Тема 1. **Загальні принципи побудови ЕОМ**

Почнемо з визначення деяких понять.

- ЕОМ (ЕЛЕКТРОННА ОБЧИСЛЮВАЛЬНА МАШИНА) – це, мабуть, найпоширеніше поняття до зовсім недавнього часу. Спочатку воно стосувалося перших електронних автоматичних пристроїв для опрацювання інформації. Однак з часом на ЕОМ почали щораз більше накладати функції інформаційного плану: зберігання, шукання, сортування, опрацювання інформації (в тім числі графічної, текстової, образної (передавання зображень), звукової). Розрізняють спеціалізовані ЕОМ; комплекси ЕОМ; універсальні ЕОМ; міні-, мікро-ЕОМ.

- ОС – обчислювальна система – пристрій, що сприймає інформацію у вигляді даних, які можуть бути зображені у аналоговій або цифровій формі, зберігає дані, опрацьовує їх з великою швидкістю, пересилає дані всередині системи через лінію зв’язку і видає результат цих дій як інформацію. (Не треба плутати з поняттям операційна система, що також має скорочене позначення ОС). Обчислювальні системи можуть бути:
  - аналоговими (безперервної дії);
  - цифровими (дискретними);
  - гібридними.

- Комп’ютер (англ. *computer* – обчислювач) – спочатку в англомовних країнах, а тепер майже всюди, назва електронної обчислювальної машини. Головно слугує для передавання, опрацювання і зберігання інформації. Найбільше вживана форма випуску – персональні комп’ютери (ПК).

1.1. Поняття про інформацію

Про сучасні комп’ютери, їхню будову, можливості та принципи функціонування йтиметься дещо пізніше, спочатку ж розглянемо, що таке інформація і як її можна відображати в ЕОМ. Слово *інформація* походить від лат. *informatio*, що означає роз'яснення, виклад, обізнаність. Це одне з найзагальніших понять науки. Поняття інформації – одне з головних понять кібернетики (подібно до поняття енергії у фізиці). Розділ науки, присв'ячений технічним проблемам

інформації (окрім семантичних та прагматичних), називають *теорією інформації.* Різні способи зображення інформації, спеціально пристосовані для конкретних випадків, пов'язаних з передаванням, зберіганням та опрацюванням інформації, розглядає теорія кодування.

У комп'ютері маємо справу з дискретними повідомленнями, або машинними одиницями інформації. Насамперед постає питання, як виразити, поміряти і передати інформацію? Вперше цю задачу розв’язано у працях Клода Шеннона [1], де запроваджено поняття кількості інформації. Дещо спрощено схема викладу цього поняття може бути така.

Нехай нам потрібно скласти слово як комбінацію з букв алфавіту. Алфавіт містить P букв (чи символів), а слово – x символів. Очевидно, таких слів можна скласти N (правда, невідомо, чи всі вони будуть мати мовний зміст):

$$N=P^x \text{ (кількість розміщень).} \qquad (1.1)$$

Припустимо, що кількість інформації у якомусь повідомленні пропорційна до його довжини. Тоді з (1.1) після логарифмування отримаємо

$$\log_a N= x \log_a P \qquad (1.2)$$

Величину $x\log_a P$ визначимо як кількість інформації I. Знайдемо найменше I, яке можна було б прийняти за одиницю інформації. Очевидно, це буде при P=2, бо при P=1 , I=0. Отже,

$$I_{\min}=x \log_a 2. \qquad (1.3)$$

Найпростіше слово складається з однієї букви, тоді

$$I= \log_a 2. \qquad (1.4)$$

Це і є кількість інформації найпростішого слова.

Вибір основи логарифма – довільний. Зважаючи на те, що в обчислювальних системах головно послуговуються двійковою арифметикою, приймемо основу логарифма a=2. Тоді

$$I= \log_2 2=1. \qquad (1.5)$$

Ця одиниця інформації у двійковому алфавіті називається *біт* і може мати значення 0 і 1 (походить від англ. **bi**nary digi**t** і, як стверджує К. Шеннон [1], це слово запропонував уживати відомий американський вчений-статистик Джон Тьюкі). Отже, кількість інформації у будь-якому слові двійкового алфавіту завжди дорівнює кількості бітів у ньому, тобто

$$I=x \log_2 2=x, \qquad (1.6)$$

а кількість можливих повідомлень N легко визначити з (1.2):

$$\log_2 N= x \log_2 2, \text{ тобто } N=2^x. \qquad (1.7).$$

Ще раз зазначимо, що таке поняття кількості інформації виникло з задач теорії зв'язку і, по суті, застосовне саме до них.

Приклад: Кількість можливих слів, які містять три біти, є 8. Ось вони: 000, 001, 010, 011, 100, 101, 110, 111. Це не що інше, як числа вісімкової системи числення.

Кількість можливих слів, які містять чотири біти $N = 2^4 =16$. Це такі комбінації: 0000, 0001, 0010, 0011, 0100, 0101, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111, тобто числа шістнадцяткової системи числення .

Біт є найменшою одиницею вимірювання інформації. На практиці частіше використовують похідні одиниці: *байт, Кбайт, Мбайт, Гбайт, Тбайт.* Між ними є такі співвідношення:

1 *байт*=8 *бітів* (може бути 2^8=256 різних байтів)

1 *Кбайт*=2^{10} *бітів*=1024 *байти* (скорочено позначають 1К)

1 *Мбайт*=1024 *Кбайти*

1 *Гбайт*=1024 *Мбайти*

1 *Тбайт*=1024 *Гбайти*

До машинних одиниць інформації належать також: слово, запис, блок, файл. Деякі машинні одиниці мають аналоги з натуральними одиницями інформації: розряд, символ, поле, масив, запис.

1.2. Основи алгебри логіки

Усі пристрої ЕОМ складаються з елементарних логічних схем. Їхнє функціонування ґрунтується на законах і правилах алгебри логіки, яка оперує двома поняттями: істина і фальш. На честь її винахідника, англійського математика Джорджа Буля, алгебру логіки ще називають булевою алгеброю. Основу цієї алгебри становлять дві бінарні операції (*кон'юнкція* та *диз'юнкція*) і одна унарна

(*заперечення*). Крім цих трьох, уводять і інші, однак доведено, що будь-яку з них можна виразити за допомогою формули, у якій використано тільки три базові. (Наприклад, для функції від двох змінних таких логічних функцій є 16).

Мовою алгебри логіки будь-яку функцію зображають у вигляді таблиці відповідності всіх можливих логічних змінних та вихідних логічних функцій. Це так звана *таблиця істинності*.

Розглянемо головні логічні функції.

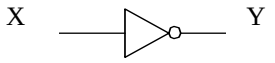
Логічне заперечення. Позначають НЕ.

Означення. Логічне заперечення НЕ змінної *X* є логічна функція *Y*, яка істинна тільки тоді, коли *X* хибна, і навпаки. Інша назва: *інверсія*.

У символах алгебри логіки записують: $Y = \overline{X}$.

