МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання практичних занять з дисципліни

«Комп'ютерна схемотехніка»

для студентів усіх форм навчання за спеціальністю
123 «Комп'ютерна інженерія»

Затверджено
редакційно-видавничою радою
університету,
протокол № 3 від 06.10.2021 р.

Харків
НТУ «ХПІ»
2021

Методичні вказівки до виконання практичних занять з дисципліни «Комп’ютерна схемотехніка» для студентів усіх форм навчання за спеціальністю 123 «Комп’ютерна інженерія». / уклад.: А. М. Носик, В. В. Онищенко, А. В. Саткус – Харків: НТУ «ХПІ», 2021. – 110 с.

Укладачі: А. М. Носик
В. В. Онищенко
А. В. Статкус

Рецензент В. В. Усик

Кафедра мультимедійних інформаційних технологій і систем

ЗМІСТ

ВСТУП

Методичні вказівки до практичних занять з дисципліни «Комп'ютерна схемотехніка» для студентів усіх форм навчання за спеціальністю 123 «Комп'ютерна інженерія» підготовлені відповідно до робочої навчальної програми з дисципліни «Комп'ютерна схемотехніка», яка вивчається студентами.

Метою дисципліни є передача студентам знань, необхідних для самостійного аналізу та синтезу операційних вузлів і пристроїв на основі методів комп'ютерної технології, а також для побудови пристроїв ЕОМ з заданими характеристиками на базі інтегральних мікросхем (ІМС). Вивчення даної дисципліни дозволяє з єдиних схемо технічних позицій розглядати та вивчати весь спектр виробів обчислювальної техніки

Виконання практичних занять проводиться індивідуально кожним студентом.

У результаті виконання практичних занять студенти повинні:

– знати: класифікацію і призначення сучасних операційних елементів та пристроїв комп'ютерних систем, включаючи їхні характеристики, схемотехніку, параметри, еквівалентні схеми, математичні моделі для машинного аналізу, типові схеми підключення тощо.

– вміти: розроблювати функціональну та принципову схеми типового вузла комп'ютера (регістра, лічильника, шифратора, дешифратора, мультиплексора та інше) у необхідній елементній базі з оптимізацією прийнятих критеріїв ефективності (складності, швидкодії та інш.).

Методичні вказівки складаються з таких підрозділів, як:

– мета роботи;

– завдання до виконання практичних занять;

– порядок виконання роботи;

– варіанти індивідуальних завдань.

1 ОСНОВНІ ПОЛОЖЕННЯ ТА ОЗНАЧЕННЯ КОМП'ЮТЕРНОЇ ЛОГІКИ. АЛГЕБРА ПЕРЕМИКАЛЬНИХ ФУНКЦІЙ

Мета: Закріплення, поглиблення та опрацювання основних понять алгебри логіки.

Теоретичні відомості

Розглянемо основні закони, аксіоми і теореми алгебри логіки [1–3].

Деякі закони звичайної алгебри застосовні і до алгебри логіки. Наприклад:

Закон комутативності (закон переміщення):

- для множення $AB = BA$;
- для додавання $A + B = B + A$.

Закон асоціативності (сполучний закон):

- для множення $ABC = A(BC) = (AB)C$,
- для додавання $A + B + C = A + (B + C) = (A + B) + C$.

Закон дистрибутивності (розподільний закон) :

- множення стосовно додавання: $A(B + C) = AB + AC$;
- додавання стосовно множення: $A + BC = (A + B)(A + C)$.

Алгебра логіки має ряд специфічних аксіом і теорем, основні з яких необхідні для аналізу і синтезу логічних ланцюгів чи схем, наведені у таблиці 2.1.

Подвійність визначається як зміна всіх знаків операції ТА на знаки операції *АБО*, усіх знаків операції *АБО* на знаки операції *ТА*, усіх нулів на одиниці і всіх одиниць на нулі.

Подвійність є одною з основних властивостей алгебри логіки .

Закони де Моргана є ілюстрацією властивості подвійності та, як уже відзначалося, можуть бути сформульовані у виді:

$\overline{ABC} = \overline{A} \vee \overline{B} \vee \overline{C}$, $\overline{A \vee B \vee C} = \overline{A}\,\overline{B}\,\overline{C}$,

З законів де Моргана випливають наслідки:

$ABC = \overline{\overline{A} \vee \overline{B} \vee \overline{C}}$ $A \vee B \vee C = \overline{\overline{A}\,\overline{B}\,\overline{C}}$.

Таблиця 2.1 – Основні аксіоми алгебри логіки

№	Назва закону	Формулювання
1.	Комутативності	$A \vee B = B \vee A$, $A \vee B = B \vee A$.
2.	Асоціативності	$(A \vee B) \vee C = (C \vee A) \vee B = A \vee B \vee C$, $(AB)C = A(BC) = (AC)B = ABC$
3.	Дистрибутивності	$A(B \vee C) = AB \vee BC$, $A \vee BC = (A \vee B)(A \vee C)$
4.	Ідемпотентності	$A \vee A = A$, $A \cdot A = A$
5.	Де Моргана	$\overline{A \vee B} = \overline{A}\,\overline{B}$, $\overline{AB} = \overline{A} \vee \overline{B}$
6.	Подвійного заперечення	$\overline{\overline{A}} = A$
7.	Подвійності	$A \vee \overline{A} = 1$, $A \cdot \overline{A} = 0$
8.	Пустої множини	$A \vee 0 = A$, $A \cdot 0 = 0$
9.	Універсальної множини	$A \vee 1 = 1$, $A \cdot 1 = A$
10.	Склеювання	$AB \vee A\overline{B} = A$, $(A \vee B)(A \vee \overline{B}) = A$
11.	Поглинання	$A(A + B) = A$, $A + AB = A$
12.	Наслідок з третього закону	$A \vee \overline{A}B = A \vee B$

Отже, функція додавання за модулем 2 представляється в такий спосіб:

$A \oplus \overline{B} = \overline{A}B \vee AB = (\overline{A} \vee \overline{B})(A \vee B)$.

Для цієї функції справедливі такі аксіоми:

$A \oplus A = 0$; $A \oplus A \oplus A = A$; $A \oplus \overline{A} = 1$; $A \oplus 1 = \overline{A}$; $A \oplus 0 = A$.

Функція $f(x_1, x_2, \ldots, x_n)$ називається *логічною* (*перемикальною*, *булевою*), якщо вона, як і її аргументи X_I, може приймати тільки два значення: 0 чи 1 (хибне чи істинне).

Логічна (булева) змінна – це така величина X, що може приймати тільки два значення: 0 чи 1.

Таким чином, логічні функції, їхні аргументи і просто логічні змінні можуть приймати тільки два значення 0 чи 1. Причому, у цих випадках цифри 0 і 1 є *символами стану*, а *не числами*. Алгебра логіки є алгеброю станів, а не

алгеброю чисел, тому цю алгебру називають також алгеброю висловлювань (тверджень).

Висловлюванням називається твердження, про яке можна чітко сказати, *істинне* воно чи *хибне***.** Якщо висловлювання *істинне,* то говорять, що його значення істинності дорівнює одиниці. Якщо ж висловлювання *хибне,* – його значення істинності дорівнює нулю. *Висловлювань одночасно істинних та хибних не буває.*

Висловлювання бувають простими і складними. *Прості окремі висловлювання* – це логічні змінні, їх прийнято позначати буквами латинського алфавіту. Наприклад, якщо просте висловлювання X істинне, то $X = 1$, якщо ж хибне, то $X = 0$.

Сукупність значень аргументів логічної функції називається *набором (або точкою)* і може позначатися зокрема, як x_1**,** x_2,..., x_n**,** де x_i дорівнює нулю чи одиниці. Очевидно, що набір значень аргументів фактично являє собою деяке двійкове число. Кожному набору значень аргументів приписується номер, який дорівнює двійковому числу, що відповідає значенню даного набору. Наприклад, для чотирьох аргументів: 0, 0, 0, 0 – нульовий набір; 0, 0, 0, 1 – перший набір; 0, 0, 1, 0 – другий набір; 1, 0, 1, 0 – десятий набір і т.д.

Логічна функція (функція алгебри логіки – ФАЛ) – це функція $f(x_1, x_2, x_n)$, яка приймає значення 0 чи 1 на наборі логічних змінних x_i, x_2, …, x_n. Кожний логічній функції даного набору аргументів також прийнято приписувати номер: 0, 1, 2,....

Будь–яка логічна функція n аргументів визначена на 2^n наборах, тобто може мати 2^n наборів. Число різних логічних функцій n аргументів кінцеве і дорівнює $\mathbf{2^{2^n}}$**.**

Неповністю визначена *логічна функція* **n** *змінних – це функція, задана на числі наборів меншому* 2^n*. Для наборів, на яких функція не визначена, її значення можна прийняти довільним.*

Функції одного і двох аргументів називаються елементарними, якщо логічні вирази цих функцій містять не більше однієї логічної операції, що є елементарними.

Наприклад, для одного аргументу маємо $2^1 = 2$ набори і $2^2 = 4$ елементарні логічні функції від однієї змінної, які наведені в таблиці 2.2.

Таблиця 2.2

x	$f_0(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$
0	1	0	0	1
1	1	0	1	0

У зв'язку з тим, що функція $f_0(x)$ дорівнює одиниці при будь-яких значеннях аргументу, ця функція називається абсолютно істинною *(константа одиниці)*, тоді функція $f_1(x)$ – абсолютно хибна функція *(константа нуля)*. Функція $f_2(x)$, що повторює абсолютне значення логічної змінної – *тотожна функція:* $f_2(x) = x$.

Функція $f_3(x)$, що приймає значення, зворотне значенню x, – *логічне заперечення* (інверсія), або функція НЕ (NOT), що позначається одним з наступних способів:

$$f_3(\overline{x}) = \overline{x} = x' = \hat{}x.$$

Логічне заперечення, чи *інверсія*, деякої логічної змінної, наприклад змінної A, це також логічна змінна, яка приймає значення, зворотне значенню змінної A, що позначається як $\overline{A}$. Якщо A =1, то $\overline{A}$=0, якщо ж A = 0, то $\overline{A}$ = 1. Але потрібно врахувати, що часто за допомогою заперечення, тобто інверсії змінної, просто позначають випадок, коли її значення дорівнює нулю.

У таблиці 2.3 наведений повний перелік усіх *шістнадцяти* елементарних логічних функцій від двох змінних x_1 і x_2.

Розглянемо деякі важливі функції.

Диз'юнкція (логічне додавання) або *функція* АБО (ИЛИ, OR) – це функція $f_7(x_1,x_2)$, що істинна тоді, коли істинна хоча б одна з її змінних. Умовне позначення цієї функції:

$f_7(x_1, x_2) = x_1 + x_2 = x_1 \vee x_2$.

Це читається у такий спосіб: функція істинна, тобто дорівнює одиниці, коли істинний аргумент x_1= 1 або аргумент x_2 = 1, або ж обидва аргументи

істинні одночасно.

Таблиця 2.3

Функція	x_1x_2				Примітка	
	00	01	10	11		
f_0	0	0	0	0	$f(x_1x_2)$=0	константа нуль
f_1	0	0	0	1	$x_1 \wedge x_2$	кон'юнкція
f_2	0	0	1	0	$x_1 \overline{x_2}$	заборона по x_2
f_3	0	0	1	1	x_1	змінна x_1
f_4	0	1	0	0	$\overline{x_1}\, x_2$	заборона по x_1
f_5	0	1	0	1	x_2	змінна x_2
f_6	0	1	1	0	$x_1 \oplus \boldsymbol{x_2}$	додавання за модулем 2
f_7	0	1	1	1	$x_1 \vee \boldsymbol{x_2}$	диз'юнкція
f_8	1	0	0	0	$x_1 \downarrow x_2$	функція Пірса
f_9	1	0	0	1	$x_1 \sim x_2$	рівнозначність
f_{10}	1	0	1	0	$\overline{x_2}$	інверсія x_2
f_{11}	1	0	1	1	$x_2 \longrightarrow x_1$	імплікація
f_{12}	1	1	0	0	$\overline{x_1}$	інверсія x_1
f_{13}	1	1	0	1	$x_1 \longrightarrow x_2$	імплікація
f_{14}	1	1	1	0	x_1 / x_2	функція Шеффера
f_{15}	1	1	1	1	f(x_1 x_2)=1	константа одиниця

Кон'юнкція (логічне множення) або функція ТА (И, AND) – це функція $f_1(x_1,x_2)$, яка істинна тоді, коли *всі її змінні одночасно істинні***.** Цю функцію умовно позначають у такий спосіб:

$f_1(x_1, x_2) = x_1 \wedge x_2 = x_1 \bullet x_2 = x_1 \,\&\, x_2$ **.**

Це читається так: функція істинна, тобто дорівнює одиниці, коли обидва аргументи одночасно істинні, тобто дорівнюють одиниці.

Функція (штрих) Шеффера або функція ТА–НЕ – це функція $f_{14}(x_1, x_2)$, яка

хибна тоді, коли всі змінні істинні. Умовне позначення цієї функції:

$f_{14}(x_1,x_2) = x_1/x_2 = \overline{\boldsymbol{x_1}} \vee \overline{\boldsymbol{x_2}} = \overline{\boldsymbol{x_1} \wedge \boldsymbol{x_2}}$

Це читається у такий спосіб: функція хибна, тобто дорівнює **0**, коли обидва аргументи одночасно істинні, тобто дорівнюють одиниці, і функція істинна, тобто дорівнює одиниці, коли обидва аргументи одночасно хибні або ж хоча б один з них хибний.

Функція (стрілка) Пірса (Вебба) або функція АБО–НЕ – це функція $f_8(x_1,x_2)$, яка істинна тільки тоді, коли всі змінні хибні. Умовне позначення цієї функції:

$f_8(x_1, x_2) = x_1 \downarrow x_2 = \overline{\boldsymbol{x_1} \vee \boldsymbol{x_2}}$.

Це читається так: функція хибна, тобто дорівнює 0, коли хоча б один з її аргументів істинний чи обидва аргументи одночасно істинні, тобто дорівнюють одиниці, і функція істинна, тобто дорівнює одиниці, коли обидва аргументи одночасно хибні.

Функція імплікації або функція ЯКЩО–ТО (IF–THEN) це функція $f_{13}(x_1, x_2)$, яка помилкова тоді і тільки тоді, коли x_1 істинне та x_2 помилкове. Аргумент x_1 називається *посилкою*, а x_2 – *наслідком*. Її умовне позначення:

$f_{13}(x_1,x_2) = x_1 \longrightarrow x_2 = \overline{\boldsymbol{x_1}} \vee x_2$.

Виключаюче АБО (XOR) – це функція $f_6(x_1, x_2)$, що позначається знаком $\oplus$. Ця операція, як видно з таблиці, реалізує функцію *нерівнозначності*, тобто фактично реалізується процедура *підсумовування за модулем* 2:

$f_6(x_1,x_2) = x_1 \oplus x_2$.

Всі розглянуті функції є елементарними.

Як буде показано далі, будь–які логічні операції над логічними змінними можна звести до визначеної сукупності елементарних логічних функцій, наприклад, таких як ТА, АБО, НЕ та Виключаюче АБО.

Елементарні логічні операції в комп'ютерах виконуються також над двійковими числами, *порозрядно*. Розглянемо кілька простих прикладів виконання логічних операцій над двома двійковими числами (рис. 2.1).

ТА	АБО	Виключне АБО	НЕ
01011010	0101010	0101010	01011010
11110000	1111000	1111000	10100101
01010000	1111010	1010010	

Рисунок 2.1 – Приклади елементарних логічних операції

Розглянемо основні закони, аксіоми і теореми алгебри логіки

Приклад 1. Спростити вираз $y = x_1 \cdot x_2 \cdot x_3 \vee x_1 \cdot \overline{x_2} \cdot x_3 \vee x_1 \cdot x_3$

Використовуючи закон склеювання, отримаємо:

$$y = x_1 \cdot x_2 \cdot (x_2 \vee \overline{x_2}) \vee x_1 \cdot x_3 = x_1 \cdot x_3 \vee x_1 \cdot x_3 = x_1 \cdot x_3$$

Приклад 2. Спростити вираз:

$$N = \overline{\overline{(x \cdot \bar{y} \vee \bar{x} \cdot y) \cdot x} \vee (x \cdot \bar{y} \vee \bar{x} \cdot y) \cdot x}.$$

Використовуючи закон де Моргана отримаємо:

$$N = \overline{[(x \cdot \bar{y} \vee \bar{x} \cdot y) \cdot x]} \cdot \overline{[(x \cdot \bar{y} \vee \bar{x} \cdot y) \cdot x]}.$$

Використовуючи закон де Моргана отримаємо:

$$N = [\overline{(x \cdot \bar{y} \vee \bar{x} \cdot y)} \vee \bar{x}] \cdot [\overline{(x \cdot \bar{y} \vee \bar{x} \cdot y)} \vee \bar{x}].$$

Використовуючи закон ідемпотентності до першої круглої дужки і закон де Моргана до другої круглої дужки, отримамо

$$N = (x \cdot \bar{y} \vee \bar{x} \cdot y \vee \bar{x}) \cdot [(\overline{x \cdot \bar{y}} \cdot \overline{\bar{x} \cdot y}) \vee \bar{x}].$$

Послідовно використовуючи закон поглинання і закон спрощення, а до виразів з запереченями – закон де Моргана, отримаємо:

$$N = (\bar{y} \vee \bar{x}) \cdot [(\bar{x} \vee y) \cdot (x \vee \bar{y}) \vee \bar{x}].$$

Після перетворень отримаємо

$$N = (\bar{y} \vee \bar{x}) \cdot (\bar{x} \cdot \bar{y} \vee x \cdot y) \vee \bar{x}),$$

і у подальшому, використовуючи закон поглинання і закон спрощення, отримаємо

$$N = (\bar{y} \vee \bar{x}) \cdot (\bar{x} \vee y) = \bar{y} \cdot \bar{x} \vee \bar{x} \cdot y = \bar{x}.$$

Насправді мінімальну форму для цього виразу можливо отримати простіше.

Позначемо буквою $a = (x \cdot \bar{y} \vee \bar{x} \cdot y) \cdot$

Тоді отримаємо:

$$N = \overline{\overline{(x \cdot \bar{y} \vee \bar{x} \cdot y)} \cdot x \vee (x \cdot \bar{y} \vee \bar{x} \cdot y) \cdot x} = \overline{\bar{a} \cdot x \vee a \cdot x} = \overline{x \cdot (\bar{a} \cdot \vee a)} = \bar{x}.$$

Приклад 3. Спростити вираз:

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

Використовуючи закони булевої алгебри, отримаємо наступне рішення:

$$N = \overline{\overline{x} \cdot \overline{y} \cdot \overline{z} \vee \ \overline{x} \cdot y \cdot \overline{z} \ \vee \overline{x} \cdot \overline{y} \ \vee \overline{x} \vee \overline{z}} = \overline{\overline{x} \cdot \overline{z} \cdot (\overline{y} \vee y) \vee \overline{x} \cdot (\overline{y} \ \vee 1) \vee \overline{z}} =$$

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

Приклад 4. Спростити вираз:

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

Використовуючи закони булевої алгебри, отримаємо наступне рішення:

$$N = \overline{x \cdot \overline{y} \vee z \cdot x \cdot \overline{z} \ \vee x \cdot z \ \vee \overline{x} \ \cdot \ \overline{y}} = \overline{\overline{y} \cdot (\overline{x} \vee x) \vee z \vee x \cdot (z \ \vee \overline{z})} =$$

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

Приклад 5. Спростити вираз:

$$N = \overline{\overline{x} \cdot \overline{z} \vee \overline{z}} \vee \overline{\overline{\overline{z}} \cdot z} \vee x \cdot y \cdot z \cdot \ \overline{x \cdot y} \vee \overline{(\overline{x} \cdot y \cdot z \vee \overline{(x \cdot \overline{y \cdot z} \vee x \cdot \overline{y} \vee \overline{z})}}.$$

Використовуючи закони булевої алгебри, отримаємо наступне рішення:

$$N = \overline{\overline{x} \cdot \overline{z} \vee \overline{z}} \vee \overline{\overline{\overline{z}} \cdot z} \vee x \cdot y \cdot z \cdot \ \overline{x \cdot y} \vee \overline{(\overline{x} \cdot y \cdot z \vee \overline{(x \cdot \overline{y \cdot z} \vee x \cdot \overline{y} \vee \overline{z})}} =$$

$$= \overline{\overline{z} \cdot (x \vee 1)} \vee \overline{z} \vee x \cdot y \cdot z \cdot (\overline{x} \vee \overline{y}) \ \vee \overline{(\overline{x} \cdot y \cdot z \vee \overline{(x \cdot \overline{y \cdot z} \vee x \cdot \overline{y} \vee \overline{z})}} =$$

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

$$= 1 \vee 0 \vee 0 \vee \overline{\overline{x} \cdot y \cdot z \vee x \cdot \overline{y} \vee \overline{z}} = 1.$$

Приклад 6 . Побудувати таблицю істинності для функції *f,* заданої аналітичним виразом: $f = \overline{x} y \vee \overline{x \vee y} \vee x.$

Складемо таблицю істинності для формули $\overline{x}$ y∨ $\overline{x \vee y}$ ∨x яка має дві змінні *x* та *y*. В перших двох стовбцях таблиці запишемо чотири можливих пари значень цих змінних, в наступних стовбцях — значення проміжних формул та в останньому стовбці — значення формули. В результаті отримаємо таблицю 2.4:

Таблиця 2.4 – Таблиця істинності для функції *f*

Проміжні		Проміжні логічні формули					*f*
x	*y*	$\overline{x}$	$\overline{x}$y	$x \vee y$	$\overline{x \vee y}$	$\overline{x}$y∨ $\overline{x \vee y}$	$\overline{x}$y∨ $\overline{x \vee y}$∨x
0	0	1	0	0	1	1	1
0	1	1	1	1	0	1	1
1	0	0	0	1	0	0	1
1	1	0	0	1	0	0	1

З таблиці видно, що при всіх наборах значень змінних *x* та y формула $\overline{x}$y ∨ $\overline{x \vee y}$ ∨*x* приймає значення 1, тобто є тождественно істиною.

Приклад 7. Побудувати таблицю істинності для функції *f*, заданої аналітичним виразом: $f = \overline{x \vee y} \cdot (x \cdot \overline{y})$.

Аналогічно до прикладу 6 побудуємо таблицю істинності (див. табл. 2.4).

З таблиці 2.5 видно, що при усіх наборах значень змінних *x* та y формула $\overline{x \vee y \cdot} (x \cdot \bar{y})$ приймає значення 0, тобто є тотожно хибною.

Таблиця 2.5 – Таблиця істинності для формули $\overline{x \vee y} \cdot (x \cdot \overline{y})$

Проміжні		Проміжні логічні формули				*f*
x	*y*	$x \vee y$	$\overline{x \vee y}$	$\overline{y}$	$x \cdot \overline{y}$	$\overline{x \vee y} \cdot (x \cdot \overline{y})$
0	0	0	1	1	0	0
0	1	1	0	0	0	0
1	0	1	0	1	1	0
1	1	1	0	0	0	0

Приклад 8. Побудувати таблицю істинності для функції *f*, заданої аналітичним виразом: $f = \overline{\text{x} \vee \overline{y}} \vee \overline{x} \cdot z$.

З таблиці видно, що формула $\overline{x \vee \overline{y} \vee \overline{x}} \cdot z$ в деяких випадках приймає значення 1, а в деяких — 0, тобто є виконаємою.

Таблиця 2.6 – Таблиця істинності для формули $\overline{x \vee \overline{y} \vee \overline{x}} \cdot z$

Проміжні			Проміжні логічні формули					Формула
x	*y*	*z*	$\cdot\overline{y}$	$x \vee \overline{y}$	$\overline{x \vee \overline{y}}$	$\overline{x}$	$\overline{x} \cdot z$	$\overline{x \vee \overline{y} \vee \overline{x}} \cdot z$
0	0	0	1	1	0	1	0	0
0	0	1	1	1	0	1	1	1
0	1	0	0	0	1	1	0	1
0	1	1	0	0	1	1	1	1
1	0	0	1	1	0	0	0	0
1	0	1	1	1	0	0	0	0
1	1	0	0	1	0	0	0	0
1	1	1	0	1	0	0	0	0

Приклад 9. Побудувати таблицю істинності для функції *f,* заданої аналітичним виразом: $f = \overline{(x1 \vee x2 \cdot \overline{x3})} \cdot (x1 \vee x3)$ для всіх наборів значень змінних, від яких вона залежить (табл. 2.7). Використовуючи основні закони булевої алгебри, виконати перетворення функції таким чином. Щоб звільнити від загальної інверсії частини виразу $\overline{(x1 \vee x2 \cdot \overline{x3})}$. Побудувати таблицю істинності для отриманої функції для всіх наборів значень змінних, від яких вона залежить (табл. 2.8).

Таблиця 2.7 – Початкова таблиця істинності функції *f.*

$x1$	$x2$	$x3$	$a = x1 \vee x3$	$\overline{x3}$	$b = x2 \cdot \overline{x3)}$	$c = x1 \vee b$	$\overline{c}$	$f = \overline{c} \cdot a$
0	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	1	1
0	1	0	0	1	1	1	0	0
0	1	1	1	0	0	0	1	1
1	0	0	1	1	0	1	0	0
1	0	1	1	0	0	1	0	0
1	1	0	1	1	1	1	0	0
1	1	1	1	1	0	1	0	0

Виконуємо перетворення заданого виразу:

$$f = \overline{(x1 \vee x2 \cdot \overline{x3})} \cdot (x1 \vee x3) = \left(\overline{x1} \cdot \overline{(x2 \cdot \overline{x3})}\right) \cdot (x1 \vee x3) = \overline{x1} \cdot (\overline{x2} \vee x3) \cdot (x1 \vee x3)$$

Таблиця істинності для отриманого виразу (табл. 2.8) буде такою:

Таблиця 2.8 – Остаточна таблиця істинності функції f

$x1$	$x2$	$x3$	$\overline{x2}$	$m = \overline{x2} \vee x3$	$\overline{x1}$	$n = \overline{x1} \cdot m$	$k = x1 \vee x3$	$f = n \cdot k$
0	0	0	1	1	1	1	0	0
0	0	1	0	1	1	1	1	1
0	1	0	1	0	1	0	0	0
0	1	1	0	1	1	1	1	1
1	0	0	1	1	0	0	1	0
1	0	1	0	1	0	0	1	0
1	1	0	1	0	0	0	1	0
1	1	1	1	1	0	0	1	0

Значення в стовпчиках f в обох таблицях співпадають. Отже виконані перетворення заданого виразу еквівалентні.

Завдання до практичної роботи.

Варіант 1

1. Спростіть вираз:

$$y = \overline{x1} \cdot x2 \cdot x3 \vee x1 \cdot \overline{x2} \cdot \overline{\overline{x3}} \cdot x3 \vee x1 \cdot x3 \vee x1 \cdot \overline{x2} \cdot x3.$$

2. Розвязати логічну задачу за допомогою таблиць істинності

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

Варіант 2

1. Спростіть вираз:

$$y = x1 \cdot x2 \cdot x3 \vee x1 \cdot \overline{x2} \cdot x3 \vee x1 \cdot x3 \vee x1 \cdot \overline{x2} \cdot x3.$$

2. Розвязати логічну задачу за допомогою таблиць істинності

$$\overline{x \vee y} \cdot (x \cdot \overline{y}).$$

Варіант 3

1. Спростіть вираз:

$$y = x1 \cdot x2 \cdot x3 \vee x1 \cdot \overline{x2} \vee x1 \cdot x2 \cdot x3 \vee x1 \cdot \overline{x2} \cdot x3.$$

2. Розвязати логічну задачу за допомогою таблиць істинності

$$\overline{x \vee y} \cdot (x \cdot \overline{z}).$$

Варіант 4

1. Спростіть вираз:

$$y = \overline{x1 \cdot x2 \cdot \overline{x3} \vee x1 \cdot x2 \cdot x2 \cdot x3 \vee x1 \cdot x3 \vee x1 \cdot \overline{\overline{x2}} \cdot x3 \vee \overline{\overline{x2}} x3 \vee \overline{\overline{x1}} \vee \overline{x1} \cdot x2 \cdot x3}.$$

2. Розвязати логічну задачу за допомогою таблиць істинності

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

Варіант 5

1. Спростіть вираз:

$$y = \overline{x1 \cdot x2 \cdot \overline{\overline{x1}} \cdot x2 \cdot \overline{x3} \vee x2 \cdot \overline{x3} \vee \overline{x1} \cdot x2 \cdot \overline{x3} \vee \overline{x2} \vee x3 \vee x1 \vee x1 \cdot x2 \cdot x3 \cdot \overline{\overline{x3}}}.$$

2. Розвязати логічну задачу за допомогою таблиць істинності

$$(x \vee y) \cdot \overline{(x} \vee y) \cdot (\overline{x} \vee \overline{y})$$

Варіант 6

1. Спростіть вираз:

$$y = \overline{x1} \cdot x2 \cdot x3 \vee x1 \cdot \overline{x2} \cdot \overline{\overline{x3}} \cdot x3 \vee x1 \cdot x3 \vee x1 \cdot \overline{x2} \cdot x3$$

2. Розвязати логічну задачу за допомогою таблиць істинності

$$(x \cdot \overline{y}) \cdot \overline{(x} \cdot y \cdot z) \vee (x \cdot z).$$

Варіант 7

1. Спростіть вираз:

$$y = x1 \cdot x2 \cdot x3 \vee x1 \cdot \overline{x2} \cdot x3 \vee x1 \cdot x3 \vee x1 \cdot \overline{x2} \cdot x3.$$

2. Розвязати логічну задачу за допомогою таблиць істинності

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

Варіант 8

1. Спростіть вираз:

$$y = x1 \cdot x2 \cdot x3 \vee x1 \cdot \overline{x2} \vee x1 \cdot x2 \cdot x3 \vee x1 \cdot \overline{x2} \cdot x3.$$

2. Розвязати логічну задачу за допомогою таблиць істинності

$$(y \cdot \overline{z}) \vee \overline{(y} \cdot x \cdot z) \vee (x \cdot z).$$

Варіант 9

1. Спростіть вираз:

$$y = \overline{x1 \cdot x2 \cdot \overline{x3} \vee x1 \cdot x2 \cdot x2 \cdot x3 \vee x1 \cdot x3 \vee x1 \cdot \overline{\overline{x2}} \cdot x3 \vee \overline{\overline{x2}} x3 \vee \overline{\overline{x1} \vee \overline{x1} \cdot x2 \cdot x3}}.$$

2. Розвязати логічну задачу за допомогою таблиць істинності

$$(x \vee z)(\overline{x} \cdot z)(\overline{x} \vee \overline{y}).$$

Варіант 10

1. Спростіть вираз:

$$y = \overline{x1 \cdot x2 \cdot \overline{\overline{x1}} \cdot x2 \cdot \overline{x3} \vee x2 \cdot \overline{x3} \vee \overline{x1} \cdot x2 \cdot \overline{x3} \vee \overline{x2} \vee \overline{x3 \vee x1 \vee x1 \cdot x2 \cdot x3 \cdot \overline{\overline{x3}}}}$$

2. Розвязати логічну задачу за допомогою таблиць істинності

$$\overline{x} \cdot z \vee \overline{x \vee y} \vee x).$$

Варіант 11

1. Спростіть вираз:

$$y = \overline{x1} \cdot x2 \cdot x3 \vee x1 \cdot \overline{x2} \cdot \overline{\overline{x3}} \cdot x3 \vee x1 \cdot x3 \vee x1 \cdot \overline{x2} \cdot x3.$$

2. Розвязати логічну задачу за допомогою таблиць істинності

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

Варіант 12

1. Спростіть вираз:

$$y = x1 \cdot x2 \cdot x3 \vee x1 \cdot \overline{x2} \cdot x3 \vee x1 \cdot x3 \vee x1 \cdot \overline{x2} \cdot x3$$

2. Розвязати логічну задачу за допомогою таблиць істинності

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

Варіант 13

1. Спростіть вираз:

$$y = \overline{x1 \cdot x2 \cdot \overline{\overline{x1}} \cdot x2 \cdot \overline{x3} \vee x2 \cdot \overline{x3} \vee \overline{x1} \cdot x2 \cdot \overline{x3} \vee \overline{x2} \vee \overline{x3 \vee x1 \vee x1 \cdot x2 \cdot x3 \cdot \overline{\overline{x3}}}}.$$

2. Розвязати логічну задачу за допомогою таблиць істинності

$$(x \cdot \overline{y} \vee z)(\overline{x} \vee y) \vee \overline{z}.$$

Варіант 14

1. Спростіть вираз:

$$y = \overline{x1 \cdot x2 \cdot \overline{x3} \vee x1 \cdot x2 \cdot x2 \cdot x3 \vee x1 \cdot x3 \vee x1 \cdot \overline{\overline{x2}} \cdot x3 \vee \overline{\overline{x2}} x3 \vee \overline{\overline{x1} \vee \overline{x1} \cdot x2 \cdot x3}}.$$

2. Розвязати логічну задачу за допомогою таблиць істинності

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

Варіант 15

1. Спростіть вираз:

$$y = \overline{x1} \cdot x2 \cdot x3 \vee x1 \cdot \overline{x2} \cdot \overline{\overline{x3}} \cdot x3 \vee x1 \cdot x3 \vee x1 \cdot \overline{x2} \cdot x3.$$

2. Розвязати логічну задачу за допомогою таблиць істинності

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

Контрольні запитання

1. Які змінні та функції являються булевими?
2. Які булеві операції використовуються в булевих виразах?
3. Охарактеризуйте різні способи подання булевих функцій.
4. Наведіть приклад схемного відтворення булевої функції.
5. Перелічіть найменування булевих функцій однієї і двох змінних.
6. Які основні закони алгебри логіки ви знаєте?
7. Продемонструйте використання законів алгебри логіки для спрощення логічних виразів.

2 МЕТОДИ МІНІМІЗАЦІЇ ПЕРЕМИКАЛЬНИХ ФУНКЦІЙ

Мета: закріпити теоретичні знання, отримані при вивченні методів мінімізації перемикальних функцій.

Теоретичні відомості

Під мінімізацією будемо розуміти процес знаходження такого еквівалентного вираження функції алгебри логіки, яке містить мінімальну кількість входжень змінних (букв в вираженні). Хоча, в загальному випадку, під мінімізацією може матися на увазі отримання виразів з мінімальним числом інверсних змінних або з мінімальним числом входжень будь-якої однієї змінної і т.п. Більшість методів мінімізації орієнтовані на отримання МДНФ (МКНФ), проте доведено, що мінімальне вираження в класі ДНФ буде або мінімальним, або відрізнятися від мінімального на якийсь число входжень змінної в класі інших форм функції [1, 2, 4].

Наведемо деякі визначення.

Елементарне множення – множення будь-якого числа букв (змінних), взятих з запереченням або без.

ДНФ (діз'юнктівная нормальна форма) – диз'юнкція елементарних множників.

Минтерм (констітуента1) – множення всіх змінних, взятих з запереченням або без, на яких функція приймає значення 1.

СДНФ (досконала ДНФ) – диз'юнкція всіх мінтермов.

Імпликанта – елементарне множення, яке покриває принаймні один з мінтермів булевої функції, але не призводить до нових (непотрібним) канонічним сумам елементарних множників.

Інше визначення імпліканти.

Імплікантой називається елементарне множення, рівне 1 на одному або декількох наборах, де дана функція дорівнює 1, і дорівнює 0 на всіх наборах, де

дана функція дорівнює 0. Імпликанта покриває один або кілька мінтермів розглянутої булевої функції. Зазвичай, імпликанта – це результат склеювання відповідних мінтермів або імплікант.

Проста імпліканта – імпліканта, яка містить мінтерм функції, але перестає бути імпликантою після видалення будь-якого аргументу (іншими словами, це імпліканта, до якої не можна застосувати операцію склеювання).

Скорочена ДНФ – диз'юнкція усіх простих імплікант.

