Значення напруги $I_{к0} \cdot R_{б}$ – порядку десятків мілівольт, тому с запасом $E_I \approx (0{,}3 \div 0{,}5)$ В. Спрощена модель ключа являє собою розімкнуті колектор і емітер, при напрузі на базі (0,3÷0,5) В.

4.4 Контрольні питання

1. Намалюйте RC-ланку і встановіть умови, при яких вона буде такою, що диференціює.

2. При яких умовах RC-ланка буде розділовою?

3. Назвіть основні джерела похибок в інтегруючих колах, напишіть формули для їхнього визначення.

4. Назвіть переваги інтеграторів на ОППС.

5. У чому перевага паралельних діодних обмежувачів відносно до послідовних? Назвіть основні недоліки діодних обмежувачів.

6. Наведіть повні і спрощені лінійні моделі транзисторів в областях насичення і відсічення. Які коефіцієнти передач транзистора в цих областях?

7. Наведіть методику розрахунку транзисторного ключа в режимі насичення, модель ключа, поясніть зміст коефіцієнта насичення *S* і його рекомендовані значення.

1. Викладіть методику розрахунку транзисторного ключа в режимі відсічення, наведіть модель ключа.

5 МАТЕМАТИЧНІ ОСНОВИ ПОБУДОВИ ЦИФРОВИХ ПРИСТРОЇВ

5.1 Системи числення

Система числення – це спосіб запису (зображення) чисел.

Системи числення, в яких ваговий коефіцієнт кожної цифри залежить від її положення у послідовності цифр, що зображає число, називаються позицій-

ними. У непозиційних системах значення кожної цифри постійне і не залежить від місця її розташування в числі. Всі системи числення, які використовуються в цифровій схемотехніці, є позиційними.

При розгляді позиційних систем важливим виступає поняття *базису*. *Базис системи числення* – це послідовність чисел, яка задає значення (вагу) кожної цифри в залежності від місця її розміщення.

Приклади базисів:

- десяткової системи числення: 10^0, 10^1, 10^2, ..., 10^n, ...;
- двійкової – 2^0, 2^1, 2^2, ..., 2^n, ...;
- вісімкової – 8^0, 8^1, 8^2, ..., 8^n, ...;
- шістнадцяткової – 16^0, 16^1, 16^2, ..., 16^n, ...

У загальному плані для позиційних систем числення базис можна записати в вигляді послідовних членів геометричної прогресії:

$$...P^{-m}, ..., P^{-2}, P^{-1}, P^0, P^1, P^2, ..., P^n, ...$$

Число P називається *основою системи числення*. У подальшому при розгляді систем числення основа зображатиметься у вигляді нижнього індексу в кінці числа.

Сукупність різних цифр, які використовуються в позиційній системі числення для запису чисел, називається *алфавітом системи*.

Будь-яке натуральне число A в P-ічній системі числення записується у розгорнутій і згорнутій формах запису. Наприклад, число A в P-ічній системі числення представляється в згорнутій формі так:

$$A = (a_n a_{n-1} \ldots a_1 a_0 a_{-1} a_{-2} \ldots a_{-k})_P ; \qquad (5.1)$$

у розгорнутій:

$$A = a_n \cdot P^n + a_{n-1} \cdot P^{n-1} + \ldots + a_1 \cdot P^1 + a_0 \cdot P^0 + a_{-1} \cdot P^{-1} + a_{-2} \cdot P^{-2} + \ldots + a_{-k} \cdot P^{-k} \quad (5.2)$$

Приклад 5.1 Перевести двійкове число $A = 11011_2$ в десяткову систему числення.

Розв'язання. Запишемо число в розгорнутій формі:

$A = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 2\,(2^2\ (2+1)+1)+1 = 27.$

При програмуванні як на мовах високого рівня, так і на мові Асемблера часто необхідно знати значення ступенів двійки до 16. У цифровій та мікропроцесорній техніці важливо пам'ятати ступені до 10. Приведемо їх, починаючи з 5:

$2^5 = 32;\quad 2^6 = 64;\quad 2^7 = 128;\quad 2^8 = 256;\quad 2^9 = 512;\quad 2^{10} = 1025.$

Знання цих чисел дає можливість більш спрощено розв'язувати задачі переведення десяткових чисел у двійковий код.

Приведене вище правило переведення цілих чисел у десяткову систему числення може бути використаним і для переведення дробових чисел.

Приклад 5.2 Перевести число $A = 0{,}11_2$ в десяткову систему числення.

Розв'язання. Запишемо число A в розгорнутій формі:

$A = 1 \cdot 2^{-1} + 1 \cdot 2^{-2} = 0{,}75_{10}.$

Формула (5.2) здебільшого використовується для переходу від системи числення з меншою основою до системи числення з більшою основою.

Ірраціональні дробові числа представляються у скороченій формі і переводяться аналогічно, або для них використовуються спеціальні алгоритми.

Приклад 5.3 Перетворити у двійковий код число 105_{10}.

Розв'язання. Операція перетворення виконується у послідовності, наведеній нижче.

$105 : 2 = 52 +$ залишок $1 = a_0$

$52 : 2 = 26 +$ залишок $0 = a_1$

$26 : 2 = 13 +$ залишок $0 = a_2$

$13 : 2 = 6 +$ залишок $1 = a_3$

$6 : 2 = 3 +$ залишок $0 = a_4$

$3 : 2 = 1 +$ залишок $1 = a_5$

$1 : 2 = 0 +$ залишок $1 = a_6$

Тобто, $105_{10} = A_2 = a_6\, a_5\, a_4\, a_3\, a_2\, a_1\, a_0 = 1101001_2$.

Типові помилки при реалізації такого алгоритму наступні: порушення порядку запису цифр, що одержуються; неправильне вписування крайньої ліворуч цифри.

Використовуючи ступені числа 2 з прикладу 1.5, скористаємось спрощеним способом переводу числа A_{10} у двійкову систему числення. Дійсно, можемо записати нерівності

$$2^7 > 105 > 2^6.$$

Звідси витікає, що двійкове число представляється 7-ма розрядами, старший з яких $2^6 = 65$. Оскільки різниця $105 - 64 = 41$ знаходиться в інтервалі $2^6 > 41 > 2^5$, то стверджуємо, що і наступний по старшинству розряд – шостий ($2^5 = 32$) – дорівнює 1. Наступна різниця $41 - 32 = 9 = 1001_2$. П'ятий розряд дорівнює нулю, і в результаті отримуємо ту ж саму відповідь.

Переведення числа A, що має дробову частину, з десяткової системи числення у двійкову має ту особливість, що ціла і дробова частини переводяться окремо.

Сформулюємо тепер правило переводу дробової частини з десяткової системи числення в P-ічну. Знову представимо її у розгорнутому вигляді:

$$A_{10} = a_{-1} \cdot P^{-1} + a_{-2} \cdot P^{-2} \ldots + a_{-k} \cdot P^{-k} + \ldots \qquad (5.3)$$

Перемножуючи ліву і праву частини (5.3) на P в правій частині виразу отримуємо:

$$a_{-1} + a_{-2} \cdot P^{-2} \ldots + a_{-k} \cdot P^{-k} + \ldots \qquad (5.4)$$

З отриманого результату можемо зробити висновок, що перша цифра a_{-1} дробової частини числа A в P-ічній системі числення дорівнює цілій частині результату перемноження десяткової дробової частини на P. Після чергового перемноження залишку дробової частини на P отримаємо значення a_{-2}:

$$(a_{-2} \cdot P^{-1} + \ldots + a_{-k} \cdot P^{-k+1} + \ldots) \cdot P.$$

Цей процес продовжується до тих пір, поки дробова частина результату перемноження лівої частини не стане рівною нулю або поки не буде виділений період повторності цифр.

Приклад 5.4 Перевести число $A = 0{,}375_{10}$ в двійкову систему числення.

Розв'язання. Виконуємо операцію множення в наведеній нижче послідовності.

0,375	×	2	=	0,75	0 – перша цифра результату
0,75	×	2	=	1,5	1 – друга цифра результату
0,5	×	2	=	1	1 – остання цифра результату

Внаслідок виконання перетворень отримали результат $0{,}375_{10} = 0{,}011_2$.