Графічно на схемах позначають кружечком на вході, чи виході логічного символу (у системі ЄСКД):

або

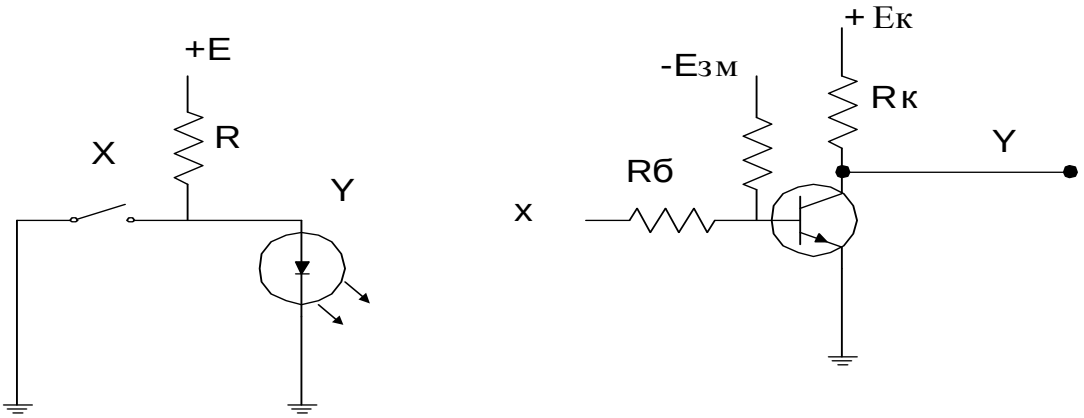


в американському стандарті.

Таблиця істинності:

X	0	1
Y	1	0

Схемна реалізація – інвертор

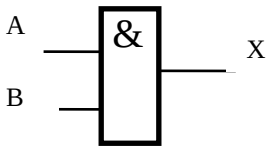


Логічне множення. Позначають І.

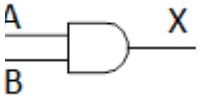
Означення. Логічним множенням двох змінних *A* і *B* є логічна функція *X*, яка істинна тільки тоді, коли одночасно істинні вхідні змінні. Інша назва: *кон'юнкція.*

Запис: $X = A * B,$ або $X = A \wedge B$.

Графічне зображення:



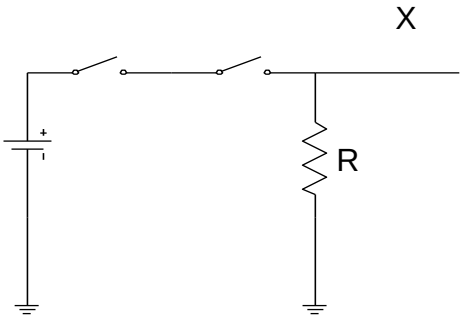
або у американському стандарті

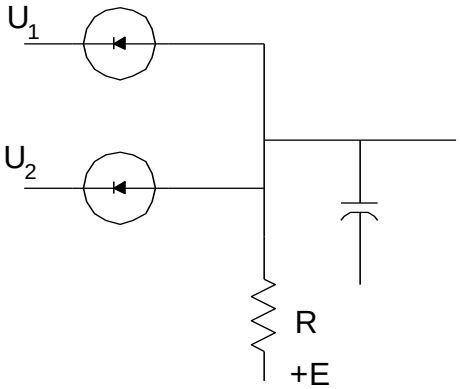


Таблиця істинності:

A	0	0	1	1
B	0	1	0	1
X	0	0	0	1

Схемна реалізація:





Логічне додавання. Позначають АБО.

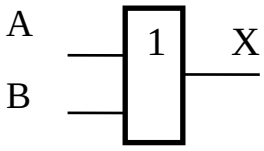
Означення. Логічною сумою змінних *A* і *B* є логічна функція *X*, яка істинна, якщо хоча б одна із вхідних величин істинна.

Інша назва: *диз'юнкція.*

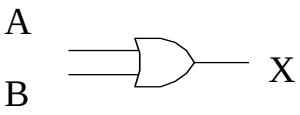
Запис:

$X = A + B,$ або $X = A \vee B$.

Графічне зображення:



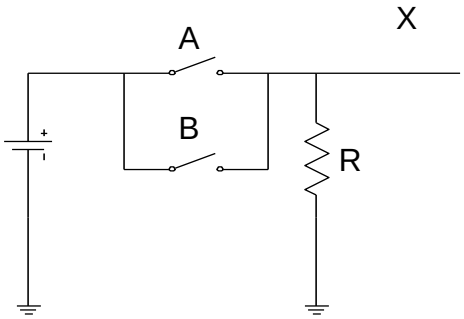
або у американському стандарті

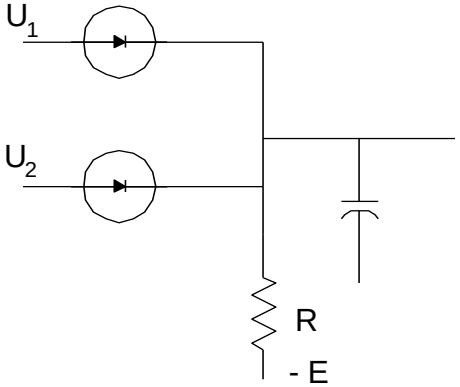


Таблиця істинності:

A	0	0	1	1
B	0	1	0	1
X	0	1	1	1

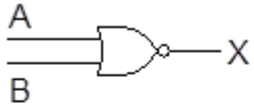
Схемна реалізація:

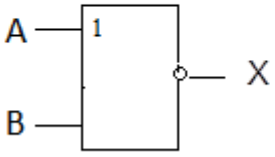


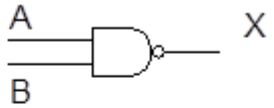


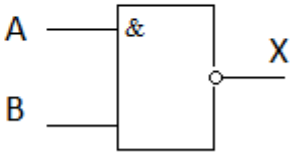
У реальних схемах символічні ключі замінюють діодами, транзисторами чи просто використовують напилені *p-n* - переходи у схемах більшої інтеграції. Схеми також можуть об’єднуватись відповідно до потреби реалізації певної логічної функції. Наприклад, дуже часто застосовують елементи

АБО-НЕ (елемент Пірса) чи І-НЕ (елемент Шеффера):









Функції, утворені із логічних змінних, можна перетворювати відповідно до правил або законів алгебри логіки.

Приклад.

1. Логічний добуток будь-якого аргументу на нуль завжди дорівнює нулю (використовують для "маскування" вхідних змінних).

2. Логічний добуток будь-якого аргументу на 1 дорівнює значенню самого аргументу.

3. Логічний добуток аргументу з його інверсією дорівнює нулю.

4. Правила де Моргана (закони інверсії) :

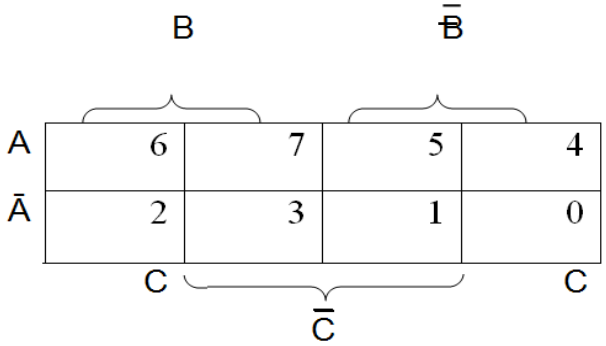
$$\overline{A * B} = \bar{A} + \bar{B}$$

$$\overline{A + B} = \bar{A} * \bar{B}$$

і т. д.

Ці правила і закони застосовують для спрощення логічних функцій і зведення їх до вигляду, що полегшує схемну реалізацію. Для забезпечення мінімізації логічних рівнянь використовують запис вхідних даних у вигляді діаграм Карно (Карнау).

