МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ВОЛОДИМИРА ДАЛЯ

КОНСПЕКТ ЛЕКЦІЙ

дисципліни **«Комп'ютерна логіка та цифрові автомати»**
(для здобувачів вищої освіти денної та заочної форми навчання
галузь знань: F – Інформаційні технології
спеціальності: F3 – Комп'ютерні науки та
F7 – Комп'ютерна інженерія)

Затверджено
на засіданні кафедри
Інформаційних технологій та
програмування
Протокол № 8 від 02.06.2026 р.

Київ 2026

УДК 681.32 (075.8)

Рецензент: О. І. Рязанцев, д.т.н., професор

Конспект лекцій дисципліни “Комп’ютерна логіка та цифрові автомати“ (для здобувачів вищої освіти денної та заочної форми навчання галузь знань: F – Інформаційні технології; спеціальності: F3 – Комп’ютерні науки та F7 – Комп’ютерна інженерія) / Укладач О.К. Лифар. – Київ: Вид-во СНУ ім. В. Даля, 2026. – 209 с.

Конспект лекцій підготовлений відповідно до робочої програми дисципліни “Комп’ютерна логіка та цифрові автомати“. Спрямований на вивчання і засвоєння студентами теоретичних та практичних основ виконання арифметичних та логічних операцій, побудови комбінаційних схем, принципів роботи цифрових автоматів.

ЗМІСТ

ЧАСТИНА 1

1. ВСТУП В СПЕЦІАЛЬНІСТЬ

1.1 Історія розвитку обчислювальної техніки

З'явившись більше 80 років тому, ЕОМ відкрили нову сторінку в історії людських знань і можливостей, вивільнили тисячі обчислень, полегшили працю вчених, дали можливість вивчати складні процеси. Зараз важко назвати галузь народного господарства, де неможливо було б застосувати ЕОМ.

Поява ЕОМ було підготовлено історичним розвитком засобів обчислень. Найдавнішим рахунковим інструментом була людська рука (звідси 5-ричная і 10-кова системи числення). Надалі "обчислювальні засоби" удосконалювалися: з'являлися дерев'яні палички з карбами, мотузки з вузликами, камінчики і т. д. і незабаром з'явився найдавніший прилад - "абак", в якому по жолобках пересувалися камінці. Також в 16-17 столітті з'явилися рахівниці. А в 17 ст. з'явилися перші логарифмічні лінійки.

З розвитком суспільства обчислення ставали все більш трудомісткими.

У 1623 р. з'явився «Рахуючий годинник» Вільгельма Шикарда - перший механічний калькулятор, що виконував чотири арифметичні дії.

У 1623 році 1642 р.: Блез Паскаль створив «Паскаліну» - механічний пристрій для додавання та віднімання, який не знайшов практичного застосування, але зайняв гідне місце в історичному розвитку обчислювальних пристроїв і став перехідним етапом від простих пристроїв до механічних.

Корінний перелом у створенні лічильних пристроїв стався в середині 19 століття, коли з'явилася необхідна технологічна база, було винайдено клавішний пристрій введення. Паралельно створювалися і арифмометри.

Перша діюча лічильно-аналітична машина була створена Холлеритом в 1880 р. в США для автоматизації роботи з обробки даних перепису населення.

Кінець 19 початок 20 століття характеризуються бурхливим розвитком електротехніки, радіотехніки, телефонії, що не могло не відбитися на розвитку обчислювальної техніки.

У 1947 р. була закінчена робота над першою релейною обчислювальною машиною "Марк-2", в якій вперше використовувалася двійкова система числення, а для запам'ятовування чисел, виконання арифметичних операцій і операцій

управління використовувалися електромеханічні реле (13 тис.), що володіють двома стійкими станами.

У 1943 р. в Гарвардському університеті приступили до створення електронної обчислювальної машини. До того часу вже були відомі діод (1904р.), тріод (1905р.), тригер (1918р.). Машина створювалася за замовленням артилерійського управління, була закінчена в 45 році і отримала назву ЕНІАК. Створення обчислювальної машини ЕНІАК послужило початком бурхливого розвитку ЕОМ нового покоління.

У СРСР перша мала електронна лічильна машина (МЕОМ) була створена в 1951 р. під керівництвом С.А. Лебедєва. Для неї характерна наявність універсального рахункового пристрою, оперативної пам'яті. Вона була однією з перших ЕОМ з паралельною обробкою кодів. У 1953 р. була створена БЕСМ. У цьому ж році - перша ЦВМ - "Стріла". У 1954 р. - ЕОМ "Урал", М-3, Мінськ-2 і т.д. Це були ЕОМ **першого покоління**.

Характерними рисами ЕОМ першого покоління можна вважати не тільки використання електронних ламп в основних і допоміжних схемах, а й наявність паралельного арифметичного пристрою, поділ пам'яті на швидкодіючу оперативну і повільну зовнішню, застосування напівпровідникових діодів і магнітних сердечників, перфострічок, перфокарт та ін.

ЕОМ другого покоління - у них на зміну ламповим схемам прийшли транзисторні. Основу технічної бази становили напівпровідникові діоди і транзистори. ЕОМ другого покоління відрізняються більшою надійністю, швидкодією, меншим споживанням енергії. Прикладом ЕОМ 2-го покоління є БЕСМ-6. Для неї характерна паралельна робота окремих блоків.

Поява малих інтегральних схем (МІС) стало базою для створення машин **3-го покоління.**

Основні характерні риси ЕОМ 3-го покоління наступні.

1. Оперують довільною літерно-цифровою інформацією, тому можуть застосовуватися для ділового, комерційного та наукового напрямів.

2. Змінився порядок роботи ЕОМ 3-го покоління: вони побудовані за принципом незалежної паралельної роботи окремих пристроїв: процесорів, зовнішньої пам'яті. Завдяки цьому ЕОМ може виконувати серію операцій: пересилати інформацію для чергової задачі з магнітної стрічки або магнітного диска, виводити інформацію для відповідного пристрою, вводити інформацію та ін.

Типові представники ЕОМ 3-го покоління - машини єдиної системи (ЄС ЕОМ).

ЕОМ 4-го покоління створюються на великих інтегральних схемах (ВІС). У результаті досягнуто суттєве підвищення продуктивності, зріс обсяг пам'яті.

Продуктивність традиційних обчислювальних систем підвищувалася двома шляхами: розвитком елементної бази та розвитком архітектури самих систем. Якщо по першому напрямку майже досягнута межа, то по другому є ще великі резерви, які відкриваються у зв'язку з використанням методів паралельної обробки інформації.

Системи 5-го покоління в структурному аспекті відрізняються саме застосуванням таких паралельних структур. Другою відмітною рисою є здатність виконувати не тільки числові обчислення, а й обробку смислової інформації з виконанням операції аналізу та виводу. Третя відмінна риса - елементна база: НВІС, оптоелектроніка та ін.

Цифрові електронні обчислювальні машини (ЕОМ), або комп'ютери — це системи з програмним керуванням, призначені для автоматизації оброблення цифрової інформації. Організація ЕОМ базується на певних принципах, які складають методологічну основу цифрової обчислювальної техніки (ЦОТ). ЕОМ належить до класу складних систем, аналіз і синтез яких базується на ієрархічному підході до питань їх організації. Існують різні рівні опису ЕОМ, серед яких найпоширенішими є віртуальний, функціональний, логічний та фізичний. Кожний з них потребує певної деталізації компонентів ЕОМ. У навчальному посібнику застосовуються функціональний та логічний рівні опису, які базуються на теорії цифрових автоматів та є найефективнішими для розгляду принципів організації та функціонування вузлів та пристроїв ЕОМ, пов'язаних з обробленням дискретної (цифрової) інформації. При цьому не розглядаються фізичні процеси, що становлять предмет цифрової електроніки.

В.М.Глушков створив першу у світі персональну ЕОМ, названу «машиною для інженерних розрахунків» (МИР). У пристрої були використані всі основні принципи роботи сучасного персонального комп'ютера, тому СВІТ сміливо можна назвати його першим прообразом. «Цю машину Радянський Союз продемонстрував 1967 року на виставці у Лондоні, де її купили представники компанії ІВМ. Глушкову видали міжнародний сертифікат, який підтверджує, що перший у світі персональний комп'ютер створив саме він, а американці отримали нову радянську розробку. І незабаром у них з'явився свій аналог».

Дискретна інформація відображується у вигляді позначень, за допомогою яких із сукупності можливих об'єктів виділяється той, що позначається. Для побудови позначень використовується набір символів - алфавіт, на основі котрого створюють послідовності символів, що визначають різні числові й логічні значення, найменування величин, дії над величинами і таке інше. Так, наприклад, для числових значень у десятковій системі числення використовуються символи 0, 1,2,9, «+», «-», «,».

Для позначення нечислової інформації використовуються букви різних алфавітів та математичні символи. Інформація, подана у вигляді послідовності символів певного алфавіту, називається *символьною інформацією*.

Оскільки набір символів і правила запису можуть обиратися довільно, то інформація може подаватися у вигляді, відмінному від звичного загальноприйнятого. Прикладом цього є різні коди. В ЕОМ найбільше поширення знайшли двійкові коди, що складаються із символів 0 та 1.

У цифрових пристроях інформація зберігається і перетворюється з використанням фізичних елементів. Найширше застосовують фізичні елементи з двома станами, що позначаються символами 0 і 1.

1.2 Структурні схеми ЕОМ і обчислювальних систем

Можна дати таке визначення ЕОМ:

Обчислювальна машина - це фізична система (пристрій або комплекс пристроїв), призначена для механізації або автоматизації процесу алгоритмічної обробки інформації і обчислень.

Вид перероблюваної інформації впливає на структуру обчислювальних машин, які залежно від цього ділять на два основні класи: аналогові і цифрові.

АОМ - машина, що оперує інформацією, поданою у вигляді фізичних величин, які безперервно змінюються, наприклад сили струму.

Багато явищ в природі математично описуються одними і тими ж рівняннями. Отже, за допомогою одного фізичного процесу можна моделювати різні процеси, що мають один і той же математичний опис.

ЦОМ - машина, що оперує інформацією, представленою в дискретному вигляді.

В даний час в математиці розроблені методи чисельного рішення багатьох видів рівнянь. Отже, з'явилася можливість розв’язувати різні рівняння і задачі за допомогою набору простих арифметичних і логічних операцій. Тому будь-яка ЦОМ є універсальним обчислювальним пристроєм.

Існують ВМ, в яких поєднані позитивні якості ЦОМ (точність і універсальність) і АОМ (оперативність введення інформації та швидкодія у виконанні операцій). Такі машини отримали назву **комбінованих або гібридних**.

За принципом дії основних вузлів АОМ і ЦОМ поділяють на механічні, змішані, (гідромеханічні, електромеханічні), електронні та оптоелектронні.

ЕОМ можна розділити на два види: універсальні і проблемно-орієнтовані (спеціалізовані).

Універсальна ЕОМ - машина, що володіє широкими можливостями у вирішенні завдань для різних галузей науки і техніки. Універсальні ЕОМ характеризуються швидкодією.

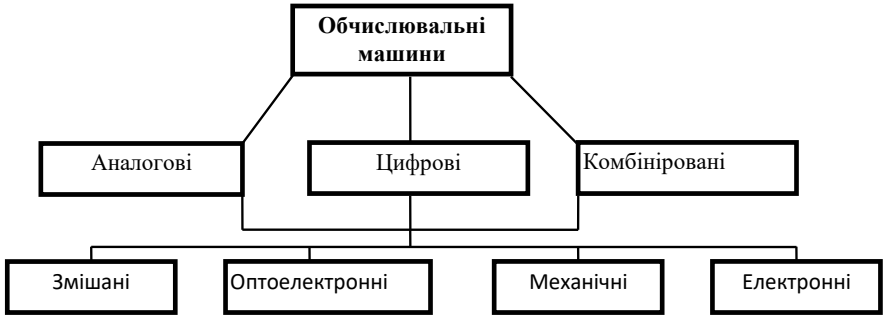


Рис. 1.1-Основні класи обчислювальних машин

Серед проблемно-орієнтованих машин доцільно виділити обчислювальні машини настільного типу, персональні та керуючі ЕОМ. Вони відрізняються від універсальних ЕОМ тим, що шляхом звуження можливостей ЕОМ у відношенні класів розв'язуваних завдань вдається істотно спростити структуру самої машини.

Далі в основному будуть розглядатися універсальні ЕОМ.

Для автоматичного вирішення завдань універсальна ЕОМ повинна включати в себе наступні пристрої:

1. *Пристрій введення інформації*, основне призначення якого - перетворення вхідної інформації, представленої в символах вхідного алфавіту, в інформацію, записану символами внутрішнього алфавіту.

2. *Пристрої підготовки даних* - для перетворення числової, текстової, графічної та іншої інформації в інформацію, записану в символах вхідного алфавіту.

3. *Пам'ять ЕОМ* - функціональна частина ЕОМ, призначена для запам'ятовування і (або) видачі вхідної інформації, проміжних і остаточних результатів, допоміжної інформації. У пам'яті машини перебувають також програми вирішення завдань, через команди якої здійснюється управління роботою всієї машини. Вся інформація в пам'яті машини представляється символами внутрішнього алфавіту. Основні параметри, що характеризують пам'ять - ємність і час звернення до пам'яті.

4. Арифметико - логічний пристрій (АЛП) - функціональна частина ЕОМ, що виконує логічні і арифметичні дії, необхідні для переробки інформації, що

зберігається в пам'яті. Він характеризується: часом виконання елементарних операцій, середньою швидкодією, набором елементарних дій, які він виконує, видом алфавіту або системою числення , в якій виконуються всі дії.

5. Пристрій управління (ПУ) - функціональна частина ЕОМ, призначена для автоматичного керування ходом обчислювального процесу, що забезпечує взаємодію всіх частин машини у відповідності з програмою вирішення завдання. ПУ звертається в пам'ять машини, вибирає чергову команду, розшифровує її і виробляє сигнали, що вказують іншим пристроям , що їм належить робити .

6. Вивідні пристрої - пристрої, які здійснюють перетворення результатів вирішення задачі, представлених в символах внутрішнього алфавіту, у вихідну інформацію, представлену символами вихідного алфавіту і видачі інформації з машини. Залежно від виду вихідної інформації розрізняють пристрої вихідні друкуючі, графічні, ті що відображають та ін.

Розглянутий склад структурної схеми ЕОМ можна назвати класичним.

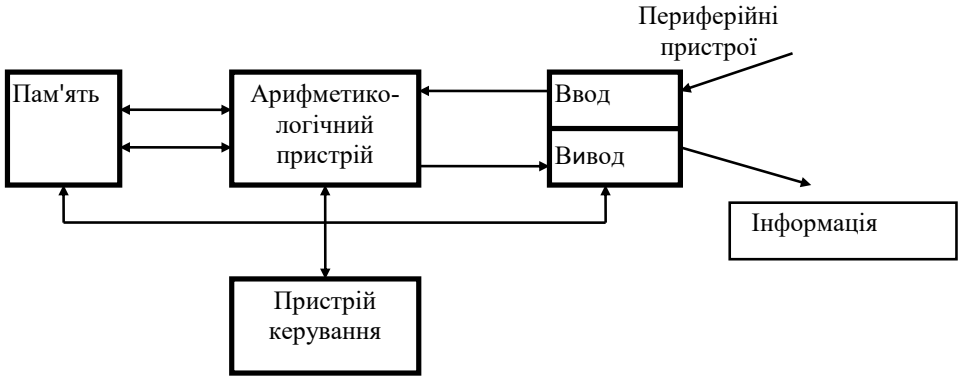


Рис. 1.2 - Склад пристрою ЕОМ

Така схема характерна для ЕОМ 3-го покоління. В даний час ЕОМ будуються на основі мікропроцесорів.

Комплекс пристроїв, що охоплює АЛУ, частина оперативної пам'яті і пристрій управління, називається процесором. Процесор-самостійна функціональна програмно керована частина ЕОМ, безпосередньо здійснює процес перетворення інформації та управління нею.

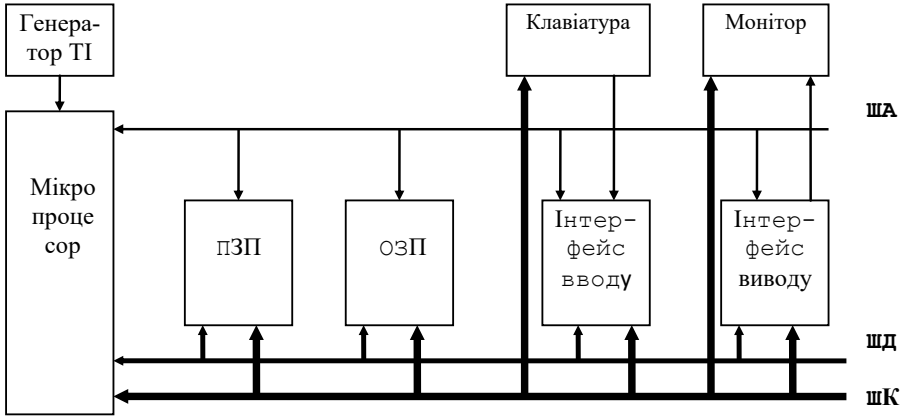


Рис. 1.3- Структурна схема мікроЕОМ

Обчислювальні машини, побудовані на основі мікропроцесорів, називаються мікроЕОМ і відрізняються тим, що мають два види пам'яті: оперативну та постійну.

Структурна схема мікроЕОМ представлена на рис. 1.3.

1.3 Поняття про системи ЕОМ

Розширення сфери застосування обчислювальної техніки призвело до включення до складу машини великого комплексу периферійних пристроїв для введення інформації, запам'ятовування, зберігання, реєстрації, відображення. Конкретні застосування пред'являють різні вимоги до складу периферійних пристроїв. Це призвело до того, що замість універсальної ЕОМ з фіксованим складом устаткування змінила агрегатована ОМ - система зі змінним складом обладнання. При такому підході окремі пристрої виконуються у вигляді модулів, які в потрібній конфігурації об'єднуються в ЕОМ.

У розвитку даної концепції основна увага приділялася створенню рядів або систем ЕОМ, що складаються з інформаційно і програмно сумісних машин, що володіють різними характеристиками.

Інформаційна сумісність передбачає єдині способи кодування інформації та формати даних або, принаймні, однакові або кратні довжини слів.

Програмна сумісність передбачає, що програми, складені для однієї моделі, можуть виконуватися на інших моделях. Це досягається єдиною системою команд.

Крім цього ще має бути апаратурна сумісність, яка полягає в можливості підключення до будь-якої моделі.

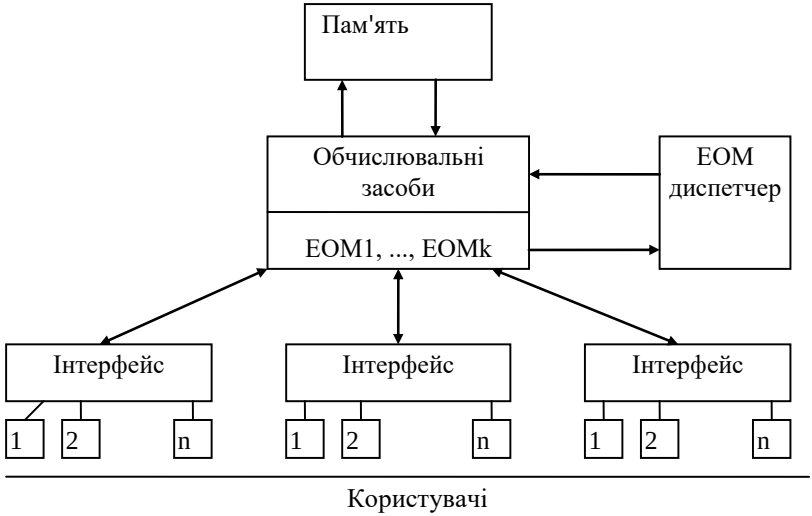


Рис. 1.4-Структурна схема системи ЕОМ

Під обчислювальною системою розуміється сукупність засобів обчислювальної техніки, що включає в себе не менше двох процесорів або обчислювальних машин, одна з яких виконує роль основного процесора. Структурна схема обчислювальної системи представлена на рис.1.4.

1.4 Обчислювальні мережі

Розвиток архітектури, апаратних і програмних засобів і успіхи в розвитку методів передачі даних по каналах зв'язку дозволили створити обчислювальні системи нового типу - обчислювальні мережі.

Обчислювальної мережею або мережею ЕОМ називається багатомашинна система, що складається з розподілених по великій території ЕОМ, пов'язаних між собою каналами передачі даних.

Обчислювальну мережу можна розглядати як систему з розподіленими по території апаратними, програмними та інформаційними ресурсами.

2 ПРЕДСТАВЛЕННЯ ЧИСЕЛЬНОЇ ІНФОРМАЦІЇ В ЕОМ. СИСТЕМИ ЧИСЛЕННЯ

2.1 Поняття системи числення. Класифікація систем числення. Позиційні і непозиційної системи числення

Системи числення були створені в процесі господарської діяльності людини, коли у неї з'явилася потреба в підрахунку, а в міру розвитку наукової і технічної діяльності виникла також необхідність записувати числа і робити над ними обчислення.

Системою числення називається сукупність символів і прийомів, що дозволяють однозначно зображати числа. Або, в загальному випадку, це спеціальна мова, алфавітом якої є символи, звані цифрами, а синтаксисом - правила, що дозволяють однозначно сформувати запис чисел. Запис числа в деякій системі числення називається кодом числа. У загальному випадку число записується таким чином:

$$A=a_n\ a_{n-1} \ldots\ a_2\ a_1\ a_0$$

Окрему позицію в записі числа прийнято називати розрядом, а номер позиції - номером розряду, кількість розрядів у записі числа - це розрядність і вона збігається з довжиною числа. У технічному плані довжина числа інтерпретується як довжина розрядної сітки. Якщо алфавіт має *p* різних значень, то розряд a_i в числі розглядається як *p*-а цифра, якій може бути присвоєно кожне з *p* значень.

Кожній цифрі a_i даного числа A однозначно відповідає її кількісний (числовий) еквівалент - К(a_i). При будь якій кінцевій розрядній сітці кількісний (числовий) еквівалент числа A буде приймати в залежності від кількісних окремих розрядів значення від $К(A)_{min}$ до $К(A)_{max}$.

Діапазон представлення (D) чисел в даній системі обчислення - це інтервал числової осі, укладений між максимальними і мінімальними числами, що представлені заданою розрядністю (довжиною розрядної сітки):

$$D=К(A)_{(p)\ max} - К(A)_{(p)\ min}.$$

Існує незліченна множина способів запису чисел цифровими знаками. Однак, будь-яка система числення, призначена для практичного використання, повинна забезпечувати:

1) можливість подання будь-якого числа в заданому діапазоні чисел;

2) однозначність подання;

3) стислість і простоту запису чисел;

4) легкість оволодіння системою, а також простоту та зручність оперувати нею.

Залежно від цілей застосування використовують різні системи числення: 2-у, 10-у, 8-у, 16-у, римську, а для обчислення часу - система числення часу і т.д.

В залежності від засобу запису чисел і засобу обчислення їх кількісного еквівалента системи числення можна класифікувати таким чином (рис. 2.1)

В основному системи обчислення будуються за наступним принципом:

$$A_{(p)} = a_n p_n + a_{n-1} p_{n-1} \ldots\ldots + a_1 p_1,$$

де $A_{(p)}$- запис числа в системі з базисом p_i;

a_i- база або послідовність цифр системи числення з p_i-ним алфавітом;

p_i - базис системи числення (сукупність ваг окремих розрядів системи числення). Базис десяткової системи числення 10^0, 10^1, 10^2, 10^3, ..., 10^n.

База системи числення може бути додатною (0,1,2...9), але може бути й змішаною (1,$\overline{1}$).

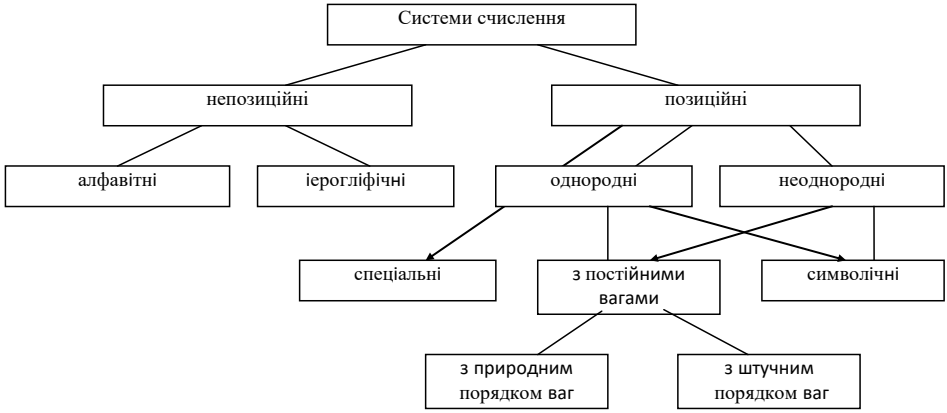


Рис. 2.1- Класифікація систем числення

Основою системи числення називається кількість різноманітних символів (цифр), що використовуються в кожному з розрядів для відображення числа в даній системі числення.

Вага розряду R_i в будь-якій системі числення - це відношення $R_i=p_i/p_0$.

2.1.1 Непозиційні системи числення

Непозиційні системи числення - це системи числення, алфавіт яких містить необмежену кількість символів (цифр), причому кількісний еквівалент будь-якої цифри постійний і залежить тільки від її зображення і не залежить від положення в числі. Такі системи будуються за принципом адитивності, тобто кількісний еквівалент числа визначається як сума цифр у числі. Найбільш відомими представниками непозиційних систем числення є ієрогліфічні та алфавітні, зокрема, ієрогліфічна система - римська система числення. Запис чисел у алфавітних системах числення будується за таким же принципом.

До основних недоліків непозиційних систем числення можна віднести:

1) відсутність нуля;

2) необхідність утримання нескінченної кількості символів;

3) складність арифметичних дій.

Основну увагу приділимо позиційним системам числення.

2.1.2 ***Позиційні системи числення***

Позиційними називаються такі системи числення, алфавіт яких містить обмежену кількість символів, причому значення кожної цифри визначається не тільки її зображенням, а й перебуває в строгій залежності від позиції в числі. Основна перевага позиційних систем числення - зручність виконання обчислень.

Позиційні системи числення поділяються на ряд підкласів.

