-1}*.

У цифрових пристроях шифратори використовуються для таких операцій: перетворення унітарного вхідного коду у вихідний двійковий позиційний код; введення десяткових даних з клавіатури; покази старшої одиниці в слові; передавання інформації між різними пристроями при обмеженому числі ліній зв'язку.

Пріоритетний шифратор клавіатури

Одне з основних застосувань шифратора – введення даних з клавіатури, наприклад, десяткових цифр. Натискання клавіші з десятковою цифрою *0, 1,..., 9* дають приводи до передачі в цифровий пристрій двійково-десяткового коду цієї цифри. Для цього використовується неповний шифратор «*з 10 в 4*».

Шифратори, які при одночасному натисканні декількох клавіш створюють код тільки старшої цифри, називаються пріоритетними. Пріоритетні шифратори, які призначені для пошуку старшої (лівої) одиниці в слові та формування на виході двійкового номера шуканого розряду, називаються покажчиками старшої одиниці. Їх застосовують у

пристроях нормалізації чисел з плаваючою крапкою, в системах з пріоритетним обслуговуванням запитів на переривання роботи комп'ютера.

Логіка роботи пріоритетного шифратора на вісім входів наведена в табл. 2.4, де прийняті такі позначення: F_0, F_1, ... F_7 – вхідні інверсні сигнали, записані в порядку зростання пріоритету: F_0 – найнижчий, F_7 – найвищий; X_3, X_2, X_1 – вихідний інверсний позиційний код; W – стробування; P – функція, яка вказує на находження вхідного сигналу; V – функція, яка вказує на відсутність вхідних сигналів.

Таблиця 2.4 – Таблиця станів входів і виходів шифратора

$\overline{W}$	$\overline{F_7}$	$\overline{F_6}$	$\overline{F_5}$	$\overline{F_4}$	$\overline{F_3}$	$\overline{F_2}$	$\overline{F_1}$	$\overline{F_0}$	$\overline{X_3}$	$\overline{X_2}$	$\overline{X_1}$	$\overline{P}$	$\overline{V}$
1	x	x	x	x	x	x	x	x	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	1	0	1	1	1	0	1
0	1	1	1	1	1	1	0	x	1	1	0	0	1
0	1	1	1	1	1	0	x	x	1	0	1	0	1
0	1	1	1	1	0	x	x	x	1	0	0	0	1
0	1	1	1	0	x	x	x	x	0	1	1	0	1
0	1	1	0	x	x	x	x	x	0	1	0	0	1
0	1	0	x	x	x	x	x	x	0	0	1	0	1
0	0	x	x	x	x	x	x	x	0	0	0	0	1

У табл. 2.4 значення вхідних змінних праворуч від діагоналі, утвореної цифрами 1, не повинні визначати вихідний код (вони позначені хрестиком). Це пояснюється тим, що сигнал з більшим пріоритетом блокує запити з меншими пріоритетами.

Із табл. 2.4 отримуємо вирази для вихідного коду шифратора $\overline{X_3}, \overline{X_2}, \overline{X_1}$ та функції $\overline{P}$, які відповідно визначають відсутність інформаційних сигналів на всіх виходах та наявність сигналу хоч би на одному вході. Для спрощення виразів використовуємо тотожність $F_i = F_i F_k = F_i + F_k$ та закони де Моргана.

$$\overline{X_3} = \overline{W} + WY_1;$$
$$\overline{X_2} = \overline{W} + WY_1\overline{F_3}\,\overline{F_2} + W\overline{F_7}\,\overline{F_6}F_5 + W\overline{F_7}\,\overline{F_6}F_4;$$
$$\overline{X_1} = \overline{W} + WY_1\overline{F_3}F_2 + WY_1\overline{F_3}\,\overline{F_1} + W\overline{F_7}\,\overline{F_6} + WF_7F_5F_4;$$
$$\overline{P} = \overline{W} + WY_1Y_2;\ \overline{V} = W + \overline{Y_1}\,\overline{Y_2}W + \overline{Y_1} + \overline{Y_2};$$
$$Y_1 = \overline{F_7}\,\overline{F_6}\,\overline{F_5}\,\overline{F_4};\ Y_2 = \overline{F_3}\,\overline{F_2}\,\overline{F_1}\,\overline{F_0}. \qquad (2.8)$$

На основі цих виразів побудована (рис. 5.8) схема пріоритетного шифратора «із 8 в 3». При $\overline{W} = 1$ робота блокується і незалежно від сигналів

на входах маємо на інверсних виходах $\overline{X_3}\,\overline{X_2}\overline{X_1}=111$, $\overline{P}=1$, $\overline{V}=1$. Якщо, наприклад, $\overline{F_6}=0$, $\overline{F_2}=0$, то схема формує на виходах код номера входу із старшим пріоритетом: $\overline{X_3}\,\overline{X_2}\overline{X_1}=001$ або в прямому коді $X_3X_2X_1=110_2=6_{10}$. Активний стан виходу відображається значеннями функцій $\overline{P}=0$, $\overline{V}=1$, які передаються в процесор, а також використовуються при каскадуванні шифраторів. Схема, зображена на рис. 2.8, є аналогом шифратора К555ИВ1.

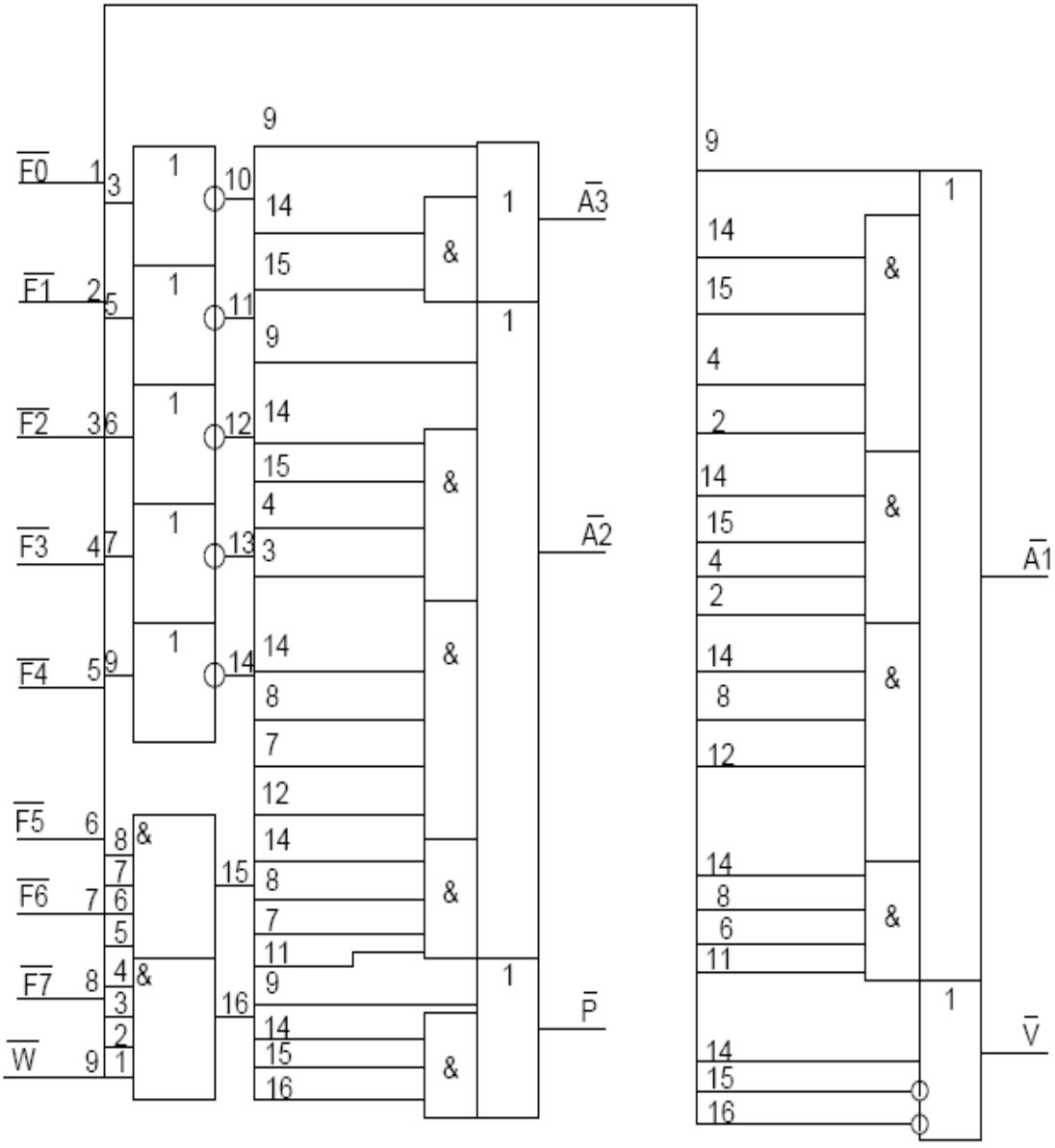


Рисунок 2.8 – Схема пріоритетного шифратора «8→3»

2.3 Мультиплексори

Мультиплексори – це керовані кодом перемикачі декількох інформаційних входів (D) до спільного виходу (Q). Вибір входу здійснюється за допомогою двійкового цифрового коду, який подається на входи адреси (A) і визначає номер входу (D), що з'єднується з

виходом. Існують мультиплексори на 2, 4, 8 і 16 інформаційних входів. На рис. 2.9 наведено приклад мультиплексора, який має 4 інформаційних входи (D0-D3) і відповідно 2 адресних входи (A0, A1).

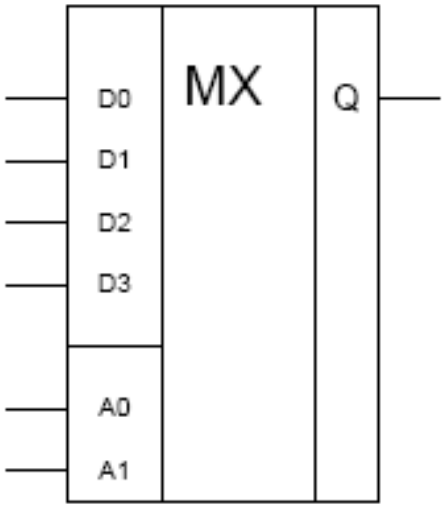


Рисунок 2.9 – Позначення 4-канального мультиплексора

Таблиця 2.5 – Таблиця істинності для 4-канального мультиплексора

A1	*A0*	*Q*
0	*0*	*D0*
0	*1*	*D1*
1	*0*	*D2*
1	*1*	*D3*

Запишемо булеву функцію:

$$Q = D0\overline{A1}\,\overline{A0} + D1\overline{A1}A0 + D2A1\overline{A0} + D3A1A0. \qquad (2.9)$$

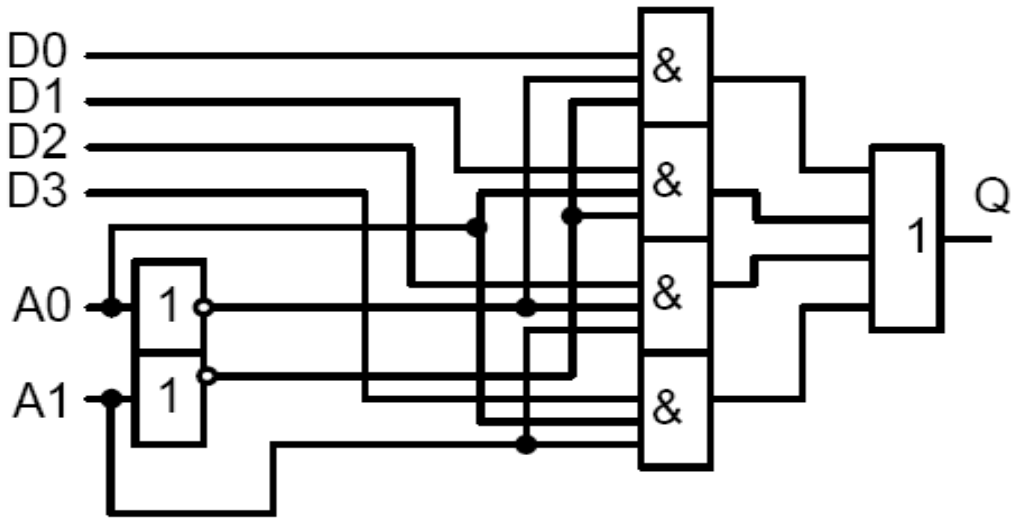


Рисунок 2.10 – Принципова схема 4-канального мультиплексора

Каскадування мультиплексорів

Для збільшення кількості інформаційних входів використовують багатоступінчасте вмикання мультиплексорів (принцип мультиплексного дерева). Для цього виходи мультиплексорів першого рівня підключають до входів мультиплексорів другого рівня. Таким чином, мультиплексори першого рівня керуються молодшими розрядами адресного слова, другого рівня – старшими розрядами. На рис. 2.11 наведено приклад побудови мультиплексора на 16 входів з використанням 4-канальних мультиплексорів.

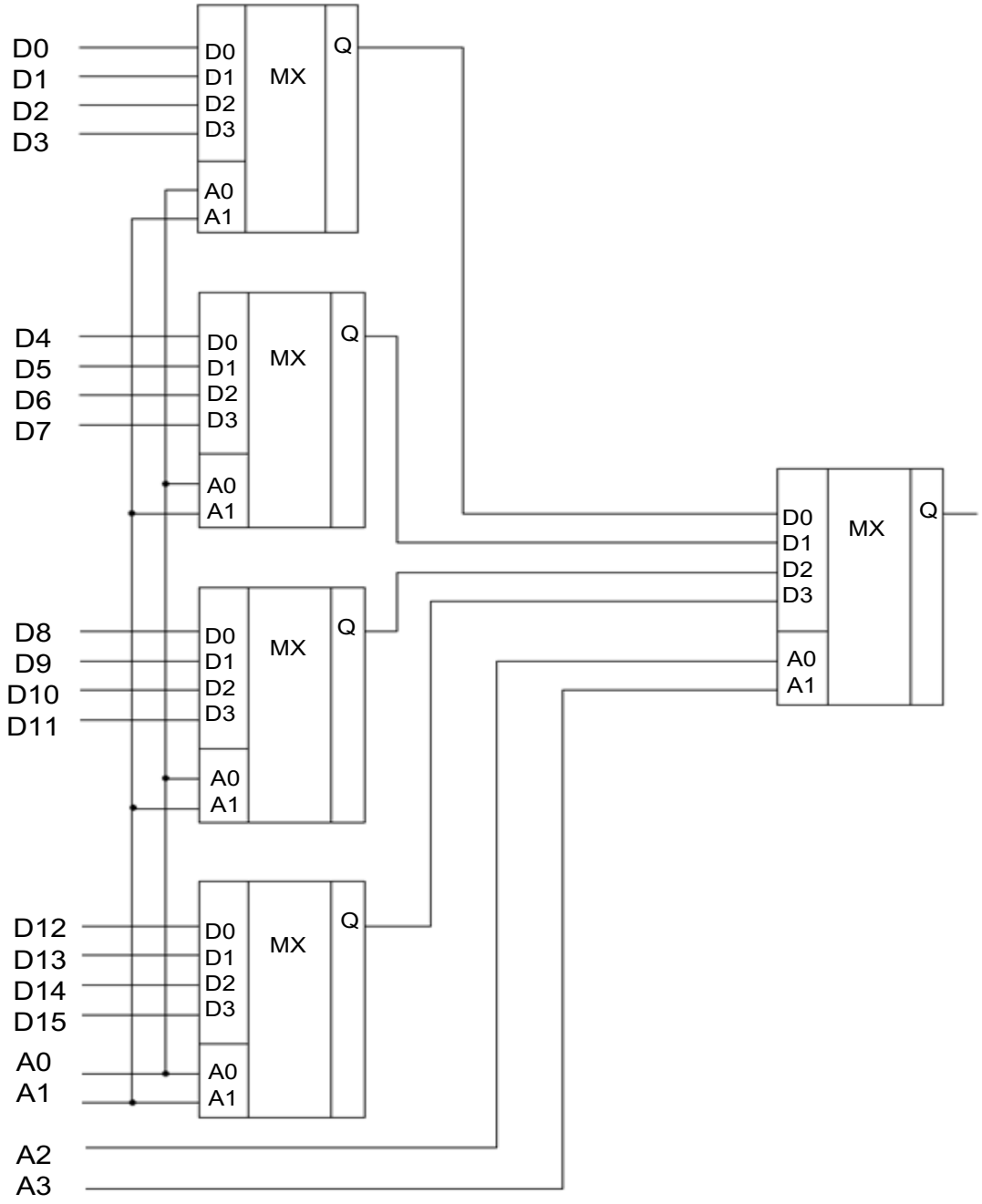


Рисунок 2.11 – Приклад мультиплексорного дерева

Мультиплексор другого рівня, керуючись входами адреси А2, А3, під'єднує до виходу один з 4-х мультиплексорів першого рівня, який в свою чергу, відповідно до входів адреси А0, А1 (які є спільними для мультиплексорів першого рівня), під'єднує до виходу один з 4-х своїх входів.

Застосування мультиплексорів

Мультиплексори можна застосовувати для перетворення паралельного коду в послідовний. Для цього на інформаційні входи подається паралельний код, потім послідовно змінюють код адреси.

Мультиплексор можна застосовувати для створення булевих функцій декількох змінних. Для цього аргументи функції подаються на адресні входи мультиплексора, а інформаційні входи під'єднуються до 0 або 1 відповідно до заданої функції. На рис. 2.12 зображено включення мультиплексора для створення функції 3-х змінних:

$$Y = \overline{x}_3 x_2 \overline{x}_1 + \overline{x}_3 x_2 x_1 + x_3 \overline{x}_2 \overline{x}_1 + x_3 x_2 \overline{x}_1$$

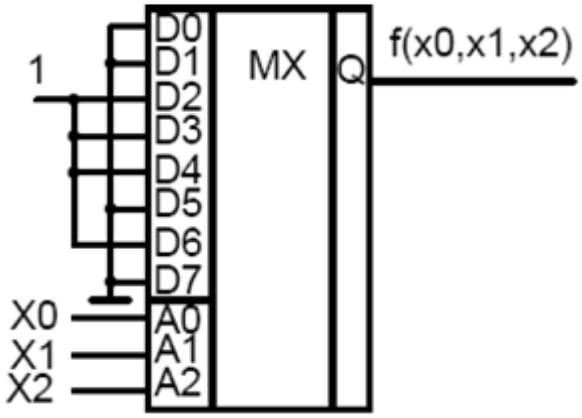


Рисунок 2.12 – Приклад реалізації булевої функції 3-х змінних

Коли на адресні входи потрапляє код 010 (код входу D2), 011 (D3), 100 (D4), 110 (D6), на виході з'являється логічна одиниця. Таким чином на 8-канальному мультиплексорі можна створити булеву функцію 3-х змінних.

Мультиплексори застосовуються для побудови генераторів послідовності логічних станів. Для цього на адресні входи мультиплексора подається двійковий код (наприклад, від лічильника імпульсів), а на інформаційні входи 0 або 1 – залежно від заданої послідовності.

Розглянемо приклад побудови генератора на 8-канальному мультиплексорі. Для отримання імпульсної послідовності (рис. 2.13, а) необхідно на інформаційні входи мультиплексора подати 01100101 (рис. 2.13, б).

Для підвищення функціональних можливостей і каскадування мультиплексорів вводять додатковий сигнал керування E (сигнал стробування), за допомогою якого можна закрити вихід мультиплексора (Q=0) або перевести його в третій стан (стан з високим вихідним опором).

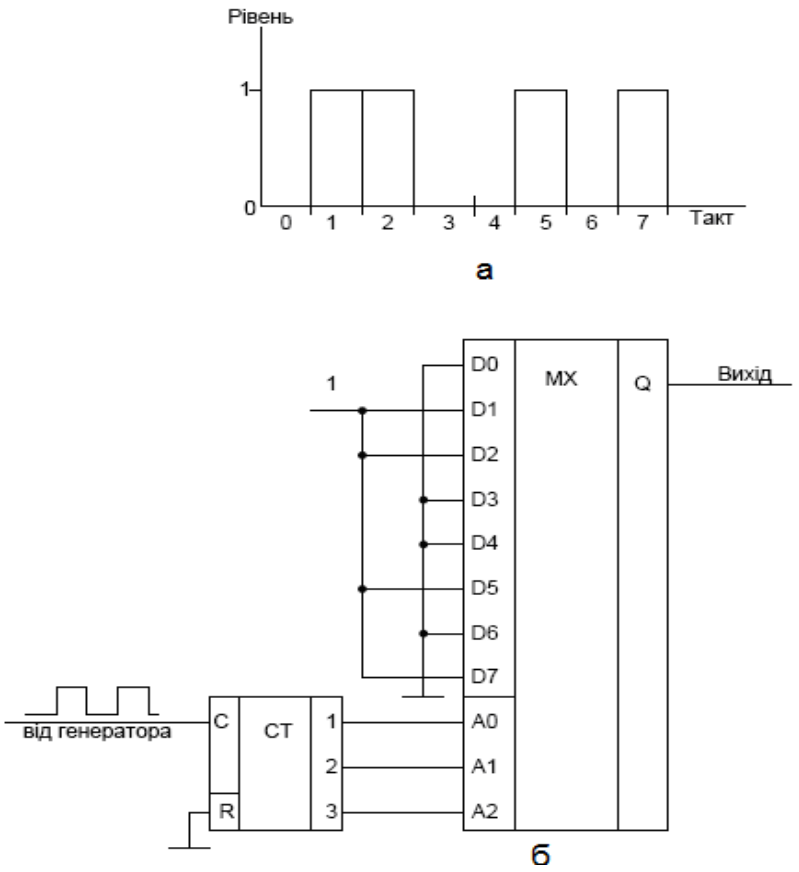


Рисунок 2.13 – Приклад побудови мультиплексного генератора:
а) – послідовність імпульсів; б) – схема

Приклад

Синтезувати функціональний перетворювач на мультиплексорах $Y = X^2$.

Побудуємо таблицю істинності для такого перетворювача

Таблиця 2.6 – Таблиця істинності для перетворювача $Y = X^2$

X	x_2	x_1	x_0	$Y = X^2$	y_5	y_4	y_3	y_2	y_1	y_0
0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	1
2	0	1	0	3	0	0	0	0	1	1
3	0	1	1	9	0	0	1	0	0	1
4	1	0	0	16	0	1	0	0	0	0
5	1	0	1	25	0	1	1	0	0	1
6	1	1	0	36	1	0	0	1	0	0
7	1	1	1	49	1	1	0	0	0	1

Для побудови перетворювача використаємо 6 мультиплексорів, вихід кожного з яких буде визначати відповідний двійковий розряд функції Y (y_0, y_1, y_2, y_3, y_4, y_5).

Адресним входом кожного із мультиплексорів буде двійковий код [x_2, x_1, x_0], що відповідає десятковому значенню аргументу X.

На інформаційні входи кожного мультиплексора подаються постійне значення логічного «0» чи «1» відповідно до таблиці істинності для кожного двійкового розряду числа Y. Наприклад, підключення першого мультиплексора показано на рис. 2.14.

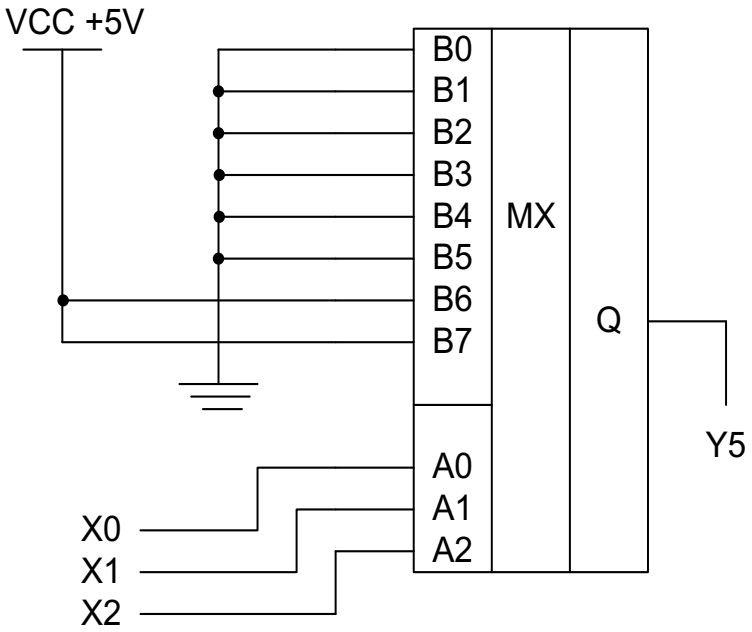


Рисунок 2.14 – Підключення мультиплексора, вихід якого визначає Y5

Підключивши відповідним чином всі шість мультиплексорів, отримаємо таку схему перетворювача.

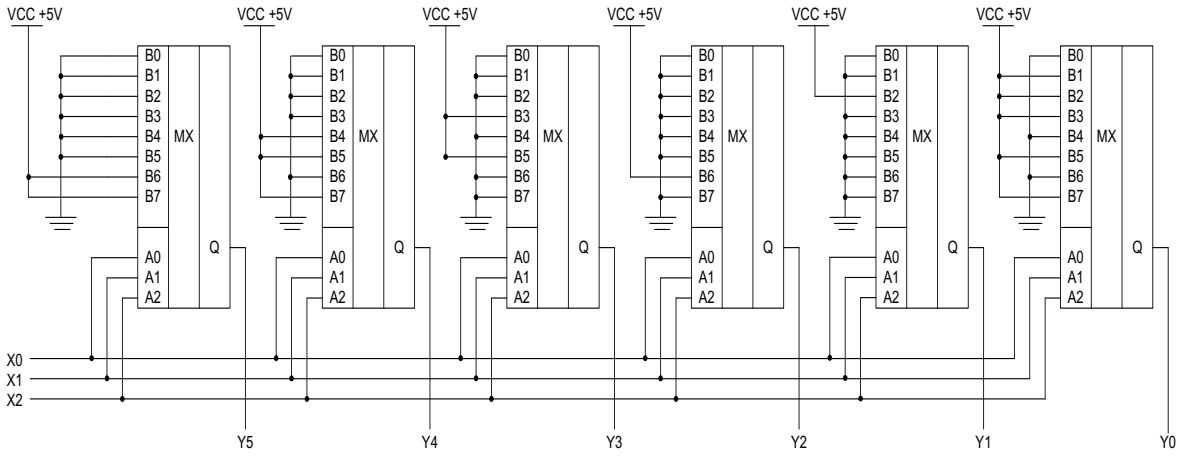


Рисунок 2.15 – Схема синтезованого функціонального перетворювача

2.4 Демультиплексори

Демультиплексори – це керовані кодом перемикачі інформаційного входу (D) до одного з виходів (Q). Вибір виходу здійснюється за допомогою двійкового цифрового коду, який подається на входи адреси (A) і визначає номер виходу (Q), що з'єднується з входом. На рис. 2.16 наведено приклад демультиплексора, який має 4 виходи (Q0-Q3) і відповідно 2 адресних входи (A0, A1).

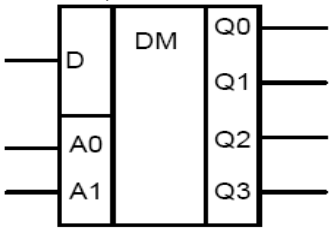


Рисунок 2.16 – Позначення 4-канального демультиплексора

Таблиця 2.7 – Таблиця істинності для 4-канального демультиплексора

A1	A0	Q0	Q1	Q2	Q3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D

Запишемо булеві функції:

$$\begin{aligned} Q0 &= D \cdot \overline{A1A0}, \\ Q1 &= D \cdot \overline{A1}A0, \\ Q2 &= D \cdot A1\overline{A0}, \\ Q3 &= D \cdot A1A0. \end{aligned} \qquad (2.10)$$

На їх основі побудуємо схему 4-канального демультиплексора (рис. 2.17).

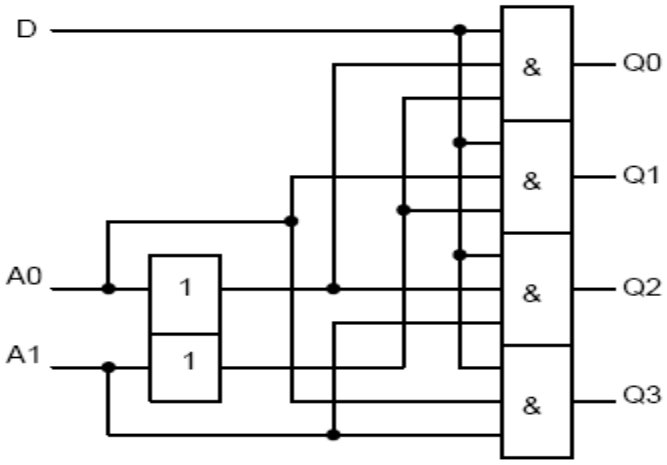


Рисунок 2.17 – Принципова схема 4-канального демультиплексора

Каскадування демультиплексорів

Для збільшення кількості інформаційних виходів використовують багатоступінчасте вмикання демультиплексорів (аналогічно до мультиплексорів). Для цього входи демультиплексорів другого рівня підключають до виходів демультиплексорів першого рівня. Таким чином, демультиплексори першого рівня керуються молодшими розрядами адресного слова, другого рівня – старшими розрядами.

На рис. 2.18 наведено приклад побудови демультиплексора на 16 виходів з використанням 4-канальних демультиплексорів.

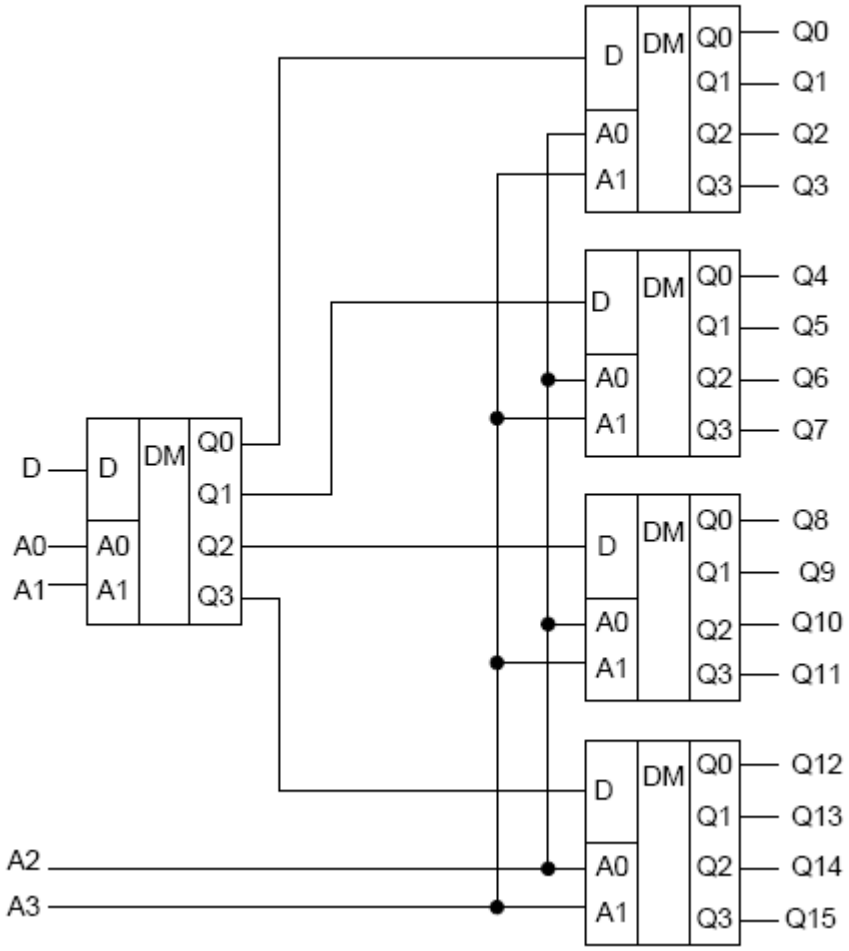


Рисунок 2.18 – Приклад каскадування демультиплексорів

Демультиплексори можна застосовувати для перетворення послідовного коду в паралельний. Для цього на вхід подається послідовний код, потім, послідовно змінюючи код адреси, почергово отримуємо на відповідних виходах розряди паралельного коду.

2.5 Компаратори

Схемою порівняння (компаратором) називається функціональний вузол комп'ютера, призначений для вироблення ознак відношень між двійковими словами (числами). Ознаки відношень записуються у вигляді:

Fi:=A*K або Fi, A*K або FA*K;

Fi: = A*B або Pi, A*B або FA*B, (2.11)

де A і B – двійкові або двійково-десяткові числа; K – двійкова константа; i – номер відношення (часто пропускається); * – операція відношення вигляду =, ≠, <, >, т. ін.; Fi – функція, що задає результат відношення: лог. 1 – якщо відношення виконується, тобто істинне, і лог. 0 – якщо відношення не виконується, тобто помилкове. Функція компаратора позначається буквами COMP (comparator) або знаками =. Основними відношеннями вважаються: «рівність» FA=B, «більше» PA>B і «менше» PA<B. Часто схеми, що реалізовують відношення PA>B або PA<B, називають схемами порівняння «на більше» або «на менше». Маючи в своєму розпорядженні основні ознаки відношень, можна на їхній основі отримати ряд додаткових ознак, наприклад: $FA \neq B = \overline{F}A = B$; $FA \leq B = \overline{F}A > B$; $FA \leq B = FA = BUFA < B$.

Ознаки відношення використовуються як логічні умови (сигнали повідомлень) в мікропрограмах, командах передачі керування, а також у пристроях контролю і діагностики. Після виконання кожної команди в машині автоматично формуються ознаки результатів операції. Ці ознаки, які називаються прапорцями (прапорами), вміщуються в спеціальний регістр прапорців. До прапорців звичайно відносять ознаки нульового результату, переповнення розрядної сітки, знак результату, наявність перенесень із старшого розряду суматора, парне або непарне число одиниць в результаті та ін.

Зазначимо, що формування і використання ознак (прапорців) – це основна відмінність комп'ютера від калькулятора. Тільки за допомогою прапорців машина приймає рішення про хід обчислювального процесу, тобто має інтелектуальні властивості.

Приклади схем для порівняння слова з константою, для порівняння двійкових слів, для порівняння двох слів «НА БІЛЬШЕ» і схеми контролю за парністю наведено в шостому розділі, відповідно підрозділи 6.1, 6.2, 6.3, 6.4.

Застосування компараторів

Контроль (виявлення) і корекція (виправлення) результатів операцій є важливою умовою грамотної експлуатації машин. Контроль може бути програмним або апаратним. До апаратних методів відносяться дублювання операцій і відновлення вхідних сигналів.

Контроль операцій додавання методом дублювання реалізовується двома однаковими суматорами (SM), на входи яких одночасно надходять доданки A(n) і B(n). Обидва результати S1(n) і S2(n) надходять на входи схеми порівняння (рис. 2.19, а). Якщо обидва результати рівні, то на виході схеми порівняння значення ознаки FS1=S2 = 1 і помилок немає. При

нульовому значенні ознаки операцію потрібно повторити або зупинити роботу ЕОМ.

Схема контролю методом відновлення вхідних сигналів показана на рис. 2.19, б. Дворозрядне слово A2A1 декодується і значення унітарного коду з виходів дешифратора поступає на входи шифратора. При правильній роботі дешифратора і шифратора вхідний код A2A1 має збігатися з вихідним кодом шифратора B2B1. При цьому на виході схеми порівняння встановиться одиничне значення ознаки FA=B.

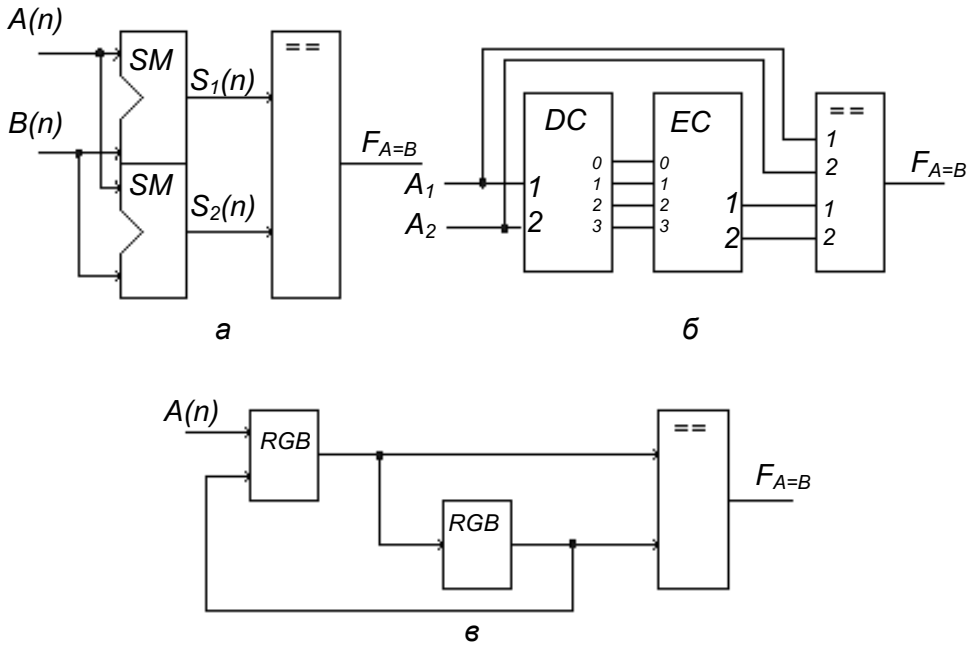


Рисунок 2.19 – Застосування схеми порівняння для контролю операцій

При передаванні інформації з одного регістра в інший контроль правильності пересилання може здійснюватися порозрядним порівнянням вмісту цих двох регістрів. На рис. 2.19, в показано один з варіантів контролю пересилання слів між регістрами. Після передачі інформації з регістра А в регістр В (або навпаки) проводиться порівняння їхнього вмісту. Якщо значення двох слів збігаються, то значення ознаки рівності набуває одиничного значення, інакше – виробляється сигнал помилки.

2.6 Суматори

Суматором називається комбінаційний логічний пристрій, який призначений для виконання арифметичної операції додавання двійкових кодів чисел, і з допомогою простих мікрооперацій, таких як: зміна коду числа, зсув коду числа, запис коду з одного регістра в інший і т. д., можна виконувати операції віднімання, множення і ділення.

Багаторозрядний суматор складається з одноразрядних суматорів, що з'єднані між собою колом перенесень або схемами прискореного утворення і розповсюдження перенесень.

Розрізняють два типи суматорів:

1. Накопичувальні;
2. Комбінаційні.

Накопичувальні будуються на основі тригерів, що працюють в лічильному режимі. Однак вони мають ряд недоліків (потребують нормованих сигналів, поява сигналу перенесення в різний час, необхідність схем часової затримки і т. д.). Тому такі суматори використовуються рідко.

Комбінаційні суматори будуються на логічних елементах. Схеми таких суматорів підлягають синтезуванню.

Однорозрядний двійковий комбінаційний суматор

Позначення одноразрядного двійкового комбінаційного суматора (півсуматора) на два входи зображено на рис. 2.20. Вони використовуються для додавання молодших розрядів кодів чисел.

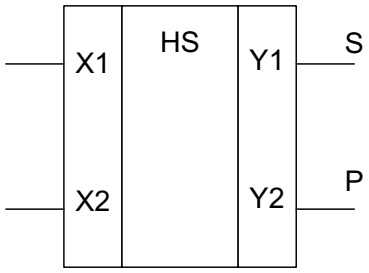


Рисунок 2.20 – Позначення одноразрядного комбінаційного суматора (напівсуматора) на два входи

X1 – вхід наймолодшого розряду першого доданка;
X2 – вхід наймолодшого розряду другого доданка;
Y1(S) – корисний сигнал суми;
Y2(P) – сигнал переносу в старший розряд.

Логічне рівняння суми для півсуматора таке:

$$Y1 = \overline{X1} \cdot X2 + X1 \cdot \overline{X2}. \qquad (2.12)$$

Це логічний елемент нерівнозначності «MOD2». Логічне рівняння перенесення таке:

$$Y2 = X1 \cdot X2. \qquad (2.13)$$

Це логічний елемент «І».

Нижче наведено таблицю істинності півсуматора для додавання однорозрядних двійкових кодів, а також його функціональну схему (рис. 2.21).

Таблиця 2.8 – Таблиця істинності для додавання однорозрядних двійкових кодів

X1	X2	Y1(S)	Y2(P)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

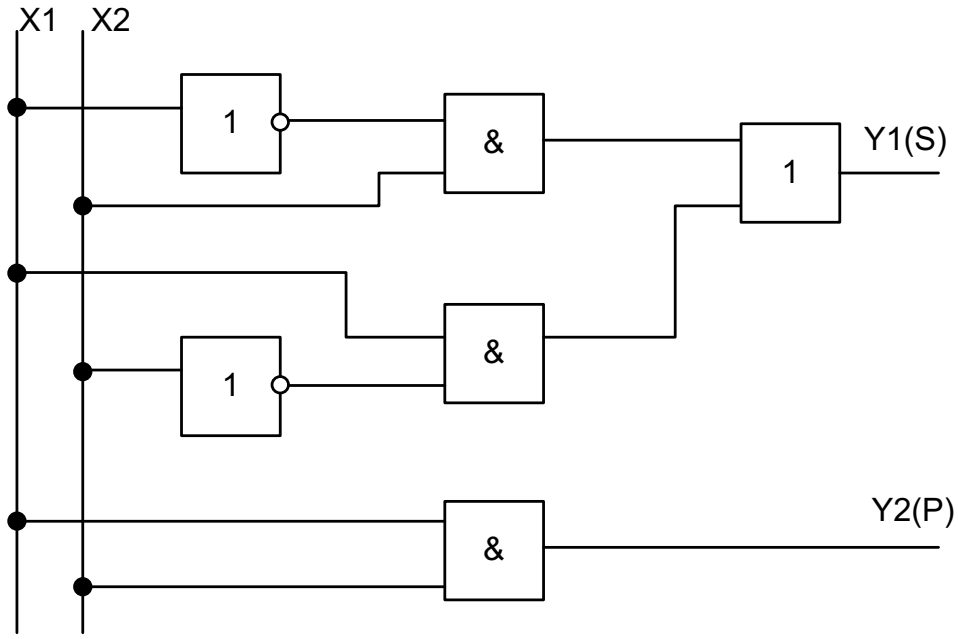


Рисунок 2.21 – Функціональна схема півсуматора

Синтез однорозрядного двійкового комбінаційного суматора на 3 входи

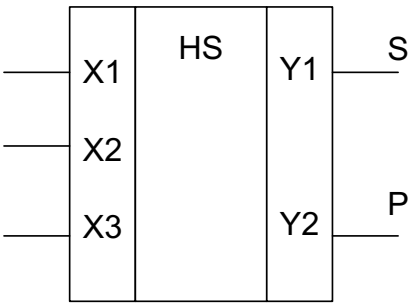


Рисунок 2.22 – Позначення двійкового комбінаційного суматора на три входи

X1 – вхід n-го розряду першого доданка;
X2 – вхід n-го розряду другого доданка;
X3 – вхід сигналу переносу з n-1 в n-й розряд;
Y1(S) – корисний сигнал суми;
Y2(P) – сигнал переносу з n-го в n+1 розряд.

Нижче наведено таблицю істинності для виходів суматора.

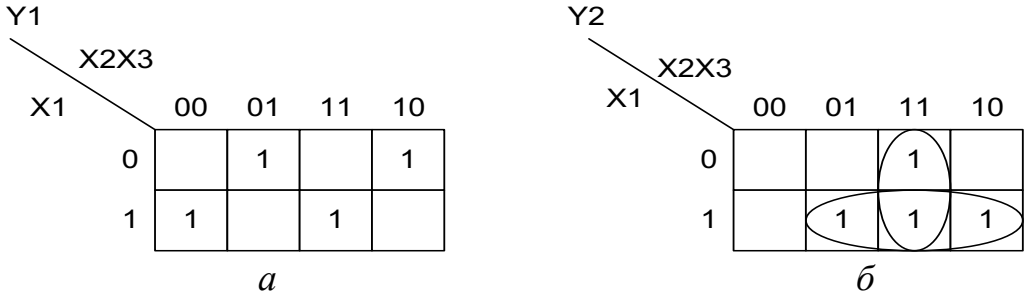
Таблиця 2.9 – Таблиця істинності для суматора на три входи

X1	X2	X3	Y1(S)	Y2(P)
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

Побудувавши за цією таблицею рівняння для виходів суматора, отримаємо:

$$\begin{aligned} Y1 &= \overline{X1}\cdot\overline{X2}\cdot X3 + \overline{X1}\cdot X2\cdot\overline{X3} + X1\cdot\overline{X2}\cdot\overline{X3} + X1\cdot X2\cdot X3, \\ Y2 &= \overline{X1}\cdot X2\cdot X3 + X1\cdot\overline{X2}\cdot X3 + X1\cdot X2\cdot\overline{X3} + X1\cdot X2\cdot X3. \end{aligned} \qquad (2.14)$$

Виходячи з діаграм Вейча (рис. 2.23), для цих рівнянь бачимо, що логічне рівняння суми не мінімізується, а рівняння для сигналу перенесення мінімізується і буде мати вигляд (2.13).



а *б*

Рисунок 2.23 – Діаграми Вейча для рівнянь суми (а) і сигналу перенесення (б)

$$Y2 = X1\cdot X3 + X1\cdot X2 + X2\cdot X3. \qquad (2.15)$$

На рис. 2.24 зображено функціональну схему синтезованого суматора. Наведена схема не є мінімальною. Існує багато схем однорозрядних суматорів з меншою кількістю елементів, наприклад, схема, що

складається з двох півсуматорів, схема в якій використані тільки логічні елементи І-НЕ та інші.