На рисунку нижче показана діаграма Карно для трьох змінних.



Кожна клітинка діаграми відповідає логічному добутку прямого чи інверсного значення змінних, які присвоєно стовпцеві або рядкові, на перетині яких вона розміщена. Наприклад, клітинка з номером 7 є на перетині рядка зі значенням змінної *A* і стовпця зі значенням змінних *B* і *C* та відповідає логічному добутку *ABC*, клітинка з номером 0 відповідатиме значенню *A B C*. В діаграмах Карно значення змінних присвоюються так, щоб сусідні клітинки відрізнялися між собою тільки на одну змінну. Після цього згідно з описом логічної функції кількох змінних у клітинки записують значення логічних добутків з таблиці істинності. Якщо добутки є у сусідніх клітинках, то з загального виразу можна вилучити одну змінну, яка трапляється у прямому та інверсному кодах. Якщо добутки утворюють квадрат, то з загального виразу можна вилучити дві змінні.

1.3. Елементна база ЕОМ

Головні логічні елементи та вузли

Елементами ЕОМ називають пристрої, які виконують логічні функції, запам'ятовують інформацію, перетворюють її, а також формують та підсилюють сигнали.

З деякими найпростішими елементами ми вже ознайомлені. Це інвертор, пристрій для логічного додавання і множення. Відомо також, що використовуючи їхню різноманітну комбінацію, можна будувати складні логічні схеми.

Головним елементом пам'яті ЕОМ є ***тригери*** - логічні пристрої, які мають два стійкі стани. Для перемикання тригерів з одного стану в інший використовують вхідні логічні схеми. За способом перемикання розрізняють такі тригери:

- *RS*-тригери з роздільним установленням 0 і 1;
- *D*-тригер із затримкою;
- *JK*-тригери (універсальні);
- *T*-тригери (лічильні);

Ці назви походять від перших букв вхідних сигналів: *S* (set – встановити); *R* (reset – вимкнути); *T* (toggle – релаксатор); *J* (jerk – різко увімкнути); *K* (kill – різко вимкнути); *D* (delay – затримка). Вихідний сигнал тригера прийнято позначати буквою *Q*.

RS-тригери з роздільним установленням 0 та 1

Найпростіший запам'ятовувальний елемент має два входи: *S* і *R*. У разі комбінації сигналів *S*=1, *R*=0 тригер буде встановлений в одиничний стан, тобто *Q*=1.

Якщо ж комбінація вхідних сигналів *S*=0, *R*=1, то тригер встановиться в нуль, *Q*=0. Комбінація *S*=0, *R*=0 залишить тригер у попередньому стані, а от комбінація *S*=1, *R*=1 буде невизначеною, отже, забороненою для такого елемента ($Q=\overline{Q}=0$).

Функціонування *RS*-тригера описує вираз

$$Q(t+1)= S(t)\vee Q(t)R'(t),$$

де $S(t)R(t)=0$, t – момент часу, який передує зміні стану. (Тут і надалі в тексті позначення інверсії $\bar{Y}\equiv Y'$).

Такі тригери реалізують на вже відомих нам логічних елементах І-НЕ, АБО-НЕ, з'єднуючи їхні входи і виходи навхрест. На схемах їх зображають прямокутником, у якому з лівого боку позначають вхідні сигнали *S*, *R*, а з правого – вихідні. Вхід *S* називають установлювальним, а вхід *R* – скидальним. Приклади таких *RS*-тригерів показано на рис.1.1.

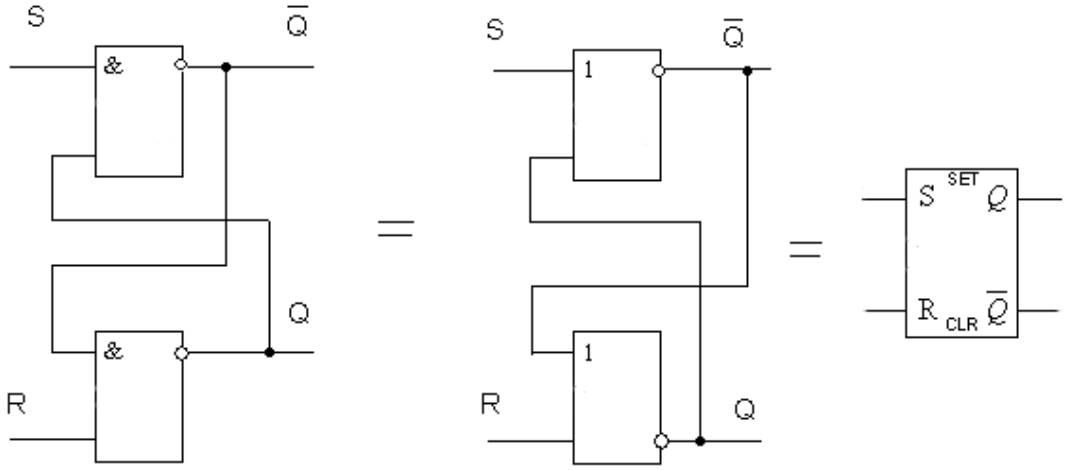


Рис. 1. 1. *RS* - тригери на базі елементів І та АБО

Іноді вводять синхронізацію зміни стану тригера (позначають вхід *C* (clocking – синхронізація)). При *C*=1 тригер перемикається за законом *RS*-тригера, при *C*=0 він зберігає попередній стан.

***D*-тригер (затримки)**

Такий тригер має один інформаційний вхід і вхід для синхроімпульсу. Головне призначення такого тригера – затримка і збереження сигналу, який подають на вхід за умови *C*=1. На рис.1.2 показана схема синхронізованого двотактного *D*-тригера, який затримує сигнал на один період; і його описує формула:

$$Q(t+1)= D(t).$$

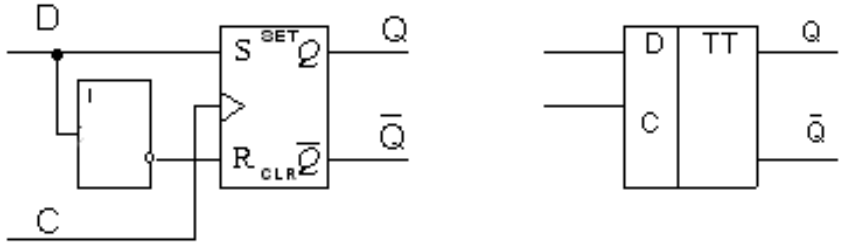


Рис. 1. 2. Синхронізований *D*-тригер

Різновидом *D*-тригера є *DV*-тригер (*V*-value – вентиль), у якому через вхід керування *V* дозволено перемикання, або тригер не реагує на перемикання при *V*=0.

***JK*-тригер (універсальний)**

Працює за принципом *RS*-тригера, проте в ньому комбінація *J*=*K*=1 не заборонена, у цьому разі він змінює свій стан на протилежний до того, у якому був. Переважно його реалізують за двоступеневою схемою (див. рис.1.3).