Істотна імпліканта – проста імпліканта, утворена склеюванням таких мінтермів, що принаймні для одного з них ця операція була єдиною

Істотні імпліканти утворюють ядро функції.

Тупикова ДНФ – диз'юнкція простих імплікант, з яких жодна не є зайвою.

Найкоротша ДНФ – тупикова ДНФ, що має мінімальне число простих імплікант.

МДНФ (мінімальна ДНФ) – тупикова ДНФ з мінімальним числом входжень змінних (мінімальним числом букв) в порівнянні з іншими тупиковими формами цієї функції.

Елементарна сума, КНФ, макстерм (констітуенти 0), СКНФ, імпліцента, проста імплицента, скорочена КНФ, істотна імпліцента, тупікова КНФ, найкоротша КНФ, МКНФ визначається аналогічно.

Елементарна сума – диз'юнкція будь-якого числа букв (змінних), взятих з запереченням або без.

КНФ (кон'юнктивна нормальна форма) – кон'юнкція елементарних диз'юнкцій.

Макстерм (констітуенти 0) – диз'юнкція усіх змінних, взятих з запереченням або без, на яких функція приймає значення 0.

СКНФ (досконала КНФ) – кон'юнкція всіх макстермів.

Імпліцент – результат склеювання макстермів або імпліцент.

Мінімізація логічних функцій методом Вейча.

Розглянемо візуальний метод мінімізації булевих функцій за допомогою карт (діаграм) Вейча, який є одним з найбільш зручних методів при невеликому

числі змінних.

Мінімізацію логічних функцій методом Вейча застосовують для функцій $f(x_1, x_2,..., x_n)$, які містять, як правило, не більше чотирьох змінних і повинні бути задані в аналітичній формі у вигляді ДДНФ або ДКНФ. Однак, цей метод може застосовуватись і для більшого числа змінних.

У методі Вейча для мінімізації використовують таблиці, які являють собою прямокутник, що вміщує *n* клітинок, до яких заносять одиниці при мінімізації логічних функцій, які задані в ДДНФ або нулі, у випадку мінімізації логічних функцій, поданих у ДКНФ. Якщо функція записана в ДНФ або КНФ, то її слід попередньо перевести до ДДНФ або ДКНФ.

Структури таблиць Вейча для двох, трьох і чотирьох логічних змінних приведені на рис. 3.1.

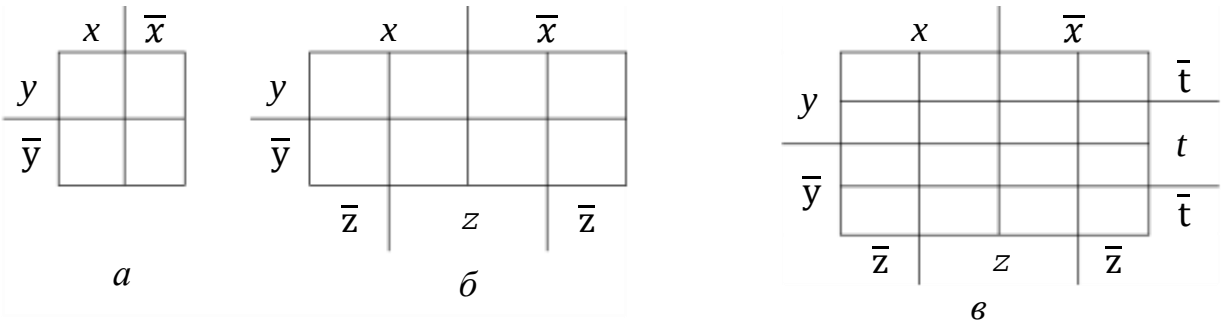


Рисунок 3.1 – Структури таблиць Вейча

Як випливає із рис. 3.1, структура таблиці для двох змінних має вид таблиці 2x2, для трьох — 2x4, а для чотирьох — 4x4.

Із таблиць випливає, що кожна змінна і її заперечення містяться з одного боку таблиці. За такого розміщення дві різні змінні із запереченням чи без покривають 2n~2 клітинки. Такі покриття відповідають логічним добуткам *n* змінних — конституантам одиниці функції. Наприклад, структура таблиці Вейча з відповідними мінтермами і абстрактними змінними в клітинках для трьох змінних показана на рис. 3.2.

З рисунка 3. 2 випливає, що важливою властивістю цих мінтерн є те, що ті з них, які належать до сусідніх клітинок і до клітинок, що містяться з краю одних і тих же рядків і стовпчиків таблиці, дозволяє здійснювати мінімізацію функції

безпосередньо за таблицею в наочній формі.

	x		$\overline{x}$	
y	$xy\overline{z}$	xyz	$\overline{x}yz$	$\overline{x}\,y\,\overline{z}$
$\overline{y}$	$x\overline{y}\,\overline{z}$	$x\overline{y}z$	$\overline{x}\,\overline{y}\,z$	$\overline{x}\,\overline{y}\,\overline{z}$
	$\overline{z}$	z		$\overline{z}$

Рисунок 3.2 – Структура таблиці Вейча для трьох змінних

Для того, щоб занести ДДНФ функції $f(x_1, x_2,..., x_n)$,в таблицю Вейча необхідно розмістити одиниці в клітинках, що відповідають її конституентам одиниці. Якщо одиниці розташовані в сусідніх клітинках, наприклад, таких, що відповідають добуткам xyz і $\overline{x}yz$, то внаслідок того, що вони відрізняються знаком заперечення лише в одній змінній (у даному випадку x), відбувається операція склеювання за змінною x. Результатом склеювання для цього випадку буде $xyz \vee \overline{x}yz = yz\ (x \vee \overline{x}) = yz$. Дві змінні y і z покривають спільно разом дві клітинки.

Якщо одиниці розташовані в чотирьох клітинках, то це означає, що відбувається склеювання чотирьох сусідніх конституент за стовпчиками і рядками. У результаті цього буде отримана одна змінна, яка покриває всі чотири сусідні клітинки. Тобто, склеювання як у першому, так і у другому випадках проводиться графічно за допомогою об'єднання клітинок у групи.

Склеювання клітинок у таблиці Вейча і отримання мінімальної ДНФ відбувається за таким правилом.

1. Клітинки об'єднуються у групи, що позначають операції склеювання. В об'єднанні беруть участь тільки ті сусідні клітинки, в яких містяться одиниці.

2. В групу дозволяється об'єднувати кількість клітинок 2^n, $n = 1, 2, 3...$ При цьому група може мати лише прямокутну або квадратну форму.

3. При склеюванні необхідно знайти набір максимальних груп клітинок. Під максимальною групою розуміють групу, яка не входить цілком у жодну іншу групу і відповідає простій імпліканті функції. Кількість груп у такому наборі

повинна бути мінімальною, оскільки така група відповідає мінімальній тупиковій ДНФ. Кожна одиниця таблиці Вейча повинна входити хоча б до однієї групи, що забезпечує покриття функції отриманим набором імплікант.

4. Кожна група клітинок, що отримана після склеювання, відповідає тій імпліканті функції, реальні змінні якої мають однакове значення для всіх клітинок групи.

5. Диз'юнкція всіх отриманих простих імплікант зображує результат мінімізації формули і є мінімальною ДНФ.

Оскільки при мінімізації на множині ДНФ у склеюванні беруть участь тільки клітинки, які містять одиниці, то нулі в таблиці Вейча, як правило, не вказують, а мають на увазі, що порожні клітинки містять нулі.

Приклад 1. Знайти МДНФ логічної функції

$$f(x, y, z) = xy\bar{z} \vee x\bar{y}\,\bar{z} \vee x\bar{y}z \vee \bar{x}y\bar{z} \vee \bar{x}\,\bar{y}\,\bar{z}.$$

Розв'язання. У відповідності з рис 3.1б, будуємо таблицю Вейча, до якої заносимо одиниці згідно із заданою логічною функцією, рис. 3.3.

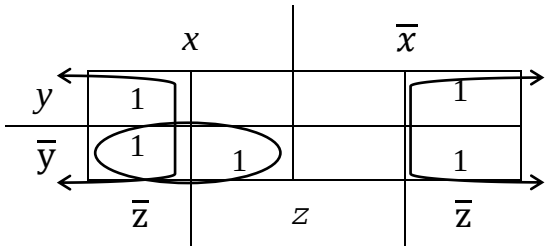


Рисунок 3.3 – Таблиця Вейча

Виконуємо склеювання клітинок таблиці відповідно з розглянутими вище правилами. При склеюванні необхідно об'єднати чотири клітинки, де містяться чотири одиниці, що перебувають у двох клітинках першого і останнього стовпчиків, які накриваються змінною $\bar{z}$, а одиниця, яка залишилась у другому стовпчику, склеюється з одиницею першого стовпчика нижнього рядка таблиці. У результаті цього отримаємо ДНФ логічної функції $f(x, y, z) = \bar{z} \vee x\,\bar{y}$.

Для мінімізації логічної функції $f(x_1, x_2, ..., x_n)$, поданої у ДКНФ, таблицю Вейча заповнюють нулями, які заносять у клітинки, що відповідають логічним сумам, на яких функція f дорівнює нулю. У всьому іншому процедура мінімізації

відбувається за тими ж правилами, що були розглянуті для логічної функції, поданої у ДНФ.

Приклад 2. Знайти МКНФ логічної функції

$f(x_1, x_2,..., x_n)=(x \vee y \vee z)(\overline{x} \vee y \vee z)(\overline{x} \vee \overline{y} \vee \overline{z})(\overline{x} \vee \overline{y} \vee \overline{y})$.

Розв'язання. У відповідності з рис.1б будуємо таблицю Вейча, до якої заносимо нулі згідно із заданою логічною функцією, рис. 3.4.

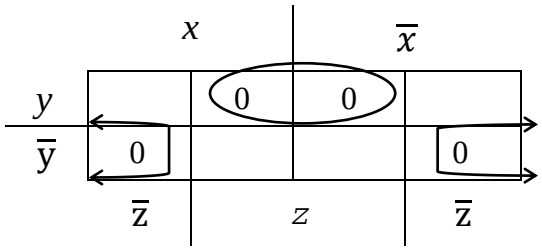


Рисунок 3.4 – Таблиця Вейча

Для отримання МКНФ цієї функції слід склеїти нулі, які розміщені поряд, і нулі, які розміщені в лівому і правому нижньому куті таблиці. Окреслені суцільними лініями по два нулі, покриваються відповідними логічними сумами у $\vee z$ і $y \vee \overline{z}$. Звідси випливає, що МКНФ заданої функції матиме вид

$f(x, y, z) = (y \vee z)(\overline{y} \vee \overline{z})$.

Застосування методу Вейча для часткового визначених логічних функцій розглянемо на прикладі.

Приклад 3. Знайти МДНФ логічної функції

$f(x, y, z, t) = x\overline{y}\,\overline{z}\,\overline{t} \vee \overline{x}\,\overline{y}\,\overline{z}\,\overline{t} \vee x\,y\,z\,\overline{t}$, яка невизначена при $\overline{x}\,y = 1$.

Розв'язання. У відповідності з рис. 2в, будуємо таблицю Вейча, до якої згідно із заданою логічною функцією заносимо одиниці і її невизначеність при $\overline{x}\,y = 1$, рис. 3.5.

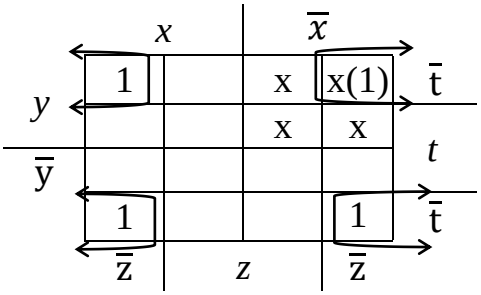


Рисунок 3.5 – Таблиця Вейча

де *x* — знак невизначеності функції.

За рахунок невизначеності функції при *xy* = 1 в крайню праву верхню клітинку ставимо одиницю. Виконуємо склеювання клітинок таблиці відповідно з розглянутими правилами. У результаті цього отримаємо МДНФ частково визначеної логічної функції

$$f(x, y, z, t) = y\bar{z}\,\bar{t} \vee \bar{y}\,\bar{z}\,\bar{t} = \bar{z}\,\bar{t}\,(y \vee \bar{y}) = \bar{z}\,\bar{t}.$$

Мінімізація частково визначених КНФ логічних функцій відбувається аналогічно мінімізації частково визначених ДНФ.

Метод Вейча може використовуватись і для більшого числа змінних. Так, наприклад, на рис. 3.6 приведена розмітка таблиці Вейча при її використанні для мінімізації логічних функцій, які мають п’ять змінних.

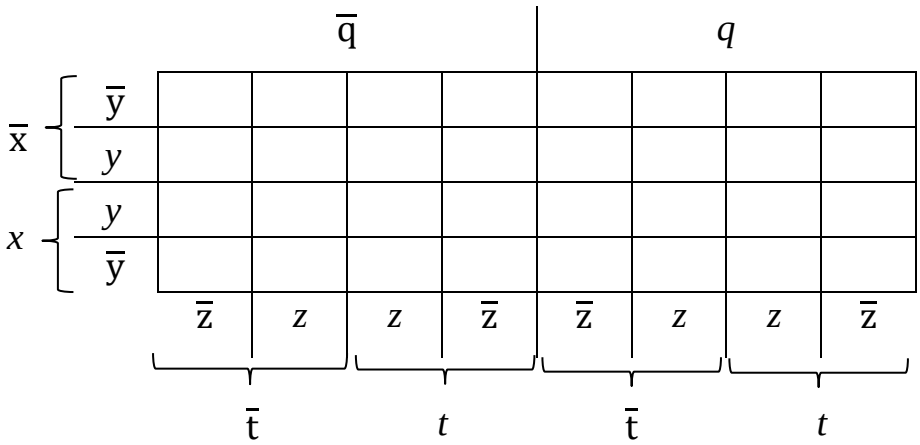


Рисунок 3.6 – Розмітка таблиці Вейча для п’яти змінних

Вона має 32 клітинки для розміщення конституент одиниці при мінімізації ДНФ або конституент нуля при мінімізації КНФ. Аналогічно будують таблиці Вейча для мінімізації логічних функцій, у яких кількість змінних більша п’яти. Вони матимуть 64 клітинки для шести змінних, 128 клітинок для 7 змінних і т.д.

Метод карт Карно. Цей метод був запропонований у 1953 році Карно після раніше опублікованого методу Вейча. За своєю суттю він повторює метод Вейча, але має дещо відмінну структуру таблиць за формою, які приведені на рис. 3.7.

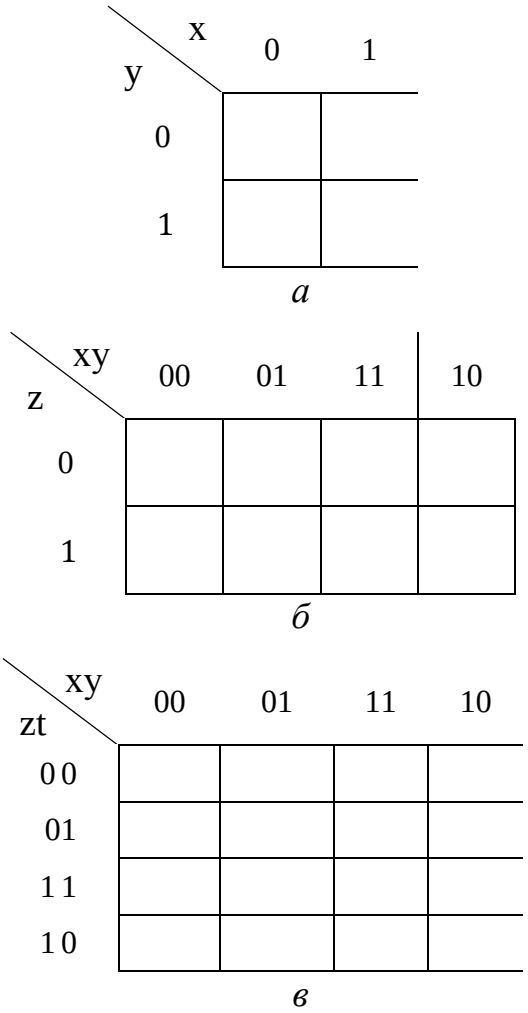


Рисунок 3.7– Структура таблиць метода Карно

На рис. 3.7*а* показана таблиця Карно для двох змінних, на рис. 3.7*б* — для трьох, а на рис. 3.7*в* — для чотирьох змінних. Для двох змінних структура таблиці Карно має вид 2x2, для трьох — 2x4, а для чотирьох — 4x4.

На відміну від таблиці Вейча, в таблицях Карно значення змінних розташовані у заголовках рядків і стовпчиків. Кожній конституенті *одиниці або* нуль *функції відповідає одна* клітинка таблиці. Нуль або одиниця в клітинці показує значення функції на даній інтерпретації. Значення змінних розташовані так, щоб сусідні рядки і стовпчики таблиці відрізнялися значенням тільки однієї змінної, тобто так само, як і у таблиці Вейча. На відміну від таблиці Вейча, для інтерпретації змінних у таблицях Карно застосовуються двійкові послідовності (0, 0), (0, 1), (1,1), (1,0). У цій послідовності перша та остання інтерпретації також відрізняються значенням тільки однієї змінної, тому перший і останній рядки (стовпчики) вважаються сусідніми. При такому розташуванні мінтерми, до яких

застосована операція склеювання, розташовуються у сусідніх клітинках таблиці, і склеювання проводиться графічно за допомогою об'єднання клітинок у групи. Так, наприклад, структура таблиці Карно з відповідними мінтермами і абстрактними змінними в клітинках для трьох змінних показана на рис. 3.8.

z \ xy	00	01	11	10
0	$\overline{x}\,\overline{y}\,\overline{z}$	$\overline{x}\,y\,\overline{z}$	$xy\overline{z}$	$x\overline{y}\,\overline{z}$
1	$\overline{x}\,\overline{y}\,z$	$\overline{x}\,yz$	xyz	$x\overline{y}z$

Рисунок 3.8 – Приклад заповнення карт Карно

Заповнення таблиць Карно для логічної функції аналогічне заповненню таблиць Вейча.

Всі правила склеювання клітинок і запису МДНФ і МКНФ в таблицях Карно аналогічні правилам склеюванням клітинок і запису МДНФ і МКНФ в таблицях Вейча.

Приклад 4. Знайти МДНФ для функції

$f(x, y, z) = x\,y\,\overline{z} \vee x\,\overline{y}\,\overline{z} \vee x\,\overline{y}\,z \vee \overline{x}\,y\,\overline{z} \vee \overline{x}\,\overline{y}\,\overline{z}$.

Розв'язання. У відповідності з рис 3.8 будуємо таблицю Карно, до якої заносимо одиниці згідно із заданою логічною функцією, рис. 3.9.

z \ xy	00	01	11	10
0	1	1	1	1
1				1

Рисунок 3.9 – Таблиця Карно

Виконуємо склеювання клітинок таблиці у відповідності з розглянутими правилами. При склеюванні об'єднаємо чотири клітинки, де містяться чотири одиниці першої стрічки, які накриваються змінною *z*, а одиниця, яка залишилась в останньому стовпчику, склеюється з одиницею останнього стовпчика першої стрічки. У результаті цього отримаємо МДНФ логічної функції

$$f(x,y,z) = \bar{z} \vee x \bar{y}.$$

Приклад 5. Знайти МКНФ для функції

$$f(x, y, z)=(x \vee y \vee z)(\bar{x} \vee y \vee z)(x \vee \bar{y} \vee \bar{z})(\bar{x} \vee \bar{y} \vee \bar{z}).$$

Розв'язання. У відповідності з рис. 3.7в, будуємо таблицю Карно, до якої заносимо нулі згідно із заданою логічною функцією, рис. 3.10.

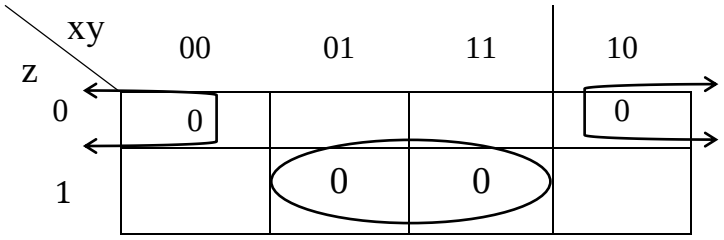


Рисунок 3.10 – Таблиця Карно

Для отримання МКНФ функції необхідно склеїти нулі, які стоять поряд, і нулі, які розміщені в лівому і правому кутках першої стрічки таблиці. Окреслені суцільними лініями по два нулі покриваються відповідними логічними сумами $y \vee z$ і $\bar{y} \vee \bar{z}$. Звідси випливає, що МКНФ заданої логічної функції матиме вид

$$f(x, y, z)=(y \vee z)(\bar{y} \vee \bar{z}).$$

Застосування методу Карно для частково визначених логічних функцій аналогічний методу Вейча.

Приклад 6. Знайти МДНФ для частково визначеної логічної функції, яка не визначена, якщо $x y = 1$.

$$f(x, y, z,t)= \bar{x} \bar{y} z \bar{t} \vee \bar{x} y z \bar{t} \vee x \bar{y} z \bar{t} \vee x \bar{y} \bar{z} \bar{t} \ .$$

Розв'язання. У відповідності з рис. 3.7в, будуємо таблицю Карно, до якої згідно із заданою логічною функцією заносимо одиниці і її невизначеність при $x y = 1$, рис. 3.11.

Виконуємо склеювання клітинок таблиці у відповідності з розглянутими

правилами У результаті цього отримаємо МДНФ частково визначеної логічної функції.

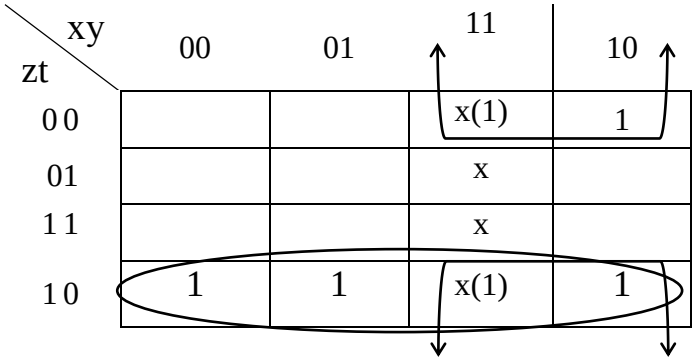
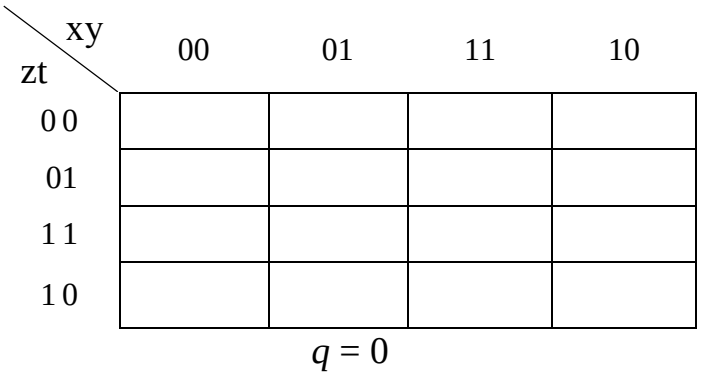


Рисунок 3.11 – Таблиця карт Карно

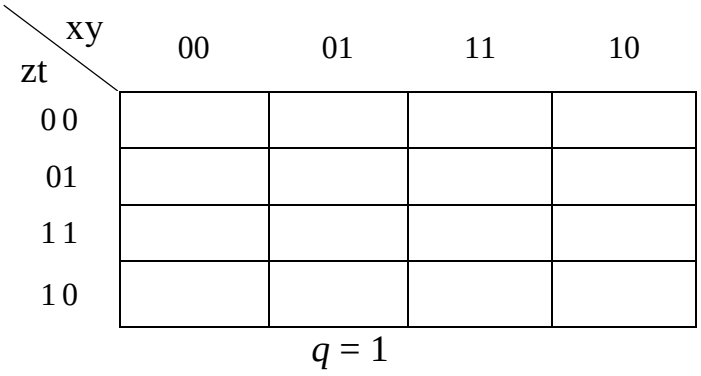
$$f(x, y, z, t) = z\bar{t} \vee x\bar{t}$$

Мінімізація частково визначених КНФ логічних функцій відбувається аналогічно мінімізації частково визначених ДНФ.

Метод Карно може використовуватись і для більшого числа змінних. Так, наприклад на рис. 3.12 приведена розмітка таблиць Карно при їх використанні для мінімізації логічних функцій, які мають п’ять змінних.



$q = 0$



$q = 1$

Рисунок 3.12 – Розмітка таблиць Карно для п’яти змінних

Дана таблиця (рис. 3.12) має дві таблиці. В одній із них значення п’ятої

змінної $q = 0$, а в іншій – $q = 1$. В сумі вона має 32 клітинки для розміщення конституант одиниці при мінімізації ДНФ або конституант нуля при мінімізації КНФ. Склеювання клітинок таблиці відбувається у відповідності з розглянутими правилами.

Завдання до практичної роботи

1. Побудувати таблиці Вейча і Карно та мінімізувати за ними такі логічні функції:

а) $f(x, y, z) = x\,\bar{y}z \vee \bar{x}\,y\,\bar{z} \vee x\,\bar{y}\,\bar{z} \vee \bar{x}\,\bar{y}\,z \vee \bar{x}\,\bar{y}\,\bar{z}$;

б) $f(x, y, z) = x\,y\,z \vee \bar{x}\,y\,z \vee x\,\bar{y}\,z \vee x\,\bar{y}\,\bar{z} \vee \bar{x}\,\bar{y}\,z$;

в) $f(x, y, z, t) = xy\,\bar{z}\,\bar{t} \vee x\,\bar{y}\,\bar{z}\,t \vee x\,\bar{y}\;\bar{z}\,t \vee \bar{x}\,\bar{y}\;\bar{z}\,t \vee xyzt$;

г) $f(x, y, z, t) = \bar{x}\,\bar{y}z\,t \vee \bar{x}\,\bar{y}\,\bar{z}\,t \vee x\,\bar{y}\,\bar{z}\,\bar{t} \vee x\bar{y}\,\bar{z}\,t \vee \bar{x}yz\bar{t}$.

2. Знайти мінімальні ДНФ і КНФ, що задані такими таблицями Вейча:

а)

	x		$\bar{x}$		
y	1	1	1	0	$\bar{t}$
	0	0	1	1	t
$\bar{y}$	0	0	1	1	
	1	1	1	0	$\bar{t}$
	$\bar{z}$	z		$\bar{z}$	

в)

	x		$\bar{x}$		
y	1	0	0	1	$\bar{t}$
	0	1	1	0	t
$\bar{y}$	0	1	1	0	
	1	0	0	1	$\bar{t}$
	$\bar{z}$	z		$\bar{z}$	

б)

	x		$\bar{x}$		
y	0	0	0	0	$\bar{t}$
	1	1	0	1	t
$\bar{y}$	1	1	0	1	
	1	1	1	1	$\bar{t}$
	$\bar{z}$	z		$\bar{z}$	

г)

	x		$\bar{x}$		
y	0	1	1	0	$\bar{t}$
	1	1	0	1	t
$\bar{y}$	0	0	1	1	
	1	1	0	1	$\bar{t}$
	$\bar{z}$	z		$\bar{z}$	

3. Знайти мінімальні ДНФ і КНФ, що задані такими таблицями Карно:

а)

zt \ xy	00	01	11	10
00	0	0	0	0
01	1	1	0	0
11	0	0	1	1
10	0	0	0	0

б)

zt \ xy	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	1	1	0	1
10	1	1	1	1

в)

zt \ xy	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	1	0	1	0
10	1	0	1	0

г)

zt \ xy	00	01	11	10
00	1	0	1	0
01	0	1	1	1
11	0	1	1	1
10	1	0	1	0

Контрольні запитання

1. В чому полягає мінімізація логічних функцій?
2. Яку імпліканту та імпліценту називають простою?
3. Що таке скорочена ДНФ та КНФ?
4. Дайте визначення МДНФ та МКНФ?
5. Що розуміють під операцією повного і неповного склеювання логічної функції?
6. Запишіть формули операцій диз’юнктивного склеювання і поглинання.
7. Сформулюйте правила склеювання клітинок і отримання МДНФ (МКНФ) за методом Вейча.
8. Сформулюйте правила склеювання клітинок і отримання МДНФ (МКНФ) за методом Карно.
9. У чому відмінність методів Вейча і Карно, а також їх таблиць для мінімізації?

3 РЕАЛІЗАЦІЯ ЛОГІЧНИХ ФУНКЦІЙ В ЗАДАНОМУ БАЗИСІ

Мета: отримання навичок реалізації логічних функцій в заданому базисі

Теоретичні відомості

Запис логічних виразів зазвичай здійснюють в *кон'юнктивній* або *диз'юнктивній нормальних формах*. У диз'юнктивній нормальній формі логічні вирази записуються як логічна сума логічних добутків, в кон'юнктивній нормальній формі – як логічний добуток логічних сум. Порядок дій такий самий, як і в звичайних виразах алгебри. Логічні вирази пов'язують значення логічної функції зі значеннями логічних змінних [4–6].

Будь-який логічний вираз, складений із n змінних $x_n, x_{n-1}, ..., x_1$ за допомогою кінцевої кількості операцій алгебри логіки, можна розглядати як деяку функцію n змінних. Таку функцію називають логічною. Відповідно до аксіом алгебри логіки функція може приймати залежно від значень змінних значення 0 або 1. Функція n логічних змінних може бути визначена для 2^n значень змінних, що відповідають всім можливим значенням n – розрядних двійкових чисел.

Фізичний пристрій, що реалізовує одну з операцій алгебри логіки або просту логічну функцію, називається логічним елементом. Схема, що складається з кінцевого числа логічних елементів за певними правилами, називається логічною схемою. Основним логічним функціям відповідають схемні елементи, які їх виконують.

Якщо число логічних змінних не перевищує шести, перетворення логічних рівнянь зручно проводити за допомогою карт Карно або діаграм Вейча. Мета перетворень – отримання компактного логічного виразу (мінімізація). Мінімізацію проводять об'єднанням наборів (термів) на карті Карно. Набори, що об'єднуються, повинні мати однакові значення функції (всі нулі або всі одиниці).

Для реалізації функцій на логічних елементах І–НЕ або АБО–НЕ необхідно перетворити логічний вираз у базис логічних елементів І–НЕ або АБО–НЕ, тобто записати вираз за допомогою операцій логічного множення та інверсії.

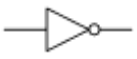
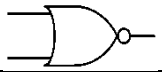
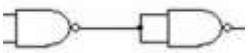
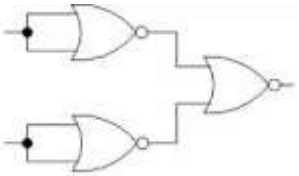
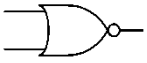
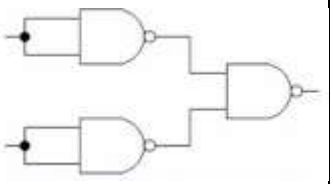
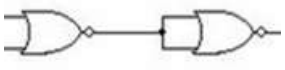
До переваг логічного пристрою реалізованого в базисі І–НЕ і АБО–НЕ належить:

– зменшення номенклатури елементів до одного типу дозволяє спростити компоновку пристрою і його ремонт;

– наявність в кожному елементі інвертора (підсилювача), який компенсує загасання потенціалів при передачі їх через кон'юнктор або діз'юнктор елемента, дозволяє не накопичувати загасання сигналу при проходженні його через ряд послідовно включених елементів, що могло б призвести до зниження рівня U1 (логічна "1").

Схеми та вирази логічних операцій, які реалізовані в базисі І, АБО, НЕ, базисі І–НЕ та АБО–НЕ наведено у таблиці 3.1

Таблиця 3.1 – Схеми та вирази логічних операцій

Операція	Базис І, АБО, НЕ	Базис І–НЕ	Базис АБО–НЕ
НЕ			
	$Y = \overline{X}$	$Y = \overline{X \cdot X} = \overline{X}$	$Y = \overline{X \vee X} = \overline{X}$
І			
	$Y = X_1 \cdot X_2$	$Y = \overline{\overline{X_1 \cdot X_2}} = X_1 \cdot X_2$	$Y = \overline{\overline{X_1} \vee \overline{X_2}} = X_1 \cdot X_2$
АБО			
	$Y = X_1 \vee X_2$	$Y = \overline{\overline{X_1} \cdot \overline{X_2}} = X_1 \vee X_2$	$Y = \overline{\overline{X_1 \vee X_2}} = X_1 \vee X_2$

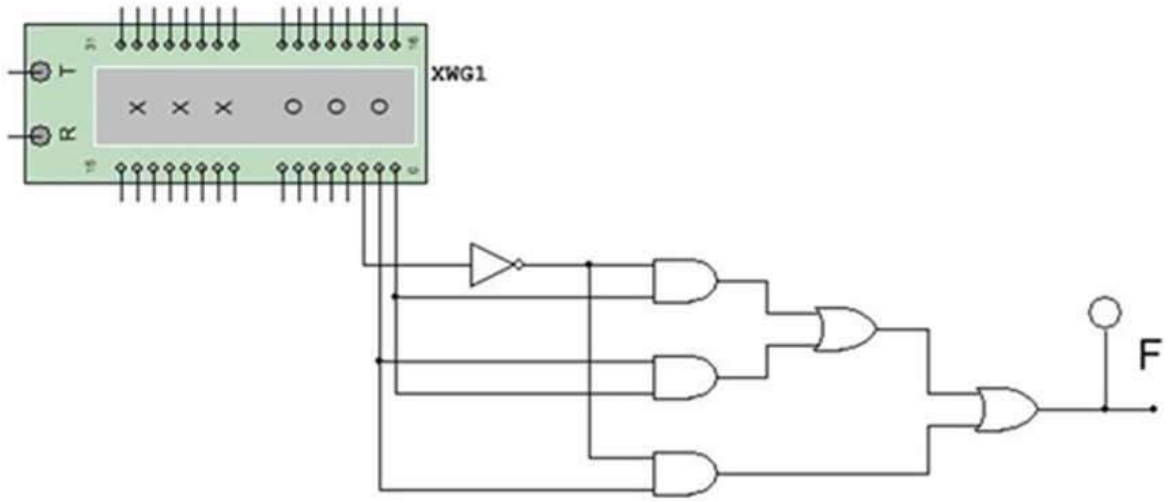


Рисунок 3.1 – Логічна схема в базисі І, АБО, НЕ

2. У покроковому режимі заповнюємо таблицю істинності для заданої функції (табл. 3.2).