5.2 Коди та їх характеристика

5.3.1 Коди з паралельною формою представлення інформації

Система числення, в якій використовуються лише два знаки для відображення інформації, називається *двійковою*, основою якої є число 2. За аналогією з десятковою, двійкова система числення є позиційною, і будь-яке ціле десяткове число може бути представлене двійковим рядом, що вміщує лише "**1**" та "**0**" у відповідності з алгоритмом:

$$A_{10} = \sum_{i=0}^{k-1} a_i \cdot 10^i \rightarrow A_2 = \sum_{j=0}^{p-1} a_j \cdot 2^j , \qquad (5.5)$$

де $a_i = 0\ldots9$ – цифри i-го розряду десяткового числа; $a_j = 0\ldots1$ – відповідно, цифри j-го розряду двійкового числа. Розряди чисел рахуються зліва направо, починаючи зі старшого. Дробові числа представляються доповненням суми (5.5) від'ємними ступенями числа 2.

Електронні системи, що оперують сигналами, які відповідають лише рівням "**1**" та "**0**", називаються *цифровими.* Схеми, на основі яких реалізуються такі системи, також називаються *цифровими*, а розділи електроніки, що вивчають принципи побудови таких схем – *цифровою схемотехнікою*.

Як у теорії інформації, так і на практиці застосування цифрової схемотехніки використовується багато різноманітних кодів. Визначимось з основною термінологією.

Код – це універсальний спосіб відображення інформації при її зберіганні, передачі і обробці у вигляді системи однозначних відповідностей між елементами повідомлень і сигналами, за допомогою яких ці елементи можна зафіксувати. Іншими словами, *кодування* – це однозначне перетворення символів одного алфавіту в символи іншого, а код – правило, закон, алгоритм, при якому від-

бувається це перетворення. Комбінації символів, що належать до даного коду, називаються *кодовими словами*. Символи, за допомогою яких повідомлення трансформується в код, є *вторинним алфавітом*. Процес відновлення вмісту повідомлення за допомогою відповідного коду називається *декодуванням.* Необхідною умовою декодування є взаємно однозначна відповідність кодових слів у вторинному алфавіті символам первинного алфавіту та їх комбінаціям. При передачі кодових символів по лініях зв'язку вони повинні бути розділені так, щоб кожен символ міг бути прийнятим самостійно, що виконується з використанням різних принципів їх розділення. Розділення символів може бути *просторовим*, *часовим* і *якісним*. *Просторове* розділення по суті є багатоканальним зв'язком, і при його використанні відпадає необхідність у спеціальних методах кодування. При *якісному* розділенні між символами повинно бути як мінімум дві розподільчі ознаки (наприклад, тривалість імпульсу, паузи), які легко відрізняються на приймальній стороні лінії зв'язку. *Якісне* розділення дає можливість одночасної передачі інформації від різних об'єктів по одному каналу зв'язку. Прикладом якісного розділення є частотне розділення (моногармонічний сигнал при імпульсі має одну частоту, а при паузі – іншу). При часовому розділенні використовуються спеціальні комутатори на передаючій та приймальній сторонах, які почергово з'єднують необхідні лінії зв'язку.

Двійкове кодування десяткових чисел в відповідності з (5.5) не є єдиним. При роботі з двійковими числами широко використовуються й інші коди, які в різних практичних ситуаціях мають свої переваги перед двійковим. Деякі з них для позитивних чисел в інтервалі 0…15 представлені у nабл. 5.1.

Прямий двійковий код A_2 також називають кодом 8-4-2-1 у відповідності з ваговими коефіцієнтами розрядів.

Зворотній код $B_2 = b_3\, b_2\, b_1\, b_0$ отримується шляхом інверсії кожного розряду прямого коду:

$$B_2 = b_3\ b_2\ b_1\ b_0 = \overline{A_2} = \overline{a_3}\ \overline{a_2}\ \overline{a_1}\ \overline{a_0}\,.$$

Він використовується як самостійно в логічних структурах цифрових систем, так і при виконанні арифметичних операцій для одержання доповнюючого коду D_2. Останній застосовується при виконанні арифметичних операцій і знаходиться відповідно до формули:

$$D_2 \quad B_2 \quad 1 \quad \overline{A_2} \quad 1 \quad \overline{a_3}\,\overline{a_2}\,\overline{a_1}\,\overline{a_0} \quad 1,$$

де число 1 додається шляхом двійкової арифметики.

Код Грея, який часто називається *циклічним*, має ту особливість, що при переході з одного числа до сусіднього проходить зміна "**0**" на "**1**" або навпаки тільки в одному розряді. Як видно з таблиці, код, представлений двома, трьома або чотирма розрядами, завжди створює циклічну послідовність, тобто адекватну можливість переходу від самого старшого кодового значення числа до самого молодшого. Ця особливість дозволяє використовувати його при кодуванні кутових переміщень у перетворювачах кута повороту у цифровий код. Код Грея знаходить також широке використання у різних перетворювачах "аналог - код", де його властивість дає можливість звести похибки неоднозначності при зчитуванні інформації до одиниці молодшого розряду.

Табл. 5.1. Значеннння деяких двійкових кодів в інтервалі від 0 до 15

A_{10}	A_2 (двійковий)				B_2 (зворотний)				D_2(доповнюючий)				Код Грея			
	a_3	a_2	a_1	a_0	b_3	b_2	b_1	b_0	d_3	d_2	d_1	d_0	g_3	g_2	g_1	g_0
0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0	1
2	0	0	1	0	1	1	0	1	1	1	1	0	0	0	1	1
3	0	0	1	1	1	1	0	0	1	1	0	1	0	0	1	0
4	0	1	0	0	1	0	1	1	1	1	0	0	0	1	1	0
5	0	1	0	1	1	0	1	0	1	0	1	1	0	1	1	1
6	0	1	1	0	1	0	0	1	1	0	1	0	0	1	0	1

7	0	1	1	1	1	0	0	0	1	0	0	1	0	1	0	0
8	1	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0
9	1	0	0	1	0	1	1	0	0	1	1	1	1	1	0	1
10	1	0	1	0	0	1	0	1	0	1	1	0	1	1	1	1
11	1	0	1	1	0	1	0	0	0	1	0	1	1	1	1	0
12	1	1	0	0	0	0	1	1	0	1	0	0	1	0	1	0
13	1	1	0	1	0	0	1	0	0	0	1	1	1	0	1	1
14	1	1	1	0	0	0	0	1	0	0	1	0	1	0	0	1
15	1	1	1	1	0	0	0	0	0	0	0	1	1	0	0	0

В літературі описані декілька різних прийомів для одержання коду Грея. Один з них дозволяє будувати код Грея безпосередньо з двійкового, використовуючи наступне правило: *i*-й біт коду Грея встановлюється в нуль, якщо *i*-й та (*i* + 1)-й біти відповідного двійкового коду однакові; у протилежному випадку біт *i* = 1. У тому випадку, коли (*i* + 1)-й біт виходить за рамки розрядності двійкового коду, його значення приймається рівним нулю.

При записі слів двійкового коду може використовуватись шістнадцяткове числення

$$A_{16} = \sum_{j=0}^{S} a_j^{16} \cdot 16^j \ .$$

При його використанні десяткові числа від 10 до 15 замінюються відповідно латинськими літерами A, B, C, D, E, F. Двійковий і шістнадцятковий коди легко взаємно переводяться. Для цього двійкове слово будь-якої довжини розбивається на тетради, починаючи з молодшого розряду і, відповідно до табл. 5.1 та вищезазначеними еквівалентами літер та цифр, записується його шістнадцяткове представлення. Наприклад, $A_2 = 111011_2 = 0011\ 1011_2 = 3B_{16}$.

Для позначення цього коду використовують букву h (*hexadecimal*), яку ставлять після його числового значення: $3B_{16} = 3Bh$.

5.3.2 Коди з послідовною формою представлення інформації

Паралельний формат зберігання і передачі даних використовується при малих відстанях між цифровими пристроями і при зберіганні у напівпровідникових запам'ятовуючих пристроях. У той же час, при записі або зчитуванні на магнітні або оптичні носії цифрова інформація повинна передаватись у послідовному форматі. Послідовний формат використовується і при передачі на великі відстані по телефонних і кабельних лініях зв'язку. У кожному з таких випадків представлена в паралельному форматі інформація повинна перетворюватись у послідовний за допомогою спеціальних апаратно-програмних засобів.