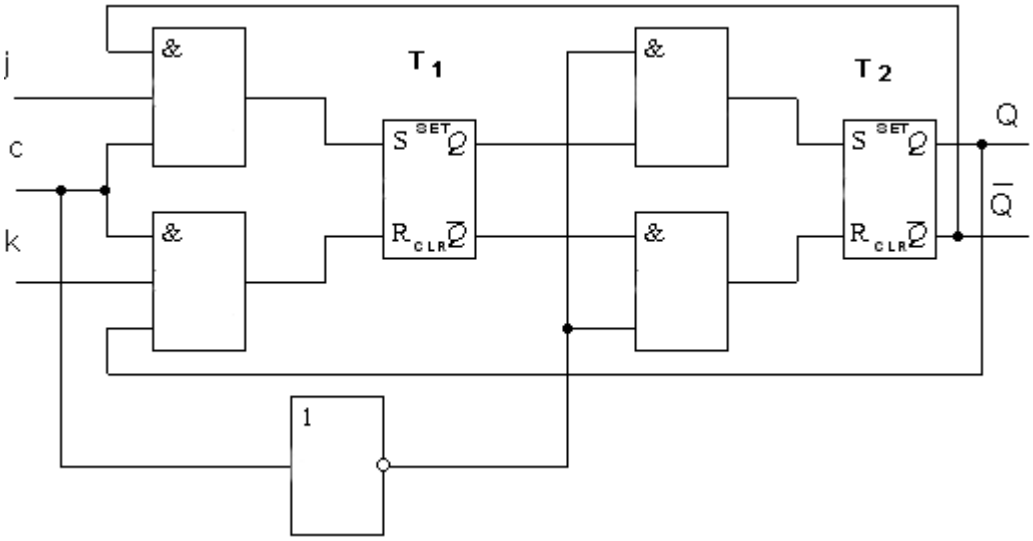
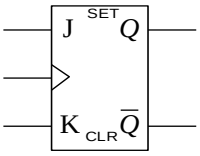


Рис. 1. 3. Схемна реалізація *JK*-тригера

На схемах позначають так:



При *C*=1 відбувається записування інформації в *T1*, при *C*=0 – переписування інформації з *T1* в *T2*. Функція переходів *JK* за умови *R*=*S*=1 має вигляд

$$Q(t+1)= Q(t)K'(t)\vee Q'(t)J(t).$$

***T*-тригер (лічильний)**

Цей тригер змінює свій стан у разі надходження кожного вхідного імпульсу. Він може бути реалізований на базі *JK*-тригера. При значеннях змінних *J*=*K*=1 сигналом *C* можна змінювати стан тригера. Позначають як на рис. 1. 4.

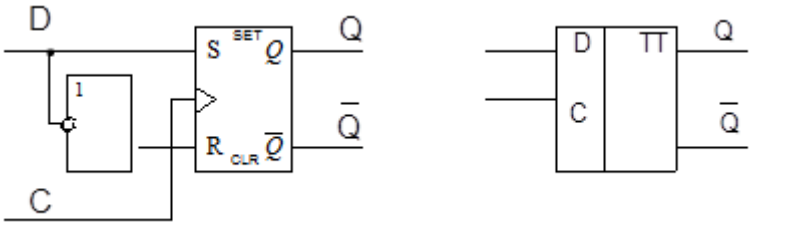


Рис. 1. 4. Схемна реалізація лічильного тригера

T-тригер реалізує таку функцію:

$$Q(t+1)= Q(t)T'(t)\vee Q'(t)T(t).$$

За допомогою описаних логічних пристроїв можна будувати різні блоки чи пристрої ЕОМ. До них належать: регістри, лічильники, шифратори, дешифратори, мультиплексори, суматори та арифметико-логічні пристрої як цілочислової так і плаваючої арифметики.

Мультиплексор – пристрій, який виконує передавання сигналів від однієї з вхідних ліній у вихідну. Вхідну лінію вибирають кодом, який надходить на входи керування мультиплексора. Уживається також інша назва – комутатор (рис. 1. 5).

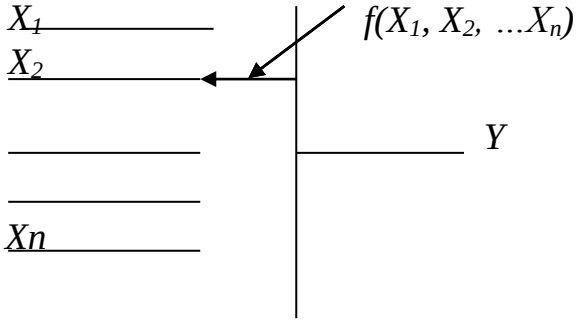


Рис. 1. 5. Схема комутування вхідних сигналів *X* на вихідну лінію *Y* за допомогою функції $f(X_1, X_2, \ldots Xn)$

Регістр **–** це вузол ЕОМ, який виконує тимчасове збереження та перетворення інформації. Регістри будують на основі тригерних схем. Кількість тригерів визначає розрядність слів, які записують чи зберігають у регістрі. Регістри є послідовні і паралельні, одно- і двотактні, зсувні і перетворювальні. Розглянемо кілька прикладів схем регістрів. Побудуємо регістр, наприклад, трирозрядний. Очевидно, що для цього треба мінімум три тригери, які будемо використовувати для запису кожного розряду.

Використаємо звичайні *RS*-тригери. Трохи добудуємо схему ліворуч і праворуч. Ліворуч організуємо синхронний запис

інформації, а праворуч – формування прямого та оберненого коду. Перед записом інформації всі тригери сигналом керування R поставило в нуль $\Rightarrow$ (Q=0). Запис у тригери виконується за тактовим імпульсом $Ti_{1.}$, Власне ці два сигнали (R і Ti_1) визначають тип регістра, який називають двотактовим паралельної дії (рис. 1. 6).

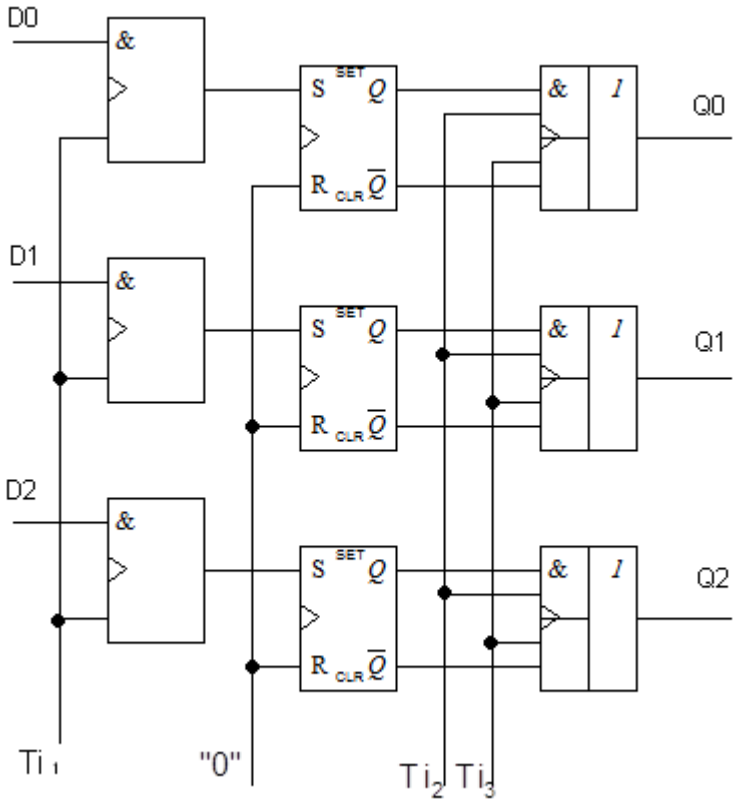


Рис. 1.6. Двотактовий регістр паралельної дії

Код регістра видають за допомогою другого і третього тактових імпульсів: Ti_2 - прямий код, Ti_3 – обернений код.