Таблиця 3.2 – Таблиця істинності

X	Y	Z	F
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	0
1	1	1	1

3. Використовуючи таблицю перетворення в базис І–НЕ, будуємо в *Multisim* логічну схему (рис. 3.2) з використанням логічних двоходових елементів І–НЕ.

$$\boldsymbol{F = \overline{X}Z \vee YZ \vee \overline{X}Y}$$

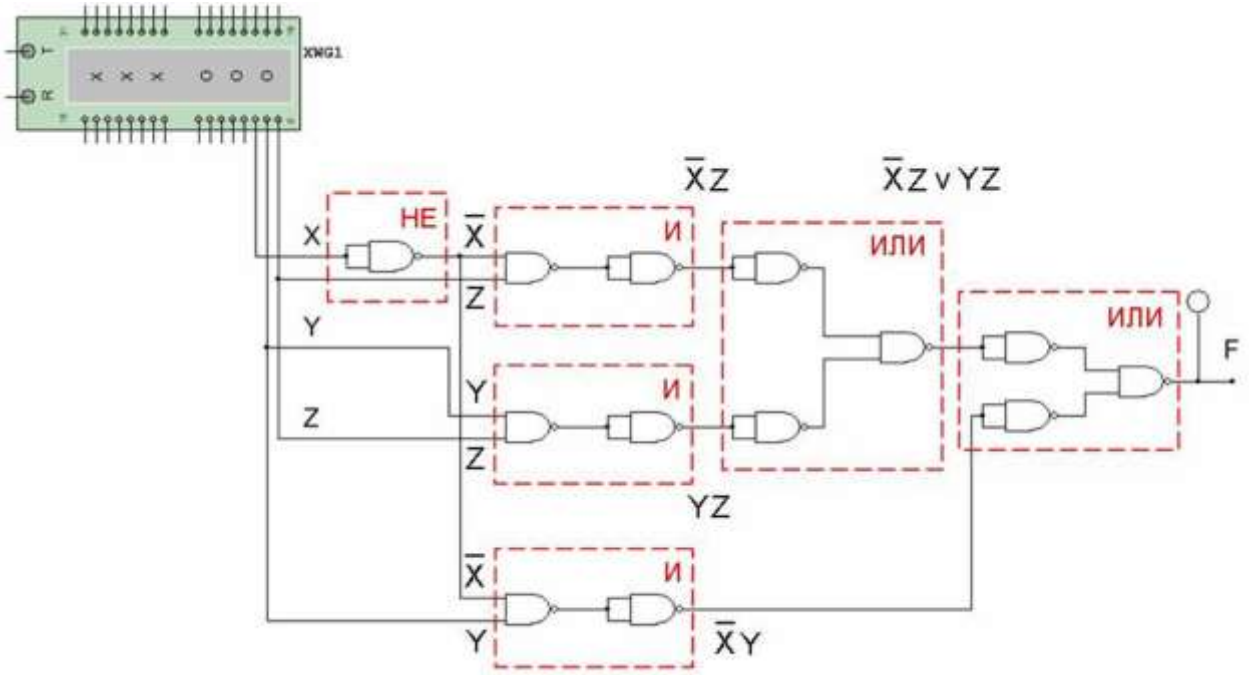


Рисунок 3.2 – Логічна схема в базисі І–НЕ

4. Видаляємо фрагменти обведені пунктиром з логічної схеми (рис. 3.2), що містять поспіль два інвертора $\boldsymbol{F = \overline{X}Z \vee YZ \vee \overline{X}Y}$.

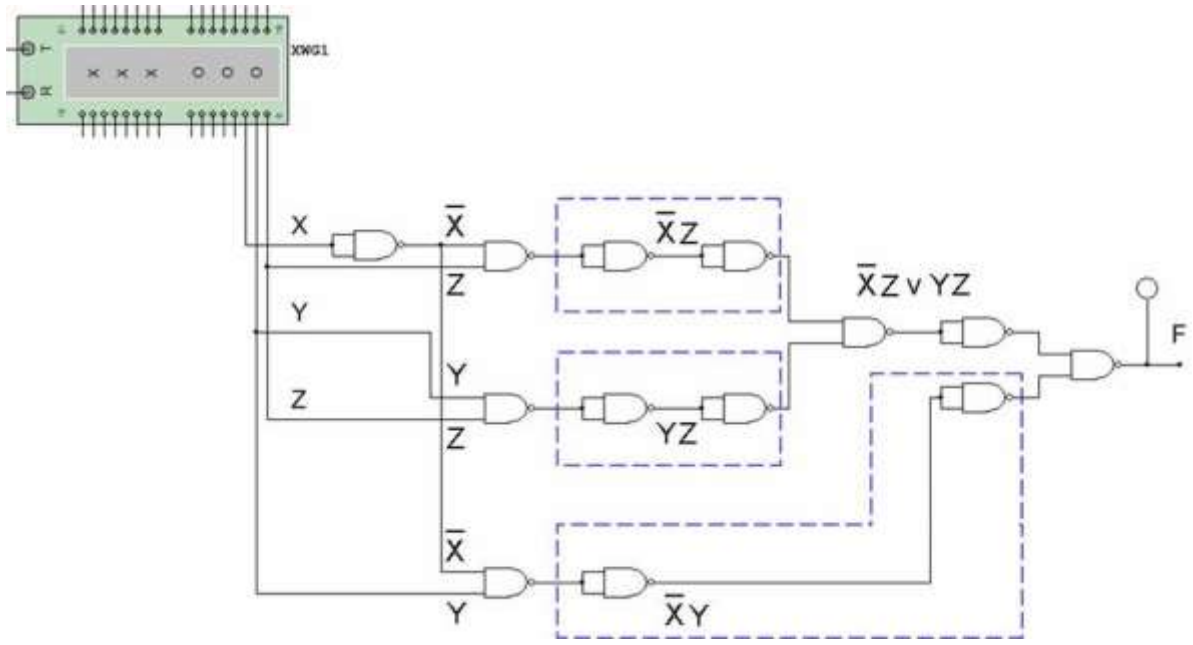


Рисунок 3.3 – Логічна схема в базисі І–НЕ

5. Отримаємо логічну схему в базисі І–НЕ, яка наведена на рисунку 3.3.

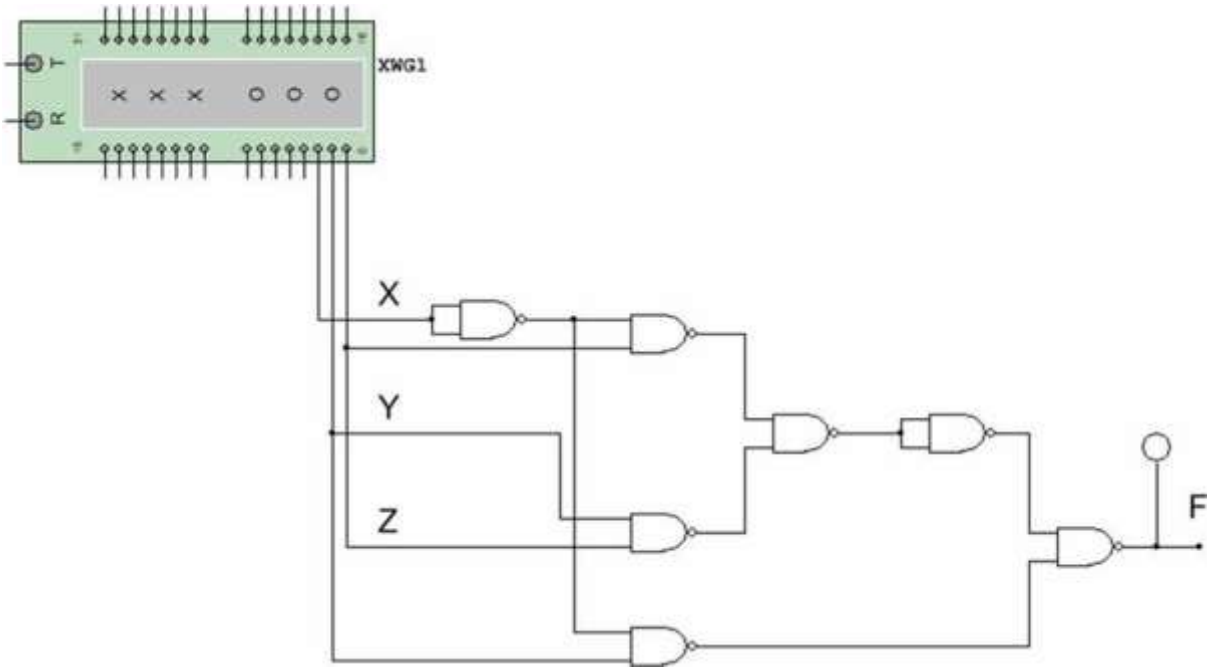


Рисунок 3.3 – Спрощена логічна схема в базисі І–НЕ

6. У покроковому режимі заповнюємо таблицю істинності (ТІ) для заданої функції і порівнюємо її з ТІ, отриманої раніше (див. таблицю 3.2).

7. Нумеруємо елементи І–НЕ для підрахунку необхідної кількості мікросхем *74LS00N* та їх закріплення за ними (рис. 3.4).

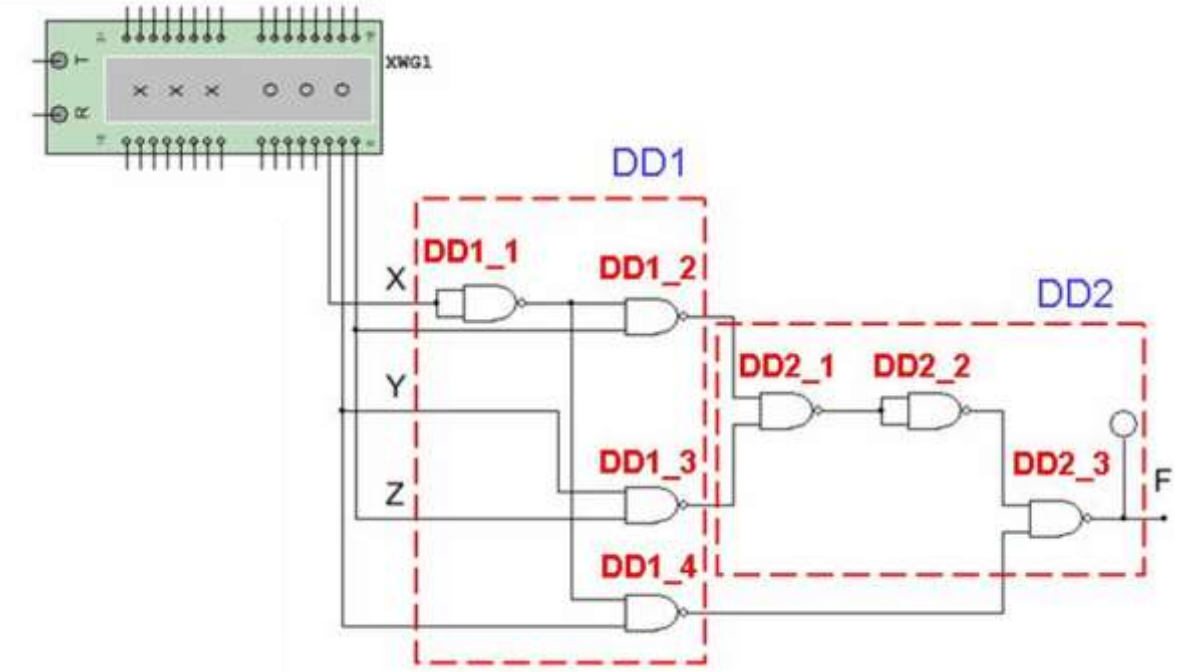


Рисунок 3.4 – Розподіл логічних елементів у мікросхемах

8. Будуємо схему з використанням мікросхем *74LS00N* (рис. 3.5)

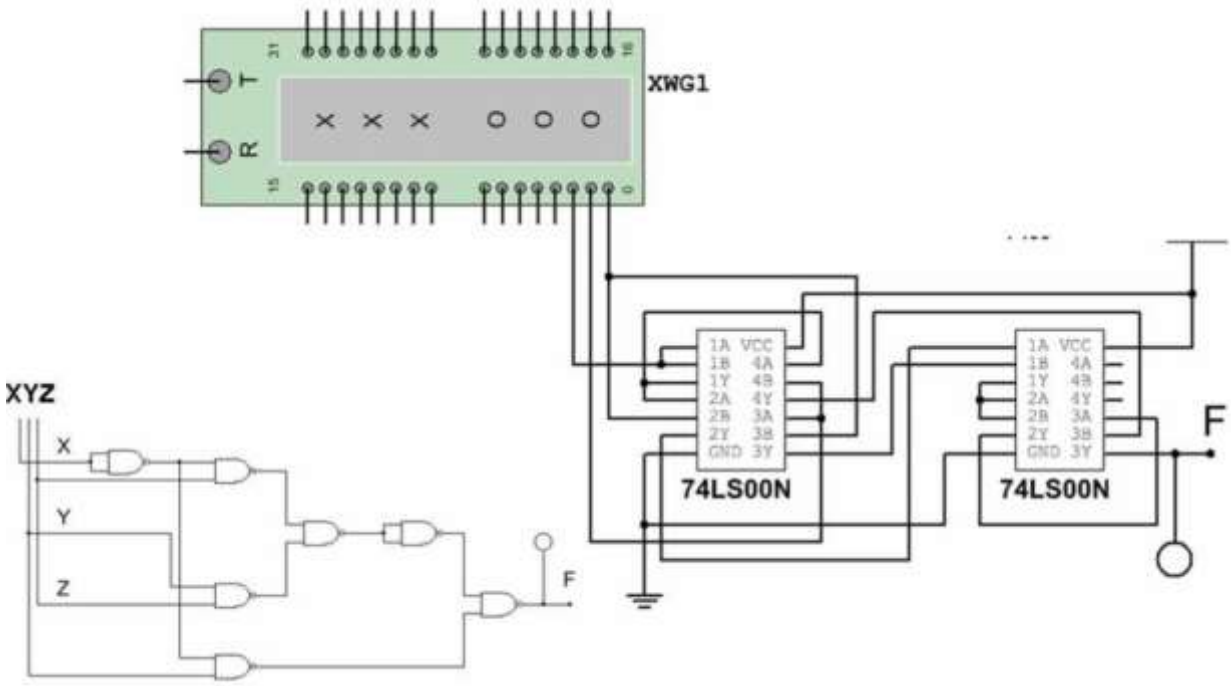


Рисунок 3.5 – Логічна схема реалізована на мікросхемі*74LS00N*

З рисунку 3.5 можна зробити висновок, що одна з мікросхем *74LS00N* буде задіяна повністю, а друга на 3/4.

Побудова схем логічної функції заданої в базисі АБО–НЕ відбивається аналогічно, як і для схемам реалізованим в базисі І–НЕ.

Завдання до практичної роботи

1. За таблицею істинності згідного свого варіанта для функції *F* побудуйте її СДНФ.

2. Мінімізуйте отриманий логічний вираз за допомогою карт Карно.

3. За мінімальної ДНФ побудуйте в *Multisim* логічну схему в базисі І, АБО, НЕ і перевірте її працездатність.

4. Побудуйте отриману схему в базисі І–НЕ і перевірте її працездатність.

5. Аналогічно пунктам 1–4 побудуйте СКНФ, мінімальну КНФ і відповідні їй схеми в базисах І, АБО, НЕ і АБО–НЕ.

Варіант 1

№	*X*	*Y*	*Z*	*F*
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Варіант 2

№	*X*	*Y*	*Z*	*F*
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

Варіант 3

№	*X*	*Y*	*Z*	*F*
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

Варіант 4

№	*X*	*Y*	*Z*	*F*
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

Варіант 5

№	*X*	*Y*	*Z*	*F*
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

Варіант 6

№	*X*	*Y*	*Z*	*F*
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

Варіант 7

№	*X*	*Y*	*Z*	*F*
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Варіант 8

№	*X*	*Y*	*Z*	*F*
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

Варіант 9

№	*X*	*Y*	*Z*	*F*
0	0	0	0	1
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	0

Варіант 10

№	*X*	*Y*	*Z*	*F*
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

Варіант 11

№	*X*	*Y*	*Z*	*F*
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

Варіант 12

№	*X*	*Y*	*Z*	*F*
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

Варіант 13

№	*X*	*Y*	*Z*	*F*
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Варіант 14

№	*X*	*Y*	*Z*	*F*
0	0	0	0	1
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

Варіант 15

№	*X*	*Y*	*Z*	*F*
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

Варіант 16

№	*X*	*Y*	*Z*	*F*
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

Варіант 17

№	*X*	*Y*	*Z*	*F*
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

Варіант 18

№	*X*	*Y*	*Z*	*F*
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Контрольні запитання

1. Назвіть способи завдання логічних функцій.
2. Назвіть особливості побудови карт Карно.
3. Як відбувається мінімізація за методом карт Карно.
4. Опишіть послідовність пошуку мінімальної ДНФ за методом Вейча.
5. Який порядок реалізації функції в базисі І–НЕ?
6. Який порядок реалізації функції в базисі АБО–НЕ ?

4 ДОСЛІДЖЕННЯ ШИФРАТОРА І ДЕШИФРАТОРА

Мета: ознайомитися з принципом дії шифраторів і дешифраторів. Реалізувати і дослідити функціональні модулі на основі даних пристроїв.

Теоретичні відомості

Комбінаційними цифровими пристроями (КЦУ) називаються логічні мікросхеми, реалізують однозначну відповідність вихідних сигналів в певний момент часу значеннями вхідних сигналів [4–6].

Шифратор (coder, encoderj– КЦУ, перетворює активний сигнал на одному з входів D (десяткове число на клавіатурі) в двійковий код на виходах А (рис. 4.1а). Число входів D (адресних шин) і виходів А пов'язані між собою:

$D = 2^A$. Шини EO, GS є управляючими. На рисунку 4.1а зображено умовне позначення шифратора «8 в 3», тобто має 8 входів і 3 виходи. Розглянемо таблицю істинності (табл. 4.1) шифратора «4 в 2».

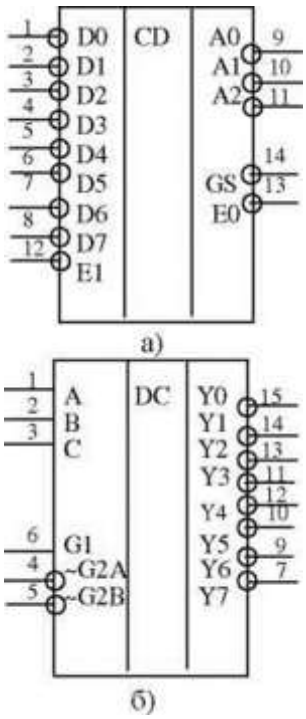


Рисунок 4.1

Таблиця 4.1

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

Для виходів можна записати:

$A0 = \overline{D2} \vee \overline{D3}$ та $A1 = \overline{D1} \vee \overline{D3}$.

Принципова схема такого шифратора наведена на рисунку 4.2.

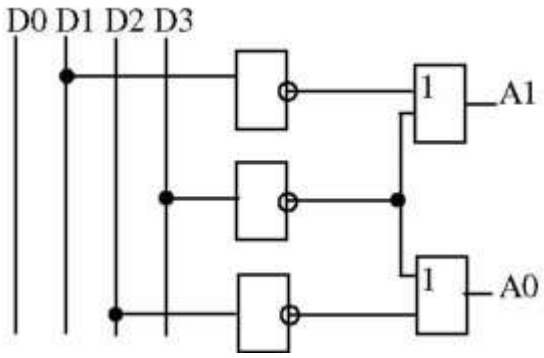


Рисунок 4.2 – Принципова схема шифратора «4 в 2»

Дешифратор (decoder)– КЦУ, на входи якого подаються двійкові коди, а активний сигнал з'являється тільки на одному з виходів (рис. 4.1.б). Має *n* інформаційних входів (вхідні шини *A, B, C*) і 2^n виходів (*Y0 ... Y7*). *G1, G2A, G2B* – керуючі шини, які дозволяють функціонування дешифратора або переводять його в пасивний стан, тобто на всіх виходах встановлюється одиниця. Дозволяючий вхід дешифратора може бути прямим (*Gl* = l) і інверсним (*G1* = 0). Зазвичай входи управління використовують для каскадування (збільшення розрядності) дешифратора.

Рівні сигналів на виходах описуються виразами:

$$Y0 = \overline{\overline{C} \cdot \overline{\mathrm{B}} \cdot \overline{A}}\ ;$$

$$Y1 = \overline{\overline{C} \cdot \overline{\mathrm{B}} \cdot A}\ ;$$

$$Y2 = \overline{\overline{C} \cdot B \cdot \overline{A}}\ ;$$

$$Y3 = \overline{\overline{C} \cdot B \cdot A}\ ;$$

$$Y4 = \overline{\mathrm{C} \cdot \overline{\mathrm{B}} \cdot \overline{\mathrm{A}}}\ ;$$

$$Y5 = \overline{\mathrm{C} \cdot \overline{\mathrm{B}} \cdot A}\ ;$$

$$Y6 = \overline{\mathrm{C} \cdot B \cdot \overline{\mathrm{A}}}\ ;$$

$$Y7 = \overline{C \cdot B \cdot A}.$$

Дешифратор може бути використаний в якості *демультиплексора* – логічного

комутатора, що підключає вхідний сигнал до одного з виходів. У цьому випадку функцію інформаційного входу виконує один з керуючих входів, а стану входів задають номер виходу, на який передається сигнал з входу дозволу.

Завдання до практичної роботи.

1. Дослідження принципу дії шифратора

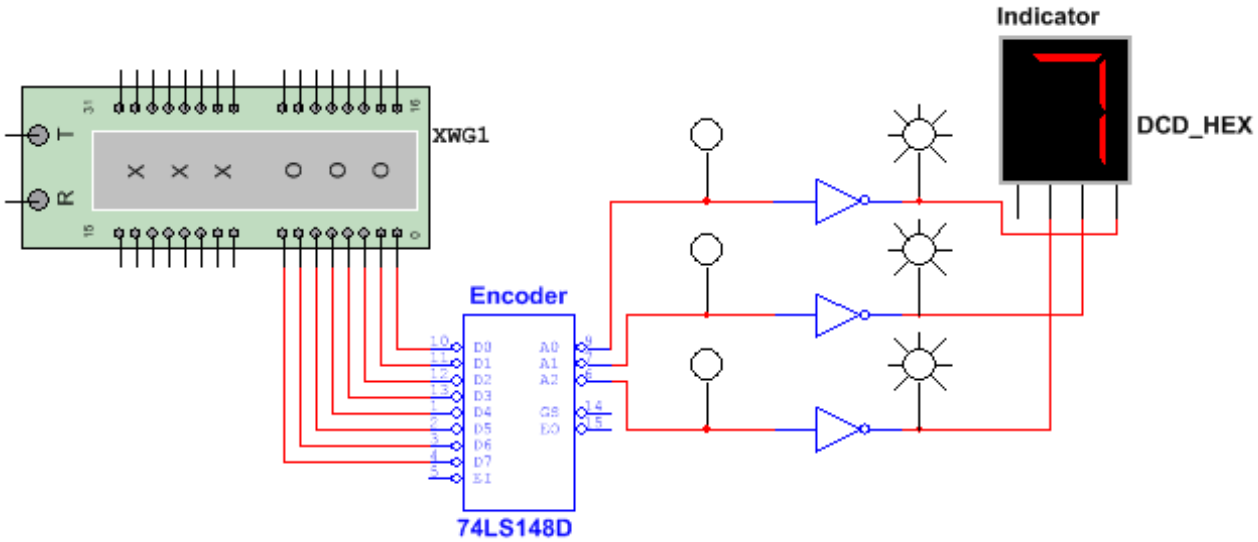


Рисунок 4.3 – Схема дослідження шифратора

Користуючись програмою *Multisim,* зібрати схему, представлену на рисунку 4.3. Як дешифратора використовувати мікросхему *74LS148* (8 *line to* 3 *line encoder*) на основі логічних елементів транзисторно–транзисторної логіки – ТТЛ (*Group – TTL*). До виходів *A*0 ... *A*2 підключити пробники (*Group – Indicators*; *Family – Probe*) і інвертори (*TTL –74LS04*) для полегшення сприйняття закодованої інформації в двійковому коді. Для подачі вхідних сигналів використовувати генератор слів (*Word Generator*) в покроковому режимі (*Step*). Комбінації вхідних сигналів задати відповідно до рис. 4.4. Позиції першої і останньої комбінації відзначити в меню, що викликається правою кнопкою миші (*Set Initial* (*Final*) *Position*). Скласти таблицю істинності і пояснити отримані результати.

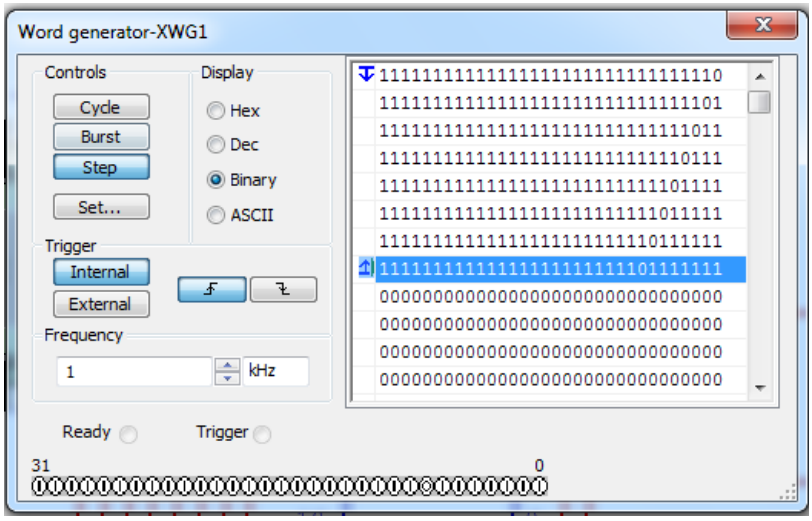


Рисунок 4.4 – Комбінації вхідних сигналів

2. Дослідження принципу дії дешифратора

Зібрати схему, зображену на рисунку 4.5. Від генератора слів (*Word Generator*) подати сигнали на входи *A, B, C* дешифратора. Комбінації слів на генераторі встановити від 000 до 111 (рис. 4.6). Дослідження проводити в покроковому (*Step*) режимі. Керуючий вхід *G1* можна підключити до генератора слів або за допомогою ключа до джерела живлення (*Power Sources – VCC*) і заземлення; визначити при якому логічному рівні *G1* дешифратор переводиться в пасивний стан. Скласти таблицю істинності і пояснити отримані результати.

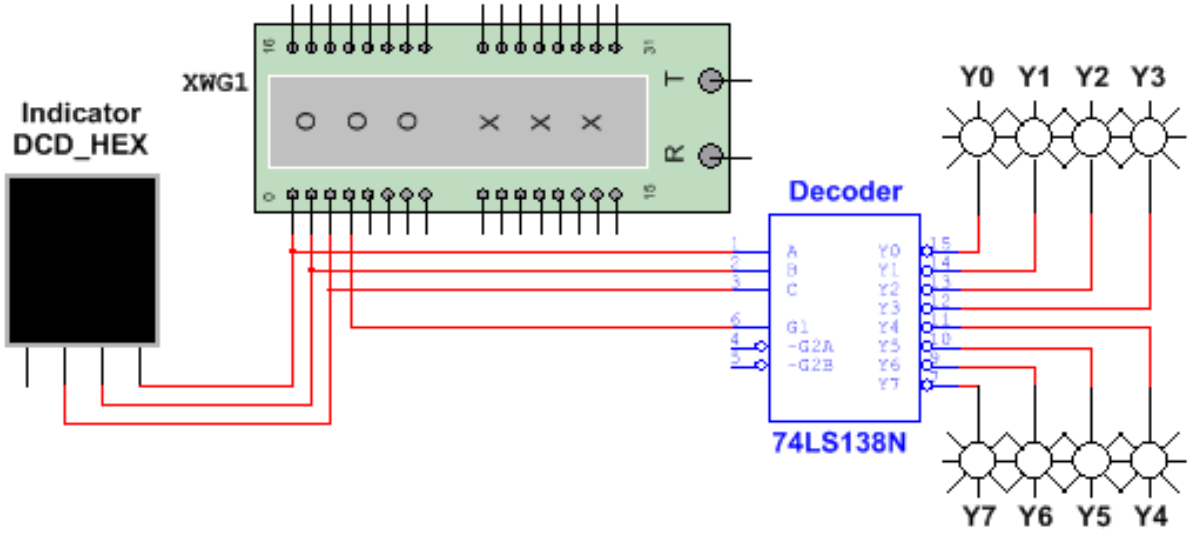


Рисунок 4.5 – Схема дослідження дешифратора

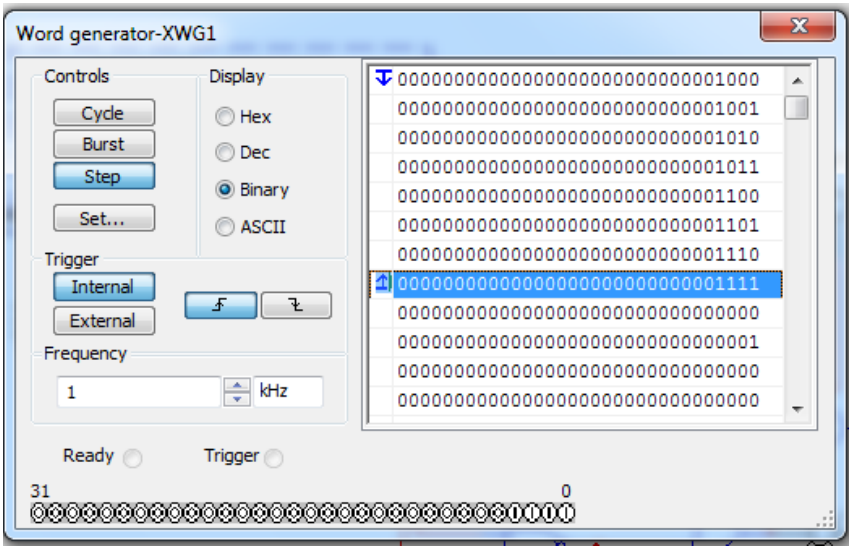


Рисунок 4.6 – Комбінації слів

Слід врахувати, що декодер, представлений на рис. 4.5 має інверсні виходи.

3. Дослідження роботи дешифратора з логічною схемою на виході

а) Зібрати схему, зображену на рисунку 4.7.

Від генератора слів (Word Generator) в покроковому (Step) режимі подати сигнали на входи A, B, C дешифратора. Комбінації слів на генераторі встановити від 000 до 111 (рис. 4.6). Керуючий вхід *G*1 повинен перебувати в активному стані, для цього підключити його до джерела живлення (Power Sources – VCC) або до генератора слів (як в завданні 2).

б) Спостерігати рівень логічного сигналу на виході схеми за допомогою пробника *Y*12, скласти таблицю істинності функції F, яка реалізується схемою. Скласти аналітичний вираз, відповідне представленій схемі (рис. 4.8), провести логічні перетворення і спростити його. Порівняти з функціями, представленими в таблиці 4.2. Визначити номер варіанта, для якого була розроблена і реалізована дана схема.

Таблиця 4.2

Варіант	Функція
1	$F = \overline{B} \cdot \overline{A} \vee C \cdot \overline{B} \vee \overline{C} \cdot B \cdot A$
2	$\overline{A} \cdot \overline{C} \vee B$
3	$F = \overline{C} \cdot \overline{B} \cdot \overline{A} \vee B \cdot A \vee C \cdot B$
4	$F = C \cdot \overline{A} \vee \overline{B} \cdot A \vee \overline{C} \cdot A$
5	$F = C \cdot \overline{A} \vee \overline{B} \cdot C$

Слід врахувати, що декодер, представлений на рисунку 4.7 має інверсні виходи

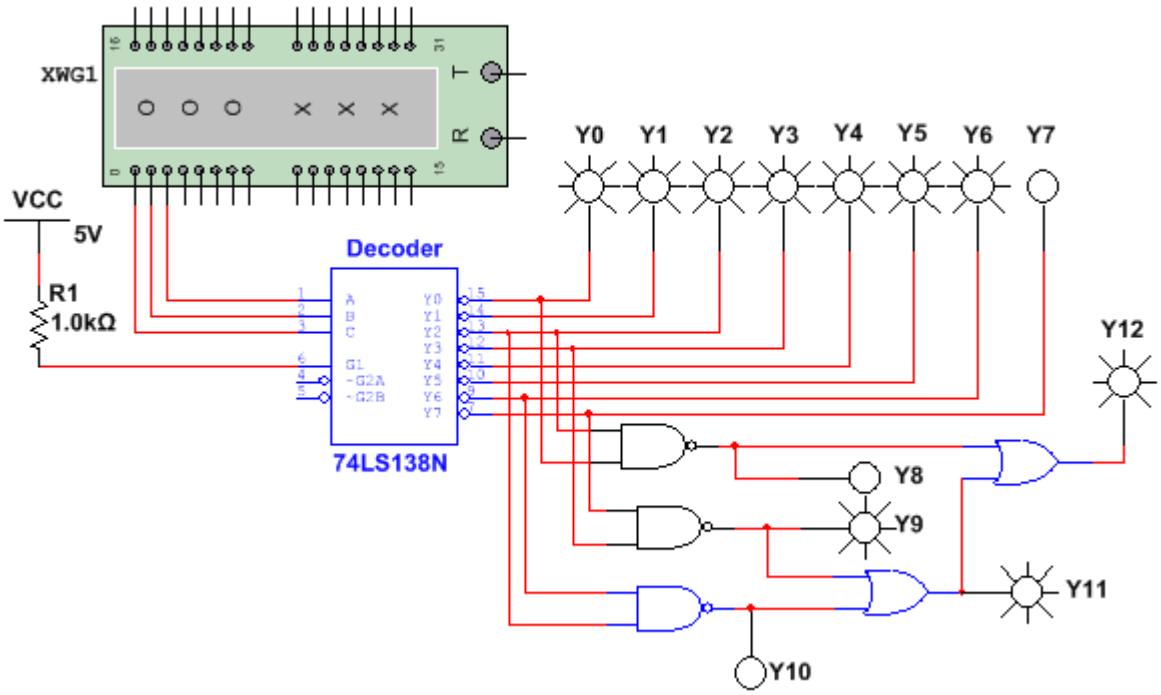


Рисунок 4.7 – Схема базисного дешифратора

Контрольні запитання

1. Принцип дії, принцип побудови шифраторів. Застосування. Умовне графічне позначення і призначення його виводів.
2. Принцип дії, принцип побудови дешифраторів. Застосування. Умовне графічне позначення і призначення виводів.
3. На основі двох дешифраторів 2х4 розробити один 3х8.
4. На основі дешифраторів 2х4 розробити один 4х16.

5 ПРОЕКТУВАННЯ СУММАТОРА

Мета: отримання навичок проектування суматорів на рівні регістрових передач і з використанням поведінкової моделі на мові опису апаратури.

Теоретичні відомості

Суматором називається комбінаційний логічний пристрій, призначений для виконання операції арифметичного складання чисел, представлених у вигляді двійкових кодів [4–6].

Суматори є одним з основних вузлів арифметико-логічних пристроїв. Термін суматор охоплює широкий круг пристроїв, починаючи з простих логічних схем, до складних цифрових вузлів. Загальним для всіх цих пристроїв є арифметичне складання чисел, представлених в двійковій формі.

Розглянемо ЛФ, що описує операції арифметичного складання двох однорозрядних двійкових кодів x_1 і x_0. Алгоритм її виконання пояснюється таблицею відповідності (табл. 5.1). У графі s наведено значення результату складання (суми), а в графі p – набутого при цьому значення перенесення в старший розряд. Слід звернути увагу на відмінності результатів, що отримуються при арифметичному і логічному складаннях, при логічному складанні в останньому рядку стовпця s було б присутнє значення 1. Це відмінність результатів даних операцій не дозволяє застосувати для арифметичного підсумовування елемент АБО, а вимагає розробки спеціалізованого пристрою.

Значення сигналу перенесення, рівного одиниці в останньому рядку табл. 5.1 свідчить про те, що результат, отриманий при виконанні операції арифметичного складання, в цьому випадку не може бути представлений двійковим кодом, розрядність якого рівна розрядності слів доданків. Для представлення результату необхідне слово, що має на один розряд більше, ніж коди доданків.

Таблиця 5.1 – Таблиця відповідності складання двох однорозрядних двійкових кодів

x_1	x_0	s	p
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Використовуючи наведену таблицю, легко записати систему ЛФ, що описують алгоритм операції арифметичного складання

$$s = \overline{x_1} x_0 + x_1 \overline{x_0}\,, \tag{5.1}$$

$$p = x_1 x_0\,. \tag{5.2}$$

Функція, що описується виразом (5.1), дуже часто зустрічається при розробці цифрових пристроїв. Її називають функцією що «Виключає АБО», або *сумою по модулю два.* Таким чином, для підсумовування двох двійкових однорозрядних кодів необхідно виконати логічну операцію що «Виключає АБО».

З метою спрощення вираз (5.1) зазвичай записують таким чином:

$$s = x_1 \oplus x_0\,.$$

Логічні елементи, що виконують операцію що «Виключає АБО», завжди мають тільки два входи, тобто операції завжди виконуються тільки над двома змінними.