Базовою концепцією передачі інформації у послідовному форматі є строга узгодженість побітної передачі з сигналами синхронізації, тобто кожному періоду синхросигналу повинен ставитись у відповідність один біт інформації, що передається. Інтервал часу між двома тактами синхросигналу є *бітовим інтервалом*, протягом якого по інформаційному каналу передається "**0**" або "**1**". У такому випадку, незалежно від довжини слова, яке передається, кількість інформаційних провідників не перевищуватиме трьох – загальний, інформаційний, синхронізації. В інтервалі тактового періоду генератора синхросигналів на інформаційний провідник від джерела сигналу повинен передаватись один біт у вигляді високого або низького рівня напруги. Частота передачі інформаційних сигналів однозначно визначається частотою синхросигналу, розділеною на довжину (у бітах) слів, що передаються. Здебільшого початок інформаційного сигналу (його фронт) співпадає з початком синхроімпульсу, але така особливість передачі не є обов'язковою.

На жаль, описана форма передачі інформації нереальна, оскільки на приймальній стороні неможливо з потоку інформаційних біт виділити окремі слова, принаймні при неузгодженості роботи приймача і передавача. Тому в ряді випадків до раніше визначених провідників додається четвертий, призначений для визначення початку слів. Така форма обміну інформації використову-

ється в комп'ютерній техніці, а також у системах телекомунікацій.

В інших інформаційних системах – наприклад, телефонних і кабельних мережах – використовується лише двопроводова лінія передачі. У такому випадку всі сигнали спеціально поєднуються, створюючи достатньо складний код послідовного формату, який скоріше можна розглядати як аналоговий сигнал з складними видами модуляції. Використовується і спосіб, при якому в потоці інформаційних біт має місце встановлений порядок, який відомий і строго витримується як на стороні передавача, так і на стороні приймача. Такий порядок називається *протоколом обміну*.

При передачі інформації у послідовному форматі використовується декілька способів побудови послідовних кодів. Один з них полягає у потенціальному представленні логічних рівнів "**1**" та "**0**" або протягом всього тактового інтервалу, або на половині його. В обох випадках цифровий сигнал представляється у вигляді однополярних імпульсів. Перший з них у літературі називається *NRZ – non-return to zero* (рис. 5.1).

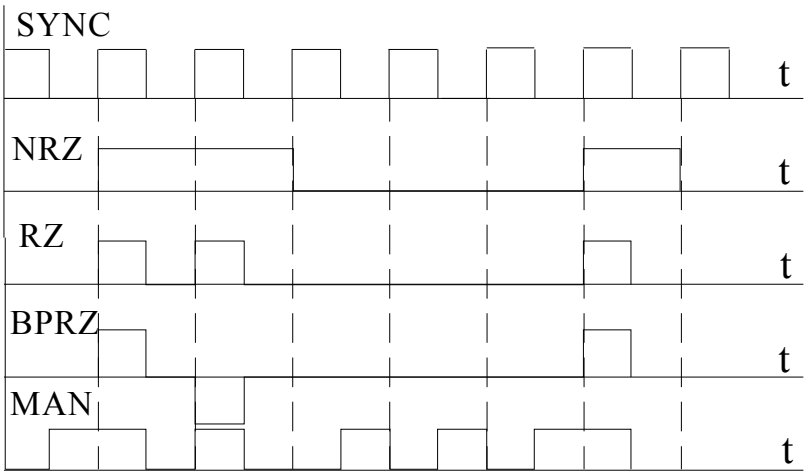


Рис. 5.1. Форми представлення цифрових сигналів

Інший спосіб формування цифрової послідовності полягає в тому, що логічні сигнали "**1**" і "**0**" можуть представлятись у вигляді різнополярних імпульсів, дія яких триває на всьому періоді тактового сигналу, або на його частині.

Прикладом однополярного коду є код *RZ* (*return to zero*). Після встановлення рівня **лог. "1"** у момент появи синхросигналу інформаційний сигнал діє на половині періоду, після чого встановлюється в нуль. Код з інверсією одиниці представляється на рис. 5.1 біполярним кодом *BPRZ*. Його особливість полягає у відсутності постійної складової в інформаційному сигналі, що підвищує завадостійкість кіл, які отримують цей сигнал.

Широко використовуються коди, в яких логічні сигнали кодуються не як потенціальні рівні, а як фронти переходу з "**0**" в "**1**" і з "**1**" в "**0**". Прикладом таких кодів є Манчестерський код (*MAN*). Головна перевага такого коду полягає в тому, що незалежно від символів, що передаються, він забезпечує як мінімум одну передачу в бітовому інтервалі. **Лог. "0"** у такому коді передається як перехід з "**0**" в "**1**" посередині бітового інтервалу, а "**1**" – як перехід з "**1**" в "**0**". Оскільки Манчестерський код має більше переходів **0 – 1 – 0**, ніж інші коди, він вимагає більшої смуги пропускання лінії зв'язку. Такі коди широко використовуються в міжкомп'ютерних системах зв'язку.

Кожен із способів має свої недоліки і переваги, які враховуються при прийнятті рішення про їх використання.

5.3 Виконання арифметичних операцій у двійковій системі

Основною операцією, яка використовується в цифрових системах при виконанні різних обчислень, є операція *алгебраїчного додавання*. Вона виконується на основі правил виконання операцій у двійковій системі зображення чисел, які для однорозрядних чисел мають такий вигляд:

$$\begin{array}{cccc} 0 & 1 & 0 & 1 \\ +\ 0 & +\ 0 & +\ 1 & +\ 1 \\ \hline 0 & 1 & 1 & 10 \end{array}$$

Перенесення до старшого розряду виконується тоді, коли в одному розряді обох складових є одиниці. Операція знаходження суми в багаторозрядних числах виконується послідовно, починаючи з молодшого розряду. У зв'язку з цим, починаючи з другого розряду, виконується складання трьох цифр – двох розрядних складових і перенесення з молодшого розряду.

Приклад 5.5 Скласти два додатних двійкових числа $A_2 = 1001_2$; $B_2 = 1101_2$.

Розв'язання. При виконанні операції додавання мають місце переповнення у першому і четвертому розрядах і, відповідно, перенесення одиниці з першого розряду у другий і з четвертого у п'ятий.

$$\begin{array}{rcccccccc} & & \overset{1}{\downarrow} & & & \overset{1}{\downarrow} & & & \\ A_2 & = & & 1 & 0 & 0 & 1 & = & 9_{10} \\ + \quad B_2 & = & & 1 & 1 & 0 & 1 & = & 13_{10} \\ \hline (A+B)_2 & = & 1 & 0 & 1 & 1 & 0 & = & 22_{10} \end{array}$$

Операція віднімання в цифрових схемах виконується за допомогою операції додавання, зображуючи від'ємне число у доповнюючому коді.

Приклад 5.6 Знайти суму двох чисел $N = 854_{10}$ і $K = -387_{10}$ з використанням доповнюючого коду.

Розв'язання. При виконанні вказаної операції в десятковій системі числення необхідно для числа K знайти відповідний доповнюючий код. Він знаходиться за тими ж правилами, що і в двійковій системі. Зворотній код числа знаходиться як доповнення до дев'ятки цифри кожного розряду. Для числа $K = 387_{10}$ зворотній код $B = 612_{10}$.

Доповнюючий код для числа K буде $D = B + 1 = 612 + 1 = 613$.

Виконаємо операцію додавання. При цьому введемо знакові розряди, які позначимо апострофом, що встановлюється після знакової цифри:

$$N + D = 0'\,854 + 1'\,613 = 10'\,467 = 467.$$

Перенесення, що з'являється зі знакового розряду, відкидається.

Аналогічно виконується операція віднімання в двійковій системі числення.

Приклад 5.7 Додати два числа $N = 0'\,11011_2 = 27_{10}$ і $K = 1'\,01101_2 = -13_{10}$.

Розв'язання. Знаходимо додатковий код від'ємного числа K:

$D = 1'\,10011$.