Якщо на вхід можна подавати парафазний код (тобто вхідне значення подають у прямому й оберненому коді), то відпадає потреба у такті установлення в "0", тобто отримаємо однотактний паралельний регістр.

Регістр є дуже зручним пристроєм для "зсування" інформації праворуч чи ліворуч або перетворення послідовного коду у паралельний (рис. 1. 7). Для цього тригери регістра потрібно з'єднати послідовно через невелику затримку.

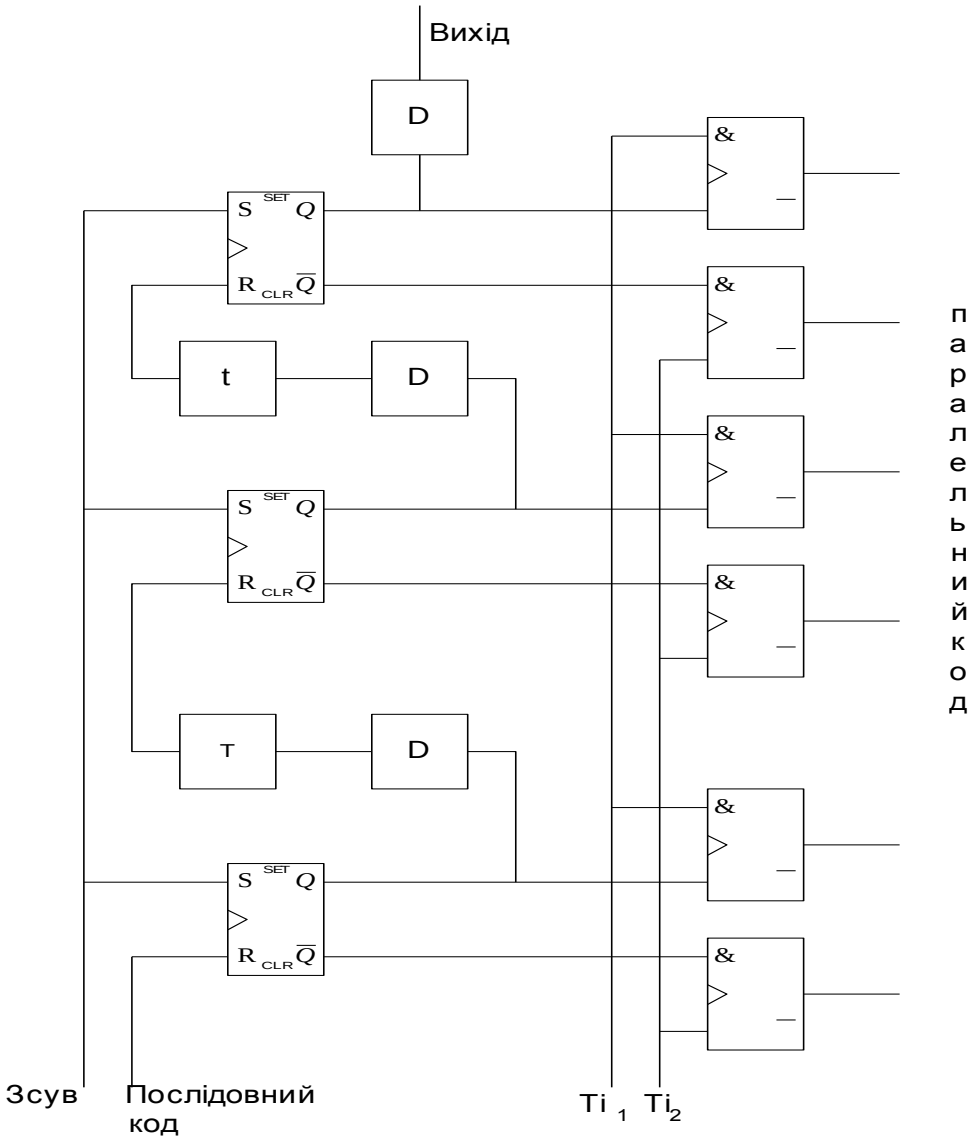


Рис. 1. 7а. Регістр для перетворення послідовного коду у паралельний. D – диференціювальна ланка, яка формує імпульс для встановлення тригера по входу *R; t* – невелика лінія затримки

На вхід *R* першого тригера надходить послідовний код, наприклад, 101. Входи *S* усіх тригерів об'єднані і на них надходить імпульс зсуву, який одночасно виконує функцію встановлення в "0" регістра. Тригери спрацьовують тільки від додатних імпульсів. Це означає, що встановлення наступного тригера можливе лише у разі перекидання попереднього з "1" у "0". Не важко зауважити, що через три кроки послідовний код 101 буде занесений у регістр і може бути прочитаний у паралельному (прямому або інверсному) коді.

Таку ж функцію виконує регістр, зображений на рис 1.7б.

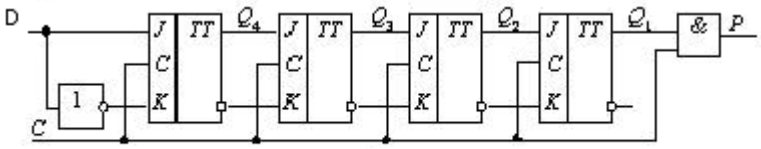


Рис. 1. 7б. Регістр для перетворення послідовного коду у паралельний на базі лічильних тригерів.

Ще розглянемо роботу однотактового регістра зі зсуванням числа праворуч (рис. 1. 8).

Тут для спрощення схеми ланки зчитування числа пропущено. Для зсуву числа потрібно подати *n* імпульсів зсуву.

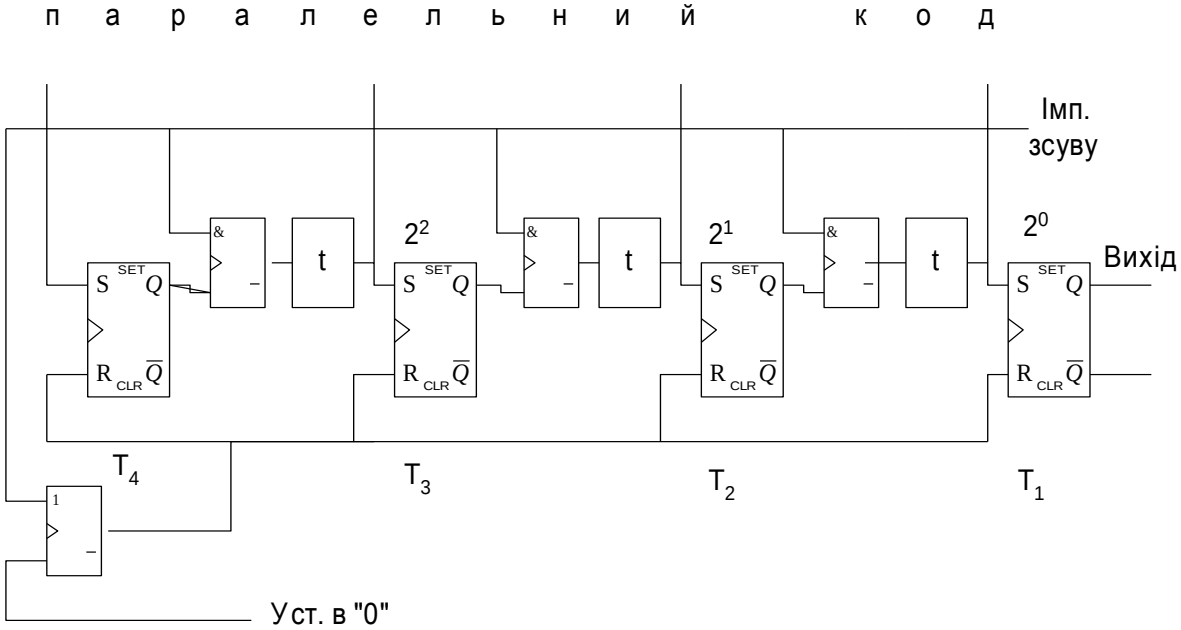


Рис. 1. 8. Однотактовий регістр для зсування коду праворуч

Нехай у регістрі є код 1011. Стан тригерів після кожного імпульсу зсуву t_i буде змінюватися відповідно до таблиці:

Імпульс	t_1	t_2	t_3	t_4
T_4	1	0	0	0
T_3	0	1	0	0
T_2	1	0	1	0
T_1	1	1	0	1

$t_2 = t_1 + t_{затр.}$

$t_3 = t_2 + t_{затр.}$

$t_4 = t_3 + t_{затр.}$

Якщо використовувати парафазний (такий, у якому присутні одночасно прямі та інверсні значення) код, то нема потреби у лініях затримки.

Затримку можна виконати також за допомогою тригера, що значно підвищує надійність роботи схеми. Зсув можна робити як праворуч, так і ліворуч. Такі регістри називають *реверсивними.*

Якщо вихід T_1 подати на вхід T_4 то легко зауважити, що в разі подання імпульсів зсуву інформація у регістрі буде циркулювати. Такі регістри називають *кільцевими.*

Лічильник – пристрій, призначений для підрахунку кількості імпульсів. Лічильники бувають підсумовувальні, віднімальні та реверсивні. Реверсиний лічильник залежно від перекомутації може бути підсумовувальним або віднімальним. Будують лічильники на основі тригерів, використовують у пристроях керування та в арифметичних пристроях для рахунку номерів команд, кількості циклів програми, кількості тактів у разі множення і ділення, а також як суматори. Приклад 4-розрядного підсумовувального двійкового лічильника показано на рис. 1. 9.

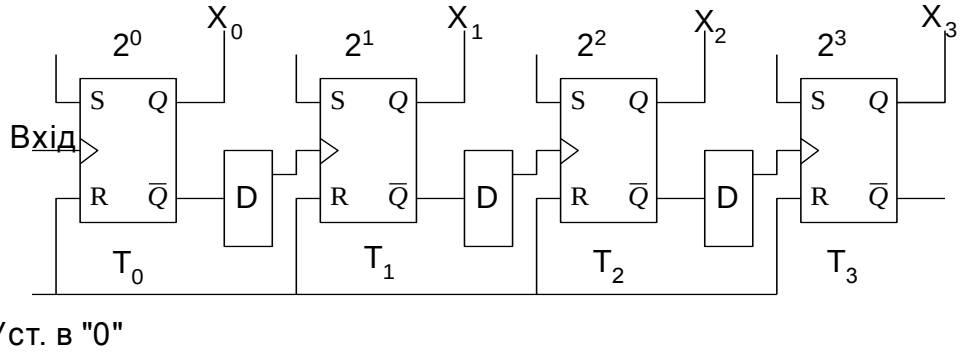


Рис. 1. 9. Схема 4-розрядного підсумовувального двійкового лічильника

Такий лічильник порахує до 16 (1111) і знову стане в "0". Максимальна кількість імпульсів, яку може підрахувати двійковий лічильник, що складається з *n* розрядів, дорівнює 2^n-1.

Віднімальний лічильник отримаємо тоді, коли сигнал знімати з інверсного виходу тригера.

Лічильники, які однаково можна використовувати як для додавання, так і для віднімання імпульсів, називають *реверсивними*.

Входи *S, R* застосовують для занесення початкового коду у лічильник або скидання в "0".

Іноді потрібно отримати значення $K \neq 2^n$. Тоді будують спеціальні перерахункові схеми, де від певних розрядів уводять обернений зв'язок. Перерахункові схеми з K=10 називають декадними лічильниками, які застосовують для побудови десяткових лічильників (рис. 1.10).

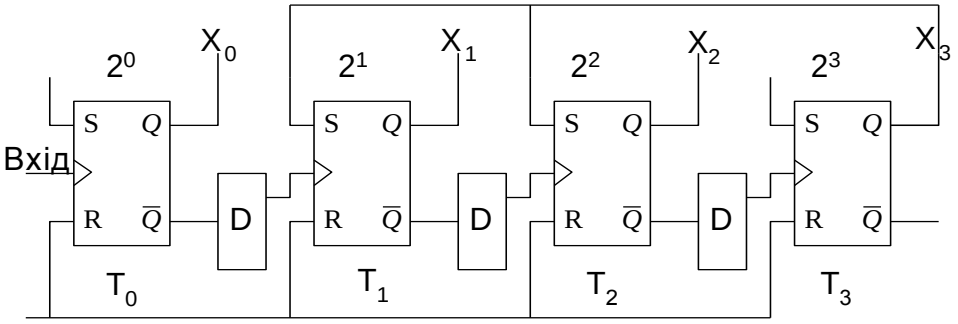
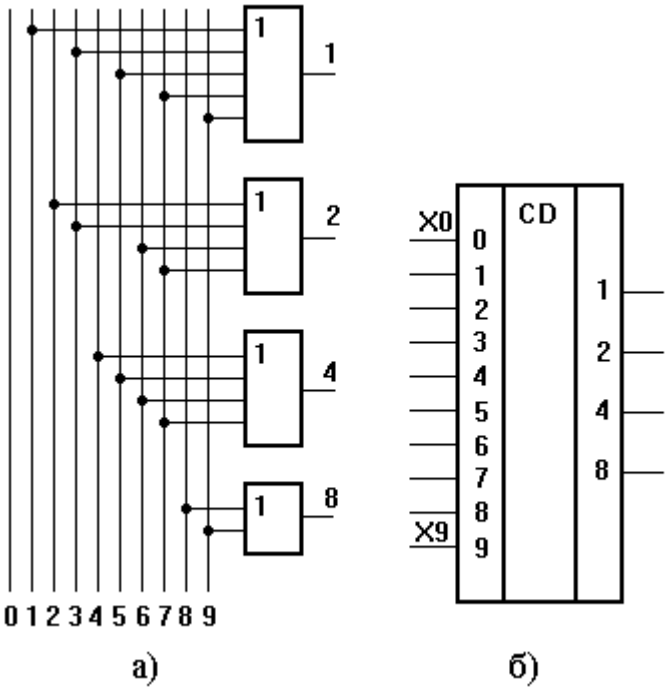


Рис. 1. 10. Декадний лічильник

У разі подання перших семи імпульсів схема працює у звичайному режимі. Восьмий імпульс установить тригери T_0, T_1 і T_2 в стан "0", а тригер T_3 – у стан "1". У цьому випадку внаслідок оберненого зв'язку сигнал перенесення з T_3 встановить в одиничний стан T_1 і T_2: отримаємо стан лічильника 1110. Дев'ятий імпульс установить в "1" T_0, а десятий скине в "0" усі тригери схеми. Після цього схема знову готова рахувати до 10.