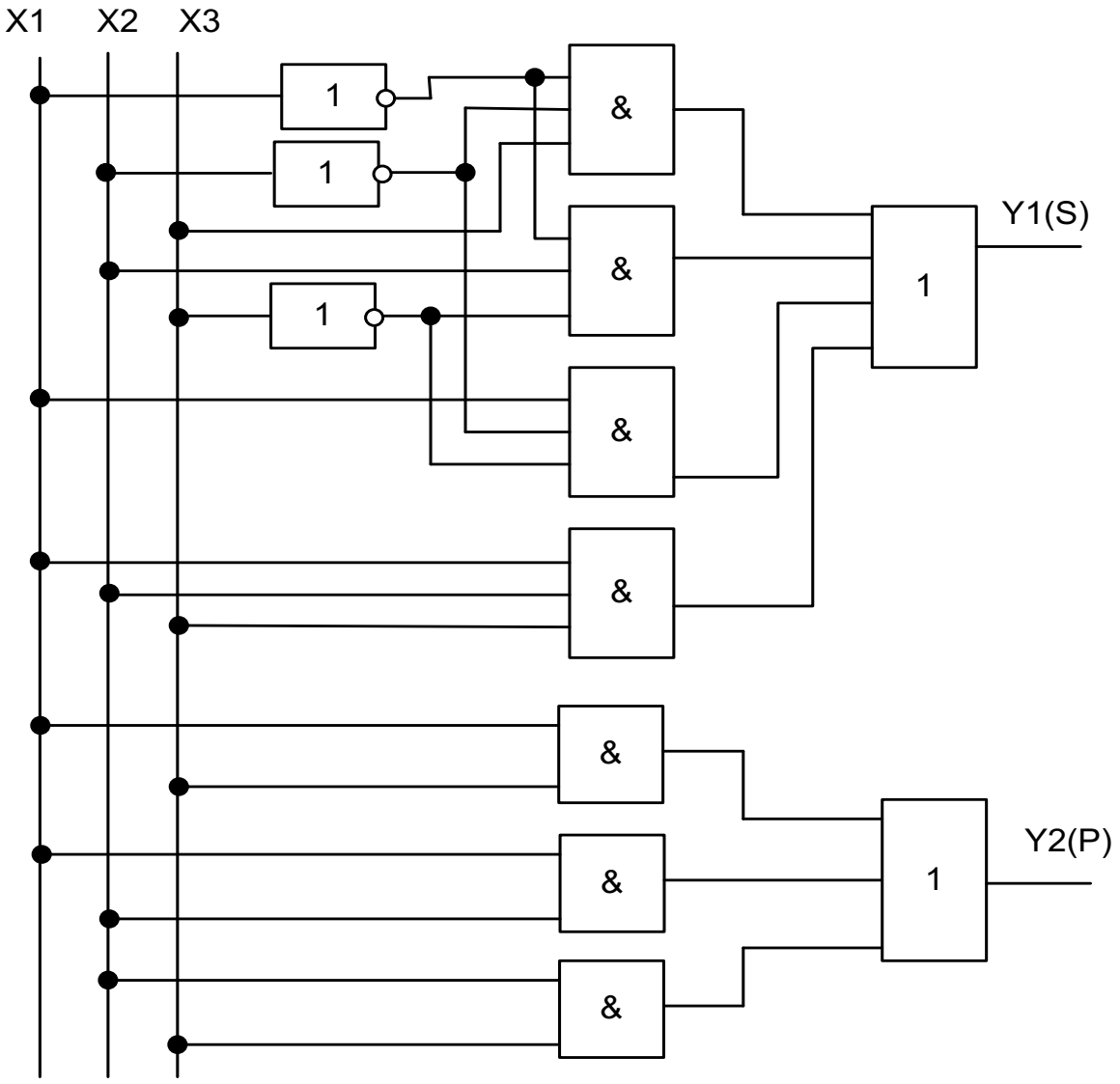


Рисунок 2.24 – Функціональна схема синтезованого суматора на три входи

Багаторозрядний паралельний комбінаційний суматор з послідовним перенесенням

Такий суматор (рис. 2.25) складається з одноразрядних суматорів, які зв'язані між собою колом перенесень. Розряди доданків на такий суматор подаються одночасно і паралельно.

Розряди першого доданка подаються на входи a1, a2, …, an. Розряди другого доданка – на входи b1, b2, …, bn. Поки на вході «подача доданків» діє логічний 0 (низький потенціал) вхідні вентилі подачі доданків закриті. Розряди суми (S1, S2, …, Sn) рівні 0. Подаючи на вхід «подача доданків» сигнал логічної 1, розряди доданків одночасно і паралельно подаються на входи X1 і X2 одноразрядних суматорів. Правильне значення, наприклад,

на S2 і P2 з’явиться тільки тоді, коли на його входах діятимуть сигнали X1, X2, X3. Однак сигнал перенесення з першого однорозрядного суматора з запізненням. Це відбувається тому, що кожен логічний елемент має власну часову затримку.

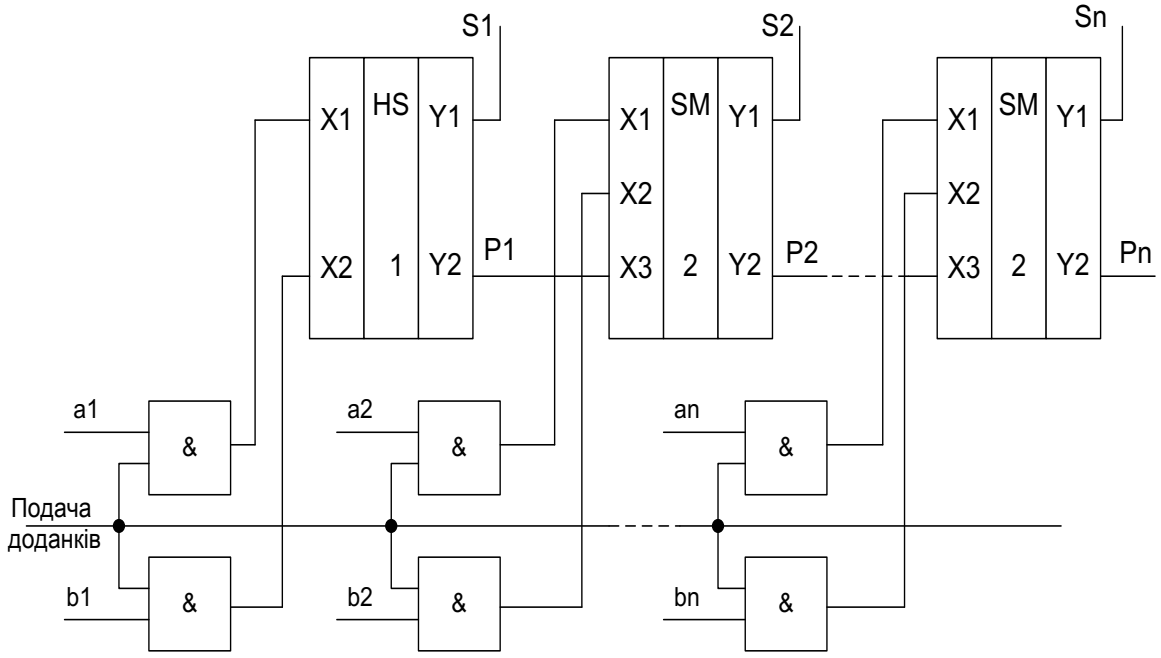


Рисунок 2.25 – Функціональна схема багаторозрядного паралельного комбінаційного суматора з послідовним перенесенням

Розряди доданків подаються одночасно і паралельно, а перенесення утворюються і поширюються послідовно. Найбільш несприятливий випадок коли перенесення проходить по найдовшому колу, тобто від наймолодшого розряду до найстаршого. Для цього потрібен певний час. Певний час потрібен також для запам’ятовування розрядів суми регістром. Це означає, що тривалість сигналу «подача доданків» повинна відповідати цим умовам. Тривалість цього сигналу залежить від розрядності суматора. Чим більша розрядність суматора, тим менша його швидкодія. Існують різні методи прискорення утворення і поширення перенесень (паралельний, груповий, одночасний). Існують також окремі мікросхеми для реалізації цього процесу.

Десятковий суматор

При обробленні десяткових чисел вони подаються в двійково-десятковій формі, тобто для кожного десяткового розряду відводиться тетрада (чотири двійкових розряди). Однак тетрада працює за модулем 16, тобто перенесення з тетрад виникає, коли сума більша 15. Нам необхідно, щоб перенесення виникало, коли сума тетрад більша 9. Тому до тетрад, значення яких більше 9, необхідно додати коректувальне число 6 $(0110)_2$ з врахуванням міжтетрадних перенесень.

Три ознаки корекції суми

Однорозрядний десятковий суматор повинен:

1. Сформувати тетраду початкової суми двох тетрад доданків;
2. Визначити значення тетради початкової суми;
3. Провести, якщо це необхідно, корекцію початкової суми коли значення цієї суми більше 9 відповідно до трьох ознак коректування.

В таблиці 2.10 виділено ознаки корекції суми.

Перша ознака – наявність одиниць в першому і третьому розрядах початкової суми.

Друга ознака – наявність одиниць в другому і третьому розрядах початкової суми.

Третя ознака – наявність перенесень з тетради початкової суми.

Таблиця 2.10 – Ознаки коректування суми

Десяткове значення початкової суми	Тетрада початкової суми			
	3	2	1	0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
16	1	0	0	0
17	1	0	0	0
18	1	0	0	1
19	1	0	0	1

Тетрада першого доданка подається на входи a0, a1, a2, a3.

Тетрада другого доданка – на входи b0, b1, b2, b3. На виходах Y1 суматорів 1, 2, 3, 4 утворюється тетрада початкової суми. Логічна схема, що складається з двох вентилів і логічного елемента АБО працює відповідно до трьох ознак коректування. На виході цієї схеми утворюється сигнал логічної 1, якщо початкову суму необхідно коректувати. Цей сигнал діє на входи X2 суматорів 5 і 6, додаючи 0110_2. На виходах S0, S1, S2, S3 з'явиться тетрада результату в двійково-десятковій формі подання (рис. 2.26).

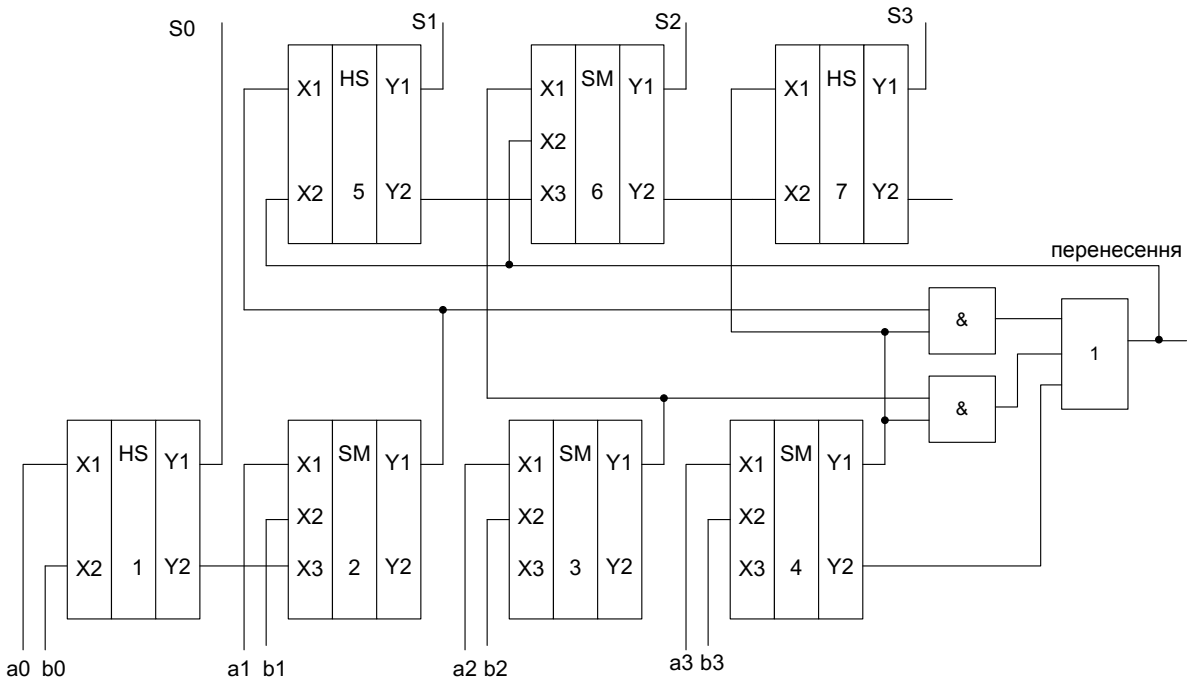


Рисунок 2.26 – Функціональна схема десяткового суматора

Принцип будови накопичувального суматора

Такий суматор використовується у випадку існування багатьох доданків (рис. 2.27).

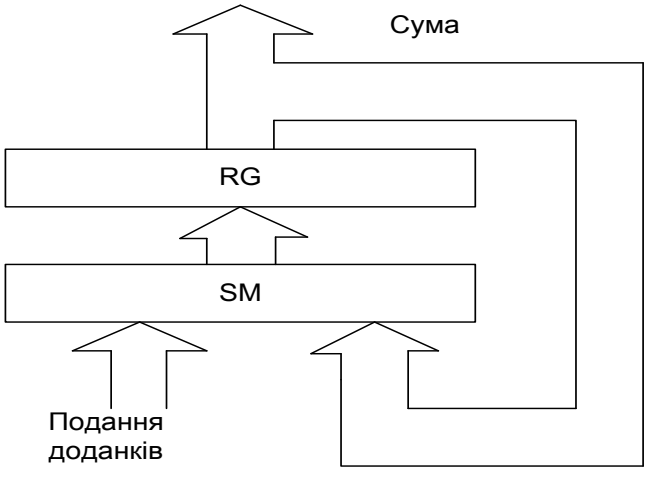


Рисунок 2.27 – Функціональна схема накопичувального суматора

Наступний доданок подається на входи. Подача доданків одночасно з сумою попередніх доданків. Після завершення подання доданків суму можна зчитати з виходів RG суми. SM – багаторозрядний комбінаційний суматор.

Принцип побудови послідовного суматора

Суматори послідовної дії використовуються у випадках, коли швидкодія не є домінуючою, тобто такий суматор має малу швидкодію, однак скорочуються апаратні затрати і вартість. Схема послідовного суматора складається з однорозрядного комбінаційного суматора, трьох зсувних регістрів (два для зберігання доданків і один для зберігання суми) і елемента затримки (D-тригер). За один такт додається один розряд суми, код якої подається молодшими розрядами вперед на старший тригер RG суми і запам'ятовується тригером перенесення, що подається на вхід X3 одночасно з подачею наступних розрядів доданків (рис. 2.28).

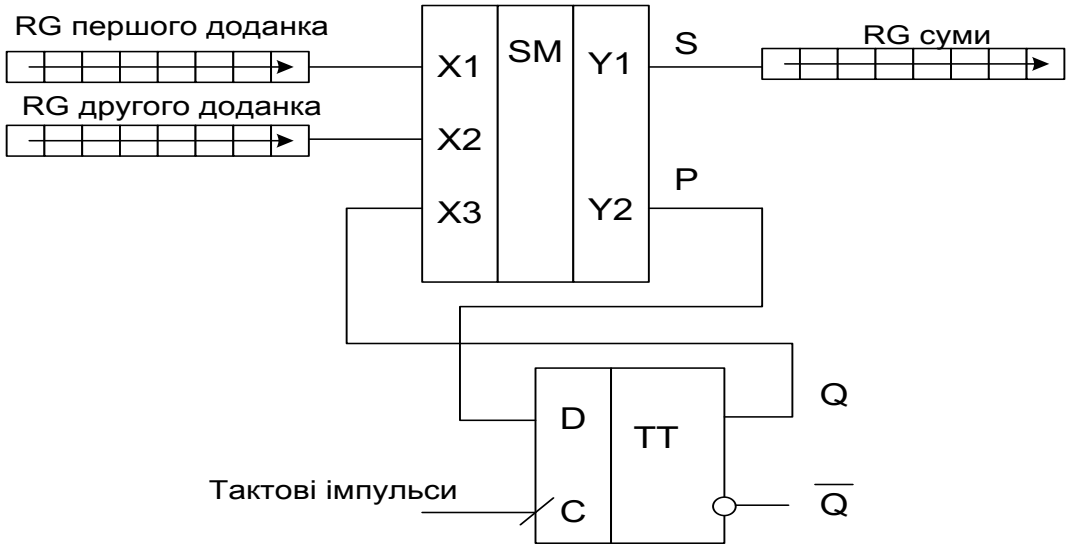


Рисунок 2.28 – Функціональна схема послідовного суматора

2.7 Перетворювач код – код на інтегральних схемах для керування цифровими індикаторами

Для управління цифровими індикаторами у ЦІП первинні коди $N_{(8,4,2,1)}$ або $N_{(1,2,4,8)}$ перетворюються при використанні газорозрядних цифрових індикаторів з десятьма окремими цифрами в одинично-десятковий код $N_{(1-10)}$. Для управління катодолюмінесцентними, рідкокристалічними або напівпровідниковими цифровими індикаторами, в яких десяткові цифри утворюються з семи сегментів, застосовуються інтегральні ПКК з вихідним сімковим, тобто сімково-десятковим, кодом $N_{(7)}$.

Приклад

Синтезувати перетворювач для семисегментного десяткового індикатора (рис. 2.29).

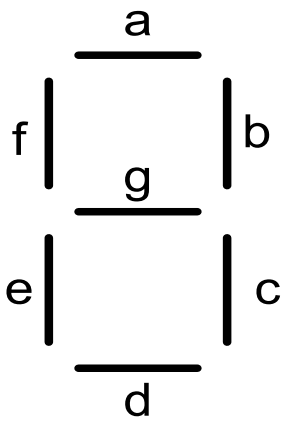


Рисунок 2.29 – Зображення семисегментного індикатора

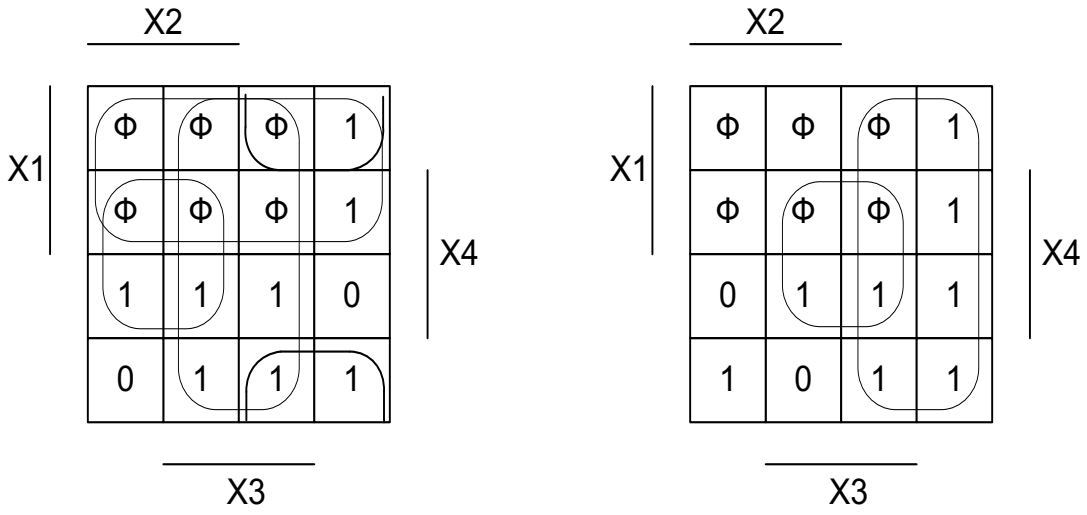
У цьому ПКК вхідним кодом зазвичай є двійково-десятковий код, а вихідним – сімковий чи сімково-десятковий код. Таблиця відповідності для даного ПКК складається з урахуванням рисунка сегментів (табл. 2.11).

Таблиця 2.11 – Таблиця відповідності ПКК для управління семисегментними цифровими індикаторами

i	x_1	x_2	x_3	x_4	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

Вихідний сигнал цього ПКК для управління першим сегментом подається функцією *a*, яка в нормальній диз'юнктивній формі буде мати вигляд:

$$a = \overline{x}_1\overline{x}_2\overline{x}_3\overline{x}_4 \vee \overline{x}_1\overline{x}_2x_3\overline{x}_4 \vee \overline{x}_1\overline{x}_2x_3x_4 \vee \overline{x}_1x_2\overline{x}_3x_4 \vee \overline{x}_1x_2x_3\overline{x}_4 \vee$$
$$\vee\overline{x}_1x_2x_3x_4 \vee x_1\overline{x}_2\overline{x}_3\overline{x}_4 \vee x_1\overline{x}_2\overline{x}_3x_4. \qquad (2.16)$$



а б

Рисунок 2.30 – Діаграми Вейча для мінімізації двох функцій перетворювача двійково-десяткового коду в $N_{(7-10)}$ сімково-десятковий (табл. 2.11), а) – функції *a*; б) – функції *b*

Функцію мінімізуємо за допомогою діаграми Вейча (рис. 2.30, а). Відповідно до цієї діаграми функція мінімізується до вигляду:

$$a = x_1 \vee x_3 \vee \overline{x}_2\overline{x}_3 \vee x_2x_4 . \qquad (2.17)$$

Якщо схема будується на логічних елементах І-НЕ, наприклад, на ЛЕ серії 155, функцію відповідно до теорем Моргана слід перетворити до вигляду:

$$a = \overline{\overline{x}_1\overline{x}_3\,\overline{\overline{x}_2\overline{x}_4}\,\overline{x_2x_4}} . \qquad (2.18)$$

Функції *b* відповідає діаграма (рис. 2.30, б). З діаграми витікає:

$$b = \overline{x}_2 \vee x_3x_4 \vee \overline{x}_3\overline{x}_4 \qquad (2.19)$$

або

$$b = \overline{x_2\overline{x}_3\overline{x}_4\,\overline{\overline{x}_3\overline{x}_4}} . \qquad (2.20)$$

Аналогічно отримуємо рівняння для решти сегментів індикатора.

Дана система функцій після перетворень зображена на рис. 2.31.

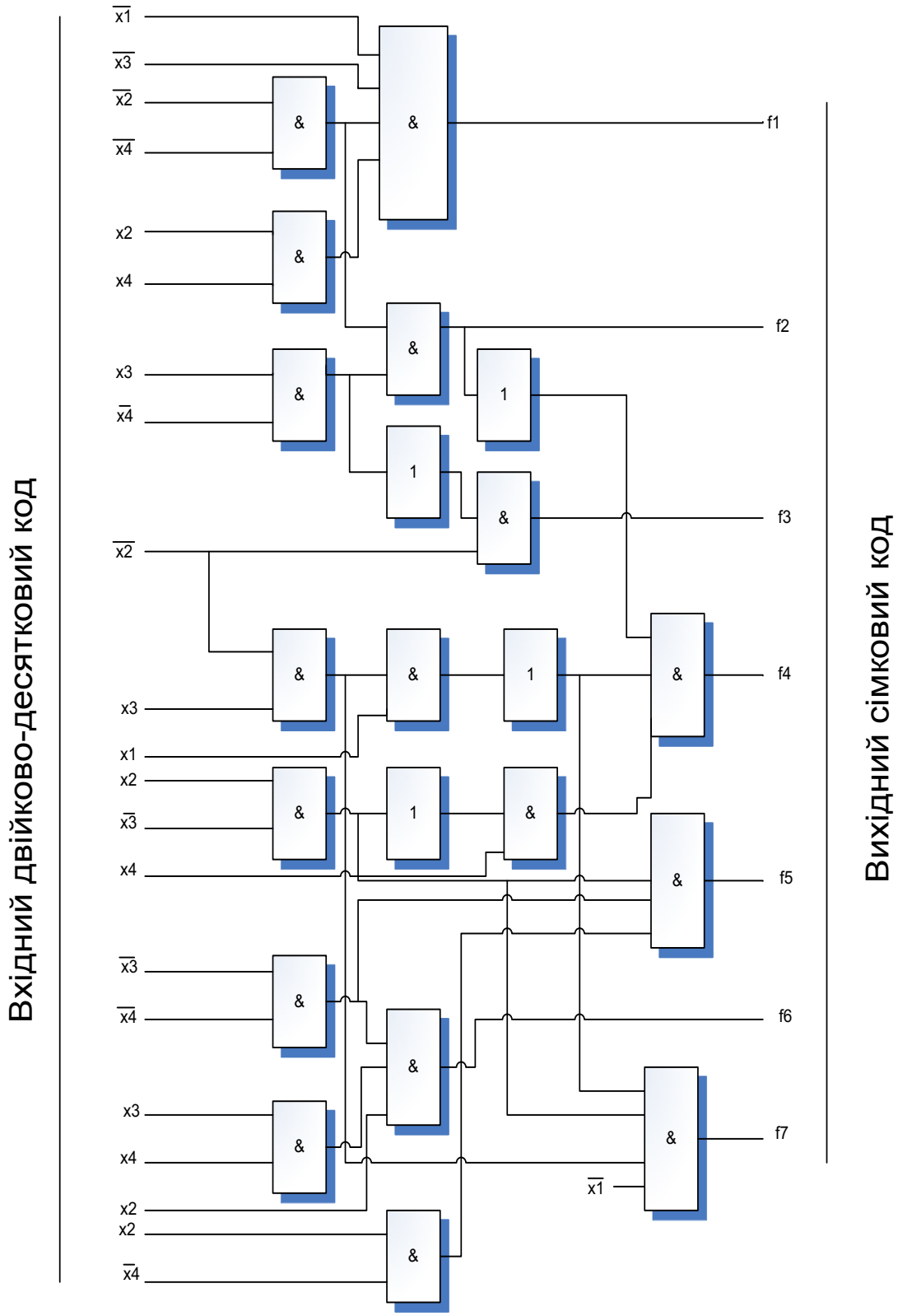


Рисунок 2.31 – Схема перетворювача двійково-десяткового коду в $N_{(7\text{-}10)}$ сімково-десятковий, згідно з табл. 2.11, за системою спрощених функцій

Контрольні запитання

1. *Опишіть поняття дешифратора.*
2. *Наведіть класифікацію дешифраторів.*
3. *Для чого використовуються дешифратори?*
4. *Як позначаються дешифратори на схемах?*
5. *Наведіть приклади використання дешифраторів.*
6. *Синтезуйте дешифратор на тригерах.*
7. *Для чого використовують каскадування дешифраторів?*
8. *Яким чином виконують каскадування дешифраторів?*
9. *Опишіть поняття шифратора.*
10. *Для чого використовують шифратори?*
11. *Як позначаються шифратори на схемах?*
12. *Синтезуйте шифратор на тригерах.*
13. *Опишіть поняття мультиплексора.*
14. *Для чого застосовується мультиплексор?*
15. *Для чого використовують каскадування мультиплексорів?*
16. *Яким чином виконують каскадування мультиплексорів?*
17. *Як позначаються мультиплексори на схемах?*
18. *Наведіть приклад використання мультиплексора.*
19. *В чому різниця між мультиплексором і дешифратором?*
20. *Опишіть поняття демультиплексора.*
21. *Для чого застосовується демультиплексор?*
22. *Для чого використовують каскадування демультиплексорів?*
23. *Яким чином виконують каскадування мультиплексорів?*
24. *Як на схемах позначаються демультиплексори?*
25. *Опишіть поняття компаратора.*
26. *Для чого застосовують компаратор?*
27. *Наведіть приклад використання компаратора.*
28. *Як позначається компаратор на схемах?*
29. *Опишіть поняття суматора.*
30. *Для чого застосовують суматор?*
31. *Наведіть приклад використання суматора.*
32. *Охарактеризуйте різницю між півсуматором і повним суматором.*

3 СИНТЕЗ ЦИФРОВИХ АВТОМАТІВ

Керувальний автомат (КА) можна розглядати як пристрій, який реалізовує конкретний алгоритм функціонування, визначений порядком виконання окремих операцій або процедур із керування деяким об'єктом.

Під час роботи КА згідно з алгоритмом функціонування, який він реалізовує, виробляє послідовність сигналів керування.

Керувальний автомат подають у вигляді дискретного набору з n входами $X = \{x_1, x_2, ..., x_n\}$ та k виходами $Y = \{y_1, y_2, ..., y_k\}$.

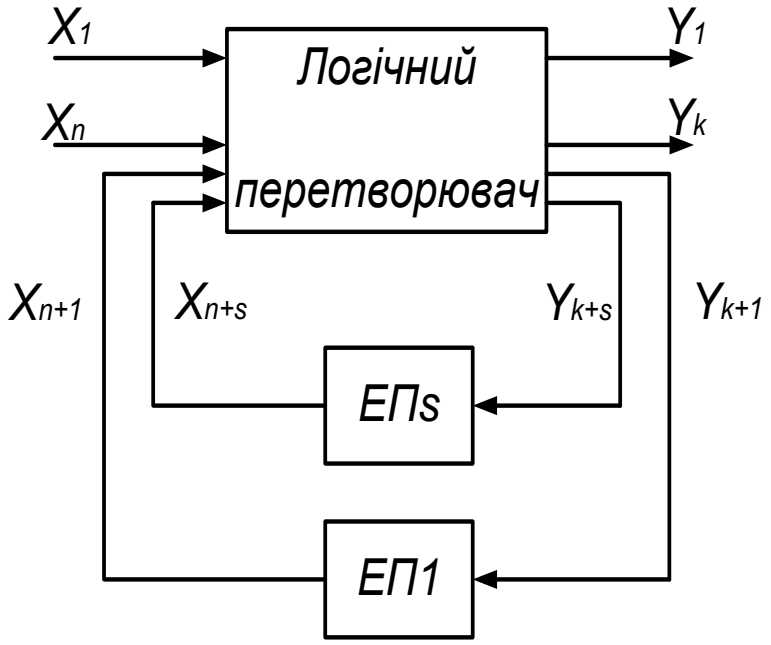


Рисунок 3.1 – Структурна схема керувального автомата

Частина приладу, в якій сконцентровані логічні елементи, утворює однотактну схему, яку називають логічним перетворювачем. На входи елементів пам'яті, які ще не виконують функцію затримки, надходять сигнали з додаткових внутрішніх виходів ЛП yk+1…yk+s. З виходів ЕП сигнали надходять на додаткові (внутрішні) входи ЛП xn+1…xn+s. Автомат функціонує в дискретні моменти часу t.

Протягом одного такту зберігається незмінним стан входу, стан виходу і внутрішній стан автомата. Час, протягом якого не відбувається зміни стану виходу автомата, позначається через T, і залежно від величини цього інтервалу часу розрізняють два класи автоматів: синхронні і асинхронні.

Синхронний автомат має генератор тактових сигналів (ТГ), і вхідні сигнали можуть діяти на автомат тільки за наявності сигналу від ТГ. Частоту ТГ вибирають такою, щоб до появи наступного сигналу від ТГ автомат встиг перейти до наступного стану (внутрішнього).

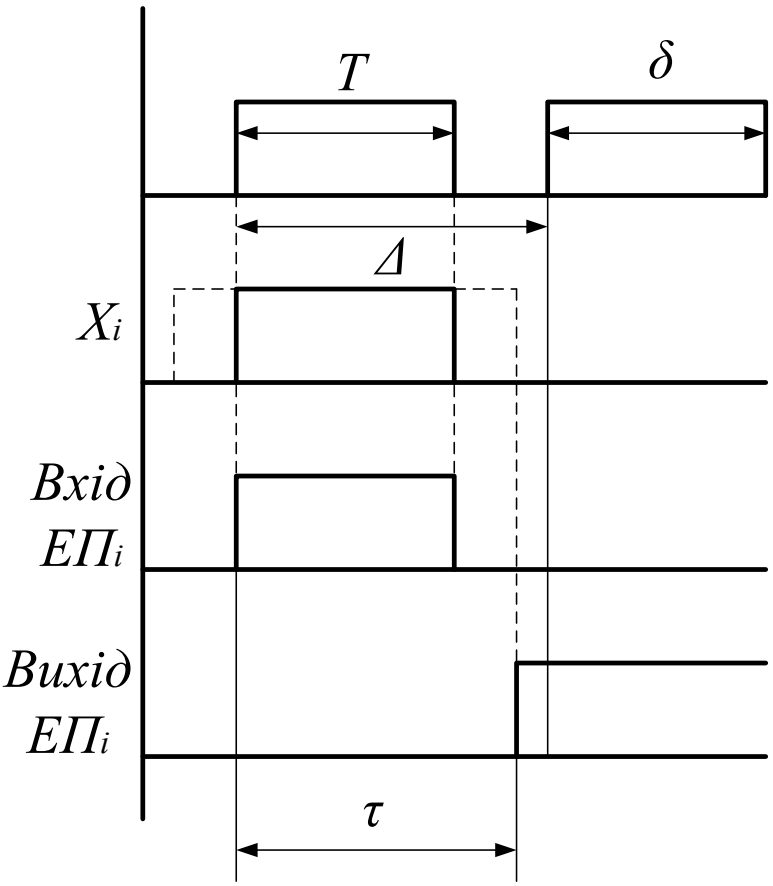


Рисунок 3.2 – Часова діаграма роботи автомата

В асинхронних автоматах довжина інтервалу часу Т, протягом якого залишається незмінним стан виходу, є величиною змінною і визначається тільки моментами зміни стану входу. Двом послідовним інтервалам часу завжди відповідають різні стани входу. Перехід до нового такту викликається не тільки зміною стану входу, але і зміною внутрішнього стану автомата.

Такт, який передує моменту зміни внутрішнього стану автомата, називається нестійким. Довжина нестійкого такту визначається часом τ переходу автомата із одного внутрішнього стану до іншого. Протягом інтервалу часу Т може бути кілька нестійких тактів (рис. 3.3).

Таким чином, для синхронного автомата характерним є:

1. Вхідні сигнали діють на автомат в строго фіксовані моменти часу і T=const;

2. Зміна внутрішнього стану автомата відбувається в інтервалі, коли немає дії внутрішніх сигналів.

Для асинхронного автомата характерним є:

1. Довжина інтервалу часу Т є величиною змінною і визначається зміною стану входу автомата;

2. Перехід до нового внутрішнього стану відбувається при незмінному стані входу.

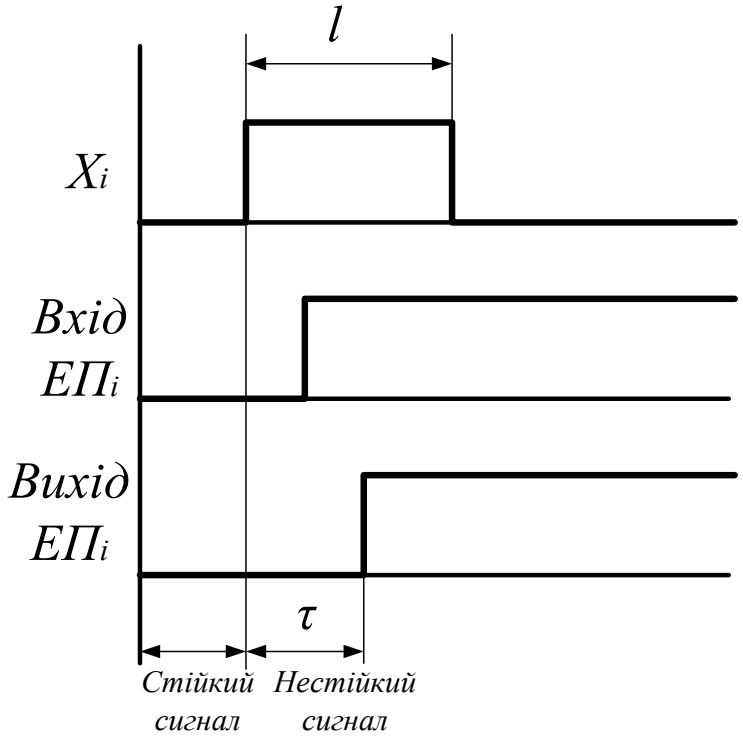


Рисунок 3.3 – Часова діаграма стійких та нестійких сигналів

3.1 Визначення закону функціонування автомата Мілі

Нехай мікропрограма задана у вигляді закодованого графа (рис. 3.4) і для її інтерпретації використовується автомат Мілі з законом функціонування:

$$\begin{cases} A(t+1) = \delta[A(t),\ X(t)], \\ Y(t) = \lambda[A(t),\ X(t)], \end{cases} \tag{3.1}$$

де A = {a_1, a_2, …, a_N} – множина станів автомата, t = 0, 1, 2, ... і A (0) = a_1 – початковий стан автомата. Мікропрограма оперує з чотирма логічними умовами X = {x_1, ..., x_4} і п'ятьма мікрооперaціями B = {y_1, ..., y_5} (рис. 3.4). Тому автомат, що реалізовує цю мікропрограму, повинен мати чотири входи x_1, ..., x_4 і п'ять виходів y_1, ..., y_5. Необхідний набір станів автомата визначається шляхом оцінювання графа мікропрограми, що здійснюється в такому порядку:

1) символом a_1 позначається вхід першої вершини, наступної за початковою, а також вхід кінцевої вершини;

2) входи вершин, які ідуть за операторними вершинами, позначаються символами a_2, a_3,...;

3) входи двох різних вершин, за винятком кінцевої, не можуть позначатись однаковими символами;

4) вхід вершини може позначатися тільки одним символом.

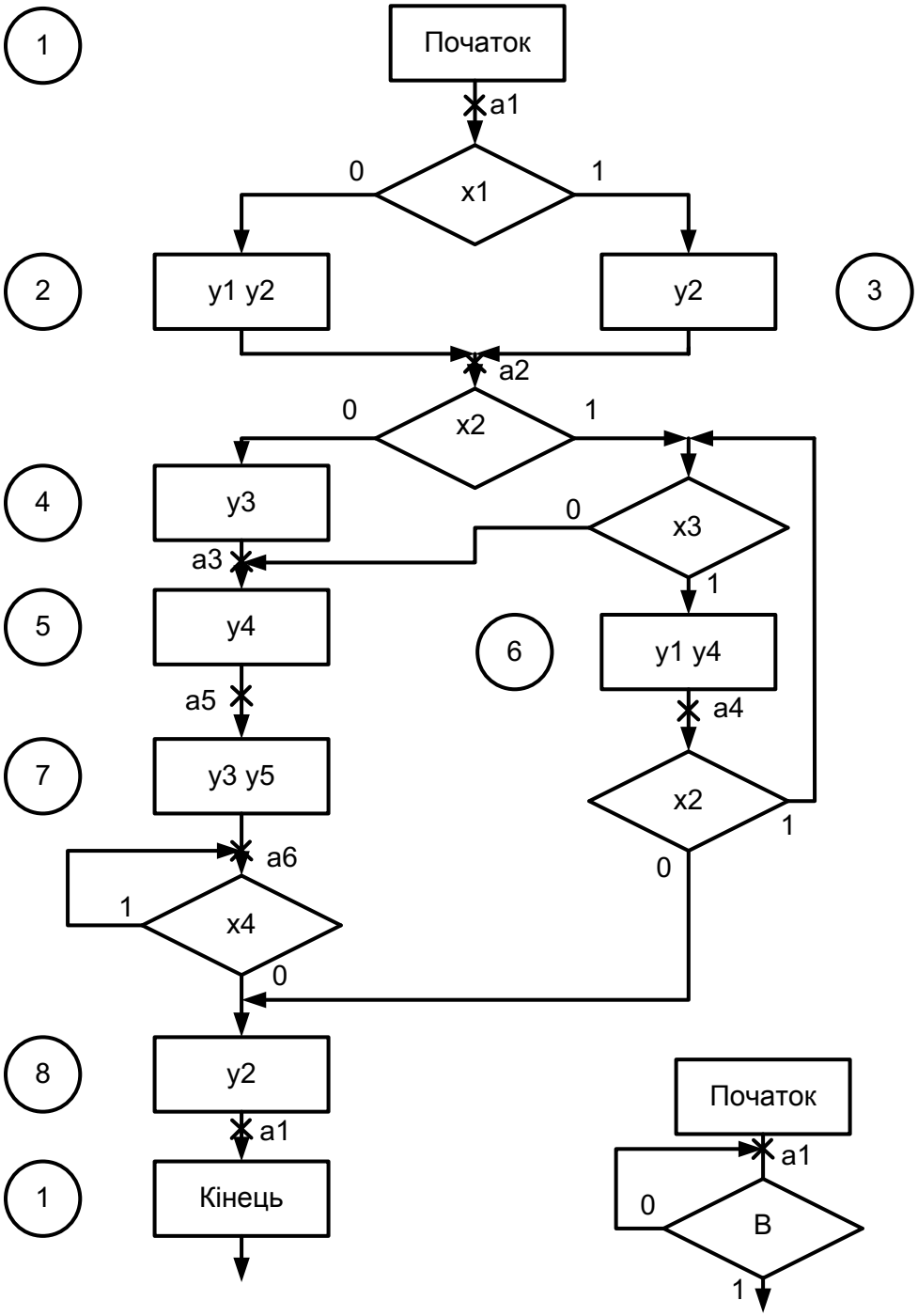


Рисунок 3.4 – Алгоритм мікропрограми

Приклад відзначення графа мікропрограми станами a_1, a_2, ..., a_6 автомата зображений на рис. 3.4. Допускається, що в початковому стані a_1 автомат формує нульові значення на усіх виходах $y_{1,...,}$ y_5. Після закінчення роботи автомат повертається в цей же стан a_1.

Наприклад, на графі з позначки a_1 в позначку a_2 можна перейти двома шляхами: $a_1\bar{x}_1y_1y_2a_2$ і $a_1x_1y_2a_2$, а з позначки a_2 в a_4 – по єдиному шляху $a_2x_2x_3y_1y_4a_4$. Шлях, що не проходить через умовні вершини, містить порожню множину логічних умов $a_iy_ay_b...y_\omega a_j$. Деякі шляхи, що ведуть до

позначки a_1 – кінцевої вершини графа, можуть містити порожню множину мікрооперацій:

$$a_i \tilde{x}_\alpha \tilde{x}_\beta .... \tilde{x}_\omega a_1. \qquad (3.2)$$

Очевидно, що вся множина шляхів визначає множину переходів між станами a_i і a_j автомата Мілі.

Якщо станам a_1, a_2, …, a_N поставити у відповідність вершини графа, а шляхам – дуги, направлені з вершини a_i в вершину a_j і відзначені наборами $\tilde{x}_\alpha \tilde{x}_\beta .... \tilde{x}_\omega / y_a y_b ... y\omega$ значень вхідних і вихідних сигналів, то отриманий граф буде визначати закон функціонування автомата Мілі. Так, граф мікропрограми (рис. 3.4) породжує автомат Мілі з законом функціонування відповідного графа (рис. 3.5).

Відмітимо, що дуги на графі автомата відзначаються тільки тими вхідними сигналами *x*, що визначають можливість переходу зі стану a_i у стан a_j, і тими вихідними сигналами *y*, які в даній ситуації набувають значення 1. При цьому передбачається, що всі інші вихідні сигнали мають нульове значення.

Коли автомат не працює (мікропрограма не виконується), він знаходиться в початковому стані a_1. При запуску (ініціюванні мікропрограми) автомат зберігає стан a_1 протягом одного такту, за час якого виконуються мікрооперації, що відповідають поточним значенням вхідних сигналів *x*. Після закінчення першого такту автомат перемикається в наступний стан a_j, описаний законом функціонування, і в ОА починає виконуватися наступний набір мікрооперацій. Момент закінчення мікропрограми відзначається поверненням автомата до початкового стану a_1.

Для запуску автомата використовується спеціальний сигнал *B*, що відноситься до групи вхідних сигналів і має довжину, рівну такту. Щоб виключити можливість появи вихідних сигналів *y* в моменти, коли автомат знаходиться в стані a_1 і не працює, дугам, що виходять з вершини a_1, додатково приписується сигнал запуску *B*, тільки при одиничному значенні якого вихідним сигналам *y* присвоюється значення 1 і стає можливим перехід автомата до наступного стану. При $B = 0$ автомат зберігає початковий стан і всі вихідні сигнали мають нульові значення. Введення сигналу B в закон функціонування автомата відповідає включенню вершини, що знаходиться в стані очікування, *B* в граф мікропрограми, за допомогою якої відзначається той факт, що виконання мікропрограми починається тільки при $B = 1$.

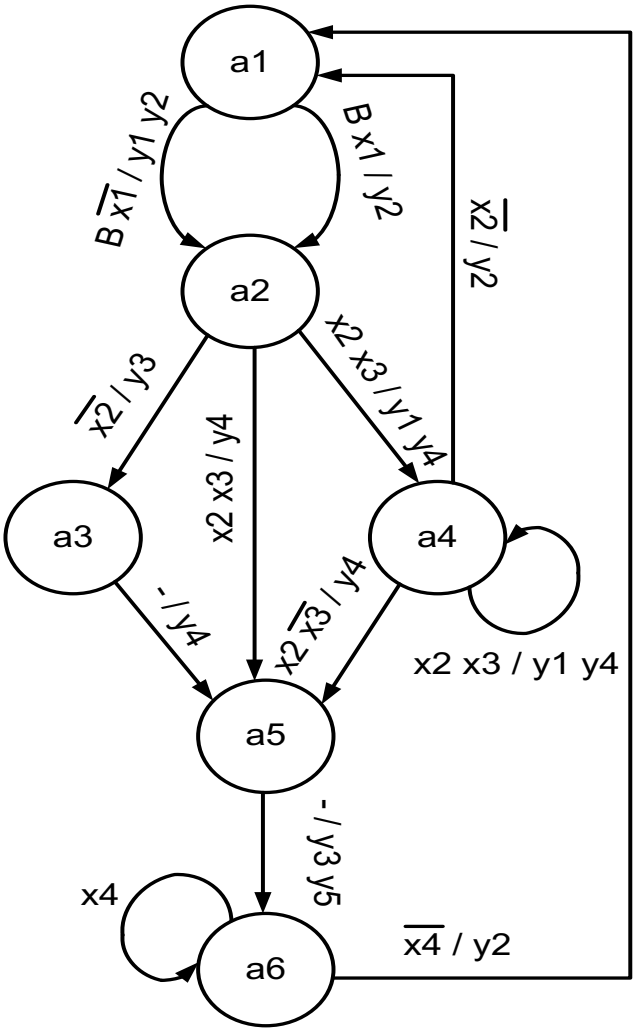


Рисунок 3.5 – Граф автомата Мілі

Автомат Мілі може інтерпретувати мікропрограму коректно тільки в тому випадку, якщо для всіх переходів виконується умова незалежності логічних умов $x_\alpha, x_\beta, ..., x\omega$ від результатів виконання мікрооперацій $y_a, y_b, ..., y_\omega$.