Неоднорідні позиційні системи числення (зі змішаною основою)

У таких системах числення в кожному розряді кількість допустимих символів може бути різною, значення не залежать один від одного і можуть приймати будь-які значення. Прикладом неоднорідною позиційної системи числення може служити система числення часу, для якої P_0- 1сек,P_1- 60 сек, P_2- 60 хв, P_3- 24 години, P_4- 365 діб.

Однорідні позиційні системи числення.

Це окремий випадок позиційних систем числення, в них ваги окремих розрядів представляють собою ряд членів геометричної прогресії зі знаменником р. Тому число в однорідних системах може бути представлено у загальному випадку поліномом виду:

$$A_{(p)} = a_n p^n + a_{n-1} p^{n-1} + \dots a_1 p^1 + a_0 p^0 \,, + a_{-1} p^{-1} + \dots + a_{-k} p^{-k} \,,$$

або

$$A = \sum_{i=-k}^{n} a_i \cdot p^i$$

Основою однорідної позиційної системи може бути будь-яке ціле число, так як у визначенні позиційних систем числення не накладено ніяких обмежень на величину основи. Тому можлива незліченна множина позиційних систем числення.

Звичайно число в однорідній системі числення записується в скороченому вигляді:

$$A_{(p)} = a_n\, a_{n-1} \dots a_1 a_0 a_{-1} \dots a_{-k},$$

а назву системи числення визначає її основа: десяткова, двійкова, вісімкова, шістнадцяткова і т.д. Для будь-якої позиційної системи числення справедливо, що її основа зображується символами 10 у своїй системі.

Кодовані системи числення

Це такі системи, в яких цифри однієї системи числення кодуються за допомогою цифр іншої системи. Прикладом може служити двійковій-десяткова система з вагами (8-4-2-1) або (8-4-2-1+3).

У позиційних системах числення кожному розряду присвоюють певну вагу - звідси випливає, що число визначається не тільки цифрою, а й місцем цієї цифри в ряду інших. Позиційні системи числення ділять на два підкласи з природним і штучним порядком ваг - розрядів.

У символічних кодах ваг немає.

Якщо для позначення кожного допустимого знака використовують один символ, то говорять, що це система числення з безпосереднім зображенням чисел.

Якщо для позначення кожного допустимого знака використовують набір символів з меншою основою, то систему числення називають з кодованим

зображенням чисел (код). У кодованих системах числення розрізняють великі і малі розряди пов'язані відповідно до основної та допоміжної систем числення.

У цифрових пристроях, в якості допоміжної системи числення використовують 0 і 1 (виконавчі кодовані системи). Найбільшого поширення набули системи з основою $N=2^n$, де n – це двійкова система числення, і система числення з основою 10 - десяткове двійкове кодування (двійково-десятковий код).

У двійково-десятковому коді великі розряди мають основу 10, а малі - 2. Природно розрізняються і ваги - зовнішні та внутрішні.

Якщо коди зважені, то будь-яке число N_i (0 - 9) можна представити у вигляді суми добутків.

$N_i=a*m_4+b*m_3+c*m_2+d*m_1$

I = 0, 1, … 9

A, b, c, d – 0 або 1, причому a – старший розряд

Використовуються 4 двійкових розряди m_4, m_3, m_2, m_1 – це й є вагові коефіцієнти.

У такому коді числа записують тетрадами. Код - це перерахування вагових коефіцієнтів, починаючи зі старшого розряду.

10^1	10^0	- вагові коефіцієнти старшого розряду (зовнішні)
2	7	- десятковий код
0010	0111	- двійково-десятковий код.

Якщо кодова комбінація вихідного десяткового числа отримана відповідно до наведеної формули, кажуть, що число представлено в прямому коді.

Якщо кодова комбінація відповідає числу, що є доповненням вихідного (тобто представленого в прямому коді) до 9-ти (у загальному випадку до основи системи числення -1), то говорять, що вихідне число представлене в оберненому коді, а якщо до 10-ти - то в додатковому коді.

0111 – 7 в прямому вигляді
0010 – 7 в оберненому вигляді (9-7=2)
0011 – 7 в додатковому коді (10-7=3)

2.2 Переведення чисел з однієї системи числення в іншу

Існує два основні методи переведення чисел з однієї системи числення в іншу: табличний і розрахунковий [2].

Табличний метод прямого переводу заснований на зіставленні таблиць відповідності чисел різних систем числення. Цей метод дуже громіздкий і вимагає дуже великого обсягу пам'яті для зберігання таблиць, але застосовується для будь-яких систем числення.

Розрахунковий метод переводу застосовується тільки для позиційних однорідних систем числення.

2.2.1 Переведення цілих чисел з однієї позиційної системи числення в іншу

Нехай задано число А в довільній позиційній системі числення з основою L і його необхідно перевести в нову систему числення з основою Р, тобто перетворити до виду:

$$A_{(p)} = a_n p^n + a_{n-1} p^{n-1} + \dots a_1 p^1 + a_0 p^0 , \qquad (2.1)$$

де $a_i = 0 \div (p-1)$ - база нової системи числення.

Це вираз можна записати у вигляді:

$$A = A_1 p + a_0 ,$$

де $A_1 = (a_n p^{n-1} + a_{n-1} p^{n-2} + \dots a_2 p^1 + a_1)$ - ціла частина частки,

a_0 - залишок від ділення А/р, котрий є цифрою молодшого розряду числа, яке потрібно знайти.

При діленні числа А1 на р. отримаємо залишок а1 і т.д. Іншими словами, якщо записати вираз (2.1) за схемою Горнера:

$$A = \left(\dots\left(\left(a_n p + a_{n-1}\right)p + a_{n-2}\right)p + \dots + a_1\right)p + a_0 ,$$

після чого праву частину необхідно послідовно поділити на основу нової системи обчислення р, то отримаємо коефіцієнти:

$$A = A_1 p + a_0$$

$$A_1 = A_2 p + a_1$$

...

$$A_{n-1} = A_n p + a_{n-1}$$

$$A_0 = 0 \cdot p + a_n$$

При цьому ділення продовжується до тих пір, поки не виявиться, що $A_n < p, A_{n+1} = 0$

Правило переведення цілих чисел з однієї позиційної системи числення в іншу формулюється таким чином:

Щоб перевести ціле число з однієї позиційної системи числення в іншу необхідно вихідне число послідовно ділити на основу нової системи числення, записану у вихідній системі числення, до отримання частки, рівної нулю. Число у новій системі числення записується із залишків від ділення, починаючи з ***останнього***.

Розглянемо як приклад переведення цілого числа 138 у двійкову, вісімкову, шістнадцяткову системи числення.

138, 69, 34, 17, 8, 4, 2, 1, 0 - частка
0 1 0 1 0 0 0 1 - залишок
138, 17, 2, 0- частка
2 1 2
138, 8, 0
10 8

$[138]_{10}=[10001010]_2=[212]_8=[8A]_{16}$

При переведенні з двійкової системи числення в десяткову вихідне число необхідно ділити на основу нової системи, тобто на 1010_2.

Ділення виконувати у двійковій системі важко, тому на практиці зручніше користуватися загальним записом числа у вигляді полінома. При переведенні двійкових чисел в десяткову систему числення зазвичай підраховують суму ваг розрядів основи 2, при яких коефіцієнти a_i рівні 1. Розрахунки при цьому ведуться в десятковій системі.

2.2.2 Переведення правильних дробів.

Нехай правильний дріб А, заданий у довільній позиційній системі числення з основою L необхідно перевести в нову систему з основою Р, тобто перетворити його до вигляду:

$$A= a_{-1}p^{-1} +...+ a_{—k}\, p^{-k}, \qquad (2.2)$$

якщо, аналогічно переведенню цілих чисел розділити обидві частини виразу на p^{-1}, тобто помножити на p, то отримаємо:

$$Ap = a_{-1} + A_1,$$

де $A_1= a_{-2}p^{-1} + a_{-3}p^{-2} +...+ a_{—k}\, p^{-k+1}$ - дробова частина добутку,
a_{-1} - ціла частина результату.

Отримана при цьому цифра цілої частини результату і буде першою цифрою шуканого числа. Помноживши тепер дробову частину результату на основу нової системи числення, отримаємо:

$$A_1p = a_{-2} + A_2,$$

де A_2 - дробова частина добутку ,
a_{-2} - наступна цифра шуканого числа.

Отже, при переводі вираз (2.2) представляється за схемою Горнера:

$$A = p^{-1}(a_{-1} +p^{-1}(a_{-2} + ... + p^{-1}(a_{-k+1} + p^{-1}a_{-k})...)).$$

Для переведення правильного дробу з однієї позиційної системи числення в іншу її треба послідовно перемножувати на основу нової системи числення до тих пір, поки в новому дробі не буде потрібної кількості цифр, яка визначається необхідною точністю подання дробу. Правильний дріб в новій системі числення записується з цілих частин додатків, що виходять при послідовному множенні, причому перша ціла частина буде старшою цифрою нового дробу.

Переведення дробу в загальному випадку являє собою нескінченний процес. Число цифр у новій системі числення необхідно визначати з умови, що точність подання числа в новій системі повинна відповідати точності у вихідній системі.

Розглянемо як приклад переведення правильного дробу 0,536 в двійкову, вісімкову, шістнадцяткову системи числення:

$[0,536]_{10}=[0,10001001]_2=[0,422335]_8=[0,8937]_{16}$

0,	536 2
1,	072 2
0,	144 2
0,	288 2
0,	576 2
1,	152 2
0,	304 2
0,	608

0,	536 8
4,	288 8
2,	304 8
2,	432 8
3,	456 8
3,	648 8
5,	184

0,	536 16
8,	576 16
9,	216 16
3,	456 16
7,	296

2.2.3 Переведення неправильних дробів

При переведенні неправильного дробу необхідно окремо перевести цілу і дробову частини по вищевикладеним правилам і записати число в новій системі числення, залишивши незмінним положення коми.

2.2.4 Переведення чисел з системи обчислення в систему з кратною основою

Якщо основи систем числення кратні одна одній, тобто пов'язані залежністю: $l = p^m$, то кожна цифра системи числення з основою l може бути представлена m цифрами в системі з основою p.

Отже, для того, щоб перевести число з вихідної системи в нову, основа якої кратна основі вихідної системи, досить кожну цифру числа, яке необхідно перевести, записати за допомогою m цифр в новій системі числення, якщо основа вихідної системи більше основи нової системи числення. В іншому випадку кожні m цифр вихідного числа необхідно записати за допомогою однієї цифри в новій системі числення, починаючи для цілих чисел з молодшого розряду і для правильних дробів - зі старшого

Приклад.
$[0,536]_{10}=[0,100'010'010]_2=[0,422]_8$;
$[0,1000'1001'0000]_2=[0,89]_{16}$
$[138]_{10}=[010'001'010,]_2=[212]_8$:
$[1000'1010,]_2=[8A,]_{16}$

2.3 Вибір системи числення для використання в ЭОМ

Очевидно, що непозиційні системи числення непридатні для застосування в ЕОМ через свою громіздкість та труднощі виконання арифметичних операцій .

З позиційних найбільш зручні однорідні. З точки зору застосування в ЕОМ враховуються такі чинники.

1. Наявність фізичних елементів, здатних зобразити символи системи.

2. Економічність системи, тобто кількість елементів необхідну для подання багаторозрядних чисел.

3. Трудомісткість виконання арифметичних операцій в ЕОМ .

4. Швидкодія обчислювальних систем.

5. Наявність формального математичного апарату для аналізу і синтезу обчислювальних пристроїв.

6. Зручність роботи людини з машиною.

7. Завадостійкість кодування цифр на носіях інформації.

Історично склалося так, що для застосування в ЕОМ була обрана двійкова система числення, яка найбільш повно відповідає цим критеріям.

У сучасній обчислювальній техніїі застосовуються як двійкова, так і десяткова системи числення. Причому цифри останньої кодуються двійковими символами, тобто йдеться насправді не про десяткову, а про двійково-десяткову систему числення. Кожна із зазначених систем має свої переваги і недоліки, а також свої області застосування.

Достоїнствами двійкової системи числення щодо двійковій-десяткового є:

1) економія близько 20% обладнання;

2) приблизно в 1,5 рази більш високу швидкодію;

3) спрощення логічної побудови і значна економія обладнання в схемах управління і в допоміжних ланцюгах.

Достоїнствами двійковій-десяткового системи є:

1) відсутність необхідності переведення вихідних даних і результатів розрахунків з однієї системи в іншу;

2) зручність контролю проміжних результатів шляхом виведення їх на індикацію для візуального спостереження;

3) більш широкі можливості для автоматичного контролю через наявність у двійковій-десятковому коді надлишкових комбінацій.

Двійкову систему числення застосовують у великих і середніх ЕОМ, призначених для вирішення науково-технічних завдань, для яких характерний великий обсяг обчислень і порівняно малий обсяг вихідних даних і результатів обчислень. Її також доцільно застосовувати в ЕОМ, призначених для керування технологічними процесами.

Двійковій-десяткову систему обчислення застосовують для вирішення економічних завдань, які характеризуються великим обсягом вихідних даних, порівняльною простотою і малим обсягом виконуваних над ними перетворень і великою кількістю результатів обчислень. Цю систему доцільно також застосовувати в калькуляторах, ЕОМ, призначених для інженерних розрахунків, а також у персональних ЕОМ.

2.4 Двійкова система числення

Під двійковою системою числення розуміють таку систему, в якій для зображення чисел використовуються два символи, а ваги розрядів змінюються за законом $2^{\pm к}$, де к - довільне ціле число. Класичною двійковою системою є система з символами 0, 1. Її двійкові цифри часто називають бітами. У загальному вигляді всі двійкові числа представляються у вигляді:

$$A= \sum a_i 2^i, \text{ (і від -к до n)} \qquad (2.3)$$

$$A_{(p)}= 1x2^n + a_{n-1}p^{n-1} + ... \; 1x2^1 + 1x2^0 \; , \; 1x2^{-1} + 1x2^{-2} + ...1x2^{-k}$$

Щоб оволодіти будь-якою системою числення, треба вміти виконувати в ній арифметичні операції. Арифметичні операції у двійковій системі обчислення виконуються так само, як і в десятковій відповідно з таблицями порозрядних обчислень.

Додавання у двійковій системі числення проводиться за правилами додавання поліномів. Тому при додаванні чисел А і В і-й розряд суми S_i і перенос $П_i$ з даного розряду в (і+1) розряд буде визначатися у відповідності з наступним виразом:

$a_i + b_i + П_{i-1} = S_i + П_{i+1}$

a_i	b_i	$П_{i-1}$	S_i	$П_{i+1}$
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

Таблиця множення двох двійкових чисел повністю визначається двома правилами:

- Множення будь-якого числа на нуль дає в результаті нуль,
- Множення будь-якого числа на 1 залишає його без зміни, тобто результат дорівнює вихідному числу.

0x0=0
0x1=0
1x0=0
1x1=1

2.4.1 Навички поводження з двійковими числами

Хоча всі правила виконання операцій у двійковій системі числення дуже прості, але проте при роботі з двійковими числами через відсутність навичок виникають різного роду незручності. Нижче наведені деякі прості прийоми, які дозволяють досить вільно поводитися з двійковими числами.

Таблиця 2.1.

n	0	1	2	3	4	5	6	7	8	9	10	11	12
2^n	1	2	4	8	16	32	64	128	256	512	1024	2048	4096

1. Число $1\underbrace{00...00}_{\text{n нулів}} = 2^n$.

Необхідно запам'ятати десяткові значення чисел, представлених в таблиці 2.1.

2. Число $\underbrace{111...11}_{\text{n одиниць}} = 2^n - 1$.

3. Необхідно знати напам'ять десяткові значення двійкових чисел від 0 до 31 включно. Ці числа надалі будуть називатися "малими числами".

4. Двійкове число

$$A = \underbrace{a_{n-k+5}\, a_{n-k+4}\, a_{n-k+3}\, a_{n-k+2}\, a_{n-k+1}}_{\text{мале число a}} \underbrace{000...000}_{\text{k нулів}} = a \cdot 2^k$$

Приклад. $11011000 = 11011 \times 2^3 = 27 \times 8 = 216$.

$$A = \underbrace{a_{n-k+5}\, a_{n-k+4}\, a_{n-k+3}\, a_{n-k+2}\, a_{n-k+1}}_{\text{мале число a}} \underbrace{0\ 0\ b_5 b_4 b_3 b_2 b_1}_{\text{мале число b}} = a \times 2^k + b$$

k розрядів

Приклад. $10110000101 = 1011 \times 2^7 + 101 = 11 \times 128 + 5 = 1413$.

5. Якщо в n-розрядному числі багато одиниць і мало нулів, то для визначення його кількісного еквівалента можна з n розрядного числа, записаного одними одиницями, відняти мале число, в якому розряди зі значенням 1 відповідають розрядам вихідного числа з нульовим значенням і навпаки.

Приклад. 11111101001 відповідає

$$\begin{array}{rcll} 11111111111 & = & 21^{11} & -1 \\ 10110 & = & 22 & \\ \hline 11111101001 & & & \end{array}$$

тобто $11111101001 = 2048 - 1 - 10110 = 2047 - 22 = 2025$.

Числа можливо читати різними засобами. Наприклад:

$1110101 = 2^7 - 1 - 1010 = 127 - 10 = 127$

$1110101 = 111 \times 2^4 + 101 = 7 \times 16 + 5 = 127$

6. Читання двійкових дробів

$A = 0,\underbrace{000...001}_{\text{n-1 нулів}} = 2^{-n}$

Дріб $A = 0,\underbrace{111...111}_{\text{k одиниць}} = 1 - 2^{-k}$.

Двійковий дріб читається за тими ж правилами, що і десятковий: розряди праворуч від коми читаються як ціле число, яке є чисельником; знаменник

читається як ціле число, що є 2^k, причому k - номер молодшого розряду праворуч від коми.

2.5 Форми представлення двійкових чисел в ЭОМ

Машинне подання числа - це подання числа в розрядної сітці ЕОМ.

Машинне зображення числа умовно позначають [А]. при цьому $\mathbf{A=[A]k_A}$, де k_A – масштабний коефіцієнт, величина якого залежить від форми подання числа в ЕОМ.

Під формою представлення числа в ЕОМ розуміють звід правил, що дозволяє встановити взаємну відповідність між записом числа і його кількісним еквівалентом.

Якщо довільне дійсне число $A`=[A]k_A$, то таке число представлене в розрядній сітці машини точно. Якщо $A`\neq[A]k_A$, то довільне дійсне число може бути представлене в машині наближено або взагалі не може бути представлено. При наближеному представленні дійсне число А` замінюється деяким числом [А], що належить до множини машинних чисел. Множині машинних чисел належать тільки числа, кратні двом, так як будь-які два попарно сусідніх машинних числа відрізняються один від одного на величину 2^{-n}, де n - кількість розрядів.

$A_{min} < |A| < A_{max}$

Якщо $|A| < A_{min}$, таке число називають машинним нулем. Числа, більші ніж A_{max}, не можуть бути представлені. У цьому випадку говорять про переповнення розрядної сітки.

Існує три форми представлення чисел у ЕОМ: природна, з фіксованою комою і нормальна (з плаваючою комою).

Природною формою запису числа називається запис числа у вигляді полінома, представленого в скороченому вигляді:

$$A= a_n a_{n-1} \dots a_1 a_0, a_{-1} a_{-2} \dots a_{-k} \qquad (2.4)$$

При цьому відлік ваг розрядів ведеться від коми. Кома ставиться на строго визначеному місці - між цілою і дробовою частиною числа. Тому для кожного числа необхідно вказати положення його коми в одному з розрядів коду, тобто в загальному випадку місце положення коми має бути передбачено в кожному розряді. Зазвичай таку форму подання використовують у калькуляторах.

Якщо місце коми в розрядній сітці машини заздалегідь фіксовано, то таке представлення називається представленням з фіксованою комою (крапкою).

У більшості ЕОМ з фіксованою комою числа, з якими оперує машина, менше одиниці і представлені у вигляді правильних дробів, тобто кому фіксують перед старшим розрядом числа, причому числа, більше одиниці, наводяться до такого виду за допомогою масштабного коефіцієнта $К_А$. Представлення чисел у вигляді правильних дробів обумовлено необхідністю зменшити можливість переповнення розрядної сітки машини, тобто зменшити небезпеку втрати значущих цифр старших розрядів при виконанні арифметичних операцій.

Результат множення ніколи не виходить за межі розрядної сітки, якщо кома розташована перед старшим розрядом. Але в цьому випадку результати додавання і ділення можуть вийти за межі розрядної сітки при операції додавання - не більше ніж на один розряд).

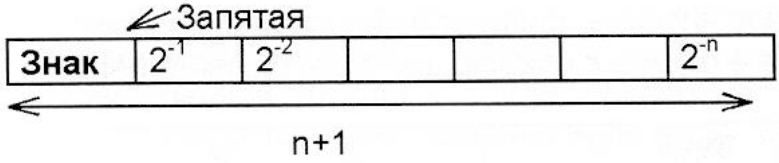
Можна було б оперувати лише малими числами, тому що ймовірність переповнення при їх додаванні мала. Однак це призводить до зниження точності подання чисел та точності обчислень. Тому завжди прагнуть використовувати числа, величини яких близькі до максимального значення. Однак при цьому на них накладаються наступні обмеження:

1) абсолютна величина суми двох чисел повинна бути менше одиниці,

2) дільник за абсолютною величиною повинен бути більше діленого .

У комірці машини з **фіксованою перед старшим розрядом** коми число записується у розрядну сітку у вигляді значущої частини дробу зі своїм знаком, тобто для запису n-значного дробу розрядна сітка повинна містити n + 1 розряд .

Розрядна сітка або формат числа в двійковій системі числення має вигляд:



Тут n розрядів використовують для зображення цифрової частини числа і 1 - для знаку. Величини чисел, що представляються в машинах з фіксованою перед старшим розрядом коми, лежать в межах:

$$\mathbf{2^{-n} \le |A| \le 1-2^{-n}}$$

У цьому випадку: $|A|_{min}$=0,...01 = 2^{-n} , а $|A|_{max}$= 0,1...1= $1-2^{-n}$. (Кома розділяє цілу і дробову частини).

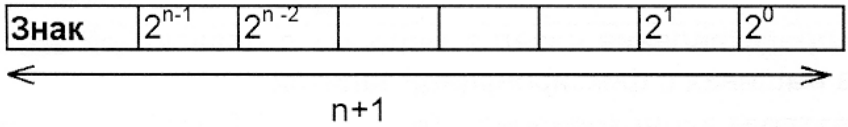
Починаючи з обчислювальних машин 2-го покоління, формати чисел в ЕОМ представляються кратними байту, тобто n = 8, або 16, 32, 64.

У всіх розглянутих форматах можуть зображуються числа, які за своєю абсолютною величиною менше 1, що спрощує конструкцію, зменшує об'єм обладнання.

Недоліком такого представлення чисел є необхідність виконання трудомісткого розрахунку масштабних коефіцієнтів в процесі підготовки завдання для вирішення у ЕОМ.

Нерідко кому **фіксують після молодшого розряду числа**. Тоді всі дані представляються у вигляді цілих чисел. У цьому випадку також необхідно масштабування вихідних даних.

Ваги розрядів у форматі числа, що містить n +1 розряд (1 знаковий) представлені на малюнку:



Окремих розрядів для запису цілої частини числа (0) і коми не виділяється, так як їх становище зумовлено формою запису чисел.

Знак числа зазвичай кодується таким чином: знаку «+» відповідає 0 у знаковому розряді, знаку «-» - 1.

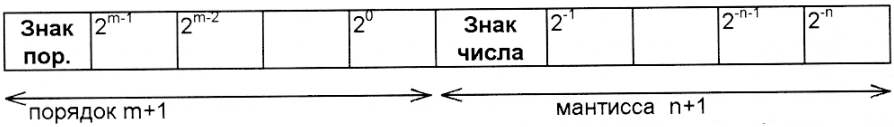
При представленні чисел з фіксованою комою в разі виконання арифметичних дій над довільними числами програміст може прийняти будь-яке умовне положення коми в межах формату. Але при написанні програми він повинен стежити за положенням коми під час обчислень, щоб не виникло переповнення.

Необхідність розрахунку масштабних коефіцієнтів, необхідність стеження за положенням коми під час обчислень виключаються при **представленні чисел з плаваючою комою.**

У загальному випадку число можна представити у вигляді добутку цілого ступеня основи системи і цифрової частини, що є правильним дробом:

$A = p^m a = p^m \sum a_i p^{i-m}$. (і від -k до n), де a - мантиса, m - порядок.

Формат числа, представленого у формі з плаваючою комою, має вигляд:



У розрядній сітці передбачено наявність розряду для фіксації знаку мантиси, який відповідає знаку числа.

Представлення числа з плаваючою комою можна проілюструвати на наступному прикладі:
$987.54 = 10^3 * 0.98754$,
$987.54 = 10^4 * 0.098754$,
$987.54 = 10^5 * 0.0098754$.
$987.54 = 10^2 * 9{,}8754$,
$987.54 = 10^1 * 98{,}754$

З метою однозначного представлення будь-якого числа введено поняття "нормалізоване число". Нормалізованим вважається число $\mathbf{A = p_a m_a}$, мантиса якого задовольняє нерівності:

$\mathbf{p^{-1} \le \lvert m_a \rvert \le 1 - p^{-k}}$	
0,100	0,1111…1
0,5	0,99999…9

Іншими словами, нормалізованим вважається те число, у якого старший розряд дорівнює 1.

Діапазон представлення порядку числа лежить в межах:

$\mathbf{2^k - 1 \ge p_a \ge -(2^k - 1).}$

Звідси випливає, що діапазон представлення чисел для p = 2:
мінімальне число:

$$\left|A_{min}\right| = 2^{-\left(2^k - 1\right)} \cdot 2^{-1} = 2^{-2^k}$$

і максимальне:

$$\left|A_{max}\right| = 2^{\left(2^k - 1\right)} \cdot \left(1 - 2^{-n}\right) = 2^{2^k - 1}$$

Очевидно, що діапазон представлення чисел в машинах з плаваючою комою значно більший, ніж у машинах з фіксованою комою:

$$D = \frac{\left|A_{max}\right|}{\left|A_{min}\right|} \approx \frac{2^{k-1}}{2^{-2^k}} = 2^{2^k - 1}$$

Зіставляючи між собою дві основні форми подання чисел в ЕОМ, можна прийти до наступних висновків.