Шифратор – це вузол ЕОМ, що виконує операцію формування відповідного двійкового коду в разі появи сигналу на одному з входів шифратора. У обчислювальній техніці застосовують переважно багатоімпульсні шифратори, які дозволяють кодувати дані при записуванні програм на носії інформації чи дані, виражені якоюсь фізичною величиною (напр., напругою). Приклад шифратора на 10 входів показано на наступному рисунку (а – принципова схема, б – схемне позначення стандартних шифраторів в документації).



Дешифратор – пристрій який виконує перетворення *n*-розрядного двійкового коду в однорозрядний з основою $p=2^n$, тобто функцію обернену до шифрування. Використовують у багатьох пристроях, зокрема у пристроях керування, для розшифровування коду операції та видавання сигналів керування у ті кола машин, які повинні працювати під час виконання цієї операції; у запам'ятовувальних пристроях для розшифрування адреси чи команди, записування або читання коду з певної комірки пам'яті.

Розглянемо роботу простої схеми дешифрування на два розряди. Як і в попередніх випадках, побудуємо її на основі тригерних схем (рис. 1. 11). Пояснення роботи такого пристрою не потребує особливих зусиль.

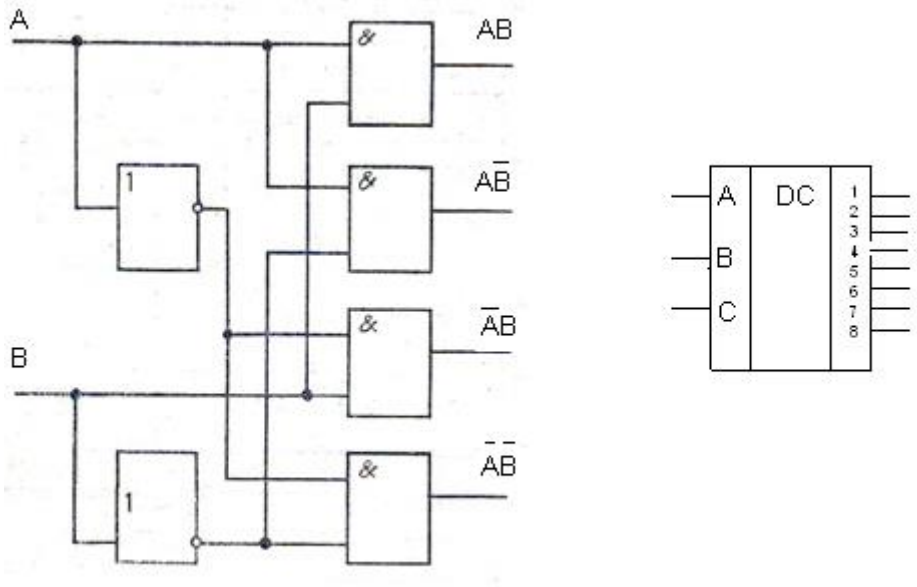


Рис. 1. 11. Схема дворозрядного дешифратора (з правого боку показано схемне позначення трирозрядного дешифратора)

Суматор – пристрій, що виконує сумування чисел на підставі правил порозрядного додавання з урахуванням переносів, які спрямовують у старші розряди.

Суматори класифікують:

1. за виглядом елементів, які використовують – комбінаційного та накопичувального типів;
2. за способом уведення-виведення чисел – паралельної та послідовної дії;
3. за способом зображення чисел – двійкові та десяткові;
4. за способом організації перенесення – з послідовним та наскрізним перенесенням.

Розглянемо суматори комбінаторного типу (комбінація логічних елементів І, АБО, НЕ)

Сигнал на виході (сума) утворюється тільки в разі визначенної комбінації вхідних сигналів (доданків), які подаються одночасно. Після зникнення вхідних сигналів вихідний сигнал зникає (запам’ятовувальних властивостей немає). Тому такі суматори працюють з регістром, у який записується результат.

Процес сумування і перенесення з сусіднього розряду в однорозрядній сумувальній схемі розбито на дві аналогічні операції: сумування дворозрядних доданків і сумування з отриманим результатом одиниці перенесення. Кожну з функцій виконує схема, яку називають *напівсуматором.*

Розглянемо роботу напівсуматора, яку описують логічними виразами для суми:

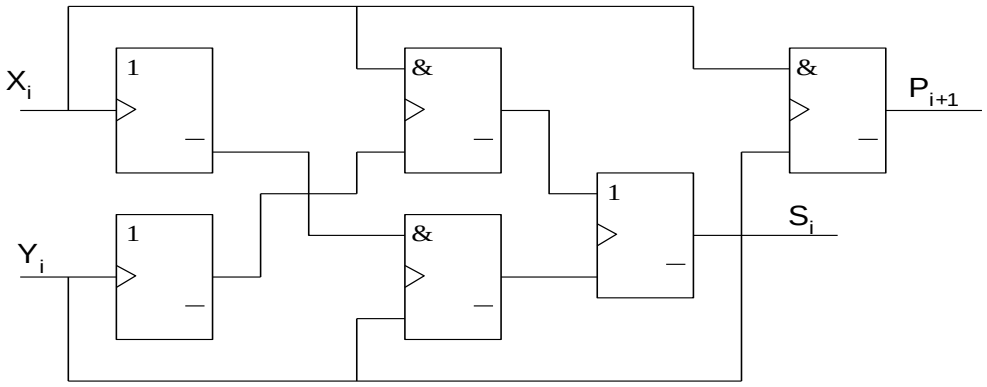
$$S_i = X_i\overline{Y}_i + \overline{X}_iY_i = X_i\overline{Y}_i + \overline{X}_iY_i + X_i\overline{X}_i + Y_i\overline{Y}_i =$$

$$=X_i(\overline{X}_i+\overline{Y}_i) + Y_i(\overline{X}_i+\overline{Y}_i) = (X_i+Y_i)(\overline{X}_i+\overline{Y}_i)=$$

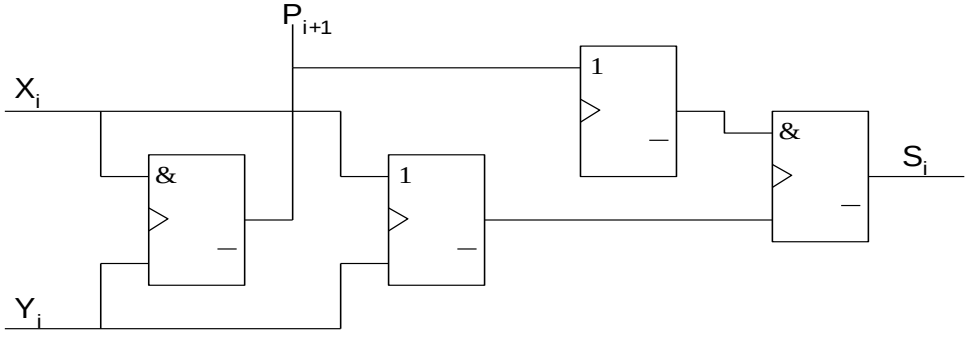
$$=\overline{X_iY_i}(X_i+Y_i)$$

та перенесення $P_{i+1} = X_iY_i$.

Результат для суми отримано внаслідок певних алгебричних перетворень. Для прикладу розглянемо схеми, які реалізують проміжний і кінцевий етапи перетворень (рис. 1. 12).