Таблиця 5.1 застосовна тільки для складання однорозрядних двійкових кодів або молодших розрядів багаторозрядних слів. Таблиця складання старших розрядів багаторозрядних двійкових слів повинна бути доповнена змінною можливого перенесення з більш молодшого розряду (табл.5. 2). Логічні функції, що описують результати складань, в цьому випадку матимуть вид

$$s = (x_1 \oplus x_0)\overline{p_1} + \overline{(x_1 \oplus x_0)}p_{-1}, \tag{5.3}$$

$$p = x_1 x_0 + (x_1 \oplus x_0)p_{-1}. \tag{5.4}$$

Таблиця 5.2 – Таблиця відповідності складань розрядів накопичуючих двійкових кодів

x_1	x_0	p_{-1}	s	p
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

З виразу (5.2) виходить, що для отримання суми двох старших розрядів необхідно спочатку виконати операцію що «Виключає АБО» над початковими доданками *x1* і *x0* і потім ще одну операцію що «Виключає АБО» над результатом першої операції що «Виключає АБО» і сигналом перенесення з попереднього розряду. Для отримання сигналу перенесення також необхідно скористатися результатом операції що «Виключає АБО» над доданками *x1* і *x0.*

Класифікація суматорів може бути виконана по різних ознаках. Розглянемо ті, що найчастіше зустрічаються з них.

За кількістю виводів розрізняють: напівсуматори, накопичуючи однозарядні суматори, накопичуючи багаторозрядні суматори.

Напівсуматором називається пристрій, що призначений для складання двох кодів, має два входи і два виходи і що формує з сигналів вхідних доданків сигнали суми і перенесення в старший розряд.

Однозарядним суматором називається пристрій, що призначений для складання двох кодів, має три входи і два виходи, і що формує з сигналів вхідних доданків і сигналу перенесення з молодших розрядів сигнали суми і перенесення в старший розряд.

Багаторозрядним суматором називається пристрій, що призначений для складання двох накопичуючих кодів, формує на виході код суми і сигнал перенесення у випадку, якщо результат складання не може бути представлений кодом, розрядність якого співпадає з розрядністю кодів доданків.

У свою чергу, накопичуючи суматори підрозділяються на *послідовні* і *паралельні.* У послідовних суматорах операція складання виконується послідовно розряд за розрядом, починаючи з молодшого. У паралельних всі розряди вхідних кодів підсумовуються одночасно.

Розрізняють *комбінаційні* суматори – пристрої, що не мають власної пам'яті, і *накопичуючи* суматори, забезпечені власною внутрішньою пам'яттю, в якій акумулюються результати виконаної операції. При цьому кожен черговий доданок додається до значення, що вже було в пристрої.

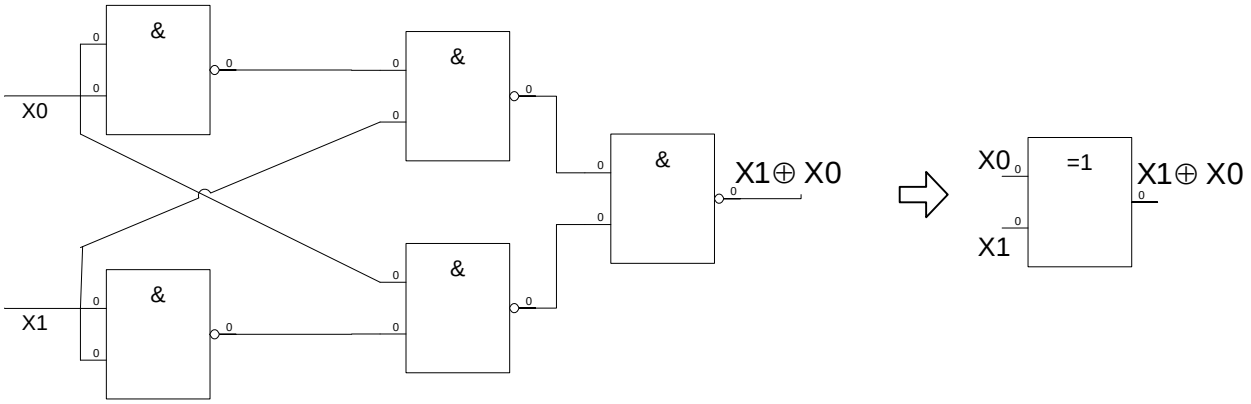
За способом тактування розрізняють синхронні і асинхронні суматори. *У синхронних* суматорах час виконання операції арифметичного підсумовування двох кодів не залежить від виду самих кодів і завжди залишається постійним. У *асинхронних* суматорах час виконання операції залежить від виду доданків. Тому після закінчення виконання підсумовування необхідно виробляти спеціальний сигнал завершення операції.

Залежно від використовуваної системи числення розрізняють двійкові, двійково-десяткові і інші типи суматорів.

Двійковий напівсуматор. Згідно визначенню, вихідні сигнали двійкового напівсуматора повинні відповідати системі логічних функцій (5.1) та (5.2). Для її технічної реалізації необхідні логічні елементи І і «виключне АБО». Оскільки раніше елемент «виключне АБО» не був описаний, розглянемо можливість його побудови на вже відомих елементах. Для цього перетворимо вираз (5.1) до базису І–НЕ.

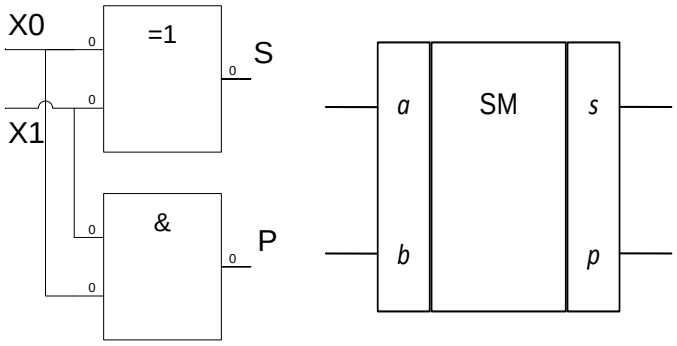
$$s = x_1 \oplus x_0 = \overline{x_1} x_0 + x_1 \overline{x_0} = \overline{\overline{x_1 x_0} + x_1 \overline{x_0}} = \overline{\overline{x_1 x_0} \cdot x_1 \overline{x_0}} = \left(\overline{x_1} \mid x_0\right) \mid \left(x_1 \mid \overline{x_0}\right)$$

Технічна реалізація отриманого виразу наведена на рис. 5.1. На цьому ж малюнку показано умовне позначення елементу «виключне АБО».



а *б*

Рисунок 5.1 – Структурна схема реалізації операції «виключне АБО» (*а*) та її умовне позначення (*б*)



а *б*

Рисунок 5.2 – Напівсуматор (*а*) та його умовне позначення (*б*)

З використанням сказаного легко можна синтезувати логічну схему двійкового напівсуматора (рис. 5.2, а). Умовне графічне зображення двійкового напівсуматора показане на рис. 5.2, б.

Двійкові суматори. Однорозрядний суматор.

Функціонування однорозрядного суматора визначається системою ЛФ (5.2) та (5.3). Технічна реалізація даної ЛФ може бути виконана на ЛЕ будь–якого типу. Розглянемо, наприклад, побудову однорозрядного суматора з використанням схем двійкових напівсуматорів (рис. 5.3, а). Очевидно, що для цієї мети необхідно два напівсуматори та елемент АБО.

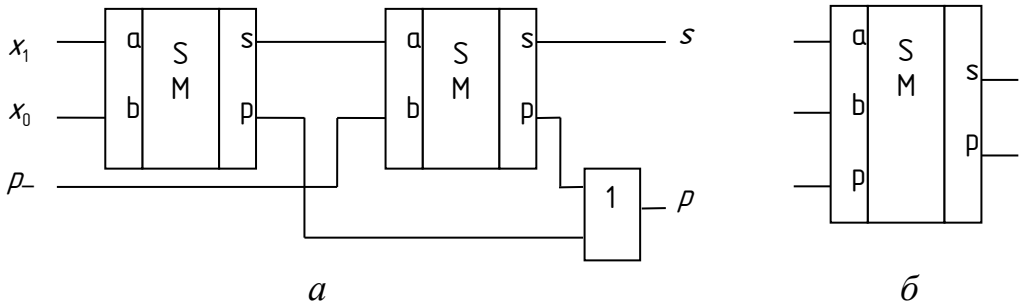


Рисунок 5.3 – Однорозрядний суматор (*а*) та його умовне позначення (*б*)

Слід зазначити, що якщо синтезувати схему однорозрядного суматора безпосередньо по табл. 5.2, щодо елементарних ЛЕ можна отримати простіше технічне рішення. Умовне графічне позначення однорозрядного суматора наведене на рис. 5.3, б.

Багаторозрядний суматор паралельної дії. У цьому суматорі (рис. 5.4) операції підсумовування повинні виконуватися одночасно по всіх розрядах початкових двійкових чисел. З цього виходить, що такий суматор повинен мати окремі апаратні засоби для виконання підсумовування в кожному розряді.

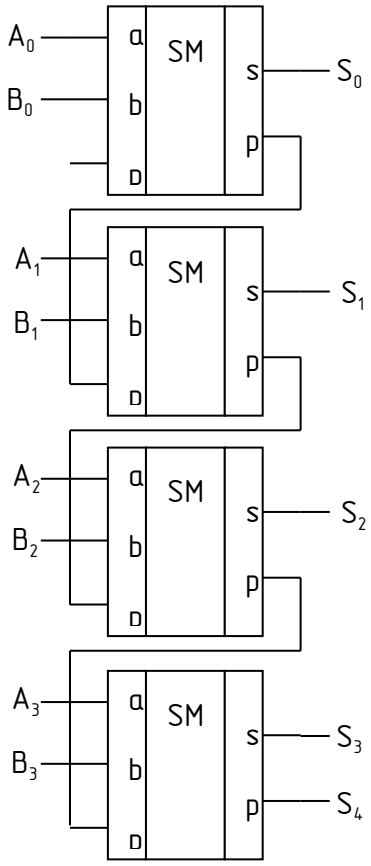


Рисунок 5.4 – Багаторозрядний суматор паралельної дії

Для отримання на виході сигналу, рівного реальній сумі вхідних кодів, необхідно, щоб сигнал перенесення послідовно сформувався на виходах суматорів всіх розрядів. Отже, не залежно від того, що для підсумовування в кожному розряді використовується окремий суматор, реальний час виконання операції в даній схемі визначається послідовним перенесенням сигналу *p* з розряду в розряд. Тому результат, який може бути знятий з виходу схеми через час, рівний часу підсумовування в одному розряді, не буде реальним значенням шуканої суми.

Двійково-десяткові суматори.

Окрім двійкових, в обчислювальній техніці часто використовуються так звані двійково-десяткові коди. Вони відображають виражені у виді послідовності двійкових розрядів десяткові числа. Очевидно, що для представлення десяткових цифр необхідний, як мінімум, 4–розрядний двійковий код. При цьому з 16 можливих його комбінацій використовується тільки 10. Це припускає розробку великого числа різних двійково-десяткових кодів.

На практиці великого поширення набув клас так званих *зважених кодів.* У цих кодах кожному розряду двійкового числа приписується цілком певний ваговий коефіцієнт. Як приклад в табл. 5.3 наведена відповідність десяткових чисел і їх двійкових і двійково-десяткових еквівалентів. Вагові коефіцієнти його двійкових розрядів відповідно рівні 8, 4, 2, 1.

З наведеної таблиці 5.3 виходить, що 4–розрядні двійкові коди з 1010 по 1111 не мають 4–розрядного двійково-десяткового еквівалента. Так, число 12 в двійково-десятковому коді представляється 8–розрядним кодом 00010010, а число 16 – кодом 00010110.

Описана особливість двійково-десяткового коду припускає використання для підсумовування спеціальних логічних схем. Сенс їх побудови полягає в тому, що спочатку двійково-десяткові коди підсумовуються як двійкові. Якщо результатом підсумовування є неіснуючий двійково-десятковий код, його необхідно зменшити на 10, і додатково сформувати сигнал перенесення. Зменшення коду на 10 може виконуватися його підсумовуванням з додатковим кодом числа 10 (0110).

Таблиця 5.3 – Двійково-десятковий код

Двійковий код $x_3x_2x_1x_0$	Двійково-десятковий код	Десяткове число
0 0 0 0	0 0 0 0 0 0 0 0	00
0 0 0 1	0 0 0 0 0 0 0 1	01
0 0 1 0	0 0 0 0 0 0 1 0	02
0 0 1 1	0 0 0 0 0 0 1 1	03
0 1 0 0	0 0 0 0 0 1 0 0	04
0 1 0 1	0 0 0 0 0 1 0 1	05
0 1 1 0	0 0 0 0 0 1 1 0	06
0 1 1 1	0 0 0 0 0 1 1 1	07
1 0 0 0	0 0 0 0 1 0 0 0	08
1 0 0 1	0 0 0 0 1 0 0 1	09
1 0 1 0	0 0 0 1 0 0 0 0	10
1 0 1 1	0 0 0 1 0 0 0 1	11
1 1 0 0	0 0 0 1 0 0 1 0	12
1 1 0 1	0 0 0 1 0 0 1 1	13
1 1 1 0	0 0 0 1 0 1 0 0	14
1 1 1 1	0 0 0 1 0 1 0 1	15

Необхідність виконання такого підсумовування згідно табл. 5.3 виражається ЛФ . Очевидно, що таке ж підсумовування необхідно виконувати і у випадку, якщо в результаті першого підсумовування отриманий сигнал перенесення в старший розряд. З урахуванням сказаного, ЛФ необхідності виконання додаткового підсумовування має вид

$$F = x_3(x_1 + x_2) + P. \qquad (5.5)$$

Таким чином, для реалізації операції складання двох двійково-десяткових кодів необхідно два багаторозрядні суматори і логічну схему, що забезпечує формування вихідного сигналу у відповідність з ЛФ (5.4).

Приклад реалізації такого пристрою показаний на рис. 5.5.

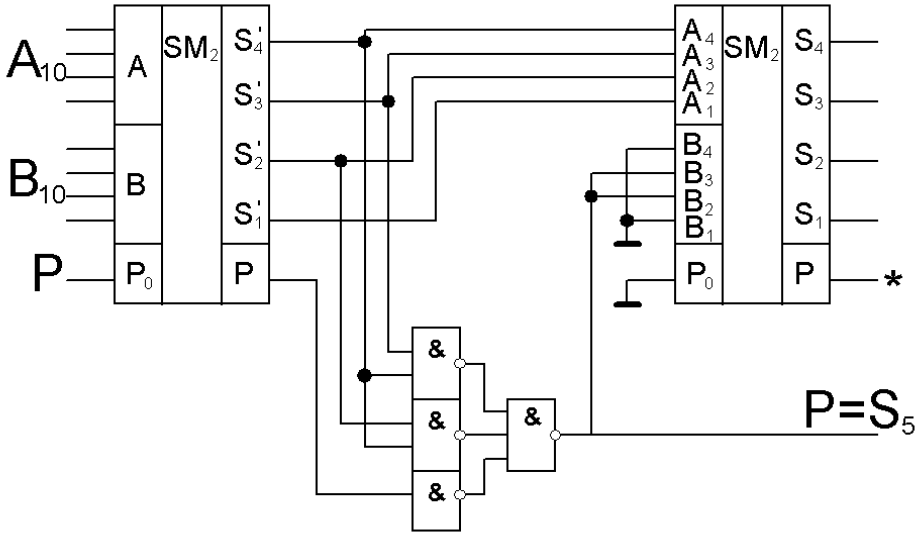


Рисунок 5.5 – Двійково-десятковий суматор

Чотирьохрозрядний суматор *DD1* виконує арифметичне складання початкових двійково-десяткових кодів. Логічна схема на елементах *DD2*, *DD3* і *DD4*, реалізовуючи ЛФ (5), визначає необхідність додаткового підсумовування виконуваного суматором *DD5*.

Приклад 1. Побудова однорозрядних схем суматорів

Напівсуматор.

Таблиця 5.4 – Таблиця відповідності складання двох однорозрядних двійкових кодів

x_1	x_0	s	p
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$s = \overline{x_1} x_0 + x_1 \overline{x_0} = x_1 \oplus x_0,$$

$$p = x_1 x_0.$$

Схема однорозрядного напівсуматора наведена на рис. 5.6.

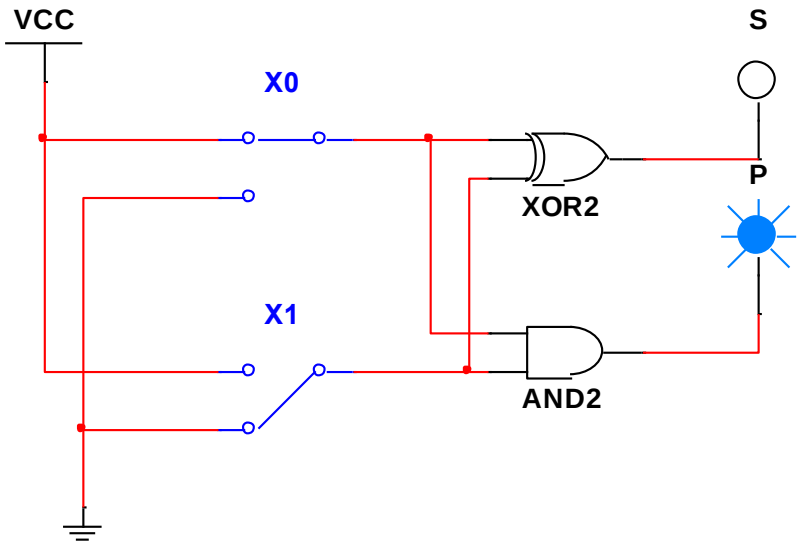


Рисунок 5.6 – Схема однорозрядного напівсуматора реалізованого в пакеті *Multisim*

Повний суматор

Таблиця 5.2 – Таблиця відповідності складань розрядів багаторозрядних двійкових кодів

x_1	x_0	p_{-1}	s	p
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

$$s=(x_1 \oplus x_0)\overline{p_1}+\overline{(x_1 \oplus x_0)}p_{-1},$$

$$p=x_1x_0+(x_1 \oplus x_0)p_{-1}.$$

Функціональна схема однорозрядного суматора наведена на рис. 5.7.

Принципова схема однорозрядного суматора наведена на рис.5.7.

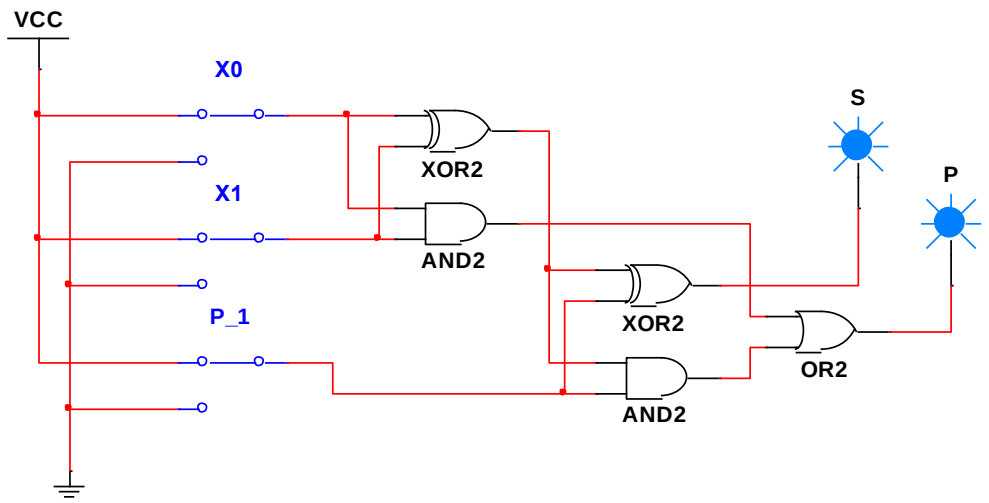


Рисунок 5.7 – Принципова схема однорозрядного суматора

Побудова багаторозрядних схем суматорів.

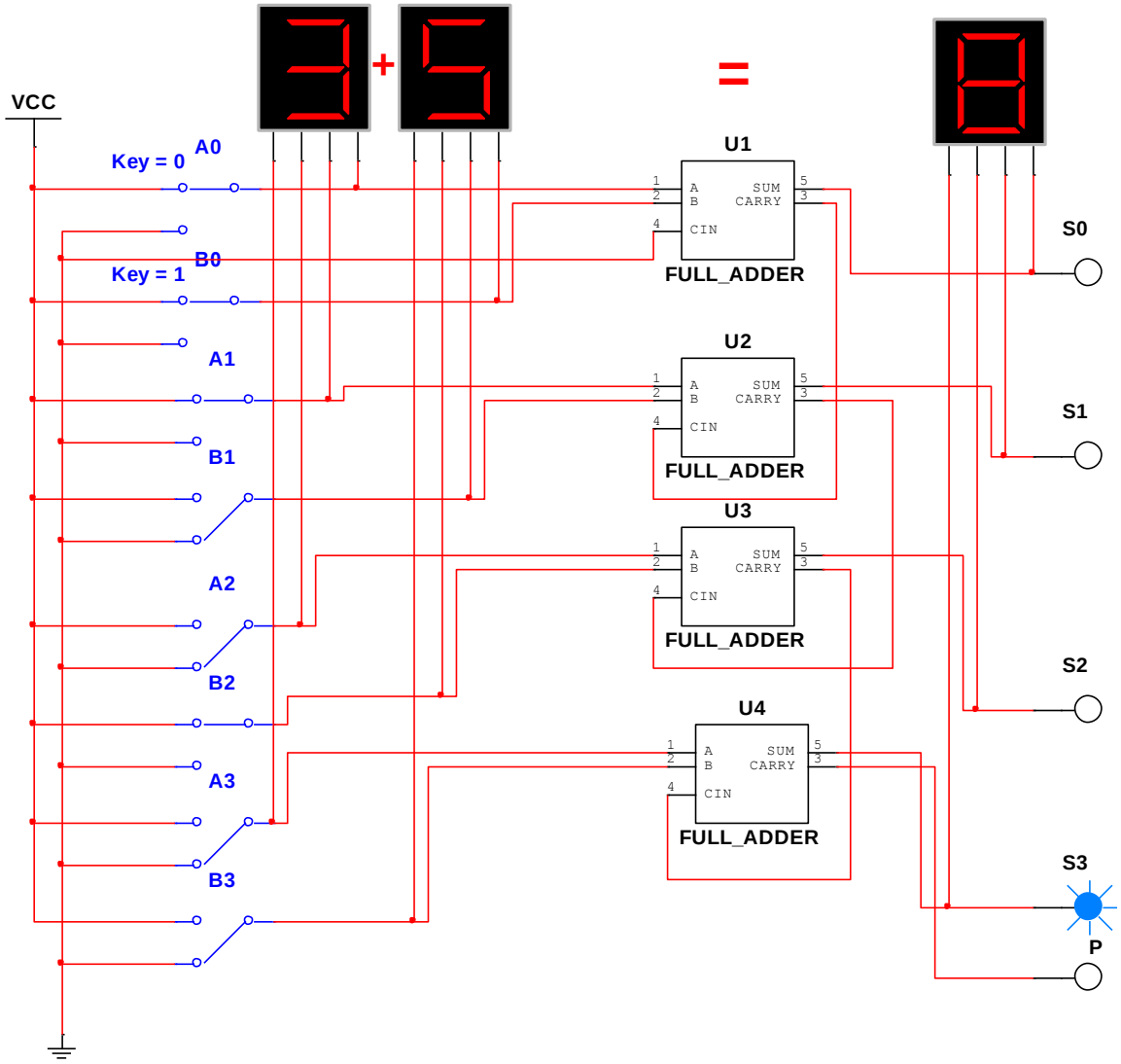


Рисунок 5.9 – Принципова схема багаторозрядного суматора паралельної дії на однорозрядних суматорах

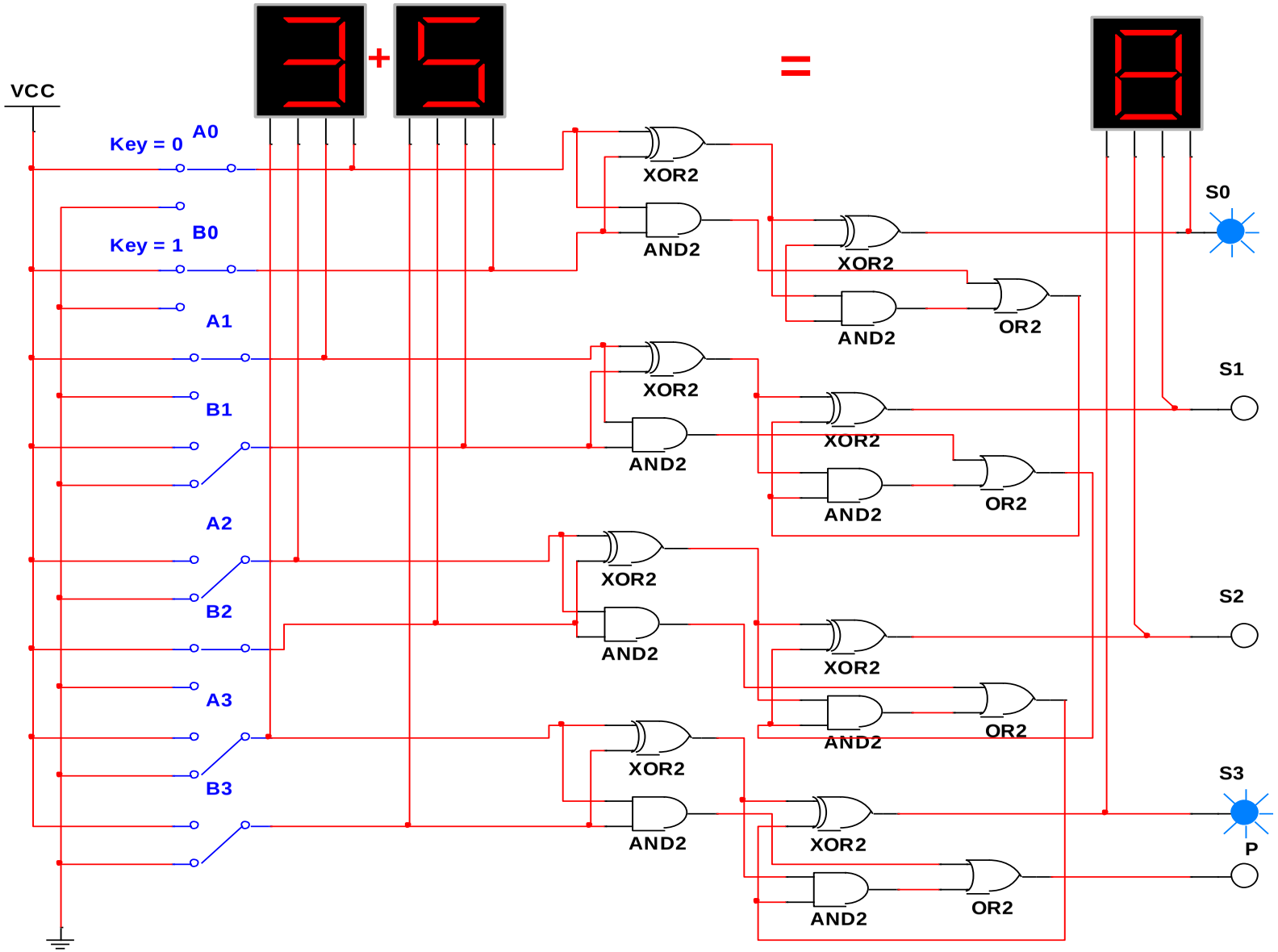


Рисунок 5.10 – Принципова схема багаторозрядного суматора паралельної дії на логічних елементах

Завдання до практичної роботи

1. Вивчити теоретичні відомості щодо роботи напівсуматорів, двійкових суматорів та двійково-десяткових суматорів.

2. За допомогою програми *Multisim* побудувати схему однорозрядного напівсуматора (рис. 5.6), схему однорозрядного суматора (рис. 5.7) та дослідити їх роботу.

3. Побудувати принципову схему багаторозрядного суматора паралельної дії на логічних елементах (рис. 5.9) та за допомогою програми *Multisim* досліди її роботу.

4. Побудувати принципову схему багаторозрядного суматора паралельної дії на однорозрядних суматорах (рис. 5.11) та за допомогою програми *Multisim* досліди її роботу.

5. Зробити висновки та оформити звіт з практичного заняття.

Контрольні запитання

1. Що таке акумулятор?

2. Які класифікації суматорів вам відомі?

3. Намалюйте схему і поясніть принцип роботи послідовного суматора.

4. Намалюйте схему та поясніть принцип роботи паралельного суматора.

5. Які принципові переваги і недоліки мають послідовні і паралельні суматори

6 ЦИФРОВІ КОМПАРАТОРИ

Мета: вивчення правил виконання операції порівняння двійкових чисел і дослідження принципів побудови цифрових компараторів.

Теоретичні відомості

Компаратором (пристроєм порівняння) називають функціональний вузол, що забезпечує порівняння двох чисел А і В. Якщо А і В – n–розрядні двійкові числа, то компаратор називають цифровим [4–6].

Найпростіші компаратори формують на виході однобітний сигнал рівності або нерівності порівнюваних чисел А і В. Ці відносини використовуються, як логічні умови в пристроях контролю та діагностики ЕОМ, в пристроях автоматики. Компаратори використовуються для сигналізації про вихід величин за встановлені межі і т.п .

Компаратори будуються на основі порозрядних операцій над однойменними розрядами обох слів. Слова рівні, якщо попарно рівні всі однойменні їх розряди. Ознака (умова) рівності і–х розрядів порівнюваних слів *A* і *B*:

$$r_{ip} = a_i b_i + \bar{a}_i \bar{b}_i = \overline{a_i \bar{b}_i + \bar{a}_i b_i} = \overline{a_i \oplus b_i} = 1 \quad (6.1)$$

Умова нерівності їх розрядів:

$$r_{iн} = a_i \bar{b}_i + \bar{a}_i b_i = a_i \oplus b_i = 1 \quad (6.2)$$

Схемних реалізація наведених умов зображена на рис. 6.1, а.

Схема *n*-розрядного компаратора на рівність показана на рис.6.1, б.

Більш складні компаратори виявляють не тільки факт рівності двох *n*–розрядних чисел, але і порівнюють числа за значенням. Такі компаратори мають три виходи: $A > B$, $A = B$, $A < B$ і в залежності від співвідношення величин *A* і *B* активний рівень (– рівень логічної 1) з'являється на одному з цих виходів.

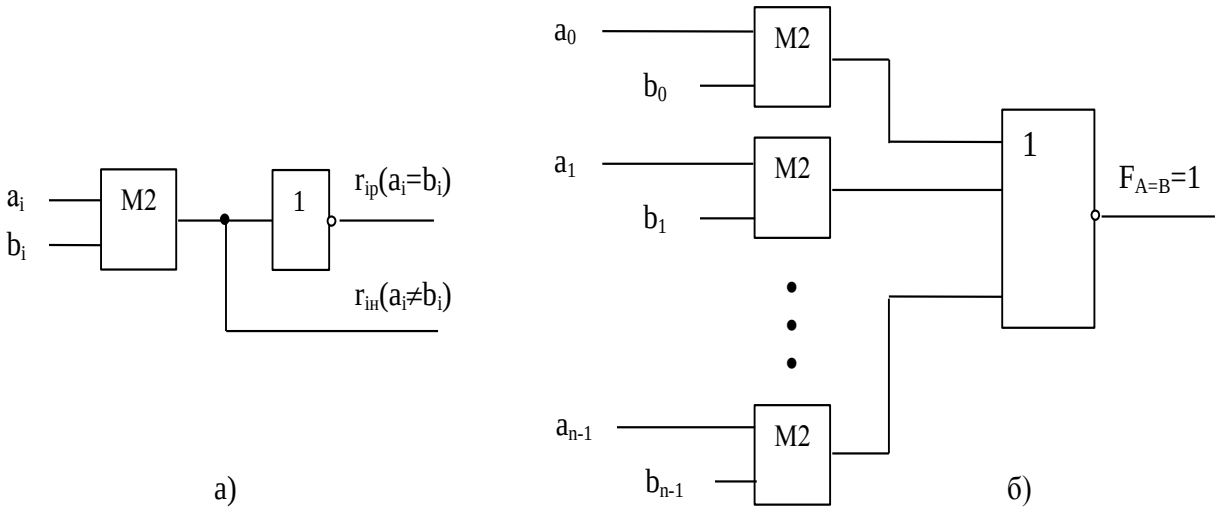


Рисунок 6.1 – Схема компаратора 2–х змінних (а) і *n* – розрядних слів (б)

Побудувати такий компаратор можна на базі двійкового суматора, виконавши на ньому операцію віднімання *A–B* і проаналізувавши отриманий результат. Для цього на суматор потрібно число *B* подати в додатковому коді. Тоді вихідний перенос суматора (P_1) буде дорівнює 0 лише в тому випадку, коли *A* строго менше *B*. Рівність різниці 0 є ознакою того, що *A* = *B*. Одиниця перенесення при нульовій сумі вказує на те, що *A* строго більше *B*. Сказане ілюструє такі приклади:

A> B	A = B	A <B
A 13	A 12	A 11
$-B$ 12	$-B$ 12	$-B$ 12
1101	1100	1011
+ 0100	+ 0100	+ 0100
1.0001	1.0000	0.1111
/		/
$P_1 = 1$ $s \neq 0$	$s = 0$	$P_1 = 0$

Примітка. Віднімання з числа *A* числа *B* = 12_{10} = 1100_2 замінено додатком до *A* додаткового коду числа *B*, рівного 0100_2.

Правила справедливі, якщо числа *A* і *B* розглядаються, як позитивні величини, без знака. Якщо ж їх старші розряди трактуються, як знаки, то правила

будуть дещо інші. Їх легко вивести самостійно, якщо є навички поводження зі зворотними (оберненими) і додатковими кодами.

Схема, що реалізує цей алгоритм, зображена на рис. 4.2.