Діапазон представлення чисел в машинах з фіксованою комою значно менший, ніж у машинах з плаваючою комою, а точність залежить від величини вихідних чисел. Програмування для машин з фіксованою комою значно складніше, тому що доводиться вводити масштабні коефіцієнти, щоб уникнути переповнення масштабної сітки при виконанні арифметичних операцій.

Однак машини з плаваючою комою конструктивно більш складні, оскільки необхідно вводити додаткове обладнання для виконання операцій над порядками чисел, а також передбачити операцію нормалізації і вирівнювання порядків чисел. Час виконання операцій над числами в машині з плаваючою комою більший, ніж в машині з фіксованою комою, що обумовлено необхідністю роботи з порядками.

Як і при фіксованій комі, тут можливо переповнення розрядної сітки, яке виражається в тому, що результат будь-якої операції має порядок більше припустимого. Це призводить до аварійної ситуації. При виконанні операцій можливе отримання чисел, що мають порядок менше допустимого і нормалізовану мантису. Ці числа розглядаються як машинні нулі, так само як і числа, що мають нульову мантису і допустимий порядок.

Іноді нормальну форму представлення чисел називають напівлогарифмічною, так як порядок числа р виражений у логарифмічній формі.

2.6 Точність представлення чисел в ЭОМ

При вирішенні різних завдань потрібна різна точність одержуваних результатів. Так, при вирішенні інженерних завдань достатня точність до 3-4 десяткових знаків (10-13 двійкових), при вирішенні наукових завдань - 5-6 десяткових або 16-20 двійкових знаків і при вирішенні особливо точних завдань - до 50 двійкових розрядів.

При обмеженій довжині машинних слів множина чисел, які можна представити в машині, є кінцевою. Тому представлення чисел в ЕОМ, як правило, тягне за собою появу похибок, величина яких залежить як від форми представлення чисел, так і від довжини розрядної сітки. Точність представлення числа характеризується абсолютною і відносною похибками.

Абсолютна похибка - це різниця між істинним значенням величини А і її значенням, отриманим з машинного зображення [А], тобто

$$\Delta[A] = A - [A]$$

Усереднена абсолютна похибка представлення чисел у машинах з фіксованою комою визначається як середнє арифметичне між мінімально представленим числом і його мінімальною втратою, тобто

$$\Delta = \frac{A_{min} + 0}{2} = \frac{2^{-n} + 0}{2} = 2^{-(n+1)}$$

У машинах з фіксованою комою абсолютна похибка постійна і дорівнює половині молодшого розряду.

Відносна похибка представлення визначається як відношення усередненої абсолютної похибки до самого числа:

$$\varepsilon = \left|\frac{\Delta}{A}\right|.$$

Так як саме число з фіксованою комою змінюється в межах

$$A_{min} = 2 - n \le |A| \le 1 - 2^{-n} = A_{max},$$

то і відносна похибка є величиною змінною, що змінюється в межах

$$\varepsilon_{min} \le \varepsilon \le \varepsilon_{max}$$

Для машин з фіксованою комою вона визначається таким чином:

$$\varepsilon_{\min} = \left|\frac{\Delta}{A_{\max}}\right| = \frac{2^{-(n+1)}}{1 - 2^{-n}} \approx 2^{-(n+1)};$$

$$\varepsilon_{\max} = \left|\frac{\Delta}{A_{\min}}\right| = \frac{2^{-(n+1)}}{2^{-n}} \approx 2^{-1}.$$

Таким чином, відносна похибка для машин з фіксованою комою залежить від величини числа і коливається в межах від $2^{-(n+1)}$ для великих чисел, до 2^{-1} для малих чисел. У машинах з плаваючою комою абсолютна похибка представлення числа визначається таким чином:

$$\Delta = \Delta a \cdot 2^m$$

де Δa - похибка представлення мантиси, яка визначається так само, як абсолютна похибка представлення чисел в машині з фіксованою комою, тобто

$\Delta a = 2^{-(n+1)}$; m - порядок числа, котрий змінюється в межах

$$-(2^k - 1) \le m \le +(2^k - 1).$$

Отже, на відміну від машин з фіксованою комою, в машинах з плаваючою комою абсолютна похибка представлення чисел залежить від порядку числа: мінімальна при найбільшому від'ємному m і максимальна при найбільшому додатному та визначаються наступним чином:

$$\Delta_{\min} = 2^{-(n+1)} \cdot 2^{-(2^k-1)} = 2^{-(n+2^k)};$$

$$\Delta_{\max} = 2^{-(n+1)} \cdot 2^{2^k-1} = 2^{2^k-n-2}.$$

Відносна похибка представлення чисел у машинах з плаваючою комою визначається за загальним правилом:

$$\varepsilon = \left|\frac{\Delta}{A}\right| = \frac{\Delta a \cdot 2^m}{a \cdot 2^m} = \frac{\Delta a}{a},$$

т. е. ε не залежить від порядку числа і змінюється в межах

$$\varepsilon_{min} = \frac{\Delta a}{a_{max}} = \frac{2^{-(n+1)}}{1-2^{-n}} = 2^{-(n+1)};$$

$$\varepsilon_{max} = \frac{\Delta a}{a_{min}} = \frac{2^{-(n+1)}}{2^{-1}} = 2^{-n}.$$

Отже, в машинах з плаваючою комою, на відміну від машини з фіксованою комою, відносна похибка зображення чисел у всьому діапазоні представлення практично постійна і для чисел з нормалізованою мантисою залежить від кількості розрядів мантиси: чим їх більше, тим менше похибка представлення.

У деяких обчислювальних засобах інформаційною одиницею є не окремі числа, а їх блоки або масиви, тобто послідовності, що складаються з сотень і тисяч чисел. У цих випадках нерідко застосовується проміжна форма представлення чисел в ЕОМ, так зване подання з поблочно плаваючою комою, при якому всьому масиву чисел присвоюється загальний порядок та масив вважається нормалізованим, якщо хоча б одне його слово є нормалізованим. Природно, що відносна похибка подання окремих елементів масиву буде при цьому різною. Як і у випадку подання з фіксованою комою, максимальний по абсолютній величині елемент буде представлений з мінімальною, тоді як мінімальний за абсолютною величиною елемент масиву - з максимальною відносною похибкою. Однак це не має істотного значення, оскільки основне інформаційне навантаження в цих випадках несуть максимальні елементи масивів. Разом з тим завдяки представленням чисел з поблочно плаваючою комою вдається при прийнятній точності обчислень значно скоротити об'єм обладнання, а головне - час виконання операції, так як дії над порядками в цьому випадку виконуються тільки один раз за час обробки всього масиву чисел.

З цього випливає, що не можна віддати перевагу будь-якій одній формі представлення чисел. Зазвичай в ЕОМ загального призначення застосовують нормальну форму. Цим забезпечується великий діапазон представлення чисел, висока точність обчислень, простота програмування. Ускладнення апаратури цих ЕОМ має другорядне значення.

У спеціалізованих ЕОМ частіше застосовують фіксовану або по блоках плаваючу кому, якщо інформація обробляється окремими масивами, так як ці форми забезпечують простоту конструкції ЕОМ. Діапазон зміни величин відомий заздалегідь, масштабні коефіцієнти підбираються один раз, необхідна точність обчислень також відома заздалегідь і визначає довжину розрядної сітки.

У сучасних ЕОМ використовуються обидві форми представлення чисел. При цьому в більшості випадків формат чисел з фіксованою комою служить для представлення цілих двійкових і десяткових чисел і виконання операцій над ними, що, наприклад, необхідно для операцій над кодами адрес (операції індексної арифметики) .

3 ТЕМА. ВИКОНАННЯ ОПЕРАЦІЙ АЛГЕБРАЇЧНОГО ДОДАВАННЯ І ЗСУВУ В ЕОМ

3.1 Формальні правила двійкової арифметики

Популярність двійкової системи числення пояснюється простотою правил двійкової арифметики.

Додавання	Віднімання	Множення
0+0=0	0-0=0	0x0=0
0+1=1	0-1=1 (позика) 1	0x1=0
1+0=1	1-0=1	1x0=0
1+1=0 1	1-1=0	1x1=1
(перенос в ст. розр.)		

Основною операцією в ЕОМ є додавання. За способом її виконання арифметичні пристрої можуть бути паралельної, послідовної, паралельно-послідовної дії. Послідовне додавання має виконуватися на підставі наступного рівняння:

$$a_i+b_i+П_{i-1}= S_i+П_i$$

3.2 Операція алгебраїчного додавання в ЕОМ.

При обчисленні суми двох чисел можливі два варіанти: доданки мають однакові знаки і доданки мають різні знаки. У результаті цього алгоритми отримання суми для кожного з них різні.

Для операндів з однаковими знаками:

1. Додати два числа.
2. Сумі присвоїти знак одного з доданків.

Алгоритм отримання алгебраїчної суми:

1. Порівняти знаки доданків, і якщо вони однакові, то виконати додавання по першому алгоритму.
2. Якщо знаки доданків різні, то порівняти доданки за абсолютною величиною.
3. Відняти з більшого менший.
4. Результату присвоїти знак більшого доданка.

З цього випливає, що перший алгоритм простіше другого. Отже, бажано перетворити від’ємні числа таким чином, щоб операцію віднімання замінити

операцією додавання, тобто S=A+(-B).

Для того, щоб вирішити цю проблему, необхідно вводити спеціальні коди: ***прямий, обернений, додатковий***.

Спосіб побудови цих кодів визначається функціями кодування, які повинні забезпечити:

1 . Запис алгебраїчного знака числа.

2 . Представлення від'ємних чисел за допомогою допоміжних, додатних, які відрізняються по зображенню від додатних вихідних чисел, тобто області зображень додатних і від'ємних чисел не повинні перетинатися.

3 . Повну ідентичність алгоритмів виконання операцій над числами різних знаків, і отже, повну ідентичність необхідного при цьому устаткування.

3.2.1 Прямий код

Прямим кодом від’ємного числа називається його зображення в природній формі запису, у якого в знаковому розряді ставиться 1. Прямий код додатного двійкового числа збігається з його звичайним зображенням у природній формі, так як знак кодується нулем.

Згідно з визначенням, функція кодування чисел в прямому коді правильних дробів виду: $A=a_{зн}\, a_{-1}\, a_{-2}\ldots\, a_{-n}$ запишеться наступним чином:

$$|A|\cdot p=\begin{cases} A & \text{при } A>0; \\ 1+|A| & \text{при } A<0. \end{cases}$$

Величина А буде визначатися в прямому коді наступним виразом:

$$A=(1-2a_{зн})\sum_{i=-k}^{n} a_i p^i,$$

при цьому знаковому розряду не приписується ніякої ваги. Очевидно, що діапазон зміни машинних зображень для прямого коду двійкового дробу лежить в межах:

$$-(1-2^{-n})\le |A|_{пр}\le (1-2^{-n}).$$

У геометричній інтерпретації область додатних чисел буде збігатися з областю їх зображень, а для від’ємних чисел ці області будуть відрізнятися. Геометрична інтерпретація прямого коду представлена на рис. 3.1.

Рис. 3.1. Геометрична інтерпретація прямого коду

У прямому коді нуль має два значення: додатне 0,000 .. 0 і від'ємне 1,000 ... 0. Зазвичай в ЕОМ використовується додатний нуль, але в процесі обчислень може виникнути і його від'ємне зображення. Обидва зображення повністю еквівалентні і застосування будь-якого з них не призводить до помилки.

Приклад запису числа в прямому коді:
A=+0,101011 $A_{пр}$= 0,101011;
B=-0,110011 $B_{пр}$=1,110011.

3.2.2 Додавання у прямому коді.

Правила додавання чисел в прямому коді не відрізняються від звичайних правил додавання, тобто якщо обидва доданків мають однакові знаки, то їх числові розряди додаються, а сумі присвоюється знак одного з них. Якщо доданки мають різні знаки, то з числових розрядів більшого по абсолютній величині числа віднімається менший, а сумі приписується знак більшого з доданків. При цьому числові розряди коду обробляються окремо від знакових, так як останні не мають ваги.

Розглянемо можливі 4 випадки отримання суми чисел в прямому коді.

1) A>0, B>0, C>0.
A=+0,101001B=+0,000101
$A_{пр}$=0,101001 $B_{пр}$=0,000101 $C_{пр}$=A+B
+0,101001
0,000101
0,101110

2) A>0, B<0, C>0.

A=+0,101001 B=-0,000101
$A_{пр}$=0,101001 $B_{пр}$=1,000101 $C_{пр}$=A-|B|
-0,101001
1,000101
0,100100

3) A<0, B>0, C<0.
A=-0,101001 B=+0,000101
$A_{пр}$=1,101001 $B_{пр}$=0,000101 $C_{пр}$=1+(|A|-|B|)
0,101001
0,000101
0,100100 $C_{пр}$=1+0,100100=1,100100

4) A<0, B<0, C<0.
A=-0,101001 B =-0,000101
$A_{пр}$=1,101001 $B_{пр}$=1,000101 $C_{пр}$=1+|A|+|B|
+0,101001
0,000101
0,101110 $C_{пр}$=1+ 0,101110=1,101110

Таким чином, у прямому коді знаковий розряд і цифрову частину не можна розглядати як єдине ціле. Крім того, необхідно окрім суматора мати і віднімач. У результаті цього ***прямий код не застосовується для виконання операції алгебраїчного додавання***, але застосовується для виконання операцій множення і ділення.

3.2.3 Додатковий код

У додатковому коді операція віднімання замінюється операцією алгебраїчного додавання. При цьому знаковий розряд і цифрова частина розглядаються як єдине ціле.

Розглянемо особливості перетворення в додатковий код. Від'ємне число замінюється деяким допоміжним додатним числом, причому:

$$[A]_{Д} = \begin{cases} A & \text{при } A > 0 \\ p + A \text{ при } A < 0 \text{ или } p - /A/ \end{cases}$$

При цьому для дробових від'ємних чисел завжди має місце: $|A|+|[A]_{Д}|=1$

Геометрична інтерпретація додаткового коду правильного дробу при p = 2 представлена на рис. 3.2.

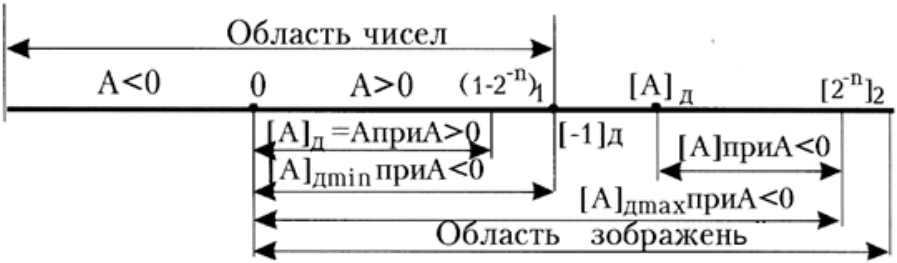


Рис. 3.2- Геометрична інтерпретація додаткового коду

З ростом абсолютної величини додатковий код додатного числа зростає, а від'ємного - зменшується.

З огляду на те, що область чисел і область зображень рівні по довжині модуля р = 2, між числами та їх зображеннями має місце однозначна відповідність. При цьому область додатних чисел збігається з областю зображень. Тому зображення додатних двійкових дробів не відрізняються від їх звичайного двійкового запису, а для зображення правильного від'ємного дробу до нього треба додати модуль 2, по якому порівнюється число з його зображенням, тобто отримати доповнення до двох.

Додатковий код додатного числа збігається з його поданням в прямому коді. Правило *перетворення від'ємного числа з прямого коду в додатковий* :

Для перетворення прямого коду від'ємного числа в додатковий необхідно все значущі розряди замінити на протилежні (інвертувати) і додати 1 до молодшого розряду . Знаковий розряд залишається без зміни.

$[A]_{пр}$=0,10110100; $[A]_{дк}$=0,10110100;

$[\hat{A}]_{пр}$=1,10111101000;

$[B]_{пк}$=1,10111101

$[B]_{дк}$=1,01000011.

3.2.4 Алгебраїчне додавання в додатковому коді

У додатковому коді операція віднімання замінюється операцією алгебраїчного додавання. При цьому знаковий розряд і цифрова частина числа розглядаються як єдине ціле, в результаті чого з від'ємними числами у додатковому коді операція віднімання замінюється операцією алгебраїчного додавання. При цьому знаковий розряд і цифрова частина числа розглядаються як єдине ціле, в результаті чого з від'ємними числами машина оперує як з неправильними дробами.

Правильний знак суми виходить автоматично в процесі додавання вмісту знакових розрядів операндів і одиниці переносу зі цифрової частини, якщо вона є.

Розглянемо всі можливі варіанти додавання чисел в додатковому коді:

1) A>0, B>0, C>0.

A=+0,101101 B=+0,000111

$A_{пр}$=0,101101 $B_{пр}$=0,000111

$A_{дк}$=0,101101 $B_{дк}$=0,000111 $C_{дк}=A_{дк}+B_{дк}$

+0,101101

0,000111

0,110100

2) A>0, B<0, C>0.

A=+0,101101 B=-0,000111

$A_{пр}$=0,101101 $B_{пр}$=1,000111

$A_{дк}$=0,101101 $B_{дк}$=1,111001 $C_{дк}=A_{дк}+B_{дк}$

+ 0,101101

1,111001

1←0,100110 (1 переносу з знакового розряду суми не враховується)

3) A<0, B>0, C<0.

A=-0,101101 B=+0,000111

$A_{пр}$=1,101101 $B_{пр}$=0,000111

$A_{дк}$=1,010011 $B_{дк}$=0,000111 $C_{дк}=A_{дк}+B_{дк}$

+ 1,010011

0,000111

1,011010 ($C_{пр}$= 1,100110)

4) A<0, B<0, C<0.

A=-0,101101 B=-0,000111

$A_{пр}$=1,101101 $B_{пр}$=1,000111

$A_{дк}$=1,010011 $B_{дк}$=1,111001 $C_{дк}=A_{дк}+B_{дк}$

+ 1,010011

1,111001

1←1,001100 ($C_{пр}$= 1,110100)

При додаванні в додатковому коді можливо переповнення розрядної сітки (у першому і четвертому випадках). Ознакою переповнення є відмінність знаку отриманої суми від знаків доданків.

3.2.5 Обернений код

Числа у оберненому коді представляються в наступному вигляді для двійкових чисел:

$$[A]_{об} = \begin{cases} A & \text{при } A > 0 \\ p - p^{-n} + A & \\ m.e. 2 - 2^{-n} + A & \text{при } A < 0 \end{cases}$$

Обернений код додатного числа збігається з його представленням в прямому коді. Обернений код від'ємного числа отримують інвертуванням всіх розрядів числа, крім знакового.

$[A]_{пр}$=0,10110100; $[A]_{ок}$=0,10110100;

$[B]_{пр}$=1,10111101; $[B]_{ок}$=1,01000010.

Геометрична інтерпретація оберненого коду представлена нп рис. 3.3.

Рис. 3.3 - Геометрична інтерпретація оберненого коду

3.2.6 Додавання у оберненому коді.

У оберненому коді, як і в додатковому, операція віднімання замінюється операцією додавання. При цьому знаковий розряд і цифрова частина числа розглядаються як єдине ціле. Правильний знак суми виходить в результаті підсумовування цифр знакових розрядів операндів і одиниці переносу зі цифрової частини, якщо вона є. Характерною особливістю додавання у оберненому коді є наявність циклічного переносу (якщо він виникає) із знакового розряду в молодший розряд цифрової частини, завдяки якому здійснюється корекція суми на 2^{-n}.

Розглянемо всі можливі варіанти додавання чисел у оберненому коді:

1) A>0, B>0, C>0.

A=+0,101101 B=+0,000111

$A_{пр}$=0,101101 $B_{пр}$=0,000111
$A_{ок}$=0,101101 $B_{ок}$=0,000111 $C_{ок}=A_{ок}+B_{ок}$

+0,101101
0,000111
0,110100

2) A>0, B<0, C>0.
A=+0,101101 B=-0,000111
$A_{пр}$=0,101101 $B_{пр}$=1,000111
$A_{ок}$=0,101101 $B_{ок}$=1,111000 $C_{ок}=A_{ок}+B_{ок}$
+ 0,101101
1,111000
1←0,100101 (1 переносу з знакового розряду суми додається в
→ 1 молодший значущий розряд результату)
0,100110

3) A<0, B>0, C<0.
A=-0,101101 B=+0,000111
$A_{пр}$=1,101101 $B_{пр}$=0,000111
$A_{ок}$=1,010010 $B_{ок}$=0,000111 $C_{ок}=A_{ок}+B_{ок}$
+ 1,010010
0,000111
1,011001 ($C_{пр}$= 1,100110)

4) A<0, B<0, C<0.
A=-0,101101 B=-0,000111
$A_{пр}$=1,101101 $B_{пр}$=1,000111
$A_{ок}$=1,010010 $B_{ок}$=1,111000 $C_{ок}=A_{ок}+B_{ок}$
+ 1,010010
1,111000
1←1,001010 (1 переносу з знакового розряду суми додається в
→ 1 молодший значущий розряд результату)
1,001011 ($C_{пр}$= 1,110100)

При додаванні в оберненому коді, як і в додатковому, в 1-му і 4-му випадках можливе переповнення.

Для спрощення виявлення переповнення розрядної сітки ЕОМ

використовуються модифікований обернений і модифікований додатковий коди.

$$[A]_{об}^{м} = \begin{cases} A & \text{при } A > 0 \\ 4 + 2^{-n} + A & \text{при } A < 0 \end{cases}$$

При представленні додатних чисел в модифікованому коді в знакових розрядах використовується 00, а від'ємних - 11. Ознакою переповнення розрядної сітки є різні значення в знакових розрядах.

Приклад 1.

$[A]^{м}_{ок}$=00,101101 $[B]^{м}_{ок}$=00,011100

00,101101

00,011100

01,001001 - додатне переповнення

Приклад 2.

$[A]^{м}_{ок}$=11,010010 $[B]^{м}_{ок}$=11,100011

11,010010

11,100011

1←10,110101

→ + 1

10,110110 - від'ємне переповнення.

$[A]^{м}_{ок}$ =11,010010 $[B]^{м}_{ок}$ =11,100011

11,010010

\+

11,100011

1←10,110101

\+ → 1

10,110110

3.3 Операція зсуву в ЕОМ

До операції зсуву доводитися звертатися при виконанні додавання в машині з плаваючою комою, а також при виконанні операцій множення і ділення в ЕОМ обох типів. У всіх цих операціях проводиться зсув мантис, тому будуть розглядатися тільки числа з фіксованою комою.

Зсув прямого коду числа на k розрядів вправо еквівалентний множенню

цього числа на 2^{-k}. З огляду на те, що при зсуві вправо молодші розряди зсунутого числа виходять за межі розрядної сітки машини і губляться, похибка представлення зсунутого коду числа має від'ємний знак для кодів додатних чисел і додатний знак для кодів від'ємних чисел. Для її зменшення необхідно вживати округлення чисел. При зсуві прямого коду від'ємного дробу зсувається тільки його мантиса, а знак залишається без зміни.

Приклад 1.

A= 1,011010, k=1 (лівий зсув) k=-1 (правий зсув)

2^{1}xA=1,<u>110100</u> 2^{-1}xA=1,0<u>01101</u>

Зсув прямого коду числа вліво на k розрядів еквівалентний множенню числа на 2^{k}. Ця операція коректна до тих пір, поки старші значущі цифри не почнуть виходити за межі розрядної сітки, тобто поки число за абсолютною величиною не стане більше 1. При зсуві вліво, розряди, що звільняються праворуч, заповнюються 0.

Зсув додатного числа вліво або вправо в додатковому або оберненому кодах нічим не відрізняються від зсуву додатного числа в прямому коді.

Під ***зсувом від'ємного*** числа А, записаного інверсним (додатковим або оберненим) кодом, розуміється перетворення інверсного коду від'ємного числа А в інверсний код від'ємного числа $Ax2^{-k}$ у разі зсуву вправо і $Ax2^{k}$ у разі зсуву вліво.

Загальним правилом зсуву дробів вправо в інверсному коді є наявність передачі з знакового розряду в старший цифровий розряд і відновлення знака, тобто розряди, що звільняються праворуч, заповнюються 1.

При зсуві вліво розряди, що звільняються праворуч в оберненому коді заповнюються 1, а в додатковому - 0. Кількість зсувів правильного дробу вліво обмежено умовою | Ax2k | <1, тобто зсув допустимий лише до тих пір, поки в розряді праворуч від коми не з'явиться 0 (поки зберігається знак результату). Зміна знака результату при зсуві вліво є ознакою переповнення, який для від'ємних і додатних чисел збігається з ознакою переповнення, що виникає при додаванні кодів двох чисел.

Приклад 2.

Зсув у оберненому коді:

$[A]_o$= 1,011010, k=1 (лівий зсув) k=-1 (правий зсув)

2^{1}xA=1,110101 2^{-1}xA=1,101101

Зсув в додатковому коді:

$[A]_д$= 1,011010, k=1 (лівий зсув) 2^1xA=1,110100 k=-1 (правий зсув) 2^{-1}xA=1,101101

3.4 Алгоритм додавання чисел в машинах з плаваючою комою

Результат додавання двох чисел A = $m_a p_a$; B = $m_b p_b$, представлених у формі з плаваючою комою, повинен бути теж числом виду C = $m_c p_c$. При цьому має виконуватися рівність:

$m_a p_a + m_b p_b = m_c p_c$

При додаванні чисел, представлених в нормальній формі, можна виділити 4 етапи:

1. Вирівнюються порядки доданків: менший порядок збільшується до більшого, а мантиса перетвореного числа зсувається вправо на відповідну кількість розрядів.

Для цієї мети проводиться віднімання порядків чисел. Знак і модуль різниці будуть визначати відповідно, який з доданків потрібно перетворювати і на скільки розрядів зсунути мантису.

2. Проводиться перетворення мантис в один з модифікованих інверсних кодів: додатковий або обернений.

3. Виконується додавання мантис за правилами додавання чисел з фіксованою комою.

4. Проводиться нормалізація результату і перетворення в прямий код, приписується загальний порядок доданків і виконується округлення мантиси результату.

Приклад 1. Прямий код.

A: p_a=0,011; m_a=1,101010; B: p_b=0,101; m_b=0,110010;

т. я. p_a - $p_в$=- 2 (p_a < $p_в$ на 2) то необхідно p_a збільшити до $p_в$ і відкоригувати мантису числа A зсувом на 2 розряди вправо.