Знаходимо суму:

$$\begin{array}{rcccccc} & \mathrm{N} & = & 0' & 1 & 1 & 0 & 1 & 1 \\ + & & & & & & & & \\ & \mathrm{D} & = & 1' & 1 & 0 & 0 & 1 & 1 \\ \hline (N+D) & = & & 10' & 0 & 1 & 1 & 1 & 0 \end{array}$$

Відкидаючи 1 переносу в знаковому розряді, отримуємо

$N + D = 0'\,01110_2 = 14_{10}$.

5.4 Основи булевої алгебри

5.4.1 Основні визначення

У практиці інженерної діяльності часто мають місце ситуації, при яких має значення не рівень сигналів, що поступають з відповідних датчиків, а лише наявність чи відсутність таких сигналів. Наприклад, у системах охоронної сигналізації необхідно знати, замкнені чи не замкнені двері або вікна в приміщенні, що охороняється. У системах автоматики часто необхідно знати, чи не перевершує кількість рідини в цистерні заданий рівень, чи не є тиск у котлі нижчим визначеної межі, чи не перевершує температура в приміщенні задану величину і т. п.

Схеми, що дають можливість розв’язувати поставлені задачі, можуть описуватись виразами типу: "лампочка на пульті охоронної сигналізації горить, якщо всі вікна замкнені (точніше, замкнено перше **і** друге **і** третє **і**… вікно)". Або "лампочка не горить, якщо хоча б одне вікно відкрите (тобто може бути відкритим перше **або** друге **або** третє **або** перше **і** друге **або**...)". Такі вирази називаються *логічними*.

При проектуванні подібних систем задаються відповідним рівнем напруги живлення, і наявність чи відсутність її дає можливість одержувати відповіді на поставлені питання. Оскільки рівень напруги може бути різним і задаватись прийнятою елементною базою, то з метою формалізації опису подібних схем приймаються деякі умови. Як приклад, високий рівень напруги приймається за "**1**", низький – відповідно, за "**0**". У такому разі наведені вище вирази можуть бути формалізовані: якщо контакти, що фіксують положення вікон, позначити як аргументи x_1, x_2, …, x_n, які можуть приймати лише значення "**1**" або "**0**", то напругу на лампочці можемо розглядати як функцію y, яка теж приймає одне з двох аналогічних значень.

Математичний апарат, що оперує з аргументами та функціями, які набувають тільки двох значень – "**0**" та "**1**" – називається *двійковою* (*булевою*) *алгеброю* або *алгеброю логіки*. Такий математичний апарат для розв’язання задач формальної логіки розробив ірландський математик Дж. Буль.

Логічні змінні (аргументи), як і змінні звичайної алгебри, позначаються літерами латинського алфавіту з різними індексами – наприклад, x_0, x_1, x_2, x_3, … Індекс при змінній може одночасно означати розряд двійкового числа.

Якщо змінна x_i набуває значення $x_i = 1$, то таке її значення називають *істинним*. Протилежне $x_i = 0$ називають *хибним* і умовно позначають $\overline{x_i}$, що означає заперечення істинного значення аргументу (в зарубіжній практиці операція заперечення позначається апострофом x'). Два елементи булевої алгебри – подія істинна і подія хибна – називають її *константами*.

Булева функція позначається літерою *y* і є двійковою функцією двійкових аргументів. Умовне її позначення $y = f(x_1, x_2, ..., x_n)$.

Булева функція, яка залежить від *n* аргументів, називається *n*-вимірною і є повністю визначеною, якщо вказані значення її для всіх двійкових наборів значень її аргументів. Кількість таких наборів дорівнює 2^n. *Тобто, областю визначеності функції* ***n*** *змінних є сукупність дискретних точок* ***n****-вимірного простору, причому кожна з точок є комбінацією значень цих змінних* (*кодовою комбінацією*). Оскільки можливі 2^n різних комбінацій логічних змінних, то область визначення функції складається зі скінченої величини – 2^n точок. Це, в свою чергу, означає, що кожна функція може бути задана таблицею значень, які вона приймає в точках її області визначеності.

Функція повністю визначена, якщо задані її значення в усіх точках області визначеності. Значення функції вибираються з множини "**0**" і "**1**". Якщо ж значення функції не задано в одній або кількох точках, то вона є *неповністю визначеною*. Кодові комбінації, при яких функція невизначена, називаються *факультативними*. У практиці цифрової схемотехніки існує велика кількість неповністю визначених функцій. Довизначення їх, якщо це необхідно, забезпечується встановленням їх значень – "**0**" або "**1**" –довільним шляхом.

Усі можливі логічні функції *n* змінних можна створити за допомогою трьох основних операцій:

а) логічне заперечення (інверсія, операція **НІ**); позначається рискою над відповідною функцією або аргументом;

б) логічне додавання (диз'юнкція, операція **АБО**), яке позначається символами (V), (+);

в) логічне множення (кон'юнкція, операція **І**), яке позначається символами (∧), (·), (&). Для позначення еквівалентності логічних виразів використовується знак (=).

Запереченням (*інверсією*) називається такий зв'язок між аргументом *x* та функцією *y*, при якому *y* істинна тоді і тільки тоді, коли значення *x* хибне, і навпаки.

Логічним множенням (*кон'юнкцією*) декількох змінних називається така функція, яка істинна тоді і тільки тоді, коли одночасно істинні всі логічні змінні.

Логічним додаванням (*диз'юнкцією*) декількох змінних називається така функція, яка хибна тоді і тільки тоді, коли одночасно хибні всі додавані змінні.

Слід пам'ятати, що операція кон'юнкції є старшою операцією і виконується раніше диз'юнкції.

Прикладом найпростіших функцій є наступні:

$y_1 = \overline{x_1}$; $y_2 = x_1 \cdot x_2$; $y_3 = x_1 + \overline{x_2}$.

У табл. 5.2 наведені приклади деяких логічних функцій двох змінних.

Табл. 5.2. Логічні функції двох змінних

Технічна реалізація булевих функцій, а, відповідно, і їх фізична інтерпретація добре ілюструється за допомогою контактних схем, в яких логічна змінна x_i відповідає замкненому контакту.

5.4.2 Закони і тотожності алгебри логіки

Назва функції	Логічний вираз	X_1X_2			
		00	01	10	11
Інверсія (НІ)	$y = \overline{x}_1$	1	1	0	0
Диз'юнкція (АБО)	$y = x_1 + x_2 = x_1 \vee x_2$	0	1	1	1
Кон'юнкція (І)	$y = x_1 \cdot x_2 = x_1 \wedge x_2$	0	0	0	1
Еквівалентність	$y = x_1 = x_2 = \overline{(x_1 + x_2)(\overline{x}_1 + \overline{x}_2)}$	1	0	0	0
Функція Пірса (АБО-НІ)	$y = \overline{x_1 + x_2} = \overline{x_1 \vee x_2} = x_1 \cdot x_2$	1	0	0	0
Функція Шеффера (І-НІ)	$y = \overline{x_1 x_2} = \overline{x_1 \wedge x_2} = \overline{x}_1 + \overline{x}_2$	1	1	1	0
Сума по модулю 2 (виключне АБО)	$y = x_1 \oplus x_2 = (\overline{x}_1 + \overline{x}_2)(x_1 x_2)$	0	1	1	0

В алгебрі логіки використовується ряд аксіом (тотожностей) та законів. Основними з них є наступні: переміщувальний (властивість комутативності); сполучний (властивість асоціативності); розподільний (властивість дистрибутивності); інверсії (теорема де Моргана). Головні аксіоми та закони булевої алгебри наведені у табл. 5.3.