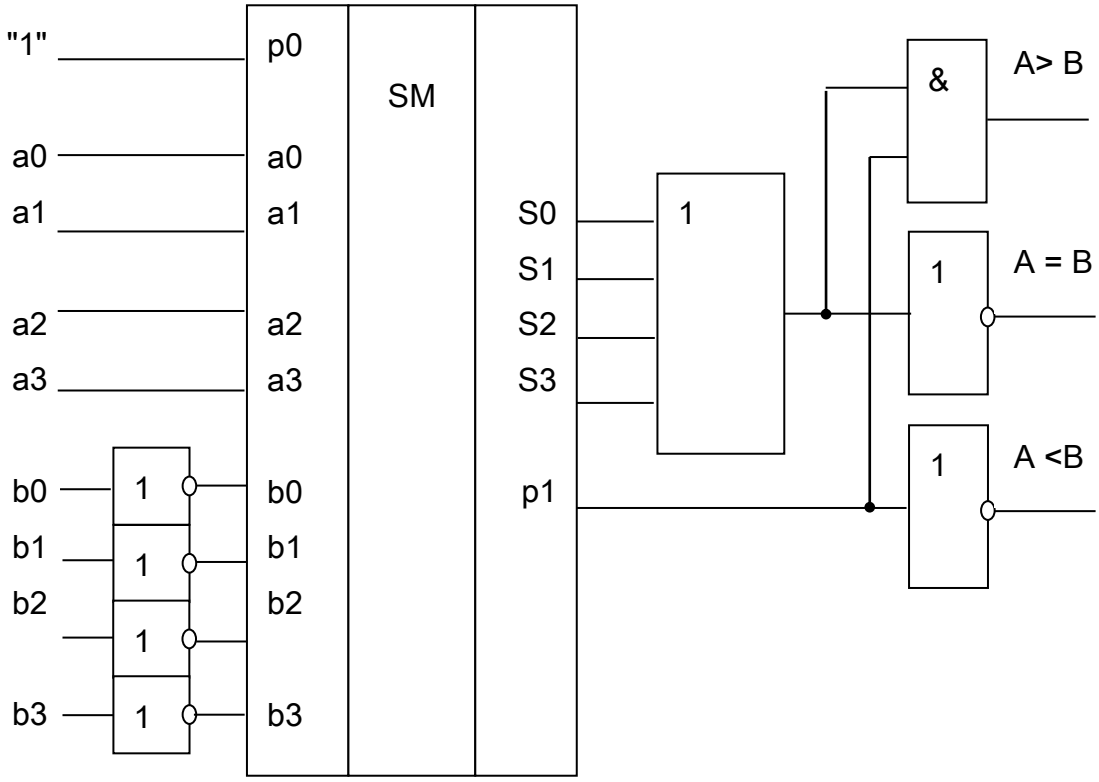


Рисунок 6.2 – Схема 4–розрядного компаратора на базі двійкового суматора

Прикладом компаратора двійково–кодованих чисел може служити ІС 4–розрядного компаратора К555СП1 (*74LS85*) (рис. 6.3). Компаратор має 11 входів. Чотири пари входів *ai bi* (*i* = 0,1,2,3) використовуються для подачі на них відповідних розрядів порівнюваних чисел, входи *A* < *B, A* = *B, A* > *B* дозволяють каскадувати кілька ІС компараторів для збільшення розрядності порівнюваних чисел. Компаратор має три виходи результатів порівняння: *A* > *B, A* = *B* і *A* <*B.* При каскадувані виходи *A*>*B, A* = *B* і *A*<*B* схеми, що порівнює молодші розряди, слід приєднати до однойменних входів наступного каскаду. Цим способом за допомогою двох компараторів СП1 можна порівнювати два восьмирозрядних слова. Неважко підрахувати необхідне число каскадів для будь–якої більшої довжини порівнюваних слів.

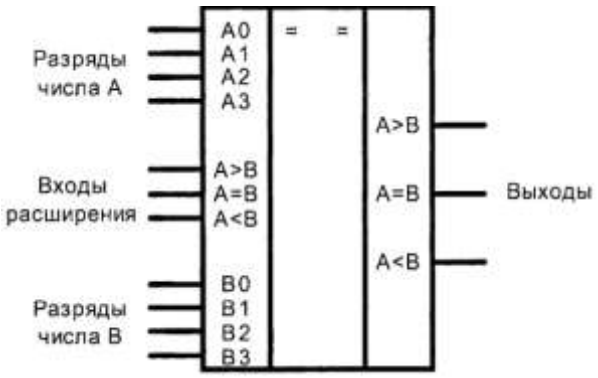


Рисунок 6.3 – Умовне зображення ІС компаратора К555СП1

Всі можливі комбінації порозрядних співвідношень вхідних кодів, а також рівнів на входах каскадування зведені в таблицю, де показані відповідні результуючі рівні на виходах $A > B$, $A = B$ і $A < B$ (табл. 6.1).

Таблиця 6.1– Таблиця істинності компаратора

	Входи порівняння даних				Входи нарощування			Виходи		
	A3, B3	*A2, B2*	*A1, B1*	*A0, B0*	*I(A>B)*	*I(A<B)*	*I(A=B)*	*A>B*	*A<B*	*A=B*
1.	*A3>B3*	x	x	x	x	x	x	1	0	0
2.	*A3<B3*	x	x	x	x	x	x	0	1	0
3.	*A3=13*	*A2>B2*	x	x	x	x	x	1	0	0
4.	*A3=B3*	*A2<B2*	x	x	x	x	x	0	1	0
5.	*A3=B3*	*A2=B2*	*A1>B1*	x	x	x	x	1	0	0
6.	*A3=B3*	*A2=B2*	*A1<B1*	x	x	x	x	0	1	0
7.	*A3=B3*	*A2=B2*	*A1=B1*	*A0>B0*	x	x	x	1	0	0
8.	*A3=B3*	*A2=B2*	*A1=B1*	*A0<B0*	x	x	x	0	1	0
9.	*A3=B3*	*A2=B2*	*A1=B1*	*A0=B0*	0	0	1	0	0	1
10.	*A3=B3*	*A2=B2*	*A1=B1*	*A0=B0*	0	0	1	0	0	1
11.	*A3=B3*	*A2=B2*	*A1=B1*	*A0=B0*	1	0	0	1	0	0
12.	*A3=B3*	*A2=B2*	*A1=B1*	*A0=B0*	0	1	0	0	1	0
13.	*A3=B3*	*A2=B2*	*A1=B1*	*A0=B0*	1	1	0	0	0	1
14.	*A3=B3*	*A2=B2*	*A1=B1*	*A0=B0*	1	1	1	1	1	1

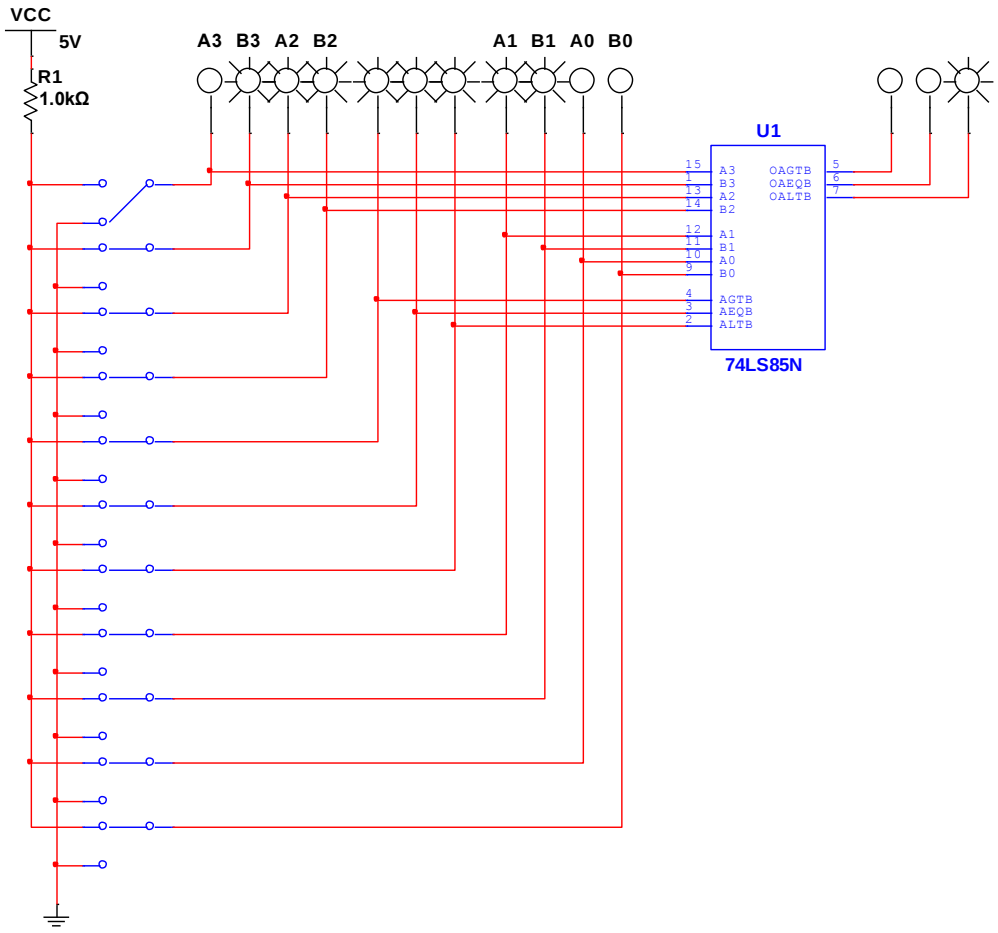


Рисунок 6.4 – Схема дослідження функціонування компаратора на 74*LS*85 в пакеті *Multisim*

Завдання до практичної роботи

1. Вивчити теоретичні відомості щодо роботи компараторів.

2. Провести дослідження функціонування компаратора К555СП1 (74*LS*85).

3. Зібрати сему для дослідження компаратора КС555СП1 в статистичному режимі роботи згідно рис.6. 4.

4. Дослідити реакцію компаратора КС555СП1 на статистичні комбінації логічних сигналів вхідного коду. Реакцію на виводах $A > B$, $A < B$, $A = B$ компаратора КС555СП1 визначати на світлодіодах індикаторах.

Результати дослідження функціонування компаратора К555СП1 занести до таблиці 6.2.

Таблиця 6.2 – Результати дослідження функціонування компаратора К555СП1

Данні, що порівнюються				Входи нарощування			Виходи		
A3, B3	*A2, B2*	*A1, B1*	*A0, B0*	*I(A>B)*	*I(A<B)*	*I(A=B)*	*A>B*	*A<B*	*A=B*

5. Варіанти індивідуальних завдань наведені в таблиці 6.3.

Таблиця 6.3 – Варіанти індивідуальних завдань

Варіант	A	B	I(A>B)	I(A<B)	I(A=B)
1	5	8	1	0	0
2	5	A	0	1	0
3	9	E	1	1	0
4	F	A	x	x	x
5	E	9	0	1	0
6	C	D	x	x	x
7	7	D	1	0	1
8	E	F	x	x	x
9	E	5	0	1	0
10	1	A	x	x	x
11	B	2	1	1	0
12	A	A	0	0	0
13	E	A	x	x	x
14	4	F	0	1	0
15	9	F	x	x	x
16	F	5	x	x	x
17	C	3	x	x	x
18	B	2	0	0	1

Контрольні запитання

1. Наведіть визначення цифрового компаратора і перерахуйте його можливі застосування.

2. Запишіть умови рівності (нерівності) однойменних розрядів порівнюваних чисел *A* і *B*.

3. Доведіть справедливість виразів (6.1) і (6.2).

4. Чому дорівнює значення виходу схеми (рис. 6.1, б) при а) $A = B$, б) $A < B$ і в) $A > B$?

7 СИНТЕЗ І АНАЛІЗ РОБОТИ ПЕРЕТВОРЮВАЧА КОДІВ

Мета: вивчення принципів побудови і методів синтезу перетворювачів двійковій-десяткових кодів (ДДК).

Теоретичні відомості

Перетворювачем коду називається функціональний вузол комп'ютера, призначений для перетворення двійкового коду з однієї форми в іншу [4–6].

Для подання інформації використовують різноманітні двійкові та двійково-десяткові коди: прямий, обернений, доповняльний і їхні модифікації, циклічний з лишком три та інші. Існує велика кількість кодів, які забезпечують:

– простоту виконання арифметико-логічних операцій;

– зручність переведення чисел з десяткової системи в двійковий код;

– надійність виконання заданих алгоритмів функціонування і ефективний контроль результатів обчислень;

– зменшення апаратних витрат при побудові цифрових пристроїв. До перетворювачів коду відносяться шифратори і дешифратори, однак за традицією ці функціональні вузли виділені в окремі самостійні класи.

Двійковій-десятковий код 8421 отримав найбільш широке застосування; в цьому коді десяткові цифри 0,1 ..., 9 зображуються чотирирозрядний двійковими числами – тетрадами 0000 0001 ... .. 1001 відповідно. Цифри 8,4,2,1 в позначенні коду – це ваги розрядів двійковій тетради. Двійкові тетради 1010,1011, 1100,1101, 1110 і 1111 не використовуються для зображення десяткових цифр, тому є надлишковими і називаються псевдотетрадами. На прикладі перетворювача ДДК 8421 в ДДК 2421 (код Айкена) поясняються основні етапи синтезу.

У таблиці 7.1 наведено кодування десяткових цифр в зазначених ДДК.

Таблиця 7.1 – Кодування десяткових цифр

Десяткові цифри	Двійково-десятковий код 8421				Двійково-десятковий код 2421			
	ваги розрядів				ваги розрядів			
	8	4	2	1	2	4	2	1
	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	0
3	0	0	1	1	0	0	1	1
4	0	1	0	0	0	1	0	0
5	0	1	0	1	1	0	1	1
6	0	1	1	0	1	1	0	0
7	0	1	1	1	1	1	0	1
8	1	0	0	0	1	1	1	0
9	1	0	0	1	1	1	1	1

Для перетворення ДДК 8421 в ДДК 2421 і побудови схеми перетворювача з табл.7.1 потрібно знайти функції:

B0 = f0 (A3, A2, A1, A0);

B1 = f1 (A3, A2, A1, A0);

B2 = f2 (A3, A2, A1, A0);

B3 = f3 (A3, A2, A1, A0).

Мінімізувати функції, застосовуючи один з методів (метод Квайна, метод Квайна із застосуванням карт Карно і ін.) з урахуванням надлишкових (зайвих) наборів змінних *A3, A2, A1, A0.* Мінімізовані функції перетворити в базис функції І–НЕ і скласти схему на логічних елементах І–НЕ.

Для перетворення ДДК 2421 в ДДК 8421, тобто для зворотного перетворення, і побудови схеми перетворювача з табл. 7.1 потрібно знайти функції:

A0 = φ0 (B3, B2, B1, B0),

A1 = φ1 (B3, B2, B1, B0),

A2 = φ2 (B3, B2, B1, B0),

A3 = φ3 (B3, B2, B1, B0).

Мінімізувати функції, перетворити їх в базис функції І–НЕ і скласти схему на логічних елементах І–НЕ.

Завдання до практичної роботи

1. Варіанти завдання двійковій-десяткових кодів, в які потрібно перетворити ДДК 8421 і виконати зворотні перетворення, наведені в табл. 7.2. Порядковий номер прізвища студента у списку групи є номером варіанту.

Таблиця 7.2 – Двійковій-десяткових кодів, в які потрібно перетворити ДДК 8421

Варіант	Десяткові номери довічних наборів змінних, що зображують десяткові цифри 0,1, ..., 9
1	3, 4,5,6,7,8,9,10,11,12
2	0,1,2,3,5,10,12,13,14,15
3	0,1,4,5,7,8,10,12,14,15
4	0,1,2,3,4,5,8,9,10,11
5	0,1,2,3,4,5,6,8,9,10
6	0,1,2,3,6,9,12,13,14,15
7	5,6,7,8,9,10,11,12,13,14
8	0,1,2,3,4,8,9,10,11,12,
9	0,1,3,4,5,7,8,10,11,12
10	0,1,2,4,5,6,7,8,9,10
11	0,1,2,3,4,5,6,7,12,13
12	0,1,2,3,7,8,12,13,14,15
13	0,1,2,4,5,6,8,9,10,12
14	2,3,4,5,6,7,8,9,10,11
15	0,1,3,4,5,7,11,12,13,15
16	0,1,2,3,5,6,9,10,12,13

2. Виконати синтез схеми перетворювача коду. Результати синтезу уявити в базисі І–НЕ.

3. Виконати синтез схеми двійковій-десяткового лічильника на *JK*–тригерах

за безвентельною схемою з природним порядком зміни станів; скомутувати схеми лічильника і перетворювача коду; вихідні сигнали лічильника використовувати в якості змінних *A3, A2, A1, A0*.

4. Дослідити схему перетворювача коду в статичному і динамічному режимах. У статичному режимі сигнали вихідні сигнали лічильника використовувати в якості змінних *A3, A2, A1, A0.* У статичному режимі сигнали на вхід лічильника подавати від ключа, в динамічному режимі – від генератора імпульсів.

Провести аналіз роботи перетворювача коду по таблиці істинності і часові діаграмі вхідних і вихідних сигналів перетворювача коду.

Контрольні запитання

1. Як визначаються надлишкові набори змінних?

2. Як враховуються надлишкові набори змінних при мінімізації функцій алгебри логіки?

3. Представлення чисел в двійково-десятковому коді (8421)+3. Особливості цього коду.

4. Правила переводу чисел, що представлені в двійково-десятковому коді.

5. Представлення чисел в двійково-десятковому коді Айкена. Особливості цього коду.

6. Що таке перетворювач коду? Етапи його синтезу.

8 ТРИГЕРИ ТА ЇХ ЧАСОВІ ДІАГРАМИ

Мета: закріпити теоретичні знання, ознайомитися з принципом дії тригерів та їх побудови їх часових діаграм.

Теоретичні відомості

Тригер — це цифровий пристрій, який має два стани стійкої рівноваги на виході. Перехід з одного стану в інший відбувається під впливом зовнішніх сигналів. Зміна стану тригера здійснюється стрибком. Не всі зовнішні сигнали можуть визивати змін стану тригера. Тому тригери називають найпростішим елементом пам'яті [4–6].

Тригери можуть бути: як одно–, так і двоступеві; як асинхронні, так і синхронні; з лічильним входом і універсальні; зі статичним і динамічним управлінням. Найбільшого поширення набули тригери *RS, D, JK* типів.

Будь-який тригер має два виходи — прямий і інверсний. Коли характеризують роботу тригера зазвичай говорять про прямий вихід, оскільки на іншому виході сигнал завжди буде інверсний.

Робота будь-якого тригера може бути добре зрозуміла за його часовими діаграмами.

Тригер *RS* – типу має два інформаційні входи *R* і *S*. Тригер *RS* – типу може бути реалізований на двох логічних елементах типу «2І–НЕ»" (рис. 8.1). У пристроях на основі *RS* – тригерів є заборонена комбінація вхідних сигналів, яку слід уникати. Для тригера, наведеного на рис. 8.1 така заборонена комбінація сигналів виникає, коли на обидва входи подано два однакових сигнали, рівень яких становить лог. «0».

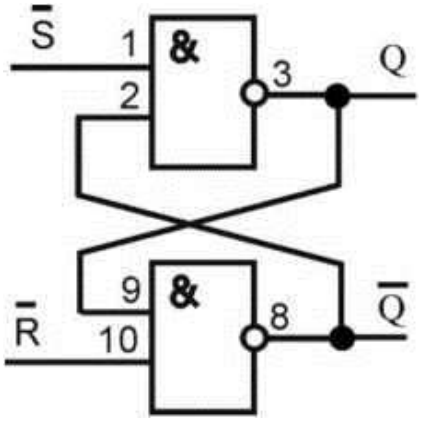


Рисунок 8.1 — Схема підключення виводів мікросхеми КР1533ЛА3 для створення асинхронного *RS* – тригера на логічних елементах

На рис. 8.2 подана часова діаграма роботи асинхронного *RS* – тригера, наведеного на рис. 8.1. При включенні живлення тригер може встановитися у будь-який стан. На рис. 8.2 вихід *Q* у початковий момент часу знаходиться у стані лот. «1». У час *t1* на *R* вхід тригера подається сигнал низького рівня, а для цього тригера саме він є керуючим, то тригер встановиться у стан лот. «0», у момент часу *t2* дія керуючого сигналу на *R* вході закінчиться, але тригер не змінить свого стану і зберігатиме його до моменту *t3*, коли вже на інший вхід прийде інший керуючий сигнал. У момент часу *t4*, коли закінчиться дія керуючого сигналу на вході *S*, стан тригера також не зміниться, тобто тригер залишатиметься у попередньому стані до часу *t5*, а потім до часу *t7*.

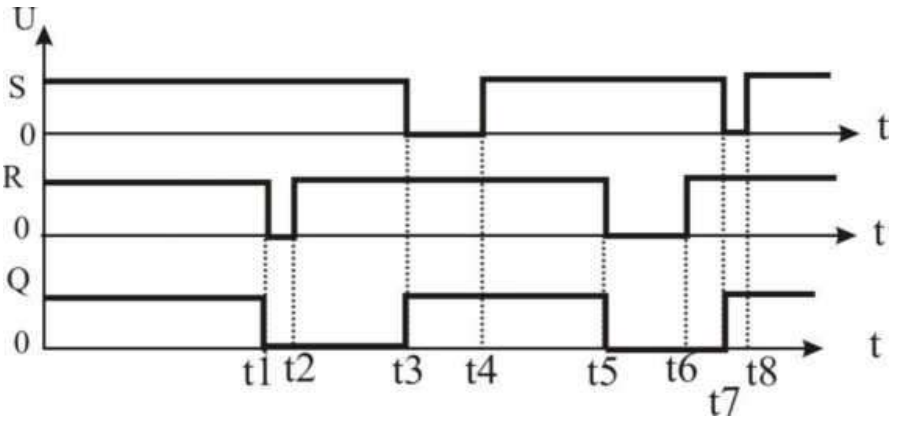


Рисунок 8.2 – Часові діаграми роботи RS– тригеру

Тригер *D* – типу є одним з найбільш використовуваних. Тригер *D* – типу називають також тригером з динамічним керуванням, він має інформаційний

вхід *D* та вхід синхронізації *C* (тактовий). У такому тригері інформація фіксується у той момент, коли тактовий сигнал змінює низький рівень на високий. Зазвичай тригери *D* – типу мають також входи *R* і *S*. Входи *R* і *S* мають перевагу над усіма іншими входами.

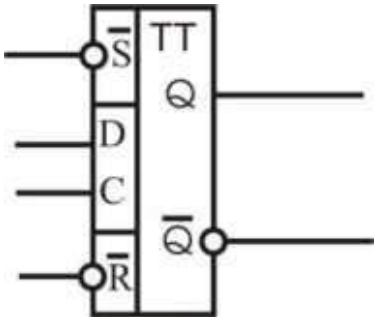


Рисунок 8.3 — Тригер *D*– типу (типу ТМ2), що має *RS* – входи

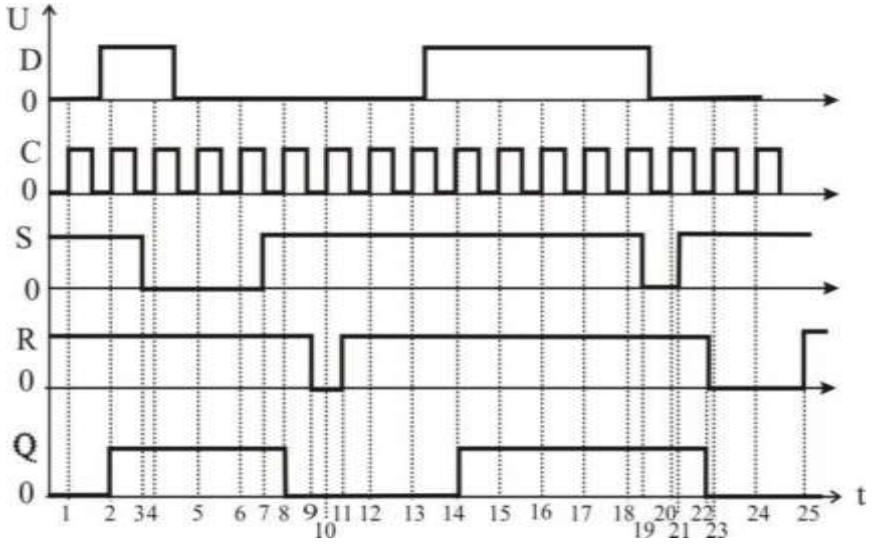


Рисунок 8.4 — Часові діаграми, які описують роботу тригеру *D* – типу

Для тригеру *D*– типу (рис. 8.3) часові діаграми мають вид поданий на рис. 8.4. Коли на керуючих входах *S* та *R* відсутній керуючий сигнал, то тригер працює за інформацією на вході *D*, причому переключення тригеру у стан *D* відбуватиметься по приходу переднього фронту тактового імпульсу на вході *C*. Так у момент часу t1 тригер підтвердить стан у якому знаходився до цього часу, у момент часу *t2* тригер переключиться у стан лог. «1», у час *t3* прийде сигнал на керуючий вхід *S*, тому з моменту часу *t3* до моменту часу *t7* тригер утримуватиметься у стані лог. «1» і не реагуватиме на зміну стану входу *D* та

прихід тактових імпульсів. Стан тригеру зміниться лише у момент часу *t8*, коли тактовий імпульс встановить на виході той сигнал, який є присутнім на вході D, тобто лог. «0». З часу *t9* до часу *t11* стан тригера визначатиме керуючий сигнал на вході R, у моменти часу *t12* та t13 переключення тригера не відбудеться, оскільки на інформаційному вході тригера присутній сигнал лог. «0», лише у момент часу *t14* стан тригеру зміниться відповідно до сигналу на вході *D* і у відповідності до цього сигналу на виході тригера утримуватиметься лог. «1» до моменту часу *t19*, ця ж лог. «1» з моменту часу *t19* до часу *t21* утримуватиметься відповідно до дії керуючого сигналу на вході S і лише у момент часу *t22* тригер переключиться у стан лог. «0» відповідно до керуючого сигналу на вході *R*.

Особливу цікавість становить включення тригеру *D* – типу зі зворотним зв’язком, коли його інформаційний вхід *D* підключено до інверсного виходу (рис. 8.5). Часові діаграми роботи тригера у такому включенні подані на рисунку 8.6. На рисунку показано, що зміна сигналу на виході тригера завжди відбувається з деякою мізерною затримкою, саме ця затримка забезпечує присутність попереднього стану на *D* вході тригера, а відповідно тригер буде встановлюватись у той стан, у якому знаходиться вхід *D*.

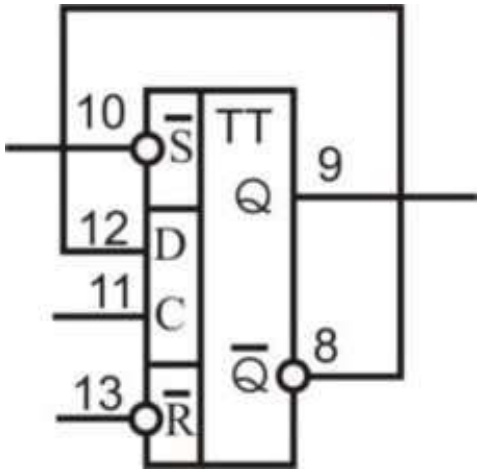


Рисунок 8. 5 — Включення *D* – тригеру у режимі ділення частоти на два

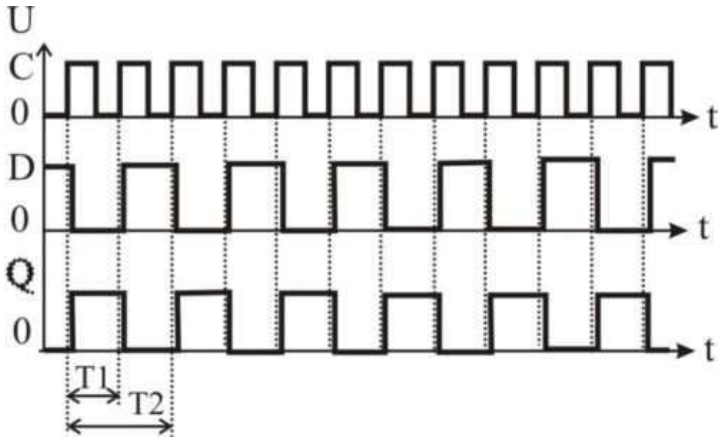


Рисунок 8.6 — Часові діаграми роботи *D* – тригеру у режимі ділення частоти навпіл

В результаті отримаємо, що період вхідного сигналу становить *T1* (між двома передніми фронтами тактового імпульсу), а період вихідного сигналу на виході *Q* становить *T2*. Тобто таке включення тригеру дозволяє виконувати ділення вхідної частоти на 2. Саме ця властивість використовується при побудові лічильників.

JK–тригер. Тригер *JK* – типу належить до універсальних. Він може мати *R* і *S* входи, як *RS*–тригер. Тригер *JK* – типу має два інформаційні входи *J* і *K*, а також тактовий вхід синхронізації *C*. Лог. «1» на вході *J* інформує про те, що тригеру слід переключитися у стан лог. «1» (на виході); лог. «1» на вході *K* інформує про те, що тригеру слід переключитись у стан лог. «0» (на виході). З надходженням тактового імпульсу на вхід *C* тригер переключатиметься відповідно до стану інформаційних входів *J* і *K*, але на відміну від тригера *D* – типу він переключатиметься, коли на вході С сигнал змінюватиметься з лог. «1» на лог. «0», ще кажуть, що такий тригер переключається за заднім фронтом. У тому випадку, коли на входах *J* і *K* одночасно присутній рівень лог. „1”, тригер переключатиметься почергово, то у стан лог. „1”, то у стан лог. «0» з надходженням кожного наступного тактового імпульсу. За наявності на входах *J* і *K* одночасно двох лог. «0» з надходженням тактового імпульсу тригер не переключатиметься і зберігатиме попередній стан.

Для тригеру *JK* – типу (рис.8.7) часові діаграми його роботи подані на рисунку 8.9.

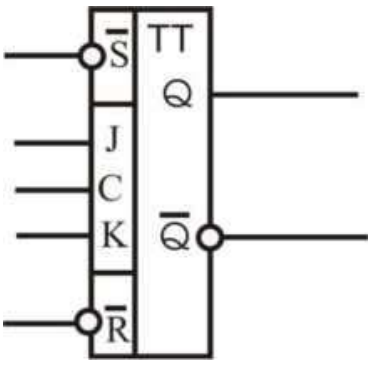


Рисунок 8.7 — Тригер *JK* – типу (типу ТВ9), що має *RS* – входи

На відміну від *D* – тригеру *JK* – тригер працює за заднім фронтом тактового імпульсу. Всі інші дії є ідентичними. Отже, у початковий момент часу на прямому виході тригера встановлено рівень лог. «0», оскільки при приходу першого тактового імпульсу обидва інформаційні входи *J* і *K* знаходяться у стані лог. «0», то тригер зберігатиме у момент часу t1 попередній стан. У момент часу *t2*, оскільки на інформаційному вході *J* уже встановлено лог. «1», то вихід тригера також встановиться у стан лог. «1». У момент часу *t3*, керуючий сигнал на вході *S* підтвердить цей стан (лог. «1») та утримуватиме тригер у цьому стані до часу *t6*, але у час *t6* стан тригеру також не зміниться, бо немає ніяких сигналів Лише у момент часу *t7*, на виході тригера встановиться рівень лог. «0», оскільки вхід *K* тригера встановлено у стан лог. «1», тому по приходу заднього фронту тактового імпульсу тригер змінить свій стан. Цей стан (лог. «0») підтвердить управляючий сигнал на *R* вході тригера у час *t8*. У час *t10*, *t11* стан тригеру також не буде змінюватись, оскільки на вхід *K* подано рівень лог. «1».

У час *t12*, *t13* відбуватиметься кожного разу переключення тригера у інший стан, оскільки на обидва інформаційні входи подано лог. «1». У момент часу *t14* тригер встановиться у стан лог. «1», оскільки на вході *J* присутня лог. «1», а на вході *K* – рівень лог. «0». У момент часу *t15* на вході *J* ще присутня лог. «1», тому вихід тригера залишиться у попередньому стані, який зберігатиметься до часу *t20* (управляючий сигнал на вході *S* з моменту часу *t17* до моменту часу *t19*, також підтверджуватиме цей стан). У час *t20* на інформаційному вході K уже встановлено рівень лог. «1», отже в цей час тригер перейде у стан лог. «0», який у

час *t21* і до кінця поданої часової діаграми підтверджуватиме керуючий сигнал на вході *R* цього тригера.

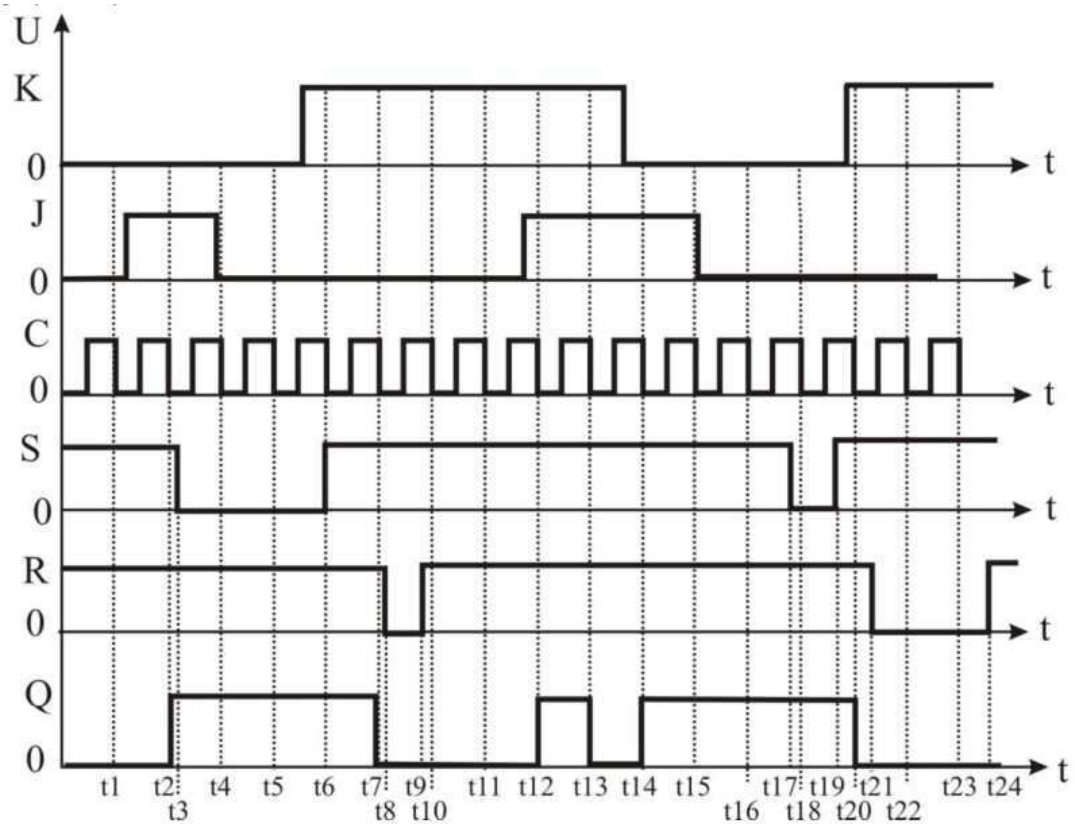


Рисунок 8.8 — Часові діаграми, які описують роботу тригеру *JK* – типу

JK – тригер також може виконувати ділення частоти на два подібно *D* – тригеру. У цьому випадку, входи *J* і *K* мають бути об’єднані і підключені до лог. «1» (рис. 8.9). Часова діаграма подібна до часової діаграми *D* – тригеру, але зміна стану тригеру відбуватиметься за заднім фронтом.