A': p'_a=0,101; m'_a =1,001010;

$[m'_a]_д$=11,110110; $[m_b]_д$=00,110010; (після вирівнювання порядків)

+11,110110;

00,110010

1 00,101000 p_c=0,101; m_c=0,101000. Результат нормалізований.

3.5 Денормалізація чисел. Види денормалізації і методи усунення

Залежно від абсолютних величин мантис доданків результат може вийти нормалізованим, або денормалізованним (вліво - переповнення, або вправо). Додатні нормалізовані числа мають 1 в старшому розряді мантиси, а від'ємні числа, записані в інверсному коді, мають 0 в знаковому розряді. Розбіжність цифр у знакових розрядах свідчить про денормалізацію вліво (переповнення), а збіг цифр знакової і старшого значущого розряду мантиси - про порушення нормалізації вправо (права денормалізація).

Правила усунення денормалізації.

При денормалізації вліво мантиса зсувається на один розряд вправо, а порядок збільшується на 1.

При денормалізації вправо мантиса зсувається вліво до появи в старшому розряді 1, при знаку 0, або 0 при знаку 1, а з порядку віднімається кількість 1, яка дорівнює кількості зсувів мантиси.

Порядок перевірки денормалізації.

Спочатку виконується перевірка, чи не порушена нормалізація вліво і, якщо порушена, то усувається. Якщо нормалізація вліво не порушена, то перевіряється наявність правою денормалізації, і, якщо вона є, то усувається. Ліва денормалізація можлива тільки на 1 розряд, а права - на n (кількість розрядів, на яку може бути порушена нормалізація вправо, обмежена тільки довжиною розрядної сітки ЕОМ). Після виконання граничного числа зсувів мантису результату представляють машинним нулем. Мантису результату представляють також машинним нулем, якщо в процесі її зсуву порядок числа виявиться менше допустимого, тобто абсолютна величина результату буде менше, ніж мінімально можливе машинне число.

При додаванні може статися справжнє переповнення розрядної сітки числа, тобто переповнення розрядної сітки порядку. У цьому випадку мінімум один із операндів повинен мати максимальний порядок, а мантиса результату повинна вийти денормалізованою вліво. При цьому в ЕОМ формується ознака переповнення порядку.

Приклад 1.

$[A]_{пр}$ =0 101 1,10101;

$[B]_{пр}$ =0 011 0,11001; Найти C=A+B

порядок мантиса

$[B']_{пр}$ =0 101 0,0011001; (після зрівнювання порядків)
$[m_a]_д$=11,0101100
$[m_b]_д$=00,0011001
$[m_c]_д$=11,1000101

Так як мантиса результату денормалізована вправо на 1 розряд, то її необхідно зсунути на 1 розряд вліво і при цьому відняти з порядку 1.
$[C]_д$=0 100 11,000101; $[C]_{пр}$=0 100 11,111011;

Приклад 2.
$[A]_{пр}$ =0 101 1,10101;
$[B]_{пр}$ =0 100 1,11001; Найти C=A+B
порядок мантиса
$[B']_{пр}$ =0 101 1,011001; (після зрівнювання порядків)
$[m_a]_д$=11,010110
$[m_b]_д$=11,100111
$[m_c]_д$=10,111101

Так як мантиса результату денормалізована вліво, то її необхідно зсунути на 1 розряд вправо і при цьому порядок збільшити на 1.
$[C]_д$=0 110 11,0111101; $[C]_{пр}$=0 110 11,1000011.

3.6 Округлення чисел в ЕОМ

Вибір системи числення і довжина розрядної сітки ЕОМ, а також форми подання числа в машині залежать значною мірою від необхідної точності обчислень. Точність обчислень визначається також похибкою виконання арифметичних операцій при використанні в ЕОМ чисел, представлених у формі з фіксованою і плаваючою комою. Можна вважати, що в машині з фіксованою комою операції додавання і віднімання, за умови відсутності переповнення, виконується точно.

Джерелами похибок при додаванні в машині з плаваючою комою є зсув вправо мантиси одного з вихідних чисел при вирівнюванні порядків, зсув вправо мантиси при нормалізації результату, а також штучне встановлення нуля в якості результату при від'ємному переповненні порядків. Тому при нормальній формі представлення чисел сама операція алгебраїчного додавання є джерелом похибок.

Таким чином, причинами похибок обчислень в ЕОМ можуть бути:

1) неточне завдання вихідних даних, що беруть участь у виконанні операції (або через обмеження розрядної сітки машини, або через похибки переводу чисел з однієї системи числення в іншу);

2) використання наближених методів обчислень, що саме по собі дає методичну похибку (наприклад, використання методу прямокутників, трапецій при інтегруванні);

3) округлення результатів елементарних операцій, що в свою чергу може призвести до появи накопичених похибок.

3.6.1 Округлення чисел в прямому коді

Якщо припустити, що вихідна інформація не містить ніяких помилок і всі обчислювальні процеси виконуються абсолютно точно, то завжди існує третій тип помилок - помилки округлення, які виникають при переведенні чисел з однієї системи числення в іншу і наступному представленні їх у розрядної сітці машини, а також при отриманні усередині машини чисел, розрядністю більшою, ніж це допустимо, наприклад, при множенні. У цьому випадку число А округлюють, тобто замінюють його машинним числом [А] заданої розрядності. **Округлення (позначимо його знаком ☐) називається оптимальним, якщо для будь-якого машинного числа [А] справедливо ☐А=[А].** Нехай $[A]_1$ або $[A]_2$ - два послідовних машинних числа, тоді при оптимальному округленні дійсне число А таке, що $[A]_1<A<[A]_2$ замінюється або числом $[A]_1$, або числом $[A]_2$. Якщо ☐А≤А, то говорять про округлення по недоліку, якщо ☐А ≥А, то говорять про округлення по надлишку. Округлення називають симетричним, коли ☐А= -☐(-А). Розрізняють три види симетричного округлення.

1. Округлення у напрямку до нуля, коли дійсне число округлюється до найближчого до нуля машинного числа.

2. Округлення у напрямку від нуля, коли округлення здійснюється до машинного числа, що лежить далі від нуля, ніж дійсне число *А*.

3. Округлення по доповненню, коли округлення здійснюється до найближчого машинного числа.

В якості параметрів, за якими будуть порівнюватися способи округлення, доцільно використовувати максимальну величину модуля похибки, тобто Δ_{max}, де Δ = А-[А], і математичне очікування похибки округлення $\overline{\Delta}$.

Округлення до нуля або усічення. Для конкретності вважаємо, що числа в машині представлені в прямому коді з комою, фіксованою перед старшим розрядом, [А]= $a_{-1}...a_{-n}$ (правильний дріб). Нехай у результаті яких-небудь дій над

машинними числами усередині машини сформувалося число [A]', що має $k = n + t$ розрядів. Очевидно, що найпростіший спосіб округлення полягає у відкиданні хвоста числа [A]', який складається із зайвих розрядів, тобто розрядів з номерами a_{-n-1}, a_{-n-2}, ... ,a_{-n-t}.

Якщо вважати, що поява чисел з абсолютною величиною *A,* але різних знаків рівноймровірна і рівноймовірні всі значення хвоста чисел одного знаку, то математичне очікування похибки в даному випадку дорівнює нулю, тобто $\overline{\Delta} = 0$.

Зазвичай ймовірність появи чисел різного знаку при виконанні певної програми не однакова, тому представляє інтерес округлення абсолютних величин, тобто фактично чисел одного знаку.

При діях з числами одного знаку похибка усічення носить систематичний характер, що призводить до накопичення похибки. Ця обставина змушує досліджувати інші способи округлення, які розглядаються поки для прямих кодів.

Округлення від нуля. Реалізація даного способу вимагає аналізу хвоста на нуль, потім відкидається хвіст і, якщо частина, яка відкидається, не дорівнює нулю, до абсолютної величини частини, що залишилася додається одиниця в молодший розряд. Це додавання може викликати поширення переносів через всі розряди числа, що вимагає в загальному випадку виконання операції додавання для реалізації даного способу округлення. Крім додаткових тимчасових витрат це може призвести до переповнення розрядної сітки. Отже, спосіб складніший в реалізації, хоча основні його характеристики точно такі ж, як і при усіканні.

Округлення через нестачу. Реалізація даного способу базується на аналізі знаку числа, яке округлюється. Якщо [A]'> 0, то округлення полягає у відкиданні хвоста. Якщо ж [A]' < 0, то хвіст також відкидається, а до величини частини, що залишилася додається одиниця в молодший розряд, якщо хвіст не дорівнює нулю. Таким чином, реалізація даного способу ще більш ускладнена порівняно зі способом округлення від нуля за рахунок аналізу знака числа [A]', хоча величина Δ_{max} залишилася при цьому незмінною.

Якщо розглядати округлення чисел тільки одного знака, то при $A' > 0$ даний спосіб збігається з урізанням, а при A' < 0 - з округленням від нуля. Звідси ясно, що він не може конкурувати з урізанням результатів.

Округлення по надлишку. Цей спосіб в усьому подібний до попереднього, з тією відмінністю, що додавання одиниці в молодший розряд частини числа, яка залишається, здійснюється, коли воно більше нуля і хвіст не дорівнює нулю. При A'<0 хвіст просто відкидається. Характеристики даного способу точно такі ж, як у попереднього, за винятком знака величини $\overline{\Delta}$, який змінюється на протилежний.

Округлення по доповненню. Даний спосіб являє собою об'єднання способів округлення від нуля і до нуля. Його реалізація пов'язана з корекцією частини числа, яка залишається, A', яка виробляється за результатами аналізу значення старшої цифри частини, яка відсікається a_{-n-1}, тобто цифри додаткового $(n + 1)$-го розряду (ДР). Коли (n + 1 розряд) = 0, відбувається округлення до нуля, в іншому випадку - від нуля.

У разі рівноймовірно появи чисел різних знаків і рівномірного розподілу ймовірностей появи різних значень хвоста числа математичне очікування похибки округлення дорівнює нулю. Однак, при округленні чисел одного знаку значення $\overline{\Delta}$ відмінне від нуля. Тому при округленні чисел одного знаку даний спосіб дає систематичні помилки округлення, хоча і менші, ніж при усіканні.

За своїми характеристиками спосіб округлення по доповненню кращий, ніж усічення. Систематичні помилки під час округлення чисел одного знаку обумовлені в даному випадку тим, що округлення особливо неточно виробляється, коли значення частини числа, яка відсікається, близько до половини одиниці молодшого розряду, який зберігається. Цей недолік усувається в наступному способі округлення.

Удосконалене округлення по доповненню. У цьому випадку рішення про корекцію частини числа *A'*, що зберігається, приймається на основі аналізу значення всіх розрядів його частини, що відсікається, а не тільки старшого розряду. Таким чином, удосконалений спосіб дає гарне округлення у разі чисел одного знаку, проте це досягається за рахунок ускладнення його реалізації.

Спрощене округлення по доповненню. При реалізації способів округлення по доповненню через виникнення переносів при підсумовуванні частини числа, що зберігається, з одиницею округлення необхідно виконати операцію додавання, що вимагає додаткового часу, тобто знижує реальну швидкодію ЕОМ і, крім того, може спричинити за собою переповнення розрядної сітки. Цей спосіб округлення полягає в тому, що молодший розряд частини числа, що зберігається, примусово встановлюється в одиницю, якщо старший розряд відкидаємої частини дорівнює одиниці.

При цьому, якщо в неокругленому результаті розряд a_{-n} дорівнює одиниці, то він не змінюється при округленні. Якщо в неокругленому результаті операції значення розряду a_{-n} нуль, то установка його в одиницю вносить похибку.

Таким чином, при рівноймовірній появі нуля і одиниці в молодшому розряді частини, що зберігається для знакозмінних чисел знову отримаємо симетричний розподіл похибок.

Ймовірнісне округлення. Для такого округлення необхідно мати датчик випадкових величин (0 або 1), одиниця з виходу якого додається до молодшого розряду частини числа, що зберігається. Похибка округлення при рівноймовірному розподілі значень відкидаємої частини є випадковою величиною з нульовим математичним очікуванням.

Таким чином, найпростішим способом округлення є усічення, при якому не потрібно додаткових витрат часу і устаткування. Однак на практиці найважливіше точність обчислень. Тільки для трьох способів округлення по доповненню максимальна помилка близька до половини одиниці молодшого розряду машинного числа, тобто є найменшою. Найбільш швидкодіючим з них є спрощений спосіб, а найбільш точним - вдосконалений. Тому перевагу тому чи іншому способу округлення слід віддавати тільки після аналізу вимог, що пред'являються до швидкодії і похибки обчислень конкретної машини. Будемо надалі користуватися способом округлення по доповненню.

3.6.2 Особливості округлення чисел, заданих інверсними кодами

Так як додатні числа в прямому, оберненому і додатковому кодах представляються однаково, то і правила округлення додатних чисел у всіх трьох кодах будуть тими ж. Однак округлення від'ємних чисел, записаних в інверсних кодах, має ряд особливостей.

Обернений код. При округленні по доповненню завжди округлюється абсолютна величина, внаслідок цього з додаткового розряду (ДР) від'ємного дробу необхідно відняти одиницю, тобто додати до округленого дробу обернений код - $1_{окр}$ = 1,11...10, де цифра 0 записана в ДР. Ланцюжок циклічного переносу повинен при цьому охоплювати і цей ДР.

Додавати код 1,11...10 і перебудовувати ланцюжок циклічного переносу для охоплення ДР, тобто (n+ 1)-го розряду суматора, вельми незручно. Тому від'ємні дроби зазвичай округлюються після їх переведення з оберненого в прямий код.

Додатковий код. При округленні від'ємного дробу, заданого в додатковому коді, розрізняють два випадки. Якщо праворуч від ДР знаходиться хоча б одна одиниця, то в ДР додається одиниця округлення, після чого всі розряди, починаючи з додаткового, відкидаються. Якщо в ДР знаходиться одиниця і ця одиниця є молодшою в коді числа, то всі розряди, починаючи з додаткового, просто відкидаються.

4 ТЕМА . ВИКОНАННЯ ОПЕРАЦІЇ МНОЖЕННЯ І ДІЛЕННЯ У ЕОМ

4.1 Виконання операції множення в ЕОМ

Операція множення є найчастішою після додавання. Множення може виконуватися підсумовуванням зсунутих на один або кілька розрядів часткових добутків, кожен з яких є результатом множення множимого на відповідний розряд (розряди) множника.

При точному множенні двох чисел кількість значущих цифр добутку може гранично досягти подвійної кількості значущих цифр співмножників. Ще складніша виникає ситуація при множенні кількох чисел. Тому у добутку тільки в окремих випадках використовують подвійну кількість розрядів.

Найбільш просто операція множення виконується у прямому коді. При цьому на першому етапі визначається знак добутку шляхом складання знакових розрядів співмножників за модулем 2, потім виконується перемножування модулів співмножників згідно з двійковою таблицею множення. Результату присвоюється отриманий знак.

Так як множення проводиться у двійковій системі числення, часткові добутки або рівні 0 (при множенні на 0), або самому множнику (при множенні на 1), зсунутому на відповідну кількість розрядів.

Добуток можна отримати двома шляхами:

1) зсувом множимо на необхідну кількість розрядів і додаванням отриманого чергового часткового добутку до раніше накопиченої сумі часткових добутків;

2) зсувом суми раніше отриманих часткових добутків на кожному кроці на 1 розряд і подальшим додавання до зсунутої суми нерухомого множимого або 0. Причому кожен з цих методів може змінюватися ще й тим, з молодших або зі старших розрядів починається множення.

Приклад.

A=0,1101; B=0,1011;

1а)

```
  0,1101
  0,1011
   1101
  1101
 0000
1101
10001111
```

1б)

```
  0,1101
  0,1011
   1101
    0000
     1101
    1101
10001111
```

Ґрунтуючись на вищевикладеному можна створити 4 основні методи машинного множення в прямому коді:

1) множення молодшими розрядами множника із зсувом накопичуваної суми часткових добутків вправо;

2) множення молодшими розрядами множника зі зсувом множимого вліво;

3) множення старшими розрядами множника із зсувом накопичуваної суми часткових добутків вліво;

4) множення старшими розрядами множника зі зсувом множимого вправо;

Розглянемо більш детально кожну зі схем множення.

1) Множення молодшими розрядами множника із зсувом накопичуваної суми часткових добутків вправо.

Алгоритм отримання результату по даному методу може бути наступним:

а) множиме множиться на молодший (1-й) розряд множника;

б) результат множення підсумовується з нульовим вмістом суматора;

в) вміст суматора зсувається на 1 розряд вправо;

г) множиме множиться на наступний (2-й) розряд множника;

д) результат підсумовується з вмістом суматора;

е) вміст суматора зсувається на 1 розряд вправо;

ж) пункти г, д, е повторюються n раз.

Реалізація даного алгоритму вимагає n-розрядного зсувного регістру множника, n схем "І", що пропускають на вхід суматора при $b_i=1$ і забороняють його передачу при $b_i=0$, n- розрядного регістра множимо і 2n-розрядного суматора, в якому накопичується сума і є ланцюги для її зсуву на 1 розряд вправо. Структурна схема операційного пристрою, що виконує множення за даним алгоритмом представлена на рис. 4.1.

Якщо виключити додаткові розряди суматора, то добуток буде обчислюватися з тією ж кількістю розрядів, що й співмножники. Однак для округлення результату бажано подовжити СМ на один додатковий розряд з тим, щоб після обчислення добутку додати до додаткового розряду 1 і вивести з СМ результат без додаткового розряду.

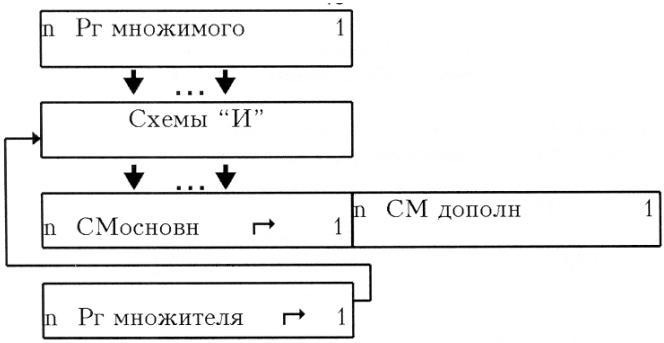


Рис. 4.1 Схема множення по першому алгоритму.

Приклад.

Задані операнди A=0,0101; B=0,1011, виконати операцію множення.

Таблиця 4.1.

№ п/п	Роз-ряд мн-ля	Найменування операції	1	2	3	4	5	6	7	8
		обнуління	0	0	0	0	0	0	0	0
1	B_1=1	Ах B_1	0	1	0	1	0	0	0	0
		$\sum$	0	1	0	1	0	0	0	0
		$\rightarrow$	0	0	1	0	1	0	0	0
2	B_2=1	Ах B_2	0	1	0	1	0	0	0	0
		$\sum$	0	1	1	1	1	0	0	0
		$\rightarrow$	0	0	1	1	1	1	0	0
3	B_3=0	Ах B_3	0	0	0	0	0	0	0	0
		$\sum$	0	0	1	1	1	1	0	0
		$\rightarrow$	0	0	0	1	1	1	1	0
4	B_4=1	Ах B_4	0	1	0	1	0	0	0	0
		$\sum$	0	1	1	0	1	1	1	0
		$\rightarrow$	0	0	1	1	0	1	1	1

C=0,00110111.

2) Множення молодшими розрядами множника зі зсувом множимого вліво.

Алгоритм отримання результату по даному методу може бути наступним:

а) множиме множиться на молодший (1-й) розряд множника;

б) результат множення підсумовується з нульовим вмістом суматора;

в) множиме зсувається на 1 розряд вліво;

г) множиме множиться на наступний (2-й) розряд множника;

д) результат підсумовується з вмістом суматора;

е) множиме зсувається на 1 розряд вліво;

ж) пункти г, д, е повторюються n раз.

Реалізація даного способу множення вимагає n-розрядного зсувного регістру множника, 2n – розрядний зсувний регістр множимого, 2 - n схем "І", які пропускають на вхід суматора при $b_i=1$ і забороняють його передачу при $b_i=0$, 2n-розрядний суматор. Структурна схема операційного пристрою, що виконує множення за даним алгоритмом представлена на рис. 4.2.

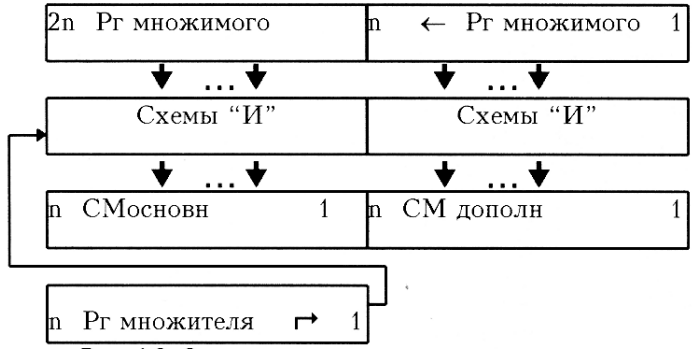


Рис. 4.2 Схема множення по другому алгоритму.

Виконання множення за 3-м і 4-м способами множення можна розглянути за аналогією до вище розглянутих способів.

Аналіз наведених схем множення показує, що тривалість процесу множення по будь-якій схемі становить n циклів: $T_у=n\tau_ц$.

Однак тривалість циклів у різних схемах не однакова. Так у другій і четвертій схемах $\tau_ц=\tau_{см}$ і, враховуючи, що $\tau_{см}>\tau_{сдв}$, ці схеми дозволяють прискорити процес виконання операції множення за рахунок суміщення операції додавання часткових добутків і зсувів множимого. За кількістю обладнання перевагу слід віддати першій, а потім третій схемі множення.

Найбільш зручними для застосування в ЕОМ є 1 і 4 схеми множення.

4.2 Множення чисел, представлених у формі з плаваючою комою.

Якщо операнди задані у формі з плаваючою комою: $A=m_a p_a$ і $B=m_b p_b$, то їх добуток $C=AxB$ и $C=m_c p_c$.

Алгоритм множення нормалізованих чисел складається з наступних етапів:

1. Визначення знака добутку шляхом складання знакових розрядів мантис операндів за модулем 2.

2. Алгебраїчне додавання порядків співмножників з метою визначення порядку добутку.

3. Множення модулів мантис співмножників за правилами множення чисел з фіксованою комою.

4. Нормалізація та округлення мантиси результату. Слід врахувати, що мантиси співмножників є нормалізованими числами. Тому денормалізація мантиси добутків можлива тільки на один розряд вправо. Вона усувається шляхом зсуву мантиси на один розряд вліво і віднімання 1 з порядку результату.

5. Присвоєння знака результату.

Перші три операції можуть виконуватися одночасно, так як вони незалежні. Наявність операції визначення знака добутку припускає, що множення мантис виконується в прямому коді.

При виконанні операції множення в машині з плаваючою комою може вийти переповнення від'ємного порядку, яке буде інтерпретовано як машинний нуль, якщо програмою користувача ігнорується ознака зникнення порядку. Може також виникнути додатне переповнення порядку. У цьому випадку в першу чергу необхідно нормалізувати мантису результату. Якщо і після цього переповнення порядку не усувається, то машиною формується ознака переповнення порядку.

4.3 Методи прискорення операції множення

Будь-яке прискорення операції множення навіть ціною ускладнення арифметичних і логічних схем дозволяє істотно підвищити продуктивність ЕОМ, тому що приблизно 70% машинного часу витрачається на виконання цієї операції.

Відомі способи прискорення множення, спрямовані на скорочення загальної кількості і часу виконання операцій додавання, необхідних для утворення добутку. Ці способи діляться на логічні і апаратні.

Під **апаратними** розуміють такі способи, які вимагають для своєї реалізації введення додаткового обладнання в основні арифметичні ланцюги, завдяки чому

досягається поєднання в часі окремих складових частин процесу множення. Вони поділяються на способи 1-го і 2-го порядків. Для реалізації способів 1-го порядку необхідно обсяг обладнання, пропорційний числу розрядів машинного слова n. Для реалізації способів 2-го порядку потрібно обсяг обладнання, пропорційний n^2.

Під **логічними** розуміють такі способи прискорення, при реалізації яких зберігається основна структура арифметичних ланцюгів помножувача, а прискорення досягається тільки за рахунок ускладнення схеми керування.

Найпростішим логічним способом є пропуск тактів підсумовування в тих випадках , коли чергова цифра множника дорівнює 0.

У середньому , кількість додавань при цьому скорочується вдвічі.

Можна скоротити і середню і максимальну кількість додавань при використанні як прямих, так і інверсних передач множимо в суматор. Тут враховується та обставина, що час множення значно скорочується, якщо за наявності в розрядах множника декількох нулів підряд виробляти його зсув відразу на кілька розрядів. Для цього видозмінюють код множника з метою представлення його з меншою кількістю розрядів, що містять одиницю. Наприклад, групу одиниць у множнику 011. ..110 можна перетворити в групу $100...0\bar{1}0$, тобто перейти до системи з цифрами $1,0,\bar{1}$.

Таким чином, в основі способу лежить представлення числа як сукупності таких послідовностей: нулів, одиниць, нулів з ізольованими одиницями, одиниць з ізольованими нулями. При цьому: два або більше сусідніх нулів або сусідніх одиниць розглядаються як послідовність. Наприклад, якщо множення починається з молодших розрядів і множник містить послідовність одиниць, то проводиться віднімання множимо з відповідною (молодшою) вагою розрядів, а потім виконується зсув через всі ці одиниці.

Зсув через послідовність одиниць припиняється на першому нулі. Якщо відразу за цим нулем розташована одиниця, то множиме віднімається і виконується зсув через послідовність одиниць. Якщо за цим нулем безпосередньо випливає другий нуль, то множиме додається, а потім виконується зсув через послідовність нулів , який припиняється на першій одиниці.

Якщо за цією одиницею слідує нуль , то множиме додається і виробляється зсув через послідовність нулів. Якщо за цією одиницею безпосередньо йде наступна одиниця, то проводиться віднімання множимо з відповідною вагою даного розряду, а потім виконується зсув через послідовність одиниць. Якщо в старшому розряді множника знаходиться 1, що входить в послідовність одиниць, то зсув необхідно продовжувати до першого нуля після старшого розряду множника.

Слід зазначити, що при множенні зі старших розрядів застосовуються дещо

інші правила визначення оптимального множника. І в тому і в іншому випадку в середньому на кожну операцію додавання виконується зсув на 2,9 розряду, якщо схема розрахована на зсув не більше, ніж на 6 розрядів одночасно.