Табл. 5.3. Головні аксіоми та закони булевої алгебри

Назва аксіоми чи закону	Вирази
Аксіоми (тотожності)	$0 \cdot x = 0;\ x \cdot x = x;\ x \cdot 1 = x;\ x\ \bar{x}\ 0$ $1 + x = 1;\ 0 + x = x;\ x + x = x;\ x\ \ \bar{x}\ 1$
Закони комутативності	$x_1 + x_2 = x_2 + x_1$ $x_1 \cdot x_2 = x_2 \cdot x_1$
Закони асоціативності	$x_1 + x_2 + x_3 = x_1 + (x_2 + x_3) = (x_1 + x_2) + x_3 =$ $= (x_1 + x_3) + x_2$ $x_1 \cdot x_2 \cdot x_3 = x_1 \cdot (x_2 \cdot x_3) =$ $= x_2 \cdot (x_1 \cdot x_3) = x_3 \cdot (x_1 \cdot x_2)$
Закони дистрибутивності	$x_1 \cdot (x_2 + x_3) = x_1 \cdot x_2 + x_1 \cdot x_3$ $x_1 + x_2 \cdot x_3 = (x_1 + x_2) \cdot (x_1 + x_3)$
Закони інверсії (теорема де Моргана, принцип подвійності)	$\overline{x_1\ x_2}\ \ \overline{x_1}\ \ \overline{x_2}$ $\overline{x_1 + x_2} = \overline{x_1} \cdot \overline{x_2}$
Закони поглинання	$x_1 + x_1 \cdot x_2 = x_1$ $x_1 \cdot (x_1 + x_2) = x_1$

Використовуючи наведені у табл. 5.3 закони та тотожності, які використовуються при перетворенні логічних функцій, можна створювати нові. Наприклад:

$$x_1 \cdot (\overline{x_1} + x_2) = x_1 \cdot x_2;$$

$$x_1 \cdot \overline{x_2} + x_2 = x_1 \cdot \overline{x_2} + x_2(x_1 + \overline{x_1}) = x_1 \cdot \overline{x_2} + x_1 \cdot x_2 + \overline{x_1} \cdot x_2 = x_1 + x_2.$$

У подальшому крапки, що відображають операцію логічного множення у формулах, для спрощення запису приводити не будемо.

Закони інверсії, які відображають властивість взаємного перетворення операцій логічного множення і додавання в алгебрі логіки, називають *принципом подвійності*.

5.4.3 Способи задання логічних функцій

Існують такі способи задання або запису логічних функцій – *аналітичний*, *табличний*, *за допомогою карт Карно*, *графічний* та *кубічний*.

Аналітично логічна функція може бути записана різними комбінаціями кон'юнкцій та диз'юнкцій логічних змінних. Зазвичай логічні функції записуються або у вигляді суми добутків логічних змінних (диз'юнкція кон'юнкцій) або у вигляді логічного добутку їх сум (кон'юнкція диз'юнкцій). Наведення функції у вигляді диз'юнкції кон'юнкцій називають *диз'юнктивною нормальною формою* (**ДНФ**):

$$y = x_1\overline{x_2} + \overline{x_1}x_3 + x_1x_2\overline{x_3} \ ,$$

а запис у вигляді кон'юнкції диз'юнкцій – відповідно, *кон'юнктивною нормальною формою* (**КНФ**):

$$y = (x_1 + x_2)(x_2 + \overline{x_3})(\overline{x_1} + x_2 + x_3) \, .$$

Інверсія у відповідності з теоремою де Моргана будь-якої функції, наведеній в одній формі, призводить до заміни запису на іншу форму.

Наприклад, інверсія функції $y = x_1 + x_2\overline{x_3} + x_1\overline{x_2}x_3$ представляється у вигляді $\overline{y} = \overline{x_1}(\overline{x_2} + x_3)(\overline{x_1} + x_2 + \overline{x_3})$.

Будь-яка логічна функція, задана в аналітичній формі, може бути перетворена на **ДНФ** або **КНФ** за допомогою тотожностей та законів алгебри логі-

ки. При цьому для однієї і тієї ж функції може існувати декілька рівнозначних диз'юнктивних та кон'юнктивних нормальних форм.

У той же час, існує лише один вид **ДНФ** та **КНФ**, в яких функція може бути записана єдиним чином. Такі форми називаються *досконалими диз'юнктивними* (*кон'юнктивними*) *нормальними формами* (**ДДНФ**, **ДКНФ**). Вони характеризуються тим, що в **ДДНФ** кожна кон'юнкція, а в **ДКНФ** кожна диз'юнкція містять усі логічні змінні даної функції, з інверсіями або без них.

Прикладами **ДДНФ** та **ДКНФ** запису є функції чотирьох змінних

$$y_1 = x_3\overline{x_2}x_1\overline{x_0} + \overline{x_3}x_2x_1\overline{x_0} + x_3\overline{x_2}x_1x_0 \ ;$$
$$1010 \qquad 0110 \qquad 1011$$
$$y_2 = (x_1 + \overline{x_2} + \overline{x_3} + x_4)(\overline{x_1} + x_2 + \overline{x_3} + x_4)\,.$$

Оскільки кожна кон'юнкція функції, що наведена у **ДДНФ**, визначає її істинне значення, відповідаюче "**1**", то такі кон'юнкції називаються *конституєнтами одиниці* (*мінтермами*). Аналогічно, диз'юнкції функції, що наведені у **ДКНФ**, називаються *конституєнтами нуля* (*макстермами*).

Якщо замінити логічні змінні та їх заперечення одиницями та нулями, то кожна кон'юнкція буде представляти собою двійкове число.

Це дозволяє, наприклад, вище наведену логічну функцію y_1 записати у вигляді:

$$y_1 = \bigvee_{0}^{15} 6, 10, 11\,.$$

Така форма називається досконалою скороченою диз'юнктивною формою або канонічною сумою.

Аналогічно, функцію можна зобразити і у вигляді добутку макстермів. Така форма запису називається *канонічним добутком*. Наприклад:

$$y = \bigwedge_{0}^{7} 2, 4 = (x_2 + \overline{x_1} + x_0)(\overline{x_2} + x_1 + x_0)\,.$$

Легко бачити можливість конвертації в представленні функції у вигляді макстермів та мінтермів, оскільки кожна з них доповнює функцію до повного перебору логічних змінних. Як приклади, можемо записати:

$$y = \bigvee_{0}^{7} 2, 6, 7 = \bigwedge_{0}^{7} 0, 1, 3, 4, 5 ;$$
$$y = \bigvee_{0}^{7} 0, 1, 3, 5 = \bigwedge_{0}^{7} 2, 4, 6, 7 ;$$
$$y = \bigvee_{0}^{15} 0, 4, 5, 9, 11, 13, 15 = \bigwedge_{0}^{15} 1, 2, 3, 6, 7, 8, 10, 12, 14 .$$

Індекси біля умовних позначень операцій диз'юнкції та кон'юнкції вказують на діапазон можливих мінтермів та макстермів логічних функцій. Нижній індекс іноді не вказується.

Досконала диз'юнктивна нормальна форма запису дозволяє легко перейти до інших форм запису – *табличної* та *карт Карно*. У табл. 5.4 наведені функції $y_1 \ldots y_5$ двох змінних x_0 та x_1.

Табл. 5.4. Табличний спосіб представлення логічних функцій

x_1	x_0	y_1	y_2	y_3	y_4	y_5
0	0	0	0	0	0	1
0	1	1	1	1	0	0
1	0	1	1	1	0	0
1	1	x	1	0	1	1

Табличний спосіб полягає у тому, що функція задається у вигляді таблиці відповідності (таблиці істинності станів). У таблицю вписують усі можливі комбінації аргументів у порядку зростання їх індексів і при кожній комбінації встановлюється значення функції. Кількість всіх можливих сполук аргументів, а, отже, і кількість значень функції дорівнює 2^n, де n – кількість логічних змінних. З табличної форми запису легко перейти до аналітичної, використовуючи

досконалу диз'юнктивну форму запису логічних функцій. Для цього функція записується як диз'юнкція конституєнт одиниці. Наприклад, функцію y_3 з табл.1.5 можемо записати у вигляді:

$$y_3 = \overline{x_1}\ x_0 + x_1\ \overline{x_0}\ .$$

Ця функція може бути записана і з використанням нульових її значень:

$$\overline{y_3} = \overline{x_1}\ \overline{x_0} + x_1\ x_0 .$$

Використовуючи властивість подвійної інверсії, легко встановити тотожність обох форм запису.

У практичній схемотехніці найбільш поширеними є системи, які реалізують логічні функції **І-НІ**, **АБО-НІ**, **ВИКЛ. АБО**. Вони дозволяють найбільш просто реалізовувати різні функції, мати більшу кількість входів, прості в технічній реалізації.