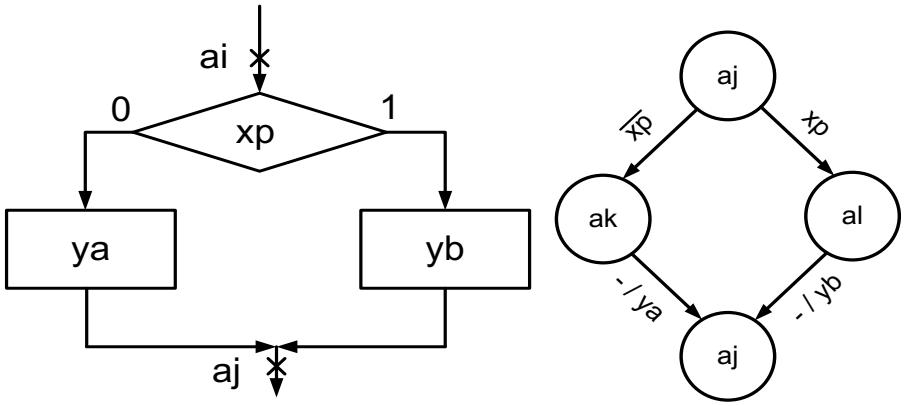
Умова незалежності порушується, якщо на деякому переході $a_i x_\alpha x_\beta .... x_\omega y_a y_b .... y_\omega a_j$ спостерігається функціональна залежність $x_\rho = f(y_k)$. В такому випадку виконання мікрооперації y_k може призвести до зміни значення умови x_ρ, і автомат, реагуючи на новий набір вхідних сигналів, сформує набір вихідних сигналів, що не відповідає мікропрограмі. Щоб уникнути цього негативного ефекту, необхідно проаналізувати всі переходи, що мають місце в автоматі, виявити вхідні сигнали $x_\rho, x_\sigma, ... x_\psi$, які залежать від мікрооперацій, що породжуються ними, і структурними засобами виключити вплив цих сигналів на процес виконання мікрооперацій. З цією метою можна використовувати такі способи:

1) запам'ятовування значень сигналів $x_\rho, x_\sigma, ... x_\psi$ на проміжку часу, що дорівнює тривалості такту;

2) введення в автомат додаткових станів;

3) реалізація автомата за схемою Мура.

Запам'ятовування сигналів $x_\rho, x_\sigma, ...x_\psi$, здійснюється на тригерах, що переключаються в стани $x_\rho, x_\sigma, ...x_\psi$, в кінці такту, одночасно з перемиканням автомата в новий стан. Порядок введення додаткових станів зображено на рис. 3.6, а.



а б

Рисунок 3.6 – Спосіб введення додаткових станів

Передбачається, що $x_\rho = (y_a, y_b)$ тобто умова незалежності для даного фрагмента мікропрограми порушується. Як видно з рис. 3.6, б, додаткові стани a_k і a_l виключають можливість перемикання сигналів y_a і y_b при зміні значення x_p. Введення додаткових станів призводить до додаткових витрат устаткування і збільшує час виконання мікропрограми. Реалізація останнього способу розглядається нижче.

Перелік переходів автомата Мілі

Керувальні автомати ЦВМ мають зазвичай кілька десятків станів, в результаті чого графи, що описують закон функціонування автоматів, втрачають свою наочність. До того ж, для цього класу автоматів характерні велика номенклатура кількості вхідних сигналів і малий ступінь зв'язку станів (для кожного стану переходи існують, в середньому, в 3-5 інших станах), в результаті чого таблиці переходів і виходів стають досить громіздкими, але майже незаповненими.

Закон функціонування керувального автомата можна описати у вигляді списку переходів автомата. Так закон функціонування автомата (рис. 3.5) подається в табл. 3.1, правила побудови якої очевидні. Списки переходів дозволяють компактно та наочно зобразити функціонування автоматів.

В деяких використаннях більш зручними стають обернені списки переходів, в яких перший стовпець таблиці подає наступний стан автомата і останній стовпець – попередній стан, з якого можливий перехід в заданий стан.

Таблиця 3.1 – Таблиця переходів автомата Мілі

Вихідний стан	Вхідний набір	Вихідний набір	Наступний стан
a_1	$B\bar{x}_1$	y_1y_2	a_2
	Bx_1	y_2	a_2
a_2	$\bar{x}_2$	y_3	a_3
	$x_2\bar{x}_3$	y_4	a_5
	x_2x_3	y_1y_4	a_4
a_3	-	y_4	a_5
a_4	x_2x_3	y_1y_4	a_4
	$x_2\bar{x}_3$	y_4	a_5
	$\bar{x}_2$	y_2	a_1
a_5	-	y_3y_5	a_6
a_6	x_4	-	a_6
	$\bar{x}_4$	y_2	a_1

Приклад

Синтез автомата Мілі на тригерах з окремими входами за таким алгоритмом функціонування

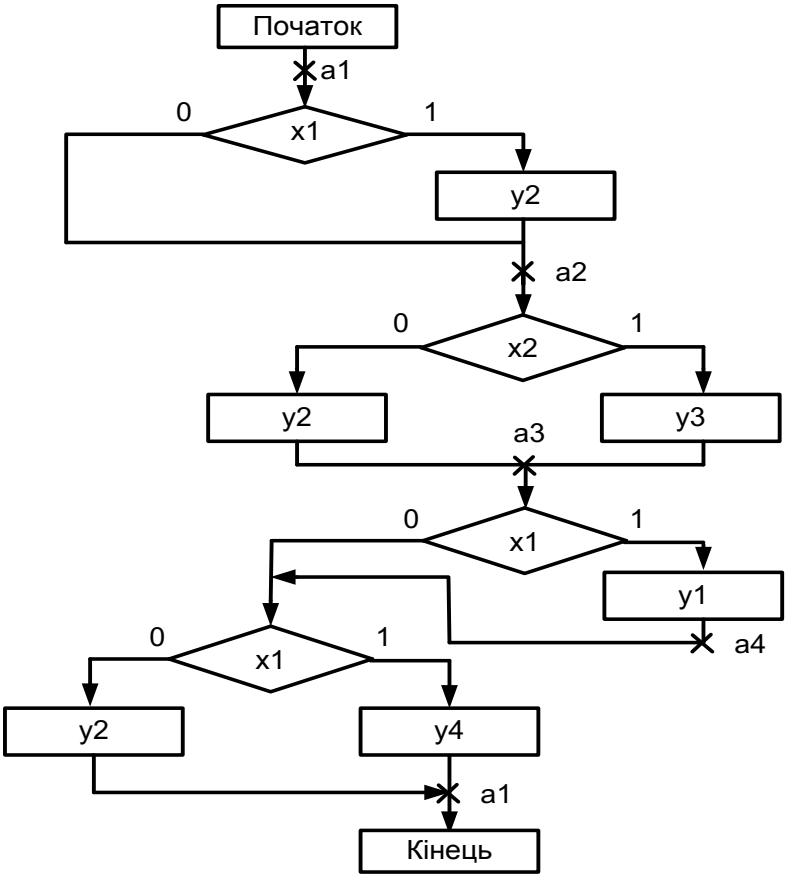


Рисунок 3.7 – Алгоритм функціонування автомата

Побудуємо граф автомата за алгоритмом функціонування (рис. 3.7). Він має 4 стани, для кодування яких необхідно n=log24=2 тригерів P_1 і P_2. Закодуємо стани автомата таким чином: a_1=00, a_2=01, a_3=10, a_4=11.

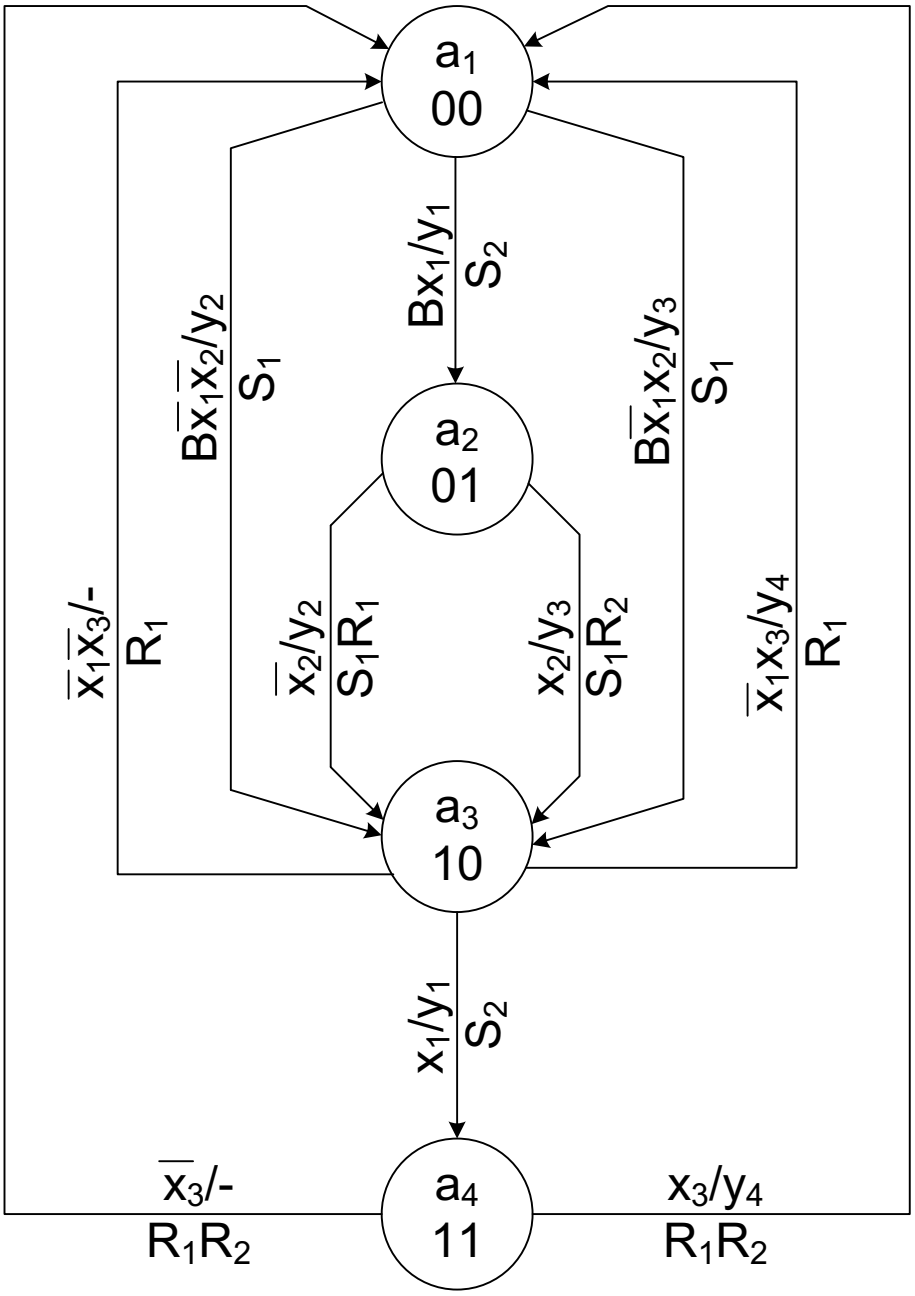


Рисунок 3.8 – Граф автомата Мілі, відмічений сигналами збудження

Для побудови функцій збудження і виходів використовується структурна таблиця автомата (табл. 3.2), що частково повторює таблицю переходів, але містить коди станів і перелік сигналів збудження, що формуються на переході (a_i, a_j). В стовпці сигналів збудження вказується сигнал *Sr*, якщо тригер *Pr* на переході (a_i, a_j) перемикається з стану 0 в стан 1, і сигнал Rr, якщо тригер перемикається з стану 1 в стан 0.

Таблиця 3.2 – Структурна таблиця автомату Мілі (рис. 3.8)

Переходи	Вихідний стан	Код вихідного	Наступний стан	Код наступного	Вхідний набір	Вихідний набір	Сигнали збудження
1	a_1	00	a_2	01	Bx_1	y_1	S_2
2			a_3	10	$B\bar{x}_1\bar{x}_2$	y_2	S_1
3			a_3	10	$B\bar{x}_1x_2$	y_3	S_1
4	a_2	01	a_3	10	$\bar{x}_2$	y_2	S_1R_2
5			a_3	10	x_2	y_3	S_1R_2
6	a_3	10	a_1	00	$\bar{x}_1\bar{x}_3$	_	R_1
7			a_1	00	$\bar{x}_1x_3$	y_4	R_1
8			a_4	11	x_1	y_1	S_2
9	a_4	11	a_1	00	$\bar{x}_3$	_	R_1R_2
10			a_1	00	x_3	y_4	R_1R_2

На основі табл. 3.2 будується канонічна система функцій виходів і збудження:

$$
\begin{gathered}
y_1 = a_1Bx_1 \vee a_3x_1;\\
y_2 = a_1B\bar{x}_1\bar{x}_2 \vee a_2\bar{x}_2;\\
y_3 = a_1B\bar{x}1x_2 \vee a_2x_2;\\
y_4 = a_3\bar{x}_1x_3 \vee a_4x_3;\\
S_1 = a_1B\bar{x}_1\bar{x}_2 \vee a_1B\bar{x}_1x_2 \vee a_2\bar{x}_2 \vee a_2x_2;\\
R_1 = a_3\bar{x}_1\bar{x}_3 \vee a_3\bar{x}_1x_3 \vee a_4\bar{x}_3 \vee a_4x_3;\\
S_2 = a_1Bx_1 \vee a_3x_1;\\
R_2 = a_2\bar{x}_2 \vee a_2x_2 \vee a_4\bar{x}_3 \vee a_4x_3.
\end{gathered}
\qquad (3.3)
$$

Мінімізуючи систему булевих функцій, одержуємо:

$$
\begin{gathered}
z_1 = a_1B\bar{x}_1 \vee a_2;\ z_2 = a_3\bar{x}_1 \vee a_4;\\
y_1 = (a_1B \vee a_3)x_1;\ y_2 = z_1\bar{x}_2;\ y_3 = z_1x_2;\ y_4 = z_2x_3;\\
S_1 = y_2 \vee y_3;\\
R_1 = z_2;\\
S_2 = y_1;\\
R_2 = a_2 \vee a_4,
\end{gathered}
\qquad (3.4)
$$

де z_1, z_2 – проміжні змінні.

Підставляючи в отримані вирази станів автомату стани p_1p_2 тригерів пам'яті:

$$a_1 = \overline{p}_1\overline{p}_2; a_2 = \overline{p}_1 p_2; a_3 = p_1\overline{p}_2; a_4 = p_1 p_2, \tag{3.5}$$

і мінімізуючи нові вирази, одержимо:

$$\begin{gathered}
z_1 = \overline{p}_1\overline{p}_2 B \vee \overline{p}_1 p_2 = \overline{p}_1\left(B\overline{x}_1 \vee p_2\right); \\
z_2 = p_1\overline{p}_2\overline{x}_1 \vee p_1 p_2 = p_1\left(\overline{x}_1 \vee p_2\right); \\
y_1 = \left(\overline{p}_1\overline{p}_2 B \vee p_1\overline{p}_2\right)x_1 = \overline{p}_2\left(B \vee p_1\right)x_1; \\
y_2 = z_1\overline{x}_2; \\
y_3 = z_1 x_2; \\
y_4 = z_2 x_3; \\
S_1 = y_2 \vee y_3; \\
R_1 = z_2; \\
S_2 = y_1; \\
R_2 = \overline{p}_1 p_2 \vee p_1 p_2 = p_2;
\end{gathered} \tag{3.6}$$

Використовуючи функцію як форму для побудови комбінаційної схеми, синтезується схема автомата (рис. 3.9).

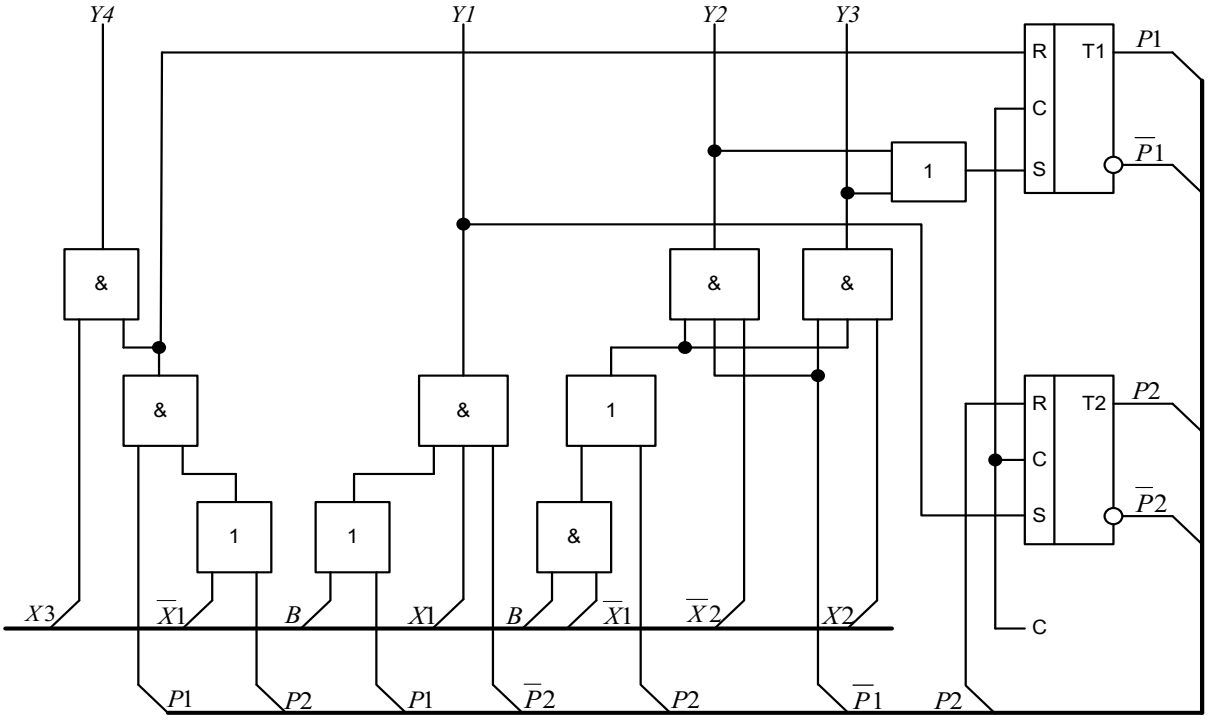


Рисунок 3.9 – Схема автомата Мілі за алгоритмом функціонування (рис. 3.7)

На даній схемі не відмічено коло початкового установлення автомата. Вважаючи, що ціна тригера з окремими входами дорівнює 8, синтезована схема має ціну С1=39. Число рівнів в комбінаційної частини автомата – 4, і

затрати часу на формування керувальних сигналів визначаються значенням $\tau_у=4\tau+\tau_г+3\tau=10\tau$.

Синтез автомата Мілі на тригерах з лічильним входом

Як елементи пам'яті автомата з законом функціонування (рис. 3.7) будемо використовувати тригери з лічильним входом. Сигнал збудження повинен подаватися на лічильний вхід тригера всякий раз, коли тригер повинен бути перемкнутий зі стану 0 в 1 чи зі стану 1 в 0. В такому випадку для забезпечення переходів, заданих табл. 3.2, повинні формуватися сигнали збудження, означені в табл. 3.3.

Таблиця 3.3 – Сигнали збудження тригерів з лічильним входом, що функціонують відповідно до табл. 3.2

Номер переходу	1	2	3	4	5	6	7	8	9	10
Сигнали збудження	T_2	T_1	T_1	T_1T_2	T_1T_2	T_1	T_1	T_2	T_1T_2	T_1T_2

Функції виходів автомата, побудованого на тригерах з лічильним входом, збігаються з функціями схеми рис. 3.8, а функції збудження визначаються з табл. 3.2 і табл. 3.3:

$$\begin{gathered} T_1 = a_1 B\bar{x}_1\bar{x}_2 \vee a_1 B\bar{x}_1 x_2 \vee a_2\bar{x}_2 \vee a_2 x_2 \vee a_3\bar{x}_1\bar{x}_3 \vee \\ \vee a_3\bar{x}_1 x_3 \vee a_4\bar{x}_3 \vee a_4 x_3; \\ T_2 = a_2\bar{x}_2 \vee a_2 x_2 \vee a_3 x_1 \vee a_4\bar{x}_3 \vee a_4 x_3. \end{gathered} \tag{3.7}$$

Мінімізуємо систему булевих функцій:

$$\begin{gathered} T_1 = a_1 B\bar{x}_1 \vee a_2 \vee a_3\bar{x}_1 \vee a_4; \\ T_2 = a_2 \vee a_3 x_1 \vee a_4. \end{gathered} \tag{3.8}$$

Підставляючи в дані вирази коди станів і мінімізуючи нові вирази, одержуємо:

$$\begin{gathered} T_1 = \bar{p}_1\bar{p}_2 B\bar{x}_1 \vee \bar{p}_1 p_2 \vee p_1\bar{p}_2\bar{x}_1 \vee p_1 p_2 = \bar{x}_1\left(B \vee p_1\right) \vee p_2; \\ T_2 = \bar{p}_1 p_2 \vee p_1\bar{p}_2 x_1 \vee p_1 p_2 = p_1 x_1 \vee p_2. \end{gathered} \tag{3.9}$$

Використовуючи вирази для функцій виходів і функції збудження, функцію КЧ автомата можна описати таким чином:

$$\begin{gathered} z_1 = \bar{p}_1\left(B\bar{x}_1 \vee p_2\right); \\ z_2 = B \vee p_1; \end{gathered}$$

$$
\begin{aligned}
y_1 &= \bar{p}_2 x_1;\\
y_2 &= z_1 \bar{x}_2;\\
y_3 &= z_1 x_2;\\
y_4 &= x_3 p_1 (\bar{x}_1 \vee p_2);\\
T_1 &= \bar{x}_1 z_2 \vee p_2;\\
T_2 &= p_1 x_1 \vee p_2.
\end{aligned}
\qquad (3.10)
$$

На основі системи булевих функцій побудована схема рис. 3.10. Якщо ціна тригера зі лічильним входом – 10, то дана схема має ціну С2=48.

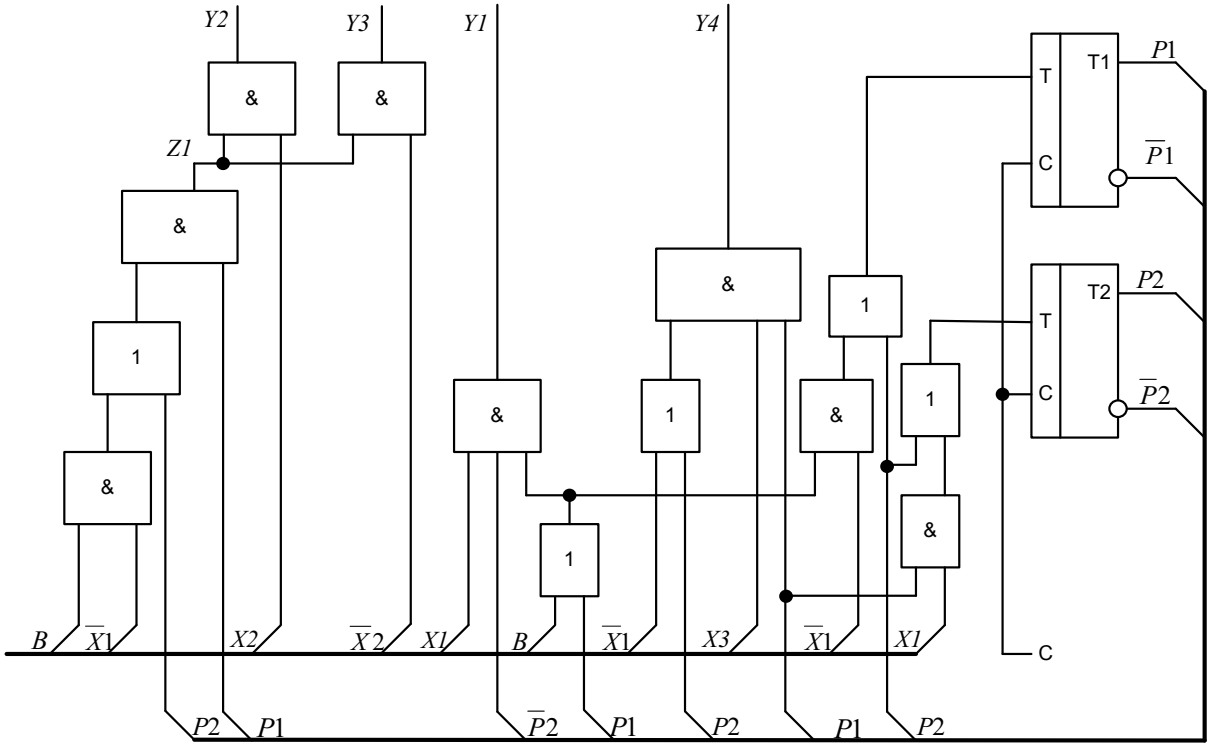


Рисунок 3.10 – Автомат Мілі на тригерах з лічильним входом

3.2 Визначення закону функціонування автомата Мура

Будь-яку мікропрограму можна інтерпретувати як автомат Мура, якому властивий такий закон функціонування:

$$
\begin{cases}
A(t+1) = \delta[A(t), X(t)];\\
Y(t) = \lambda[A(t)],
\end{cases}
\qquad (3.11)
$$

де t = 0, 1, …. і A(0) = a_1 – початковий стан автомата.

Оскільки в автоматі Мура вихідні сигнали пов'язані тільки зі станами автомата, то кожній операторній вершині графа мікропрограми варто поставити у відповідність один зі станів a_2, a_3 ... . Виходячи з цього, можна сформулювати таке правило оцінювання станів автомата на графі мікропрограми:

1) символом a_1 відзначаються початкова і кінцева вершини мікропрограми;

2) кожна операторна вершина відзначається єдиним символом a_2, a_3, ...;

3) дві різні операторні вершини не можуть бути відмічені однаковими символами.

На графі мікропрограми (рис. 3.4) символи станів a_1, a_2, ..., a_8 позначені цифрами 1, 2, ..., 8 в кружечках біля відповідних вершин.

$$a_i \tilde{x}_\alpha \tilde{x}_\beta ... \tilde{x}_\omega a_j. \qquad (3.12)$$

Граф мікропрограми визначає безліч шляхів типу

де a_i і a_j – позначки початку і кінця шляху; $\tilde{x} = x$, якщо шлях проходить через вихід умовної вершини, відзначений значенням 1, і $\tilde{x} = \bar{x}$, якщо шлях проходить через вихід, відмічений значенням 0. Так, на графі існують такі шляхи між оцінками 1, 2, ..., 8: $a_1\bar{x}_1a_2; a_1x_1a_3; a_2\bar{x}_2a_4; a_3\bar{x}_2a_4; a_2x_2x_3a_6$.... Оскільки символи a_1, a_2 ... розглядаються як стани автомата, то вся множина шляхів між позначками на графі мікропрограми визначає множину переходів серед станів автомата. Позначивши стани автомата a_1, a_2, ... a_N вершинами, приписавши цим вершинам пов’язані з ними набори вихідних сигналів y_a, y_b, ... y_w і з’єднавши вершини дугами, можна побудувати граф автомата Мура, інтерпретувавши задану мікропрограму.

Початковим станом автомата є стан a_1. Ініціювання мікропрограми зводиться до запуску автомата поданням сигналу В на виділений для цієї мети вхід автомата. Оскільки перехід автомата з початкового стану а в наступний стан можливий тільки за наявності сигналу В=1, всі дуги графа автомата, що виходять з вершини a_1, можуть бути відмічені сигналом В. На відміну від автомата Мілі, стани в автоматі Мура повинні перемикатися на початку такту. Кінець виконання мікропрограми відмічається переходом автомата в початковий стан a_1. Оскільки до моменту переходу сигнал запуску В приймає значення 0, автомат переходить в стан a_1 до надходження наступного сигналу В=1.

В загальному випадку автомат Мура має більшу кількість станів, ніж автомат Мілі, що реалізовує ту ж саму мікропрограму, оскільки для реалізації мікропрограми автоматом Мілі потрібно 6 станів, а автоматом Мура – 8. Різниця в кількості станів впливає на кількість обладнання в автоматі. Зазвичай використання автомата Мілі приводить до економії обладнання порівняно з автоматом Мура. З врахуванням цього для керувального автомата кращою є схема Мілі.

Винятком є випадок реалізації програми з великою кількістю переходів, на яких не виконується умова незалежності вхідних і вихідних сигналів.

Приклад

Синтез автомата Мура на D-тригерах

Розглянемо приклад синтезу автомата Мура з законом функціонування (рис. 3.11).

Відповідно до цього алгоритму, побудуємо граф автомата з 8 вершинами, як показано на рис. 3.11.

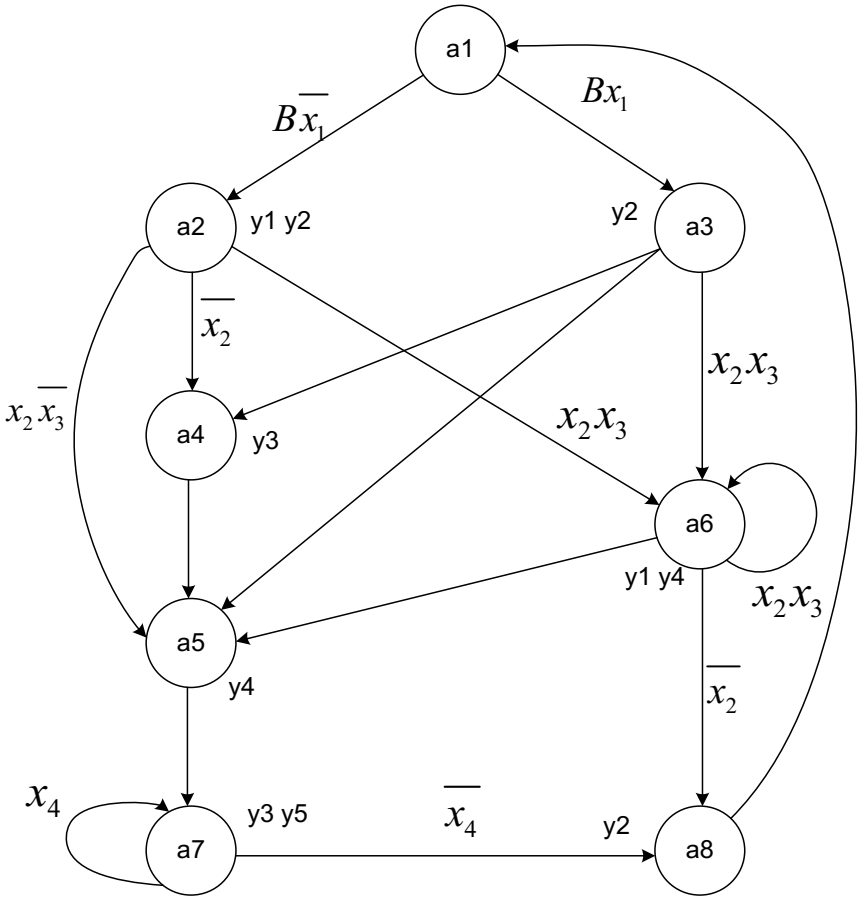


Рисунок 3.11 – Граф автомата Мура

Автомат має вісім станів, які можна закодувати такими трирозрядними кодами:

a_1=000; a_3=010; a_5=100; a_7=110;
a_2=001; a_4=011; a_6=101; a_8=111. (3.13)

Мінімізація функцій дешифрування коду p_1 p_2 p_3 виконана з використанням карти Карно, з якої витікає:

$$a_1 = \overline{p}_1\overline{p}_2\overline{p}_3; \quad a_3 = \overline{p}_1 p_2\overline{p}_3; \quad a_5 = p_1\overline{p}_2\overline{p}_3; \quad a_7 = p_1 p_2\overline{p}_3;$$
$$a_2 = \overline{p}_1\overline{p}_2 p_3; \quad a_4 = \overline{p}_1 p_2 p_3; \quad a_6 = p_1\overline{p}_2 p_3; \quad a_8 = \overline{p}_1\overline{p}_2\overline{p}_3. \quad (3.14)$$

Як елементи пам'яті будемо використовувати D-тригери, які перемикаються в стан 1, якщо сигнал збудження D = 1, і в стан 0, якщо D = 0. Отже, сигнал збудження $D_i = 1$ повинен формуватись кожен раз, коли при переході в наступний стан тригер P_i повинен знаходитись в стані 1. Структурна таблиця автомата Мура, яка відповідає графу, стани якого кодуються наборами (3.14), подана в табл. 3.4.

Таблиця 3.4 – Структурна таблиця автомата Мура

Перехо-ди	Вихід-ний стан	Код вихідного стану	Наступний стан	Код наступного стану	Вхідний набір	Вихідний набір	Сигнали збудже-ння
1 2	a_1	000	a_2 a_3	001 010	Bx_1 $B\overline{x}_1$	y_2 y_1y_2	D_1 D_2
3 4 5	a_2	001	a_4 a_5 a_6	011 100 101	$\overline{x}_2$ $x_2\overline{x}_3$ x_2x_3	y_3 y_4 y_1 y_4	D_2D_1 D_3 D_3D_1
6 7 8	a_3	010	a_4 a_5 a_6	011 100 101	$\overline{x}_2$ $x_2\overline{x}_3$ x_2x_3	y_3 y_4 y_1 y_4	D_2D_1 D_3 D_3D_1
9	a_4	011	a_5	100	-	y_4	D_3
10	a_5	100	a_7	110	$\overline{x}_4$	y_3 y_5	D_3D_1
11 12	a_6	101	a_6 a_8	101 111	x_2x_3 $\overline{x_2}$	y_1 y_4 y_2	D_3D_1 D_3D_2 D_1
13 14	a_7	110	a_7 a_8	110 111	x_4 $\overline{x_4}$	y_3 y_5 y_2	D_3D_1 D_3D_2 D_1
15	a_8	111	a_1	000	-	-	-

Канонічна система булевих функцій, що відповідає табл. 3.4, має такий вигляд:

$$\begin{cases} y_1 = a_1 B\overline{x}_1 \vee a_2 x_2 x_3 \vee a_3 x_2 x_3 \vee a_6 x_2 x_3 = a_1 B\overline{x}_1 \vee \left(a_2 \vee a_3 \vee a_6\right) x_2 x_3; \\ y_2 = a_1 B\overline{x}_1 \vee a_1 B x_1 \vee a_6 \overline{x}_2 \vee a_7 \overline{x}_4 = a_1 B \vee a_6 \overline{x}_2 \vee a_7 \overline{x}_4; \\ y_3 = a_2 \overline{x}_2 \vee a_3 \overline{x}_2 \vee a_5 \overline{x}_4 \vee a_7 x_4 = \left(a_2 \vee a_3\right) \overline{x}_2 \vee a_5 \overline{x}_4 \vee a_7 x_4; \\ y_4 = a_2 x_2 x_3 \vee a_2 x_2 \overline{x}_3 \vee a_3 x_2 x_3 \vee a_3 x_2 \overline{x}_3 \vee a_4 \vee a_6 x_2 x_3 = x_2 \left(a_2 \vee a_3 \vee a_6 x_3\right) \vee a_4; \\ y_5 = a_5 \overline{x}_4 \vee a_7 x_4; \\ D_1 = a_1 B x_1 \vee \left(a_2 \vee a_3\right)\left(\overline{x}_2 \vee x_2 x_3\right) \vee \left(a_5 \vee a_7\right) \overline{x}_4 \vee a_6 \left(x_2 x_3 \vee \overline{x}_2\right); \\ D_2 = a_1 B\overline{x}_1 \vee \left(a_2 \vee a_3\right) \overline{x}_2 \vee a_6 \overline{x}_2 \vee a_7 \overline{x}_4; \\ D_3 = \left(a_2 \vee a_3\right) x_2 \vee a_4 \vee a_5 \overline{x}_4 \vee a_6 \left(x_2 x_3 \vee \overline{x}_2\right) \vee a_7. \end{cases} \quad (3.15)$$

Відповідно до цих функцій будується схема автомата Мура аналогічно з алгоритмом побудови схеми автомата Мілі, вищенаведеної.

Контрольні запитання

1. Опишіть поняття керувального автомата.
2. Що являє собою логічний перетворювач?
3. Що являє собою закон функціонування автомата.
4. В чому різниця між синхронним і асинхронним автоматом?
5. Опишіть порядок оцінювання графа мікропрограми.
6. Поясніть поняття списку переходів автомата.
7. В чому відмінність автомата Мура від автомата Мілі?
8. В якій частині автоматів зберігається поточний стан?
9. Який з автоматів, Мілі чи Мура, має більшу кількість станів для однієї мікропрограми?
10. Опишіть порядок синтезу автомата Мілі.
11. Опишіть порядок синтезу автомата Мура.
12. Синтезуйте та промоделюйте автомат Мілі за законом функціонування, який наведений вище.
13. Синтезуйте та промоделюйте автомат Мура за законом функціонування, який наведений вище.

4 НАПІВПРОВІДНИКОВІ ЗАПАМ ЯТОВУВАЛЬНІ ПРИСТРОЇ

4.1 Загальна характеристика пам яті

Функція пам яті

Пам'яттю комп'ютера називається сукупність різних пристроїв, призначених для прийому, зберігання та видачі двійкової інформації. Окремий пристрій називається запам'ятовувальним (ЗП) або просто пам'яттю. Термін «запам'ятовувальний пристрій» вживають тоді, коли потрібно підкреслити принцип його будови: на магнітних осердях, напівпровідниках і т. д. Термін «пам'ять» застосовують, коли вказують на виконувану функцію: основна, постійна і т. п.

Пам'ять сучасних комп'ютерів класифікують за функціональним призначенням, типом носія інформації, способом організації доступу до інформації. За функціональним призначенням пам'ять комп'ютерів поділяється на дві основні групи: зовнішню і внутрішню.

Зовнішні ЗП призначені для тривалого зберігання великих масивів інформації з ємністю до гігабайта і більше та малою швидкодією. Зовнішня пам'ять містить в собі накопичувачі на магнітних стрічках, дисках, барабанах та оптичних дисках.

Внутрішні ЗП призначені для зберігання програм і даних, що обробляються в поточний момент часу. До внутрішньої пам'яті відносяться:

- *надоперативні* (регістрові) ЗП, які використовують регістри загального призначення процесора; вони мають невелику інформаційну ємність і швидкодію роботи процесора;
- *кеш-пам'ять*, яка призначена для зберігання копій інформації, використовуваної в поточних операціях обміну. Висока швидкодія кеш-пам'яті підвищує продуктивність комп'ютера;
- *оперативні*, які характеризуються високою швидкодією і інформаційною ємністю до сотень мегабайтів; оперативна пам'ять комп'ютерів перших поколінь будувалася на магнітних осердях. В наш час ОП реалізовується на напівпровідникових ВІС ЗП. У процесі роботи інформація із зовнішньої пам'яті за необхідності переписується в оперативний ЗП (ОЗП);
- *постійні*, які будуються на напівпровідникових ВІС. У постійну пам'ять інформація записується заздалегідь і її можна тільки зчитувати. Оперативні і постійні ЗП утворюють основну пам'ять комп'ютера;
- *спеціалізовані* види пам'яті – багатопортові, асоціативні, відеопам'ять та ін.

За фізичним принципом будови пам'ять комп'ютера буває:

- *магнітна* (на осердях і плівках, на циліндричних і плоских магнітних доменах);

- *ультразвукова* (магнітострикційна, електрострикційна);

- *сегнетоелектрична та голографічна* (лазерна), на основі надпровідності;

- *напівпровідникова* на ВІС і НВІС, ультра-ВІС.

Напівпровідникові ВІС ЗП, в свою чергу, характеризуються:

- технологією виготовлення: на біполярних транзисторах (ТТЛШ, ЕСЛ, $И^2Л$), на МОН-структурах (р-МОН, л-МОН, КМОН); серед новітніх розробок слід відзначити ЗП, в яких використовуються ПТШ на основі арсеніду галію;

- способом зберігання інформації – статичні і динамічні (в статичних ЗП елементом пам'яті є тригер, а в динамічних елемент пам'яті будують на конденсаторі і МОН-транзисторах);

- енергозалежністю: розрізняють енергозалежні ВІС ЗП, в яких при відключенні джерела живлення збережена інформація руйнується (що справедливо в даний час для більшості напівпровідникових мікросхем пам'яті), і енергонезалежні (зазвичай на сегнетоелектриках), в яких інформація зберігається;

- структурною організацією ВІС ЗП, символічно подається у вигляді $N \times m$, де N – кількість збережених адресованих одиниць інформації; m – розрядність (організацію у вигляді $N \times 1$ називають однорозрядною, а $N \times m$ – словниковою, при цьому $m > 1$).

Елементний базис пам'яті сучасних комп'ютерів становлять мікросхеми різного ступеня інтеграції. Основою будь-якого ЗП є елемент пам'яті (ЕП) статичного чи динамічного типу, призначений для записування, зберігання та зчитування одного біта інформації – цифри 0 або 1. Сукупність ЕП, які утворюють n-розрядне слово, називають коміркою пам'яті (КП). Сукупність КП утворює запам'ятовувальний масив, названий матрицею М елементів пам'яті.

Основні параметри пам'яті

Основними операціями в пам'яті є записування і зчитування певної одиниці інформації, наприклад, байта. Ці операції називаються також звертанням до пам'яті. Пам'ять характеризується інформаційною ємністю, фізичним обсягом, питомою ємністю і вартістю, шириною вибірки, споживаною потужністю і швидкодією.

Інформаційна ємність Е являє собою максимальний обсяг даних, який може одночасно зберігатися в пам'яті. Ємність виражається в бітах, байтах (8 біт = 1 байт), кілобайтах (2^{10} байт = 1 Кбайт), мегабайтах (2^{10} Кбайт = = 1 Мбайт) і гігабайтах (2^{10} Мбайт = 1 Гбайт) (при цьому слід враховувати, що $2^{10} = 1024$).

Питома ємність визначається відношенням інформаційної ємності ЗП до його фізичного обсягу. Питома вартість – це відношення вартості ЗП до його інформаційної ємності. Ширина вибірки подається числом розрядів, які записуються в ЗП або зчитуються з нього за одне звертання.

Споживану потужність задають або для всього ЗП, або на зберігання одного біта інформації. Основними вимогами до пам'яті є максимально велика інформаційна ємність, висока швидкодія (малий час звертання $t_{зв} < 10$ нс), мінімальна споживана потужність (менше 1 мкВт на 1 біт інформації, що зберігається).

В наш час ні один вид ЗП не задовольняє ці вимоги повною мірою. Тому в пам'яті використовуються різні види ЗП, які відрізняються принципами будови і своїми характеристиками.

Швидкодія ЗП вимірюється часом записування і зчитування, а також тривалістю відповідних їм циклів. *Час записування* t_{WR} – це інтервал між моментами появи керувального сигналу записування і установленням КП в стан, що задається вхідними сигналами. *Час зчитування* – це інтервал між моментами появи керувального сигналу зчитування t_{RD} і даних на виході пам'яті. Мінімальний допустимий інтервал між послідовними зчитуваннями t_{CYR} і записами t_{CYW} утворює відповідний цикл. Тривалість циклів може перевищувати час зчитування або записування, оскільки після цих операцій необхідна додаткова затримка для установлення вихідного стану пам'яті. Як тривалість циклу звертання до пам'яті беруть величину $t_{CY} = \max(t_{CYR}, t_{CYW})$.

Вхідні і вихідні сигнали мікросхеми пам'яті

Мікросхеми ОП мають типові виводи, на яких діють певні адресні, інформаційні та керувальні сигнали (рис. 4.1, а).

Призначення виводів і сигналів:

A (*Address*) – входи адреси, розрядність якої *k* визначається співвідношенням $k = \log_2 N$, де $N = 2^k$ – максимально можливе число даних (бітів, байтів, слів), які зберігаються в пам'яті і адресуються як єдине ціле;

DI (*Data Input*) – шина вхідних даних;

DO (*Data Output*) – шина вихідних даних;

$\overline{W}/R$ (*Write / Read*) – сигнал запису даних при $\overline{W}/R = 0$ або зчитування при $\overline{W}/R = 1$;

$\overline{CS}$ (*Chip Select*) або $\overline{CE}$ (*Chip Enable*) – сигнал дозволу при $\overline{CS(CE)} = 0$ або заборони, якщо $\overline{CS(CE)} = 1$, роботи даної мікросхеми.

Особливістю роботи динамічних ЗП є мультиплексування ША (рис. 4.1, б). Адреса, наприклад, $A = A_{15}, A_{14}, \ldots, A_0$ ділиться на старшу півадресу $A_x = A_{15}, A_{14}, \ldots, A_8$ і молодшу $A_y = A_7, A_6, \ldots, A_0$. Півадреси подаються на одні й ті ж входи адреси мікросхеми пам'яті. Подача півадреси A_x супроводжується сигналом *Row Address Strobe*, а півадреси A_y – сигналом *Column Address Strobe*. Такий спосіб адресації зменшує кількість виводів корпусу ІМС. Часто виводи *DI* і *DO* об'єднуються в спільний вивід *DIO*.