У межі середнє число додавань - віднімань, що припадає на один розряд множника, близько 3^{-1} . Це найкращий результат, якого можна досягти при використанні логічних методів.

Таким чином, перехід від одного різновиду двійкової системи числення до іншого при перетворенні множника дозволяє отримати виграш у часі виконання операції в цілому. При цьому виникають визначеної довжини послідовності 0 або 1, що, в кінцевому рахунку, призводить до необхідності одночасного аналізу декількох розрядів множника і зсуву на довільне число розрядів.

Одночасне множення на два розряди. Кількість циклів, необхідних для реалізації в ЕОМ операції множення, можна скоротити, якщо в кожному циклі аналізувати не один, а два або більше розрядів множника, виконуючи після аналізу одну передачу множимо в суматор і зсув множника на відповідне число, тобто два або більше, розрядів. Для організації прискореного множення множник можна розбити на групи по два розряди і перетворити його таким чином, щоб кожна група містила не більше однієї одиниці, розуміючи під останньою 1 або $\bar{1}$.

Для молодшої пари розрядів при множенні з молодших розрядів можливі наступні комбінації одиниць і нулів в розрядах: 00, 01, 10 і 11.

Для першої комбінації не виробляється ні додавання, ні віднімання, для другої - підсумовування множимо, для третьої - підсумовування зсунутого на 1 розряд вліво множимого, тобто помноженого на два, а для четвертої - замість двох додавань при множенні без прискорення виконується одне віднімання множимого і одне додавання після зсуву множимого на 2 розряди, тобто пара розрядів множника перетвориться до виду $10\bar{1}$.

Оскільки додавання після зсуву доводиться на множення на наступну пару розрядів, то замість того, щоб його виконувати додають одиницю в наступну за даною, пару розрядів. З урахуванням цієї дії при множенні виконуються згідно з таблицею 4.2.

Описана процедура повторюється для всіх пар розрядів множника, а також для однієї пари розрядів лівіше коми, тому що може виявитися необхідним додати до неї (до її нулів) одиницю.

Таблиця 4.2.

Аналізована пара розрядів	Перенесення з попередньої пари розрядів	Перетворена пара розрядів	Примітка
00	0	00	
01	0	01	
10	0	10	Попередній зсув множимого
11	0	01	Запам'ятовується 1 для наступної пари розрядів
00	1	01	
01	1	10	Попередній зсув множимого
10	1	01	Запам'ятовується 1 для наступної пари
11	1	10	розрядів

Слід зазначити, що в загальному випадку при множенні на 2 розряди множника двох знакових розрядів в суматорі недостатньо. Тут можливі випадки при A → 1, коли навіть у другому знаковому розряді з'являється одиниця переповнення, тобто буде спотворений знак часткового добутку. Отже, при даному способі множення суматор повинен мати три знакових розряди.

При одночасному аналізі двох розрядів, починаючи зі старших, правила перетворення множника істотно змінюються. На характер дій впливає значення сусідньої праворуч цифри стосовно аналізованої пари розрядів. При цьому аналіз розрядів завжди починається з порожньої пари. Залежно від значення сусіднього праворуч розряду відбувається зміна стану перетворених розрядів і виконання дій згідно з таблицею 4.3.

Таблиця 4.3.

пара розрядів	Перенесення з попередньої пари розрядів	Перетворена пара розрядів	Примітка
00	0	00	
01	0	01	
10	0	$\bar{1}0$	Попереднє зрушення множимого
11	0	$0\bar{1}$	Запам'ятовується 1 для наступної пари розрядів
00	1	01	
01	1	$\bar{1}0$	Попереднє зрушення множимого
10	1	$0\bar{1}$	Запам'ятовується 1 для наступної пари розрядів
11	1	00	

Слід зазначити, що об'єм обладнання АУ при множенні на 2 розряди збільшується незначно в порівнянні з АУ, працюючим без прискорення.

Одночасне множення на три і більше розряди, для реалізації якого застосуємо подібний спосіб прискорення, використовується рідше, тому що при цьому збільшується кількість необхідних типів передач. Це призводить до значного ускладнення схем, що реалізують множення.

Матричний метод множення

Коли множиме і множник розташовані в регістрах машини, неважко утворити відразу всі часткові добутки. Отже, за наявності додаткових суматорів можна складати відразу кілька часткових добутків, а в граничному випадку і всі одночасно. У цьому випадку формування добутків можна собі уявити як спуск по дереву суматорів від доданків до їх загальної суми. Час спуску по дереву буде залежати від його організації і типу застосовуваних суматорів. Розглянемо схему множення на прикладі двох чотирьохрозрядних чисел:

			A=	a_4	a_3	a_2	a_1
			B=	b_4	b_3	b_2	b_1
				a_4b_1	a_3b_1	a_2b_1	a_1b_1
			a_4b_2	a_3b_2	a_2b_2	a_1b_2	
		a_4b_3	a_3b_3	a_2b_3	a_1b_3		
	a_4b_4	a_3b_4	a_2b_4	a_1b_4			
c_8	c_7	c_6	c_5	c_4	c_3	c_2	c_1

Цю схему множення можна представити у вигляді матриці (таблиця 4.4.), Кожен елемент якої дорівнює 0 або 1 для p = 2. Для отримання добутку двох чисел елементи матриці треба підсумовувати відповідно до порядку.

Таблиця 4.4

	a_i			
b_i	a_4	a_3	a_2	a_1
b_1	a_4b_1	a_3b_1	a_2b_1	a_1b_1
b_2	a_4b_2	a_3b_2	a_2b_2	a_1b_2
b_3	a_4b_3	a_3b_3	a_2b_3	a_1b_3
b_4	a_4b_4	a_3b_4	a_2b_4	a_1b_4

Елементи, що становлять кожен і-й стовпець матриці, підсумуємо за допомогою звичайних трьохвходових двоїчних суматорів. Значення переносів з цих суматорів повинні бути складовими для (і +1)-го стовпця. Це призведе до того, що в кожному (і +1)-м стовпці поява доданків відбуватиметься з запізненням по часу відносно і-го стовпця.

Схема пристрою для реалізації цього дерева спуску, яке представляє собою одну з різновидів матричного алгоритму множення, представлена на рис. 4.3.

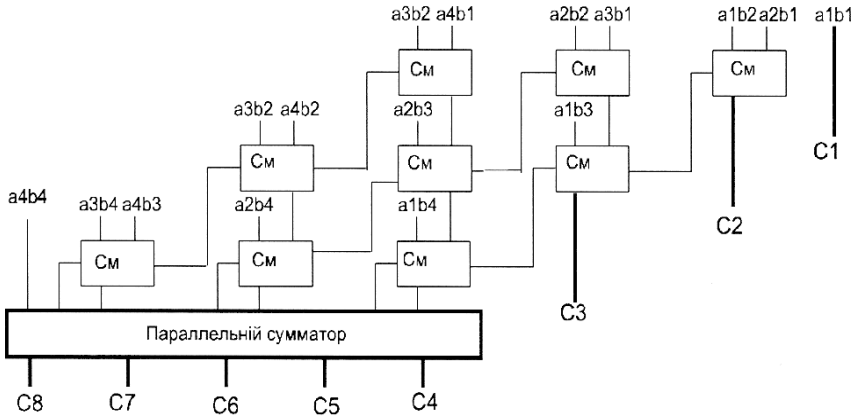


Рис. 4.3- Схема пристрою для реалізації матричного методу прискорення операції множення

Для однорозрядних суматорів час утворення суми більший, ніж час утворення переносів $\tau_п$, тому найбільший час спуску по дереву даного виду є $T_{уск}=(n-1)(\tau_{сум}+\tau_п)$; якщо тільки всі кон'юнкції виду a_ib_j надходять на дерево спуску одночасно. Цей час складається з спуску по найдовшій вертикалі, а потім послідовного поширення переносів через суматори нижнього ряду аж до самого лівого. Розглянута структура дерева спуску не єдина Час спуску можна ще зменшити, перетворивши дерево спуску.

4.4 Виконання операції ділення в ЕОМ

Операція ділення зустрічається порівняно рідко (P = 0,02), однак, реалізація її по підпрограмі займає досить великий час. Тому в більшості сучасних ЕОМ ділення реалізується спеціальними операційними блоками.

Ділення в ЕОМ, також як і множення, найпростіше виконується в прямому коді. Але на відміну від множення дробових чисел, де не може виникнути переповнення розрядної сітки, при діленні правильних дробів таке переповнення можливо у випадку, коли ділене більше дільника. Ознакою переповнення є поява цілої частини в частці, що грубо спотворює результат.

Знак частки при діленні визначається додаванням по модулю 2 знакових розрядів діленого і дільника і присвоюється результату наприкінці операції ділення.

$0 \oplus 0 = 0$

$0 \oplus 1 = 1$

$1 \oplus 0 = 1$

$1 \oplus 1 = 0$

Частка визначається шляхом ділення модулів діленого і дільника. При цьому, щоб уникнути переповнення розрядної сітки має дотримуватися умова: $|A|<|B|$, де A - ділене, B - дільник. Крім того: $B \neq 0$.

Відомі два основних алгоритми виконання операції ділення:

- Ділення з відновленням залишків;
- Ділення без відновлення залишків.

4.4.1 Ділення чисел з відновленням залишків

Нехай необхідно розділити A на B. Тоді частка від ділення

$$Y_i = A/B = 0,y_1y_2y_3... y_{i-1}y_i = y_1 2^{-1} + y_2 2^{-2} + y_3 2^{-3} ... y_{i-1}2^{-i-1} + y_i 2^{-i}. \quad (4.1.)$$

При будь-якому значенні n повинна виконуватися нерівність:
$0 \leq A - BY_i \leq B2^{-i}$, тобто залишок від ділення повинен бути менше дільника, помноженого на 2^{-i}, або: $0 \leq (A - B\ Y_i)2^i \leq B$, позначимо $(A - B\ Y_i)\ 2^i = R_i$ і отримаємо:

$$0 \leq R_i \leq B. \quad (4.2)$$

При діленні цифри частки визначаються послідовно, починаючи зі старшого розряду. Припустимо, що в результаті виконання n циклів отримані старші *i* розрядів частки, тобто наближене значення частки Y_i . В наступному (i+1)-циклі буде отримано значення (i+1)-го розряду частки. Вихідними даними для цього циклу є Y_i і R_i.

Цифра y_{i+1} може мати одне з двох значень: 1 або 0. Якщо $y_{i+1}=0$, то

$Y_{i+1}= Y_i+ y_{i+1} \times 2^{-(i+1)}= Y_i;$ (4.3.)

$R_{i+1}=(A-B\ Y_{i+1}) \times 2^{i+1}= 2\ R_i,$ (4.4.)

тобто в розряд частки записується 0 за умови $0 \le R_{i+1}=2R_i < B.$ (4.5.)

Якщо $y_{i+1}=1$, то $Y_{i+1}= Y_i+ 2^{-(i+1)};$ (4.6.)

$R_{i+1}=(A-B \times Y_{i+1}) \times 2^{i+1}= (A - B \times Y_i - B \times 2^{-(i+1)}\ 2^{(i+1)}=2\ R_i - B,$ (4.7.)

тобто цифра частки дорівнює 1, якщо виконується умова:

$0 \le R_{i+1}=2R_i - B < B$ (4.8.)

або $B \le 2R_i < 2B.$ (4.9.)

Так як завжди виконується одна з умов (4.5) Або (4.9), То для визначення поточної цифри частки достатньо перевірити одну з них.

Зазвичай перевіряють умову (4.5). Ліва частина цієї нерівності виконується заздалегідь, так як згідно (4.2) $0 \le Ri$, тобто черговий залишок перед початком наступного кроку ділення завжди є додатним числом.

Для перевірки правої частини нерівності порівняємо різницю $(2R_i - B)$ частки з нулем. Якщо ця різниця виявиться від'ємною, то в (i +1) розряд частки запишемо 0 і для підготовки вихідних даних для (i +2) -го циклу визначимо Ri +1 наступним чином:

$R_{i+1}=(2R_i - B) + B =2R_i .$ (4.10.)

Якщо різниця $2R_i-B$ виявиться додатною, то запишемо в (i +1) розряд частки 1, а в якості вихідного значення для наступного (i +2)-го циклу використовуємо обчислену різницю (див. 4.7.):

$R_{i+1}=2\ R_i - B,$

Вихідними даними для 1-го циклу є : $Y_0=0.$

$R_0=(A-BY_0)2^0= A< B,$

тобто за умовою нерівності (4.2.) виконується і перед початком першого циклу. Після закінчення n-го циклу отримаємо n-значну частку Yn, обчислену з недостачею $Rn = (A - BYn)2^n$, якиа дорівнює залишку від ділення А на В, зсунутому вліво на n розрядів.

Правило ділення з відновленням залишків формулюється таким чином.

Дільник віднімається з діленого і визначається знак нульового (по порядку) залишку. Якщо залишок додатний, тобто | A | > | B |, то в псевдознаковому розряді частки проставляється 1, при появі якої формується ознака переповнення розрядної сітки і операція припиняється. Якщо залишок від'ємний, то в псевдознаковому розряді частки записується 0, а потім проводиться відновлення діленого шляхом додавання до залишку дільника. Далі виконується зсув відновленого діленого на один розряд вліво і повторне віднімання дільника. Знак одержуваного таким чином залишку визначає першу значущу цифру частки: якщо залишок додатний, то в першому розряді частки записується 1, якщо від'ємний,

то записується 0. Далі, якщо залишок додатний, то він зсувається вліво на 1 розряд і від нього віднімається дільник для визначення наступної цифри частки. Якщо залишок від'ємний, то до нього додається дільник для відновлення попереднього залишку, потім відновлений залишок зсувається на 1 розряд вліво і від нього віднімається дільник для визначення наступної цифри частки і т.д. до отримання необхідної кількості цифр частки з урахуванням одного розряду для округлення, тобто до забезпечення необхідної точності ділення.

Приклад.

A=0,10011; B=0,11001; $[-B]_{доп}$= 11,00111; |B|= 0,11001

1. Визначення знака частки: 0⊕0=0
2. Визначення модуля частки. (див. Таб. 4.4.1)

Таким чином, цифри частки можна отримати як інверсне значення знакових розрядів поточного залишку, які приймають значення 00 або 11. Однак, при зсуві залишку вліво в знакових розрядах може виникнути комбінація 01. У деяких випадках, для того щоб цифри частки формувалися як пряме значення знакового розряду поточного залишку, ділення виконують з інверсними знаками. При цьому ділене передається в суматор не прямим, а інверсним кодом, а на нульовому кроці виконується операція «+ B», замість операції «-B».

Таб. 4.4.1

№ цик-лу	№ так-ту	Найменування операції	Дія			Розряди частки
0	1	Віднім. дільника	A	00	10011	
	2	із діленого	$[-B]_д$	11	00111	
			R_0	11	11010	0, 1 1 0 0
	3	Відновлення	+B	00	11001	
		0-залишку	R_1 1	00	10011	
1	1	Зсув залишку	$\leftarrow R_1$	01	00110	
	2	Віднім. дільника	$[-B]_д$	11	00111	
		формування	R_2	1 00	01101	
		розряду частки				
2	1	Зсув залишку	$\leftarrow R_2$	00	11010	
	2	Віднім. дільника	$[-B]_д$	11	00111	
		формування	R_3	1 00	00001	
		розряду частки				
3	1	Зсув залишку	$\leftarrow R_3$	00	00010	
	2	Віднім. дільника	$[-B]_д$	11	00111	
		формування		11	01001	
		розряду частки				
	3	відновл. залишку	+B	00	11001	
			R_4	1 00	00010	
4	1	Зсув залишку	$\leftarrow R_4$	00	00100	
	2	Віднім. дільника	$[-B]_д$	11	00111	
		формування		11	01011	
		розряду частки				
	3	відновл. залишку	+B	00	11001	
				1 00	00100	

C=0,1100

4.4.2 Ділення без відновлення залишків

Розглянутий спосіб ділення з відновленням залишків є аритмічним процесом зі змінним числом кроків того чи іншого виду в кожному конкретному випадку (3 кроку при 2Ri <B і 2 кроки при 2Ri> B). Для ритмізації процесу на кожну цифру частки необхідно затратити по 3 кроки, в результаті чого збільшується час виконання операції. Разом з тим, операцію можна спростити і отримати кожну цифру частки за 2 кроки.

Розглянемо випадок, коли Ri <0. У попередньому способі в цьому випадку виконувалися наступні операції.

Відновлення залишку:

$R'_i = 2\ R_i + |B| = 2\ R_{i-1} - |B| + |B| = 2\ R_{i-1}$

Зсув відновленого залишку вліво:

$\leftarrow R'_i = 2\ R'_i = 2\ R_{i-1} \times 2 = 4\ R_{i-1}$.

Віднімання модуля дільника з відновленого і зсунутого вліво залишку для визначення наступного залишку:

$R_{i+1} = 4\ R_{i-1} - |B|$

Якщо не відновлювати залишок, а відразу зсунути від'ємний Ri на один розряд вліво, то отримаємо

$R'_{i+1} = 2\ R_i = 2(2\ R_{i-1} - |B|) = 4R_{i-1} - 2\ |B|$.

Результат в даному випадку відрізняється від дійсного на величину + | B |. Тому в якості другого кроку необхідно провести корекцію результату на цю величину:

$R_{i+1} = 4\ R_{i-1} - 2|B| + |B = 4\ R_{i-1} - |B|$

В результаті отримуємо необхідну величину наступного залишку R_{i+1} , за 2 кроки .

Таким чином, щоб визначити чергову цифру частки, необхідно зсунути поточний залишок вліво на один розряд, а потім алгебраїчно додати до нього модуль дільника, якому приписується знак, протилежний знаку поточного залишку. Знак отриманого таким чином наступного залишку і визначає наступну цифру частки: якщо залишок додатний, то в частку записується 1, якщо від'ємний - записується 0. Операція зсувів і алгебраїчних додавань повторюється до тих пір, поки в частці не вийде необхідна кількість цифр.

Приклад. Дано A=0,101; B=0,110 $[-|B|]_{доп}$= 11,010; |B|= 00,110

11,110

11,010

1. Визначення знака частки: 0⊕0=0

2. Визначення модуля частки.

$[-|B|]_{доп}$= 11,010; |B|= 00,110

№ цикл	№ такт	Найменування операції	Дія			Розряди частки
0		Віднім. дільника із діленого	A	00	101	
			$[-B]_д$	11	010	
			R_0	11	111	0,
1	1	Зсув залишку	← R_0	11	110	
	2	Додавання	+B	00	110	
		формування розряду частки	R_1 1	00	100	1
2	1	Зсув залишку	← R_1	01	000	
	2	Віднім. дільника	$[-B]_д$	11	010	
		формування розряду частки	R_2 1	00	010	1
3	1	Зсув залишку	← R_2	00	100	
	2	Віднім. дільника	$[-B]_д$	11	010	
		формування розряду частки	R_3	11	110	0
4	1	Зсув залишку	← R_3	11	100	
	2	Додавання дільн.	+B	00	110	
		формування розряду частки		00	010	1

C=0,1101

В даний час у всіх ЕОМ ділення проводиться за способом без відновлення залишків . Це , по-перше , спрощує схему управління процесом ділення і , по-друге, збільшує швидкодію ЕОМ, так як тривалість операції ділення без відновлення залишків дорівнює мінімальній тривалості операції ділення з відновленням залишків.

При виконанні операції ділення результат вийде однаковим, якщо виконувати зсув залишків від ділення вліво або дільник вправо. Отже , можливі дві схеми виконання ділення:

1) ділення без відновлення залишків із зсувом дільника вправо;

2) ділення без відновлення залишків із зсувом залишку вліво.

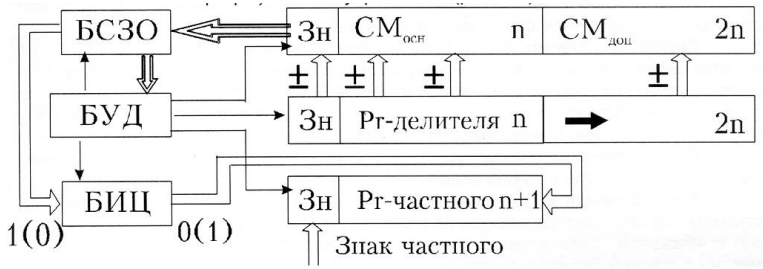


Рис. 4.4. Схема операційного пристрою для виконання операції ділення (1 варіант).

Передача в суматор дільника або доповнення модуля дільника забезпечується блоком управління діленням (БУД), який аналізує знакові цифри залишків, що знімаються з суматора за допомогою блоку знімання знаків залишків (БСЗО). Ці знакові цифри залишків, проходячи через блок інверсії цифр (БІЦ), інвертуються і подаються в молодший розряд зсувного регістра частки вже як цифри частки.

Для реалізації другого варіанту необхідні: n - розрядний регістр дільника; (n + 1) - розрядний регістр частки із зсувом вліво; n - або (n + 1) - розрядний суматор із зсувом вліво і схема управління. Суматор виробляє зсув поточного залишку вліво і алгебраїчне додавання його з дільником. Передача в СМ модуля дільника або його доповнення забезпечує БУД, який аналізує зсунуті з СМ знакові цифри залишків. Ці цифри інвертуються в БИЦ і подаються в молодший розряд Рг частки як цифри частки.

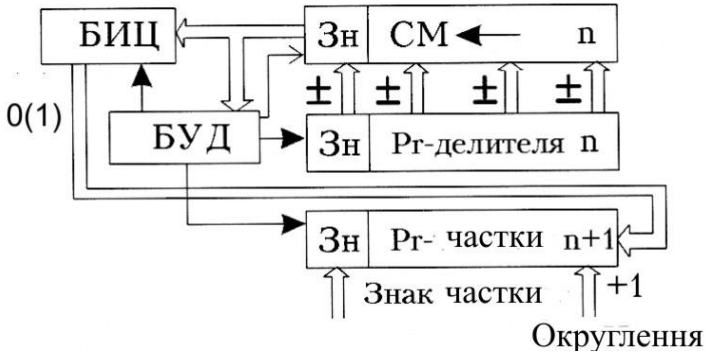


Рис. 4.5. Схема операційного пристрою для виконання операції ділення (2 варіант).

Аналіз обох схем показує, що другий варіант приблизно на 40 % економічніше по обладнанню в порівнянні з першим.

Вибір типу пристрою для виконання операції ділення при проектуванні машини зазвичай не є самостійною задачею. Тому на практиці спочатку за заданими технічними умовами вибирається схема пристрою для виконання операції множення внаслідок того, що множення є приблизно в 10 разів більш частою операцією. Після цього вибирається найбільш сумісна з пристроєм множення схема блоку для виконання операції ділення. Однак при проектуванні спеціалізованих ЕОМ може бути прийнятий інший порядок вибору структур окремих пристроїв.

Якщо порівнювати наведені схеми ділення зі схемами пристроїв для виконання операції множення, то виявляється, що схема першого варіанту ділення багато в чому збігається з четвертою схемою множення. Другий варіант схеми ділення добре сумісний з третьою схемою множення.

4.5 Способи прискореного ділення

Необхідність прискорення ділення слідує з наявності дуже ефективних методів прискорення множення. Способи прискореного ділення діляться на дві групи: для першої групи в кожному циклі формується одна або декілька цифр частки і новий залишок; друга група передбачає виконання ділення через множення або з використанням іншої процедури.

Суть одного із способів прискореного ділення першої групи полягає в тому, що в частку можна записати одразу послідовність однакових цифр (нулів або одиниць), якщо в результаті чергового кроку ділення отримано залишок за абсолютною величиною або досить малий, або близький до дільника. У першому випадку, якщо залишок має k нульових старших розрядів, то для визначення чергових цифр частки немає потреби віднімати дільник k раз із залишку. Необхідно в k чергових розрядів частки відразу записати нулі, зсунути залишок на k розрядів вліво, додати до нього алгебраїчно дільник і продовжити операцію ділення.

У другому випадку потрібно спочатку провести віднімання дільника із залишку, потім різницю зсунути на k розрядів вліво, після чого до отриманого числа додати дільник. При цьому в частку записується k одиниць.

4.6 Ділення чисел в машинах з плаваючою комою

Якщо числа A і B задані в нормальній формі, то їх частка дорівнюватиме:

$C = a:b = (a:b)\, 2^{(pa-pb)}$

де a і b - мантиси, а p_a і p_b-порядки відповідно чисел A і B. Звідси випливає, що операція ділення в машинах з плаваючою комою виконується в п'ять етапів.

1-й етап. Визначення знака частки шляхом додавання за модулем 2 знакових цифр мантис операндів.

2-й етап. Ділення модулів мантис операндів за правилами ділення чисел з фіксованою комою.

3-й етап. Визначення порядку частки шляхом віднімання порядку дільника з порядку діленого.

4-й этап. Нормалізація результату та його округлення.

5-й етап. Присвоєння знаку мантисі результату.

Два перші етапи повністю збігаються з етапами ділення чисел з фіксованою комою. Третій етап являє собою звичайне додавання в інверсних кодах.

При діленні нормалізованих чисел денормалізація результату можлива тільки вліво і тільки на один розряд. Це обумовлено тим, що мантиса будь-якого нормалізованого числа лежить в межах $2^{-1} \le |a| < 1 — 2^{-n}$.

Тоді найменша і найбільша можливі величини мантиси частки рівні відповідно

$$|y_{min}| = \frac{2^{-1}}{1-2^{-n}} > 2^{-1}; |y_{max}| = \frac{1-2^{-n}}{2^{-1}} = 2 - 2^{-(n-1)} < 2;$$

тобто мантиса частки лежить в межах $2^{-1} \le |y| < 2$.

Тому на четвертому етапі може виникнути необхідність нормалізації мантиси частки шляхом її зсуву вправо на один розряд і збільшення порядку частки на одиницю. Якщо ж перед діленням зсунути ділене на один розряд вправо, то на 4-му етапі може знадобитися нормалізація результатів вліво на 1 розряд.

При виконанні третього (визначення порядку) етапу порядок частки може виявитися більшим допустимого, тобто відбудеться переповнення розрядної сітки, яке розцінюється як аварійна ситуація. Може бути також отриманий порядок частки, який менше допустимого. Якщо на другому етапі обчислено частку $|y| > 1$, то на четвертому етапі при зсуві частки вправо її порядок повинен бути збільшений на одиницю. При цьому стає остаточно ясно, по-перше, чи виникає переповнення розрядної сітки порядку чи ні, по-друге, чи буде порядок частки менше допустимого значення. Якщо має місце зникнення порядку, то результат ділення за вказівкою програміста замінюється машинним нулем.