Карта Карно (Вейча) – це компактна форма представлення таблиці істинності логічної функції. Такі карти містять у кожній клітинці окремий мінтерм і тому ще мають назву карт мінтермів. Карти мінтермів Карно та Вейча представляють собою прямокутні таблиці із клітинками, число яких дорівнює 2^n, де n – кількість змінних. Кожна одиниця, поміщена в клітку карти Карно, відповідає своєму мінтерму. Щоб нанести на карту вираз ab треба поставити 1 у всіх клітках що містять a і b одночасно. Якщо в клітці з'являється дві або більш 1, вважається, що там 1 – одиниця.

На рис. 5.2 наведено приклад позначення змінних на картах Вейча для чотирьох змінних.

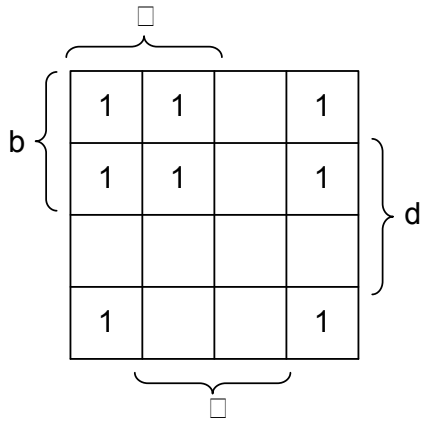


Рис. 5.2. Позначення змінних на карті Вейча

Змінна ***a*** приймає прямі значення у двох лівих стовпчиках таблиці, а у двох правих стовпчиках – приймає інверсне значення (рис. 5.2). Для зменшення кількості написів біля таблиці інверсні значення на рисунках не позначають.

5.5 Спрощення булевих функцій

5.5.1 Доцільність спрощення

б наочно пересвідчитись про необхідність спрощення розглянемо логічний вираз

$$y \quad \overline{x}_1 x_2 x_3 \overline{x}_4 \quad x_1 x_2 x_3 \overline{x}_4 \quad x_1 x_2 \overline{x}_3 \overline{x}_4 \quad \overline{x}_1 x_2 \overline{x}_3 \overline{x}_4$$
$$x_1 x_2 \overline{x}_4 (x_3 \quad \overline{x}_3) \quad \overline{x}_1 x_2 \overline{x}_4 (x_3 \quad \overline{x}_3)$$
$$x_1 x_2 \overline{x}_4 \quad \overline{x}_1 x_2 \overline{x}_4 \quad x_2 \overline{x}_4 (x_1 \quad \overline{x}_1) \quad x_2 \overline{x}_4 .$$

За допомогою винесення змінних за дужки і з урахуванням того, що $x \quad \overline{x} \quad 1$, отримаємо спрощений вираз $y \quad x_2 \overline{x}_4$.

Такий метод перетворення називається аналітичним, або методом Квайна.

Перетворена функція набагато простіше вихідної, а це означає, що для її реалізації буде потрібно набагато менше логічних елементів, ніж для реалізації початкової функції. При побудові апаратури прагнуть до реалізації структурних схем, що забезпечують мінімальну витрату компонентів та устаткування. Це і обумовлює необхідність мінімізації логічних функцій.

5.5.2 Задачі мінімізації

Під мінімізацією булевої функції найчастіше розуміють знаходження найбільш простого виразу у вигляді суперпозиції операцій, що представляють функціонально повну систему.

Найбільш простим вважається вираз, що містить мінімальне число суперпозицій.

Метою мінімізації є зменшення вартості технічної реалізації логічних функцій незалежно від використовуваних апаратних засобів.

Логічні функції апаратно реалізуються за допомогою мікросхем, орієнтованих на виконання тих чи інших операцій. Мікросхеми загального використання здебільшого можуть реалізовувати декілька простих одиночних операцій. З цієї причини справедливо стверджувати, що чим простішою є аналітична форма запису логічної функції, тим менше використовується логічних елементів і, як результат, тим менше мікросхем необхідно для її реалізації. Складність логічних функцій визначається кількістю логічних змінних, що входять до їх складу в прямому і інверсному виді, та кількістю простих логічних операцій над ними. Будь-яка логічна функція може бути записана різними аналітичними виразами різного рівня складності. Серед них можна знайти такі, які містять мінімальну кількість логічних змінних і операцій над ними. Задача знаходження таких аналітичних виразів називається *мінімізацією логічних функцій*. Звідси витікає, що *мінімізація логічної функції* – це заміна логічної функції, що представлена у вигляді логічної суми мінтермів або логічного добутку макстермів, ін-

шою логічною функцією з мінімальною кількістю логічних змінних та операцій над ними.

Задача мінімізації – це задача неоднозначна, і різними шляхами можна отримати різні вирази мінімізованої функції, які відрізнятимуться між собою кількістю змінних і операцій над ними.

5.5.3 Спрощення логічних функцій за допомогою карт мінтермів

Карта Карно відрізняється від карти Вейча порядком розташування змінних на карті.

Карти Карно виявляються більш зручними при аналізі й синтезі послідовних схем (наприклад, лічильників або регістрів). На бокових гранях карти вказують змінні, а ті області де змінні приймають інверсні значення не позначають (вважається що це і так зрозуміло). Наприклад, у верхньому лівому куту знаходиться мінтерм у якому змінна *c* приймає інверсне значення, а на гранях карти не позначено $\overline{c}$.

На рис. 5.3 та 5.4 наведено приклади позначення змінних на картах Вейча та Карно (відповідно) та розташування мінтермів у цих картах.

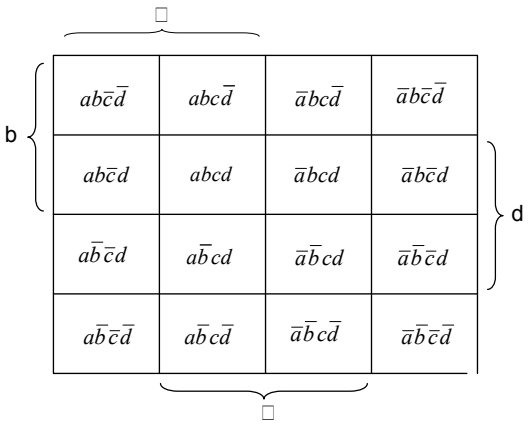


Рис. 5.3. Розміщення мінтермів на карті Вейча

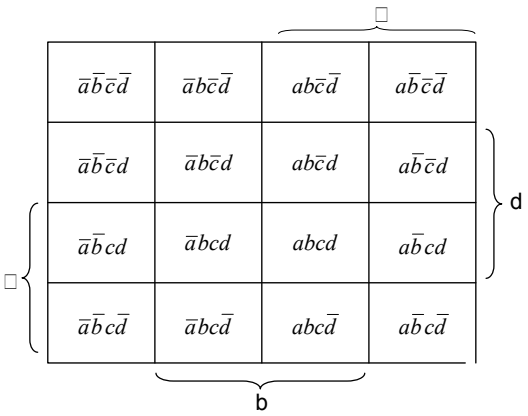


Рис. 5.4. Розміщення мінтермів на карті Карно

Для нанесення логічного виразу на карту мінтермів достатньо розташувати "1" у клітинках карти, що відповідають всім умовам цього виразу. Наприклад, вираз $y \quad ab \quad \bar{c}\bar{d} \quad \bar{a}b\bar{c}$ відповідає карті мінтермів на рис. 5.5.

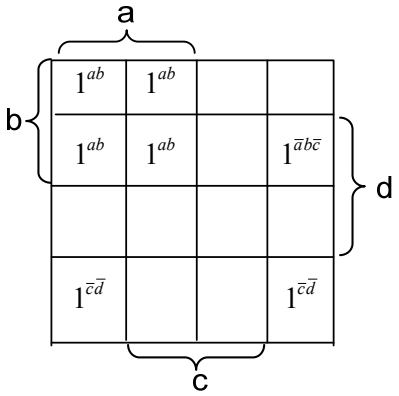


Рис. 5.5. Нанесення на карту Вейча виразів

Якщо декілька "1" треба розміщувати в одній клітинці карти, то там ставиться тільки однв одиниця.

5.5.4 Способи об'єднання мінтермів на картах Вейча або Карно

Спрощення досягається склеюванням одиниць на карті мінтермів. Склеювати можна квадрати, подвійні квадрати, повні стовпці, а також два сусідніх мінтерма і мінтерми, що перебувають у протилежних кінцях стовпчика або рядка. Приклади склеювання наведені нижче.