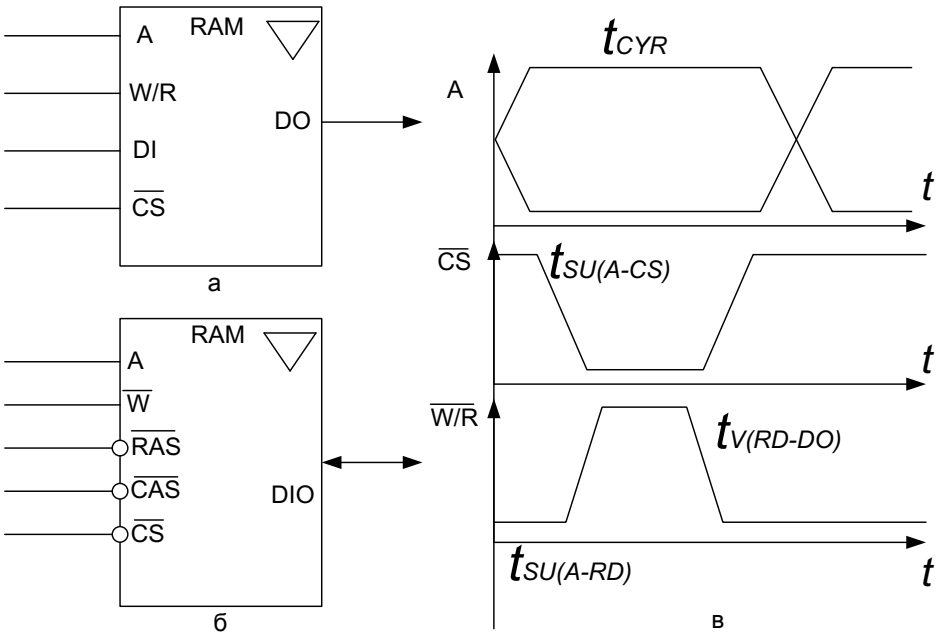


Рисунок 4.1 – Мікросхеми ОП: а), б) умовні графічні позначення; в) часові діаграми сигналів

Часові характеристики мікросхем пам'яті

Вимоги до взаємного тимчасового положення двох сигналів (*A-B*) задають такими параметрами:

- часом попереднього установлення $t_{SU\ (A\text{-}B)}$ сигналу *A* відносно сигналу *B*, тобто інтервалом між початковими моментами обох сигналів;

- часом утримання $t_{H\ (A\text{-}B)}$ – інтервалом часу між початком сигналу А і закінченням сигналу *B*;

- часом зберігання $t_{V\ (A\text{-}B)}$ – інтервалом між закінченням сигналів *A* і *B*.

Тривалість сигналів позначається як t_W (*Width* – ширина).

Для ЗП характерною є така послідовність сигналів у часі (рис. 4.1, в): спочатку адреса, потім вибір мікросхеми $\overline{CS}$, потім строб записування-зчитування $\overline{W}/R$. Індексом *A* (*Access*) позначають інтервали часу від появи керувального сигналу до появи даних на виході (рис. 4.1, в).

Способи доступу до даних у напівпровідниковій пам'яті

У напівпровідникових ЗП виділяють адресні, послідовні і асоціативні способи доступу до даних (рис. 4.2).

При адресному доступі адресний код вказує номер комірки пам'яті, з якою повинен проводитися обмін. Усі комірки у момент звертання рівнодоступні. До адресних ЗП відносять:

- *RAM* (*Random Access Memory*), інакше ОЗП;

- *ROM* (*Read Only Memory*), або ПЗП.

Оперативні ЗП зберігають дані, необхідні при виконанні поточної програми; вони можуть бути змінені в будь-який момент часу. Оперативні

ЗП здебільшого енергозалежні. У постійних ЗП вміст комірок або взагалі не змінюється, або змінюється рідко в спеціальних режимах.

Запам'ятовувальні пристрої RAM поділяються на статичні *SRAM* (*Static RAM*) і динамічні *DRAM* (*Dynamic RAM*). В статичних *RAM* елементами пам'яті є тригери. Вони зберігають свій стан, поки схема має напругу живлення і нові дані не записуються. У динамічних *RAM* дані зберігаються у вигляді зарядів конденсаторів, створюваних компонентами МОН-транзисторів. Саморозряд конденсаторів призводить до руйнування даних, тому вони періодично (кожні 2–30 мс) повинні регенеруватися. Однак щільність упакування динамічних ЕП перевищує в 4–5 разів цей же показник для статичних *RAM*. Регенерація даних здійснюється за допомогою спеціальних контролерів. Розроблено також *DRAM* з внутрішніми схемами регенерації; такі ЗП називаються квазістатичними.

Статичні ОЗП поділяють на такі типи:

- *асинхронні* – керувальні сигнали можна задавати як імпульсами, так і рівнями;

- *тактовані* – в них деякі сигнали повинні бути обов'язково імпульсами, наприклад, сигнал дозволу роботи $\overline{CS}$;

- *синхронні*, в яких організовано конвеєрний канал передачі даних, синхронізований від тактової системи процесора.

Динамічні ЗП характеризуються найбільшою інформаційною ємністю і невисокою вартістю, тому вони використовуються як основна пам'ять комп'ютерів. Статичні ЗП в 4–5 разів дорожчі за динамічні і приблизно в стільки ж разів менша їх інформаційна ємність. Їх перевагою є висока швидкодія, а типова область застосування – схеми кеш-пам'яті.

Постійна пам'ять типу *ROM(M)* програмується при виготовленні за допомогою масок, тому її ПЗП називають масковим. У таких різновидах *ROM* в позначеннях є буква *P* (від *Programmable*). Це – пам'ять, яка одноразово програмується користувачем – *PROM* (ППЗП – програмовані ПЗП) і багаторазово програмується – *EPROM*, *EEPROM*.

Пам'ять типу Flash за ЕП подібна *EEPROM* (інакше E^2PROM), проте має структурні та технологічні особливості, які дозволяють виділити її в окремий тип.

У ЗП з послідовним доступом записувані дані утворюють чергу. Зчитування виконується слово за словом в порядку записів або навпаки. Прямий порядок зчитування використовується в буферах FIFO за принципом «першим прийшов – першим вийшов (First In – First Out)», а також у файлових і циклічних ЗП.

Різниця між пам'яттю *FIFO* і файловим ЗП полягає в тому, що в *FIFO* запис в порожній буфер відразу доступний для читання (тобто, надходить в кінець ланцюжка моделі ЗП). У файлових ЗП дані надходять в початок ланцюжка і з'являються на виході після деякого числа звертань, рівного числу елементів у ланцюжку.

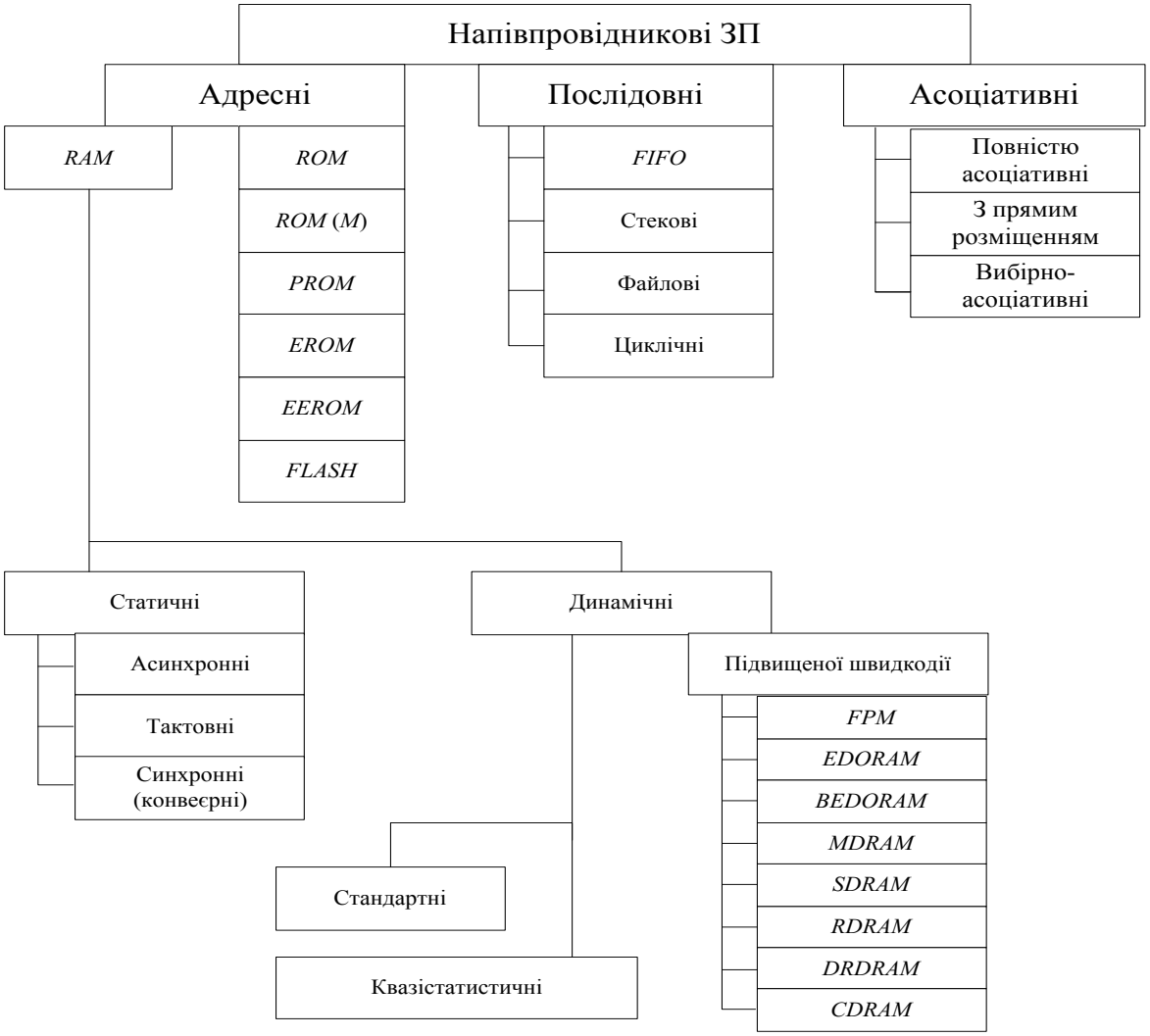


Рисунок 4.2 – Класифікація напівпровідникових ЗП

У циклічних ЗП слова доступні одне за одним з постійним періодом, визначеним ємністю пам'яті. До них відноситься відеопам'ять (*VRAM*). Зчитування в зворотному порядку властиво стековим ЗП за принципом «останнім прийшов – першим вийшов». Такі ЗП називаються буферами *LIFO* (Last In – First Out).

Час доступу до конкретної одиниці інформації, що зберігається в послідовних ЗП, є випадковою величиною. У найгіршому випадку для такого доступу потрібно переглянути весь обсяг інформації, що зберігається в цій пам'яті.

Асоціативний доступ реалізовує пошук інформації за деякою ознакою, а не за адресою. У найбільш повній версії всі слова, збережені в пам'яті, можуть одночасно перевірятися на відповідність ознакою, наприклад, на збіг певних полів слів – тегів (від *tag*) за ознакою, яку задає вхідне слово (тегова адреса). На вихід передаються слова, які задовольняють ознаку. Принцип видачі слів, якщо тег задовольняє кілька слів, і принцип записування нових даних можуть бути різними. Основна область застосування асоціативної пам'яті в комп'ютерах – кешування даних.

4.2 Основні структури напівпровідникової пам'яті

Поняття структури пам'яті

Кожна матриця М у пристрої пам'яті має систему адресних і розрядних ліній (провідників). Адресні (словникові) лінії призначені для виділення за адресою будь-якої КП. Сукупність різних адресних кодів утворює адресний простір пам'яті. Розрядні лінії записування (ЛЗП) призначені для введення в кожен розряд вибраної КП цифри 0 або 1 відповідно до вхідної інформації. Розрядні лінії зчитування (ЛЗЧ) призначені для знімання інформації, що зберігається з розряду вибраної КП. Часто використовують загальну лінію записування/зчитування (ЛЗЗ). Адресні та розрядні лінії разом називаються лініями вибірки. Якщо довжина адресного коду дорівнює k, то кількість слів N, які зберігаються в пам'яті як окремі одиниці даних, визначається зі співвідношення $N = 2^k$.

Структуру пам'яті визначає спосіб розподілу КП між адресними і розрядними лініями. За цією ознакою виділяють такі структури пам'яті. 2*D,* 3*D,* 2,5*D* і модифіковану – 2*DM* (*D* від *Dimention* – розмірність).

В системі 2*D* кожен ЕП має одну адресну лінію A_i (одну *D*), лінії записування ЛЗП і зчитування ЛЗЧ, які спільно утворюють другу *D* (рис. 4.3, *а*). У структурі 3*D* адресу розділяють на дві частини: старша A_x визначає адреси рядків, а молодша A_y – адреси стовпців; разом вони утворюють 2*D*. Лінії записування і зчитування утворюють третє *D* (рис. 4.3, *б*). У структурі 2,5*D* одна з ЛЗП або ЛЗЧ суміщена з півадресою A_{xi} чи A_{yj} (рис. 4.3, в) У модифікованій системі 2*DM* використовується загальна лінія $ЛЗЗ_j$, яка об'єднана з адресною лінією A_{yj}.

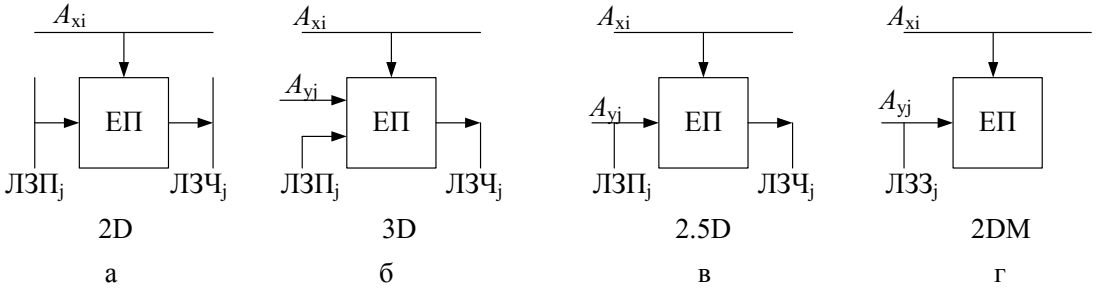


Рисунок 4.3 – Загальне поняття структури пам'яті: а) 2D; б) 3D; в) 2.5D; г) 2DM

Розглянуті структури характерні для статичних ОЗП і пам'яті типу *ROM*. Структури динамічних ОЗП мають свою специфіку.

Пам'ять зі структурою 2D

Організація мікросхеми ОЗП зі структурою 2*D* показана на рис. 4.4. До складу мікросхеми пам'яті входять:

- матриця елементів пам'яті *M*, яка містить *N* рядків і *m* стовпців (за числом розрядів слова);
- буфер *BA* і дешифратор *DCA* адрес з числом виходів $N = 2^k$;

- буферні формувачі вхідних *DI* і вихідних *DO* інформаційних сигналів в режимах записування та зчитування і *BD*;

- блок локального керування (БЛК).

При звертанні до пам яті вибираються ЕП, розташовані на збудженому виході дешифратора адреси *DCA*. Записування даних здійснюється при значенні сигналу $\overline{W} / R = 0$, а зчитування – при $\overline{W} / R = 1$. Ємність пам яті 2*D* становить $E=N \times m$ бітів.

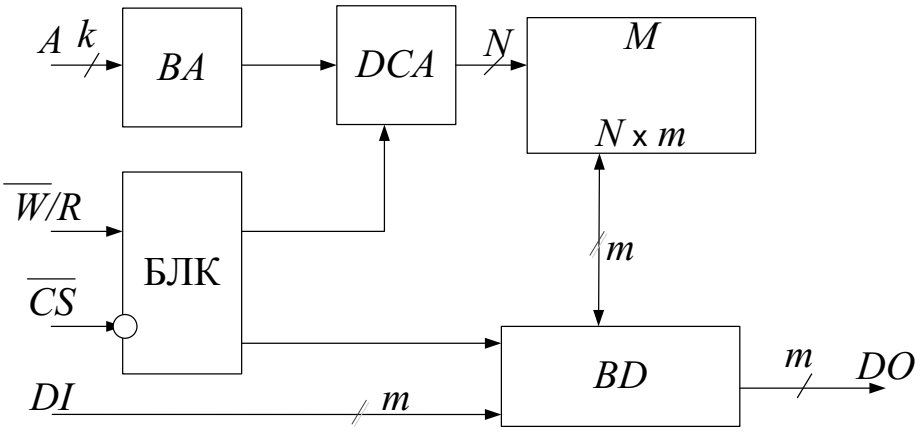


Рисунок 4.4 – Пам'ять структури 2*D*

Недоліком структури 2*D* є складність будови дешифратора адреси з числом виходів *N*, рівним числу збережених в пам'яті слів. Тому структура типу 2*D* використовується в ЗП малої інформаційної ємності.

Пам'ять зі структурою 3D

У пам'яті зі структурою 3*D* адресний код розділяється на дві рівні частини A_x і A_y (для парного *k*), кожна з яких декодується окремими дешифраторами відповідно *DCX* і *DCY* (рис. 4.5).

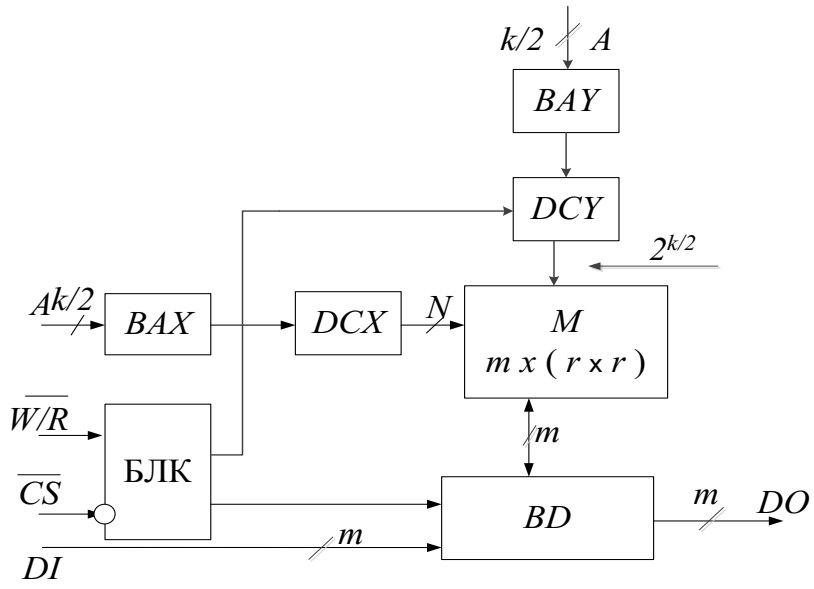


Рисунок 4.5 – Пам'ять зі структурою 3*D*

Матриця *M* складається з *m* підматриць за кількістю розрядів слова. Кожна матриця зберігає значення свого *i*-го розряду всіх *N* слів. Кожна підматриця є квадратною: *r* рядків і *r* стовпців, що записується як *r* × *r*, при цьому $r = \sqrt{N} = 2^{\frac{k}{2}}$ (рис. 4.6).

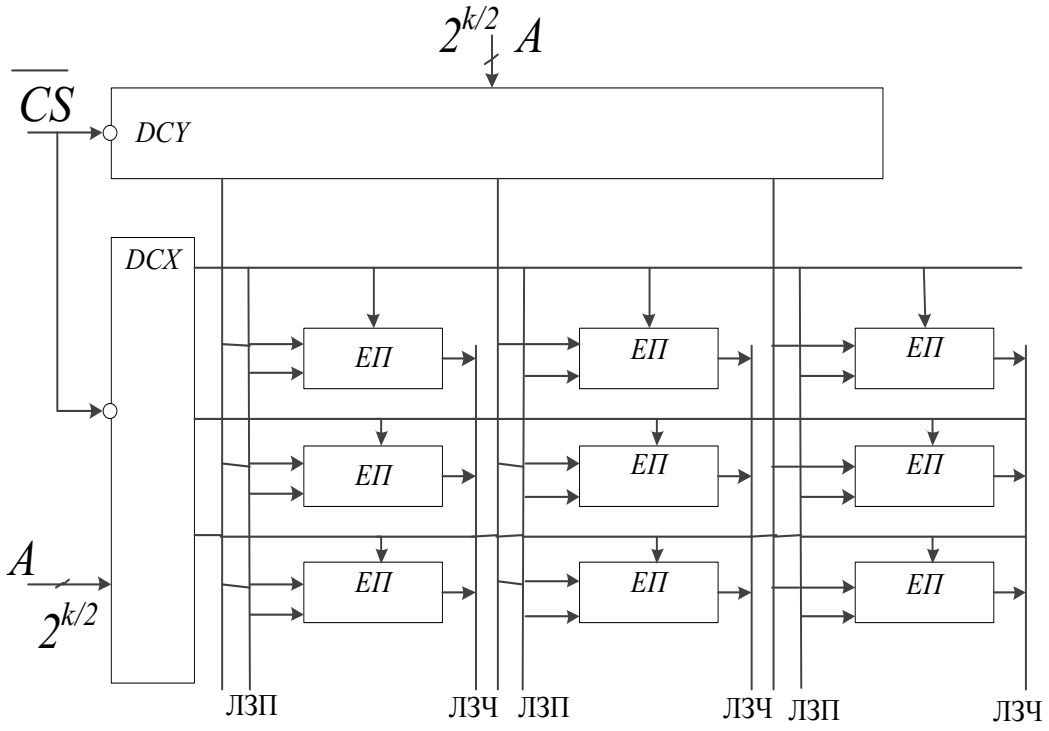


Рисунок 4.6 – Організація матриці *r* × *r* пам’яті зі структурою 3*D*

При звертанні до пам'яті вибирають *m* запам'ятовувальних елементів (по одному з кожної підматриці), які лежать на перетині рядка і стовпця виходів дешифраторів *DCX* і *DCY*. Така структура часто використовується і з однорозрядною організацією *N*×1 біт. Для цієї пам'яті ємністю 1К слів потрібно мати два дешифратори з числом виходів в кожному $N = 2^5 = 32$. Для пам’яті зі структурою 2*D* такої ж ємності дешифратор має 1024 виходи.

Недоліком структури *3D* є застосування складних ЕП, що допускають двокоординатну вибірку і складнішу структуру матриці *M*. У пам'яті за модифікованою структурою *2DM* об'єднуються переваги структур *2D* і *3D*.

Пам'ять зі структурою 2DM

У ЗУ типу *SRAM*, *ROM* зі структурою *2DM* адресний код розбивається на дві частини: $A_x = A_k\, A_{k-1} ... A_{r+1}$, що надходить на дешифратор рядків *DCX*, і $A_y = A_r\, A_{r-1} ... A_1$, що надходить на входи мультиплексорів *MUX* з організацією «$2^k \to 1$» (рис. 4.7).

Дешифратор *DCX* обслуговує 2^{k-r} рядків, кожен з яких зберігає 2^k m-розрядних слів. У кожній групі зберігаються значення однойменних

розрядів і обслуговується вона своїм буфером BD і мультиплексором *MUX* з організацією, яка показана на рис. 4.8.

Таким чином, активізований вихід дешифратора DCX вибирає рядок, а молодші розряди адреси за допомогою мультиплексорів забезпечують формування вихідного слова *D0* (по одному розряду з кожної групи).

Структура *2DM* для ЗУ типу *RAM* в загальному вигляді показана на рис. 4.9. З матриці М зчитується «довгий» рядок. Дані у відповідні групи рядка записуються або зчитуються буферами даних *BD*, які управляються сигналами з виходів дешифратора *DCY*. Буфер *BD* також визначає напрям обміну даних за допомогою сигналу $\overline{W}/R$.

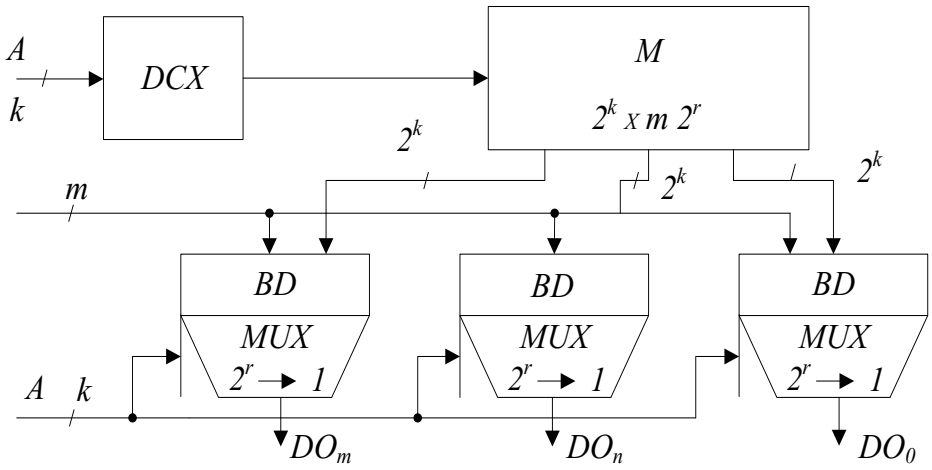


Рисунок 4.7 – Пам'ять SRAM і ROM з структурою 2DM

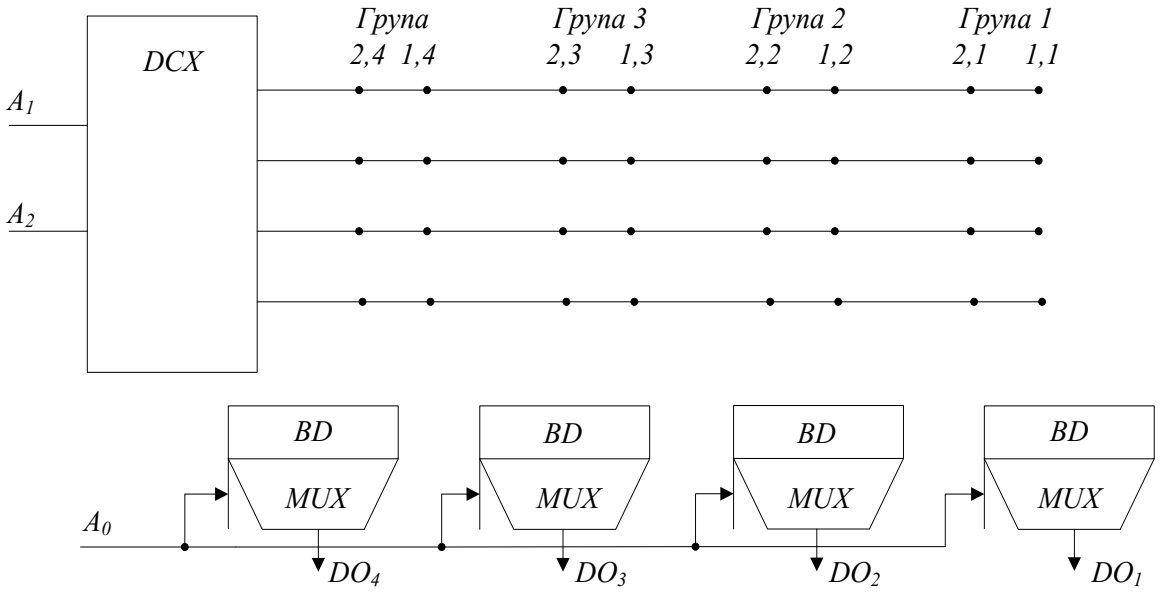


Рисунок 4.8 – Матриця М з організацією 4 × 4·2

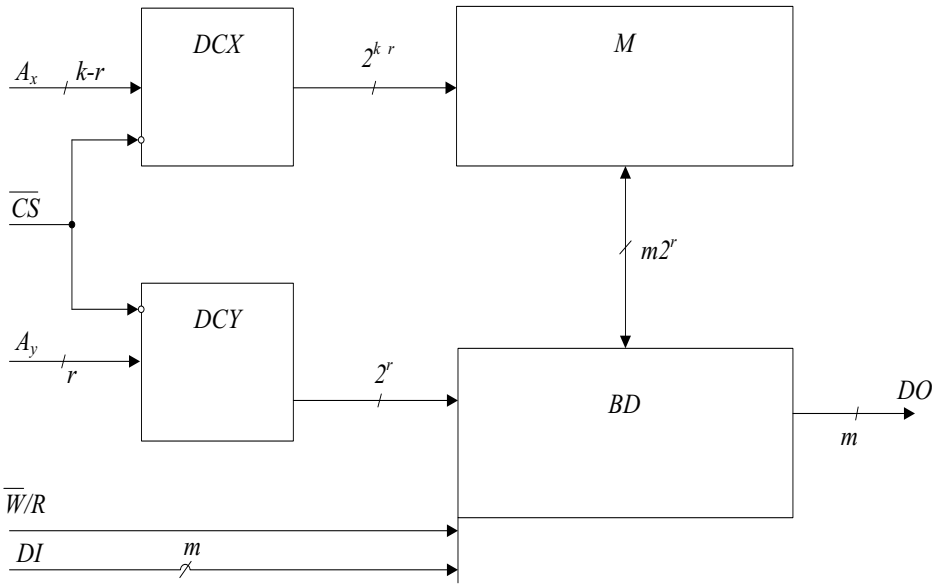


Рисунок 4.9 – Пам'ять типу RAM з структурою 2DM

Організація пам'яті зі структурою 2DM є найбільш поширеною, особливо для мікросхем великої ємності.

Пам'ять з послідовним доступом

Представниками пам'яті з послідовним доступом є буфер FIFO і стек. Буфер FIFO – це пам'ять для зберігання черг даних (списків) (рис. 4.10). Слова з черги вибираються в порядку їх надходження. Моменти записування слів в буфер і зчитування з нього задаються зовнішніми керувальними сигналами незалежно один від одного.

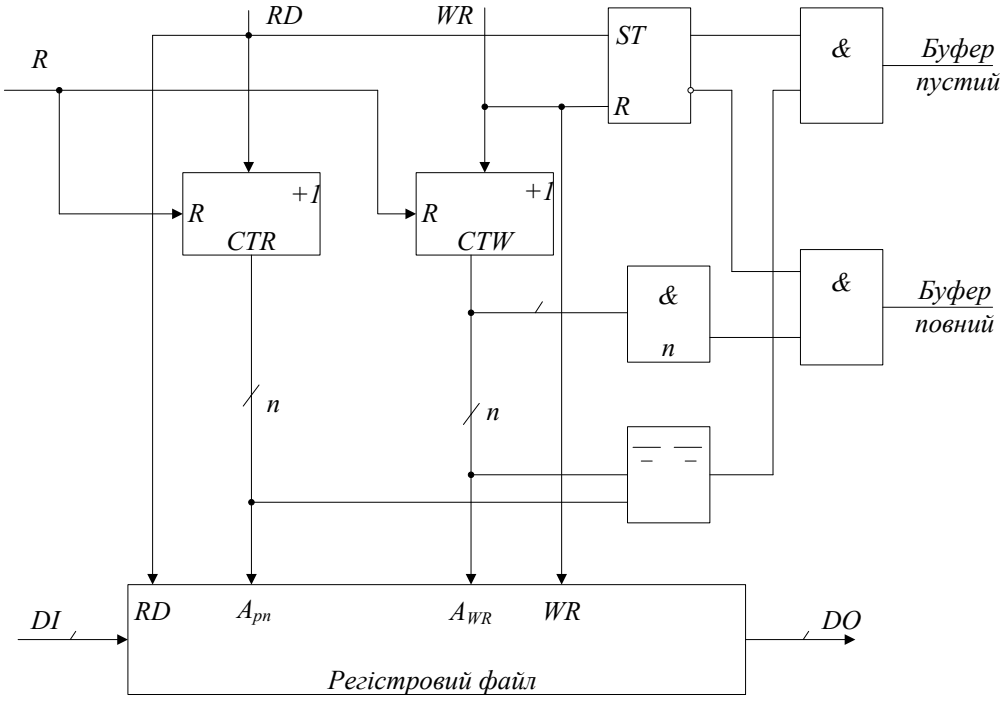


Рисунок 4.10 – Структура буфера *FIFO*

Дані надходять в темпі джерела інформації, а зчитуються з частотою, необхідною для приймача. Нове слово ставиться в кінець черги, а зчитування здійснюється з початку черги. У схемі перед початком роботи обидва лічильники адрес записування *CTW* і зчитування *CTR* скидаються.

При записуванні даних адреси збільшуються на одиницю при кожному звертанні. Те ж відбувається і при зчитуванні даних. Адреси порівнюються при зчитуванні компаратором. За допомогою схеми збігу визначається момент повного завантаження буфера. Ці перевірки автоматично виконуються при подачі сигналів *RD* або *WR*. Якщо буфер повний, то потрібно припинити прийом даних, а якщо порожній – то припинити читання.

4.3 Кеш-пам'ять

Загальна характеристика кеш-пам'яті

Кеш-пам'ять (від Cache – тайник) – це спосіб копіювання та зберігання блоків даних основної пам'яті типу *DRAM* в процесі виконання програми. Кеш-пам’ять побудована на швидкодійних тригерних ЕП, проте має невелику ємність порівняно з основною динамічною пам'яттю. Кеш зберігає обмежене число даних і тегів. Тег містить інформацію про фізичну адресу і стан даних.

При кожному звертанні до основної пам'яті спеціальний контролер перевіряє за тегом наявність цієї копії в кеші. Якщо вона є, то виробляється сигнал *Hit* (кеш-попадання) і відбувається звертання до кеш-пам'яті (рис. 4.11). Якщо копії немає (кеш-промах), то сигнал *Hit* не виробляється і виконується зчитування з ОП і одночасне розміщення лічених даних в кеші.

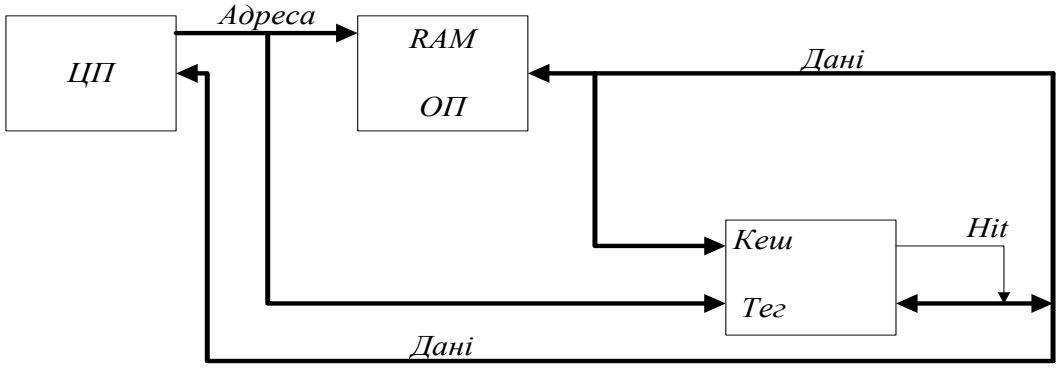


Рисунок 4.11 – Структура кеш-пам'яті

Обмін з ОП може здійснюватися двома способами:

перший: звертання до ОП поєднується з одночасним пошуком інформації в тезі. Звернення при попаданні в ОП анулюється;

другий: звертання до ОП здійснюється тільки після виявлення кеш-промаху.

У сучасних комп'ютерах кеш будують за дворівневою схемою:

- первинний кеш (L1 Cache) має обсяг десяти кілобайтів і вбудовується в процесор. Для підвищення продуктивності часто використовуються окремі кеші для команд і даних (Гарвардська архітектура);

- вторинний кеш (L2 Cache), зазвичай встановлюють на системній платі, він має обсяг декілька мегабайтів.

Більшість прикладних програм має циклічний характер і багаторазово використовує одні й ті ж дані, тому наявність кеша зменшує кількість звернень до відносно повільної ОП.

Повністю асоціативний кеш

Залежно від способу установлення відповідності між даними в кеші і ОП використовують такі структури кеш-пам'яті:

- повністю асоціативний кеш;
- кеш з прямим розміщенням;
- набірно-асоціативний кеш.

У повністю асоціативної кеш-пам'яті кожна комірка зберігає дані, а в полі Тег знаходиться повна фізична адреса одиниці інформації, копія якої записана (рис. 4.12).

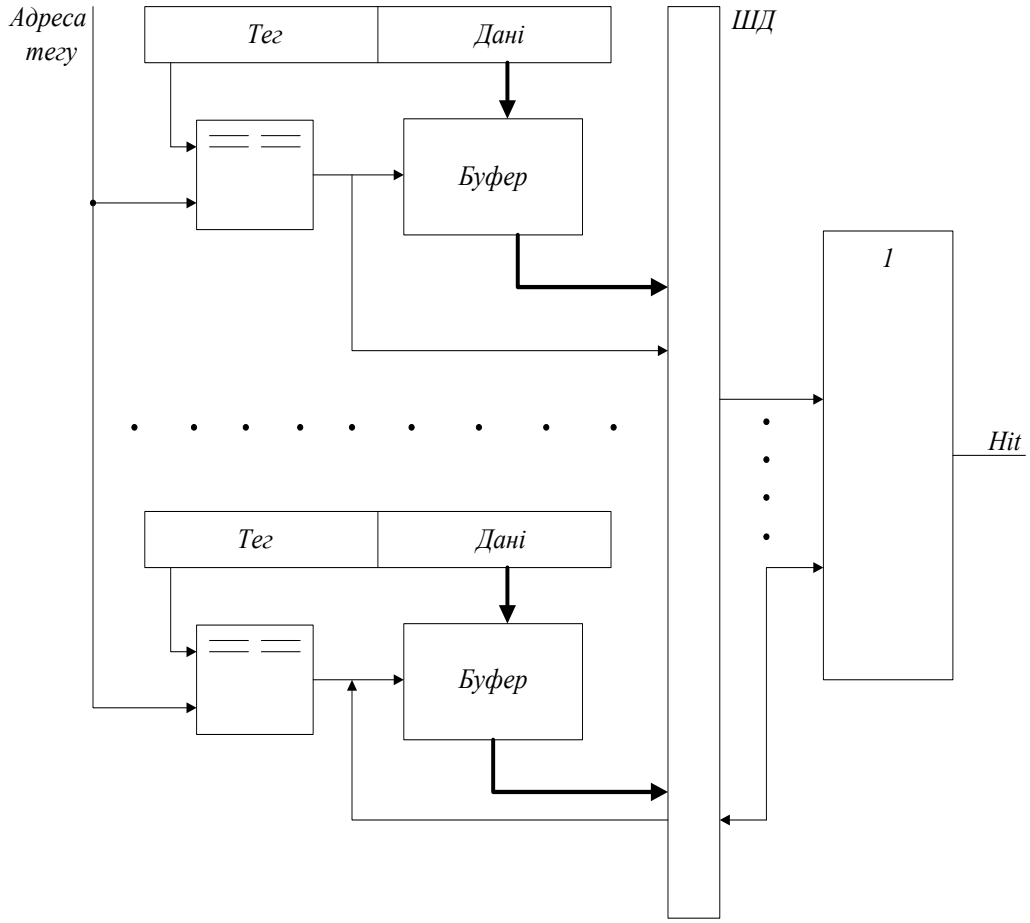


Рисунок 4.12 – Структура повністю асоціативного кеша

Під час обмінів записувана фізична адреса даних порівнюється з полем Тег всіх комірок. Якщо виявляється збіг з адресою будь-якої комірки, то вона виставляє сигнал Hit. При читанні, коли значення сигналу Hit – 1, дані видаються на ШД. Якщо збігу немає (Hit = 0), то при зчитуванні з ОП дані разом з адресою розміщуються у вільній або давно не використовуваній комірці кеш-пам'яті.

Рядки кеша завантажуються адресами і даними під час операцій зчитування з ОП. Дані без копії в кеші записуються тільки в ОП. Дані, що мають копії в кеші, записуються в ОП наскрізним або зворотним способом.

Під час наскрізного записування дані одночасно завантажуються в копію кеша і в ОП. Затрачається час на відносно велику тривалість записування в *DRAM*, що знижує швидкодію всієї пам'яті.

Під час зворотного записування дані спочатку записуються в свою копію і позначаються ознакою модифікації. Після звільнення системної шини кеш-контролер переписує модифікований рядок в ОП. Зворотний спосіб записування складніший наскрізного, однак більш ефективний. В обох випадках кеш-контролер забезпечує когерентність, тобто узгодження даних в кеші і ОП.

Повністю асоціативна кеш-пам'ять забезпечує найбільшу функціональну гнучкість, однак є дуже складним пристроєм.

Кеш-пам'ять з прямим розміщенням

У кеш-пам'яті з прямим розміщенням ОП умовно розбивається на сторінки, наприклад, 5 і 2. Розмір кожної сторінки збігається з розміром кеш-пам'яті. Дані в кеші зберігаються в рядках, які є наборами байтів (часто 32 байти). Якщо кеш-пам'ять має, наприклад, 256 рядків, то її обсяг дорівнює 256 × 32 = 8 кбайт. При цьому обсяг ОП повинен становити 512 × 8 кбайт = 4 Мбайт. У рядках кеша можуть зберігатися дані будь-якої сторінки ОП. Адреса від процесора ділиться на три частини (рис. 4.13).

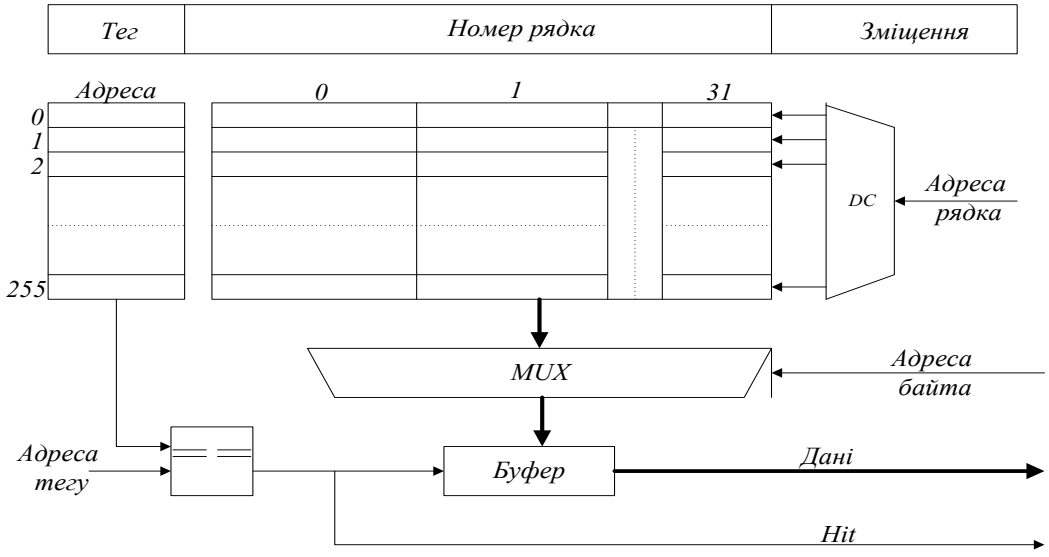


Рисунок 4.13 – Організація кеш-пам'яті з прямим розміщенням

Молодші розряди (зміщення) визначають положення молодшого байта 32-розрядного слова. Середні розряди адреси задають номер рядка. Разом номери рядка і байта в ній називаються індексом. Інформація про номер сторінки ОП, дані з якої займають відповідний рядок в кеші, називається тегом.

Кожен тег займає елемент пам'яті, який пов'язаний зі своїм рядком. Сукупність цих комірок утворює пам'ять тегів. Розрядність комірок тегу повинна бути достатньою для записування номера сторінки ОП. Наприклад, для 256 сторінок довжина комірки тегу повинна становити 8 біт.

При звертанні до ОП спочатку зчитується рядок кеша із заданим індексом, а потім порівнюються значення тегів даного рядка і заданого в адресі від процесора. При кеш-попаданні сигнал Hit = 1 і слово даних мультиплексуються з обраного рядка і пересилаються в процесор. При кеш-промаху сигнал Hit = 0 і дані вибираються з ОП; при цьому за вказаною адресою завантажується весь рядок кеша (пакетний обмін).

Кеш з прямим розміщенням має просту схему, малу розрядність тегу, однак і вагомий недолік – не допускається розміщення в кеші рядків з однаковими індексами, але різними тегами. Це призводить до безперервної черги кеш-промахів. Цей недолік усувається в набірно-асоціативному кеші.

Набірно-асоціативний кеш

У набірно-асоціативній структурі кеш-пам'ять розділяється на набори з невеликим числом рядків, кратним двійці, тобто 2, 4, 8 і т. д. Середні розряди адреси від процесора визначають не один рядок, а весь набір.

Кожен рядок в наборі обслуговується власним блоком кеш-пам'яті, тегом, компаратором і буфером даних. Це подібно паралельній і узгодженій роботі декількох каналів прямого заміщення. Контролер кеш-пам'яті приймає рішення про те, в якому із рядків наборів розміщується черговий пакет даних.

У простому випадку кожен пакет даних з ОП може завантажуватися в один з двох рядків у наборі. Такий кеш містить два блоки кеш-пам'яті: А – для парних рядків і В – для непарних (рис. 4.14). Такі кеші називаються двовхідними.

Сторінку даних з ОП можна помістити лише в той набір, номер якого дорівнює адресі сторінки за модулем, наприклад, 64 або 128 і т. д. Місце сторінки в наборі довільне.