5 ТЕМА ДВІЙКОВО-ДЕСЯТКОВА АРИФМЕТИКА

5.1 Додавання в прямих Д-кодах

Десяткові цифри	Еквіваленти в кодах	
	Д1 8 4 2 1	Д2 ((8421) +3)
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

Інші комбінації в цих кодах заборонені (не використовуються)

ЗАБОРОНЕНІ	**1 0 1 0**	**0 0 0 0**
	1 0 1 1	**0 0 0 1**
	1 1 0 0	**0 0 1 0**
	1 1 0 1	**1 1 0 1**
	1 1 1 0	**1 1 1 0**
	1 1 1 1	**1 1 1 1**

У двійковій кодованому представленні десяткового числа кожна десяткова цифра зображується тетрадою двійкових символів $a_i = \alpha_4^i \alpha_3^i \alpha_2^i \alpha_1^i$, де a_i - десяткова цифра 1-го розряду; α_j^i - двійкова цифра і-ї тетради. Отриманий таким чином десятковий код, кодований двійковими символами, для стислості називають Д-кодом.

Є деякі множини Д-кодів. Вони обумовлюються наявністю всього 10 дозволених з 16 можливих комбінацій, які допускає тетрада. Наявність заборонених комбінацій в Д-кодах відрізняє їх від звичайних позиційних систем числення, в яких всі комбінації - дозволені. З усієї множини відомих Д-кодів найбільшого

поширення в обчислювальній техніці отримали код Д1 прямого заміщення (система 8421) і код Д2 з надлишком 3 (система 8421 +3).

Через заборонені комбінації, при додаванні чисел в будь-якому з Д-кодів виникає необхідність у корекції результату і труднощі у формуванні десяткового переносу в наступну тетраду. Особливості додавання чисел в кожному з Д-кодів різні, тому розглянемо їх окремо. При цьому будемо вважати, що задані числа $A=a_n a_{n-1}...a_1a_0$ и $B=b_n b_{n-1}...b_1b_0$, где a_i, b_i - двійково-кодовані десяткові цифри (тетради). Необхідно отримати: $A + B = C = c_n c_{n-1}...c_1c_0$, причому

$c_i=a_i+b_i+\Pi_{i-1}- \Pi_i p$; $c_{n+1} =\Pi_n$

де Π_i, $\Pi_{i-1}= \{0, 1\}$ — десяткові переноси;

$p = 10$- основа системи числення.

Так як найбільше десяткове однорозрядне число дорівнює 9, то з урахуванням переносу в даний розряд, значення результату порозрядного додавання лежить в межах від 0 до 19. При цьому одиниця в другому розряді являє собою десятковий перенос в наступну тетраду, а сума виходить в двійковому коді, відмінному від необхідного двійково-десяткового представлення, тобто вона вимагає корекції.

Код Д1

При представленні чисел в коді Д1 можуть виникнути такі випадки:

1. Якщо $a_i+b_i+\Pi_{i-1}<10$, то при виконанні дій над розрядами тетради за правилами двійкової арифметики відразу виходить правильний результат.

2. Якщо $a_i+b_i+\Pi_{i-1}\geq 10$, то виникає десятковий перенос. Тому сума в даній тетраді повинна дорівнювати:

$a_i+b_i+\Pi_{i-1} - \Pi_i$ x 10, де $\Pi_i = 1$.

При цьому ознакою неправильного результату є в одному випадку виникнення потетрадного переносу $\Pi_i'=16$, у другому - поява забороненої комбінації, якщо $15\geq a_i+b_i+\Pi_{i-1}\geq 10$. У будь-якому з цих випадків необхідно **скоригувати результат в даній тетраді введенням поправки +0110**, що призведе до виникнення потетрадного переносу і в другому випадку. Корекція обумовлена тим, що кожен перенос забирає із собою з даної тетради 16 одиниць, а приносить в наступну тільки 10 одиниць.

Приклад. Додати тетради $a_i = 1000$ і $b_i=1001$ при $\Pi_{i-1} = 1$.

$c_i'=a_i+b_i+\Pi_{i-1}= 1\ 0010$.

Так як $\Pi_i = 1$, потрібна корекція результату

$c_i = 0010 +0110 = 1000$: $\Pi_i = \Pi_i' = 1$.

Результат =1 1000

Приклад. Додати a_i = 1000 і b_i= 0111 при $П_{i-1}$ = 1

c_i'=a_i+b_i+$П_{i-1}$=1111.

Так як величина c_i'= 1111 належить до заборонених комбінацій, то необхідно ввести поправку виду 0110:

c_i= 1111+0110= 1 0101, т. е. $П_i$= 1, c_i = 0101.

Загальне правило: якщо в і-й тетраді сума цифр з переносом з (і - 1)-ї тетради менше 10, то додавання проводиться без поправок, якщо ж сума цифр з урахуванням переносу дорівнює або більше 10, то проводиться корекція результату тетради введенням поправки +0110, а якщо при цьому виникає одиниця переносу, вона додається до вмісту (і + 1) -ї тетради (**потетрадний перенос не блокується, а враховується).**

При цьому, якщо в декількох тетрадах, починаючи з (і + 1) -ї, розрядна сума дорівнює 1001, то перенос призведе до формування забороненої комбінації в (і + 1) -й тетраді. В результаті цього буде потрібна корекція, яка призведе до забороненої комбінації в (і +2)-й тетраді і т. д. Отже, із-за послідовного поширення потетрадних переносів час додавання в коді Д1 складе в гіршому випадку n тактів, де n - кількість тетрад. Зазвичай схеми будують таким чином, щоб перенос, що виникає при додаванні потетрадної поправки, проходив крізь тетради, в яких попередня сума дорівнює 9_{10} = 1001_2 і скидав їх в 0 . При цьому сума завжди формується за два такти.

Приклад. Додати числа

$A = 248_{10}$ = 0010 0100 1000 і $B = 349_{10}$ = 0011 0100 1001

	0	0	1	0		0	1	0	0		1	0	0	0	
+	0	0	1	1		0	1	0	0		1	0	0	1	
	0	1	0	1		1	0	0	1	←	0	0	0	1	
+											0	1	1	0	
	0	1	0	1		1	0	0	1		0	1	1	1	$=597_{10}=248_{10}+348_{10}$

Стрілкою показані одиниці потетрадних переносів.

Код Д2

При додаванні чисел в коді Д2 можливі наступні випадки з урахуванням того, що a_i'=a_i +3; b_i' =b_i + 3, де a_i' і b_i' — тетради для коду Д2.

1. Якщо a_i'+b_i'+$П_{i-1}$≤15, то a_i'+b_i'+$П_{i-1}$=Сі+6= C_i'+3. (**без переносу 1101**)

тобто результат необхідно скоригувати на величину (-0011).

Суматори не виробляють операцію віднімання, тому поправка $(-3)_{10}= (-0011)_2$ здійснюється додаванням в дану тетраду числа $(13)_{10}$**=(1101)$_2$** і блокуванням переносу, що виникає при цьому з даної тетради в старшу, і який має значення $(16)_{10}$ - $(10000)_2$:

(— 3 =+13-16)$_{10}$ або (-0011=+1101-10000)$_2$.

2. Якщо ж a_i'+b_i'+$П_{i-1}$> 15, то $a_i+b_i+П_{i-1}= C_i$'-3. Тут виникає десятковий **перенос**, який при переході в старшу тетраду змінює свою вагу з 16 на 10. Тому в цьому випадку потрібна поправка результату на величину +**0011**.

Приклад. Задані в коді Д2 А=46= 0111 1001; В=37= 0110 1010.

Найдемо суму

	0	1	1	1	1	0	0	1		
+	0	1	1	0	1	0	1	0		
	1	1	1	0	0	0	1	1		
+	1	1	0	1	0	0	1	1		
1	1	0	1	1	0	1	1	0	=	83
←	не реалізується									

При додаванні в коді Д2 не виникає проблеми наскрізного переносу, і операція додавання виконується в два такти. Це пояснюється тим, що якщо в декількох тетрадах сума до додавання поправки дорівнює $9_{10} = 1111_2$, то при надходженні переносу з (і- 1) -й тетради в і- у , її вміст скинеться в 0 і перенос піде в (і +1) - у тетраду в цьому ж такті додавання. У і- у ж тетраду в другому такті додавання буде додана поправка +0011 .

Тому корекція результату здійснюється потетрадно з блокуванням ланцюгів потетрадного переносу. Таким чином, в коді Д2 завжди проводиться корекція проміжного результату, отриманого шляхом додавання цифр доданків за правилами двійковії арифметики. При цьому, якщо при додаванні і- х тетрад не виникає перенос, тобто $П_i = 0$, то поправка дорівнює -0011 . Якщо ж виникає потетрадний перенося $П_i = 1$, то поправка в і -й тетраді дорівнює +0011.

5.2 Додавання чисел в інверсних Д-кодах

Д-коди можуть бути представлені в розрядній сітці машини у формі або з фіксованою, або з плаваючою комою. При цьому від'ємні числа можуть бути представлені у прямому, оберненому і додатковому кодах. Тому, якщо

А=-0, $a_1a_2...a_n$, де a_i — тетради, то

$[A]_{пр}$= 1, $a_1a_2...a_n$; $[A]_о$= 1, $\overline{\overline{a}}_1\ \overline{\overline{a}}_2 ... \overline{\overline{a}}_n$; $[A]_д$= 1, $\overline{\overline{a}}_1\ \overline{\overline{a}}_2 ... \overline{a}_n$;

де $\overline{\overline{a}}_i$ - доповнення до 9 у всіх тетрадах; $\overline{a}_n$ - доповнення до 10 в молодшій тетраді.

Отже, $\overline{a}_n + a_i = p = 10$; $\overline{\overline{a}}_n + a_i = p-1 = 9$. **Обернений код Д2** виходить простим інвертуванням цифр тетрад.

Приклад. Знайти обернений і додатковий коди числа А =-0,136 для коду Д2.

$A_{Д2}$=1,0100 0110 1001; $[A_{Д2}]_о$=1,1011 1001 0110.

На підставі співвідношення отримаємо додатковий код числа *А:*

$[A_{Д2}]_д$=1,1011 1001 0111.

У коді Д1 пряме інвертування цифр тетрад означає отримання доповнення до 2^4-1 = 15. **Тому для отримання оберненого коду числа в коді Д1 в усі тетради числа додають спочатку +0110, після чого проводиться інвертування цифр тетрад.**

При додаванні в оберненому коді одиниця переносу зі знакового розряду додається в молодший розряд молодшої тетради результату, а в додатковому - блокується.

Відзначимо , що додавання 1 в молодший розряд при отриманні додаткового Д2 коду проводиться набагато складніше, ніж в коді Д1, так як ця операція виконується як звичайне додавання в Д2 коді. При додаванні в інверсних Д- кодах, як і при додаванні в прямих Д- кодах, необхідна корекція результату, яка може здійснюватися або програмними, або апаратними засобам. На практиці зазвичай застосовують другий спосіб. При цьому в процесі виконання корекції ланцюги міжтетрадного переносу в суматорах блокуються для коду Д2, а при додаванні в коді Д1 не блокуються.

Таким чином, код Д2, тобто код з надлишком 3, дозволяє досить просто формувати порозрядні доповнення до 9 і також просто виконувати алгебраїчне додавання. Однак, як і код Д1, він вимагає корекції результату, що збільшує час реалізації операцій. Код Д2 володіє ще рядом корисних властивостей. Наприклад, цифри десяткової системи числення, значення яких більше або рівні п'яти, відрізняються тим, що в старшому розряді двійково-десяткового коду завжди присутня одиниця. Завдяки цій властивості за значенням тільки одного старшого розряду тетради визначається необхідність округлення.

5.3 Зсув Д-кодів

При зсуві Д- коду числа на один десятковий розряд необхідно передбачити зсув двійкових цифр відразу на 4 розряди. Правила зсуву залишаються тими ж, що і

для двоїчних дробів, тобто зсув вправо обмежується величиною допустимої помилки, яку можна зменшити шляхом округлення зсунутого числа. Для цього вводять додатковий десятковий розряд (ДДР). При цьому додатні дроби округлюють додаванням числа 5 в ДДР, якщо цифра в ньому більше або дорівнює 5. Округлення від'ємного дробу, представленого в обернених Д- кодах, не проводиться. Округлення в додаткових Д- кодах проводиться тільки тоді, коли хвіст числа більше $0{,}5 \times 10^{-n}$, де n - число основних десяткових розрядів. Операція зсуву Д- коду вліво коректна до тих пір, поки не відбудеться переповнення розрядної сітки.

При виконанні операцій множення і ділення Д- кодів нерідко виникає необхідність множення або ділення чисел на 2. Для множення Д1 коду на 2 необхідно зсунути задане число на один двійковий розряд вліво. Поправки вносяться як і при додаванні в коді Д1 , тобто якщо в 1 -й тетраді в результаті зсуву виникла заборонена комбінація або з і-й тетради в (і + 1) - у була висунута одиниця, то до і-ї тетради додається поправка 0110 і ланцюги міжтетрадного переносів не блокують. В інших випадках корекція результату не проводиться.

При діленні коду Д1 на 2 необхідно зсунути його на один двійковий розряд вправо і відняти поправку 0011 з тих тетрад, в які перейшла одиниця з попередньої тетради або додати поправку 1101 і заблокувати міжтетрадний перенос, який при цьому виникає.

При множенні коду Д2 на 2 необхідно зсунути його на один двійковий розряд вліво. Корекція вноситься як і при додаванні, тобто якщо з і-ї тетради зсувається одиниця в (і + 1) - у , то до і-ї тетради додається 0011, якщо ж одиниця в (і +1) - у тетраду НЕ зсувається, то до 1-ї тетради додається 1101. При цьому ланцюги міжтетрадних переносів блокуються.

5.4 Особливості множення і ділення Д-кодів

Множення в Д- кодах зводиться до послідовного додавання часткових добутків, одержуваних при множенні множимого на чергову цифру множника. При цьому множення супроводжується розшифровкою значення чергової і -ї тетради множника і зсувом множника на 4 розряду відразу.

Найпростішим прийомом отримання чергового часткового добутку є послідовне віднімання одиниці із значення тетради до отримання нуля і додавання множимого в суматор на кожному такті. На практиці використовується один спосіб множення - множення молодшими розрядами множника із зсувом суми часткових добутків вправо.

Ділення починається з віднімання дільника з діленого на нульовому кроці і з зсувом залишків - на наступних кроках. Віднімання на кожному кроці замінюється додаванням в додатковому Д- коді і проводиться доти, поки не вийде від'ємний результат. При цьому кожен раз при отриманні додатного залишку додається одиниця в спеціальний лічильник, де накопичується чергова цифра частки.

Потім здійснюється зсув залишку на 4 двійкових розряди вліво до тих пір, поки не вийде залишок > 0. Кількість додавань (без останнього) є доповненням відповідної цифри частки до 9, що заноситься в лічильник чергової цифри частки.

6 ТЕМА. БУЛЕВІ ФУНКЦІЇ

6.1 Основні поняття булевої алгебри

Технічні питання, пов'язані з побудовою логічних схем ЕОМ, можна вирішити за допомогою математичного апарату, об'єктом дослідження якого є функції, що приймають, так само як і їхні аргументи, тільки два значення - "0" і "1".

Таким апаратом є математична логіка (алгебра логіки, булева алгебра).

Логіка - це наука про закони і форми мислення.

Математична логіка займається вивченням можливостей застосування формальних методів для вирішення логічних завдань. Один з розділів математичної логіки є алгебра логіки.

Основне поняття алгебри логіки - висловлювання. Висловлення - це деякий вислів, про який можна стверджувати, що він істинний або хибний.

Будь-яке висловлювання можна позначити символом x і вважати, що x = 1, якщо висловлювання істинно, а x = 0 - якщо висловлювання помилкове. Істинному висловлюванню відповідає твердження - "Так", помилковому висловлюванню - твердження - "Ні".

Логічна (булева) змінна - така величина x, яка може приймати тільки два значення x = {0,1}.

Висловлення абсолютно істинне, якщо відповідна йому логічна величина приймає значення x = 1 при будь-яких умовах.

Висловлення абсолютно помилкове, якщо відповідна йому логічна величина приймає значення x = 0 при будь-яких умовах.

Функція f, що залежить від n змінних $x_1,x_2,...,x_n$, називається булевою, або перемикальною, якщо функція f і будь-який з її аргументів $x_i, i \in \overline{1,n}$ приймають значення тільки з множини {0,1}. Аргументи булевої функції також називаються булевими.

6.2 Способи завдання булевих функцій

Довільна булева функція задається одним з трьох способів: матричним (табличним), геометричним і аналітичним.

При матричному способі булева функція $f(x_1,...,x_n)$ задається таблицею істинності (табл. 6.1 та 6.2), в лівій частині якої представляються всі можливі двійкові набори довжини n, а в правій вказується значення функції на цих наборах.

Під двійковим набором розуміється сукупність значень аргументів $x_1,x_2,...,x_n$ булевої функції f. Двійковий набір має довжину n, якщо він представлений n цифрами з множини {0,1}. У табл. 6.1 та 6.2 перераховані всі двійкові набори відповідно довжини 3 і 4.

Таблиця 6.1

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

Таблиця 6.2

Номер набору	$f(x_1,x_2,x_3)$
0	0
1	1
2	0
3	0
4	1
5	1
6	0
7	1

Таблиця 6.3

x1	x2	x3	x4	$f(x_1..x_4)$
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

Таблиця 6.4.

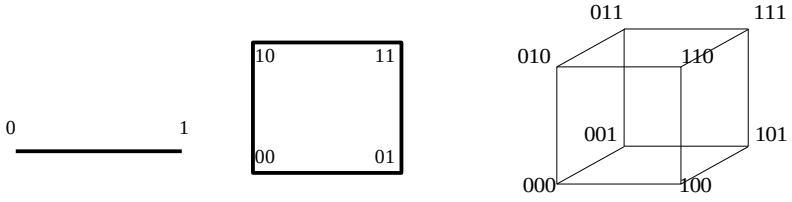
$x_1,x_2,...,x_{j-1}$	$x_j,x_{j+1},...,x_n$			
	00..0	0...1	...	11..1
00...0	0		...	
00...1	1		... $f(x_1...x_n)$	
...	...	...	...	...
11...1				

Іноді двійкові набори в таблиці істинності булевої функції зручно представляти номерами наборів. Запишемо аргументи $x_1,x_2,...,x_n$ в порядку зростання їх індексів. Тоді будь-який двійковий набір можна розглядати як ціле двійкове число N, зване номером набору. Наприклад, двійкові набори 0101 і 1000 мають номера 5 і 8 відповідно. Очевидно, будь-яка булева функція може бути задана таблицею істинності, в якій двійкові набори замінені своїми номерами (табл.6.2).

Булеві функції, що залежать від великого числа змінних, задавати таблицею істинності незручно в силу її громіздкості. Наприклад, таблиця істинності булевої функції 8 змінних буде містити $2^8 = 256$ рядків. Тому для завдання функцій багатьох змінних зручно використовувати модифікацію таблиці істинності.

Розглянемо спосіб побудови такої таблиці істинності для функції n змінних. Множина з n змінних функції розбивається на дві підмножини $x_1,x_2,...,x_{j-1}$ и $x_j, x_j,x_{j+1},...,x_n$. Змінними $x_1,x_2,...,x_{j-1}$ відзначають рядки таблиці істинності, задаючи в кожному рядку значення відповідного двійкового набору довжини j-1. Змінними $x_j,x_{j+1},...,x_n$ відзначають її стовпці, задаючи в кожному стовпці значення відповідного двійкового набору довжини n-j +1. Значення функції записується в клітинці на перетині відповідного рядка і стовпчика (табл.6.4.).

При геометричному способі булева функція $f(x_1,..., x_n)$ задається за допомогою n-мірного куба. У геометричному сенсі кожен двійковий набір є n-мірний вектор, який визначає точку n-мірного простору. Виходячи з цього, вся множина наборів, на яких визначена функція n змінних, представляється вершинами n-мірного куба. Відзначаючи точками вершини куба, в яких функція приймає одиничні (або нульові) значення, отримаємо геометричне представлення функції. Наприклад, булева функція, задана табл.6.1, геометрично представляється 3-мірним кубом (рис. 6.1.В).



а) n=1 б) n=2 в) n=3

Рис. 6.1- Геометричне завдання булевої функції: а) однієї змінної: б) двох змінних; в) трьох змінних.

При аналітичному способі булева функція задається формулами, тобто аналітичними виразами, побудованими на основі операцій булевої алгебри. Аналітичний спосіб завдання булевих функцій займає особливе місце в проектуванні цифрових автоматів. Фактично, всі перетворення над булевими функціями, необхідні для побудови цифрових автоматів, ведуться на аналітичному рівні.

Розглянемо області визначення булевих функцій. Між двійковими наборами і двійковими числами існує взаємно однозначна відповідність. Отже, існує 2^n різних наборів двоїчних змінних.

Таким чином, областю визначення булевої функції n змінних при матричному способі завдання є множина всіх можливих двійкових наборів довжини n, а при геометричному способі завдання - множина всіх вершин n-мірного одиничного куба.

Булеву функцію, визначену на всіх своїх наборах, називають повністю визначеною.

Булеву функцію n змінних називають неповністю визначеною або частковою, якщо вона визначена не на всіх двійкових наборах довжини n.

6.3 Булеві функції однієї і двох змінних

Розглянемо найбільш вживані булеві функції однієї і двох змінних. Функції однієї змінної представлені в табл. 6.5.

Таблиця 6.5.

f(x)	x		Ум. обозн	Найменування
	0	1		
$f_0(x)$	0	0	0	тотожний нуль (константа 0);
$f_1(x)$	0	1	x	тотожна функція
$f_2(x)$	1	0	$\overline{x}$	заперечення x (інверсія);
$f_3(x)$	1	1	1	тотожна одиниця (константа 1).

Функції двох змінних представлені в табл. 6.6.

Розглянуті найпростіші булеві функції дозволяють будувати нові булеві функції за допомогою узагальненої операції, званої операцією суперпозиції. Фактично операція суперпозиції полягає в підстановці замість аргументів інших

булевих функцій (зокрема аргументів). Відзначимо, що реально елементарній функції відповідає реалізуючий її елемент, а суперпозиції булевих функцій відповідає з'єднання елементів.

Таблиця 6.6.

	x_1	0	0	1	1	Найменування функції
	x_2	0	1	0	1	
$f_0(x_1,x_2)$		0	0	0	0	константа 0
$f_1(x_1,x_2)$		0	0	0	1	кон'юнкція
$f_2(x_1,x_2)$		0	0	1	0	заборона по x2
$f_3(x_1,x_2)$		0	0	1	1	змінна x1
$f_4(x_1,x_2)$		0	1	0	0	заборона по x1
$f_5(x_1,x_2)$		0	1	0	1	змінна x2
$f_6(x_1,x_2)$		0	1	1	0	сума по модулю 2
$f_7(x_1,x_2)$		0	1	1	1	диз'юнкція
$f_8(x_1,x_2)$		1	0	0	0	операція Пірса (ф-ція Вебба)
$f_9(x_1,x_2)$		1	0	0	1	логічна рівнозначність
$f_{10}(x_1,x_2)$		1	0	1	0	інверсія x2
$f_{11}(x_1,x_2)$		1	0	1	1	імплікація від x2 до x1
$f_{12}(x_1,x_2)$		1	1	0	0	інверсія x1
$f_{13}(x_1,x_2)$		1	1	0	1	імплікація від x1 до x2
$f_{14}(x_1,x_2)$		1	1	1	0	штрих Шефера
$f_{15}(x_1,x_2)$		1	1	1	1	константа 1

Суперпозиція булевих функцій представляється у вигляді логічних формул . Однак слід зазначити:

- одна і та ж функція може бути представлена різними формулами ;

- кожній формулі відповідає своя суперпозиція і, отже, своя схема з'єднань елементів;

- між формулами представлення булевих функцій і схемами, що їх реалізують, існує взаємнооднозначна відповідність.

Очевидно, серед схем, що реалізують цю функцію, є найбільш проста. Пошук логічної формули, відповідної цій схемі, представляє великий практичний інтерес, а перетворення формул булевих функцій ґрунтується на використанні співвідношень булевої алгебри.

6.4 Основні закони і тотожності булевої алгебри

Для булевої алгебри визначені одна одномісна (унарна) операція "заперечення" (інверсія) і дві двомісні (бінарні) операції кон'юнкція і диз'юнкція (позначаються «∧», «∨» відповідно). Наведемо деякі основні формули:

Формули диз'юнкції:

$0 \vee 0 = 0$ $\quad$ $0 \vee x = x$

$0 \vee 1 = 1$ $\quad$ $1 \vee x = 1$

$1 \vee 0 = 1$ $\quad$ $x \vee x \vee x \vee \ldots \vee x = x$

$1 \vee 1 = 1$ $\quad$ $x \vee \bar{x} = 1$

$x \vee x = x$

Формули кон'юнкції:

$0 \wedge 0 = 0$ $\quad$ $0 \wedge x = 0$

$0 \wedge 1 = 0$ $\quad$ $1 \wedge x = x$

$1 \wedge 0 = 0$ $\quad$ $x \wedge x \wedge x \wedge \ldots \wedge x = x$

$1 \wedge 1 = 1$ $\quad$ $x \wedge \bar{x} = 0$

$x \wedge x = x$

Формули інверсії:

$\bar{0} = 1$ $\quad$ $\bar{\bar{x}} = x$

$\bar{1} = 0$

Формули суми по модулю 2 (mod 2, або $\oplus$):

$0 \oplus 0 = 0$ $\quad$ $x \oplus x = 0$

$0 \oplus 1 = 1$ $\quad$ $1 \oplus x = \bar{x}$

$1 \oplus 0 = 1$ $\quad$ $x \oplus \bar{x} = 1$

$1 \oplus 1 = 0$ $\quad$ $x \oplus x \oplus x \ldots \oplus x = \begin{cases} x & \text{при n непарн.} \\ 0 & \text{при n парн.} \end{cases}$ (n)

$0 \oplus x = x$

У цій алгебрі справедливі наступні аксіоми:

$x \vee y = y \vee x;$ $\quad x \wedge y = y \wedge x$	закон комутативності (перестановок)
$x \vee y \vee z = (x \vee y) \vee z$	закон асоціативності (сполучний) для диз'юнкції
$x \wedge y \wedge z = (x \wedge y) \wedge z$	закон асоціативності для кон'юнкції
$(x \vee y) \wedge z = x \wedge z \vee y \wedge z$	закон дистрибутивності (розподільний) для кон'юнкції відносно диз'юнкції
$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$	закон дистрибутивності для диз'юнкції відносно кон'юнкції

Під бінарною операцією на множині *A,* в загальному випадку розуміють відображення декартового добутку множин (А х *A)* в множину А. Іншими словами, результат застосування бінарної операції до будь-якої впорядкованої пари елементів з множини А є також елемент з множини А.