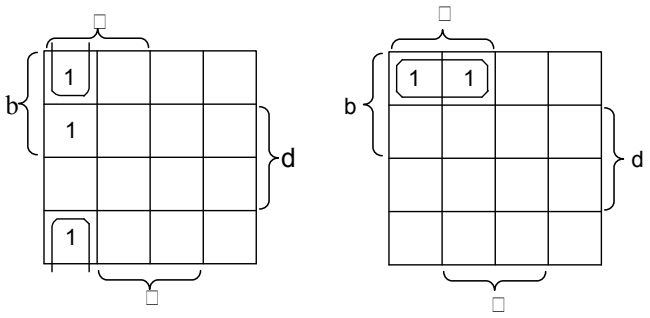
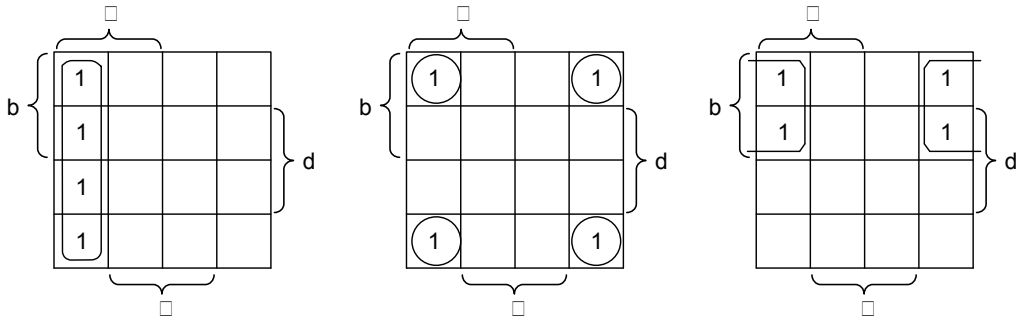


Рис. 5.6. Склевання двох мінтермів



а) *б)* *в)*

Рис. 5.7. Склеювання чотирьох мінтермів

При мінімізації за допомогою карт мінтермів ***мінімальна форма буде отримана у тому випадку коли мінімальним числом контурів охопити максимальне число одиниць***. Не треба забувати про те, що одна і та сама одиниця може попадати в декілька контурів одночасно.

Для зчитування спрощеного виразу з карти мінтермів слід притримуватись такого правила. У спрощеному виразі для кожного склеювання будуть відсутні ті змінні, що змінюють свій знак у цьому склеюванні. Наприклад, для склеювання з чотирьох одиниць (рис. 5.7,в) змінні *b* та *d* змінюють свій знак. Тому спрощений вираз для цього склеювання буде виглядати як $y \quad a\overline{c}$.

5.5.5 Позначення логічних елементів

Для реалізації логічних функцій використовуються логічні елементи, умовні позначення яких у відповідності зі стандартом наведено на рис. 5.8.

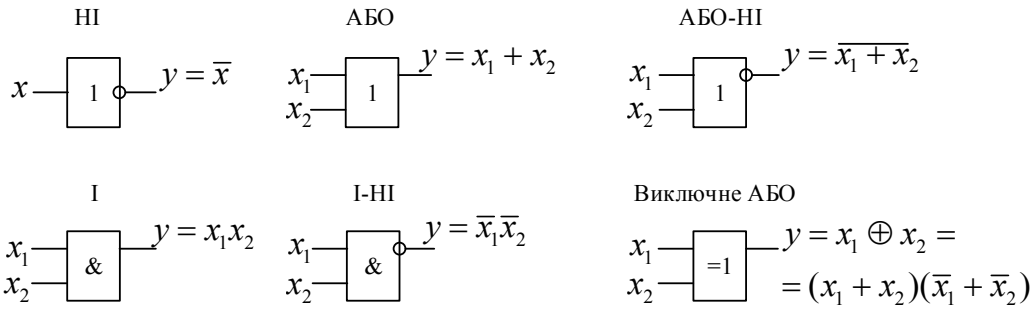


Рис. 5.8. Умовні позначення логічних елементів

На рис. 5.8 наведені умовні позначення, характерні для стандартів, прийнятих у країнах СНД, німецькому стандарті DIN та міжнародному стандарті IEC 60617. Використання наведених умовних позначень дає можливість будувати складні логічні та принципові схеми електронних пристроїв. Логічні елементи, умовні зображення яких наведені на рис. 5.8, виготовляються в різних серіях цифрових мікросхем.

Приклад 5.8 Побудувати схему, що відповідає функції $y \quad x_1 x_2 \quad \overline{x_1 x_2}$.

Розв'язання. Використовуючи лише однофункціональні логічні елементи, будується схема, що наведена на рис. 5.9.

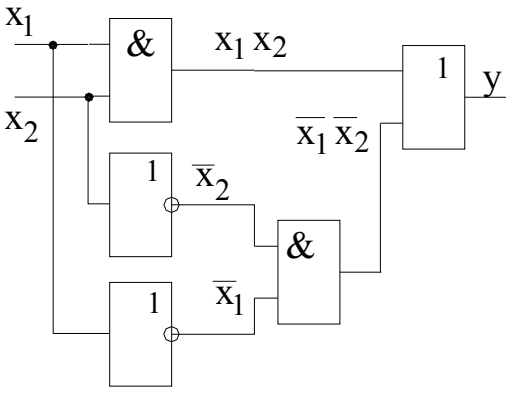


Рис. 5.9. Приклад реалізації логічного виразу за допомогою логічних елементів

Для кращого розуміння процессів, що відбуваються у цифрових схемах, використовують *часові діаграми*, які відображають часові співвідношення між вхідними логічними змінними і вихідною функцією (рис. 5.10). Часові діаграми зображають з урахуванням фронтів імпульсних послідовностей з відображенням моментів початку перехідних процесів і часто мають допоміжні вказівні стрілки, які уточнюють хід перехідних процесів у схемі.

На рис. 5.10 наведено приклад часових діаграм для логічної схеми, зображеної на рис. 5.9, що реалізує операцію цифрового компаратора. У випадку, що розглядається, приводяться лише вхідні та вихідні сигнали. При визначенні інтервалів часових затримок, тривалості перехідних процесів приводиться більша деталізація часових діаграм.

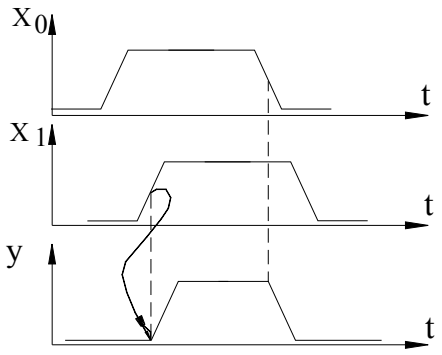


Рис. 5.10. Приклад часових діаграм сигналів у цифрових схемах

Функціонально повна система логічних елементів – набір елементів, який дозволяє реалізувати логічне вираз будь-якого ступеня складності.

Таких наборів існує три:

- НІ, І, АБО;
- І-НІ;
- АБО-НІ.

Елементи І, АБО, І-НІ, АБО-НІ, що реалізовані у вигляді логічних мікросхем, можуть мати 2, 3, 4 або 8 входів.

Розглянемо приклад створення схеми за логічним виразом $y = x_1x_2x_3 + \overline{x_3x_2x_4} + x_1\overline{x_3}$.

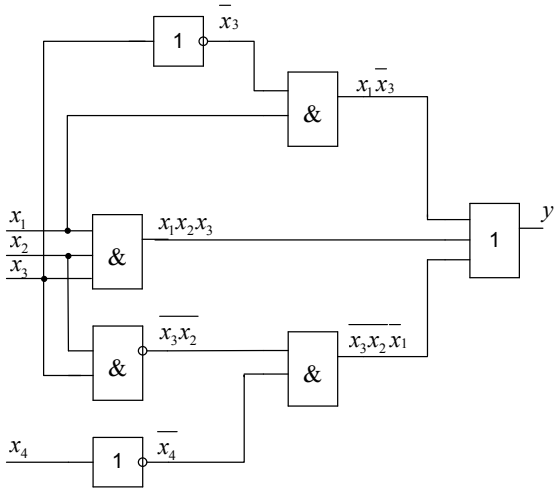


Рис. 5.11. Схема з набору логічних елементів НІ, І, АБО

Реалізуємо схему за виразом $y = \overline{x_4}\,\overline{(x_3 + x_4) + \overline{x_1x_2}(x_3 + x_4)}$ з набору І-НІ.