Одночасно зчитуються і порівнюються парні і непарні рядки (слова з них). Зчитування даних йде від того блоку, де є збіг тегу і тегів адреси від процесора. За відсутності збігів виконується звертання до ОП і заміщення рядка в одному з блоків кеша.

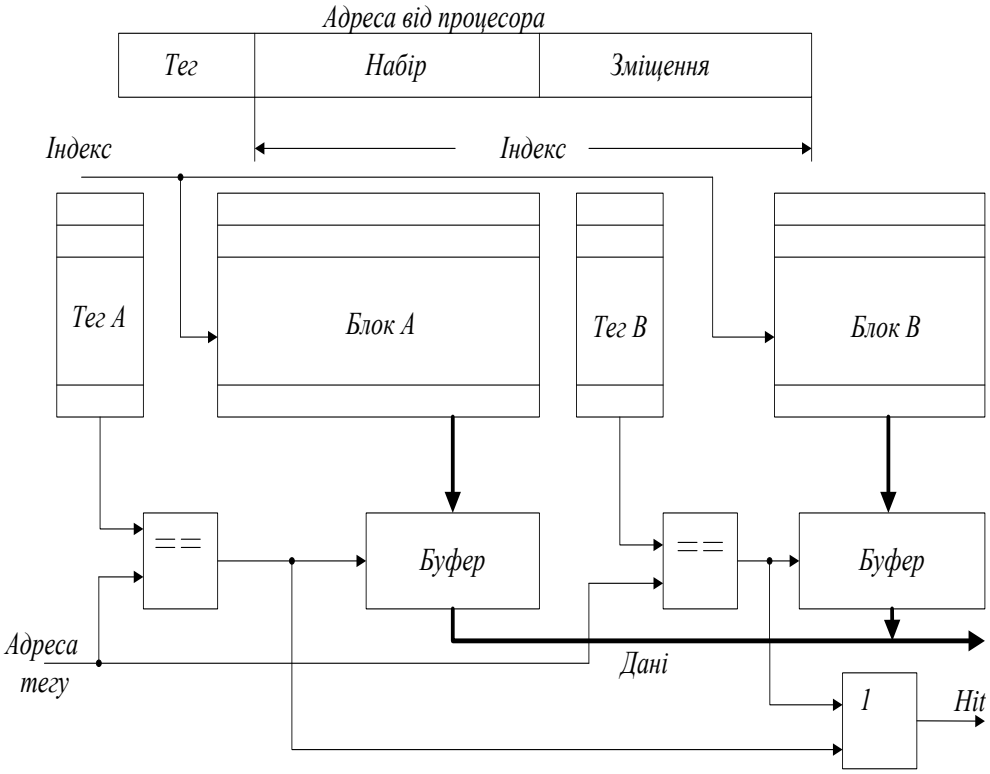


Рисунок 4.14 – Двовхідний набірно-асоціативний кеш

4.4 Постійна пам'ять

Загальна характеристика постійної пам'яті

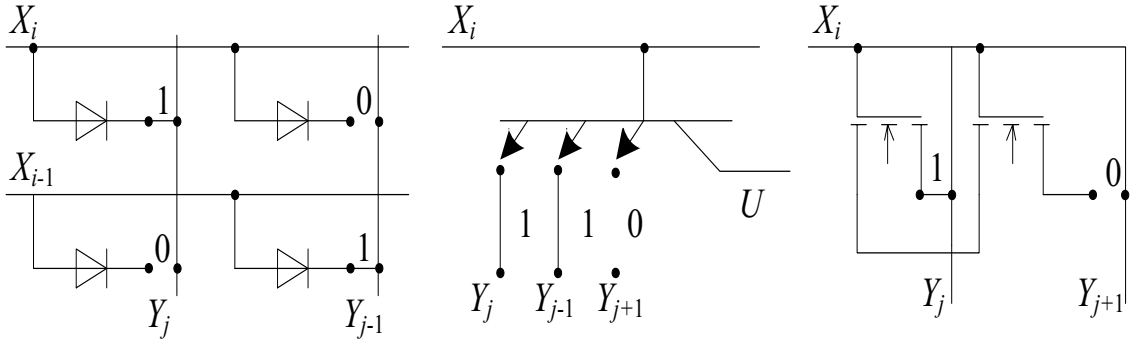
Постійна пам'ять призначена для зберігання програм, констант, табличних функцій та іншої інформації, яка записується заздалегідь і не змінюється в процесі поточної роботи комп'ютера. Вона застосовується також у перетворювачах кодів, знакогенераторах, в мікропрограмних пристроях керування. Спільним для всіх мікросхем постійної пам'яті є енергонезалежність, словникова організація та застосування режиму зчитування як основного.

Мікросхеми постійної пам'яті поділяються на такі групи:

- ПЗП або *ROM* (*Read Only Memory*) – програмуються одноразово заводом-виробником, часто називаються масковими;
- ППЗП або *PROM* (*Programmable ROM*) – програмуються одноразово електричним способом користувачем;
- РПЗП-УФ або *EPROM* (*Erasable PROM*) – програмуються багаторазово (репрограмуруются) з ультрафіолетовим стиранням та електричним записуванням;
- РПЗУ-ЕС або *EEPROM* (*Electrical EPROM*) – програмуються і стираються багаторазово електричним способом.

Мікросхеми постійних запам'ятовувальних пристроїв

У мікросхемах ПЗП і ППЗП елементами пам'яті є діоди, біполярні і МОН-транзистори, а також КМОН-структури. Елементи пам'яті розміщуються у вузлах матриці, утворених адресними лініями рядка X_i і стовпця Y_j. Стан «1» відповідає з'єднанню ЕП з розрядною лінією, а стан «0» означає відсутність зв'язку (для прямих виходів мікросхеми пам'яті). У маскових ПЗП розрив з'єднань ЕП з розрядними лініями при записуванні інформації виконує завод-виробник (рис. 4.15).



а б в

Рисунок 4.15 – Елемент пам'яті маскових ПЗП:
а) на діодах, б) на багатоемітерних транзисторах, в) на МОН-транзисторах

Характеристики ряду серій ВІС ПЗП подано в табл. 4.1, а їх умовні позначення показано на рис. 4.16.

Таблиця 4.1– Характеристики деяких ПЗП

Тип мікросхеми	Ємність, біт	Технологія	Час вибірки t_A, нс	P_A, Вт
КР1656РЕ4	8К×8	ТТЛШ	40	90
КР568РЕ3	16К×8	n-МОН	300	300
КМ568РЕ5	128К×8	n-МОН	200	300
К536РЕ2	32К×8	КМОН	500	20

Мікросхема К563РЕ2 має вмонтовану схему самоконтролю і виправлення одиничних помилок за допомогою коду Хемінга із встановленням ознаки-помилки на виході К1. Коректор можна вимкнути сигналом К2 = 0, при цьому дані зчитуються без виправлення помилок. У мікросхемах ППЗП, які постачаються заводом-виробником, всі ЕП, розташовані у вузлах матриці, з'єднані зі стовпцями плавкими перемичками. Записування інформації в ППЗП здійснюється користувачем перепалюванням перемичок або електричним пробоєм діодів Шотткі.

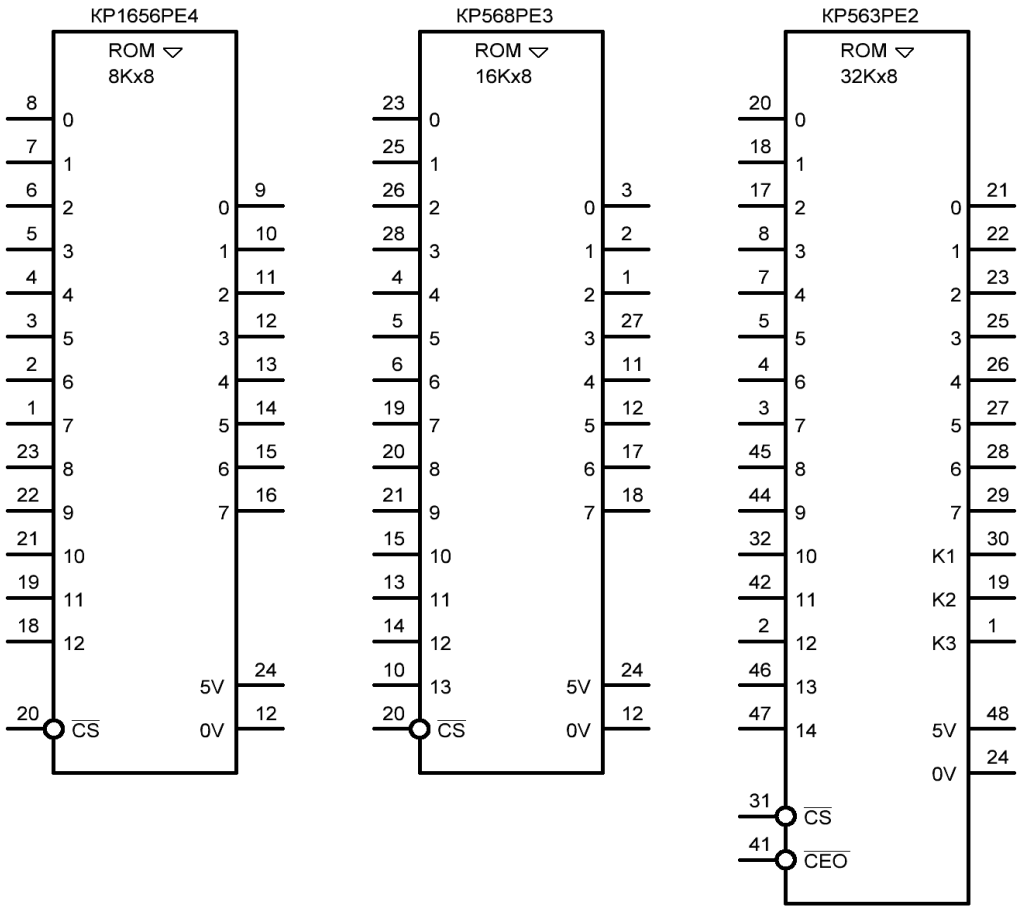


Рисунок 4.16 – Умовне позначення ВІС ПЗП

Для плавких перемичок використовують тонкі плівки з ніхрому або полікристалічного кремнію; струм перепалювання дорівнює 50 – 100 мА, в результаті чого перемичка необоротно руйнується. Пам'ять побудована за системою 2,5D з організацією 4×2 біт. Кожен багатоемітерний транзистор в матриці являє собою два ЕП (за числом емітерів) і програмується на запам'ятовування дворозрядного слова.

Мікросхема ППЗП за структурою аналогічна масковим, однак допускає одноразове програмування користувачем. Найбільш поширені мікросхеми ППЗП серії К556 (рис. 4.17), виготовлені за ТТЛШ-технологією. Функціональний склад цієї серії містить мікросхеми ємністю від 1 кбіт до 64 кбіт зі словарною (чотири- і восьмирозрядною) організацією та часом вибірки 45 – 85 нс і потужністю споживання 0,6 – 1 Вт.

Різновидом ППЗУ є програмовані логічні матриці (ПЛМ або РШ), до яких відносяться мікросхеми КР556 (РТ1, РТ2). Вони мають ідентичні характеристики і конструктивні параметри, проте відрізняються за типом виходу: РТ1 мають вихід з відкритим колектором, РТ2 – з трьома станами (рис. 4.17). Обидві мікросхеми призначені для реалізації пристроїв, що виконують логічні операції над двійковими змінними.

Проаналізовані ПЛМ розраховані на 16 вхідних змінних, інверсії від яких отримують всередині мікросхеми. Функціональні можливості даних ПЛМ: вісім логічних сум, кожна з яких може містити до 48 логічних добутків з 16 змінних і їх інверсій.

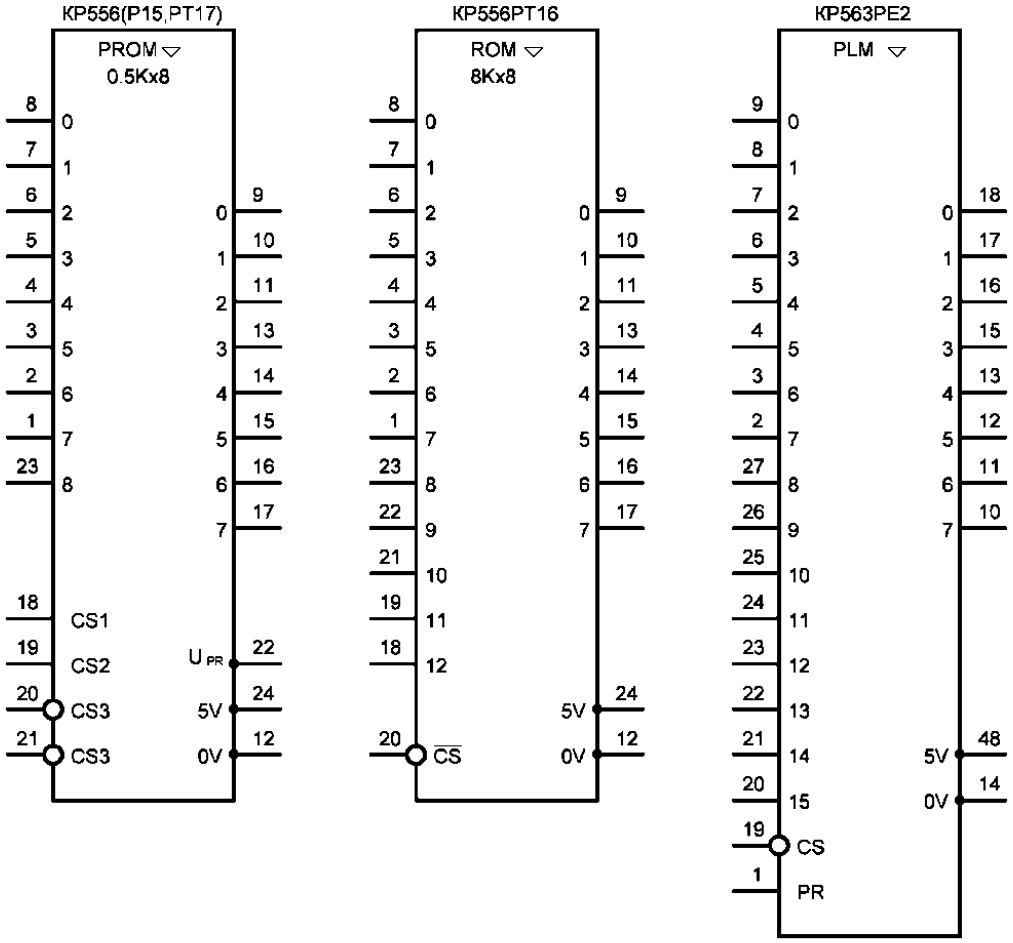


Рисунок 4.17 – Умовне позначення ВІС ППЗП

Програмування ПЛМ полягає в перепалюванні плавких перемичок у необхідних вузлах. Це виконують вмонтовані в ПЛМ спеціальні схеми, які керуються сигналом PR. Спочатку програмують матрицю І, потім матрицю 4М і вихідні підсилювачі. Для керування доступом до мікросхеми використовують сигнал $\overline{CS}$, наявність якого дозволяє також нарощувати число вхідних змінних і вихідних функцій способом об'єднання декількох ПЛМ.

Широко застосовуються мікросхеми ПЛМ, які програмуються способом замовного фотошаблона. Такі мікросхеми внесені в комплект деяких мікропроцесорних серій ВІС як ПЗП мікрокоманд.

Характеристики найбільш поширених мікросхем РПЗП-ЕС на МНОН-транзисторах подано в табл. 4.2. А умовні графічні позначення деяких з них показано на рис. 4.18.

Здатність до багаторазового програмування забезпечується застосуванням ЕП з якостями керованих перемичок. Цю функцію виконують транзистори типу ЛІЗМОП або МНОН. Вони являють собою спеціальні МОН-транзистори, в яких область під затвором і підкладкою може накопичувати і зберігати заряд, що створюється електричним способом.

Таблиця 4.2 – Характеристики найбільш поширених мікросхем РПЗП-ЕС

Тип мікросхеми	Тип транзистора	Ємність, біт	t, мкс	P, мВт	U, В
КР558РР1	p-МНОН	256×8	5,0	300	5-12
КР1601РР1	p-МНОН	1К×4	1,8	625	5-12
КР1601РР3	p-МНОН	2К×8	1,6	825	5-12
КР558РР2	n-МНОН	2К×8	0,35	480	5
КР558РР3	n-МНОН	8К×0,4	0,4	400	5
КР1611РР1	n-МНОН	8К×8	0,3	850	5

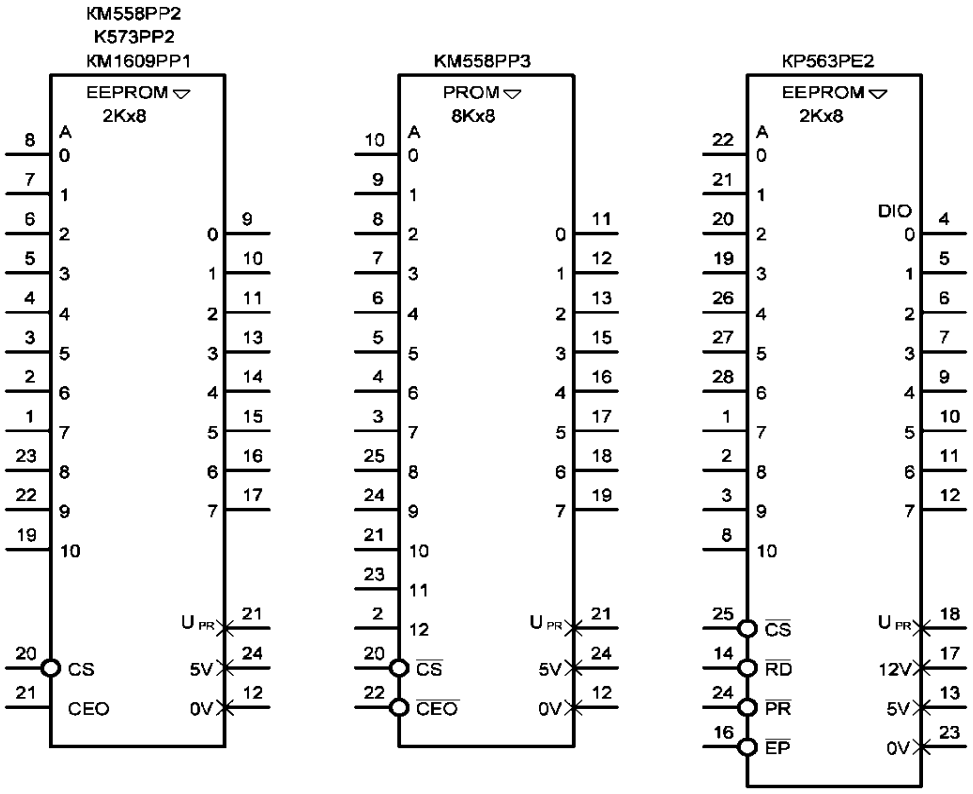


Рисунок 4.18 – Умовне позначення мікросхем РПЗП-ЕП на МНОН-транзисторах

Для програмування ППЗП і репрограмованих ПЗП використовують спеціальні пристрої – програматори. Для ряду мікросхем пам'яті програмування виконують за допомогою комп'ютера.

Контрольні запитання

1. Охарактеризуйте поняття запам'ятовувального пристрою.
2. Опишіть основні параметри запам'ятовувальних пристроїв.
3. Опишіть основні функції пам'яті.
4. Охарактеризуйте поняття зовнішньої та внутрішньої пам'яті.
5. В чому полягають відмінності зовнішньої та внутрішньої пам'яті?
6. Охарактеризуйте структури пам'яті 2D, 3D, 2,5D, 2DM.
7. Що означає позначення 2D, 3D, 2,5D, 2DM?
8. Опишіть способи доступу до даних у напівпровідниковій пам'яті.
9. Для чого використовується пам'ять з послідовним доступом?
10. Охарактеризуйте поняття динамічного запам'ятовувального пристрою.
11. В чому різниця між статичними і динамічними запам'ятовувальними пристроями?
12. Що таке кеш-пам'ять?
13. Що означає приставка «кеш»?
14. Наведіть приклади використання кеш-пам'яті.
15. Опишіть види кеш-пам'яті.
16. Наведіть структуру кеш-пам'яті.
17. Охарактеризуйте кеш-пам'ять за дворівневою системою.
18. Що являє собою пам'ять типу DRAM?
19. Що означає позначення DRAM?
20. Для чого призначена постійна пам'ять?
21. Наведіть приклади постійної пам'яті.
22. Охарактеризуйте пам'ять ROM.
23. Що означає позначення ROM?
24. Охарактеризуйте пам'ять PROM.
25. Що означає позначення PROM?
26. Охарактеризуйте пам'ять EPROM.
27. Що означає позначення EPROM?
28. Охарактеризуйте пам'ять EEPROM.
29. Що означає позначення EEPROM?
30. Охарактеризуйте пам'ять з послідовним доступом.

5 ПРОГРАМОВАНІ ЛОГІЧНІ СХЕМИ

5.1 Концепція будови запрограмованих схем

Кожна програмована інтегральна схема містить певний набір логічних схем, а також структуру шляхів, що уможливлює реалізацію сполучень між окремими логічними схемами, відповідно до потреб впроваджуваного проекту. Наочно це показано на рисунку 5.1. На рисунку 5.1, а показано приклад набору елементів і шляхів: два логічних елементи A і B, а також шляхи a, b, c, d, e, f. Якщо захочемо сполучити вихід елемента A з входом елемента B, то потрібно виконати таке сполучення як показано, наприклад, на рисунку 5.1, б.

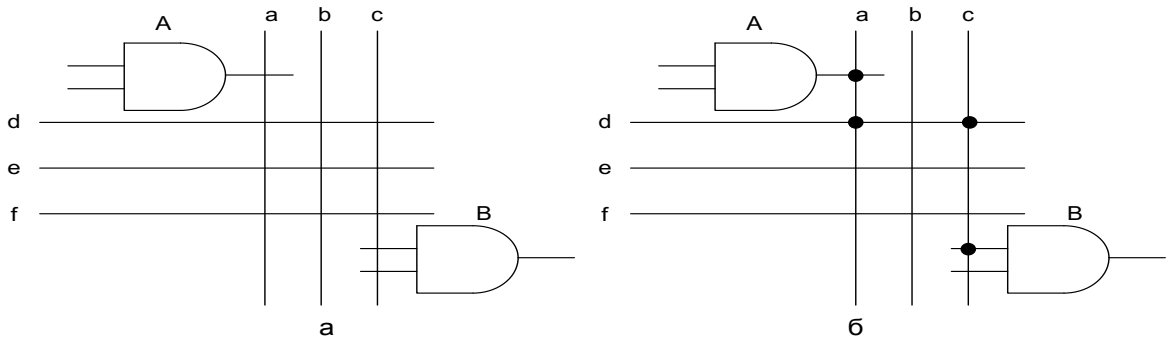


Рисунок 5.1 – а) Приклад набору елементів і шляхів в програмованій схемі, б) приклад реалізації сполучень між елементами A і B

Реалізація сполучень між доступними схемами за допомогою доступних шляхів визначається як програмування схеми або як конфігурування схеми.

Тепер з'ясуємо, яким способом, використовуючи готову (замкнену) інтегральну схему, можна програмувати (виконувати) сполучення між відповідними шляхами в інтегральній структурі. Відомих є декілька методів реалізації таких сполучень. Один із способів полягає у використанні структур запобіжників. Якщо між двома шляхами помістимо запобіжник, так як на рисунку 5.2, а, то, поки цей запобіжник не перепалиться, буде існувати сполучення між шляхами a і b. Після перепалення запобіжника сполучення перерветься (рисунок 5.2, б).

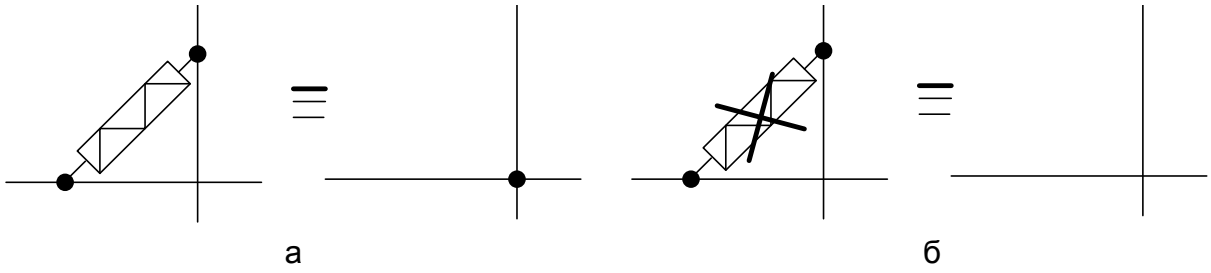


Рисунок 5.2 – Програмування сполучення з використанням запобіжника: а) робочий запобіжник, б) спалений запобіжник

При використанні запобіжників в програмованій схемі спочатку виконуються усі можливі сполучення і програмування полягає в усуненні непотрібних зв’язків, тобто у спаленні відповідних запобіжників. Програмування структури сполучень відбувається в спеціальних програматорах. В такому випадку процес програмування є необоротним. Погано запрограмована схема придатна лише для викидання.

Застосовуються також рішення, в яких замість запобіжників використовуються так звані антизапобіжники, тобто елементи, що на початку не передають, а лише після програмування стають перешкодою. В такому випадку спочатку в структурі немає жодних сполучень між шляхами і тільки процедура програмування активізує відповідні зв’язки.

В іншому випадку, замість запобіжників використовуються транзистори з подвійним (плаваючим) елементом. Зображення такого транзистора показано на рисунку 5.3. Поки на цей додатковий елемент Gf, що ізольований від середовища, не буде подано навантаження, транзистор поводиться як звичайний транзистор типу MOS – при відповідній полярності транзистор проводить струм між стоком D і джерелом S. Якщо, натомість, на додатковий елемент буде подано навантаження, то при такій самій поляризації, як перед цим, транзистор буде відімкнутий і не проводитиме струм між стоком D і джерелом S. Оскільки плаваючий елемент є ізольованим від середовища, то навантаження, подане на цей елемент, може втриматися довгий час (мінімум 10 років) і стан сполучення, реалізованого транзистором, можна вважати сталим.

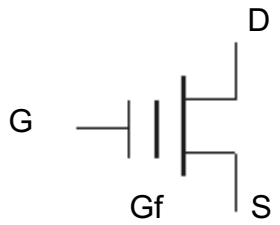


Рисунок 5.3 – Транзистор з подвійним елементом

Програмування схем з транзисторами з подвійним елементом реалізовується в спеціальних програматорах. Однак, цього разу, у випадку помилки можна змінити стан запрограмованих сполучень (усіх одночасно), використовуючи для цього спеціальний пристрій, в якому схема може бути просвітлена ультрафіолетовим промінням, що спричиняє усунення навантажень з плаваючих елементів. Далі схему можна повторно запрограмувати.

В іншому випадку для реалізації сполучень використовуються транзистори MOS, що керуються сигналами із співпрацюючих тригерів FF; приклад сполучення показано на рисунку 5.4. Якщо в тригері буде запам’ятований логічний нуль, то транзистор MOS не передає. Якщо,

натомість, в тригері буде запам’ятована логічна одиниця, то транзистор буде передавати.

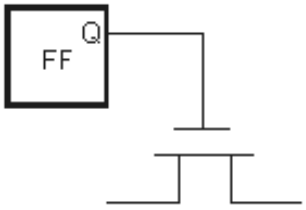


Рисунок 5.4 – Реалізація програмованих сполучень з використанням транзисторів MOS і співпрацюючих тригерів (комірок пам’яті)

В цьому випадку програмування структур полягає на записуванні інформації до тригерів, співпрацюючих з відповідними транзисторами MOS. Цю інформацію можна ввести на початку роботи з пристроєм, безпосередньо після вмикання живлення. Інформація про конфігурацію сполучень в програмованій схемі буде зберігатися до часу вимикання живлення і при повторному вмиканні живлення має бути повторно введена. Зауважмо, що в попередньо розглянутих рішеннях запрограмована конфігурація сполучень утримувалась незалежно від того, чи живлення увімкнене, чи ні.

Ще інша можливість програмування сполучень полягає у використанні схем мультиплексорів (рисунок 5.5). Це рішення вигідне тоді, коли на даний шлях ми хочемо передати сигнал від одного з декількох шляхів. Вибір відповідного вхідного шляху виконується за допомогою адресних входів, керованих допоміжними тригерами. Програмування сполучень полягає, як і дотепер, у записуванні відповідної інформації до відповідних тригерів. Додамо, що на практиці це реалізовується без виймання програмованої схеми з пристрою.

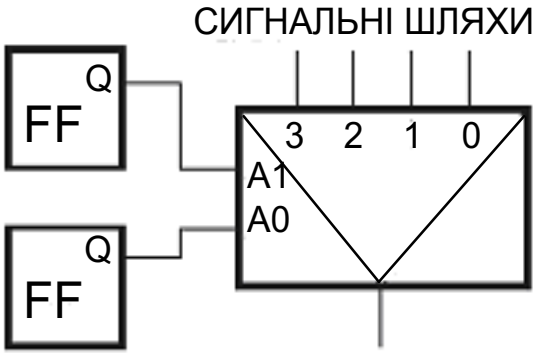


Рисунок 5.5 – Сполучення сигнальних шляхів з використанням мультиплексора

Перед тим, як перейдемо до вказаних прикладів програмованих схем, потрібно вивчити умови рисування схем, прийняті в програмованих

схемах. На рисунку 5.6, а показано класичні правила рисування зображення логічних елементів (на прикладі елемента AND), а на рисунку 5.6, б показано правила, що застосовуються в програмованих схемах. Зауважмо, що в класичних правилах шляхи вхідних сигналів зменшуються безпосередньо до символу елемента. В нових правилах до символу елемента підходить одна лінія, яку перетинають перпендикулярно лінії сигнальних шляхів. Якщо хочемо зазначити, що певний сигнальний шлях має бути під'єднаний до елемента, то на перетині цього шляху з лінією, що доходить до елемента, вставляється символ сполучення.

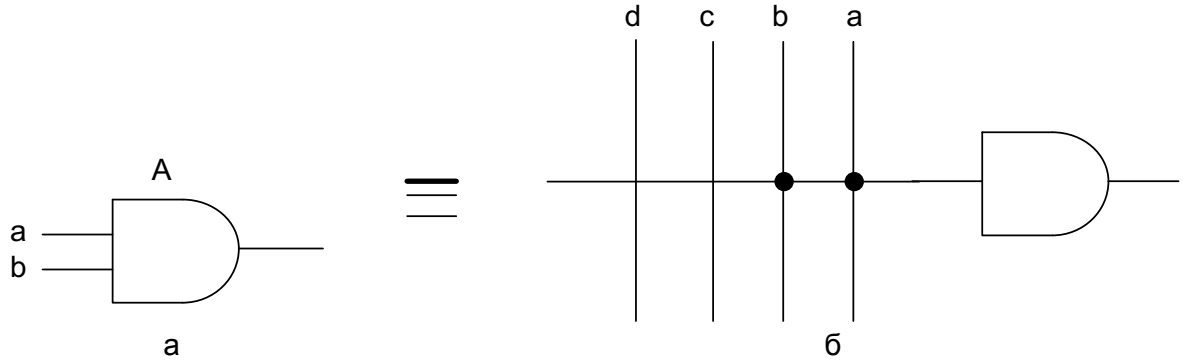


Рисунок 5.6 – Два правила позначення сполучень, що підходять до входів елементів: а) класичне, б) застосовуване в програмованих схемах

В обох випадках, показаних на рисунку, до входів елемента підходять сигнали a і b. Сигнали c і d не підходять до входів елемента.

5.2 Прості програмовані схеми

Посеред виготовлюваних програмованих схем є схеми різної складності. Застосовуються також різні назви. Прості схеми невеликої складності найчастіше містять набори елементів добутків, сум і тригерів. Зустрічаються назви PAL (Programmable Array Logic), PLA (Programmable Logic Array), PLD (Programmable Logic Devices), GAL (Generic Array Logic). Ці назви пов'язані або з певною внутрішньою архітектурою програмованої схеми, або це фірмові назви. Далі розглянемо приклад схеми з цієї групи схем, а саме схему GAL16V8.

На рисунку 5.7 показано логічне позначення схеми GAL16V8. (Зазначено тільки інформаційні сигнали). Позначення, що починаються з одної літери I, відносяться до входів схеми; позначення, що починаються з літери O, відносяться до виходів схеми; а позначення, що починаються з літер IO, відносяться до виводів схеми, які можуть виконувати функцію входу або виходу.

Складова в назві схеми 16V8 інформує про те, що, використовуючи схему, можна мати в розпорядженні 16 входів і до 8 виходів. Беручи до уваги доступний набір кінців схеми, показаний на рисунку 5.7, можна зауважити, що загальна кількість одночасно використовуваних входів і

виходів не може перевищувати 18. Літера V інформує, що активний рівень вихідного сигналу може бути змінним (низький або високий). Схема GAL16V8 може бути запрограмована на роботу в різних конфігураціях. Найчастіше використовуються такі конфігурації: комбінаційна, позначена як 16V8C, а також регістрова, позначена як 16V8R.

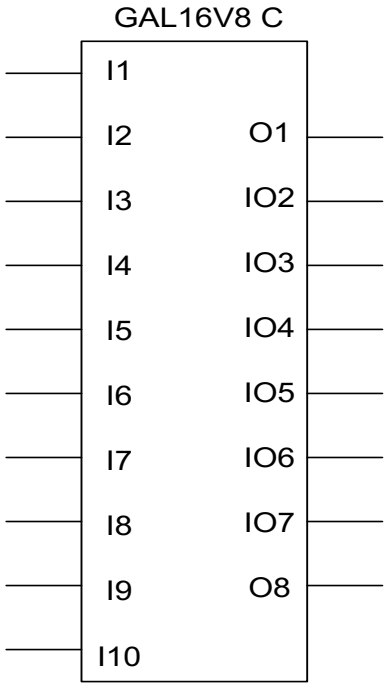


Рисунок 5.7 – Логічне позначення схеми GAL16V8C

Внутрішня структура схеми GAL16V8C містить 8 ідентичних сегментів. На рисунку 5.8 показано один з цих сегментів.

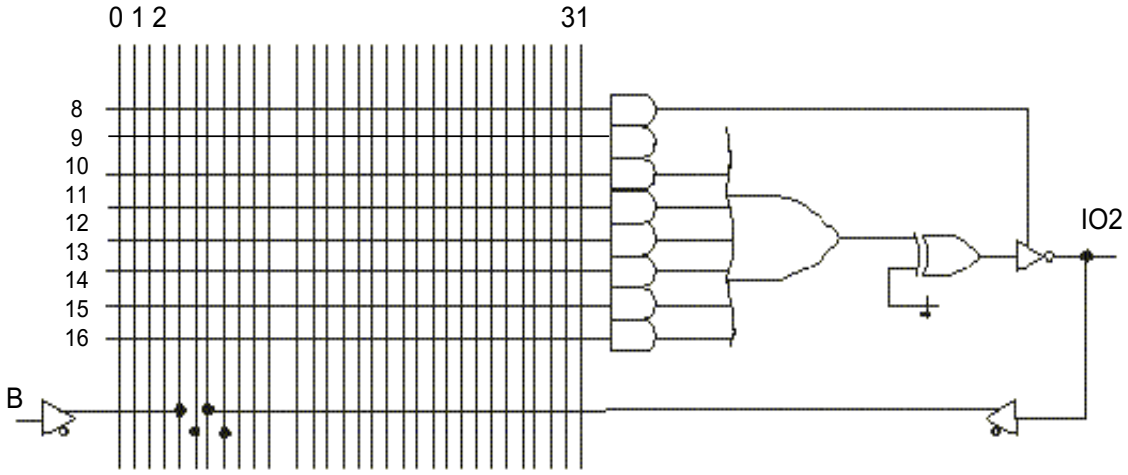


Рисунок 5.8 – Один сегмент схеми GAL16V8C

Сегмент уможливлює реалізацію функції суми добутків. Сполучення між виходами семи добутків і входами суми є сталі і не потребують програмування. Можна, натомість, програмувати сполучення з входами

відповідних добутків. До кожного добутку можна додати до 16 сигналів або їх заперечення (розрядні лінії 8 – 15 на рисунку 5.8). Вхідні сигнали і їх заперечення доступні на 32 шляхах (перпендикулярні шляхи 0 – 31 на рис. 5.8). Зауважмо, що кожен сигнал, який приходить ззовні (наприклад, I3), відразу заперечується. Це забезпечують схеми, пов'язані з відповідними входами (наприклад, схема I3), які передають всередину структури вихідні сигнали як у звичайному вигляді, так і в запереченому.

Вихід зі схеми суми може бути заперечений або ні, залежно від того, чи програмовані входи елемента XOR сполучені з логічною одиницею, чи з логічним нулем. (Пам'ятаємо, що при логічному нулі на одному з входів елемента XOR на виході елемента з'явиться те саме логічне значення, що є на другому вході, а при логічній одиниці на одному з входів елемента XOR на виході з'явиться заперечене логічне значення відносно того, що є на другому вході елемента.) Вихідний сигнал з елемента XOR передається на вихід (IO2) через тристабільний елемент. Тристабільний елемент керується сигналом з восьмого добутку в сегменті. При закритому тристабільному елементі вихід IO2 виконує роль входу, а при відкритому тристабільному елементі – роль виходу, причому вихідний сигнал є одночасно вхідним сигналом для схеми GAL 16V8R, завдяки чому можливо реалізовувати послідовнісні схеми.

В регістровій конфігурації схеми GAL 16V8R в сегменті додатково є доступний тригер. Схема логічної частини сегмента показано на рис. 5.9. В цій конфігурації для записування інформації до тригера використовується загальний сигнал CLK, поданий ззовні і доведений до часових входів тригерів у всіх сегментах схеми. Вихідний тристабільний елемент керується загальним сигналом OE, поданим ззовні. Тому «звільнений» восьмий елемент використовується як додатковий елемент, під'єднаний до входу елемента суми.

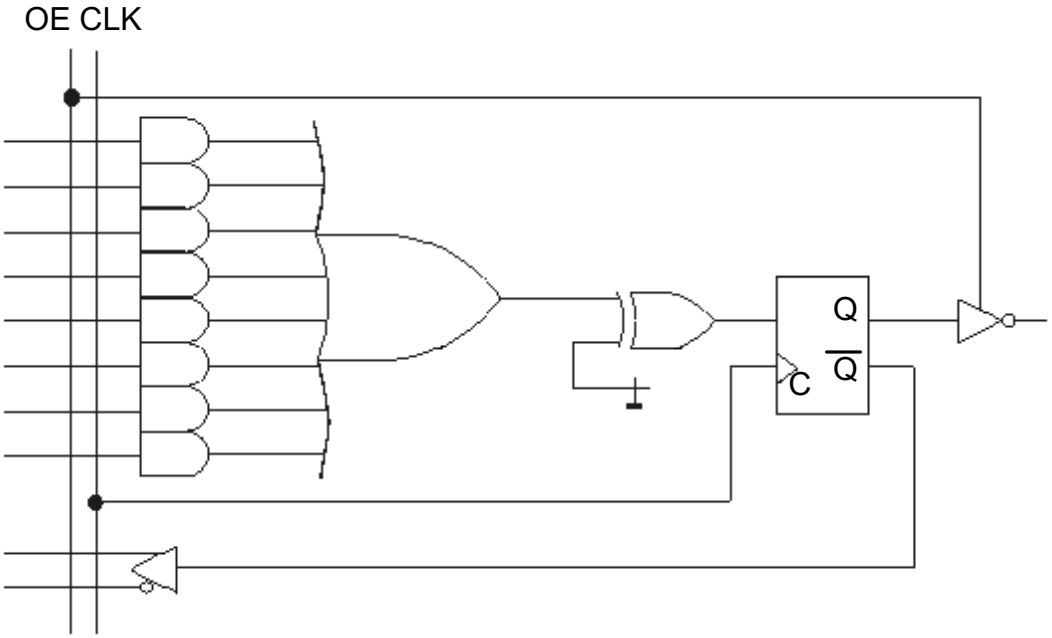


Рисунок 5.9 – Логічна частина схеми GAL16V8R

Зауважмо, що цього разу всередину схеми зворотно подається сигнал із запереченого виходу тригера, а також, що вихід сегмента не може виконувати функцію входу. Тому в регістровій конфігурації схема GAL16V8R має 8 доступних зовнішніх входів (плюс 8 входів з виходів тригерів) і 8 зовнішніх виходів (плюс входи CLK і OE). На рисунку 5.10 показано графічне зображення схеми GAL16V8R.

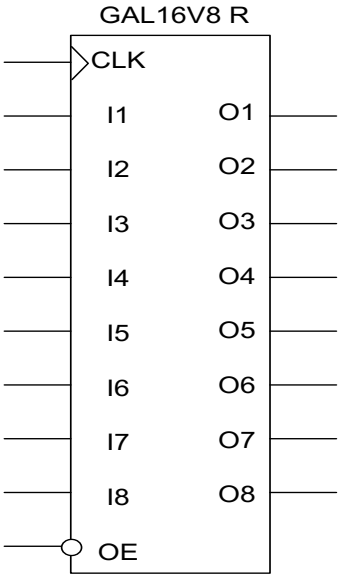


Рисунок 5.10 – Графічне зображення схеми GAL16V8R

Поряд із схемами GAL16V8 виготовляють схеми з більшою кількістю входів і виходів (GAL20V8, GAL22V10). Подібні схеми випускаються різними виробниками.

5.3 Програмовані схеми CPLD і FPGA

Розглянуті вище схеми типу GAL дозволяють реалізовувати досить складні логічні проекти. Однак для багатьох практичних застосувань ця складність є недостатньою і через це розроблено багато інших схем із значно збільшеною складністю. (Складність програмованої схеми часто оцінюється як кількість елементів NAND, котрі потрібно використати, щоб реалізувати функції за допомогою розглянутої програмованої схеми). Історично розрізняють дві концепції будови складних програмованих схем: схеми CPLD, а також схеми FPGA.

Концепція будови схем CPLD полягає на тому, щоб в одній програмованій схемі інтегрувати певну кількість простих програмованих схем і забезпечити відповідну структуру шляхів, що уможливить програмування сполучень. В концепції схем FPGA припускають, що в інтегральній структурі програмованої схеми буде міститися певна кількість так званих макрокомірок або логічних блоків (кожна макрокомірка чи логічний блок – це набір вибраних логічних схем), а також відповідна структура шляхів для сполучень. Зараз різниці між

схемами, реалізованими за допомогою цих концепцій, стають більш заплутаними і найчастіше для складних програмованих схем використовуються назви FPGA.

Складні програмовані схеми, як правило, містять три складових елементи (рисунок 5.11): набір макрокомірок, мережа сполучень, а також схеми входу/виходу. Макрокомірки становлять матрицю блоків, що займає середню частину сполучень. Мережа програмованих сполучень займає вільні місця між макрокомірками. Натомість схеми входу/виходу розміщені на краях структури і забезпечують співпрацю програмованої схеми з середовищем.

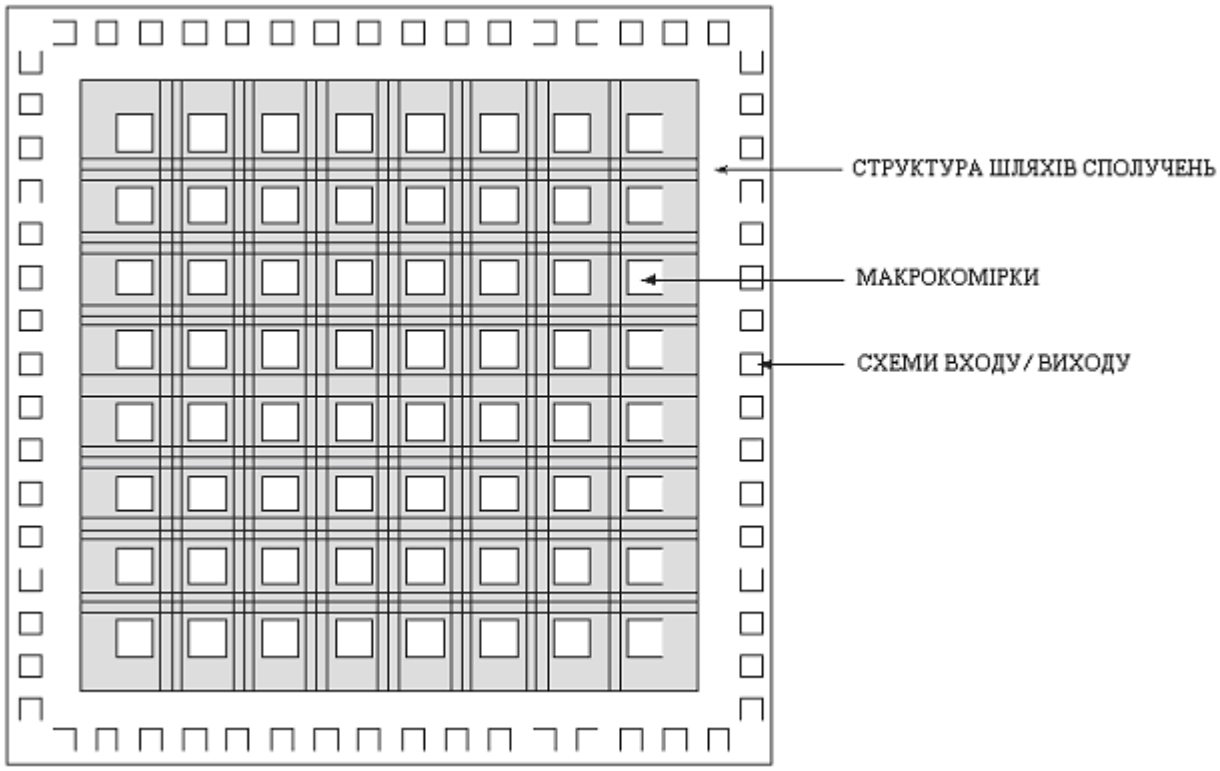


Рисунок 5.11 – Загальна архітектура схем FPGA

Кожна комірка входу/виходу, як правило, уможливлює співпрацю з різними інтегральними схемами (TTL, CMOS) з різними значеннями напруги, з тристабільними схемами або із схемами з відкритим колектором та ін. Комірки входу/виходу програмуються так само, як макрокомірки.