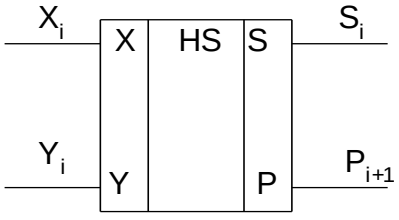
а



б

Рис. 1.12. Схеми напівсуматорів комбінаційного типу:
а – проміжний, б – кінцевий

На схемах позначають



Якщо додати ще один напівсуматор, то отримаємо схему повного суматора на три входи (рис. 1. 13).

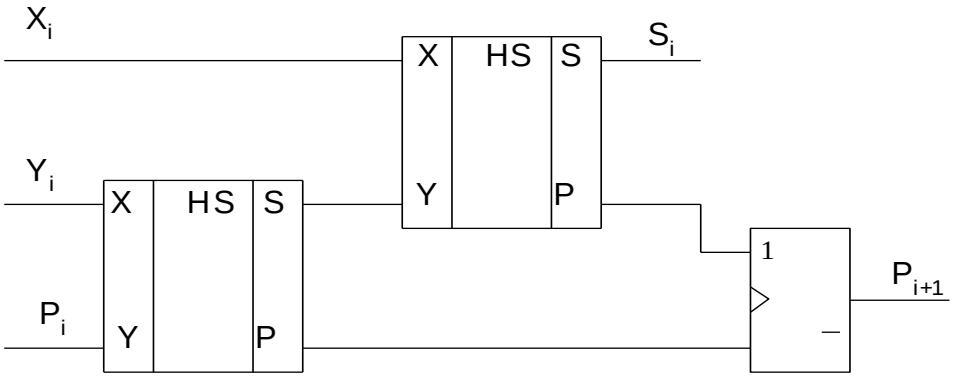
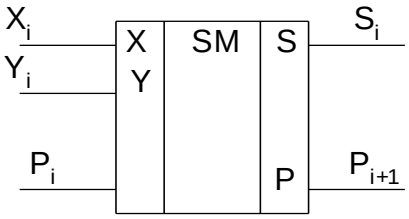


Рис.1.13. Схема повного суматора на три входи

Повний суматор на схемах позначають так:



Багаторозрядний суматор можна отримати простим послідовним з’єднанням однорозрядних суматорів (рис. 1. 14).

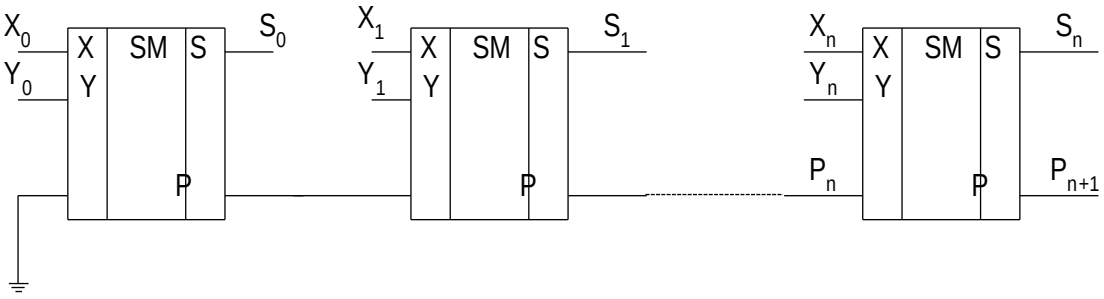


Рис. 1. 14. Схема багаторозрядного суматора

Для зменшення часу поширення сигналу перенесення використовують схеми прискореного перенесення.

Однорозрядний суматор можна використати для виконання операції віднімання. Для цього на вхід *Y* подають або прямий, або обернений код доданка, який відповідає його від'ємному значенню (рис. 1. 15).

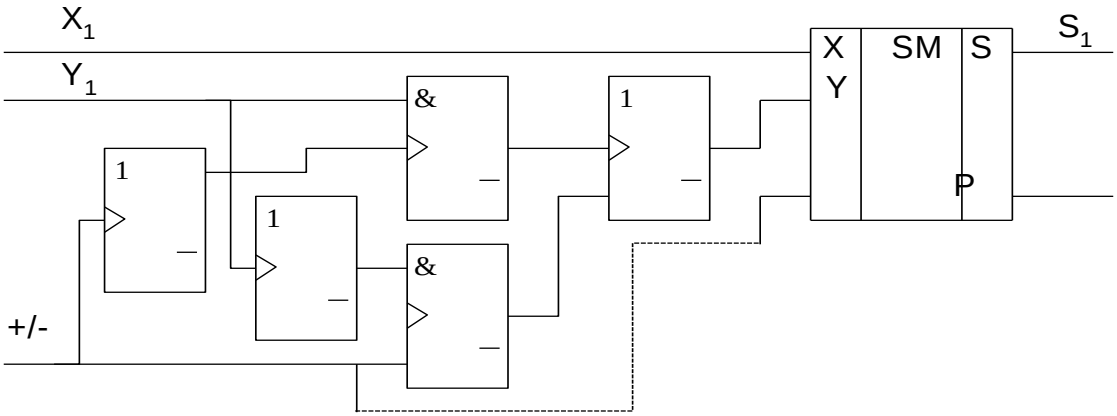


Рис. 1. 15. Використання однорозрядного суматора для операції віднімання

Залежно від сигналу керування +/- на вхід суматора подається прямий код *Y* (при +/- =0), або обернений код *Y* (при +/- =1). Для формування додаткового коду доданка до його інверсного коду треба додати 1 молодшого розряду. Це досягають з'єднанням входу керування +/- з входом суматора молодшого розряду.

Трохи складніше, однак можна зібрати з наведених вище логічних схем і вузлів пристрої для виконання операції множення та ділення двійкових чи десяткових чисел.

Список літератури до теми

1. *Злобін Г.Г, Рикалюк Р.Є.* Архітектура та апаратне забезпечення ПЕОМ: Навч.посіб. –К., 2006, 2012.
2. *Шеннон К.* Работы по теории информации и кибернетике / Пер. с англ.– М., 1963.
3. *Хэмминг Р.В.* Теория кодирования и теория информации / Пер. с англ. – М., 1983.
4. *Биркгоф Г., Барти Т.* Современная прикладная алгебра / Пер.с англ. –М.; 1976.
5. *Ланцов А.Л., Зворыкин Л.Н., Осипов И.Ф.* Цифровые устройства на комплементарных МПД интегральных микросхемах. –М., 1983.
6. *Горбунов В.Л., Панфилов Д.И., Преснухин Д.Л.* Справочное пособие по микропроцессорам и микроЭВМ. –М., 1988.

7. *Каган Б.М.* Электронные вычислительные машины и системы: Учеб.пособие для вузов.–М., 1991.
8. *Харрис Д. М., Харрис С.Л.* Цифровая схемотехника и архитектура компьютера/второе издание. - Morgan Kaufman/ © English Edition 2013.

Контрольні запитання до теми

1. Яка відмінність у термінах ЕОМ і ПК?
2. Що називаємо інформацією?
3. Дайте означення біта.
4. Назвіть три головні функції алгебри логіки.
5. Запишіть закони інверсії.
6. Що таке тригер і які типи тригерів Ви знаєте?
7. Які стани дозволені, а які заборонені у *RS*-тригері?
8. Які функції регістрів і які типи регістрів використовують у побудові ЕОМ?
9. Як побудувати десятковий лічильник?
10. Для чого використовують шифратори та дешифратори?
11. Які типи суматорів Ви знаєте?