Під унарною операцією на множині А розуміють виділення (фіксацію) якого-небудь елемента множини А.

Перетворення формул булевих функцій із застосуванням тільки аксіом булевої алгебри малоефективне. Для спрощення формул використовується цілий ряд співвідношень. Наведемо деякі з них.

З комутативності і асоціативності диз'юнкції слідує, що диз'юнкція кількох змінних може виконуватися послідовно, причому порядок взяття диз'юнкції не впливає на результат.

Формули де Моргана:

$$\overline{x \vee y} = \bar{x} \wedge \bar{y} \qquad \overline{x \wedge y} = \bar{x} \vee \bar{y}$$

Наслідки з формул де Моргана:

$$\overline{\overline{x \wedge y}} = \overline{\bar{x} \vee \bar{y}} \qquad \overline{\overline{x \vee y}} = \overline{\bar{x} \wedge \bar{y}}$$

Формули для системи функцій:

операція поглинання (х поглинає у): $x \vee xy = x$; або $x(x \vee y) = x$.

операція склеювання: $xy \vee x\bar{y} = x$, або $(x \vee y) \wedge (x \vee \bar{y}) = x$

6.5 Аналітичне представлення булевих функцій

Розглянемо універсальні (канонічні) форми представлення, що дають можливість отримати аналітичну форму безпосередньо по таблиці істинності для довільної булевої функції. Ця форма в подальшому може бути спрощена. Оскільки між множиною аналітичних представлень і множиною схем, що реалізують булеву функцію, існує взаємно однозначна відповідність, відшукання канонічної форми представлення булевої функції є початковим етапом синтезу схеми, яка її реалізує. Найбільш широке впровадження набула досконала диз'юнктивна нормальна форма ДДНФ (*совершенная дизъюнктивная нормальная форма* - СДНФ). Перш ніж перейти до її вивчення, наведемо визначення конституенти одиниці - поняття, яким будемо широко користуватися надалі.

Конституентою одиниці (кон'юнктивним термом) називається функція $f(x_1,x_2,...,x_n)$, яка приймає значення 1 тільки на одному єдиному наборі.

Конституента одиниці записується як логічний добуток *n* різних булевих змінних, деякі з них можуть бути з запереченнями. Можна легко виразити будь-яку

булеву функцію як диз'юнкцію конституент одиниці, відповідних тим наборам, на яких функція приймає значення 1.

Диз'юнкція констітуент 1, рівних 1 на тих же наборах, що і задана функція, називається досконалою диз'юнктивною нормальною формою перемикальної функції (ДДНФ).

Набори, на яких функція f приймає значення 1, часто називаються одиничними, інші - нульовими наборами. Виписувати в ДДНФ має сенс тільки конституенти одиниці, що відповідають одиничним наборам.

Приклад. Випишемо ДДНФ для функції, заданої таблицею істинності (табл. 6.7.).

Таблиця 6.7.

	$x_1x_2x_3$	$f(x_1,x_2,x_3)$				
0	0 0 0	0	0			
1	0 0 1	1	1			
2	0 1 0	1	0	1		
3	0 1 1	0	0			
4	1 0 0	0	0			
5	1 0 1	1	0		1	
6	1 1 0	0	0			
7	1 1 1	1	0			1

$$f_{СДНФ}(x_1,x_2,x_3)= \overline{X}_1\overline{X}_2X_3 \vee \overline{X}_1X_2\overline{X}_3 \vee X_1\overline{X}_2X_3 \vee X_1X_2X_3$$

$$f_{СКНФ}(x_1,x_2,x_3)=(X_1 \vee X_2 \vee X_3)\wedge(X_1 \vee \overline{X}_2 \vee \overline{X}_3)\wedge$$
$$(\overline{X}_1 \vee X_2 \vee X_3)\wedge(\overline{X}_1 \vee \overline{X}_2 \vee X_3)$$

Інша відома форма носить назву досконалої кон'юнктивної нормальної форми ДКНФ (*совершенная конъюнктивная нормальная форма* - СКНФ). Вона будується аналогічно ДДНФ.

Визначення. Конституентою нуля (диз'юнктивним термом) називається функція, що приймає значення 0 на єдиному наборі.

Конституента нуля записується у вигляді елементарної диз'юнкції всіх змінних. Кожному набору відповідає своя конституента 0. Наприклад, набору 0110 змінних x_1,x_2,x_3,x_4 відповідає конституента нуля $(X_1 \vee \overline{X}_2 \vee \overline{X}_3 \vee X_4)$.

ДКНФ представляється як кон'юнкція конституент нуля, що відповідають нульовим наборам функції.

6.6 Функціонально повні системи булевих функцій

Будь яка булева функція може бути представлена аналітично однієї з нормальних форм (ДДНФ, ДКНФ). Останні використовують обмежену кількість елементарних булевих функцій. Наприклад, для ДДНФ такими функціями є «кон'юнкція», «диз'юнкція» і «заперечення». Існують системи булевих функцій, за допомогою яких можна аналітично представити будь-яку як завгодно складну булеву функцію. Проектування цифрових автоматів засноване на знанні таких систем булевих функцій. Останнє особливо важливо для розробки комплектів інтегральних мікросхем, з яких можна побудувати довільний цифровий автомат. Проблема функціональної повноти є центральною проблемою функціональних побудов в алгебрі логіки.

Визначення: Функціонально повною системою булевих функцій (ФПСБФ) називається сукупність таких булевих функцій $\{f_1, f_2,..., f_k\}$, що довільна булева функція f може бути записана у вигляді формули через функції цієї сукупності.

До ФПСБФ слід віднести системи: $\{ \wedge , \vee , НЕ\}$, $\{ \wedge , \oplus , НЕ\}$.

Визначення властивостей ФПСБФ ґрунтується на понятті замкнутого щодо операції суперпозиції класу функцій. Клас булевих функцій, функціонально замкнутий по операції суперпозиції, є множина функцій, будь-яка суперпозиція яких дає функцію, яка також належить цій множині. Серед функціонально замкнутих класів виділяють класи особливого типу, звані предповними, які мають наступну властивість. Предповний клас S не збігається з множиною P_2 всіх можливих булевих функцій, однак, якщо в нього включити будь-яку, із тих, що не входить у S, булеву функцію, то новий функціонально замкнутий клас буде збігатися з множиною P_2. Проведені дослідження показали, що предповних класів п'ять, а для побудови ФПСБФ необхідно і достатньо, щоб її функції не містилися повністю ні в одному з п'яти предповних класів. Перерахуємо предповні класи булевих функцій:

1) булеві функції, що зберігають константу 0;
2) булеві функції, що зберігають константу 1;
3) самодвойственні булеві функції;
4) лінійні булеві функції;
5) монотонні булеві функції.

Визначення. До булевих функцій, що зберігає константу 0, відносять такі булеві функції $f(x_1,x_2,...,x_n)$, для яких справедливе співвідношення $f(0,..., 0)=0$.

Прикладами булевих функцій, що зберігають константу 0, є функції $f_0 – f_7$ (табл. 6.6.).

Визначення. До булевих функцій, що зберігає константу 1, відносять такі булеві функції $f(x_1,x_2,...,x_n)$, для яких справедливе співвідношення $f(1,1, ..., 1)=1$.

Прикладами булевих функцій, що зберігають константу 1, є функції f_1, f_3, f_5, f_7 (табл. 6.6).

Перш ніж ввести поняття класу самодвойственних функцій, дамо наступне визначення.

Визначення. Булеві функції $f_1(x_1,x_2,...,x_n)$ і $f_2(x_1,x_2,...,x_n)$, називаються двойственними одна одній, якщо виконується співвідношення $f_1(x_1,x_2,...,x_n) = \overline{f_2(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)}$

Двойственними є функції з табл. 6.6 - f_0 и f_{15} , f_1 и f_7 и т. д.

Визначення. До самодвойственних булевих функцій відносять такі булеві функції, які двойственні по відношенню до самих себе, тобто справедливе співвідношення $f_1(x_1,x_2,...,x_n) = \overline{f_2(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)}$.

Якщо домовитися називати протилежними наборами набір $\langle \gamma_1, \gamma_2, \ldots \gamma_n \rangle$ і набор $\langle \bar{\gamma}_1, \bar{\gamma}_2, \ldots \bar{\gamma}_n \rangle$, то визначення самодвойственних функцій дамо таке.

Визначення. Булева функція називається самодвойственною, якщо на будь-яких двох протилежних наборах вона приймає протилежні значення.

Самодвойственними є функції f_3, f_5, f_{10}, f_{12} з табл. 6.6. З визначення самодвойственної функції випливає, що вона повністю визначається своїми значеннями на першій половині рядків таблиці істинності.

Визначення. До лінійних булевих функцій відносять такі булеві функції, які представимо у вигляді

$$f(x_1,x_2,...,x_n)=c_0 \oplus c_1x_1 \oplus ... \oplus c_nx_n,$$

де $c_i \in \{0, 1\}$, а $\oplus$ - операція «сума по mod 2».

Лінійними є булеві функції з табл. 6.6 - f_0, f_3 , f_5 і т. д.

Перш ніж ввести поняття класу монотонних булевих функцій, дамо наступне визначення.

Визначення. Двійковий набір $\alpha = \langle \alpha_1, \alpha_2, \ldots, \alpha_n \rangle$ не менш двікового набору $\beta = \langle \beta_1, \beta_2, \ldots, \beta_n \rangle$, (тобто $\alpha \geq \beta$), якщо для кожної пари (α_i, β_i) $i \in \overline{1,n}$ справедливе співвідношення $\alpha_i \geq \beta_i$.

Так, набор 1011 ≥1010. Разом з тим набори 1011 і 0100 непорівнянні в тому сенсі, що для них не виконується ні співвідношення $\alpha \geq \beta$, ні $\beta \geq \alpha$.

Визначення. Булева функція $f(x_1,x_2,...,x_n)$ називається монотонною, якщо для будь-яких двох наборів $\alpha = \langle \alpha_1, \alpha_2, \ldots, \alpha_n \rangle$ і $\beta = \langle \beta_1, \beta_2, \ldots, \beta_n \rangle$ таких, що $\alpha \geq \beta$ має місце нерівність $f(\alpha_1, \alpha_2, \ldots, \alpha_n) \geq f(\beta_1, \beta_2, \ldots, \beta_n)$

Монотонними є булеві функції f_0, f_1,f_3, f_5 (з табл. 6.6) і т.д.

Наведемо без доведення два формулювання теореми про функціональну повноту.

Теорема. Для того щоб система S булевих функцій була функціонально повною, необхідно і достатньо, щоб ця система містила хоча б одну булеву функцію, яка не зберігає константу 0, хоча б одну булеву функцію, яка не зберігає константу 1, хоча б одну несамодвойственну булеву функцію, хоча б одну нелінійну булеву функцію і хоча б одну немонотонну булеву функцію.

Система булевих функцій є функціонально повною тоді і тільки тоді, коли вона цілком не міститься ні в одному з предповних класів. За допомогою функціонально повної системи можна реалізувати булеву функцію будь-якого ступеня складності.

Розглянемо приклади ФПСБФ. До таких ФПСБФ, найбільш поширеним в практиці побудови цифрових автоматів, слід віднести: ($\wedge$,$\vee$, НЕ); ($\wedge$, $\oplus$,НЕ); ($\vee$, НЕ); ($\wedge$, НЕ), ($\wedge$, $\oplus$,1); де символами $\vee$, $\wedge$, $\oplus$, НЕ, 1 позначені булеві функції: «диз'юнкція», «кон'юнкція», «сума по mod 2», «заперечення», «константа 1», відповідно.

6.7 Мінімізація булевих функцій

При проектуванні цифрових автоматів широко використовуються методи мінімізації булевих функцій, що дозволяють отримувати рекомендації для побудови економічних схем цифрових автоматів. Загальна задача мінімізації булевих функцій може бути сформульована таким чином: *знайти аналітичний вираз заданої булевої функції в формі, що містить мінімально можливе число букв*. Слід зазначити, що в

загальній постановці дана задача поки не вирішена, проте досить добре досліджена в класі диз'юнктивно-кон'юнктивних нормальних форм.

Визначення. Елементарною кон'юнкцією називається кон'юнкція кінцевого числа різних між собою булевих змінних, кожна з яких може мати або не мати заперечення.

Визначення. Диз'юнктивною нормальною формою (ДНФ) називається диз'юнкція елементарних кон'юнкцій.

Визначення. Мінімальною диз'юнктивною нормальною формою булевої функції називається ДНФ, що містить найменше число букв (по відношенню до всіх інших ДНФ, що представляють задану булеву функцію).

Визначення. Булева функція $g(x_1,x_2,...,x_n)$ називається імплікантою булевої функції $f_1(x_1,x_2,...,x_n)$, якщо для будь-якого набору змінних, на якому g = 1, справедливо $f = 1$.

Приклад. Булева функція f задана таблицею 6.8. Там же наведені всі її імпліканти. При запису функції та її імплікант в СДНФ маємо:

Таблиця 6.8.

x_1	x_2	x_3	f	g_1	g_2	g_3	g_4	g_5	g_6	g_7
0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0
1	1	0	1	0	1	1	0	0	1	1
1	1	1	1	1	0	1	0	1	0	1

$f = \overline{X}_1 X_2 X_3 \vee X_1 X_2 \overline{X}_3 \vee X_1 X_2 X_3$

$g_1 = X_1 X_2 X_3$

$g_2 = X_1 X_2 \overline{X}_3$

$g_3 = X_1 X_2 X_3 \vee X_1 X_2 \overline{X}_3 = X_1 X_2$

$g_4 = \overline{X}_1 X_2 X_3$

$g_5 = \overline{X}_1 X_2 X_3 \vee X_1 X_2 X_3 = X_2 X_3$

$g_6 = \overline{X}_1 X_2 X_3 \vee X_1 X_2 \overline{X}_3$

$g_7 = \overline{X}_1 X_2 X_3 \vee X_1 X_2 \overline{X}_3 \vee X_1 X_2 X_3$

Визначення. Імпліканта g булевої функції f, що є елементарною кон'юнкцією, називається простою, якщо жодна частина імпліканти g не є імплікантою функції f.

З прикладу видно, що імпліканти g_3 і g_5 є простими імплікантами функції f. Інші імпліканти не є простими, так як їх частини є імплікантами функції f, наприклад g_3 є частиною g_1. Наведемо без доведення два твердження, корисні при отриманні мінімальної ДНФ.

1. Диз'юнкція будь-якого числа імплікант булевої функції f є також імплікантою цієї функції.

2. Будь-яка булева функція f еквівалентна диз'юнкції всіх своїх простих імплікант. Така форма представлення булевої функції називається **скороченою ДНФ.**

Перебір всіх можливих імплікант для булевої функції f з розглянутого прикладу дає можливість переконатися, що простих імплікант всього дві: g_3 и g_5. Отже, скорочена ДНФ функції f має вигляд:

$$f = g_3 \vee g_5 = X_1 X_2 \vee X_2 X_3$$

Як видно з табл. 6.8., імпліканти g_3 и g_5 в сукупності покривають своїми одиницями всі одиниці функції f. Отримання скорочених ДНФ є першим етапом відшукання мінімальних форм булевих функцій. Як вже зазначалося, в скорочену ДНФ входять всі прості імпліканти булевої функції. Іноді з скороченої ДНФ можна прибрати одну або кілька простих імплікант, не порушуючи еквівалентності вихідної функції. Такі прості імпліканти назвемо зайвими. Вилучення зайвих простих імплікант зі скорочених ДНФ - другий етап мінімізації.

Визначення. Скорочена ДНФ булевої функції називається **тупиковою**, якщо в ній відсутні зайві прості імпліканти.

Усунення зайвих простих імплікант зі скороченої ДНФ булевої функції не є однозначним процесом, тобто булева функція може мати кілька тупикових ДНФ.

Твердження. Тупикові ДНФ булевої функції f, що містять мінімальну кількість літер, є мінімальними. Мінімальних ДНФ теж може бути кілька.

Розглянемо кілька методів мінімізації. Всі вони практично розрізняються лише на першому етапі - етапі отримання скороченої ДНФ. Слід зазначити, що, на жаль, пошук мінімальної ДНФ завжди пов'язаний з деяким перебором рішень. Існують методи зменшення цього перебору, проте він завжди залишається.

6.7.1 Метод Квайна

Метод Квайна ґрунтується на застосуванні двох основних співвідношень.

1. Співвідношення склеювання

$$Ax \vee A\bar{x} = Ax \vee A\bar{x} \vee A$$

де A — будь який елементарний добуток.

2. Співвідношення поглинання

$$A\tilde{x} \vee A = A, \tilde{x} \in \{x, \bar{x}\}$$

Справедливість обох співвідношень легко перевіряється.

Теорема Квайна. Якщо в СДНФ булевої функції виконати всі можливі склеювання і поглинання, то в результаті буде отримана скорочена ДНФ.

Метод застосуємо до досконалої ДНФ. Зі співвідношення поглинання випливає, що довільний елементарний добуток поглинається будь-якою його частиною.

Для доказу досить показати, що довільна проста імпліканта p = $\tilde{x}_{i1}, \tilde{x}_{i2}, ..., \tilde{x}_{ik}$, може бути отримана. Справді, застосовуючи до p операцію розгортання (обернена операції склеювання): $A = A(x \vee \bar{x}) = Ax \vee A\bar{x}$ по всіх відсутніх змінних вихідної функції f, отримуємо сукупність S констітуєнт одиниці. При склеюванні всіх констітуєнт з S отримаємо імпліканти p. Останнє очевидно, оскільки операція склеювання обернена операції розгортання. Множина S конституент обов'язково присутня в досконалій ДНФ функції f оскільки p - її імпліканта.

Мінімізація за методом Квайна виконується по наступному алгоритму:

1. Виконуються всі склеювання в ДДНФ.

2. Виконуються всі поглинання.

3. Результуюча функція перевіряється на можливість подальшого склеювання і поглинання.

4. Після отримання скороченої ДНФ будується імплікантна матриця, за якою знаходяться "зайві" імпліканти.

Приклад. Нехай є булева функція, задана таблицею істинності (табл 6.9). Її ДДНФ має вигляд:

$$f= \bar{X}_1\bar{X}_2\bar{X}_3X_4 \vee \bar{X}_1\bar{X}_2X_3X_4 \vee \bar{X}_1X_2\bar{X}_3X_4 \vee \bar{X}_1X_2X_3X_4 \vee X_1X_2X_3\bar{X}_4 \vee X_1X_2X_3X_4$$

1 2 3 4 5 6

Для зручності викладу помітимо кожну констітуенту одиниці з ДДНФ функції f будь-яким десятковим номером (довільно). Виконуємо склеювання. Конституента 1 склеюється тільки з конституентою 2 (по змінній x_3) і з

конституентою 3 (по змінній x_2) конституента 2 з конституентою 4 і т. д. В результаті отримуємо:

1-2: $\overline{X}_1\overline{X}_2X_4$

1-3: $\overline{X}_1\overline{X}_3X_4$

2-4: $\overline{X}_1X_3X_4$

3-4: $\overline{X}_1X_2X_4$

4-6: $X_2X_3X_4$

5-6: $X_1X_2X_3$

Таблиця 6.9.

x_1	x_2	x_3	x_4	f
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

Зауважимо, що результатом склеювання є завжди елементарна кон'юнкція, що представляє собою загальну частину склеюваних конституент.

Далі проводимо склеювання одержуваних елементарних кон'юнкцій. Склеюються тільки ті кон'юнкції, які містять однакові змінні. Має місце два випадки склеювання:

$$\overline{X}_1\overline{X}_2X_4 \vee \overline{X}_1X_2X_4 = \overline{X}_1\overline{X}_2X_4 \vee \overline{X}_1X_2X_4 \vee \overline{X}_1X_4$$

$$\overline{X}_1\overline{X}_3X_4 \vee \overline{X}_1X_3X_4 = \overline{X}_1\overline{X}_3X_4 \vee \overline{X}_1X_3X_4 \vee \overline{X}_1X_4$$

з появою однієї і тієї ж елементарної кон'юнкції $\overline{X}_1X_4$. Подальші склеювання неможливі. Зробивши поглинання (з отриманої ДНФ викреслюємо всі елементарні кон'юнкції, які поглинаються), отримаємо скорочену ДНФ:

$$X_2X_3X_4 \vee X_1X_2X_3 \vee \overline{X}_1X_4$$

Переходимо до наступного етапу. Для отримання мінімальної ДНФ необхідно видалити зі скороченою ДНФ всі зайві прості імпліканти. Це робиться за допомогою спеціальної імплікантної матриці Квайна. Рядки такої матриці відзначаються простими імплікантами булевої функції, тобто членами скороченої ДНФ, а стовпці - конституентами одиниці, тобто членами ДДНФ булевої функції.

Приклад (продовження). Імплікантна матриця має вид (табл. 6.10).

Таблиця 6.10.

Прості імпліканти	Конституєнти одиниці					
	$\overline{X}_1\overline{X}_2\overline{X}_3X_4$	$\overline{X}_1\overline{X}_2X_3X_4$	$\overline{X}_1X_2\overline{X}_3X_4$	$\overline{X}_1X_2X_3X_4$	$X_1X_2X_3\overline{X}_4$	$X_1X_2X_3X_4$
$\overline{X}_1X_4$	X	X	X	X		
$X_2X_3X_4$				X		X
$X_1X_2X_3$					X	X

Як вже зазначалося, проста імпліканта поглинає деяку конституенту одиниці, якщо є її власною частиною. Відповідна клітина імплікантної матриці на перетині рядка (з розглянутої простої імпліканти) і стовпця (з конституентою одиниці) відзначається хрестиком (табл. 6.10). Мінімальні ДНФ будуються за імплікантною матрицею наступним чином:

1) знаходяться стовпці імплікантної матриці, що мають тільки один хрестик. Відповідні цим хрестиками прості імпліканти називаються ***базисними*** і складають так зване **ядро булевої функції**. Ядро обов'язково входить в мінімальну ДНФ.

2) розглядаються різні варіанти вибору сукупності простих імплікант, які накриють хрестиками інші стовпчики імплікантної матриці, і вибираються варіанти з мінімальним сумарним числом букв в такій сукупності імплікант.

Приклад (продовження). Ядром нашої функції є імпліканти $X_1X_2X_3$ і $\overline{X}_1X_4$. Імпліканта $X_2X_3X_4$ - зайва, оскільки ядро накриває всі стовпці імплікантної матриці. Тому функція має єдину тупикову і мінімальну ДНФ:

$$f_{min}= X_1X_2X_3 \vee \overline{X}_1X_4$$

Слід зазначити, що число N хрестиків в одному рядку завжди є ступенем 2. Більш того, можна легко переконатися в тому, що $N = 2^{n-k}$, де k — число букв, що містяться в простій імпліканті.

Зауважимо також, що використовуючи різні співвідношення, можна розширити область застосування методу Квайна за межі досконалої ДНФ.

6.7.2 Метод Квайна-Мак-Класкі

Метод являє собою формалізований на етапі знаходження простих імплікант метод Квайна. Формалізація проводиться таким чином:

1. Всі конституенти одиниці з СДНФ булевої функції f записуються їх двійковими номерами.

2. Всі номери розбиваються на непересічні групи. Ознака належності до і-ї групи: і одиниць у кожному двійковому номері конституенти одиниці.

3. Склеювання проводять тільки між номерами сусідніх груп. Склеювані номера відзначаються яким-небудь знаком (закресленням, зірочкою і т.д.).

4. Склеювання проводять всі можливі, як і в методі Квайна. Невідмічені після склеювання номери є простими імплікантами.

Знаходження мінімальних ДНФ далі проводиться по імплікантній матриці, як і в методі Квайна. Більш докладно розглянемо метод на прикладі вирішення наступного завдання: мінімізувати методом Квайна-Мак-Класкі булеву функцію *f*, задану таблицею істинності (табл. 6.9).

$$f=\overline{X}_1\overline{X}_2\overline{X}_3X_4 \vee \overline{X}_1\overline{X}_2X_3X_4 \vee \overline{X}_1X_2\overline{X}_3X_4 \vee \overline{X}_1X_2X_3X_4 \vee X_1X_2X_3\overline{X}_4 \vee X_1X_2X_3X_4$$

1. У СДНФ функції *f* замінимо всі конституенти одиниці їх двійковими номерами: f=0001∨0011∨0101∨0111∨1110∨1111

2. Утворюємо групи двійкових номерів. Ознакою належності до і-ї групи є і одиниць в двійковому номері конституенти одиниці (табл. 6.11).

Таблиця 6.11.

Номер групи	Двійкові номери конституент 1
0	-
1	0001 *
2	0011 * 0101 *
3	0111 * 1110 *
4	1111 *

Таблиця 6.12

Номер групи	Двійкові номери імплікант
1 (1-2)	00-1 * 0-01 *
2 (2-3)	0-11 * 01-1 *
3 (3-4)	-111 111-

Таблиця 6.13

Номер групи	Двійкові номери імплікант
1	0--1 0--1
2	-111 111-

3. Склеїмо номера з сусідніх груп табл.6.11. Склеювані номера позначимо зірочками (номер позначається зірочкою у тому випадку, якщо він бере участь у склеюванні хоча б один раз). На місці розрядів, за якими відбулося склеювання, проставляються прочерки. Результати склеювання занесемо в табл. 6.12. Склеїмо номери з сусідніх груп табл. 6.12. Склеюватися можуть тільки номери, що мають прочерки в однакових позиціях. Склеювані номери відзначимо. Результати склеювання занесемо в табл. 6.13.

4. Маємо три прості імпліканти: -111, 111-, 0--1.

Таблиця 6.14.