Макрокомірки містять різні набори логічних схем, що уможливлюють реалізацію складних логічних функцій. Відповідні макрокомірки можуть самостійно реалізовувати певні функції або можуть співпрацювати з іншими комірками, що знаходяться в структурі програмованої схеми.

Мережа сполучень забезпечує можливість програмування сполучень як всередині відповідних макрокомірок, так і між ними, а також сполучень з комірками входу/виходу. Структури мереж сполучень мають різні варіанти рішень – окремі виробники використовують свої рішення, стараючись забезпечити еластичність проведення сполучень, а також швидкість

передачі сигналів. Як правило, виділяють спеціальні шляхи, що служать для поширення часових сигналів.

5.4 Проектування з програмованими схемами

Програмовані схеми уможливлюють реалізацію складних, навіть дуже складних, проектів. На практиці процес проектування підтриманий спеціальним програмуванням, забезпеченим зазвичай фірмами, що виготовляють програмовані схеми.

В процесі проектування потрібно реалізувати декілька таких завдань. Весь процес проектування схематично показано на рисунку 5.12.

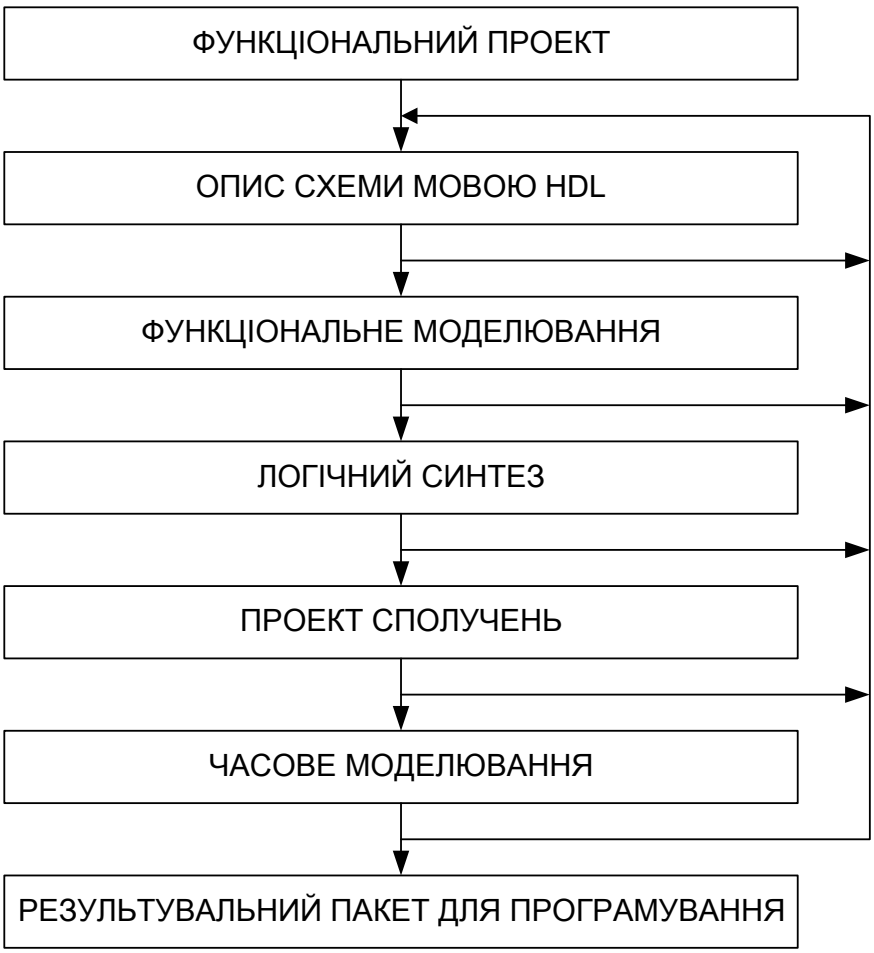


Рисунок 5.12 – Приклад процесу проектування програмованих схем

На початку процесу проектування потрібно мати функціональний проект. Мають бути визначені всі вимоги, що висуваються до проектованої схеми (пристрою).

Після формулювання функціональних припущень можливо описати проект вибраною спеціальною мовою опису пристроїв HDL. На практиці проектування найчастіше використовують мови ABEL, VHDL і VERILOG. Ці мови уможливлюють опис функцій, що має здійснювати проектований

пристрій. Мова ABEL (її фірмові версії, на приклад, AHDL) є відносно проста і її використовують у випадку простих програмованих схем. У випадку великих проектів і користування схемами FPGA використовуються мови VHDL і VERILOG. Можливості обох цих мов близькі між собою.

Після опису проекту однією з мов HDL можливе функціональне моделювання з метою перевірки правильності опису і функціонування проектованої схеми.

Почергово виконується етап логічного синтезування проекту. В результаті з'являється логічна схема проектованої схеми.

Тепер можливий перехід до відображення логічної схеми на структуру використовуваної програмованої схеми. Найперше виконується декомпозиція схеми на такі фрагменти, які можуть бути реалізовані доступним сегментом програмованої схеми (в простих схемах) або макрокомірки (в складних схемах). Далі визначається структура сполучень, яку потрібно буде потім фізично реалізувати – запрограмувати відповідні сполучення.

Почергово можна виконати повне моделювання проекту з врахуванням часових залежностей.

В процесі виконання відповідних етапів проекту може виявитися, що даний етап не можна реалізувати або що були виявлені суперечності з функціональними припущеннями (етапи моделювання). Тоді потрібно повернутися до одного з попередніх етапів і ввести відповідні зміни.

Якщо процес проектування буде успішно доведений до кінця, як результат отримуємо кінцевий пакет з інформацією, необхідною для програмування програмованої схеми або з використанням програматора, або безпосередньо в системі.

Приклад

Побудувати схему зв'язків ПЛІС за такою системою булевих функцій (табл. 5.1):

Таблиця 5.1 – Система булевих функцій

a1	*a2*	*a3*	*a4*	*a5*	*Z*	*C1*	*C2*	*C3*	*C4*
0	1	X	1	1	*Z1*	1	0	1	0
1	X	1	0	X	*Z2*	1	1	0	0
1	0	0	1	0	*Z3*	0	0	0	0
0	0	1	X	X	*Z4*	0	1	1	0
1	0	1	0	1	*Z5*	0	0	0	1
X	0	0	1	0	*Z6*	1	0	1	1
0	1	1	0	0	*Z7*	0	1	0	1
0	1	1	0	1	*Z8*	0	1	0	0

Перша колонка містить аргументи, друга колонка містить терми, третя колонка містить функції.

Аргументи *a1, a2, a3, a4, a5* можуть містити як значення одиниць, так і значення нулів в тих чи інших розрядах, які не впливають на вихідну функцію; вони позначаються «X».

Сформуємо систему рівнянь для термів *Z* таким чином:

$$\begin{cases} Z_1 = \overline{a_1} a_2 a_4 a_5; \\ Z_2 = a_1 a_3 \overline{a_4}; \\ Z_3 = a_1 \overline{a_2} \overline{a_3} a_4 \overline{a_5}; \\ Z_4 = \overline{a_1} \overline{a_2} a_3; \\ Z_5 = a_1 \overline{a_2} a_3 \overline{a_4} a_5; \\ Z_6 = \overline{a_2} \overline{a_3} a_4 a_5; \\ Z_7 = \overline{a_1} a_2 a_3 \overline{a_4} \overline{a_5}; \\ Z_8 = \overline{a_1} a_2 a_3 \overline{a_4} a_5. \end{cases} \quad (5.1)$$

Далі сформуємо систему рівнянь для функцій *C* таким чином:

$$\begin{cases} C_1 = Z_1 \vee Z_2 \vee Z_6; \\ C_2 = Z_2 \vee Z_4 \vee Z_7 \vee Z_8; \\ C_3 = Z_1 \vee Z_4 \vee Z_6; \\ C_4 = Z_5 \vee Z_6 \vee Z_7. \end{cases} \quad (5.2)$$

Для побудови матриці зв'язків необхідно мати три частини:

1. Буфер, який має парафазні значення аргументів *a1, a2,… an*;
2. Матрицю М1, яка називається «*і-матрицею*», на виході якої формуються значення термів *Z1, Z2, … Zm* (рис. 5.13);
3. Матриця М2, на виході якої формуються функції *C1, C2 … Ck* (рис. 5.14).

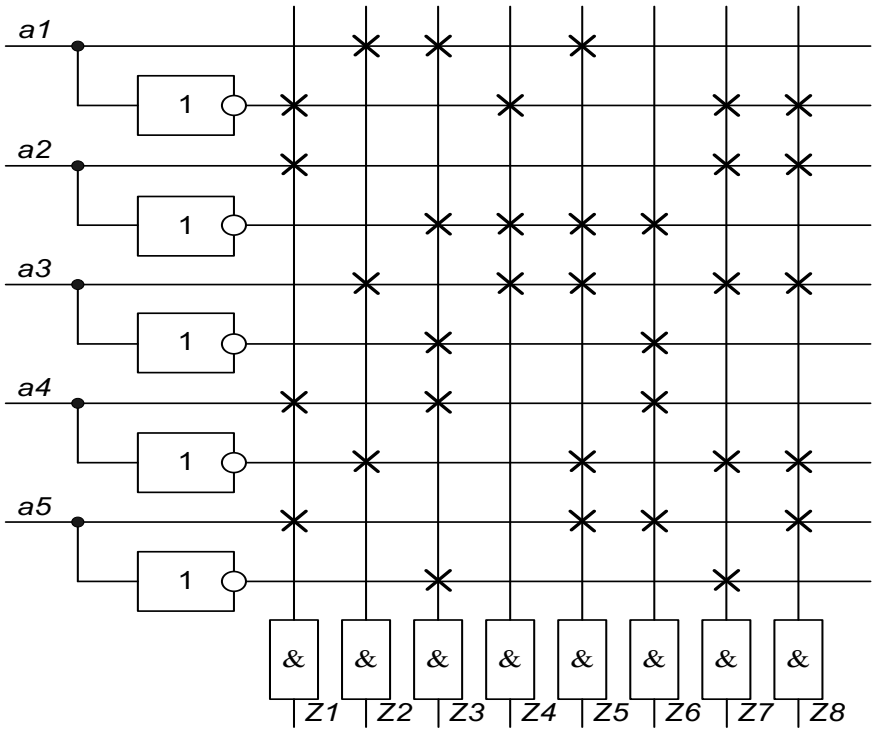


Рисунок 5.13 – Матриця зв’язків М1

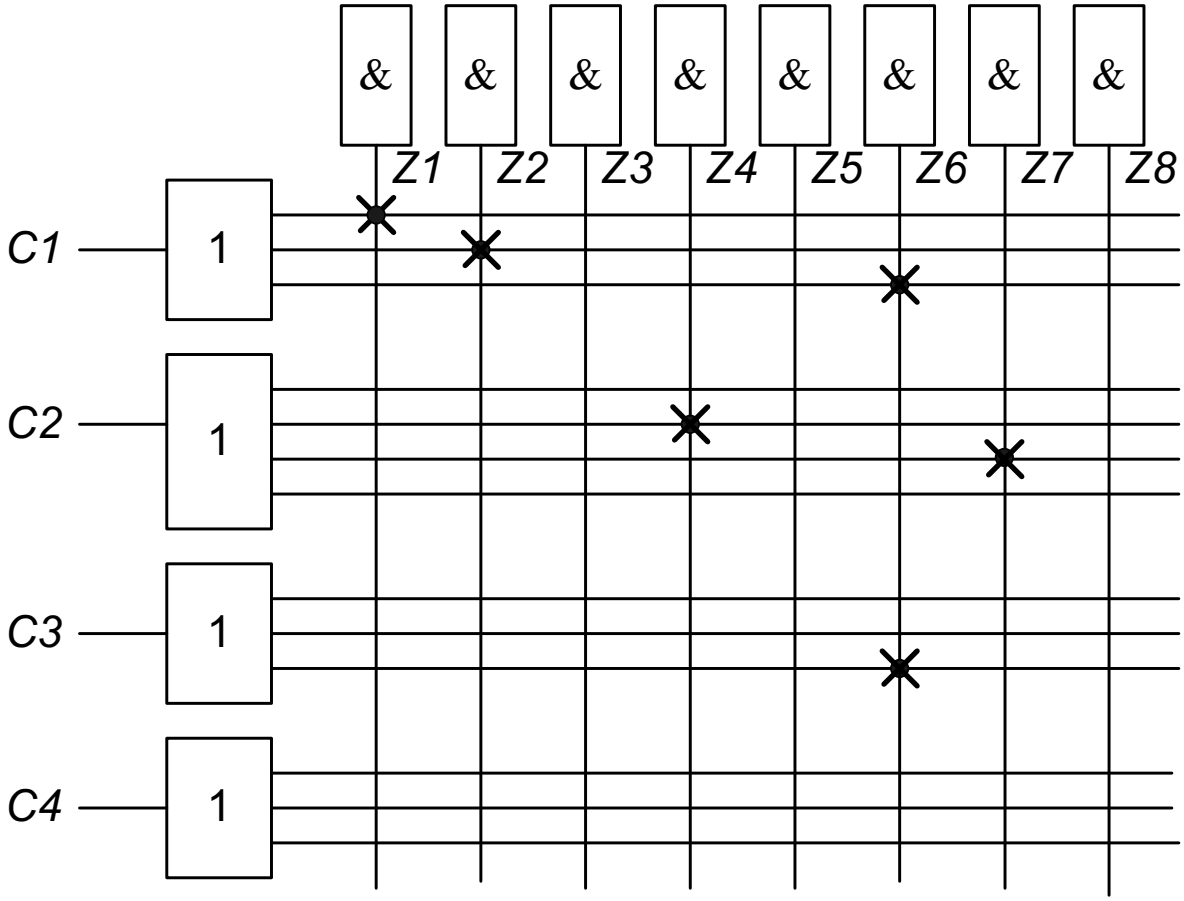


Рисунок 5.14 – Матриця зв’язків М2

Контрольні запитання

1. З чого складаються програмовані логічні інтегральні схеми?

2. Що являє собою конфігурування схеми?

3. Якими способами можна програмувати сполучення в інтегральній схемі?

4. В якому випадку доцільно використовувати програмування схем з використанням мультиплексорів?

5. Набори яких основних елементів містять прості програмовані схеми?

6. В чому полягає концепція будови схем CPL?

7. В чому полягає концепція будови схем FPGA?

8. З яких елементів складаються складні програмовані логічні схеми?

9. Наведіть послідовність процедур процесу проектування логічних схем.

6 ПРИКЛАДИ ПРОЕКТУВАННЯ СПЕЦІАЛІЗОВАНИХ АРИФМЕТИКО-ЛОГІЧНИХ ПРИСТРОЇВ

6.1 Проектування схеми порівняння слова з константою

Припустимо, що потрібно отримати ознаки відношень двійкового слова $A = A_2A_1A_0$ з такими заданими константами:

$F_1 := (A = 000)$; $F_2 := (A = 111)$ і $F_3 := (A \le 011)$.

Сформуємо таблицю відповідності кодів $A_2\,A_1\,A_0$ і $F_1\,F_2\,F_3$

Таблиця 6.1 – Таблиця відповідності кодів $A_2\,A_1\,A_0$ і $F_1\,F_2\,F_3$

A_2	A_1	A_0	F_1	F_2	F_3
0	0	0	1	0	1
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	0	0	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	0	1	0

На підставі табл. 6.1 значення ознак відношення слова А з константами запишемо у вигляді:

$$F_1 = A_2A_1A_0; \quad F_2 = A_2A_1A_0; \quad F_3 = \overline{A_2}. \tag{6.1}$$

Схему порівняння слова з константою згідно з виразами (6.1) показано на рис. 6.1.

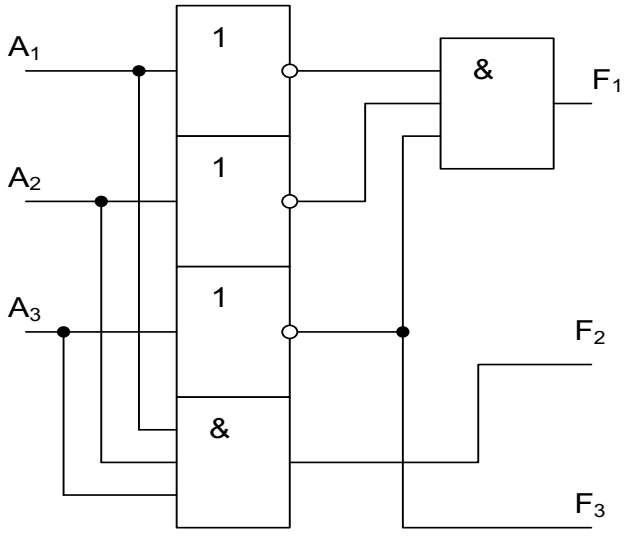


Рисунок 6.1 – Схема порівняння слова з константою

6.2 Проектування схеми порівняння двійкових слів А і В

Багаторозрядні двійкові слова рівні між собою, коли одночасно попарно дорівнюють один одному всі їхні розряди, тобто *A(n) = B(n)*, якщо $A_i = B_i$, $i = 1, 2, ..., n$.

Таблиця 6.2 – Логіка порівняння розрядів А і В

A_i	B_i	r_i
0	0	1
0	1	0
1	0	0
1	1	1

На підставі даних табл. 6.2, яка задає умову рівності r_i двох і-тих розрядів А і В, отримаємо

$$r_i = \overline{A_i}\,\overline{B_i} \vee A_i B_i = \overline{A_i \oplus B_i} = \overline{M_i}, \qquad (6.2)$$

де M_i – функція додавання за модулем два (ВИКЛЮЧАЛЬНЕ АБО).

Ознака рівності двох n-розрядних слів $P_{A=B}$ визначається логічним добутком порозрядних умов r_i:

$$F_{A=B} = r_n\, r_{n-1} ... r_1 = \overline{M_n}\,\overline{M_{n-1}} ... \overline{M_1}\,. \qquad (6.3)$$

Схему порівняння двох чотирирозрядних слів *A* і *B* згідно з виразом (6.3) показано на рис. 6.2. Схема вміщує чотири логічних елементи ВИКЛЮЧАЛЬНЕ АБО і один кон'юнктор.

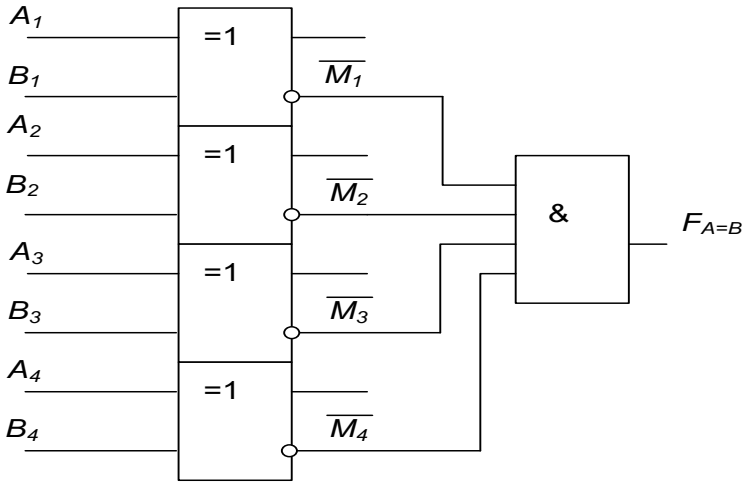


Рисунок 6.2 – Схема порівняння двох чотирирозрядних слів А і В

6.3 Проектування схеми порівняння двох слів «НА БІЛЬШЕ»

Схема порівняння двох слів A і B «на більше» за абсолютним значенням виробляє ознаку $F_{A>B}$ і будується за таким алгоритмом:

- аналіз нерівності слів А і В виконується послідовно в напрямку від старших розрядів до молодших;
- молодші розряди включаються в аналіз в тому випадку, коли старші розряди однакові (еквівалентні);
- для отримання ознаки $P_{A>B}$ будується диз'юнктивна сума порозрядних умов.

Логіку порівняння розрядів A і B наведено в табл. 6.3, де C_i – ознака $A_i > B_i$; r_i – умова підключення до аналізу сусідніх молодших розрядів обох слів.

Таблиця 6.3 – Логіка порівняння розрядів А і В

A_i	B_i	C_i	r_i
0	0	0	1
0	1	0	0
1	0	1	0
1	1	0	1

На підставі даних табл. 6.3 порівняння розрядів А і В отримуємо такі вирази:

$$C_i = A_i \overline{B_i}; \; r_i = \overline{A_i}\,\overline{B_i} \vee A_i B_i = \overline{A_i \oplus B_i} = \overline{M_i}. \qquad (6.4)$$

З урахуванням виразу (6.4) і алгоритму аналізування функцію ознаки $F_{A>B}$ записуємо у вигляді:

$$F_{A>B} = C_n \vee r_n C_{n-1} \vee \ldots \vee r_n \, r_{n-1} \ldots r_2 C_1. \qquad (6.5)$$

Для порівняння двох чотирирозрядних слів «на більше» ознаку нерівності згідно з виразом (6.5) подаємо так:

$$\begin{aligned} F_{A>B} &= C_4 \vee r_4 C_3 \vee r_4 \, r_3 C_2 \vee r_4 r_3 r_2 C_1 = \\ &= A_4 \overline{B_4} \vee \overline{M_4} \, A_3 \overline{B_3} \vee \overline{M_4}\,\overline{M_3} \, A_2 \overline{B_2} \vee \overline{M_4}\,\overline{M_3}\,\overline{M_2} \, A_1 \overline{B_1}. \end{aligned} \qquad (6.6)$$

Схему порівняння «на більше» двох чотирирозрядних слів А і В згідно із співвідношенням (6.6) показано на рис. 6.3.

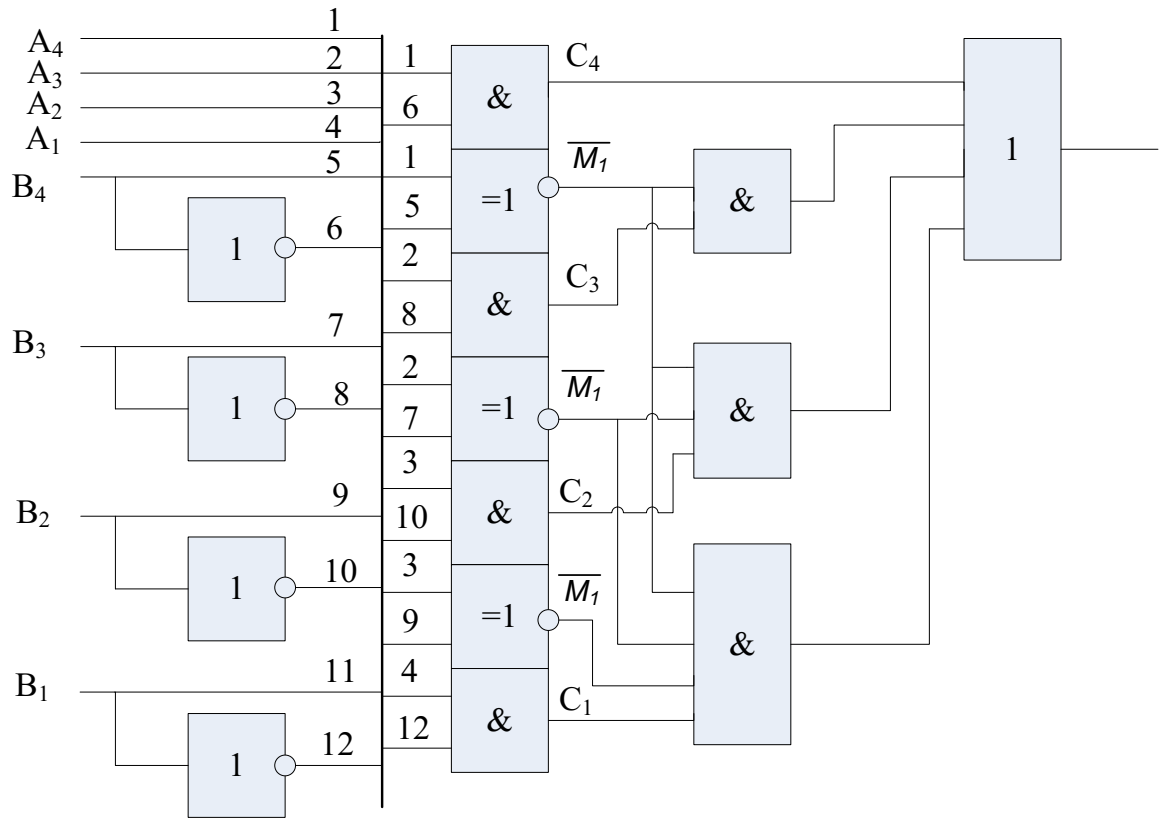


Рисунок 6.3 – Схема порівняння двох слів «на більше»

6.4 Проектування схем контролю за парністю

У разі контролю за парністю значення контрольного (паритетного) біта визначається додаванням за модулем два значень розрядів байта:

$$F_{к.п.} = A_1 \oplus A_2 \oplus A_3 \oplus A_4 \oplus A_5 \oplus A_6 \oplus A_7 \oplus A_8 . \qquad (6.7)$$

У разі контролю за непарністю значення контрольного біта набуває такого виразу:

$$F_{к.п.} = \overline{A_1 \oplus A_2 \oplus A_3 \oplus A_4 \oplus A_5 \oplus A_6 \oplus A_7 \oplus A_8} . \qquad (6.8)$$

Умову парності отримуємо додаванням за модулем два восьмирозрядного слова, що реалізовується за допомогою ступінчастого включення двоходових елементів ВИКЛЮЧАЛЬНЕ АБО (рис. 6.4):

- на першому рівні отримують функції $F_1 – F_4$:

$$F_{к.п.} = \overline{A_1 \oplus A_2 \oplus A_3 \oplus A_4 \oplus A_5 \oplus A_6 \oplus A_7 \oplus A_8} ; \qquad (6.9)$$

- на другому і третьому рівнях реалізовуються функції:

$$F_5 = F_1 \oplus F_2, \;\; F_6 = F_3 \oplus F_4, \;\; M = F_5 \oplus F_6. \qquad (6.10)$$

Для задання ознаки контролю вводиться керувальний сигнал V, який разом із сигналом M надходить на входи схеми ВИКЛЮЧАЛЬНЕ АБО в четвертому рівні; на прямому та інверсному виходах цього рівня формуються пряме та інверсне значення контрольного розряду:

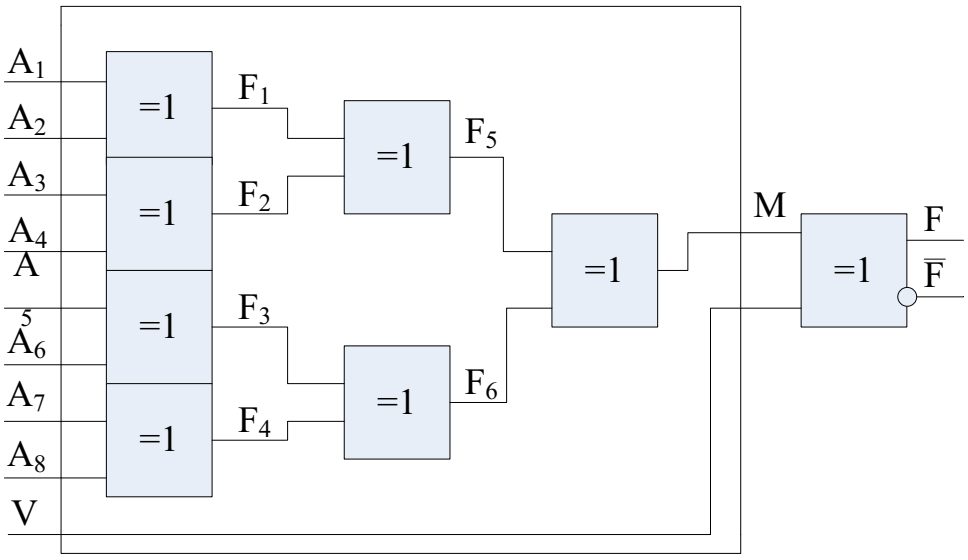


Рисунок 6.4 – Схема контролю за парністю

6.5 Проектування схеми перетворювача прямого коду на обернений

У прямому двійковому коді $X_{пр} = X_{зн}\, X_{n-1}, \ldots, X_1$ один розряд, зазвичай старший, відображає знак числа, інші – значення цифрових розрядів; при цьому для додатного числа $X_{зн} = 0$, а для від'ємного $X_{зн} = 1$.

Обернений код додатного двійкового числа збігається з прямим кодом, а для від'ємного числа цифрові розряди прямого коду інвертуються.

У процесі перетворення прямого коду в обернений значення знакового розряду $X_{зн}$ використовується як керувальний сигнал, що забезпечує отримання такого виразу:

$$Y_i = \overline{X_{зн}}\, X_i \vee X_{зн}\, \overline{X_i} = X_{зн} \oplus X_i \qquad (6.11)$$

де Y_i – значення i-го розряду оберненого коду;

X_i – значення i-го розряду додатного вхідного числа ($X_{зн} = 0$);

$\overline{X_i}$ – значення i-го розряду від'ємного вхідного числа ($\overline{X_{зн}} = 1$).

Схему п’ятирозрядного перетворювача прямого коду на обернений, побудовану на елементах ВИКЛЮЧАЛЬНЕ АБО відповідно до виразу (6.11), показано на рис. 6.5.

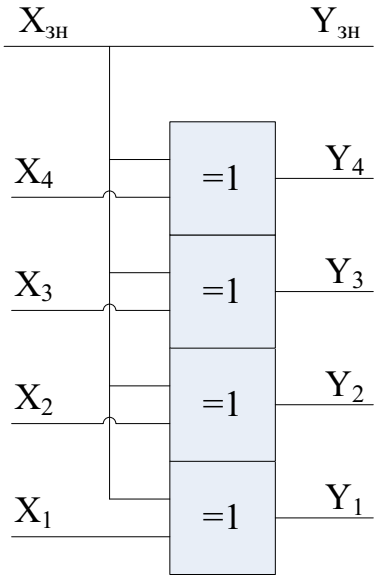


Рисунок 6.5 – Схема перетворювача прямого коду на обернений

6.6 Проектування схеми перетворювача прямого коду на доповняльний

Доповняльний код додатного двійкового числа збігається з його прямим і оберненим кодами. Доповняльний код від’ємного двійкового числа утворюється з його оберненого коду додаванням до молодшого розряду одиниці.

Знаковий розряд прямого коду використовується як керувальний сигнал: якщо $X_{зн} = 0$, то вихідний код повторює значення вхідного; якщо $X_{зн} = 1$ реалізовується перетворення згідно з табл. 6.4.

Таблиця 6.4 – Відповідність між кодами беззнакових розрядів

Прямий код				Доповняльний код				Прямий код				Доповняльний код			
X_4	X_3	X_2	X_1	Y_4	Y_3	Y_2	Y_1	X_4	X_3	X_2	X_1	Y_4	Y_3	Y_2	Y_1
0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0
0	0	0	1	1	1	1	1	1	0	0	1	0	1	1	1
0	0	1	0	1	1	1	0	1	0	1	0	0	1	1	0
0	0	1	1	1	1	0	1	1	0	1	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	0	1	0	0
0	1	0	1	1	0	1	1	1	1	0	1	0	0	1	1
0	1	1	0	1	0	1	0	1	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1	1	1	1	1	0	0	0	1

Карту Карно відповідно до табл. 6.4 для отримання мінімальних форм функцій перетворення прямого коду в доповняльний показано на рис. 6.6.

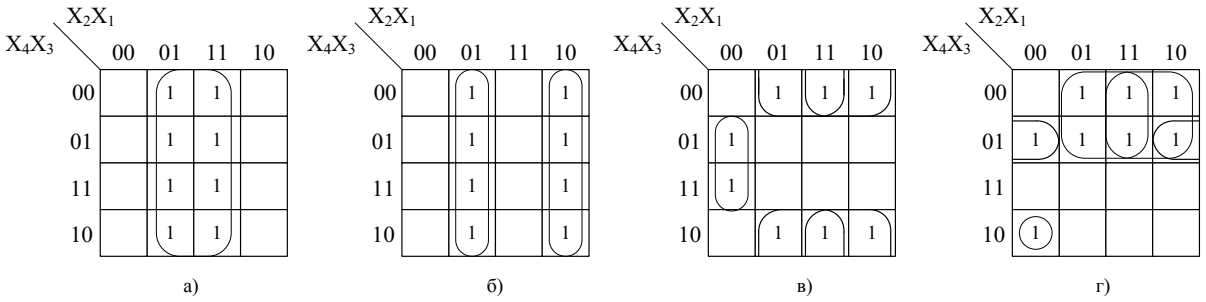


Рисунок 6.6 – Карта Карно для функцій перетворювача прямого коду в доповняльний: а) – Y_1; б) – Y_2; в) – Y_3; г) – Y_4

На основі карт Карно з урахуванням знакового розряду $X_{зн}$ прямого коду для функцій Y_1, Y_2, Y_3, Y_4, що характеризують виходи перетворювача, отримуємо:

$$Y_{зн} = X_{зн};\ Y_1 = X_1;\ Y_2 = X_2 \oplus X_1 X_{зн};$$
$$Y_3 = X_3 \oplus (X_2 \vee X_1) X_{зн};\ Y_4 = X_4 \oplus (X_3 \vee X_2 \vee X_1) X_{зн}. \qquad (6.12)$$

У загальному вигляді для Y_i справедливим є рівняння:

$$Y_i = X_i \oplus \left(X_{i-1} \vee X_{i-2} \vee \ldots \vee X_1\right) X_{зн}. \qquad (6.13)$$

Схеми перетворювачів прямого коду в доповняльний на основі виразів (6.12) і (6.13) показано на рис. 6.7.

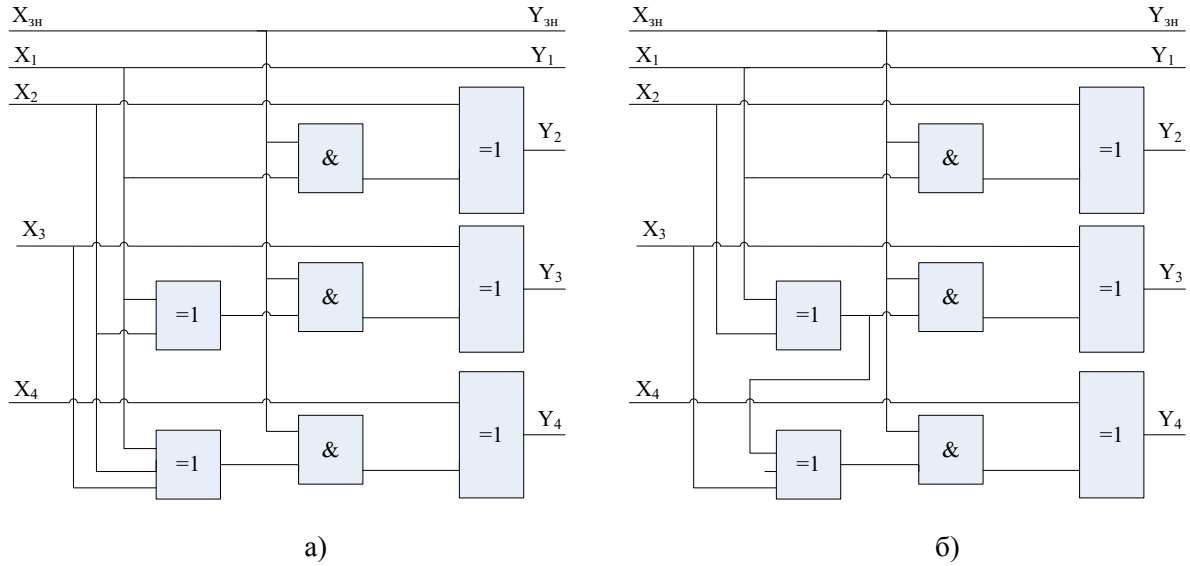


Рисунок 6.7 – Схеми перетворювачів прямого коду на доповняльний

6.7 Проектування спеціалізованого АЛП для операції додавання

6.7.1 Початкові дані до проекту:

- тип арифметичної операції – додавання двійкових чисел;
- початковий код подання операндів – доповняльний;
- розрядність операндів – 8 біт;
- код виконання операції у суматорі – доповняльний модифікований;
- структура операційного блоку – із закріпленими мікрооперaціями;
- тип керувального блока – автомат Мура з пам'яттю на JK-тригерах;
- схема логічної ознаки переповнення розрядної сітки;
- схема логічного порозрядного додавання кодів вхідних операндів *A* і *B*;
- елементна база – інтегральні схеми ТТЛШ серій К1531, КР1533.

6.7.2 Алгоритм додавання двійкових чисел

Додавання і віднімання двійкових чисел можна виконувати в обернених або доповняльних кодах та їх модифікаціях. У сучасних комп'ютерах часто операнди зберігаються в пам'яті в доповняльних кодах. Використання доповняльних кодів в операціях додавання і віднімання та для зберігання операндів у пам'яті має такі переваги:

- однозначне подання знака результату як додатного, так і від'ємного;
- під час записування в пам'ять від'ємного результату не витрачається час для його перетворення на прямий код;
- менше дій для аналізування знака результату, зокрема переповнення розрядної сітки.

Алгоритм додавання двійкових чисел:

1) у першому і другому машинних тактах із вхідної шини паралельним кодом записуються операнди A і B у відповідні регістри RGA і RGB. Зчитування операндів здійснюється ЦПК;

2) протягом одного машинного такту виконується мікрооперація додавання;

3) за відсутності переповнення розрядної сітки результат записується у регістр RGC;

4) за наявності переповнення результат не фіксується і в ЦПК подається сигнал переповнення ПП.

6.7.3 Функціональна схема АЛП для виконання операції додавання

Функціональну схему восьмирозрядного АЛП1 для виконання операції додавання показано на рис. 6.8. Схема АЛП1 містить:

- регістри RGA і RGB для приймання і подальшого зберігання із вхідної шини Ш1 першого і другого операндів;
- паралельний комбінаційний суматор з додатковим старшим розрядом знака П для створення модифікованого доповняльного коду;

- регістр результату RGC, дані з якого пересилаються по вихідній шині Ш2 в оперативну пам’ять;

- схеми електронних ключів SW1 і SW2;

- схему вироблення ознак переповнення ОР;

- схему диз’юнкторів OR для виконання операцій порозрядного логічного додавання кодів операндів А і В.

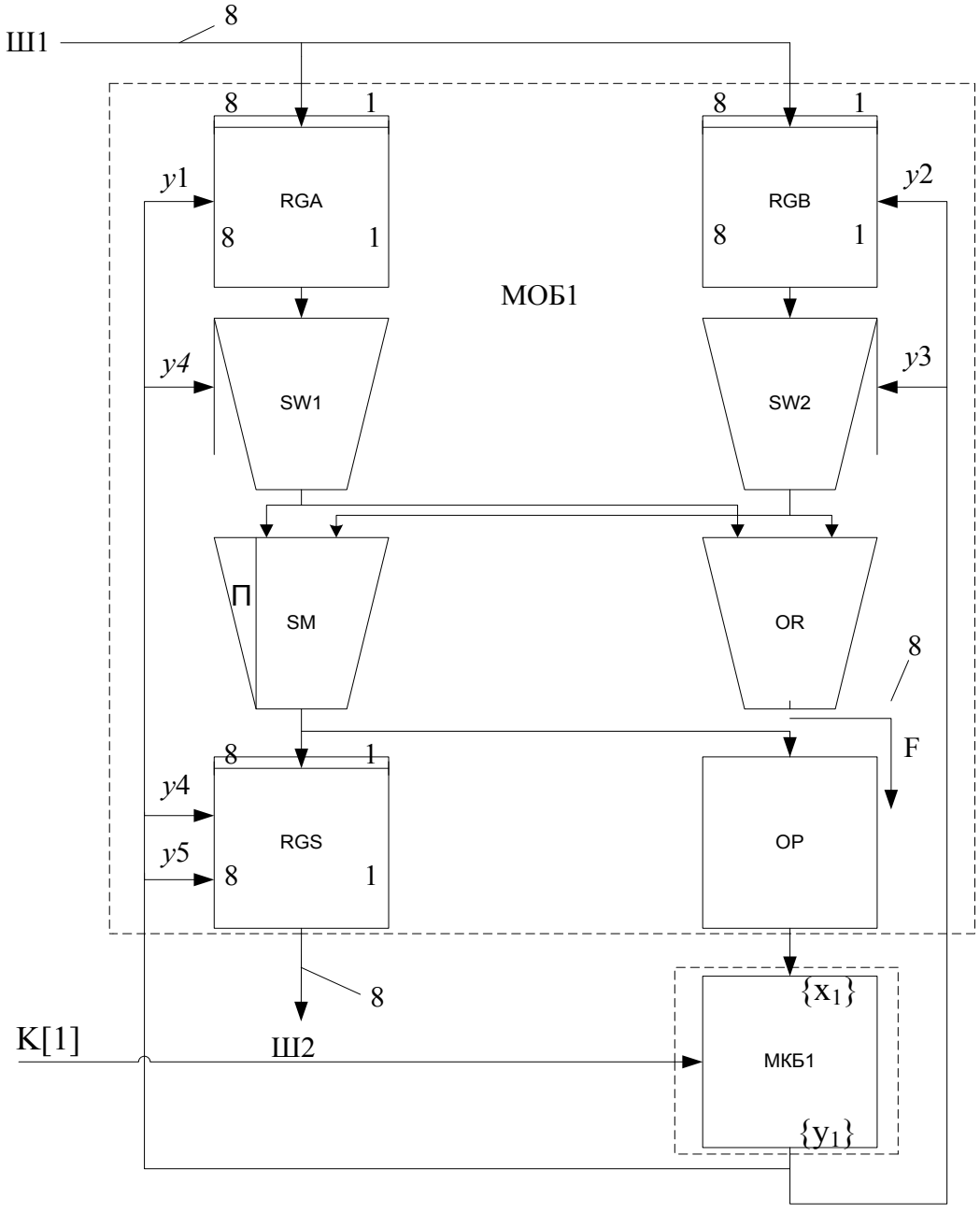


Рисунок 6.8 – Функціональна схема АЛП1 для додавання чисел:
МОБ1 – модуль операційного блока;
SW1, SW2 – схеми електронних ключів;
OR – схема порозрядного логічного додавання;
ОР – схема ознаки результату;
МКБ1 – модуль керувального блока

Ознаки результату обчислюються за допомогою булевих виразів:

$\varphi_1 = \overline{П} \cdot \overline{SM}[n]$ – додатний результат (знаки 00);

$\varphi_2 = П \cdot SM[n]$ – від'ємний результат (знаки 11);

$\varphi_3 = \overline{П} \cdot SM[n] \vee П \cdot \overline{SM}[n]$ переповнення розрядної сітки ПП (знаки дорівнюють 01 чи 10);

$\Lambda_{i=1}^{n+1} \overline{SM}[i]$ – нульовий результат.

Після закінчення операції КА аналізує ознаки результату і встановлює значення відповідних тригерів ознак. Ознака переповнення перевіряється до закінчення операції і за її наявності виконання програми переривається.

Ознака OR реалізовується за допомогою восьми логічних двовходових елементів АБО за співвідношенням

$$F_i = (A_i \vee B_i),\ i = 1, 2, ..., 8, \tag{6.14}$$

де F_i – i-й вихід вузла логічного додавання. Ця операція виконується автоматично незалежно від коду команди.

6.7.4 Мікропрограма додавання

Мікропрограма додавання двійкових чисел у доповняльних кодах має такий вигляд:

- Початок. Якщо К[1], то М1, інакше – чекати
- М1 у1 :RGA := A <приймання першого операнда>
- у2 :RGB : = B <приймання другого операнда>
- у3 :SM := A+B <додавання>
- Якщо φ_3, то М2, інакше
- у4 :RGC := SM <присвоєння результату>
- у5 :Ш2 : = RGC <пересилання в пам'ять>
- Перейти до М3
- М2 у6 :$Т_П$: ПП <тригеру переповнення $Т_П$ присвоюється ознака ПП>
- М3 Кінець.

Примітка. К[1] – однорозрядний код команди додавання.

Змістовну і закодовану блок-схеми алгоритмів мікропрограми додавання показано на рис. 6.9.