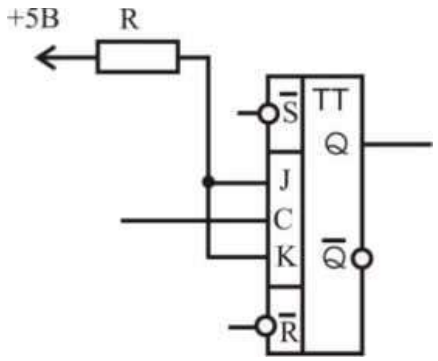
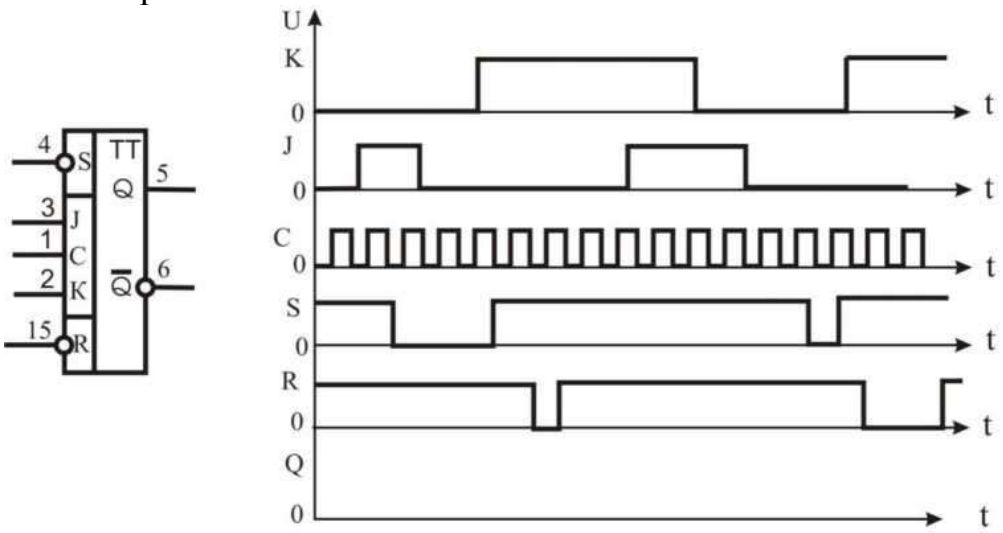


Рисунок 8.9 — Включення *JK*–тригеру у режимі ділення частоти на два

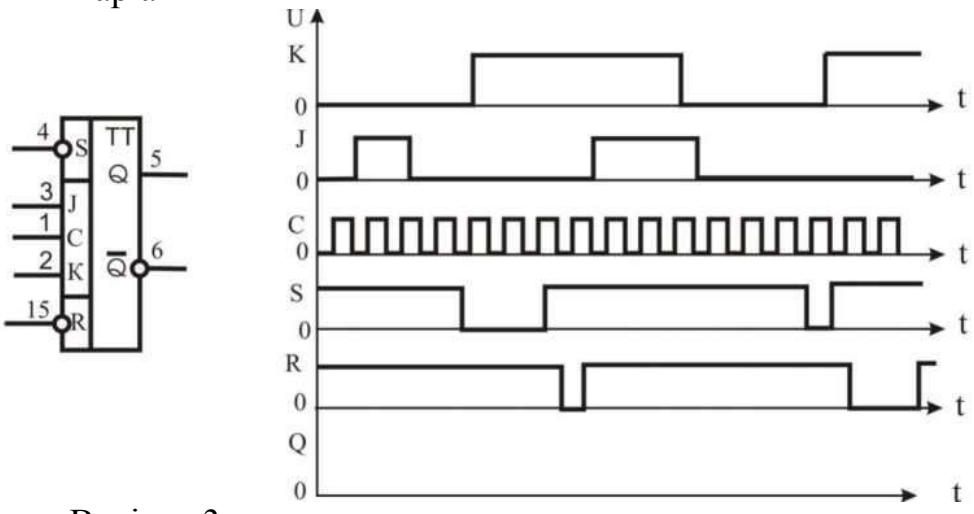
Завдання до практичної роботи

Відповідно до варіанту визначити стан виходу заданого тригера.

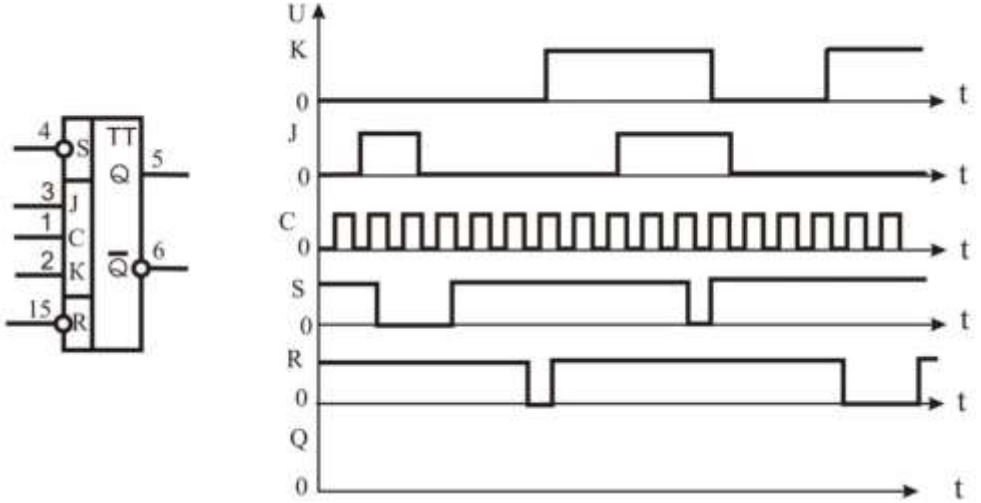
Варіант 1



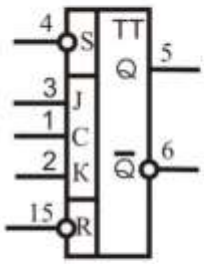
Варіант 2

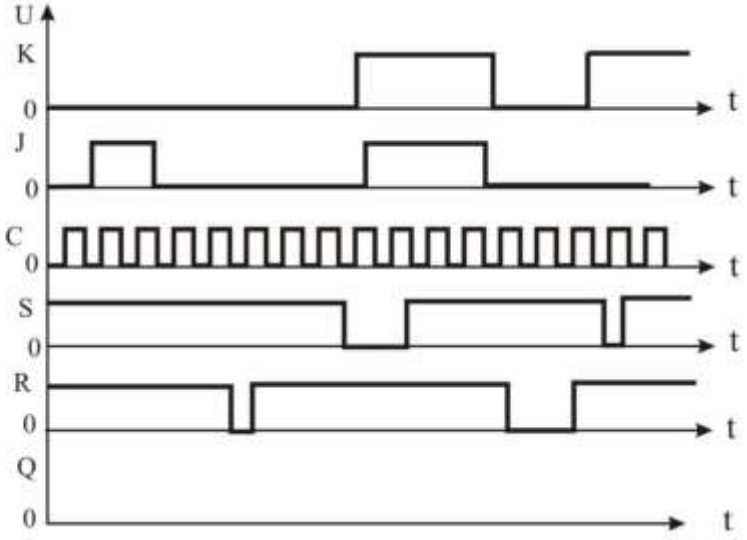


Варіант 3

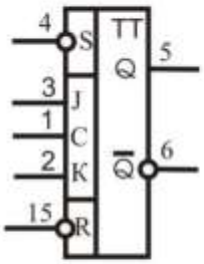


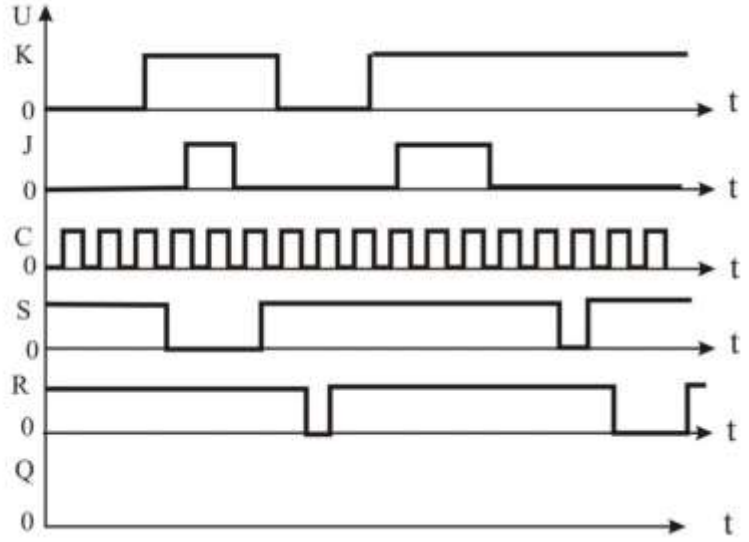
Варіант 4



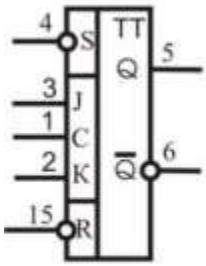


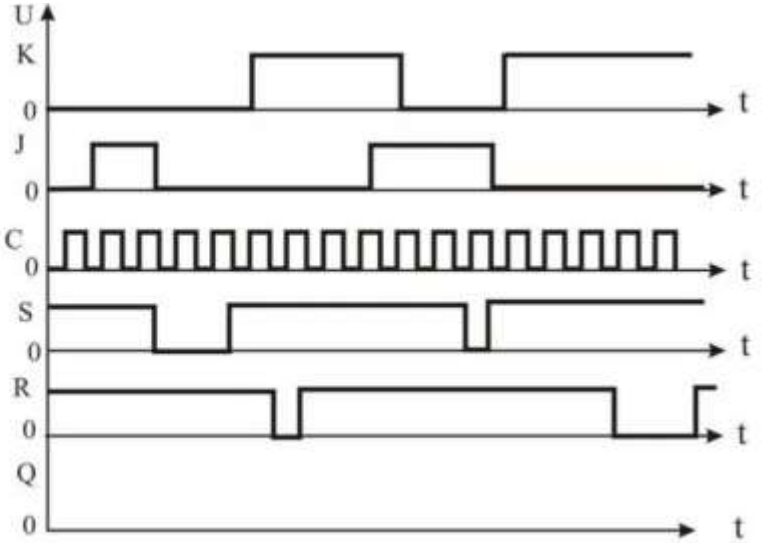
Варіант 5



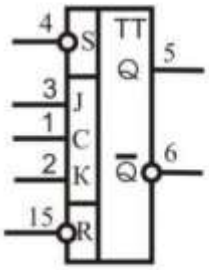


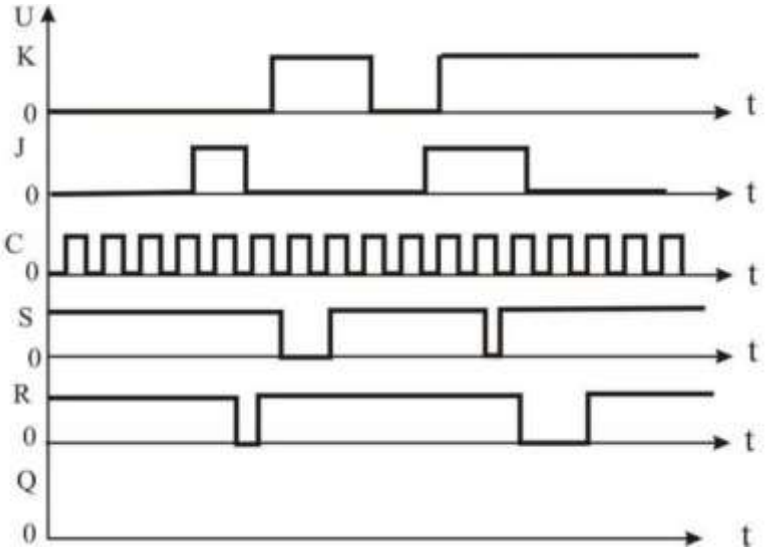
Варіант 6



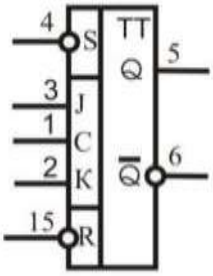


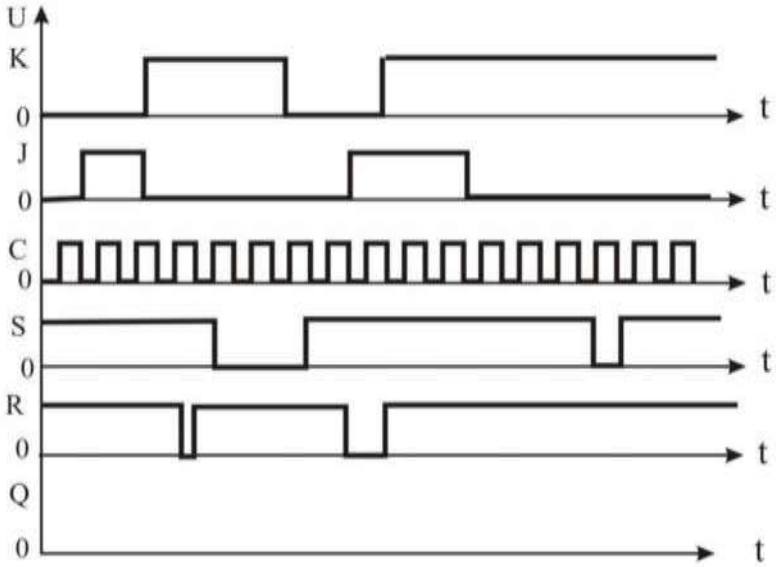
Варіант 7



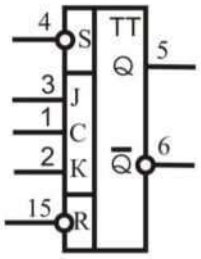


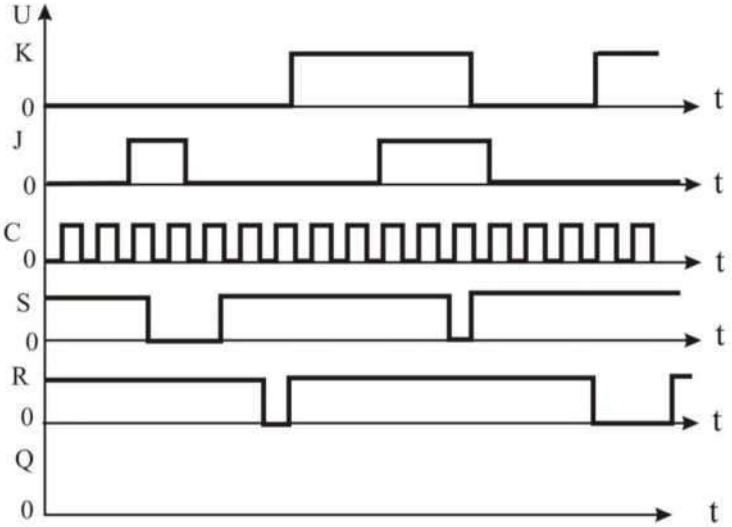
Варіант 8



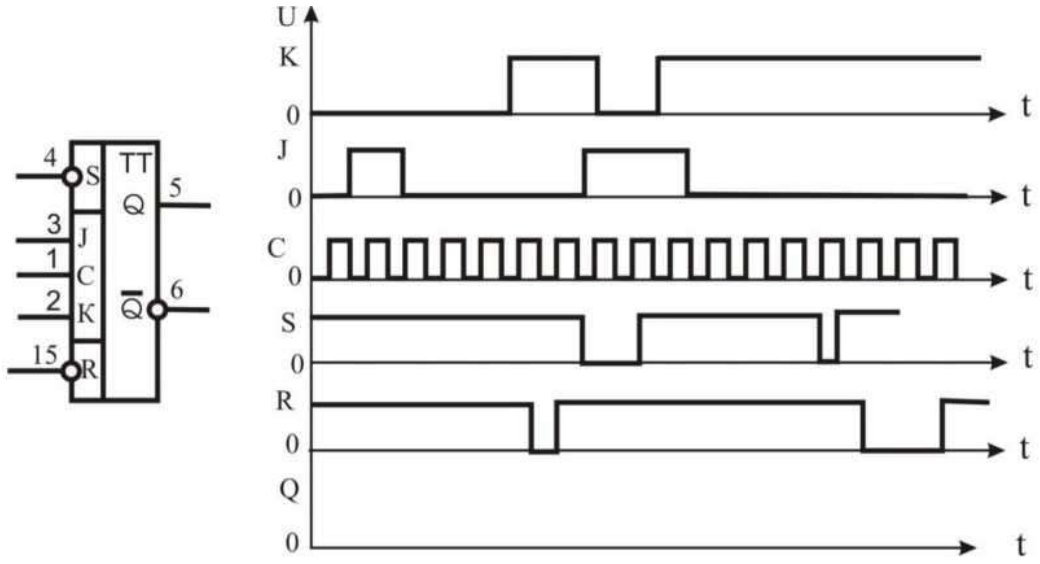


Варіант 9

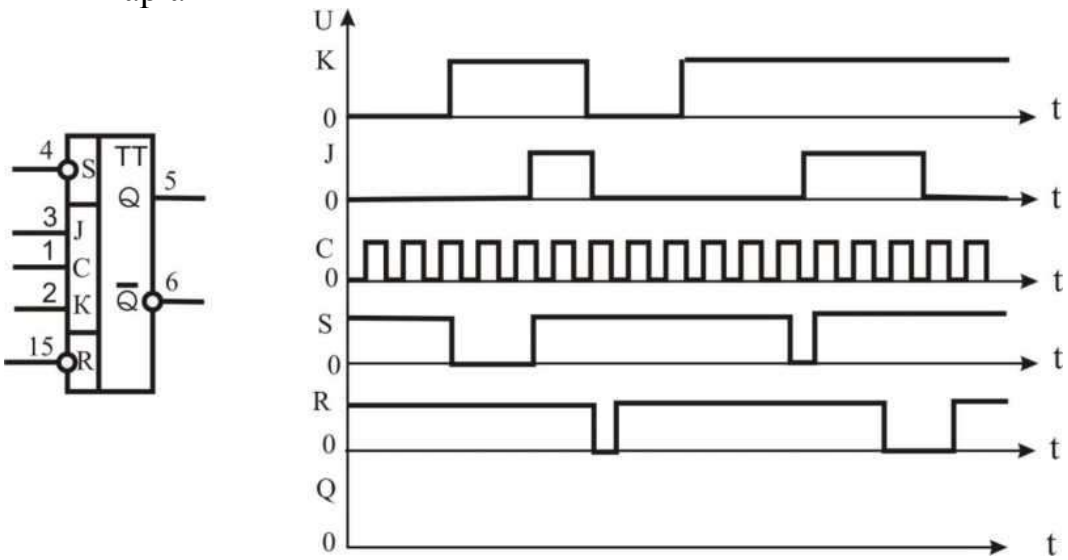




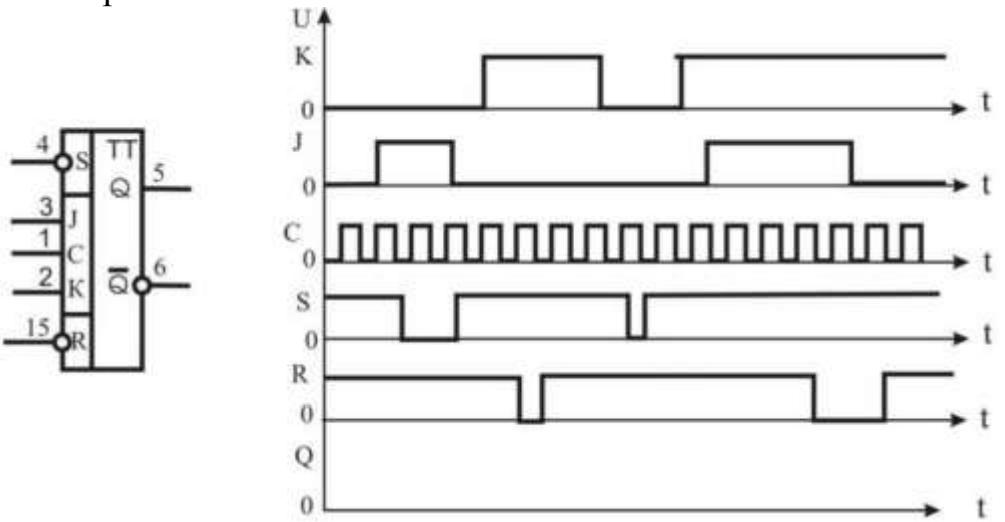
Варіант 10



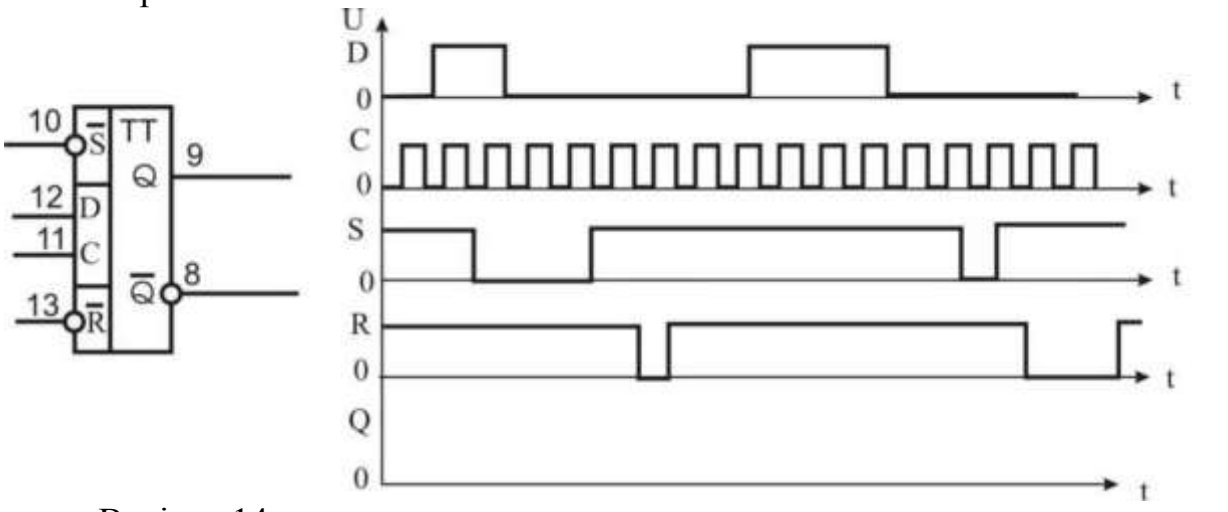
Варіант 11



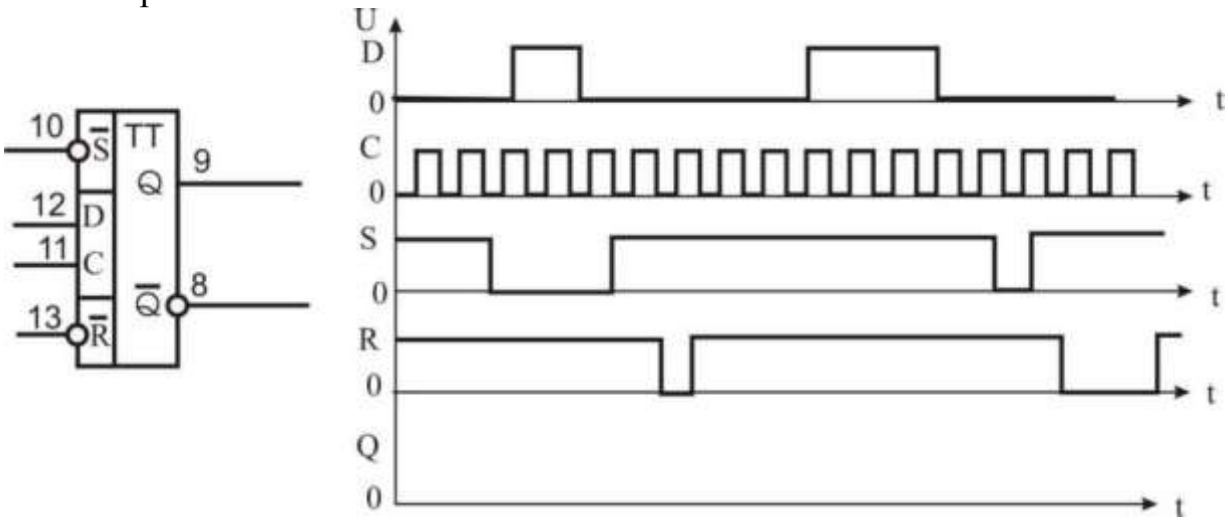
Варіант 12



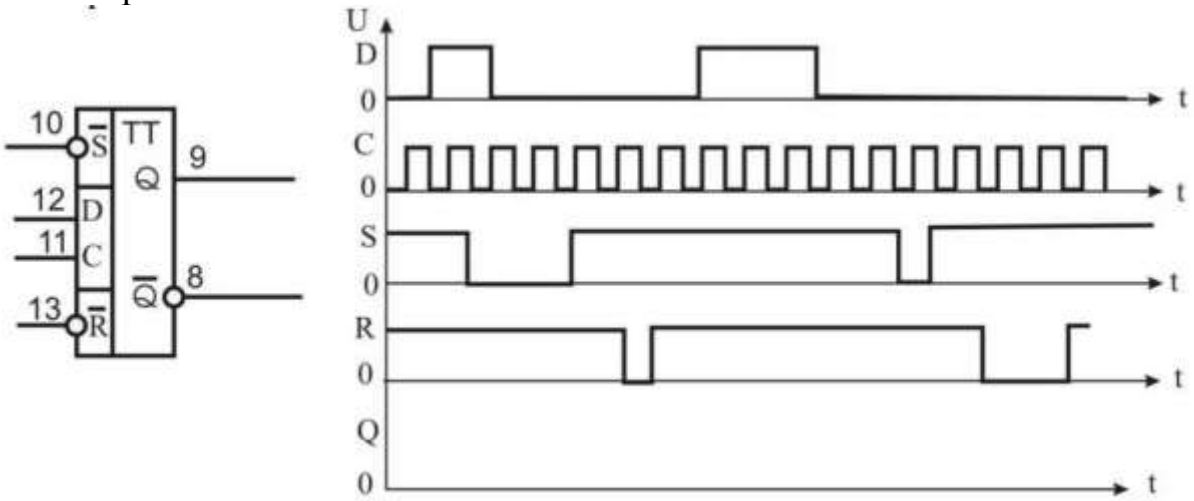
Варіант 13



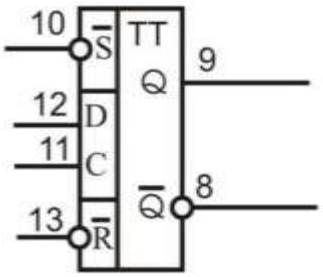
Варіант 14

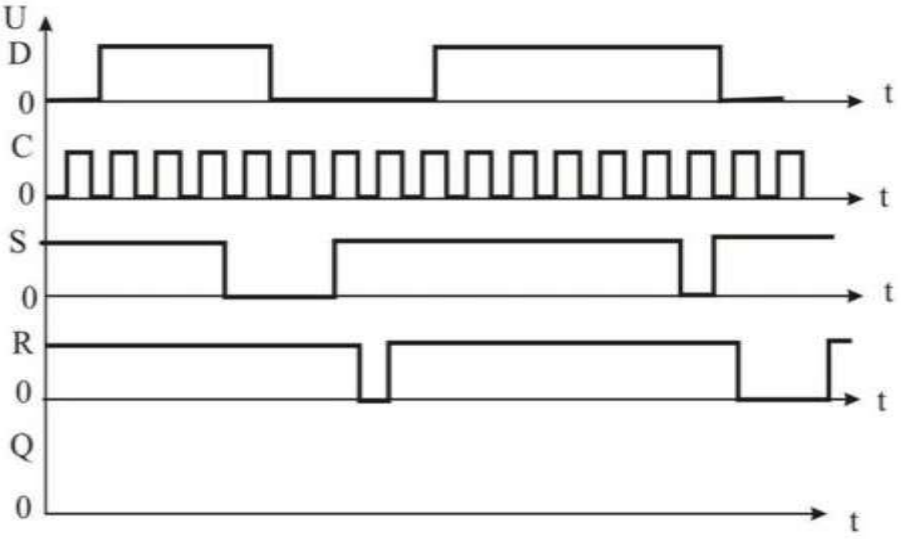


Варіант 15

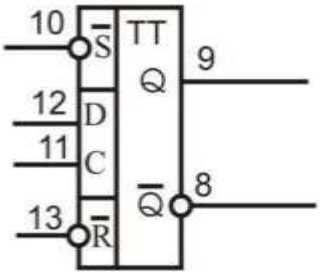


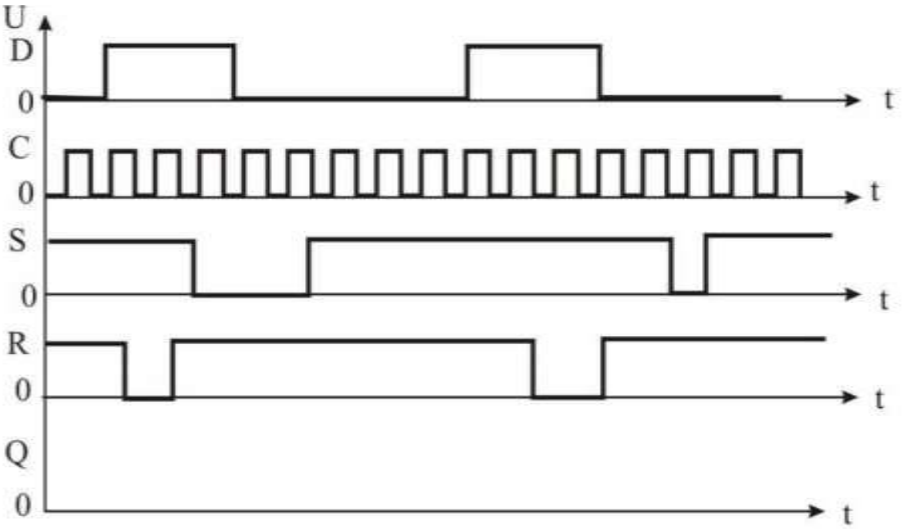
Варіант 16



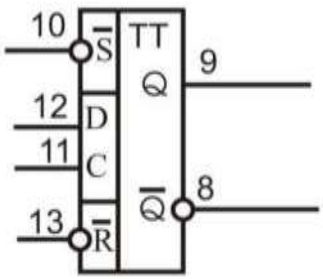


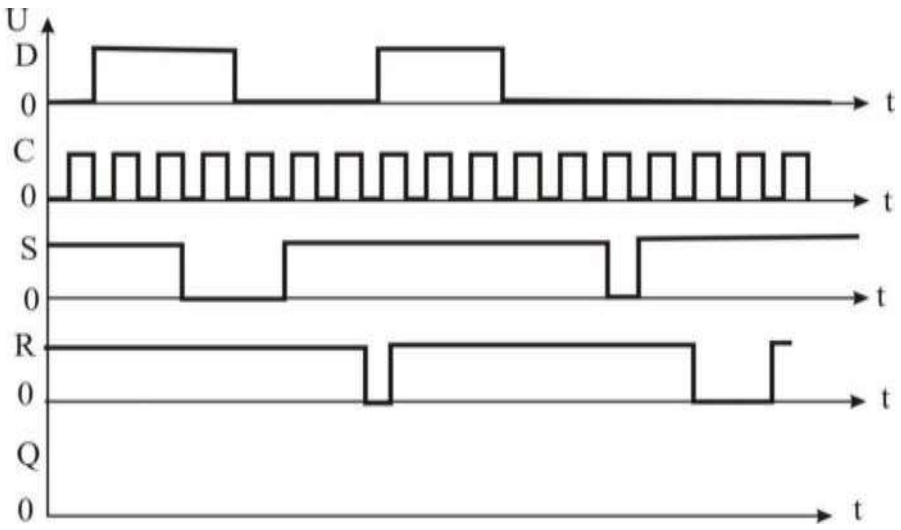
Варіант 17



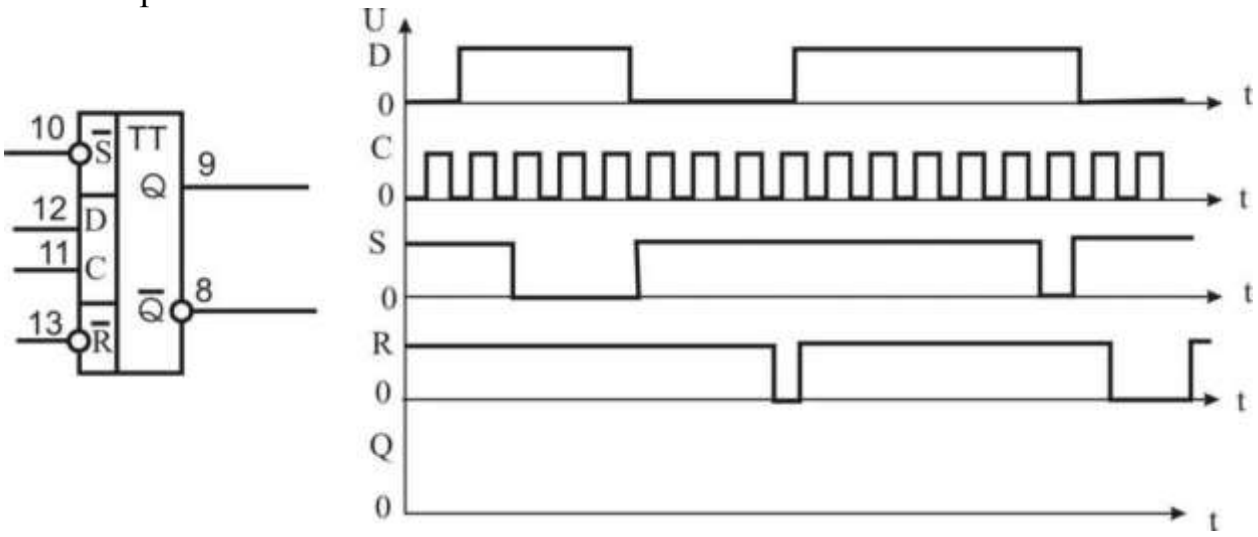


Варіант 18

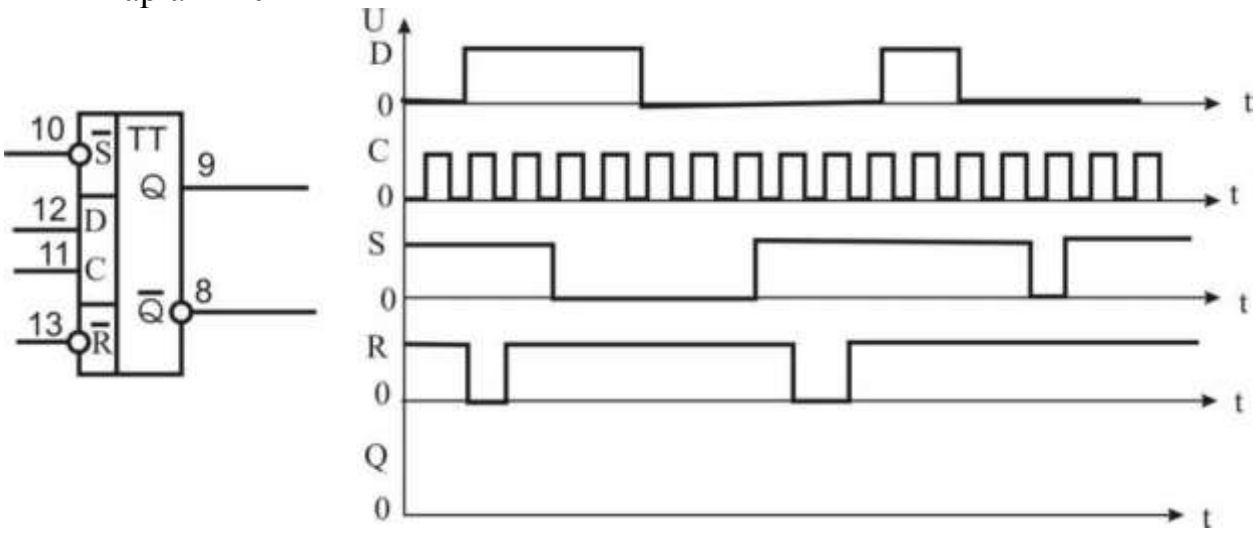




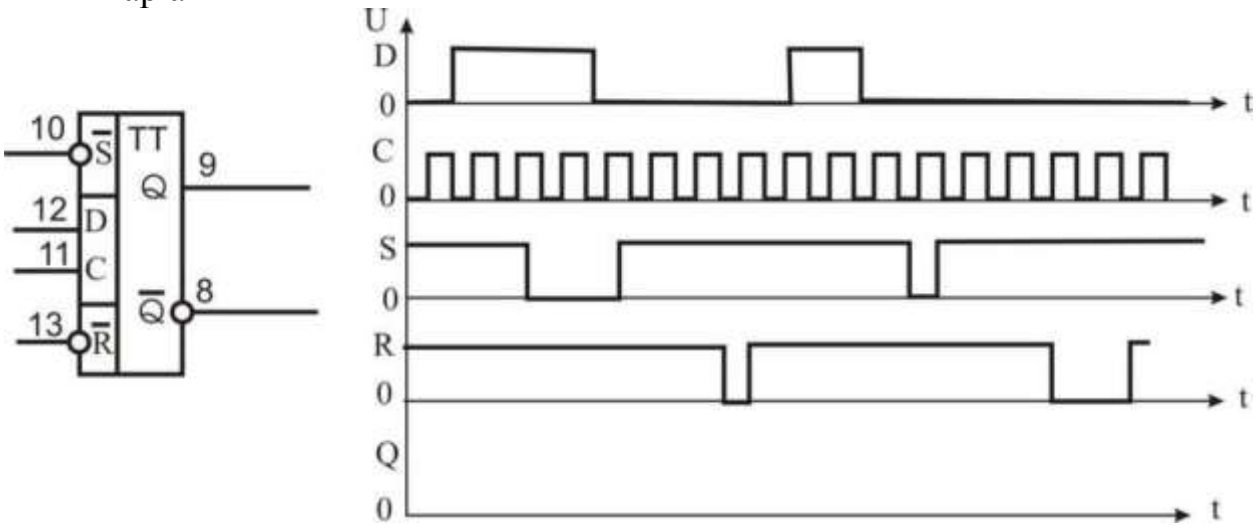
Варіант 19



Варіант 20



Варіант 21



Контрольні запитання

1. Чим відрізняється робота RS – тригера з прямими входами від роботи з інверсними входами?