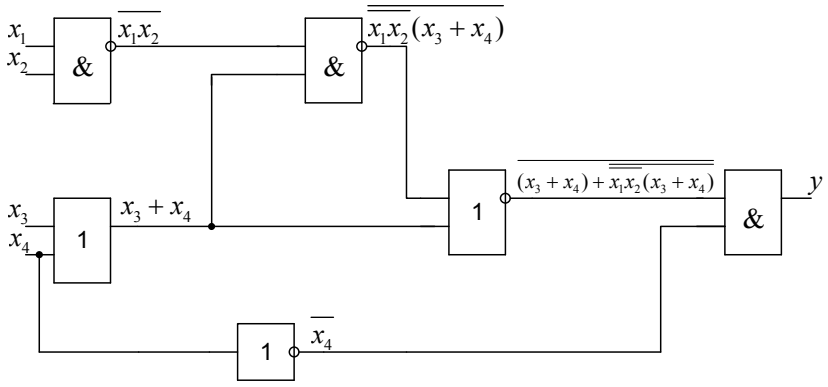


Рис. 5.12. Схема з набору логічних елементів І-НІ

Спробуємо мінімізувати розглянутий вище вираз

$$y = \overline{x_4}\ \overline{(x_3 + x_4) + \overline{x_1x_2}(x_3 + x_4)} = \overline{x_4} \cdot \overline{x_3 + x_4}\,\overline{(x_1x_2}(x_3 + x_4)) = 0.$$

Результатом є нульове значення на виході, тобто замість схеми, що наведена на рис. 5.12 треба просто з'єднати вихід з загальним проводом.

Розглянемо ще один приклад мінімізації. За виразом $y = \overline{x_1x_2x_3x_4 + x_3\overline{x_4}} + \overline{\overline{x_3x_4} + \overline{x_1x_3}}$ побудуємо схему на логічних елементах АБО-НІ, а потім зробимо мінімізацію і побудуємо схему за мінімізованим виразом.

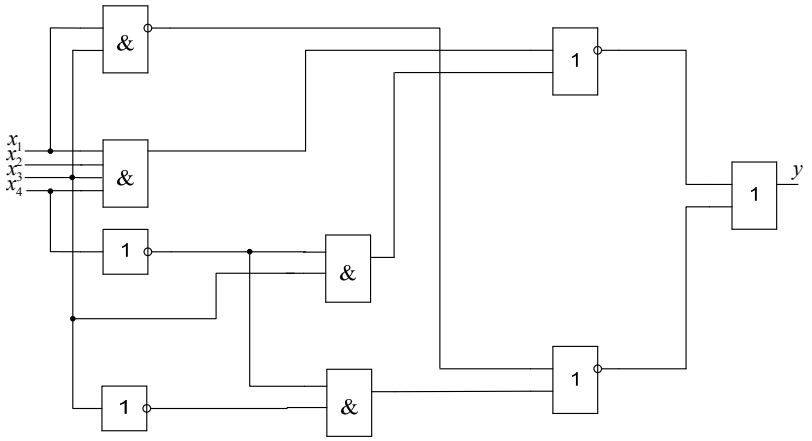


Рис. 5.13. Схема з набору логічних елементів АБО-НІ

Зробимо спрощення логічного виразу. Для цього позначимо складові як *a* та *b* і проведем спрощення спочатку для окремих складових, а потім для всього виразу.

$$y \quad \overline{x_1x_2x_3x_4 \quad x_3\overline{x_4}} \quad \overline{\overline{x_3x_4} \quad \overline{x_1x_3}};$$

$$y \quad a \quad b;$$

$$a \quad \overline{x_1x_2x_3x_4 \quad x_3\overline{x_4}} \quad \overline{x_1x_2x_3x_4} \quad \overline{x_3\overline{x_4}} \quad (\overline{x_1} \quad \overline{x_2} \quad \overline{x_3} \quad \overline{x_4})(\overline{x_3} \quad x_4)$$

$$\overline{x_1x_3} \quad \overline{x_2x_3} \quad \overline{x_3x_3} \quad \overline{x_4x_3} \quad \overline{x_1}x_4 \quad \overline{x_2}x_4 \quad \overline{x_3}x_4 \quad \overline{x_4}x_4;$$

$$b \quad \overline{\overline{x_3x_4} \quad \overline{x_1x_3}} \quad \overline{\overline{x_3x_4}} \quad x_1x_3 \quad (x_3 \quad x_4)x_1x_3 \quad x_1x_3x_3 \quad x_1x_3x_4$$

$$x_1x_3 \quad x_1x_3x_4;$$

$$y \quad x_4 \quad x_1x_3 \quad \overline{x_1x_3} \quad x_4 \quad x_1x_3 \quad \overline{x_1 \quad x_3}.$$

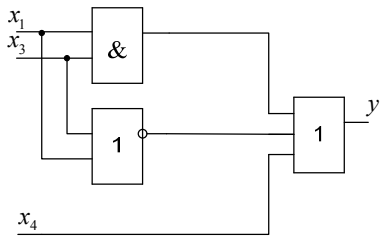


Рис. 5.14. Мінімізований варіант схеми, що показана на рис. 5.13

5.6 Контрольні питання

1. Дайте визначення системи числення. Наведіть приклади позиційних і непозиційних систем числення. Дайте пояснення.

2. Що Ви розумієте під терміном *"алфавіт системи числення"*? Наведіть приклади алфавітів двійкової системи числення; шістнадцяткової системи числення; системи числення з основою 10.

3. Дайте пояснення особливостям переведення чисел з десяткової системи числення у *двійкову*.

4. Дайте пояснення особливостям переведення дробових чисел з однієї системи числення в іншу. Наведіть конкретні приклади.

5. Поясніть взаємозв’язок між прямим двійковим, зворотнім і доповнюючим кодами.

6. Поясніть переваги та недоліки послідовного і паралельного форматів передачі даних.

7. Які способи послідовної передачі даних Вам відомі? Поясніть їх особливості.

8. Поясніть послідовність виконання арифметичних операцій додавання і віднімання в двійковій системі числення при різних знаках зменшуваного і від’ємника.

9. Операція віднімання у двійковому коді виконується з використанням доповнюючого коду. Проаналізуйте і дайте пояснення, чи справедлива подібна послідовність виконання арифметичної операції, якщо числа задаються у двійково-десятковому коді; у коді "з надлишком 3".

10. Дайте визначення терміну "алгебра логіки" ("булева алгебра").

11. Дайте пояснення диз’юнктивній і кон’юнктивній формам запису логічних функцій, а також досконалим формам запису.

12. Перелічіть відомі Вам способи запису логічних функцій. Дайте пояснення взаємозв'язку між ними.

13. Дайте пояснення суті теореми де Моргана. Приведіть приклади її використання.

14. Поясніть властивості карт мінтермів.

15. Які логічні операції використовуються для аналітичного способу мінімізації логічних функцій?

16. На яких властивостях карт Карно реалізується задача мінімізації логічних функцій?

17. Поясніть суть мінімізації логічних функцій методом Квайна.

6 СТРУКТУРИ БАЗОВИХ ЛОГІЧНИХ ЕЛЕМЕНТІВ

6.1 Характеристики цифрових сигналів

Цифрові сигнали **лог.** "**0**" і **лог.** "**1**", які використовуються в курсі дискретної математики, виступають ідеалізацією тих сигналів, що мають місце в реальних електронних схемах.

У ключових схемах, що використовуються при двійковому представленні інформації, значенням **лог.** "**0**" та **лог.** "**1**" присвоюються обмежені діапазони напруг, які розміщуються в інтервалі від нуля до величини діючої напруги живлення логічних схем.

Не розглядаючи у даному параграфі технічні характеристики апаратних засобів для реалізації операцій над цифровими сигналами, визначимось лише з основними параметрами реальних сигналів.

Які б технічні засоби не використовувались, їх параметри загалом можливо оцінити за допомогою характеристики амплітудної характеристики, що являє собою залежність $U_{вих} = f(U_{вх})$ (рис. 6.1), де, відповідно, $U_{вх}$ та $U_{вих}$ – напру-