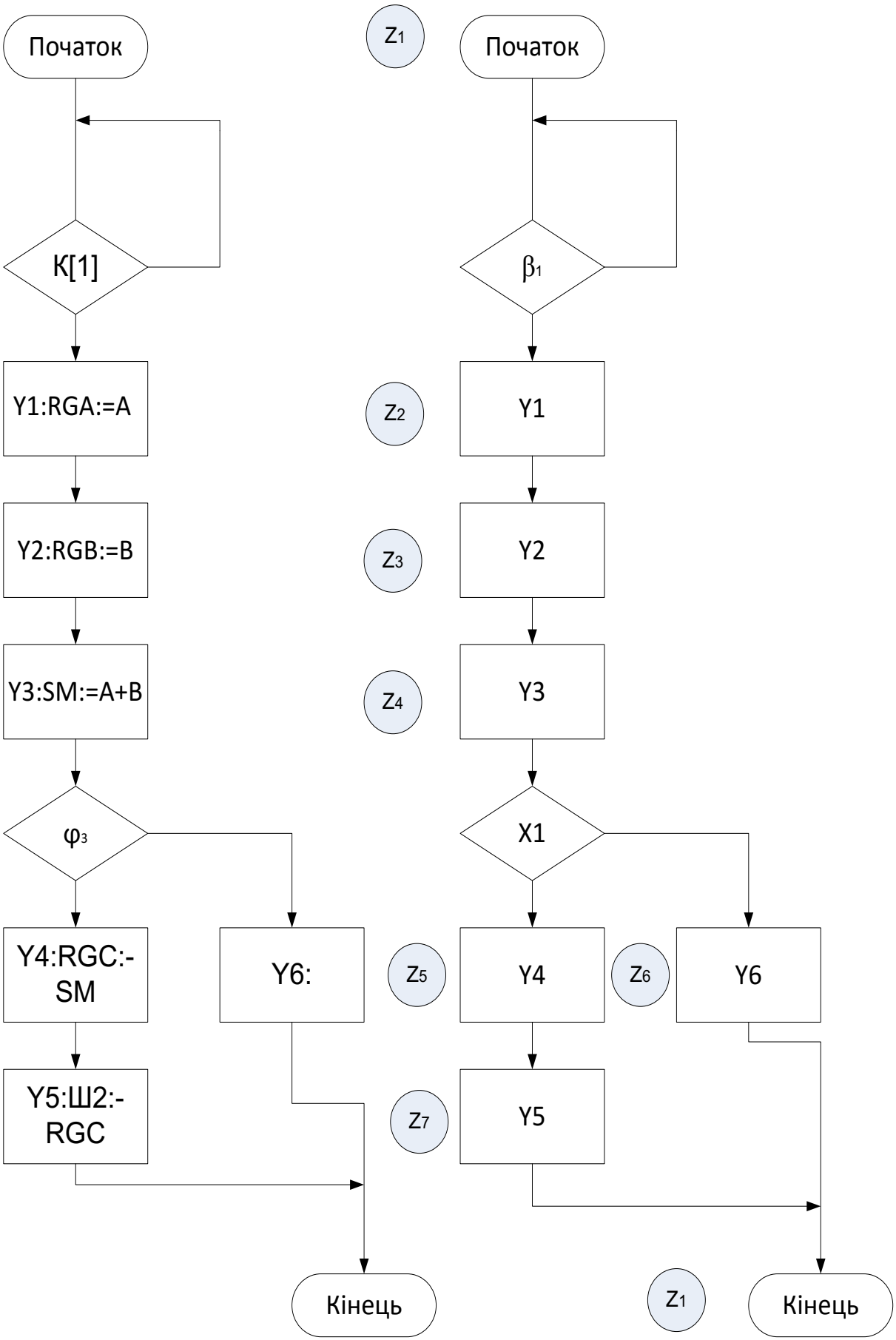


Рисунок 6.9 – Блок-схеми алгоритмів мікропрограми додавання

6.7.5 Проектування модуля операційного блока МОБ1

Модуль МОБ1 будується на мікросхемах ТТЛШ серії КР1533.

Для побудови принципової схеми модуля МОБ1 використано такі мікросхеми:

- два вхідних восьмирозрядних регістри RGA і RGB типу ИР35;
- дві мікросхеми типу ЛЛ1, кожна з яких містить по чотири двовходових логічних елементи АБО. Їх використовують для реалізації порозрядної диз'юнкції над кодами операндів A і B;
- чотири мікросхеми типу ЛИ1, кожна з яких містить по чотири двовходових кон'юнктори. Використовують для підключення виходів регістрів RGA і RGB до входів суматора SM;
- мікросхема типу ЛП5, яка містить чотири логічних елементи ВИКЛЮЧАЛЬНЕ АБО. Використовують для створення старшого знакового розряду суматора і логічної ознаки $\varphi_3 = x_1$ та $\overline{x_1}$;
- вихідний восьмирозрядний регістр RGC типу ИР22 з трьома станами. Використовують для приймання результату додавання і передавання його на вихідну шину;
- два чотирирозрядних комбінаційних суматори SM типу ИМ6.

6.7.6 Проектування модуля керувального блока МКБ1

Проектування модуля МКБ1 на основі автомата Мура з пам'яттю на JK-тригерах виконується в такій послідовності:

1. Розмічається закодований граф мікропрограми додавання. Визначається максимальна кількість станів автомата Мура, що дорівнює L=7. Для реалізації такого числа станів необхідно використати n =]log7[= 3 тригери.

2. На основі розміченого графа мікропрограми будується граф автомата Мура (рис. 6.10), який інтерпретує мікропрограму додавання.

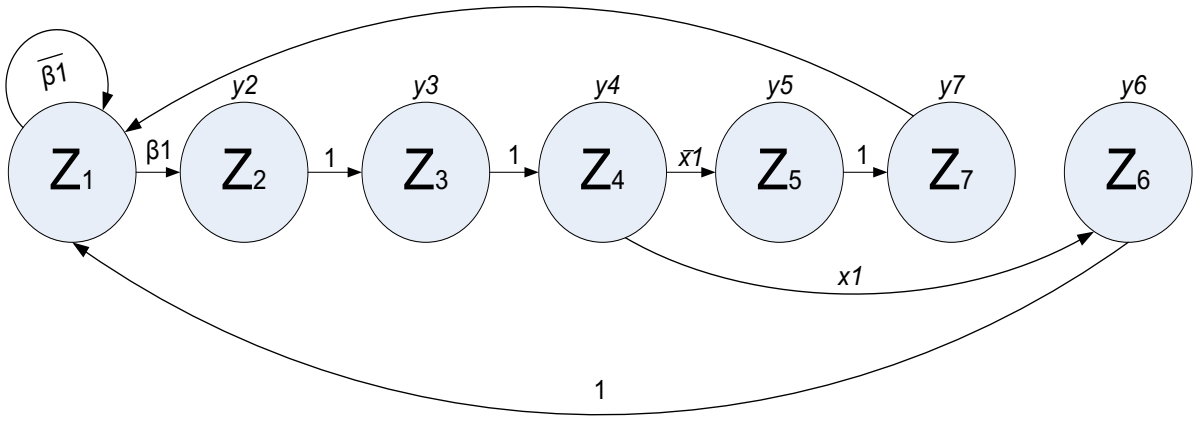


Рисунок 6.10 – Граф автомата Мура для мікропрограми додавання

3. Стани автомата Мура кодуються значеннями виходів JK-тригерів:

$$z_1 = \overline{Q_3}\,\overline{Q_2}\,\overline{Q_1},\ z_2 = \overline{Q_3}\,\overline{Q_2}Q_1,\ z_3 = \overline{Q_3}Q_2\overline{Q_1},\ ...,\ z_4 = Q_3Q_2\overline{Q_1}. \qquad (6.15)$$

4. На основі графа автомата Мура записується його структурна таблиця переходів (табл. 6.5).

Таблиця 6.5 – Структурна таблиця переходів автомата Мура

z_i	$k(z_i)$	z_j	$k(z_j)$	$\{x_i\}$	$\{y_i\}$	JK	
						K	J
z_1	000	z_1 z_2	000 001	$\overline{\beta_1}$ β_1	- y_1	- -	- J_1
z_2	001	z_3	010	1	y_2	K_1	J_2
z_3	010	z_4	011	1	y_3	-	J_1
z_4	011	z_5 z_6	100 101	$\overline{x_1}$ x_1	y_4 y_6	K_2, K_1 K_2	J_3 J_3
z_5	100	z_7	110	1	y_5	-	J_2
z_6	101	z_1	000	1	-	K_3, K_1	-
z_7	110	z_1	000	1	-	K_3, K_2	-

5. На підставі даних табл. 6.5 записуються системи логічних рівнянь для функцій збудження входів JK-тригерів і виходів:

- для функцій збудження входів:

$$J_1 = z_1\beta_1 \vee z_3,\quad J_2 = z_2 \vee z_5,\quad J_3 = z_4,$$
$$K_1 = z_2 \vee z_4\overline{x_1} \vee z_6,\quad K_2 = z_4 \vee z_7,\ K_3 = z_6 \vee z_7; \qquad (6.16)$$

- для вихідних керувальних сигналів:

$$y_1 = z_2;\ \ y_2 = z_3;\ y_3 = z_4;\ \ y_4 = z_5;\ y_5 = z_7;\ y_6 = z_6. \qquad (6.17)$$

6. Будується принципова схема модуля керування МКБ1.

6.8 Проектування спеціалізованого АЛП для операції віднімання

6.8.1 Початкові дані до проекту:

- тип арифметичної операції – віднімання двійкових чисел;
- початковий код подання операндів – доповняльний;
- розрядність операндів – 8 біт;

- код виконання операції у суматорі – доповняльний модифікований;
- структура операційного блока – із закріпленими мікрооперaціями;
- тип керувального блока – автомат Мілі з пам'яттю на D-тригерах;
- схема логічної ознаки переповнення розрядної сітки;
- схема логічного порозрядного множення кодів вхідних операндів A і B.

Елементна база, аналогічна елементній базі пристрою додавання.

6.8.2 Алгоритми віднімання двійкових чисел

Алгоритм віднімання двійкових чисел реалізовується у такій послідовності:

- у регістри RGA і RGB із вхідної шини один за одним паралельним кодом записуються відповідні їм операнди A і B;
- операнд B безумовно інвертується;
- мікрооперація віднімання виконується в доповняльних модифікованих кодах протягом одного машинного такту;
- аналізується результат віднімання. За відсутності переповнення розрядної сітки результат записується у регістр RGC і потім пересилається в оперативну пам'ять, за наявності переповнення результат не фіксується і в ЦПК передається сигнал переповнення $T_п$.

6.8.3 Функціональна схема АЛП2

Функціональна схема восьмирозрядного АЛП2 для виконання мікропрограми віднімання зображена на рисунку 6.11.

Схема містить:

- регістри RGA і RGB для приймання із вхідної шини Ш1 операндів A і B та їх зберігання протягом часу виконання мікропрограми;
- схему інвертування BIN змісту регістра RGB;
- електронні ключі SW1 і SW2 для комутації операндів;
- схему порозрядного логічного множення AND;
- комбінаційний суматор SM з додатковим знаковим розрядом П для створення модифікованого коду. На вхід перенесення першого розряду суматора подається «лог. 1»;
- регістр результату RGC;
- схему ознаки переповнення ОР;
- модуль керувального блока МКБ2 на основі автомата Мілі з пам'яттю на D-тригерах.

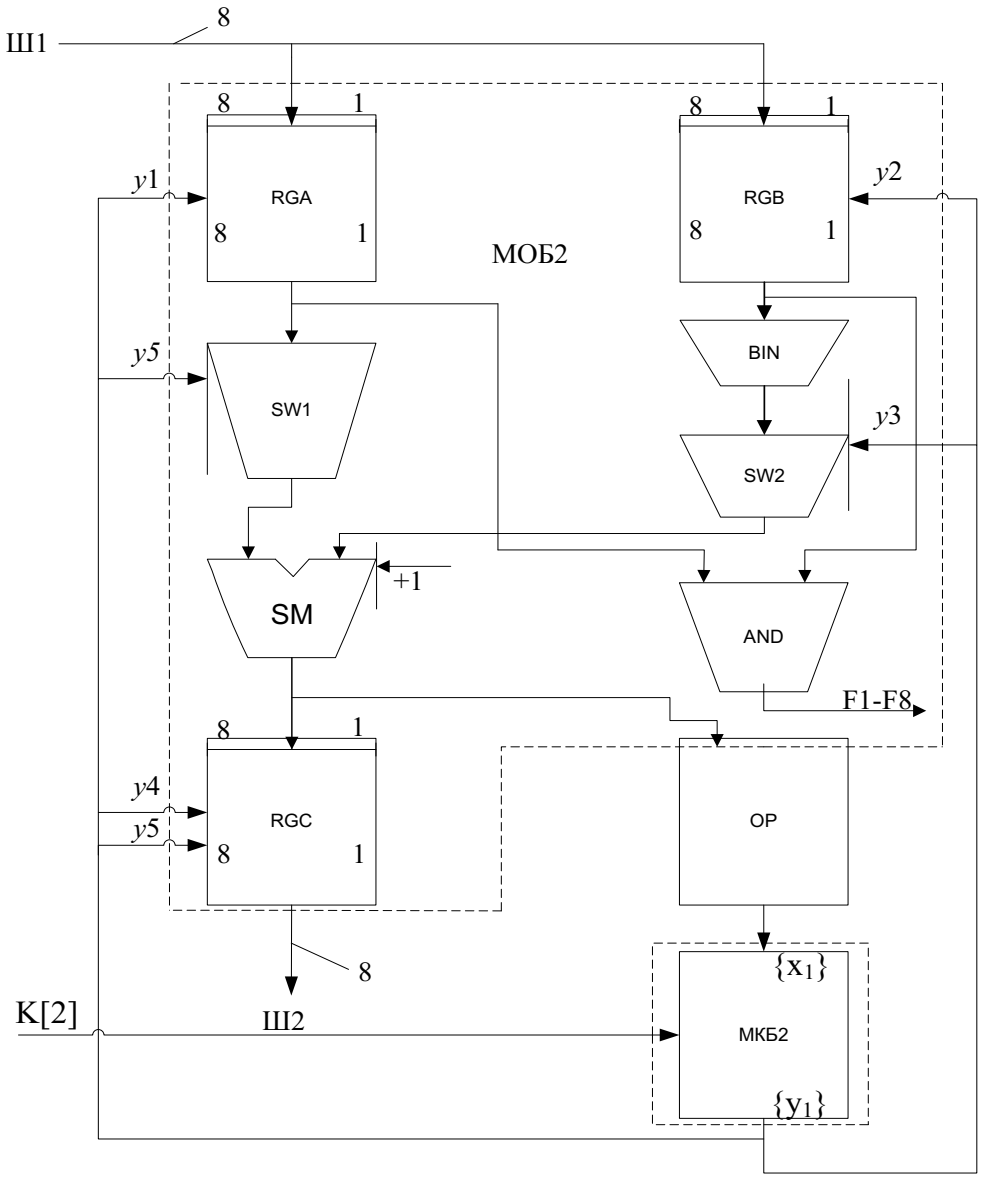


Рисунок 6.11 – Функціональна схема АЛП2
МОБ2 – модуль операційного блока;
МКБ1 – модуль керувального блока;
SW1, SW2 – схеми електронних ключів;
AND – схема порозрядного логічного множення

6.8.4 Мікропрограма віднімання

Мікропрограма віднімання двійкових чисел у модифікованих доповняльних кодах має такий вигляд:

- Початок. Якщо К[2], то М1, інакше – чекати
- М1 y1 : RGA := A <приймання першого операнда>
- y2 : RGB := B <приймання другого операнда>
- y3 : SM := A+$\overline{B}$+1 <мікрооперація віднімання в доповняльних кодах>
- Якщо φ_3, то M_2, інакше
- y4 : RGC : = SM <присвоєння результату>

- у5 : Ш2 : = RGC <пересилання в пам’ять>
- Перейти до М3
- М2 у6 :Тп:= ПП <тригеру переповнення Тп присвоюється ознака ПП>
- М3 Кінець.

Примітка. К[2] – однорозрядний код команди віднімання.

6.9 Проектування спеціалізованого АЛП для операцій додавання і віднімання

6.9.1 Початкові дані до проекту:

- типи арифметичних операцій – додавання та віднімання двійкових чисел;
- початковий код подання операндів – доповняльний;
- розрядність операндів – 8 біт;
- код виконання операцій у суматорі – доповняльний модифікований;
- структура операційного блока – із закріпленими мікрооперaціями;
- тип керувального блока – автомат Мілі з пам'яттю на *D*-тригерах;
- схема логічної ознаки переповнення розрядної сітки;
- схема логічної порозрядної операції ВИКЛЮЧАЛЬНЕ АБО кодів початкових операндів А і В.

Елементна база аналогічна елементній базі попередніх пристроїв.

6.9.2 Алгоритми додавання і віднімання двійкових чисел

Алгоритм додавання та віднімання двійкових чисел можна виконувати в обернених або доповняльних кодах. У сучасних комп’ютерах часто операнди зберігаються у пам’яті і обробляються в доповняльних кодах, перевагу яких описано в попередніх розділах.

Алгоритм додавання (код команди К[1]) або віднімання (код команди К[2]) виконується у такій послідовності:

- у регістри RGA і RGB із вхідної шини один за одним паралельним кодом записуються відповідні їм операнди А і В:
- під час операції віднімання (код команди К[2]=1) операнд В безумовно інвертується:
- мікрооперації додавання або віднімання виконуються в доповняльних кодах протягом одного машинного такту:
- аналізується результат операції. За відсутності переповнення розрядної сітки (знаки операндів А і В та результати збігаються) результат записується у регістр RGD і потім пересилається до оперативної пам'яті, за наявності переповнення (знаки операнда А і В та результат на виході суматора не збігаються) результат не фіксується і в ЦПК подається сигнал переповнення ПП.

6.9.3 Функціональна схема АЛПЗ

Функціональну схему восьмирозрядного АЛПЗ для виконання мікропрограми додавання та віднімання подано композицією модуля операційного блока МОБЗ і модуля керувального блока МКБЗ (рис. 6.12).

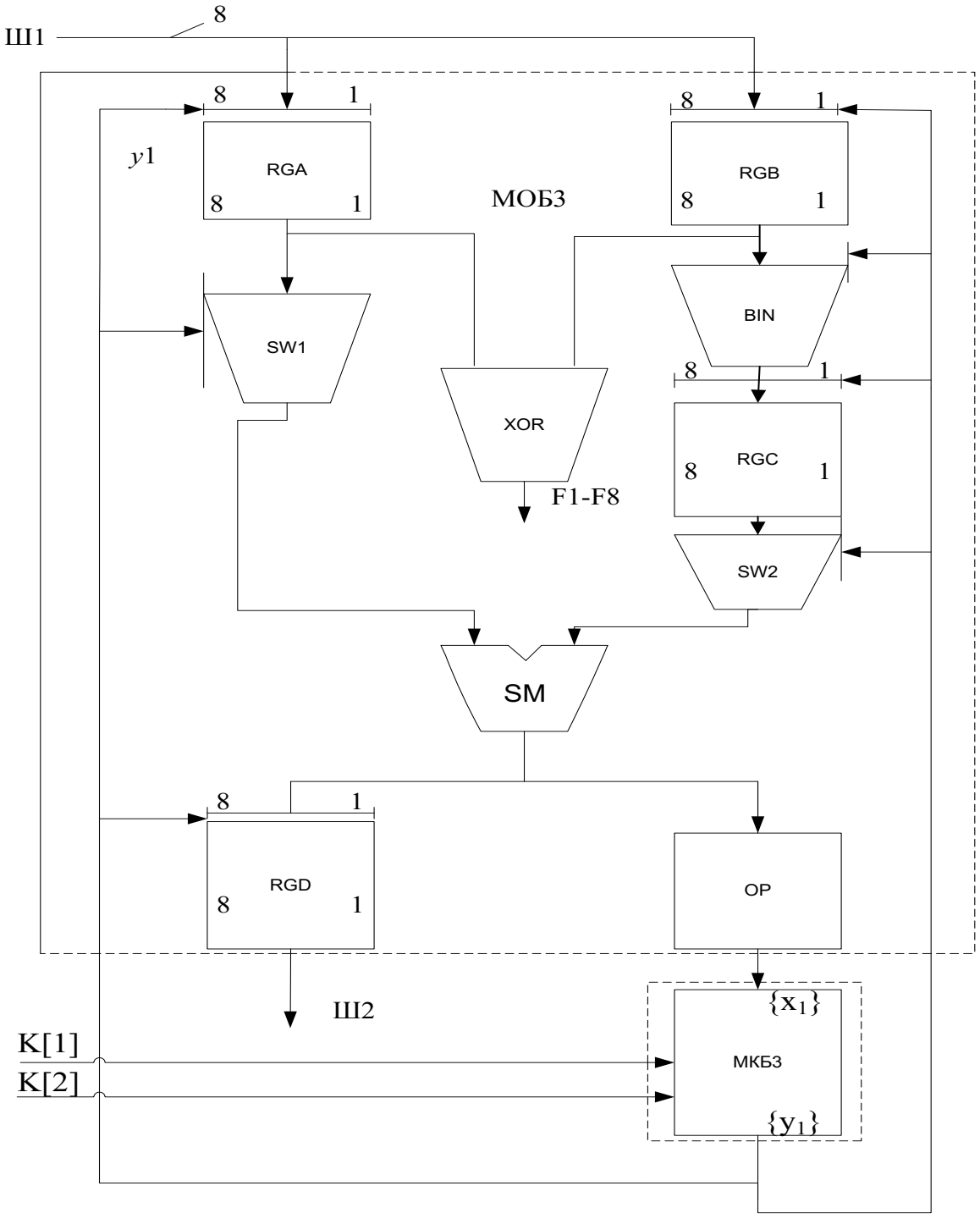


Рисунок 6.12 – Функціональна схема АЛПЗ

Схема містить:

- регістри RGA і RGB для приймання із вихідної шини Ш1 операндів A і B та їх зберігання протягом часу виконання мікропрограми;
- схеми інвертування BIN змісту регістра RGB;
- схеми ХОК для порозрядного логічного ВИКЛЮЧАЛЬНЕ АБО кодів операндів A і B;
- схеми електронних ключів SW1 і SW2 для комутації операндів;
- комбінаційний суматор SM. На вхід перенесення суматора передається лог.1 при операції віднімання;
- регістр результату RGD;
- схему ознаки переповнення ОР;
- модуль керувального блока МКБЗ на основі автомата Мілі з пам'яттю на D-тригерах.

6.9.4 Мікропрограма додавання та віднімання двійкових чисел

Суміщена мікропрограма додавання та віднімання двійкових чисел має вигляд:

- Початок. Якщо К[1] або К[2], то М1, інакше – чекати
- М1 у1 : RG*A* := *A*<приймання першого операнда>
- у2 : *RGB* := *B* <приймання другого операнда>Якщо К[1],то
- у3 : *RGC* := B, інакше
- у4 : *RGC* := $\overline{B}$ <пересилання з інвертуванням другого операнда>
- Якщо К[1],то
- у5 : *SM* := *B*+*C*, інакше у6 : *SM* := *B*+*C*+*1*
- Якщо φ_3, то перейти М2, інакше
- у7 : RG*D* := SM <присвоєння результату>
- у8 : Ш2 := D <пересилання результату в оперативну пам'ять>
- Перейти до М3
- М2 у9 :Тп := ПП <фіксація перетворення>
- М3 Кінець.

6.10 Проектування спеціалізованого АЛП для операції множення

6.10.1 Початкові дані до проекту:

- тип арифметичної операції – множення двійкових чисел;
- початковий код подання операндів – прямий;
- розрядність – 8 біт;
- код виконання мікрооперації у суматорі – доповняльний;
- структура операційного блока – із закріпленими мікрооперaціями;
- тип керувального блока – автомат Мілі з пам’яттю на RS-тригерах.

Елементна база аналогічна елементній базі попередніх пристроїв.

6.10.2 Алгоритм множення двійкових чисел із зсувом суми часткових добутків вправо

Множення двійкових чисел A і B зводиться до обчислення добутку їх модулів та присвоєння йому знака. Добуток двох n-розрядних операндів містить 2n-1 цифрових розрядів і один знаковий. Якщо перемножуються цілі числа, кома розміщується після молодшого розряду, а якщо дроби – перед старшим розрядом.

1. Множене і множник у прямих кодах послідовно записуються відповідно в регістри A і B. Регістри C і D обнуляються. У лічильник CT записується кілька циклів.

2. Для розрядів множника 1, 2, ... n-1 виконуються такі дії:

- якщо молодша цифра множника B(1) = 1, то до суми часткових добутків додається модуль множеного, інакше не додається;
- далі для обох випадків вміст регістра C і B зсувається вправо на один розряд, причому молодший розряд регістра C передається в старший розряд регістра B. Після кожного зсуву в молодший розряд регістра B надходить наступний розряд множеного, за яким визначається черговий частковий добуток (нуль або множене);
- після n-1 циклів виконується додатковий зсув вправо для передачі в тригер T знака множника і визначається знак добутку додаванням за модулем двох знаків множеного і множника.

3. Результат подається конкатенацією чисел C і B при цьому в регістрі C розміщуються старші розряди добутку, а в регістрі B – молодші розряди.

6.10.3 Функціональна схема АЛП4 для операції множення

Функціональна схема АЛП4 для множення цілих двійкових чисел із зсувом вправо часткових добутків містить (рис. 6.13):

- регістри A і B для приймання з вихідної шини Ш1 відповідно множеного та множника;
- паралельний комбінаційний суматор SM;
- регістр C для приймання часткової суми із SM при одиничному значенні синхросигналу;
- регістр D для приймання і тимчасового зберігання часткової суми регістра C при спаданні синхросигналу;
- лічильник циклів CT;
- тригер T1 для керування ключами SW1, SW2;
- тригер T2 для записування знака множника;
- регістри C і B забезпечують зсув вправо чисел, при цьому значення молодшого розряду регістра C[1] пересилається в старший розряд регістра B[n].

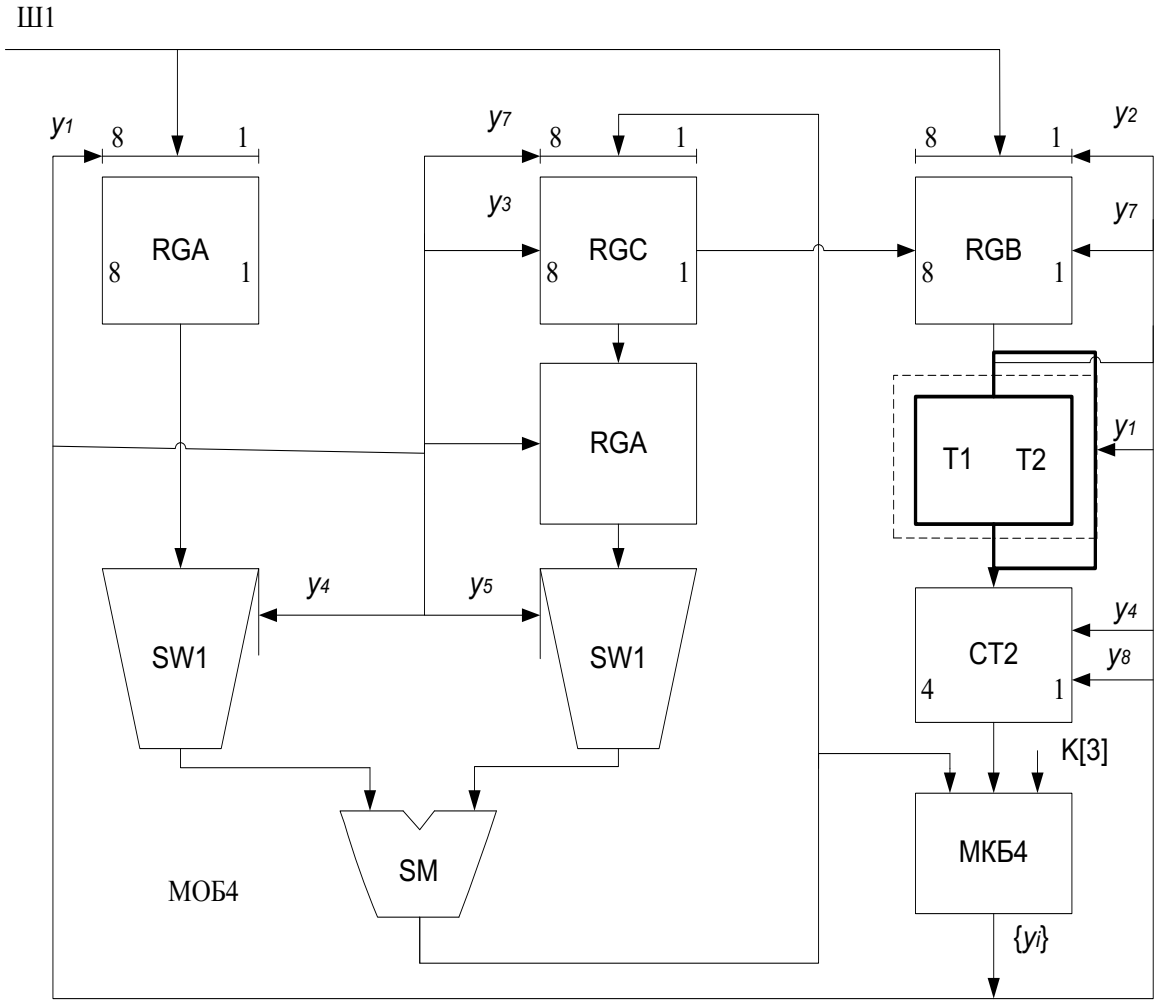


Рисунок 6.13 – Функціональна схема АЛП4
МОБ4 – модуль операційного блока;
SW1, SW2 – електронні ключі;
МКБ4 – модуль керувального блока АЛП4

6.10.4 Мікропрограма множення цілих чисел

- Початок. Якщо К[3], то М1, інакше чекати
- М1 y1: RGA:= A; T2,T1:=0.0 <приймання множеного та обнулення тригерів T1,T2 >
- y2 : RGB:= B; T2:=B[п] <приймання множника та дублювання його знака в тригері T2 >
- y3 : RGC.RGD:= 0.0 <обнулення регістрів >
- y4 : CT:= n-1 <записування кількох циклів >
- М2 Якщо $\overline{B}[1]$, то М3
- y5 : SM:= $|A| + D$; <додавання>

  y6 : RGC.:= SM <пересилання часткової суми >
- М3 y7 : RGC.RGB:= R(C,B) <однорозрядний зсув вправо >

6.11 Проектування спеціалізованого АЛП для операції ділення

6.11.1 Початкові дані до проекту:

- тип арифметичної операції – ділення двійкових чисел;
- початковий код подання операндів – прямий;
- розрядність – ділене X – 16 біт, дільник Y – 8 біт;
- код виконання мікрооперацій у суматорі – доповнювальний;
- структура операційного блока – із закріпленими мікрооперераціями;
- тип керувального блока – автомат з пам'яттю на JK-тригерах.

Елементна база аналогічна елементній базі попередніх пристроїв.

6.11.2 Алгоритм ділення цілих чисел

Операція ділення цілих чисел $Z=X/Y$ зводиться до послідовності віднімання дільника Y спочатку від діленого X, а потім від створюваних в процесі ділення залишків R_i.

Залежно від способу віднімання дільника Y розрізняють два основних алгоритми ділення: без відновлення залишку R_i та з відновленням залишку.

Обидва способи реалізовуються приблизно однаковими апаратними затратами, але для ділення без відновлення залишку потрібно більше мікрооперацій додавання і віднімання. В універсальних комп'ютерах зазвичай використовують ділення без відновлення залишку.

У разі ділення цілих чисел часто ділене X подається в $2n$-розрядному форматі, а дільник Y – у n-розрядному.

Алгоритм ділення цілих чисел у прямому коді без відновлення залишку реалізовується в такій послідовності:

1. У регістри A, B і C послівно записуються із вхідної шини n-розряднній дільник $У$ та $2n$-розрядне ділене X. У лічильник циклів CT заноситься число циклів n-1;
2. Ділене X та дільник Y аналізується на рівність нулю, якщо ділене $X=0$, то частці Z присвоюється нульове значення і ділення закінчується. Якщо дільник $Y = 0$, то ділення переривається;
3. Установлюється можливість ділення без переповнення розрядної сітки.

Для цього значення діленого подвоюється зсувом вліво на один розряд. Із зсунутого діленого віднімається дільник. Операція віднімання дільника замінюється на його додавання у доповнювальному коді і визначається перший залишок R_0 за формулою

$$R_0 = 2|X| - Y = 2|X| + |-Y|. \qquad (6.18)$$

Якщо $R_0 < 0$, то ділення можливе, якщо $R_0 > 0$, то виникає переповнення розрядної сітки і ділення припиняється.

4. Якщо ділення можливе, виконуються такі основні дії:

- частковий залишок в регістрі B і вміст регістра C зсуваються вліво на один розряд (тобто подвоюються);

- із зсунутого залишку віднімається дільник, якщо попередній залишок $R_{i\text{-}1} > 0$, або додається, якщо $R_{i\text{-}1} < 0$. Це визначається рекурентним співвідношенням

$$R_i = \begin{cases} 2R_{i-1} + |Y|; \\ 2R_{i-1} + |-Y|_{Д}, \end{cases} \tag{6.19}$$

де $i = 1, 2, \ldots n\text{-}1$.

Якщо відбувається зсув вліво, в молодший розряд регістра записується цифра частки r_i згідно з співвідношенням

$$r_i = \begin{cases} 0, \textit{ якщо } R_{i-1} < 0; \\ 1, \textit{ якщо } R_{i-1} \geq 0. \end{cases} \tag{6.20}$$

Це означає, що поточна цифра частки є інверсною знака залишку.

Значення першого (старшого) розряду частки відводиться для записування знака результату на підставі виразу $r_0 = A[n] + B[n]$.

5. Залишок ділення розміщується у регістрі *B* на місці старших розрядів діленого, а частка – в регістрі *C*. Дільник, залишок і частка мають формат *n*-розрядного числа із знаком. Залишок має мати той же знак, що і ділене, нульові залишки і частки завжди додатні. Якщо знак останнього залишку від'ємний, то він коректується додаванням до нього модуля дільника; після цього залишку присвоюється знак діленого.

6.11.3 Функціональна схема АЛП5 для мікропрограми ділення

Функціональну схему АЛП5 для виконання мікропрограми ділення показано на рис. 6.14.

Схема АЛП5 містить:

- регістр RGA для приймання восьмирозрядного дільника Y;
- регістри RGB iRGC для приймання 16-розрядного діленого X;
- комбінаційний восьмирозрядний суматор SM;
- мультиплексом MUX, який забезпечує записування інформації в регістр RGB із вхідної шини Ш1 або з виходів суматора SM;
- схему BIN для подання прямого або оберненого коду дільника на входи суматора SM;
- лічильник кількості циклів CT;
- схему вироблення ознак результату OP;
- тригери T1–T4 для створення другого рівня керування модулем операційного блока МОБ5;
- місцевий керувальний блок МКБ5 на основі автомата Мілі з пам'яттю на JK-тригерах.

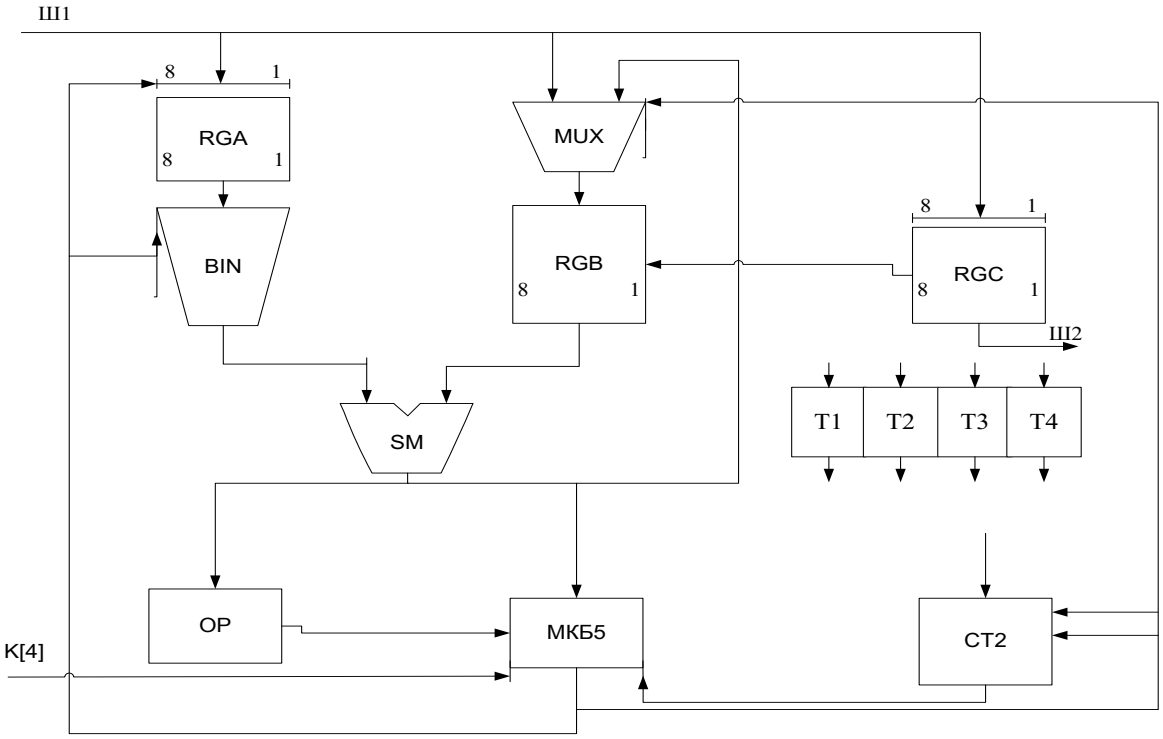


Рисунок 6.14 – Функціональна схема АЛП5

6.11.4 Мікропрограма ділення цілих чисел без відновлення залишку:

- Початок. Якщо К[4], то М1, інакше – чекати

- М1 y1 :RGA := Y; T1.T2.T3.T4 := 0.0.0.0 <завантаження дільника і обнулення тригерів>

y2 : RGB:= X1; T1:= B [*n*] <завантаження старшої частини діленого; присвоєння тригеру Т1 знака діленого>

y3 : RGC:= X2 <завантаження молодшої частини діленого>

y4 : RGB. RGC:= L (B.C) T_3 <зсув вліво на один розряд вмісту регістрів RGB і RGC. Передача значення старшого розряду регістра RGC в молодший розряд регістра RGB, а в молодший розряд регістра RGC – змісту тригера Т3>

y5 : T3 := ЗНАК= A[*n*] $\oplus$T1 <записування в тригер Т3 знака результату ділення>

y6: СТ:= *n*-1 <завантаження в лічильник СТ кількості циклів>

y7 :SM := B+[-A]$_Д$ <віднімання дільника в доповняльному коді відзначення даних в регістрі RGB>

y8 : ADR.RGB := 1.SM <завантаження результату віднімання в регістр RGB. Входу адресації ADR мультиплексора MUX надається значення одиниці>

y9 :T4 := SM [*n*] <присвоєння знака залишку тригеру Т4 >

- Якщо Т4, то М2, інакше

y10 :$T_П$:= ПП <присвоєння ознаки переповнення тригеру $T_П$ в ЦПК>. Перейти до М7 (кінець)

- М2 y4 : RGB.RGC := L (B.C.)T3.
- М3 Якщо Т4, то М4, інакше
  y7 :SM := B+[-A]$_д$
  y12 :T3.T4 := 0.0 <обнулення тригерів Т3, Т4>Перейти до М5
- М4 y11 :SM := B+| A| <додавання дільника до залишку>
  y12 : T3.T4:=0.0
- М5 y8 : ADR.RGB := 1.SM
  y9 : T4:= SM [n]
  y13 :T3:= $\overline{SM}$ [n] < присвоєння тригеру Т3 інверсного значення знака суматора – формування цифри частки>
  y14 :CT:= CT-1 <декремент вмісту лічильника>
- Якщо CT≠ 0, то М3, інакше М5
- М5 y15 : RGC : L(C).T3 <зсув вліво вмісту регістра RGC і записування останньої молодшої цифри частки>
- Якщо $\overline{B}$[n], то М6, інакше
  y11 :SM:= B+| A|
  y8 :ADR .RGB : = 1.SM
- М6 Якщо B[n] = T 1, то М. Інакше
  y15 : RGB: L(B.0) <зсув вліво залишку>
  y16 : RGB T R B <зсув вправо частки і присвоєння їй знака діленого>
- М7 Кінець.

Індивідуальні завдання

1. Синтезуйте та промоделюйте схему порівняння слова $A=A_2A_1A_0$ з заданими константами: $F_1 := (A = 011)$; $F_2 := (A > 100)$ і $F_3 := (A \leq 101)$.

2. Синтезуйте та промоделюйте схему порівняння двох восьмирозрядний двійкових слів А і В.

3. Синтезуйте та промоделюйте схему порівняння двох 6-розрядних слів А і В «НА БІЛЬШЕ».

4. Синтезуйте та промоделюйте схему контролю за парністю восьмирозрядного числа А.

5. Синтезуйте та промоделюйте схему перетворювача прямого коду на обернений для восьмирозрядного числа А.

6. Синтезуйте та промоделюйте схему перетворювача прямого коду на доповняльний для трирозрядного числа А.

7 СПЕЦІАЛЬНІ ЕЛЕМЕНТИ ЦИФРОВИХ ПРИСТРОЇВ

Значну частину сучасного цифрового пристрою складають блоки керування, обміну інформацією, індикацій, контролю, діагностики тощо. У цих блоках використовують схеми, що виконують різні спеціальні функції (перетворення рівнів, генерування різних сигналів, формування часових параметрів сигналів).

7.1 Логічні розширники

Логічні розширники – спеціальні елементи цифрових пристроїв, призначені для збільшення кількості логічних входів у логічних елементах, розширення класу реалізованих цими елементами логічних функцій і побудови нетипових схем.

Через те, що в елементах ТТЛ-типу операція І реалізовується за допомогою багатоемітерного транзистора, збільшити кількість відповідних входів зовнішнім монтажем неможливо.

В елементах ТТЛ-типу розширники призначені для розширення класу реалізовуваних функцій, тобто для реалізації функції АБО (рис. 7.1).

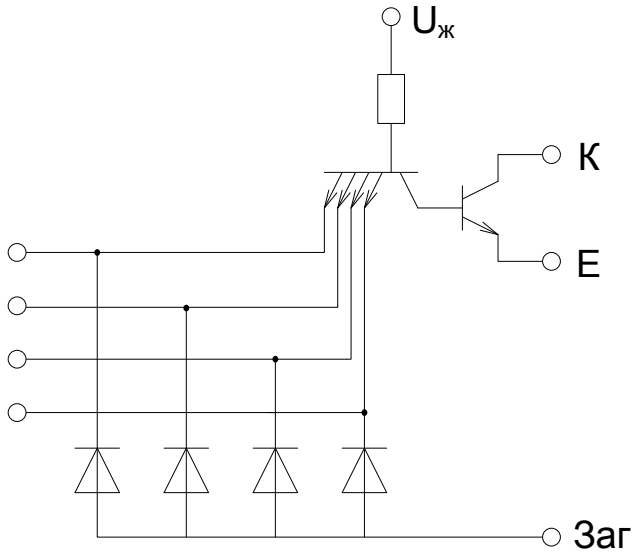


Рисунок 7.1 – Схема розширника за АБО на чотири входи для елемента ТТЛ-типу

Виводи К і Е розширника з'єднуються з відповідними виводами К і Е базових логічних елементів.

Розширення логічних можливостей базових вентилів забезпечується різною їх комбінацією, як, наприклад, показано на рис. 7.2.

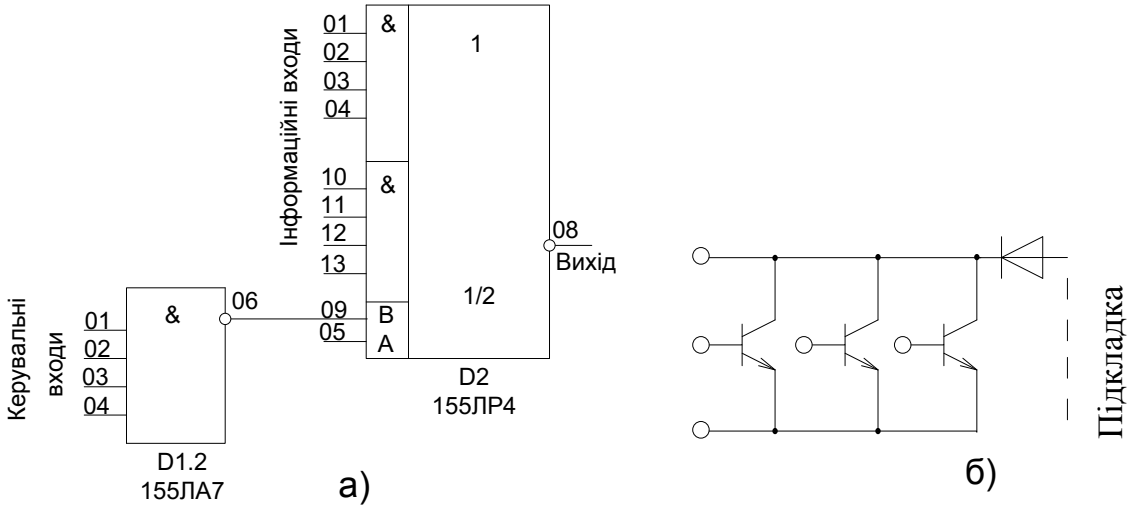


Рисунок 7.2 – Схема елемента ТТЛ-типу з трьома стійкими станами (а), типовий розширник за АБО для елементів ЕЗЛ-типу (б)

7.2 Перетворювачі рівнів

Крім частин керувальної системи, добре реалізованих засобами на основі типових комплектів ВІС мікропроцесора, у типовій апаратурі керувальної системи є значна кількість засобів сполучення з об'єктом керування, індикації, документування тощо.

Перетворювачами рівнів (адаптерами, драйверами, трансляторами) називають спеціальні елементи цифрових пристроїв, призначені для забезпечення сумісності логічних рівнів різних сімей цифрових елементів.

Крім забезпечення сумісності рівнів сигналів перетворювачі рівнів мають задовольняти спеціальні вимоги, наприклад, такі, як збереження перетворювачем граничного рівня керувального елемента, рівнів струмів, способу кодування двійкових змінних.

Більшість інтегральних схем з високим рівнем інтеграції виконано на основі p-, n- або КМДН-технології, тоді як схеми малого і середнього рівнів інтеграції — на основі ТТЛ-, ЕЗЛ- і КМДН-технології.