2. Чому комбінація сигналів 11 на входах RS – тригера називається «забороненою»?

3. В чому відмінність таблиці переходів тригера від таблиці функцій збудження?

4. Як властивість запам’ятовування відбивається в характеристичних рівняннях тригерів?

5. В чому принципова відмінність роботи синхронних тригерів від асинхронних?

6. Яка пріоритетність інформаційних і установчих входів в синхронних тригерах?

9 ДВІЙКОВІ ЛІЧИЛЬНИКИ ТА ЛІЧИЛЬНИКИ З ДОВІЛЬНИМ КОЕФІЦІЄНТОМ ДІЛЕННЯ

Мета: поглибити і закріпити знання з побудови двійкових і десяткових синхронних і асинхронних лічильників (з послідовним і паралельним перенесенням).

Теоретичні відомості

У загальному плані *лічильниками* називаються цифрові пристрої, призначені для підрахунку і фіксації кількості імпульсів, що подаються на їх інформаційні входи або синхровходи. Назва *"лічильники"* використовується до будь–яких послідовнісних цифрових пристроїв із замкнутим циклом діаграми станів. Лічильники характеризуються наступними основними параметрами. Статичний параметр – *коефіцієнт перерахування* (*модуль перерахування*) *M* – характеризує максимальну кількість імпульсів, яка може бути подана на лічильник, щоб привести його до початкового стану [4–6].

Оскільки скінченні автомати можуть бути синхронними і асинхронними, то і, відповідно, ті групи автоматів, які спеціально призначені для підрахунку кількості імпульсів, можуть також бути синхронними і асинхронними.

У синхронних лічильниках, синхросигнал на всі тригери подається одночасно, а зміна стану тригера відбувається лише тоді, коли на інформаційних входах будь-якого тригера підготовлені відповідні дані. В асинхронних лічильниках, які часто називають *послідовними*, вхідна послідовність імпульсів подається лише на перший тригер, а решта тригерів спрацьовує в залежності від зміни стану попереднього.

Прикладом *асинхронних (послідовних) лічильників* є схема, що приведена на рис. 9.1, а. Робота найпростішого двійкового лічильника (рис. 9.1, а) пояснюється даними, приведеними у таблиці 9.1 і часовими діаграмами, зображеними на рис. 9.1, б.

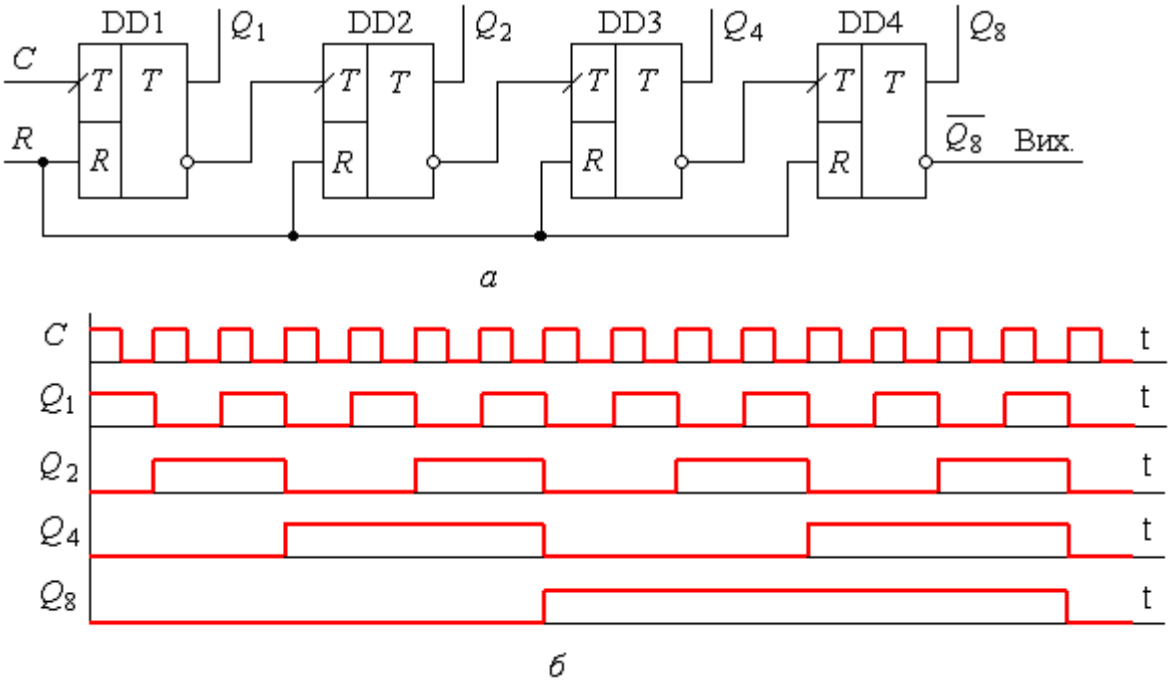


Рисунок 9.1 – Схема двійкового лічильника (а) та часові діаграмами (б)

Таблиця 9.1 – Таблиця стану виходів тригерів лічильника

Вхідні імпульси (*N*)	*Q*8	*Q*4	*Q*2	*Q*1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
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	0

Такі лічильники в своїй роботі використовують властивості *T*–тригера, оскільки вони можуть як зберігати свій стан, так і додавати за модулем 2 вхідний сигнал до інформації, записаної попередньо. Одиночний *T*–тригер ділить на 2

частоту вхідної послідовності імпульсів. Послідовне включення m таких тригерів дає можливість поділити частоту вхідних імпульсів у $M = 2^m$ разів, або утворює лічильник з коефіцієнтом перерахунку M (модуль рахунку, ємність лічильника).

Рисунок 7.1, б ілюструє часові співвідношення між станами тригерів, що зафіксовані їх прямими виходами. Зміни станів тригерів прийняті миттєвими.

З таблиці 7.1 і часової діаграми (рис. 7.1, б) бачимо, що виходи тригерів лічильника в двійковому коді відображають кількість поданих на вхід імпульсів N, якщо ця кількість менша числа M.

Якщо на вхід лічильника подано N імпульсів, то їх кількість, підрахована за допомогою лічильника, відповідає формулі:

$$N = K\,M + a_0\,2^0 + a_1\,2^1 + \ldots + a_{n-1}\,2^{n-1} = n\,M + \sum_{i=0}^{n-1} a_i\,2^i\ ,$$

де K – кількість вихідних імпульсів лічильника; $a_i \in 1, 0$ – рівень сигналу на i–ому виході лічильника; 2^i – вагові коефіцієнти кожного прямого виходу.

Для урахування вагових коефіцієнтів виходи лічильника нумерують відповідними індексами. Це дає можливість з послідовності значень виходів $Q_8\,Q_4\,Q_2\,Q_1$, рівних, наприклад, 1011, одразу ж зчитувати вміст лічильника, тобто кількість імпульсів, яка менша M.

При $N = M$ усі тригери лічильника обнуляються, і такий перехід фіксується зміною стану інверсного виходу тригера *DD4* з «0» в «1» , що наступними аналогічними схемами повинно сприйматись як фронт вихідного імпульсу. Це дає можливість безпосереднього нарощування розрядності лічильників шляхом прямого з'єднання входу наступного лічильника з виходом попереднього. При однотипних лічильниках кількість тригерів подвоюється, і загальний коефіцієнт перерахунку визначатиметься формулою:

$$M_З = 2^{2m} = 2^m \cdot 2^m = M \cdot M\ .$$

Звідси витікає, що при безпосередньому нарощуванні кількості лічильників загальний коефіцієнт перерахунку визначається добутком відповідних коефіцієнтів окремих лічильників.

Лічильники з довільним модулем рахунку мають значення M, що відрізняється від цілого ступеню числа 2. Прикладами таких лічильників можуть служити пристрої з $M = 10$; $M = 12$; $M = 24$; $M = 60$ і т. д. На практиці доводиться мати справу з лічильниками, призначеними для ділення частоти вхідних послідовностей імпульсів у сотні, тисячі і десятки тисяч разів, і далеко не завжди коефіцієнт ділення може бути кратним 2^m (m – ціле число).

На рисунку 9.2 приведена схема лічильника, в якому попереднє завантаження початкового стану забезпечується через асинхронні S–входи T–тригерів за допомогою логіки DD1…DD4 і керуючого входу PE (паралельного завантаження). Через входи D_1, D_2, D_4, D_8 у лічильник може бути записаний будь-який двійковий код у діапазоні 0000 – 1111, значення якого буде зафіксоване на виходах Q_1, Q_2, Q_4, Q_8. Запис коду забезпечується до початку подачі вхідної послідовності імпульсів на C– вхід. Тому з моменту подачі вхідних імпульсів лічильник рахуватиме, починаючи не з нуля, а з занесеного коду.

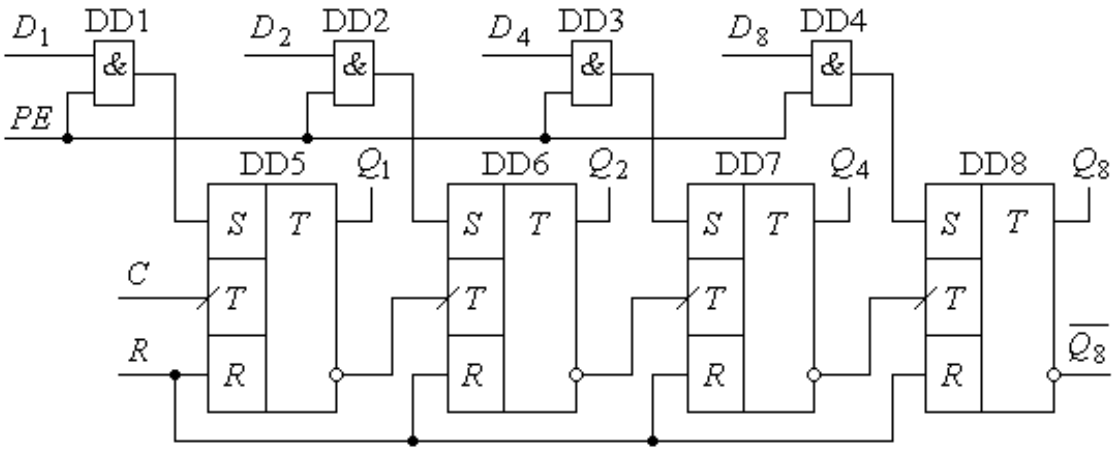


Рисунок 9.2 – Схема лічильника довільним модулем рахунку

Кількість імпульсів N, що може бути подана на C –вхід лічильника до переповнення, обчислюється за формулою:

$$N = M - D,$$

тобто з M станів лічильника виключається D перших станів.

Приклад 1. Синтезувати чотири розрядний десятковий лічильник (див. варіант 11) на основі *JK* –тригерів.

Дана, послідовність кодів (шістнадцяткові цифри): 0,1,2, 4, 5, 6, 7, 8, 9, *A*, які необхідно генерувати .

На першому етапі проектування будується таблиця переходів лічильника, поєднана з таблицею збудження входів тригерів *Ji* і *Ki* (дивись таблицю. 9.2). У колонці «Значення прямих виходів тригерів» для "Час t" записують двійкові еквіваленти шістнадцятиричних кодів. У наступній колонці для "Час *t*+l" ці ж коди записуються зі зміщенням на один рядок вгору (в таблиці 9.1 ці зсуви показані стрілками). Останній рядок в цій колонці повторює початкову рядок у колонці "Час *t*".

Таблиця 9.2 – Таблиця переходів *JK* – тригерів лічильника

Значення прямих виходів тригерів								Сигнали збудження тригерів							
Час *t*				Час *t*+1				Т4		Т3		Т2		Т1	
Q4	Q3	Q2	Ql	Q4	Q3	Q2	Q1	J4	К4	J3	К3	J2	К2	J1	К1
0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
0	0	0	1	0	0	1	0	0	X	0	X	1	X	X	1
0	0	1	0	0	1	0	0	0	X	1	X	X	1	0	X
0	1	0	0	0	1	0	1	0	X	X	0	0	X	1	X
0	1	0	1	0	1	1	0	0	X	X	0	1	X	X	1
0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	X
0	1	1	1	1	0	0	0	1	X	X	1	X	1	X	1
1	0	0	0	1	0	0	1	X	0	0	X	0	X	1	X
1	0	0	1	1	0	1	0	X	0	0	X	1	X	X	1
1	0	1	0	0	0	0	0	X	1	0	X	X	1	0	X

При побудові сигналів збудження *JK* – тригерів (табл. 9.2) необхідно враховувати умови переходів *JK* – тригера (табл. 9.3):

Таблиця 9.3 – Матриця переходів *JK* – тригера (умови переходів *JK* – тригера)

Q(t)	Q(t+1)	J	К	Умови переходу
0	0	0	X	скидання зберігання
0	1	1	X	інверсія або установка
1	0	X	1	інверсія або скидання
1	1	X	0	установка або зберігання

На наступному етапі необхідно мінімізувати перемикальні функції для всіх входів J і К, наприклад, методом діаграм карт Карно, і представити їх у відповідному базисі «І–НЕ» або «НЕ», «І», «АБО», враховуючи наявність логічних елементів у програмі Multisim.

На рисунку 9.3 наведено діаграми карт Карно для всіх входів *J* і *K* (порожні клітини в діаграмах – стани лічильника, що невикористовуються):

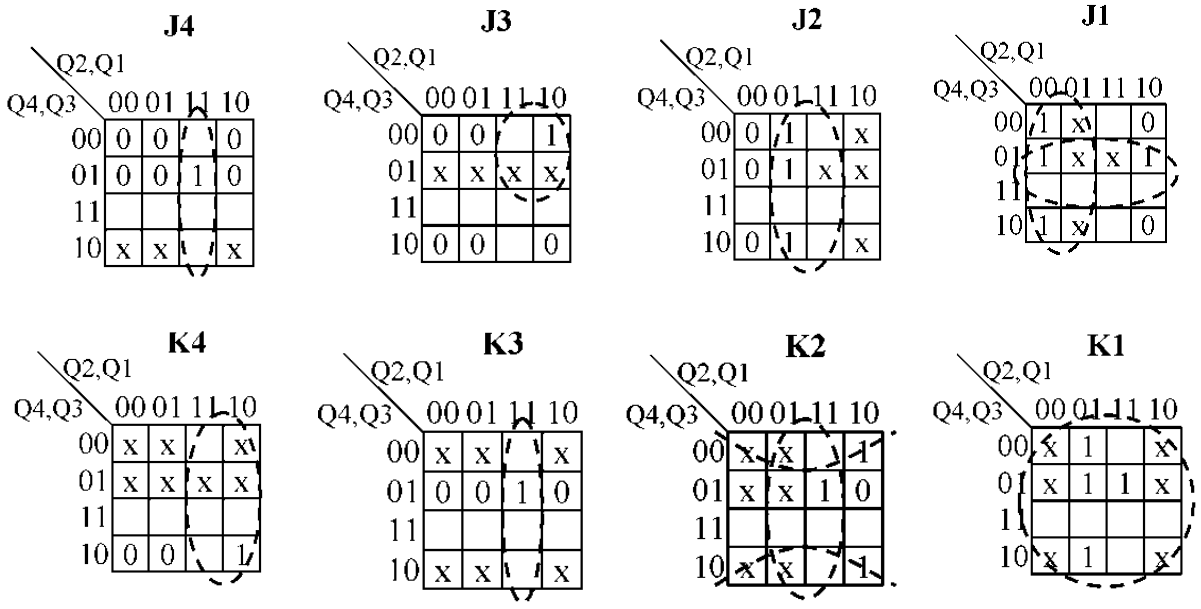


Рисунок 9.3 – Діаграми карт Карно для всіх входів *J* і *K*

Після мінімізації функції *Ji*, *Ki* мають вид:

$J1 = \overline{Q2} \vee Q3$;

$J2 = Q1$;

$J3 = Q2\overline{Q4}$;

$J4 = Q1Q2;$

$K1 = 1;$

$K2 = Q1 \vee \overline{Q3};$

$K3 = Q1\ Q2;$

$K4 = Q2.$

На рисунку 9.4 наведена схема синхронного лічильника, підготовленого для дослідження в пакеті *Multisim*.

Логічні функції для управління входами *J* та *K* реалізовані на окремих елементах «І», «АБО». Механічний контакт формує одиночний негативний імпульс тривалістю 0,5 с) після включення живлення і подає цей імпульс на інверсні входи асинхронної установки тригерів у початковий стан.

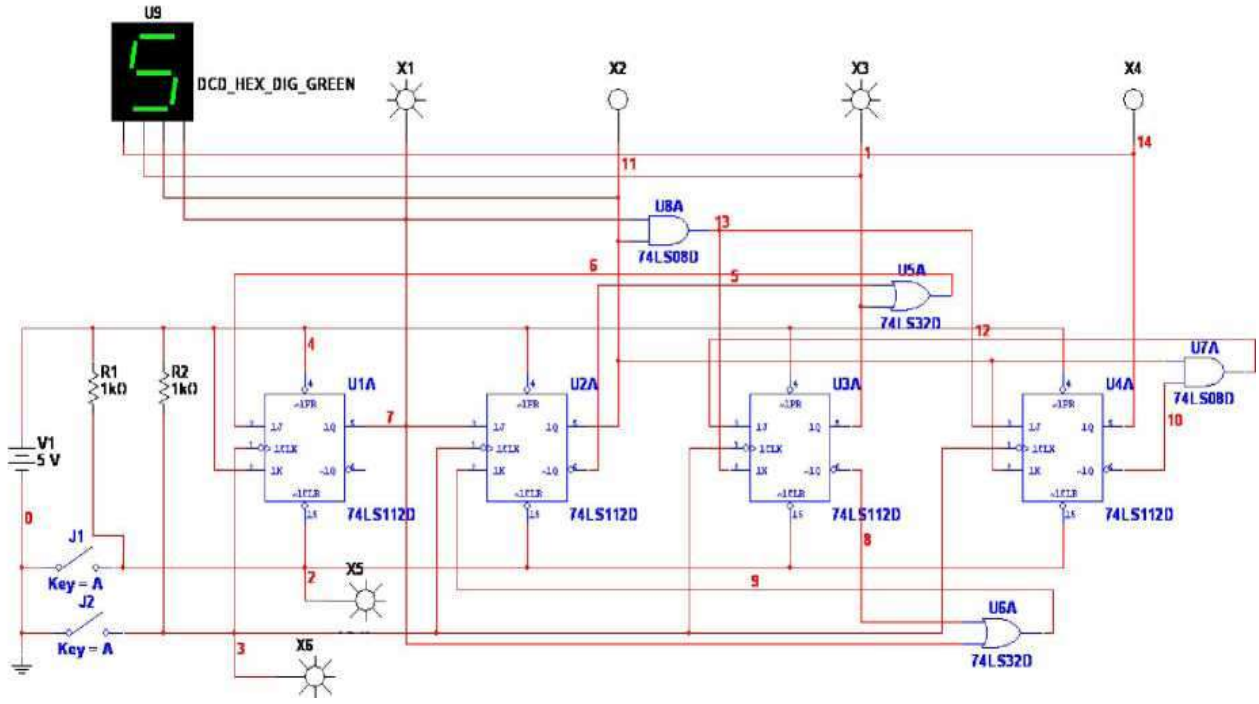


Рисунок 9.4 – Синхронний лічильник

Вхідні лічильні імпульси формуються механічним контактом, який управляється клавішею «Пробіл» («*Space*»). Для індикації станів тригерів використовуються світлодіодні індикатори, а також додатковий семисегментний індикатор з дешифратором.

Для побудови осцилограм вхідного сигналу і всіх вихідних сигналів зручно

використовувати логічний аналізатор, який може одночасно відображати до 16–ти логічних осцилограм (рис. 9.5).

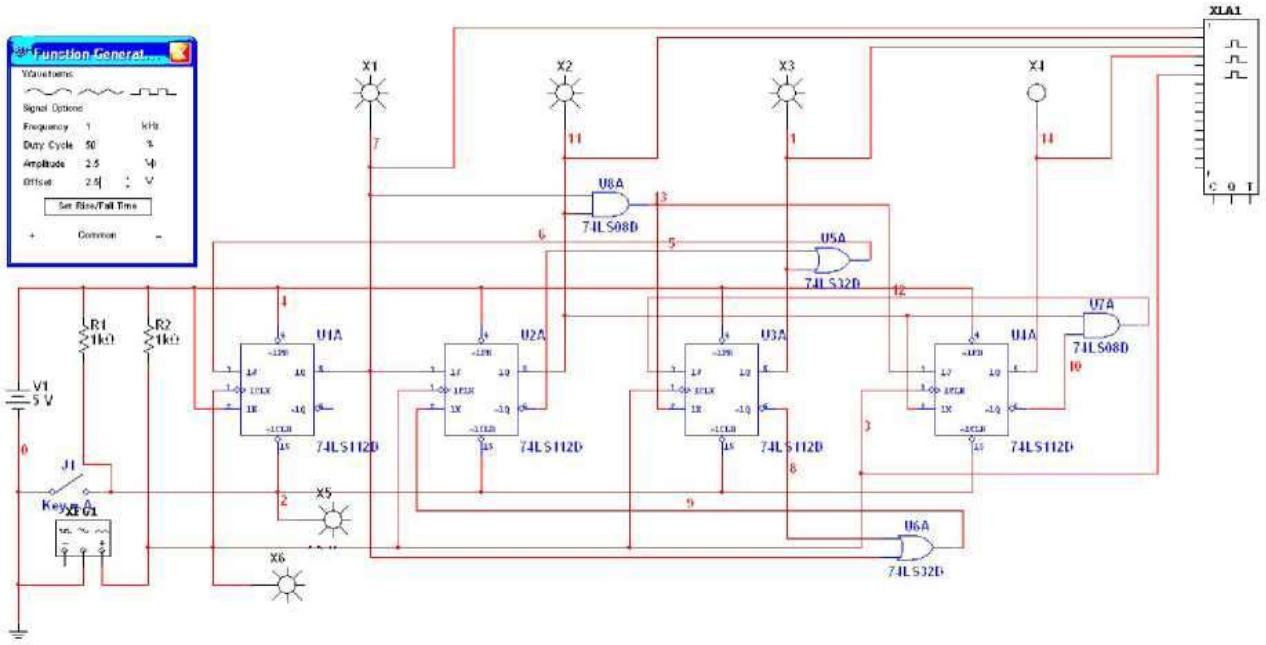


Рисунок 9.5 – Синхронний лічильник

На вхід лічильника підключається функціональний генератор (частота прямокутних імпульсів – 1 кГц, рівні вхідних сигналів – від 0 до 5 В).

За осцилограмами вихідних сигналів тригерів можна зробити висновок про те, що ці *JK* – тригери спрацьовують по спадаючому фронту вхідного прямокутного сигналу (нижня осцилограма на рис. 9.6). Двома вертикальними візирними лініями зазначено початковий нульовий стан всіх тригерів і таке нульовий стан. Між візирними лініями знаходяться десять спадаючих фронтів вхідного сигналу.

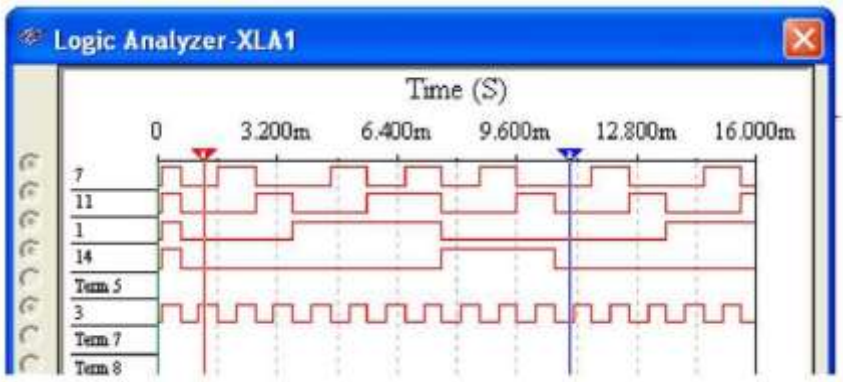


Рисунок 9.6 – Осцилограми вхідного і вихідних сигналів лічильника

Приклад 2. Реалізація лічильника з довільним коефіцієнтом ділення розглянемо на прикладі дільника на: $k = 14$:

– визначаємо кількість тригерів:

$N =] log_2 (k - 2) [,$

(знак]...[– означає: найближче більше ціле),

$N =]log_2 (14-2)[= 4;$

– переводимо в двійковий код числа «$k - 2$»:

$14-2 = 12$ D$= 1100$B;

– у підсумовуючому лічильнику з кількістю тригерів N виділяємо ті розряди, яким у двійковому коді числа «$k - 2$» відповідають одиниці; з виходів цих тригерів подаємо сигнали на елемент Шеффера; вихідний сигнал цього елемента є інформаційним для додаткового D – тригера; сигнал з виходу якого подається на вхід асинхронного скидання всіх тригерів лічильника.

Дільник реалізований на D – тригерах типу К1533ТМ2 (рис. 9.6).

Над тригерами *T1...T4* наведено двійковий код числа 12 – (молодший розряд коду – над першим тригером). Виходи тригерів *T3* і *T4* підключені до входів елемента Шеффера «І–НЕ».

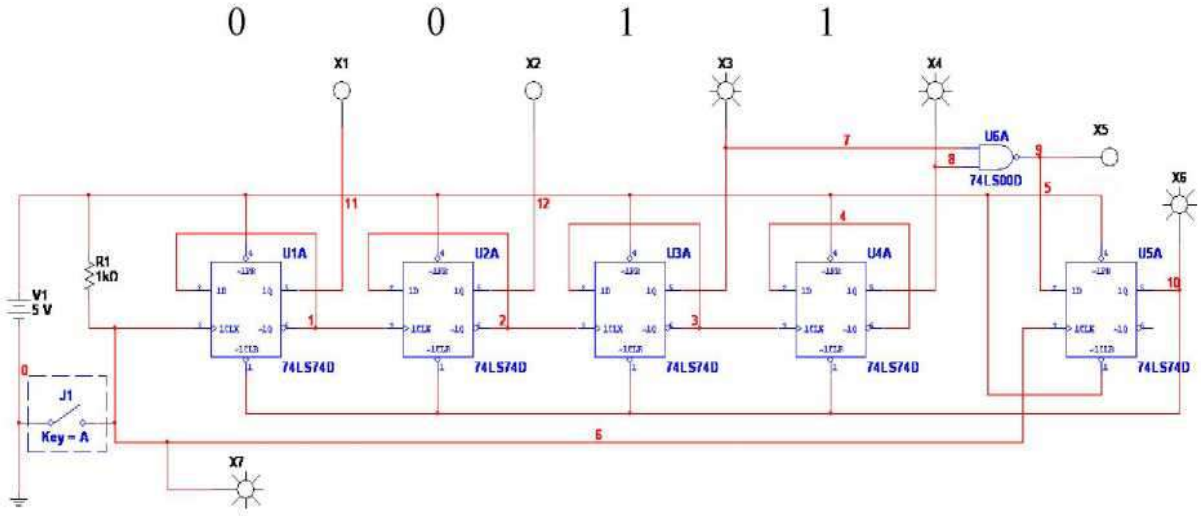


Рисунок 9.6 – Дільник на 14

Коди вихідних логічних сигналів для всіх тригерів і елемента «І–НЕ» (точка *A*) наведено в таблиці 9.4.

Таблиця 9.4 – Кодові комбінації лічильника k = 14

Код	*T4*	*T3*	*T2*	*TI*	*A*	*T5*
0	0	0	0	0	1	1
1	0	0	0	1	1	1
2	0	0	1	0	1	1
3	0	0	1	1	1	1
4	0	1	0	0	1	1
5	0	1	0	1	1	1
6	0	1	1	0	1	1
7	0	1	1	1	1	1
8	1	0	0	0	1	1
9	1	0	0	1	1	1
10	1	0	1	0	1	1
11	1	0	1	1	1	1
12	1	1	0	0	0	1
13	0	0	0	0	1	0

При всіх станах, крім 12–го, на виході елемента Шеффера встановлюється логічна «1», яка по вихідному фронту кожного вхідного імпульсу записується в додатковий тригер *T5*. Після приходу 12–го імпульсу на виході схеми «І–НЕ» встановлюється логічний «0», але в тригер Т5 логічний «0» запишеться по задньому (вихідному) фронту наступного вхідного імпульсу (див. останній рядок таблиці 9.4).

Після запису в тригер *T5* «нуля» всі тригери лічильника (*T1..T4*) встановлюються в нульовий стан низьким активним сигналом на входах асинхронного скидання *R* (див. рисунок 9.6). При цьому на виході елемента Шеффера встановлюється «логічна 1», яка переписується в тригер *T5* по закінченню наступного вхідного імпульсу і лічильник переходить в початковий стан (див. перший рядок у таблиці 9.4).

Таким чином, лічильник по черзі перебирає всі стани від «0001» до «1100» і має два нульових стани (див. перший і останній рядок таблиці 9.4). Тому при реалізації лічильника необхідно використовувати код: «*k* – 2»".

Для побудови осцилограм вхідного сигналу і всіх вихідних сигналів використовується логічний аналізатор, який може одночасно відображати до 16–ти логічних осцилограм (див. рисунок 9.7).

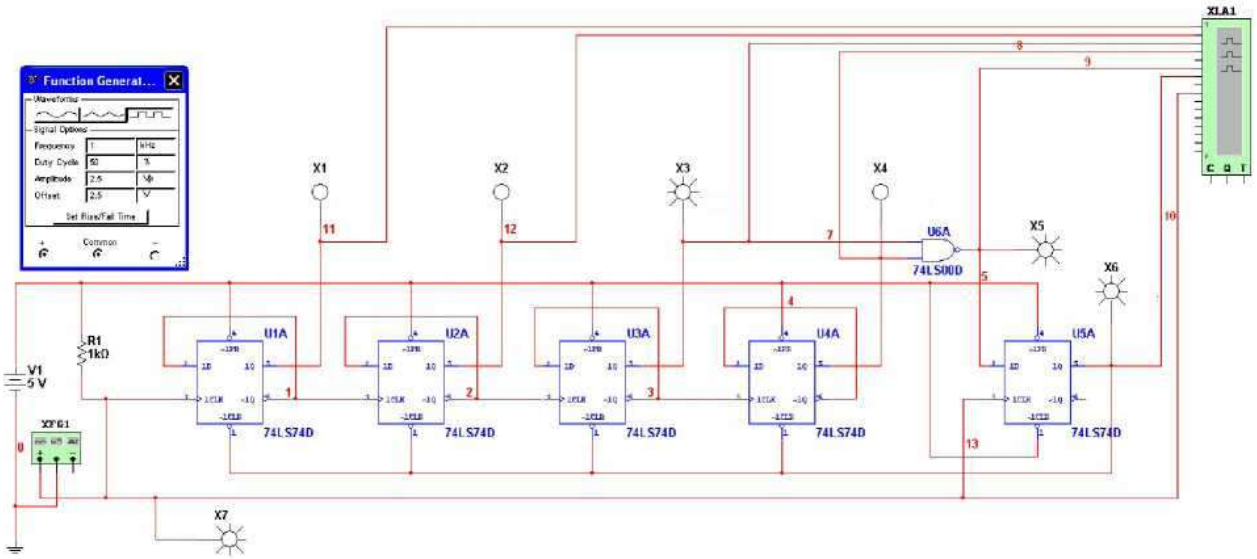


Рисунок 9.7 – Схема дільника на 14 з логічним аналізатором

На вхід лічильника підключається функціональний генератор (частота прямокутних імпульсів – 1 кГц, рівні вхідних сигналів – від 0 до 5 В).

За осцилограмами вихідних сигналів тригерів можна зробити висновок про те, що ці *D* – тригери спрацьовують по висхідному фронту вхідного прямокутного сигналу (нижня осцилограма на рисунку 9.8).

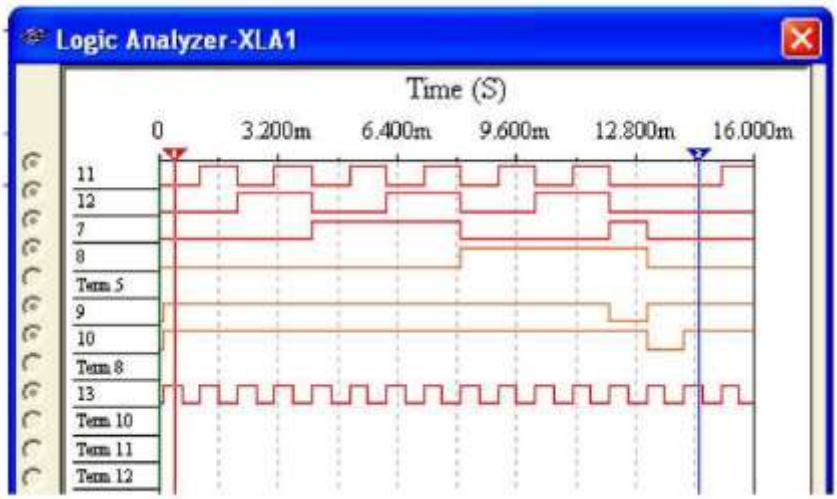


Рисунок 9.8 – Осцилограми сигналів на виходах дільника на 14

На осцилограмах (рис. 9.8) дві вертикальні візирні лінії вказують початковий стан лічильника (відповідає першій нульовою рядку в таблиці 9.4). Між цими візирними лініями знаходяться 14 висхідних фронтів вхідного прямокутного сигналу.

Завдання до практичної роботи

Перед виконанням практичного заняття необхідно вивчити методи побудови двійкових і десяткових синхронних і асинхронних лічильників з послідовним і паралельним перенесенням, а також лічильників з довільним коефіцієнтом ділення.

Завдання 1.

1.1 У відповідності з варіантом завдання (дивись таблицю 9.5) синтезувати чотирьох розрядний синхронний лічильник (аналогічно прикладу №1).

Таблиця 9.5 – Варіанти першого завдання

№ варіанта	Послідовність генеруються кодів (шістнадцяткові цифри)
1	3, 4, 5, 6, 7, 8, 9, A, B, C
2	0, 1,2,3, 5, A, C, D, E, F
3	0,1,4, 5, 7, 8, A, C, E, F
4	0,1, 2, 3,4, 5,8, 9, A, B
5	0,1, 2, 3,6, 9, C, D, E, F
6	0,1, 2, 3,4, 8, 9, A, B, C
7	0,1, 2, 3,4, 5,6, 7, C, D
8	0, 1,2,3, 5,6, 9, A, C, D
9	4, 5, 6, 7, 8, 9, A, B, C, D
10	0,1, 2, 3,4, 5, 8, 9, B, C
11	5, 6, 7, 8, 9, A, B, C, D, E

В таблиці 9.5 записані шістнадцятиричні коди, через які повинен послідовно пройти лічильник і повернутися у вхідний стан.

1.2. У відповідності з варіантом завдання (табл. 9.5) скласти схему синхронного лічильника.

1.3. Зібрати схему лічильника з індикацією станів тригерів та провести її аналіз у пакеті *Multisim* (аналогічно рисунку 9.4).

Початковий стан (відповідно з варіантом завдання), встановлюється окремим негативним імпульсом від «кнопки 0,5 сек». Цей імпульс подається на інверсні входи асинхронної установки тригерів S або R.

1.4. Подаючи поодинокі запускають імпульси, проаналізувати всі стани

лічильника і звірити з варіантом завдання (див. табл. 9.5).

1.5. Подати на вхід лічильника сигнал від генератора прямокутних імпульсів з частотою 1 кГц і перевірити роботу схеми в динамічному режимі (див. рис. 9.5). Зарисувати в звіт осцилограми вхідного сигналу і на всіх виходах тригерів (аналогічно рис. 9.6).

Завдання 2.

2.1 Розробити схему лічильника з коефіцієнтом ділення, рівним «номер варіанта + 19» (за методикою, описаною в прикладі № 2).

Для цієї схеми повторити пункти 1.3 (записати в звіт таблицю кодових комбінацій лічильника, аналогічну таблиці 9.4).

2.1 Подати на вхід лічильника сигнал від генератора прямокутних імпульсів з частотою 1 кГц і перевірити роботу схеми в динамічному режимі (див. рисунок 9.7).

2.2 Зарисувати в звіт осцилограми вхідного сигналу, на виходах всіх тригерів і на виході елемента Шеффера (аналогічно рисунку 9.8).

Контрольні запитання

1. Пояснити метод синтезу десяткового лічильника.
2. Пояснити метод синтезу лічильника з довільним коефіцієнтом ділення.
3. На яких тригерах можна реалізувати лічильники (дільники)?
4. Як реалізувати лічильний тригер на основі D–, RS–, JK – тригерів?
5. Привести схеми та пояснити роботу підсумовуючих, віднімаючих і реверсивних лічильників.
6. Який з досліджених на практичному занятті лічильників є підсумовуючим, а який віднімаючим? Як з підсумовуючого лічильника зробити віднімаючий і – навпаки?
7. Привести схеми і пояснити переваги і недоліки лічильників з послідовним та паралельним перенесенням. Який лічильник з практичного заняття є з послідовним, а який – з паралельним перенесенням?
8. Привести схеми дільників на: 3, 5, 6, 7, 9, 10, 12, 14, 15, 20 та ін.

10 ДОСЛІДЖЕННЯ АНАЛОГО-ЦИФРОВИХ ПЕРЕТВОРЮВАЧІВ

Мета: Дослідження аналого-цифрового перетворювача побудованого за принципом зчитування та одиничних приростів.

Теоретичні відомості

Аналого-цифровий перетворювача (АЦП) здійснює перетворення вхідного аналогового сигналу (струму або напруги) в вихідний код (зазвичай двійковий). Вони характеризуються великою різноманітністю схем і способів перетворення [5–7]. Найбільш поширеними структурами АЦП є:

– АЦП, що базуються на зчитуванні станів компараторів;

– АЦП з одиничними приростами компенсувального сигналу;

– АЦП з порозрядним урівноважуванням.

АЦП, що базується на зчитуванні станів компараторів. Він характеризується найбільшою швидкодією. Перетворювач містить джерело опорного сигналу U_{ref}, подільник напруги, який має 2^k резисторів R ($k = 3 \div 8$) і таку саму кількість компараторів.

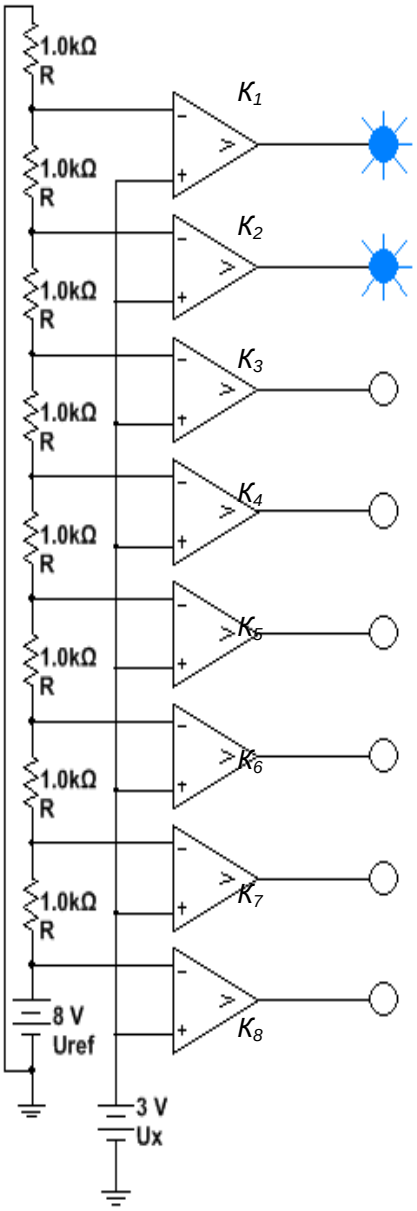


Рисунок 10.1 – АЦП, що базується на зчитуванні станів компараторів

Кожен компаратор має два входи, усі нижні входи об'єднані, і на них подається напруга U_x, що підлягає перетворенню. Кожен з верхніх входів приєднаний до одного з виводів подільника напруги. Подільник утворений ланцюгом з восьми однакових резисторів з опором R. Таким чином,

на верхні входи компараторів $K_1 - K_8$ подається напруга $n(U_{ref})/8$, де $n = 1, 2, ..., 8$. Тобто $U_{ref}/8$, $2(U_{ref})/8, 3(U_{ref})/8, ..., 8(U_{ref})/8$.

Кожен компаратор $K_1 - K_8$ на виході створює логічний сигнал: нуль, якщо $n(U_{ref})/8 \geq U_x$, і одиниця, якщо $n(U_{ref})/8 < U_x$.

Якщо $U_{ref} = 8$ В, $U_x = 3$ В (рис.10.1), то стани виходів компараторів відповідають схемі (рис. 10.1). Точність такого АЦП є невисокою, тому на практиці застосовують АЦП зчитування, що мають 64 резистори R ($k = 6$) і шестирозрядний вихідний шифратор ($n = 1 \div 64$).

Функціональна модель АЦП зчитування. АЦП, що мається в пакеті *Multisim,* за основними характеристиками схожий на АЦП, який був розглянутий раніше. Аналого–цифрове перетворення в ньому здійснюється не безперервно, а за сигналом синхронізації, що подається на вхід *SOC* (рис. 10.2). АЦП має два входи V_{ref+} і V_{ref-}, на які подається опорна напруга U_{ref} та вхід V_{in}, на який подається напруга, що призначена для перетворення.

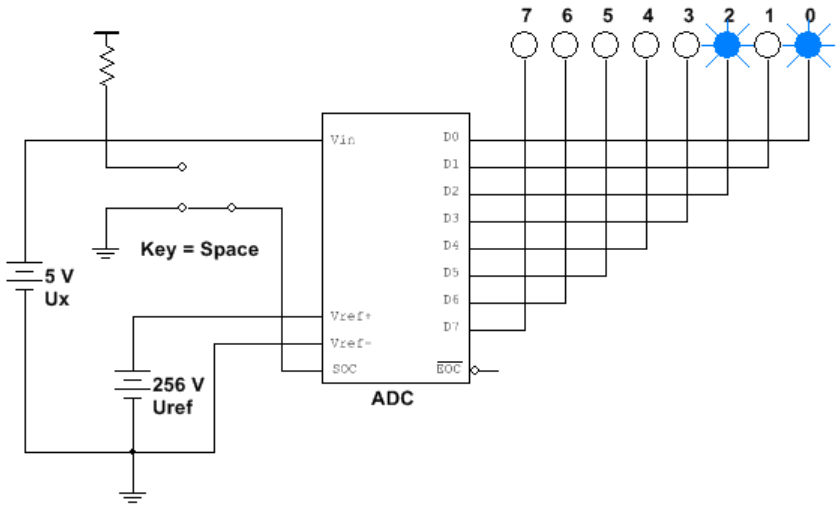


Рисунок 10.2 – Схема аналого–цифрового перетворення з застосуванням ADC для додатних значень

На восьми логічних виходах $D0, ..., D7$ формується вихідний восьмирозрядний двійковий код. За допомогою ідеального АЦП можна моделювати і АЦП з меншою кількістю розрядів. Це можна зробити, якщо не використовувати надлишкові молодші розряди.

Схема, що наведена на рис.10. 2, застосовується для перетворення у випадку, коли вимірюваний сигнал є додатним і змінюється в діапазоні від нуля до U_{max}. При використанні усіх восьми розрядів, максимальне значення коду, в який може бути перетворена напруга, дорівнює $2^8 - 1 = 255$, а значення одиниці молодшого розряду (ОМР) обчислюється за формулою

$$U_0 = U_{ref} / 2^8 = U_{ref} / 256.$$

Отже, максимальне значення напруги, яке може бути перетворене в код, складає

$$U_0 = U_{ref} \ (2^8 - 1) / 2^8 = U_{ref} \ \frac{255}{256}.$$

Якщо необхідно перетворити в код двополярний сигнал, який змінюється від мінімального від'ємного до максимального додатного значення, зручно застосовувати схему (рис. 10.3).

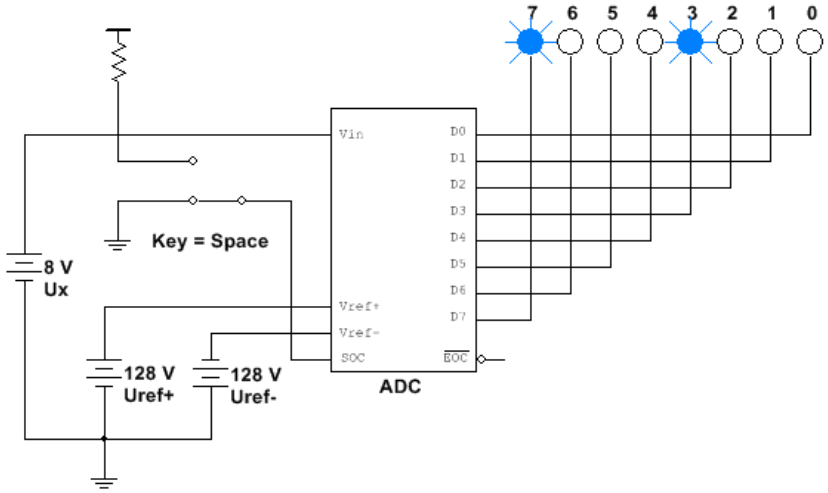


Рисунок 10.3 – Схема аналого–цифрового перетворювача з використанням ADC для двополярного сигналу

Нульовому значенню вхідного сигналу відповідає середина шкали, тобто число 128. Мінімальне від'ємне значення буде перетворено в нуль, максимальне додатне – в 255.

Характеристика вхід–вихід для випадку чотирирозрядного АЦП (використовуються чотири старших розряди) при напрузі на вході V_{ref+} – 32 В,

а на вході $V_{ref\,-}$ – 0 В, зображена на рисунку 10.4.

З рисунку 10.4 видно, що при будь–якому значенні вхідного сигналу менш ніж 2 В на виході АЦП буде формуватися двійковий код 0000, при вхідних сигналах від 2 до 4 В – код 0001. Отже, якщо на виході формується код, що дорівнює одиниці, то аналоговий сигнал, який відповідає цьому коду, можна записати у виді $A = 3 \pm 1$ В. Це визначає похибку, що обумовлена дискретністю перетворення.

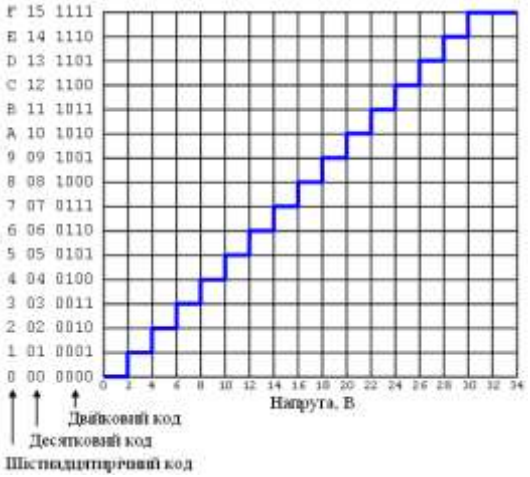


Рисунок 10.4 – Характеристика вхід–вихід для додатної напруги

Розглянемо тепер характеристику перетворення двополярної напруги. Задамо напругу на входах $V_{ref+} = V_{ref-} = 16$ В. Вихідні величини відкладені на горизонтальній осі (рис. 10.5).

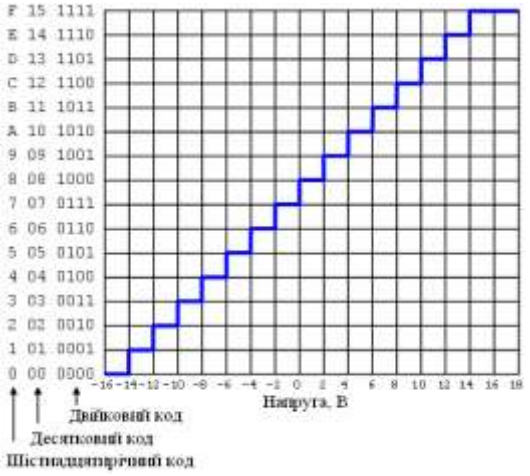


Рисунок 11.5 – Характеристика вхід–вихід для двополярної напруги

Таким чином, однаковим вихідним кодам на вертикальній осі рисунків 10.4 і 10.5 в залежності від способу підключення V_{ref} відповідають різні напруги, що підлягають перетворенню. Для зменшення максимальної похибки часто характеристику перетворення зміщують на $1/2$ ОМР, як зображено на рисунку10. 6.

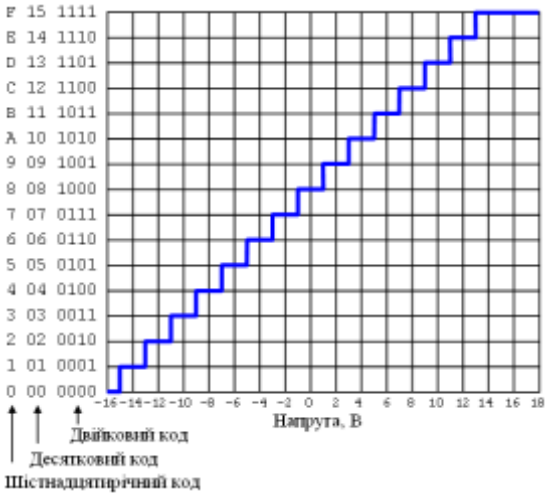


Рисунок 10.6 – Характеристика вхід–вихід для зменшення максимальної похибки

Якщо діапазон аналогових сигналів, що піддаються перетворенню, заданий, то неважко обчислити значення опорних напруг на входах V_{ref+} і V_{ref-}, за умов яких перекривається увесь діапазон і забезпечується максимальна точність перетворення:

$$V_{ref+} = \frac{U_{x\,max}}{U_{x\,max} - U_{x\,min}}; \tag{10.1}$$

$$V_{ref-} = \frac{U_{x\,min}}{U_{x\,max} - U_{x\,min}}. \tag{10.2}$$

*Похибка перетворення коду в напругу***.** Розглядаючи вихідні характеристики, неважко зрозуміти, яким чином визначається похибка, що обумовлена дискретністю перетворення.

Абсолютна похибка при будь-якому вхідному сигналі не повинна

перевищувати *1* ОМР. Максимальна відносна похибка за цих умов залежить від значення напруги, що піддається перетворенню. Вона обчислюється за формулою

$$\delta = \frac{U_0}{U_x}.$$

Похибка буде мінімальною за умов максимального значення вимірюваної напруги $U_{x\,max}$ і складатиме

$$\frac{U_0}{U_{x\,max}} = \frac{U_0}{U_0 \cdot 2^n} = \frac{1}{2^n},$$

де U_0 – значення ОМР; U_x – аналогова напруга, що підлягає перетворенню.

При зниженні напруги, що перетворюється, максимальна похибка збільшується, досягаючи 100% за умов $U_x = U_0$.

Для забезпечення точності перетворювання не менш ніж 0,1%, необхідно мати не менш ніж 10 двійкових розрядів ($1/(2^{10}-1) < 0{,}001$). Стандартні модулі АЦП зазвичай бувають восьми – десятирозрядними. Однак у подальшому, для зручності запису результатів, обмежимося при розгляді АЦП чотирма розрядами.

АЦП з одиничними приростами компенсувального сигналу. Якщо сигнал, який підлягає перетворенню, змінюється досить повільно, наприклад, у випадку виміру напруги на виході акумулятора в процесі розрядки, то швидкодія перетворювача не має значення. Завдяки цьому вдається спростити схему АЦП. Однією з можливих структур для таких застосувань може бути структура, що зображена на рис. 10.7, а.

Перетворювач містить двійковий n – розрядний лічильник, цифроаналоговий перетворювач, аналоговий компаратор з входами напруги, що перетворюється (U_x), і напруги, що компенсується ($U_к$).

За сигналом «Пуск» лічильник обнуляється (на його виходах встановлюється код 0000). Потім починається підрахунок кількості імпульсів,

що надійшли від генератора тактів (ГТ). Зі збільшенням числа імпульсів та, відповідно, й коду K_x, зростає вихідна напруга ЦАП. Коли встановлюється співвідношення $U_к \geq U_x$ компаратор сигналом «Стоп» припиняє роботу лічильника. Зафіксований в лічильнику та на індикаторі код K_x і буде результатом перетворення. Цикл повторюється з надходженням наступного сигналу «Пуск». Часові діаграми (рис. 10.7, б) пояснюють роботу АЦП.

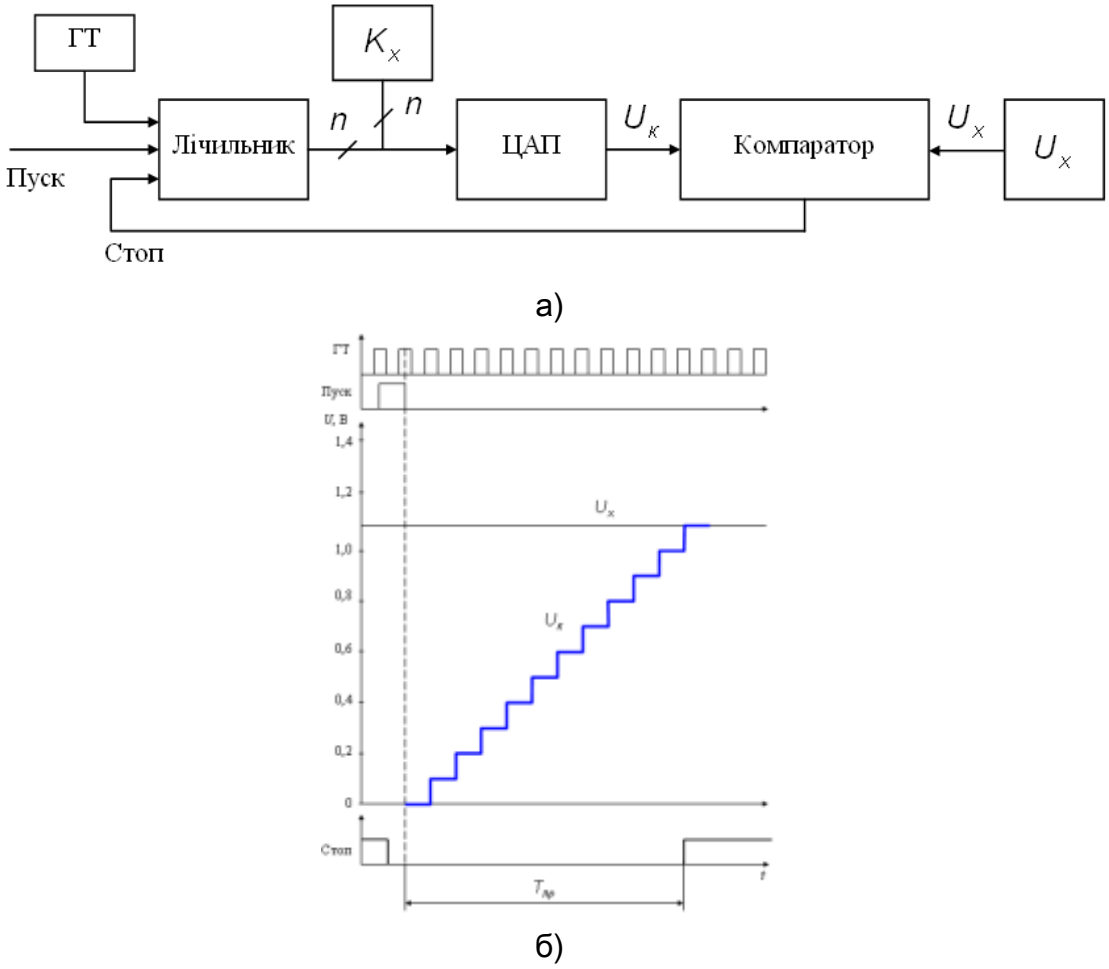


Рисунок 10.7 – АЦП з одиничними приростами компенсувального сигналу

Лічильник має n розрядів. Частоти ГТ і сигналів «Пуск» обираються таким чином, щоб за час між пусковими сигналами лічильник встиг заповнитися ($K_{max} = 1111...$), а система індикації коду встигла зареєструвати код K_x, тобто результат перетворення аналог – код.

Системи АЦП з одиничними приростами можуть виконуватися за принципом стеження. У цьому випадку лічильник повинен бути реверсивним, а

компаратор повинен фіксувати виконання однієї з умов:

1) $U_к > U_x + U_0$;

2) $U_x - U_0 \le U_к \le U_x + U_0$;

3) $U_к < U_x - U_0$.

Якщо виконується умова 1 і компенсувальна напруга ЦАП є недостатньою, то компаратор встановлює режим роботи лічильника на додавання; якщо виконується умова 3, лічильник переводиться в режим віднімання. Режим 2 означає, що з точністю $EMP = U_0$ можна вважати $U_к \approx U_x$, тобто лічильник не буде працювати до тих пір, доки компаратор не виявить переходу до умови 1 або умови 3.

Розглянута структура АЦП має назву структури з одиничними приростами компенсувального сигналу. Вона характеризується низькою швидкодією. У нашому прикладі (рис. 10.7) найбільший час перетворення $T_{пр} = 2^n T_{ГТ}$.

Завдання до практичної роботи

1. АЦП, що базується на зчитуванні станів компараторів спостереження процесів в АЦП.

Складіть схему, що зображена на рис. 10.8. Оскільки активними рівнями на входах і виходах пріоритетного шифратора є рівні логічного нуля, то до виходів компараторів підключені інвертори. В цьому випадку активний стан виводу відповідає пробнику, що світиться. Встановіть частоту генератора трикутної напруги 10 Гц, амплітуду напруги 8 В і зміщення 7 В. Запустіть схему і спостерігайте за вихідною напругою генератора на екрані осцилографа, станом входів і виходів шифратора та кодами на декодувальному семисегментному індикаторі в реальному часі (двійковими і шістнадцятирічними). Осцилограму помістіть в звіт з практичного заняття. Проаналізуйте, який вплив на роботу схеми має зміщення. Доведіть це за допомогою осцилограми.

2. АЦП, що базується на зчитуванні станів компараторів вимірювання потенціалів вузлів подільника напруги.

Виміряйте потенціали вузлів 1–15 подільника напруги, результати вимірювання занесіть до табл. 10.1.

Таблиця 10.1 – Результати вимірювань вузлів подільника напруги

Номер вузла	1	2	3	4	5	6	7	8
Напруга, В								
Номер вузла	9	10	11	12	13	14	15	16
Напруга, В								

3. АЦП, що базується на зчитуванні станів компараторів *вимірювання порогів переключення молодшого розряду шифратора.*

Другий вхід осцилографа підключіть до пробника молодшого розряду *A*0 (рис. 10.8). Встановіть частоту генератора трикутної напруги 10 Гц, амплітуду напруги 1 В і зміщення 5 В. Запустіть схему та спостерігайте за вихідною напругою генератора на екрані осцилографа, станом входів і виходів шифратора та шістнадцятирічним кодом на декодувальному семисегментному індикаторі в реальному часі. Помістіть осцилограми, що отримані на екрані осцилографа, в звіт з практичного заняття. Виміряйте поріг переключення молодшого розряду за допомогою осцилографа. Результат вимірювання занесіть до табл.10. 2.

Таблиця 10.2 – Результат вимірювання за допомогою осцилографа

Параметр	Зміщення, В						
	1	2	3	4	5	6	7
Верхній поріг переключення молодшого розряду (1)							
Верхній поріг переключення молодшого розряду (2)							
Середнє значення верхнього порогу							
Нижній поріг переключення молодшого розряду (1)							
Нижній поріг переключення молодшого розряду (2)							
Середнє значення нижнього порогу							
Потенціал у відповідних точках 1, 2, …, 7							

Порівняйте отримане значення з потенціалом у точці 5. Проведіть відповідні вимірювання для значень зміщення 1, 2, 3, 4, 6 і 7 В. Результати занесіть до табл. 10.2 та зробіть висновки.

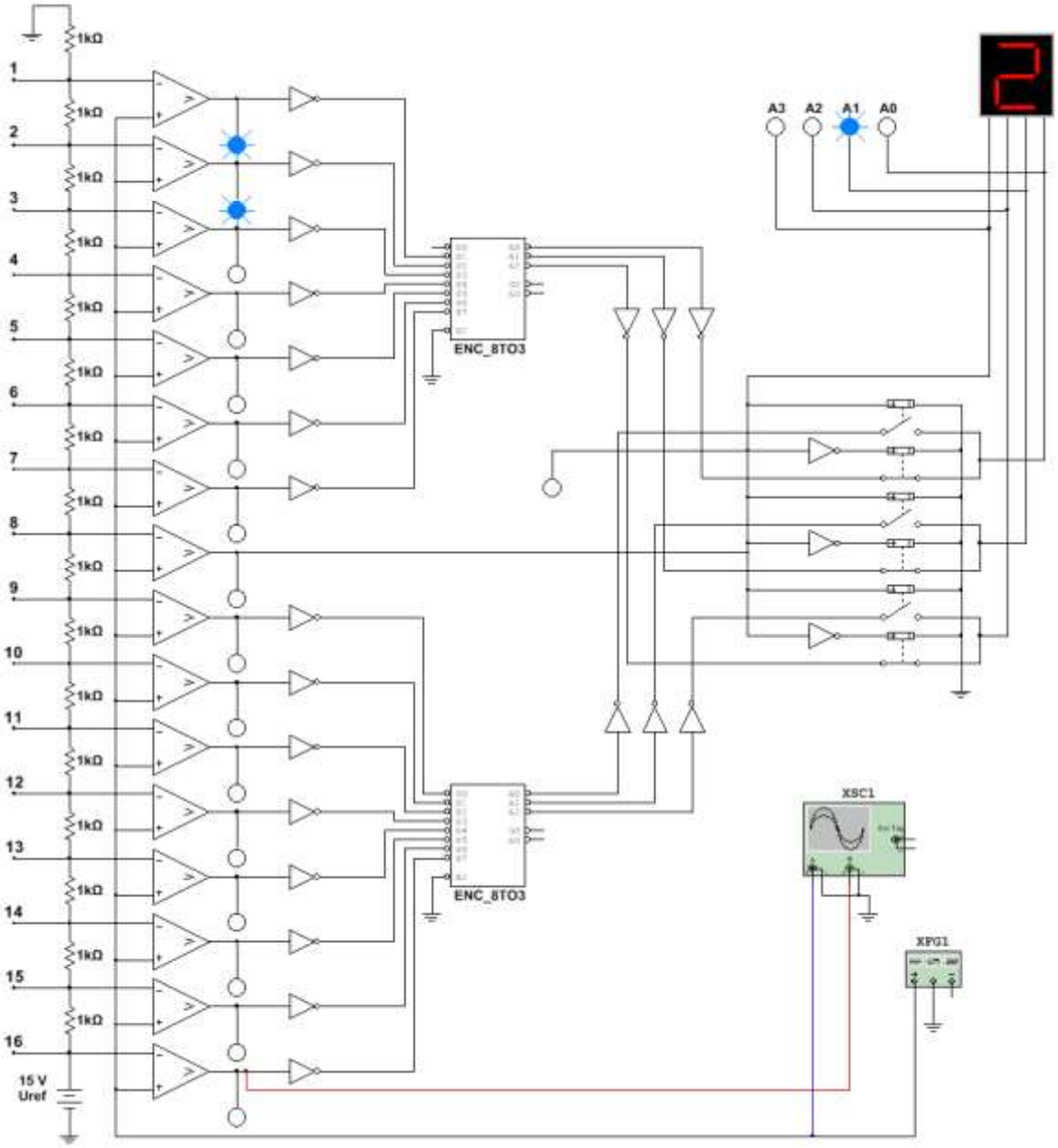


Рисунок 10.8 – Схема АЦП, що базується на зчитуванні станів компараторів

4. Формування осцилограм процесів в АЦП.

Для схеми на рис. 10.9 і даних із завдання 1, помістіть осцилограму вихідної напруги генератора і напруги на пробнику молодшого розряду в звіт з практичного заняття. Проставте на відповідних інтервалах часової діаграми напруги на пробнику двійковий код $A3, A2, A1, A0$ і код, що показує

декодувальний індикатор.

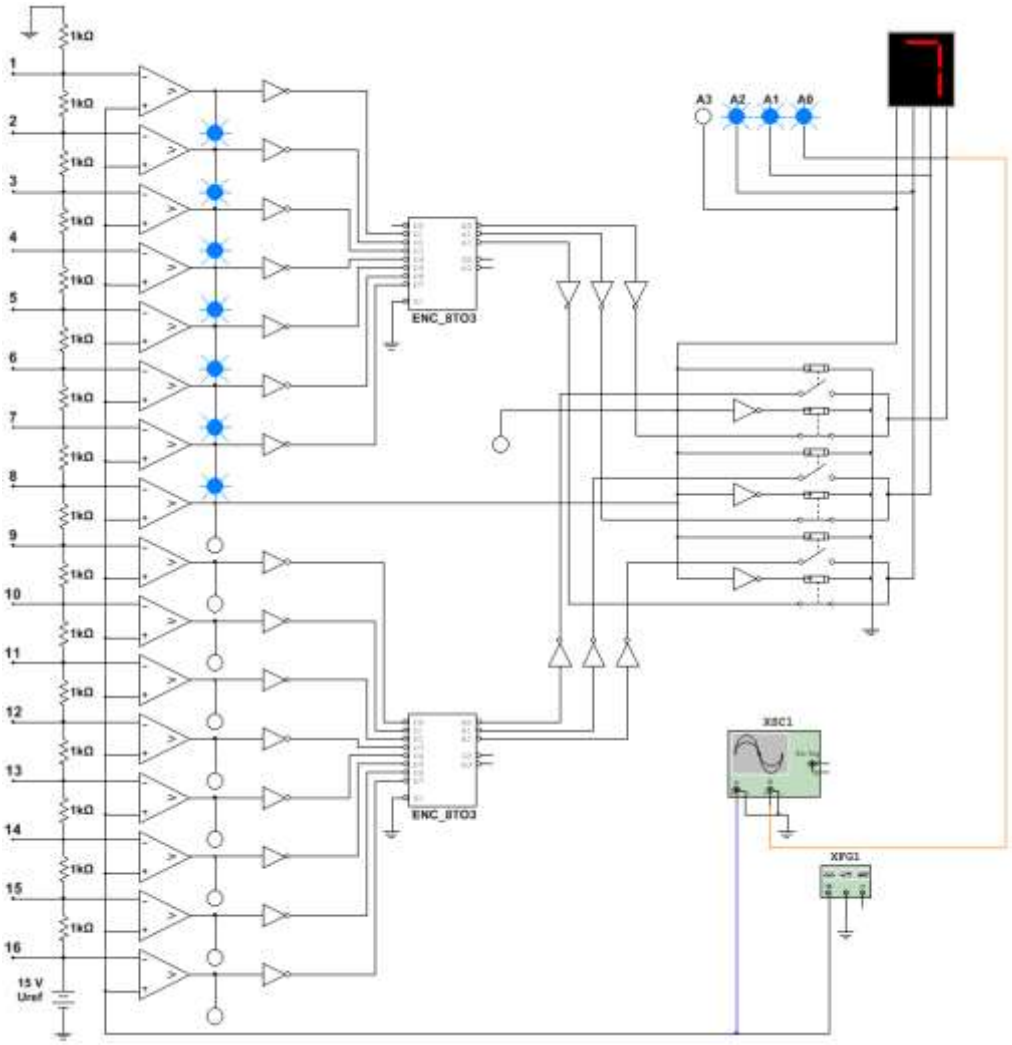


Рисунок 10.9 – Схема для вимірювання порогів переключення молодшого розряду

Контрольні запитання

1. Для чого призначені аналого-цифрові перетворювачі?

2. Що таке функціональна модель АЦП зчитування?

3. Як обчислюється максимальне значення напруги, яке може бути перетворене в код ?

4. Особливості аналого-цифрового перетворювача з використанням ADC для двополярного сигналу.

5. Похибка перетворення коду в напругу.

6. АЦП з одиничними приростами компенсувального сигналу.

ПЕРЕЛІК ДЖЕРЕЛ ІНФОРМАЦІЇ

1. Матвієнко М.П. Комп’ютерна логіка. Навчальний посібник. – Київ: ТОВ Центр навчальної літератури", 2012. – 288 с.

2. М.Ф. Бондаренко, Н.В. Білоус, А.Г. Руткас. Комп’ютерна дискретна математика: Підручник – Харків: «Компанія СМІТ», 2004. – 480 с.

3. С.Д. Шапорев. Математическая логика. Курс лекций и практических занятий – СПб: БХВ – Петербург, 2005 – 416 с.

4. Схемотехніка електронних систем: У 3 кн. Кн. 2. Цифрова схемотехніка: Підручник / В.І. Бойко, А.М. Гуржій, В.Я. Жуйков та ін. – 2–ге вид., допов. і переробл. – К.: Вища шк., 2004. – 423 с.

5. Бабич М. П., Жуков І. А. Комп’ютерна схемотехніка. Навч. посібник. – К.: МК–Пресс, 2004. — 412 с .

6. Комп’ютерна схемотехніка : підручник / [Азаров О. Д., Гарнага В. А., Клятченко Я. М., Тарасенко В. П.]. – Вінниця : ВНТУ, 2018. – 230 с.

7. Электротехника и электроника в экспериментах и упражнениях: Практикум на Electronics Workbench: Панфилов Д. И., Чепурин И. Н., Миронов В. Н., Обухов С. Г., Шитов В. А., Иванов В. С.: В 2 т. /Под общей ред. Д. И. Панфилова — Т. 2: Электроника. — М.: МЭИ, 2004. – 325 с.

Навчальне видання

Методичні вказівки
до виконання практичних занять з дисципліни

«Комп'ютерна схемотехніка»

для студентів усіх форм навчання за спеціальністю
123 «Комп'ютерна інженерія»

Укладачі: НОСИК Андрій Михайлович
ОНИЩЕНКО Валерій Валентинович
СТАТКУС Андрій Віталійович

Відповідальний за випуск проф. Порошин С. М.
Роботу рекомендував до друку проф. Заполовський М. Й.

В авторській редакції

План 2021 р., поз. 265

Підп. до друку 14.12.2021 р. Формат 60×84 1/16. Папір офсетний.
RISO–друк. Гарнітура Тайм
Ум. друк. арк. 6.4
Наклад 50 прим. Замовлення № . Ціна договірна

Видавничий центр НТУ «ХПІ».
Свідоцтво про державну реєстрацію ДК № 5478 від 21.08.2017 р.
61002, Харків, вул. Кирпичова, 2

Видавництво та друк ФО–П Єфименко С.А.
61166, Україна, м. Харків, вул. Коломенська, 27
Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру видавців – виготовлювачів і розповсюджувачів видавничої продукції
ДК № 6869 від 08.08.2019 р.