У складі схем малого і середнього ступеня інтеграції ТТЛ-, ЕЗЛ- та КМДН-типу є спеціально розроблені перетворювачі рівнів. Це перетворювачі ЕЗЛ-ТТЛ К500ПУ 125; перетворювач ТТЛ -ЕЗЛ К500ПУ 124; перетворювачі КМДН-ТТЛ 176ПУ1, 176ПУ2, 176ПУЗ, 564ПУ4, 564ЛН1, 564ЛН2; перетворювачі ТТЛ-КМДЛ ІЗЗЛНЗ, 133ЛН5 і т. ін. (рис. 7.3).

На рис. 7.4, а) наведено приклад стикування КМДН-схем, що працюють при високому рівні напруги джерела живлення, із КМДН-схемами, що працюють з низьким рівнем напруги джерела живлення.

Більш складну схему подільника зображено на рис. 7.4, б), де наведено схему перетворювача рівня КМДН —ТТЛ для випадку, коли КМДН-схеми живляться напругами більшими, ніж 5 В.

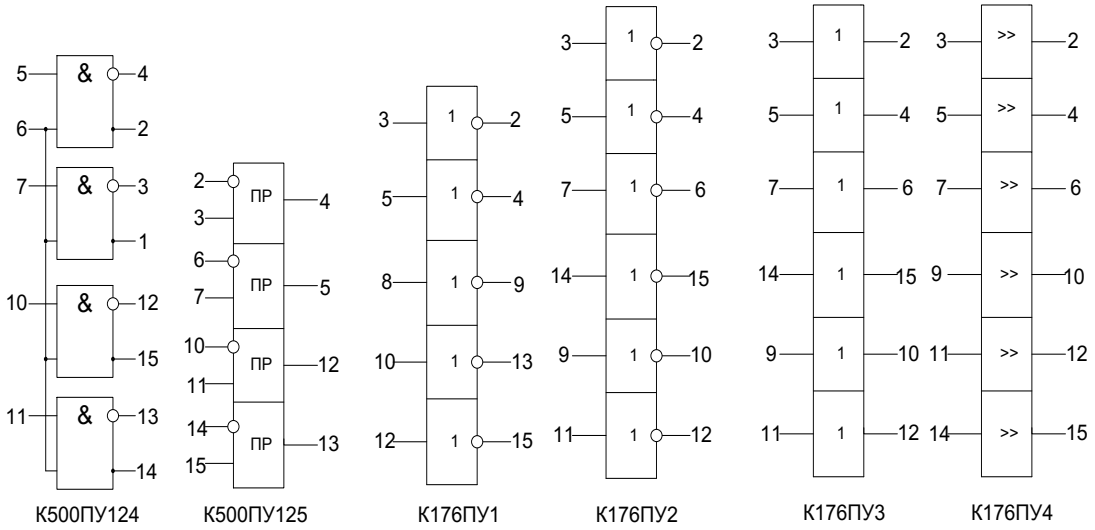


Рисунок 7.3 – Перетворювачі рівнів

«Активним» елементом перетворювача рівнів у цій схемі є будь-яка ТТЛ-схема, що заземлюється, тому транзистор VT1 завжди закритий. Зовнішній транзистор VT4 приєднується до розширювальних входів ТТЛ-схеми. Керування транзистором VT4 здійснюють подільники R5, R6.

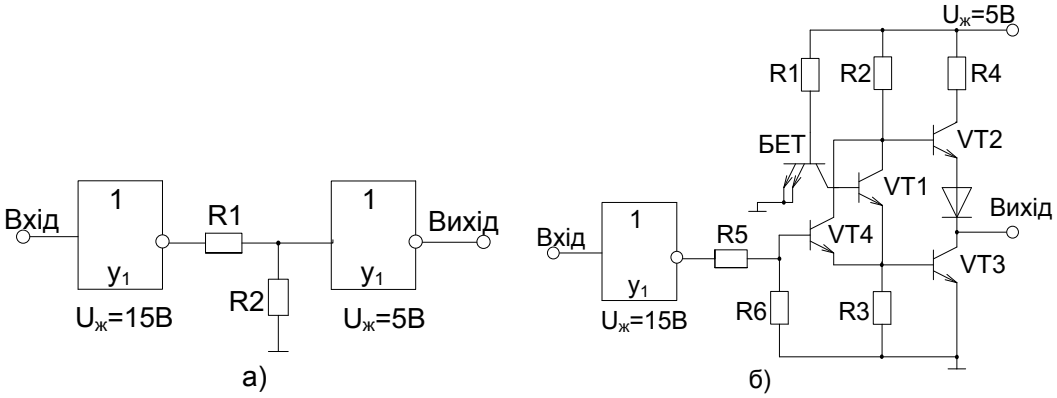


Рисунок 7.4 – Приклади схем перетворення високого рівня на низький для схем КМДН-типу (а) та ТТЛ-типу (б)

Перемикальна схема з комбінацією транзисторів n-p-n і p-n-p-типів зображена на рис. 7.5.

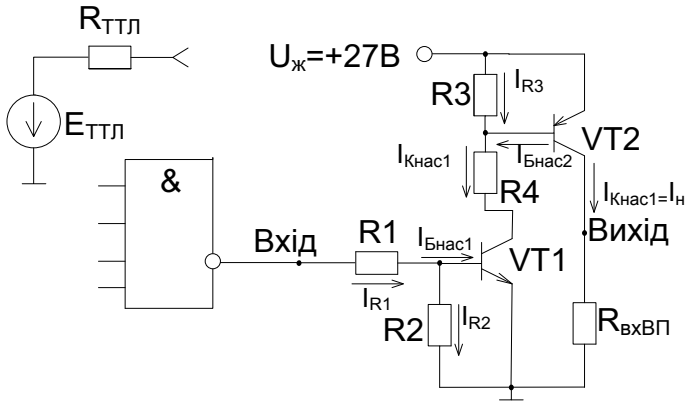


Рисунок 7.5 – Перемикальна схема з комбінацією транзисторів n-p-n і p-n-p-типів

Щоб забезпечити насичений режим транзистора VT1, опір R1 має дорівнювати 390 Ом. Напруга на виході керувального елемента знаходиться майже на межі допустимого значення. Це свідчить про те, що керувальний елемент не зможе працювати на інші елементи ТТЛ-типу при надмірно малих опорах R2.

Схеми, що працюють на принципі перемикання струму (рис. 7.6), використовують як перетворювачі рівнів у тих випадках, коли логічний перепад становитиме частки вольтів.

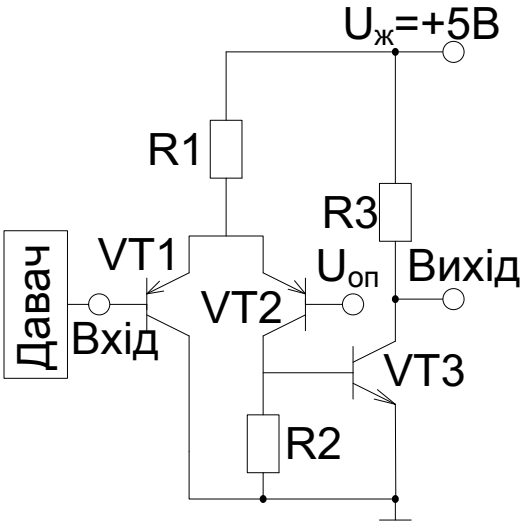


Рисунок 7.6 – Схема перетворювача рівнів на принципі перемикання струму

7.3 Генератори та одновібратори

Генератори – спеціальні елементи цифрових пристроїв, призначені для формування послідовності електричних сигналів різної форми. Послідовність сигналів може бути регулярною або з перериваннями,

зокрема зі зміною параметрів і форми електричних сигналів. Генератори забезпечують роботу цифрового пристрою в часі за законом, зумовленим внутрішньою структурою пристрою, і характеризуються частотою сигналу, стабільністю частоти, можливістю керування частотою, формою сигналу, щілястістю, видом послідовності сигналу і под. Таким чином, генератори за структурою можуть змінюватися від найпростішого автоколивального мультивібратора до складного цифрового пристрою.

На рис. 7.7 наведено схему генератора, в якому конденсатор С забезпечує час затримки, необхідний для створення позитивного зворотного зв'язку, і від його ємності залежить частота генерації.

Розрядний струм забезпечується вихідним колом вентиля Y_1 і сприймається вихідним колом вентиля Y_2 (впливом малого струму вентиля Y_3 нехтуємо).

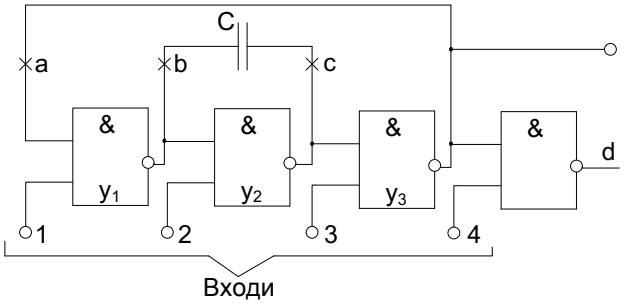


Рисунок 7.7 – Схема генератора на трьох елементах І–НЕ серії К1533

У процесі розряду конденсатора С установлюється певний режим, за якого вихідний струм закритого вентиля Y_1 дорівнює вихідному струму відкритого вентиля Y_2, причому

На рис. 7.8 а), б) наведено варіанти генераторів, виконаних на двох вентилях І–НЕ серії К1533.

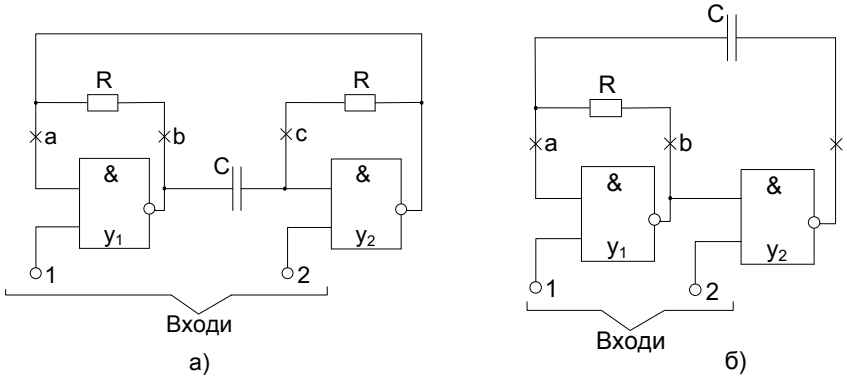


Рисунок 7.8 – Схема генераторів на двох вентилях І–НЕ

Схема керованого генератора (рис. 7.9) побудована на базі мультивібратора з емітерним зв’язком, у якому транзистори VT1 і VT2 утворюють підсилювальний каскад з позитивним зворотним зв’язком.

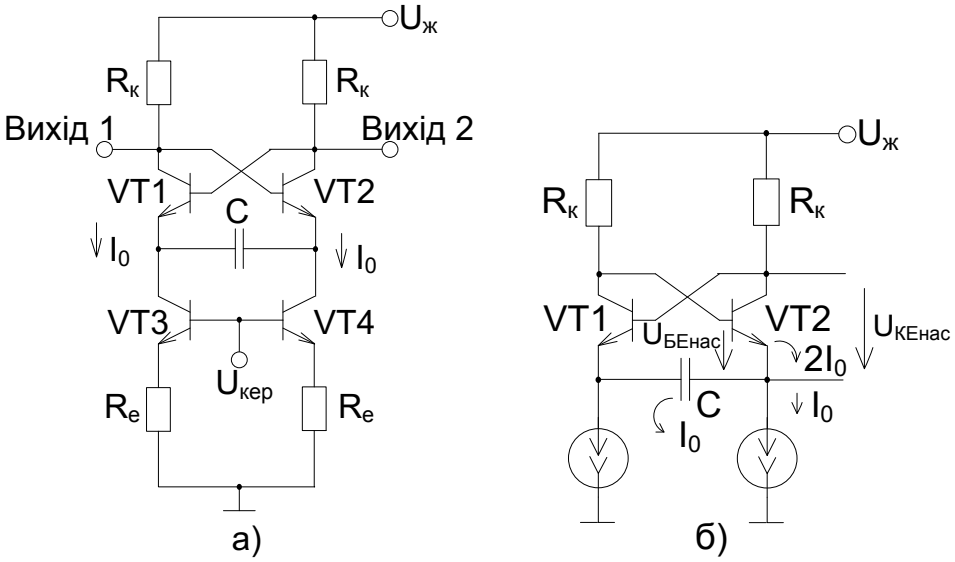


Рисунок 7.9 – Схема керованого генератора на базі мультивібратора з емітерним зв’язком: а – принципова схема; б – розрахункова схема

Розглянемо процеси, що відбуваються в схемі на рис. 7.9, б) відразу після насичення транзистора VT2. Позитивний потенціал на конденсаторі С, що існував перед моментом перемикання, надійно закриває транзистор VT1, і конденсатор починає розряджатися постійним струмом I_0. Як тільки потенціал емітера транзистора VT1 чи лівої обкладки конденсатора С стає транзистор VT1 стрибком входить у стан насичення, а транзистор VT2 закривається. Далі відбувається аналогічний процес.

Істотний недолік розглянутої схеми — у процесі регулювання значно змінюється потенціал на виходах, а логічний перепад малий (лише 0,6 В). Однак через те, що виходи 1 і 2 інверсні, цей недолік можна усунути, використовуючи швидкодійний компаратор, виконаний, наприклад, на операційному підсилювачі.

За необхідності побудови генераторів, частота яких порівнянна з граничною частотою БЛЕ, застосовують схеми, принцип дії яких оснований на використанні власних інерційних властивостей ЛЕ. У таких схемах немає зовнішніх елементів, що задають час, і частота вихідних коливань визначається часом затримки поширення імпульсу ЛЕ. Подібні схеми складаються з N послідовно ввімкнених ЛЕ, охоплених колом одиничного негативного зворотного зв'язку (рис. 7.10, а). Число послідовно ввімкнених елементів має бути непарним.

Роботу схеми пояснимо за допомогою часових діаграм, наведених на рис. 7.10, б. У вихідному стані перемикач S замкнений і схема перебуває у стійкому стані, за якого вихідні напруги всіх непарних елементів дорівнюють U^1, а парних – U^0.

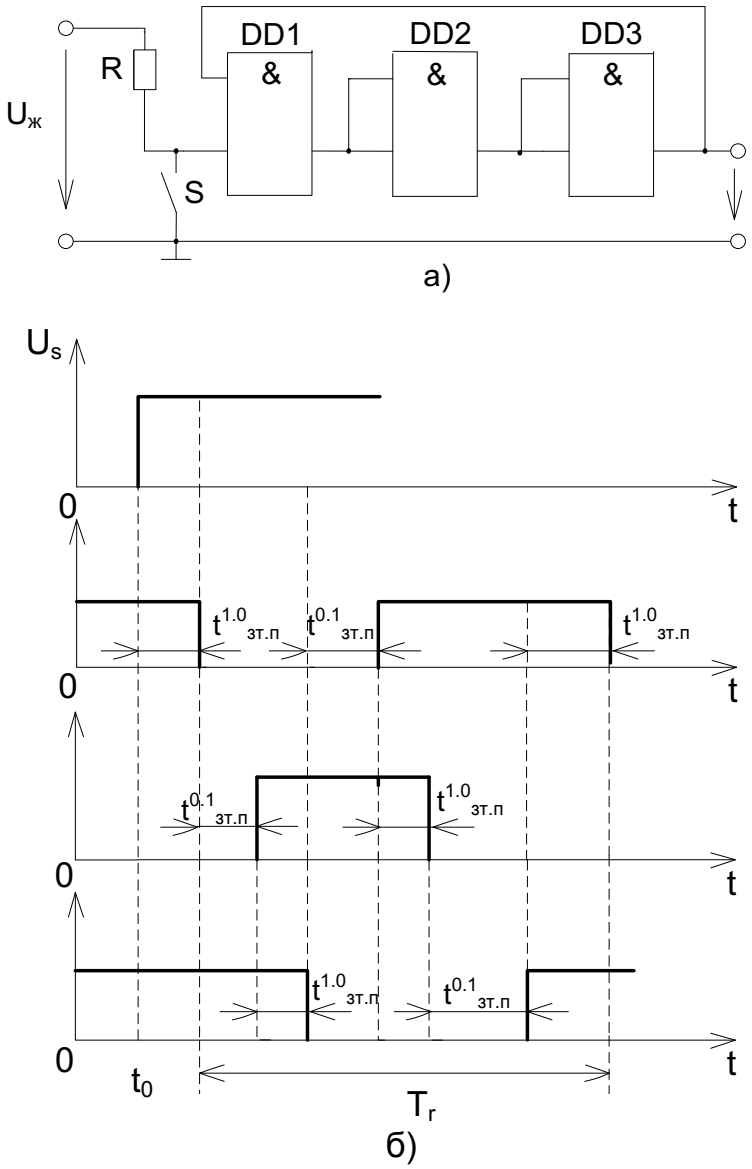


Рисунок 7.10 – Швидкодійний автогенератор (а) та часові діаграми (б)

Розмикання у момент t_0 перемикача S рівнозначне подаванню на вхід першого ЛЕ двох одиничних вхідних сигналів. Тому через час відбудеться зміна його вихідної напруги. Сигнал U^0 з виходу DD1 потрапляє на вхід другого ЛЕ, що, у свою чергу, через часовий інтервал змінить і його вихідну напругу з U^0 до U^1 і т. д. Перемикання елементів відбуватиметься послідовно один за одним.

Частота коливань такого генератора для ІС серії 555 лежить у діапазоні десятків мегагерців.

Схема, наведена на рис. 7.11, може працювати в трьох режимах.

При рівні «0» на вході 1 схема генерує на частоті кварцового резонатора А; при рівні «0» на вході 2 схема генерує на частоті кварцового резонатора В; при рівні «0» на вході 3 схема блокується.

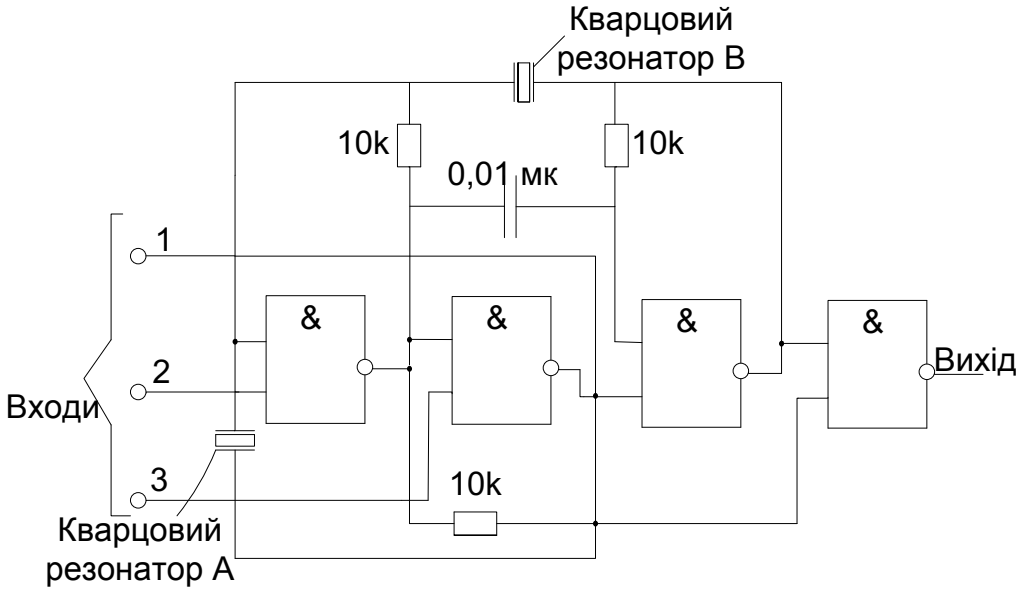


Рисунок 7.11 – Схема двочастотного генератора

Генератор формує прямокутні імпульси з коефіцієнтом заповнення приблизно 40 відсотків, що сумісні з рівнями ТТЛ.

У генераторі застосовують малопотужні елементи ТТЛ-типу, що дає змогу одержувати вихідні імпульси з частотою, не більшою ніж 1…10 МГц. Цей генератор складається з двох ідентичних схем, кожну з яких виконано за схемою, наведеною на рис. 7.8, а).

Одновібратором називають пристрій, що виробляє вихідний імпульс за одиничним перепадом вхідного сигналу. Тривалість вихідного імпульсу визначається сталою часу *RC* вбудованих або зовнішніх компонентів і не залежить від часових обмежень, що накладаються системними тактовими імпульсами.

У складі деяких серій сучасних інтегральних мікросхем є одновібратори двох типів: без повторного запуску і з повторним запуском. На рис. 7.12 зображено функціональну схему одновібратора без повторного запуску К1533АГ1.

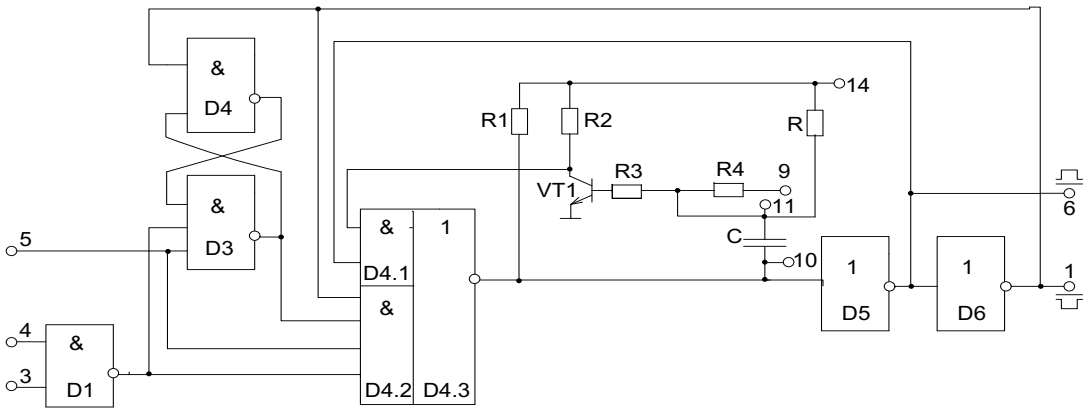


Рисунок 7.12 – Функціональна схема одновібратора К1533АГ1

Варіанти запуску цього одновібратора наведено в табл. 7.1. За будь-яких комбінацій статичних сигналів на входах 3, 4 і 5 одновібратор перебуває у стабільному стані, за якого $Q = 0, \overline{Q} = 1$.

Мікросхема К1533АГ1 належить до багатофункціональних пристроїв. За відсутності зовнішніх компонентів R і C одновібратор можна використовувати як різницевий перетворювач, як генератор імпульсів або для скидання ініціалізації цифрового автомата. Два одновібратори із взаємним запуском утворять генератор, щілястість вихідного сигналу якого можна змінювати в широких межах.

Таблиця 7.1 – Варіанти запуску К1533АГ1

Входи			Виходи	
3	4	5	6	1
0	x	↑		
x	0	↑		
1	↓	1		
↓	1	1		
Одночасний перехід ↓		1		

Стабільність тривалості вихідного сигналу можна підвищити, якщо зовнішній резистор R замінити активним генератором струму I = 0,14...2,7 мА. Застосовуючи керований генератор струму, можна побудувати широтно-імпульсний модулятор з коефіцієнтом перекриття довжини до 20.

Одновібратор з повторним запуском, наприклад мікросхема К1533АГ3, відрізняється від розглянутого раніше тим, що реагує на перепади запуску, навіть під час формування вихідного імпульсу. У цьому випадку на прямому виході залишається сигнал високого рівня і залишатиметься як завгодно довго, якщо час між перепадами запуску буде меншим, ніж тривалість вихідного сигналу, реалізованого від одиничного перепаду запуску, з урахуванням часу відновлення одновібратора. Позначення і функціональну схему ІС К1533АГ3 зображено на рис. 7.13.

Основними частинами схеми є формувач вузького імпульсу D3, внутрішні і зовнішні компоненти, що забезпечують формування тривалості вихідного імпульсу, тригер Шмідта, виконаний на транзисторах VT4 і VT5, логічні елементи й інвертори, що забезпечують стандартні рівні схем ТТЛ-типу. Область гістерезису тригера Шмідта обмежена рівнями напруги 1,1 і 1,9 В на базі транзистора VT4 та контакту 07.

Запуск одновібратора здійснюється негативним перепадом на вході $\overline{D}$ при $D = 1$ і $R = 1$ або позитивним перепадом на вході D при $\overline{D} = 0$ і $R = 1$ (табл. 7.2) або позитивним перепадом на вході R при $\overline{D} = 0$ і $D = 1$.

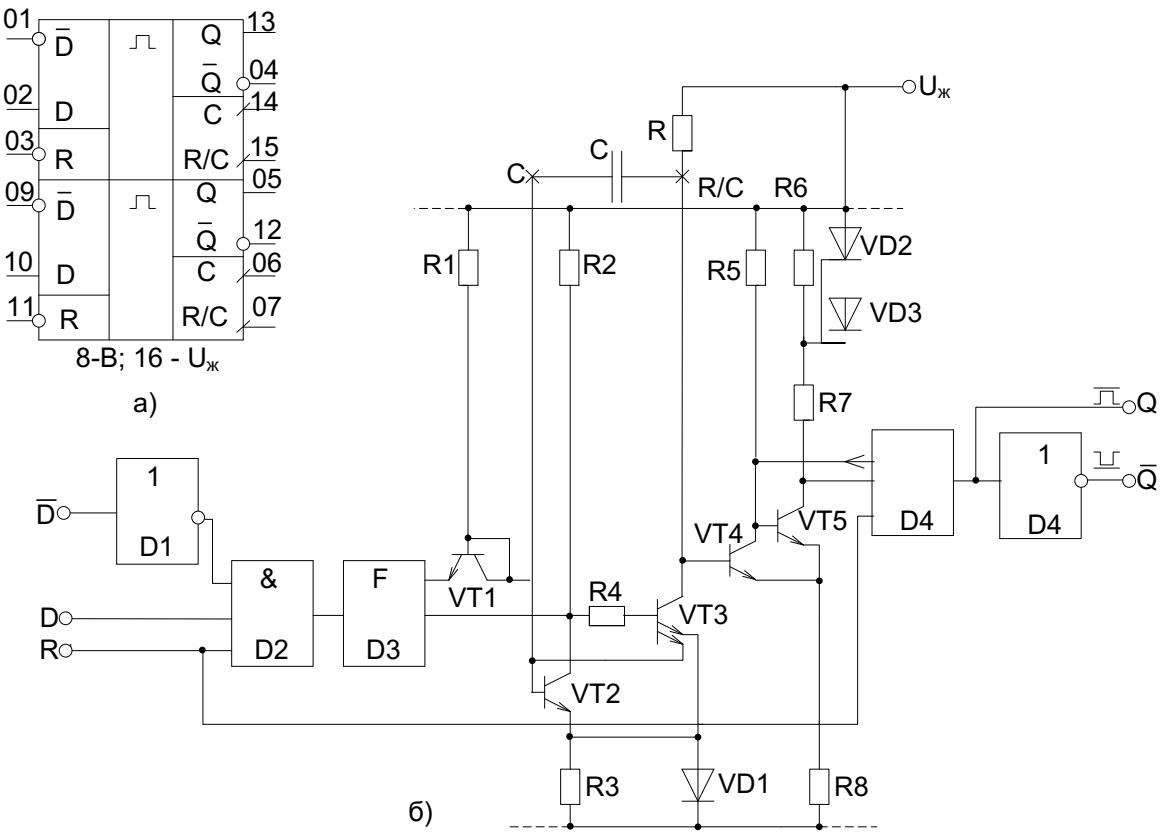


Рисунок 7.13 – Позначення (а) і функціональна схема
ІС К1533АГ3 (б)

За будь-яких комбінацій статичних сигналів на входах $\overline{D}, D$ і R одновібратор перебуває у стабільному стані, за яким $Q = 0, \overline{Q} = 1$. Зовнішні компоненти і C визначають тривалість вихідного імпульсу. Обмежень на величину ємності конденсатора C не накладається. Можливість підключення електролітичного конденсатора розглянуто нижче. Величина $R = R_{\min}$ визначається вимогою, щоб напруга на базі транзистора VT4 не перевищувала рівня 1,9 В, інакше тригер Шмідта не буде перемикатися.

Тривалість вихідного імпульсу не залежить від періоду вхідних імпульсів, якщо виконується умова

$$(T - \tau_i) > 3T_2, \tag{7.1}$$

де τ_i – тривалість вихідного імпульсу;

T – період вхідних імпульсів;

T_2 – стала часу, яка визначається $T_2 = RC$.

Якщо умова не виконується, то перепад напруги відраховуватиметься від рівня, меншого ніж 0,8 В, що приводить до зменшення τ_i.

7.4 Різницеві перетворювачі і детектори подій (фронтів)

Різницеві перетворювачі (РП) – спеціальні елементи цифрових пристроїв, призначені для вироблення вихідного сигналу, що інформує про зміну значення вхідного сигналу. На виході РП формуються імпульсні сигнали у вигляді короткочасної появи напруги U^0 або U^1 за заздалегідь визначеними переходами сигналу на вході. Тривалість вихідного імпульсу РП залежить від параметрів вхідного сигналу і компонентів РП. Якщо РП виконують на логічних елементах, він крім інформаційного може мати додаткові функціональні входи, які дають змогу враховувати додаткові умови формування вихідного імпульсу, що значно спрощує структури цифрових пристроїв, скорочуючи кількість логічних елементів.

Через те, що на вході РП можуть існувати два види переходу вхідного сигналу, а на виході можуть бути сформовані напруги U^0 і U^1 то можлива побудова чотирьох основних схем РП. Варіанти таких схем на логічних елементах наведено на рис. 7.14.

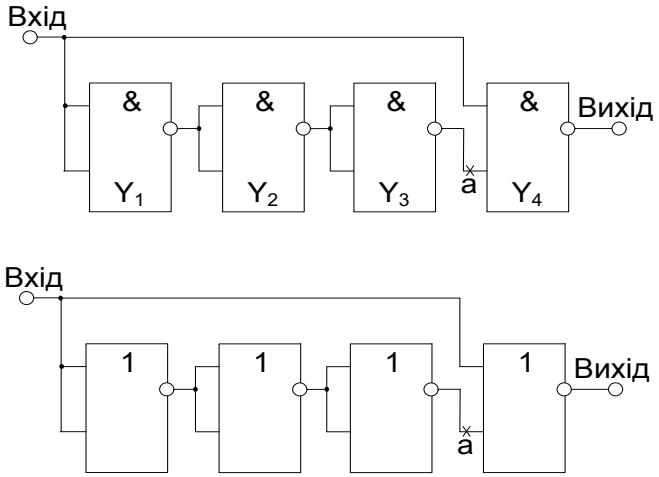


Рисунок 7.14 – Варіанти різницевих перетворювачів на логічних елементах

Якщо припустити, що всі логічні елементи мають ту саму середню затримку поширення сигналу, то тривалість вихідного й імпульсного сигналу всіх РП буде однаковою. У деяких випадках таке значення τ_i недостатнє, тому в РП використовують лінії затримки або RC-кола.

РП використовують для побудови детекторів подій, організації імпульсного керування в RS-тригерах, що усуває на їхніх входах заборонені комбінації сигналів, а також в інших типах тригерів; під час проектування послідовних структур; для вироблення імпульсних сигналів або запуску одновібраторів встановлювальних сигналів для лічильників, регістрів і под.; у разі побудови реверсивних лічильників і регістрів тощо. Усе це дає змогу зараховувати РП до багатофункціональних елементів, і саме з цієї причини в деяких сучасних серіях елементів РП використовують у вигляді інтегральних схем.

Детектори подій (фронтів). Подія в цифрових пристроях – зміна логічного стану в будь-якому колі, тобто позитивні чи негативні перепади (фронти). Детектор фронтів має формувати імпульси з фронтів будь-якої полярності. Схему детектора фронтів на елементах І–НЕ наведено на рис. 7.15.

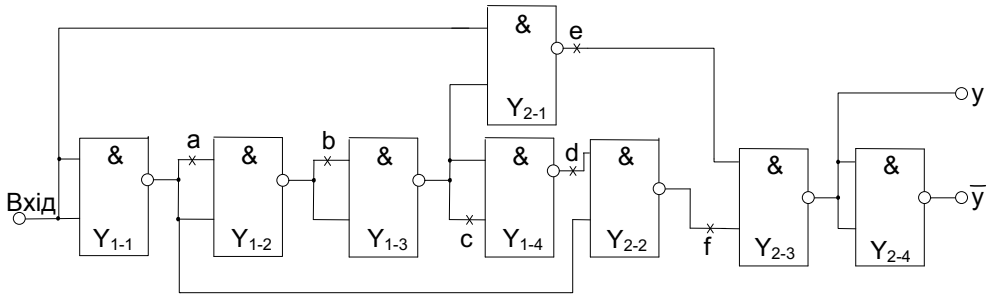


Рисунок 7.15 – Схема детектора фронтів на елементах І–НЕ

Принцип дії схеми оснований на використанні затримки поширення сигналу.

При рівні «0» на вході РП (і на першому вході вентиля $Y_{2\text{-}1}$) на виході вентиля $Y_{2\text{-}1}$ є рівень «1». На другому вході вентиля $Y_{2\text{-}1}$ діє рівень «1» з виходу вентиля $Y_{1\text{-}3}$.

Якщо вхідний рівень набуває значення 1, на виході вентиля $Y_{2\text{-}1}$ рівень «0» з'являється через відрізок часу, який дорівнює часу затримки поширення сигналу в одному вентилі.

Тим часом вхідний сигнал, проходячи через вентилі $Y_{1\text{-}1}$ – $Y_{1\text{-}3}$, зменшує потенціал на виході вентиля $Y_{1\text{-}3}$. При цьому на виході вентиля $Y_{2\text{-}1}$ формується негативний імпульс. Таким чином, використовуючи чотири вентилі, можна формувати імпульси з позитивних фронтів вхідного сигналу.

Розглянуту схему можна використовувати в лічильниках подій і як схему подвоєння частоти в цифрових системах. Детектор подій може бути складений і на інших логічних елементах чи на їх комбінаціях.

7.5 Інтегральні таймери

Таймером називають електронний пристрій, призначений для формування імпульсних сигналів з регульованими тривалістю і щілястістю. Це можуть бути і відповідні вузли цифрових пристроїв, і спеціалізовані ІС, які використовують для розроблення різних пристроїв, що задають час.

Усі існуючі на сьогодні таймери можна поділити на два класи: однотактні; багатотактні з вбудованим лічильником.

Однотактні таймери призначені для формування часових інтервалів тривалістю від одиниць мікросекунд до одиниць годин. Вони являють собою комбінацію аналогової частини (компаратора) з цифровою

послідовною схемою. Можливий варіант структурної схеми такого пристрою наведено на рис. 7.16.

Тривалість формованого таким пристроєм часового інтервалу визначається параметрами зовнішнього RC-кола. За активним значенням сигналу RS-тригер установлюється в одиничний стан, що приводить до розмикання перемикача S1.

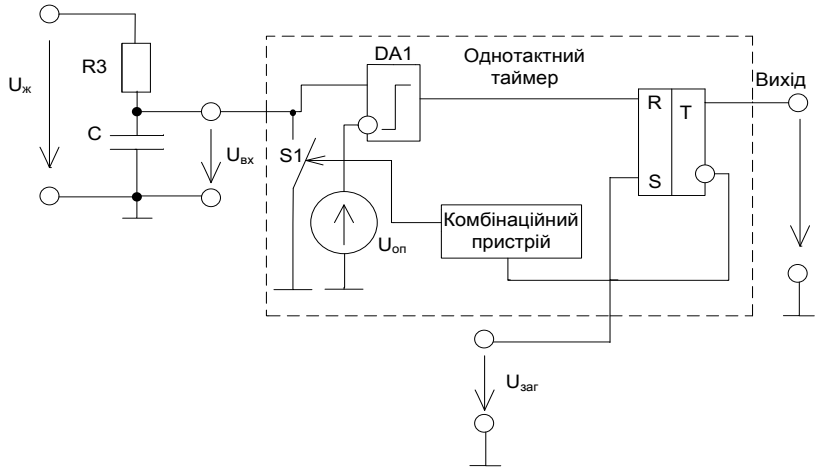


Рисунок 7.16 – Структурна схема однотактного таймера

Починається заряджання конденсатора С зовнішнього кола, який задає час. У момент, коли напруга на конденсаторі досягає рівня опорної напруги, відбувається спрацьовування компаратора DA1, і його вихідний сигнал скидає RS-тригер. Перемикач S при цьому замикається, і конденсатор С розряджається.

Однотактний таймер, побудований за описаною схемою, може формувати на виході тільки одиничні імпульси. Для забезпечення можливості формування послідовності імпульсів схему пристрою потрібно доповнити другим компаратором.

Багатотактні таймери з вбудованими лічильниками розроблені для формування імпульсів наднизької частоти з тривалістю імпульсу до кількох десятків годин. Їх можна поділити на дві підгрупи:

- програмувальні таймери, в яких часовий інтервал задається програмним способом. У найпростішому випадку це здійснюється установленням на виводах лічильника зовнішніх перемичок;
- спеціалізовані таймери, лічильник яких має жорстко заданий коефіцієнт перерахування.

Структурна схема багатотактного таймера (рис. 7.17) звичайно містить однотактний таймер і двійковий лічильник, спільну роботу яких формує додатковий логічний блок.

У багатотактному таймері фактично відбувається множення сталої часу зовнішнього RC-кола на модуль лічильника CT. Під час надходження сигналу запуску вмикається мультивібратор, виконаний на однотактному таймері. Його вихідні імпульси надходять на вхід лічильника.

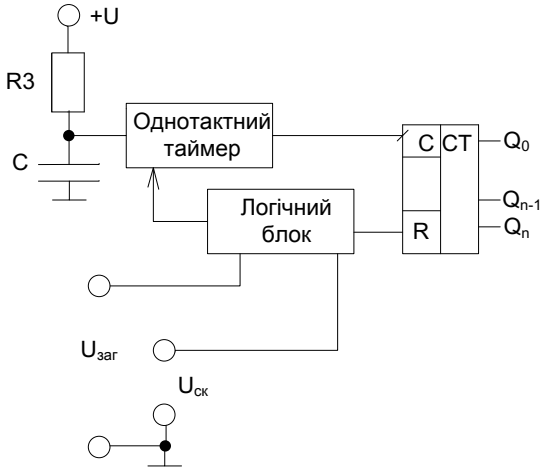


Рисунок 7.17 – Структурна схема багатотактного таймера

На виходах останнього може бути сформовано кілька послідовностей імпульсів з періодом від T_i до $(2N-1)T_i$, де T_i – період імпульсів, що знімаються з виходу однотактного таймера; N – кількість тригерів у лічильнику СТ. В табл. 7.2 подано режими роботи таймера.

Таблиця 7.2 – Режими роботи таймера

$U_{ск}$	$U_{пор.н}$	$U_{пор.в}$	$U_{вих}$	VT3
0	x	x	0	Насичений
1	$< U_ж/3$	$<2\ U_ж/3$	1	Замкнений
1	$> U_ж/3$	$>2\ U_ж/3$	0	Насичений
1	$> U_ж/3$	$>2\ U_ж/3$	Вихідний сигнал визначається попереднім значенням $U_{пор.н}$ та $U_{пор.в}$	

Контрольні запитання

1. *Що являють собою логічні розширники?*
2. *Наведіть приклад схеми логічного розширника.*
3. *Що являють собою перетворювачі рівнів, яка їх функція?*
4. *Наведіть приклад схеми перетворювача рівнів.*
5. *Що являють собою генератори?*
6. *Наведіть приклад схеми генератора.*
7. *Що являють собою одновібратори?*
8. *Наведіть приклад схеми одновібратора.*
9. *В чому різниця між генератором та одновібратором?*
10. *Що являють собою різницеві перетворювачі?*
11. *Наведіть приклад схеми різницевого перетворювача.*
12. *Що являють собою таймери?*
13. *Які Ви знаєте види таймерів?*

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Бабич Н. П. Основы цифровой схемотехники : учебное пособие / Н. П. Бабич, И. А. Жуков – М. : Издательский дом «Додэка-XXI», К. : «МК-Пресс», 2007. – 480 с.
2. Рябенький В. М. Цифрова схемотехніка : навч. посібник / Рябенький В. М., Жуйков В. Я., Гулий В. Д. – Львів : «Новий світ 2000», 2009. – 736 с.
3. Бойко В. І. Основи технічної електроніки : у 2-х кн. : підручник / Бойко В. І., Гуржій А. М., Жуйков В. Я. – К. : Вища школа, 2007. – Кн. 2 : Схемотехніка. – 2007. – 510 с.
4. Угрюмов Е. П. Цифровая схемотехника / Угрюмов Е. П. – СПб. : БХВ-Петербург, 2004. – 528 с.
5. Пухальский Г. И. Цифровые устройства : учебное пособие для ВТУЗов / Г. Пухальский, Т. Новосельцева. – СПб. : Политехника, 1996. – 885 с.
6. Зельдин Е. А. Цифровые интегральные микросхемы в информационной измерительной аппаратуре / Зельдин Е. А. – Л. : Энергоатомиздат, 1986. – 280 с.
7. Преснухин Л. Н. Расчет элементов цифровых устройств / Преснухин Л. Н., Воробьев Н. В., Шишкевич А. А. – М. : Высшая школа, 1991. – 526 с.
8. Зубчак В. И. Справочник по цифровой схемотехнике / Зубчак В. И. – К. : Техника, 1990. – 448 с.
9. Евреинова Э. В. Цифровая и вычислительная техника / Евреинова Э. В. – М. : Радио и связь, 1991. – 446 с.
10. Рицар Б. Є. Цифрова техніка / Рицар Б. Є. – К. : НМК, 1991. – 371 с.
11. Мальцев П. П. Цифровые интегральные микросхемы : справочник / П. П. Мальцев, Н. С. Долидзе. – М. : Радио и связь, 1994. – 240 с.
12. Зубчук В. И. Справочник по цифровой схемотехнике / Зубчук В. И., Сигорский В. П., Шкуро А. Н. – К. : Техніка, 1990. – 448 с.
13. Агаханян Т. М. Интегральные микросхемы : учеб. пособие для ВУЗов / Агаханян Т. М. – М. : Энергоатомиздат, 1983. – 464 с.

ПЕРЕЛІК СКОРОЧЕНЬ

DC (decoder) – Дешифратор – функціональний вузол комп'ютера, призначений для перетворення кожної комбінації вхідного двійкового коду в керувальний сигнал лише на одному із своїх виходів.

CD (coder) – Шифратор – функціональний вузол комп'ютера, призначений для перетворення вхідного *m*-розрядного унітарного коду у вихідний *n*-розрядний двійковий позиційний код.

MX – Мультиплексор – керований кодом перемикач декількох інформаційних входів до спільного виходу.

DM – Демультиплексор – це керований кодом перемикач інформаційного входу до одного з виходів.

SM – Суматор – комбінаційний логічний пристрій, призначений для виконання операції арифметичного додавання чисел у двійковому коді.

ЦІП – цифровий індикаторний пристрій.

ПКК – перетворювач код – код.

КА – керувальний автомат.

ЛП – логічний перетворювач.

ТГ – тактовий генератор.

ЗП – запам'ятовувальний пристрій.

ВІС – велика інтегральна схема.

НВІС – надвелика інтегральна схема.

ОЗП – оперативний запам'ятовувальний пристрій.

ПЗП – постійний запам'ятовувальний пристрій.

RAM (Random Access Memory) – інакше ОЗП.

ROM (Read Only Memory) – інакше ПЗП.

VRAM (Video RAM) – відео ОЗП.

SRAM (Static RAM) – статичний ОЗП.

DRAM (Dynamic RAM) – динамічний ОЗП.

PROM (Programmable ROM) – програмовані ПЗП.

EPROM (Erasable ROM) – багаторазово програмовані ПЗП.

FIFO (First In – First Out) – буфер, що працює за принципом «перший прийшов – перший вийшов».

LIFO (Last In – First Out) – буфер, що працює за принципом «останній прийшов – перший вийшов».

ЛЗП – лінії записування.

ЛЗЧ – лінії зчитування.

ЛЗЗ – лінії записування – зчитування.

БЛК – блок локального керування.

ПЛІС – програмовані логічні інтегральні схеми.

PAL, PLA – Programmable Array Logic, Programmable Logic Array.

PLD – Programmable Logic Devices.

GAL – Generic Array Logic.

CPLD – Complex PLD.

FPGA – Functional Programmable Generic Array.

Навчальне видання

Білинський Йосип Йосипович
Ратушний Павло Миколайович
Мельничук Андрій Олександрович

ЦИФРОВА СХЕМОТЕХНІКА
ЧАСТИНА 2
ЕЛЕКТРОННІ ПРИСТРОЇ І СИСТЕМИ

Навчальний посібник

Редактор Т. Старічек
Оригінал-макет підготовлено П. Ратушним

Підписано до друку 09.02.2017 р.
Формат 29,7×42¼. Папір офсетний.
Гарнітура Times New Roman.
Друк різографічний. Ум. друк. арк. 10,9.
Наклад 50 пр. Зам. № 2017-025.

Вінницький національний технічний університет,
навчально-методичний відділ ВНТУ.
21021, м. Вінниця, Хмельницьке шосе, 95,
ВНТУ, к. 2201.
Тел. (0432) 59-87-36.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.

Віддруковано у Вінницькому національному технічному університеті
в комп'ютерному інформаційно-видавничому центрі
21021, м. Вінниця, Хмельницьке шосе, 95,
ВНТУ, ГНК, к. 114.
Тел. (0432) 59-87-38.
publish.vntu.edu.ua; email: kivc.vntu@gmail.com.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.