Прості імпліканти	Констітуенти одиниці					
	$\overline{X}_1\overline{X}_2\overline{X}_3X_4$	$\overline{X}_1\overline{X}_2X_3X_4$	$\overline{X}_1X_2\overline{X}_3X_4$	$\overline{X}_1X_2X_3X_4$	$X_1X_2X_3\overline{X}_4$	$X_1X_2X_3X_4$
$\overline{X}_1X_4$(0--1)	X	X	X	X		
$X_2X_3X_4$(-111)				X		X
$X_1X_2X_3$(111-)					X	X

5. Будуємо імплікантну матрицю (табл. 6.14). По таблиці визначаємо сукупність простих імплікант - 0 - 1 і 111 -, яка відповідає мінімальній ДНФ. Для відновлення літерного виду простих імплікант досить виписати кон'юнкції тих змінних, які відповідають збереженим двійковим цифрам:

$$f_{min}= X_1X_2X_3 \vee \overline{X}_1X_4$$

Зауважимо, що розбиття конституент на групи дозволяє зменшити число попарних порівнянь при склеюванні.

6.7.3 Метод діаграм Вейча

Метод дозволяє швидко отримувати мінімальні ДНФ булевої функції f невеликого числа змінних. В основі методу лежить представлення булевих функцій діаграмами деякого спеціального виду, які отримали назва діаграм Вейча. Для булевої функції двох змінних діаграма Вейча має вигляд (табл. 6.15). Кожна клітина діаграми відповідає набору змінних булевої функції в її таблиці істинності. У табл.6.15. цю відповідність показано. У клітинці діаграми Вейча ставиться одиниця, якщо булева функція приймає одиничне значення на відповідному наборі. Нульові значення булевої функції в діаграмі Вейча не ставляться. Для булевої функції трьох змінних діаграма Вейча має наступний вигляд (табл. 6.16). Додавання до неї ще такої ж таблиці дає діаграму для функції 4-х змінних (табл. 6.17) . Таким же чином, тобто приписуванням ще однієї діаграми 4-х змінних, до щойно розглянутої, можна отримати діаграму для функції 5-ти змінних і т. д., однак діаграми для функцій з числом змінних більше 4-х використовуються рідко.

Таблиця 6.15.

	X_1	$\overline{X}_1$
X_2	11	01
$\overline{X}_2$	10	00

Таблиця 6.16.

	X_1		$\overline{X}_1$	
X_2	110	111	011	010
$\overline{X}_2$	100	101	001	000
	$\overline{X}_3$	X_3		$\overline{X}_3$

Таблиця 6.17.

	X_2		$\overline{X}_2$		
x_1	1100	1101	1001	1000	$\overline{X}_3$
x_1	1110	1111	1011	1010	x_3
$\overline{X}_1$	0110	0111	0011	0010	x_3
$\overline{X}_1$	0100	0101	0001	0000	$\overline{X}_3$
	$\overline{X}_4$	x_4	x_4	$\overline{X}_4$	

Таблиця 6.18.

	x_1			$\overline{X}_1$
X_2	1	1		1
$\overline{X}_2$				1
	$\overline{X}_3$	X_3		$\overline{X}_3$

Для наведених діаграм характерно наступне:

1) кожній клітині діаграми відповідає свій набір;

2) сусідні набори розташовані поруч у рядку або у стовпці. Сусідніми наборами називаються набори, які відрізняються однією компонентою. Нагадаємо, що конституенти, відповідні таким наборам, склеюються (див. метод Квайна-Мак-Класкі). Наприклад, для функції, заданої табл. 6.18, конституенти, що відповідають парі одиниць в лівій частині таблиці, склеюються і породжують елементарну кон'юнкцію з двох букв:

$$X_1 X_2 X_3 \vee X_1 X_2 \overline{X}_3 = X_1 X_2$$

Про пару одиниць в правій частині діаграми можна сказати те ж саме: $\overline{X}_1 X_2 \overline{X}_3 \vee \overline{X}_1 \overline{X}_2 \overline{X}_3 = \overline{X}_1 \overline{X}_3$.

Відзначимо, що отриману елементарну кон'юнкцію легко визначити відразу по діаграмі: *це кон'юнкція змінних, що приймають одне і те ж значення в обох клітинах*.

Ще одне важливе зауваження: *стовпці, розташовані по краях діаграми, теж вважаються сусідніми*. Для нашого прикладу це означає, що має місце ще одне склеювання, в результаті якого, слідуючи вказаному правилу, отримуємо елементарну кон'юнкцію $X_2\overline{X}_3$. З розглянутих раніше методів нам відомо, що можливе подальше склеювання одержуваних елементарних кон'юнкцій. На діаграмах Вейча вони теж розташовуються поруч. Загальне правило склеювання на діаграмах Вейча можна сформулювати наступним чином: *склеюванню підлягають прямокутні конфігурації, заповнені одиницями і які містять число клітин, що є ступенем 2. Отримана нова елементарна кон'юнкція визначається як кон'юнкція змінних, які не змінюють свого значення на всіх склеюваних наборах*. Число *m* змінних, які залишилися в елементарній кон'юнкції визначається легко:

$$m=n-\log_2 M,$$

де n- число змінних,

M - число склеюваних наборів.

Метод широко використовується на практиці, завдяки простоті і зручності. Після невеликого тренування досягається елементарний навик визначення мінімальної ДНФ по діаграмі «з першого погляду».

Мінімізація булевої функції полягає в знаходженні мінімального покриття всіх одиниць діаграми Вейча блоками з одиниць (зазначеної конфігурації), розташованих в сусідніх клітинах діаграми. При цьому завжди вважається, що лівий край діаграми Вейча 4 -х змінних примикає до її правого краю, а верхній край діаграми примикає до нижнього її краю. Для зручності можна припустити, що діаграма згортається в циліндр як по горизонталі, так і по вертикалі. Після отримання мінімального покриття всіх одиниць діаграми Вейча, мінімальна ДНФ булевої функції записується як диз'юнкція елементарних кон'юнкцій, відповідних виділеним блокам одиниць в діаграмі.

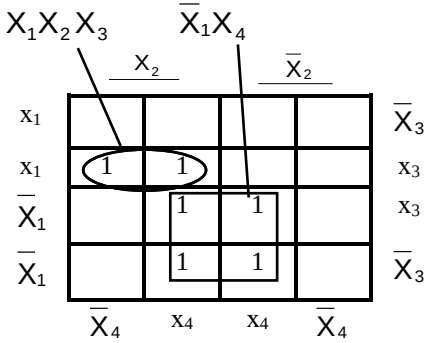
У діаграмі Вейча необхідно і достатньо, щоб кожна одинична клітина брала участь в склеюванні хоча б один раз.

Нову склейку можна утворювати в тому випадку, якщо є хоча б одна одинична клітина, яка не брала участь до цього в склеюванні.

Вся діаграма повинна бути покрита найменшою кількістю склеювань. У склейку може входити 2^n сусідніх клітин (2, 4, 8, 16. і т.д.).

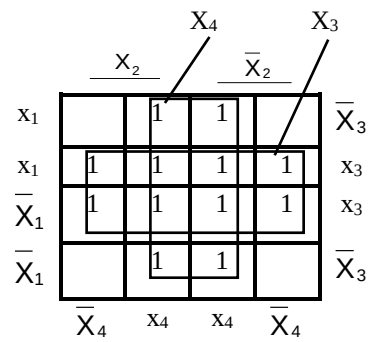
Розглянемо кілька прикладів.

Таблиця 6.19.



$f_1 = X_1X_2X_3 \vee \overline{X}_1X_4$

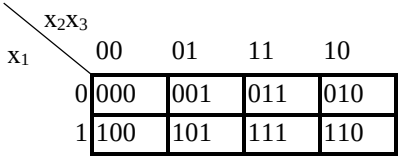
Таблиця 6.20.



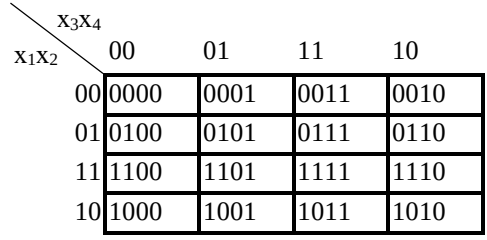
$f_2 = X_4 \vee X_3$

6.7.4 Карти Карно

Іноді зручно користуватися іншим представленням діаграм з цифровим кодом. Це карти Карно. Приклади карт Карно наведені на малюнку 6.2. По гранях карти проставляються двійкові коди - коди Грея, що дає можливість легко проставляти значення функції і знаходити результат. Правила мінімізації з застосування карт Карно такі ж, як і для діаграм Вейча.



а)



б)

Рис. 6.2- Карти Карно: а) функції 3-х змінних;
б) функції 4-х змінних.

6.7.5 Особливості мінімізації булевих функцій з великим числом змінних

Розглянемо деякі особливості роботи з картами Карно для великого числа змінних. При числі змінних, що дорівнює або більше п'яти, відобразити графічно функцію у вигляді єдиної плоскої карти неможливо. У таких випадках будують комбіновану карту, що складається із сукупності більш простих базових карт, наприклад карт для функції 4-х змінних. Процедура мінімізації в цьому випадку полягає в тому, що спочатку знаходять мінімальні форми всередині базових карт, а потім, розширюючи поняття сусідніх клітин, знаходять мінімальні накриття для сукупності карт. Сусідніми клітинами є клітини, що збігаються при накладанні базових карт одна на одну. Приклади карт Карно для булевих функцій 5-ти і 6-ти змінних представлені на рис. 6.3. і 6.4. відповідно.

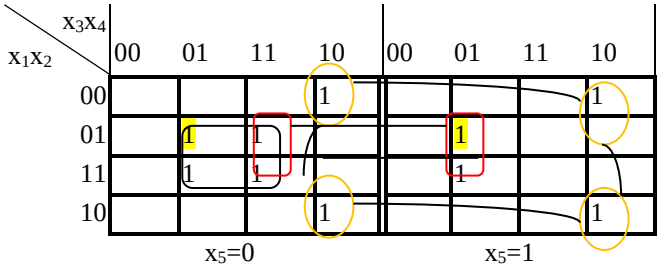


Рис. 6.3-Карта Карно для булевої функції 5-ти змінних.

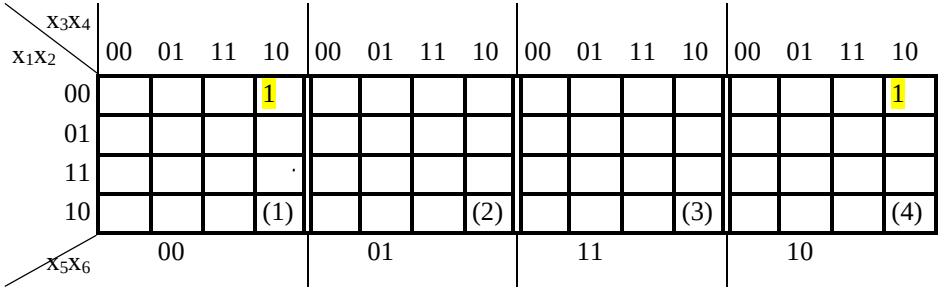


Рис. 6.4-а- Карта Карно для булевої функції шести змінних.

За малюнком 6.4.-а можна зробити висновок, що сусідніми є для 1-й базової карти - 2-а і 4-а; для 2-ї - 1-а і 3-я; для 3-ї 2-а і 4-а; для 4-ї - 1-а і 3-я.

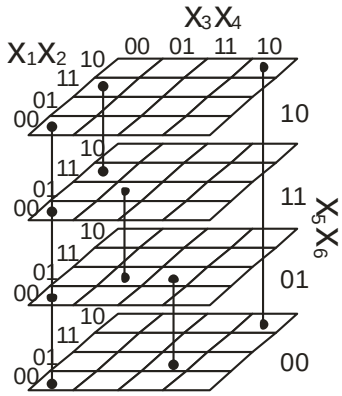


Рис. 6.4-б- Карта Карно для булевої функції шести змінних.

Приклад сусідніх клітин показано на малюнку 6.4.-б.

При збільшенні кількості змінних на одну, площа карти збільшується в два рази - до неї домальовується ще така ж карта. При цьому нова змінна дорівнює 1 на новій карті, і 0 на тій, яка була раніше.

6.7.6 Мінімізація кон'юнктивних нормальних форм

Мінімізація КНФ проводиться аналогічно розглянутим методам мінімізації ДНФ булевих функцій, тому зупинимося лише на основних положеннях.

Нагадаємо, що конституентою нуля називається функція, що приймає значення 0 на єдиному наборі. Вона виражається диз'юнкцією всіх змінних функції. Наприклад, набору 0110 відповідає конституента нуля $X_1 \vee \overline{X}_2 \vee \overline{X}_3 \vee X_4$

Визначення. Імплицентою g булевої функції f називається функція, що приймає значення 0 на підмножині нульових наборів функції f.

Взначення. Простою імпліцентою функції f називається елементарна диз'юнкція, яка є імпліцентою функції f, причому ніяка її власна частина імпліцентою функції f не являється.

Завданням мінімізації КНФ є визначення мінімальної КНФ. Це завдання також вирішується в два етапи - пошук скороченою КНФ (кон'юнкція всіх простих

імпліцент) і потім знаходження мінімальної КНФ. Другий етап мінімізації виконується за допомогою імпліцентної таблиці Квайна точно так само, як при пошуку мінімальної ДНФ, так як можливі тільки два варіанти: або дана проста імпліцента поглинає дану констітуенту нуля, або ні відповідно до співвідношення поглинання: $(A \vee x)A=A$

Що стосується першого етапу - пошуку всіх простих имплицент, то практично всі методи мінімізації ДНФ мають свої аналоги для КНФ. Розглянемо це докладніше.

Співвідношення склеювання по Квайну:

$(A \vee x)(A \vee \overline{X}) = (A \vee x)(A \vee \overline{X})A = A$

Метод Квайна для кон'юнктивних форм:

1. Виконуються всі неповні склеювання в ДКНФ.

2. Виконуються всі поглинання.

3. Результуюча функція перевіряється на можливість подальшого склеювання і поглинання.

4. Після отримання скороченої КНФ будується імпліцентна матриця, за якою знаходяться "зайві" імпліценти.

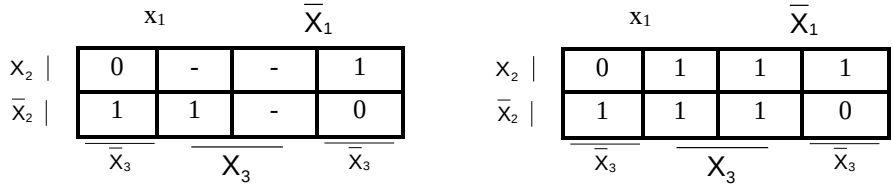
По діаграмі Вейча пошук мінімальної КНФ здійснюється так само просто, як у випадку ДНФ. Відмінність полягає лише в тому, що аналізуються нульові набори і змінні виписуються з інверсіями (за правилами запису конституент 0).

6.7.7 Мінімізація частково визначених булевих функцій

У реальних задачах дуже часто буває так, що значення булевої функції на деяких наборах не визначено і може довизначатися довільно. Вихідні сигнали на цих наборах можуть приймати будь-які значення - 0 або 1. Вхідні набори, які дають невизначені значення функції називаються забороненими. При синтезі схем, що реалізують неповністю визначені функції, вихідним сигналам, відповідним забороненим наборам, надають такі значення, при яких можна побудувати найбільш просту схему. У цьому випадку довизначення функції доцільно робити таким чином , щоб її мінімальна нормальна форма мала найменше число букв з усіх можливих варіантів довизначення. Розглянемо простий приклад (рис.6.5, 6.6.).

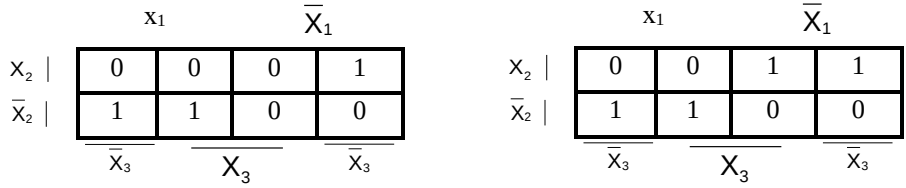
Функція задана діаграмою Вейча, представленою рис. 6.5.а. Довизначення функції на невизначених наборах одиницями (рис. 6.5.б) або нулями (рис. 6.6.а) призводить до різних мінімальних ДНФ. Однак більш проста мінімальна ДНФ виходить, якщо виконати довизначення так, як це зроблено на діаграмі Вейча (рис.

6.6.б). Алгоритм пошуку мінімальної ДНФ частково визначеної функції f можна представити таким чином.



а) б)

Рисунок 6.5



а) б)

Рис. 6.6.

1. Знайти будь-яким відомим способом скорочену ДНФ функції, це можна зробити довизначенням одиницями вихідної функції f на всіх невизначених наборах.

2. Вибрати мінімальну ДНФ по імпликантній матриці, де в стовпцях виписані лише ті конституенти одиниці функції f, які відповідають повністю визначеним одиничним наборам.

Аналогічний алгоритм (з довизначенням нульовими наборами) може бути запропонований для пошуку КНФ. При цьому довизначення таблиці істинності функції f може бути виконано по-різному для КНФ і ДНФ.

Зауважимо, що для вирішення розглянутої задачі практично достатньо тих навичок, які були отримані при мінімізації повністю визначених булевих функцій безпосередньо по діаграмі Вейча. У випадках, коли мінімальних форм кілька, наводиться одна з них.

6.7.8 Мінімізація систем булевих функцій

На практиці дуже часто доводиться реалізовувати сукупності булевих функцій. Якщо провести мінімізацію булевих функцій, що входять в систему, незалежно одна від одної, то загальна схема буде складатися з ізольованих підсхем. Її можна іноді спростити за рахунок об'єднання ділянок підсхем, що реалізують однакові члени, що входять в кілька булевих функцій системи.

Завдання мінімізації систем булевих функцій добре досліджене в класі функціонально повних систем: «диз'юнкція», «кон'юнкція», «заперечення». Розглянемо один з найбільш поширених методів мінімізації.

Визначення. Система диз'юнктивних нормальних форм булевих функцій називається мінімальною, якщо її повна множина елементарних кон'юнкцій містить мінімальну кількість букв, а кожна диз'юнктивна нормальна форма булевої функції системи включає мінімальне число елементарних кон'юнкцій найменшого рангу. При цьому диз'юнктивна нормальна форма представлення булевої функції в мінімальній системі в загальному випадку не збігається з її мінімальної диз'юнктивною нормальною формою. Мінімізація систем повністю визначених булевих функцій може проводитися за алгоритмом, аналогічним алгоритмом методу Квайна з невеликими відмінностями. Алгоритм мінімізації наступний:

1. Побудувати повну множину А елементарних кон'юнкцій системи булевих функцій, що мінімізується, вважаючи, що спочатку кожна з функцій системи представлена в СДНФ. Кожній конституенті одиниці множини А присвоїти ознаку, що містить номери функцій системи, в які входить розглянута конституента.

2. Провести мінімізацію СДНФ функції L, конституентами одиниці якої є всі елементи множини А. При виконанні склеювання двох конституент одиниці кожної знову утвореної елементарної кон'юнкції присвоїти ознаку, що складається з номерів функцій, загальних для двох склеюваних конституєнт одиниці. Останнє справедливо і для двох склеюваних елементарних кон'юнкцій з ознаками. Якщо ознаки склеюваних констітуєнт одиниці не містять загальних номерів, то склеювання не проводиться. Поглинання проводиться тільки для елементарних кон'юнкцій з однаковими ознаками. Отримані в результаті склеювання і поглинання кон'юнкції називаються простими імплікантами системи функцій.

3. Побудувати імпликантну матрицю функції L, аналогічну матриці Квайна з тією різницею, що для кожної констітуєнти одиниці виділяється стільки стовпців, скільки різних номерів функцій містить її ознака. Покриття матриці імплікантами проводиться аналогічно методу Квайна.

Найбільш цікава задача пошуку абсолютно мінімальної форми - форми, що містить мінімальну кількість операцій заданої функціонально повної системи. У

загальному вигляді ця задача не вирішена. Більш того, відомо, що абсолютно мінімальна форма не завжди може бути отримана з мінімальної ДНФ. Водночас відшукання форми, близької до абсолютно мінімальної, - реальне і практично досяжне завдання.

Тут доречно сказати про типи і класи булевих функцій у визначенні Шеннона. Вище вже згадувалося про взаємно однозначні відповідності між схемами і формами представлення булевих функцій. У той же час, очевидно, що одна і та ж схема може реалізувати кілька різних булевих функцій, якщо якісь вхідні змінні поміняти місцями. У цьому випадку функції належать до одного класу. Якщо ж змінні переставляти, допускаючи їх заперечення, то функції належать до одного типу. Якщо знайти абсолютно мінімальну форму представлення хоча б однієї функції кожного класу і побудувати відповідні таблиці, то задачу синтезу перемикальних схем можна було б звести до пошуку потрібного класу в таблиці. Такі таблиці для функцій 4-х змінних були складені (при реалізації функцій контактними схемами), практичного поширення цей підхід не отримав. Однією з причин є той факт, що число класів дуже швидко зростає із зростанням *n*.

Відоме більш просте завдання - що полягає у спрощенні диз'юнктивно-кон'юнктивної форми, допускаючи заперечення лише над змінними. Часто вона називається завданням дужкової мінімізації, і в даний час відомо досить багато методів такого спрощення. У загальному вигляді завдання факторизації не вирішене, але для булевого базису, в ряді випадків, використовуючи операцію винесення за дужки спільних членів, можна отримати дужкову форму, значно більш просту, ніж мінімальна ДНФ булевої функції.

7 ТЕМА. ПРОЕКТУВАННЯ КОМБІНАЦІЙНИХ СХЕМ

7.1 Комбінаційні схеми. Основні поняття

Комбінаційною схемою (КС) називається схема з логічних (перемикальних) елементів, що реалізує булеву функцію або сукупність булевих функцій. У загальному випадку КС можна представити схемою, наведеною на рис. 7.1, де x_1, x_2,....x_n — входи КС, f_1, f_2,...,f_m — її виходи.

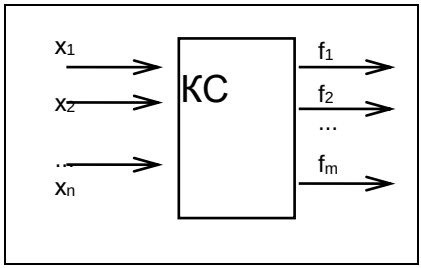
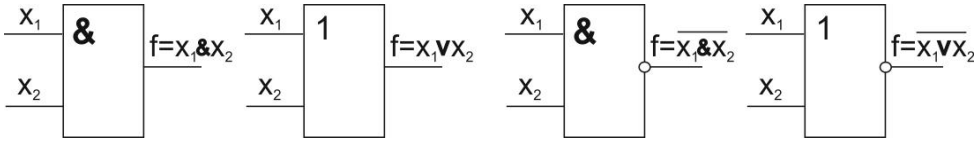


Рисунок. 7.1

Під логічним (перемикальним) елементом найчастіше розуміють технічний пристрій, що реалізує одну елементарну булеву функцію.

В рамках даного курсу ми не розглядатимемо фізичні явища, що лежать в основі розробки і функціонування логічних елементів. Зазвичай логічний елемент розуміють як «чорний ящик» і враховується тільки реалізована елементом булева функція. Приклади логічних елементів АБО - НІ, І - НІ, що реалізують відповідні булеві функції двох змінних, представлені на рис. 7.2.



а) б) в) г)

Рис. 7.2 а) кон'юнктор ; б) диз'юнктор; в) елемент І-НІ; г) елемент АБО-НІ

Під глибиною (числом рівнів) КС розуміють максимальне число логічних елементів, розташованих на шляху проходження сигналу від входів КС до її

виходу. Глибина КС істотно впливає на швидкодію КС, так як кожен логічний елемент має внутрішню затримку розповсюдження сигналу. Одно-і дворівневі КС мають максимальну швидкодію. Проте вони не завжди можуть бути використані, оскільки число входів реальних логічних елементів в інтегральному виконанні обмежене. Якщо КС реалізує одну булеву функцію, то вона називається одновиходовою КС (рис. 7.3). Якщо КС реалізує сукупність булевих функцій, то вона називається багатовиходовою КС.

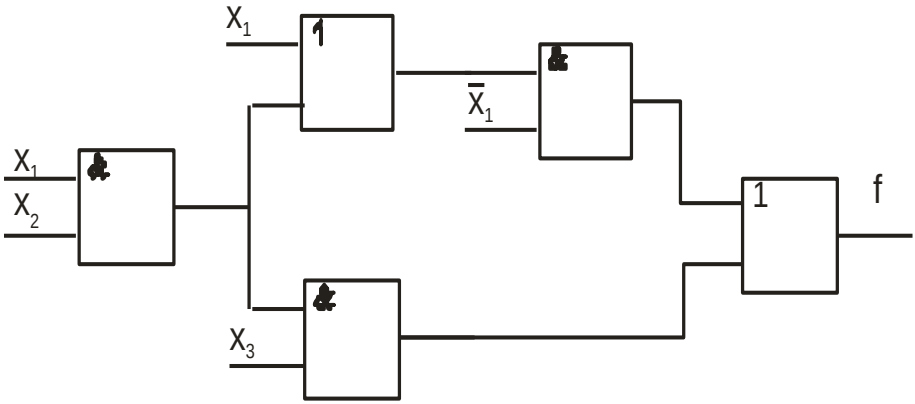


Рис. 7.3

Комбінаційним схемам відповідають схеми **без зворотних зв'язків** (під зворотним зв'язком розуміється з'єднання виходу деякого логічного елемента зі своїм входом, можливо, через ланцюжок інших логічних елементів (рис. 7.4)).

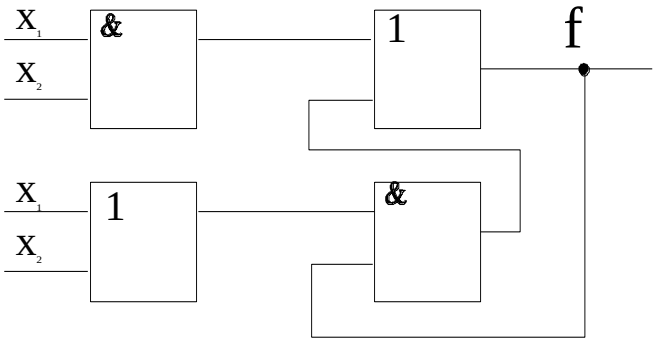


Рис. 7.4

Логічні елементи, що використовуються для побудови КС, характеризуються певними технічними параметрами, серед яких найбільш важливі коефіцієнт об'єднання по входу *I*; коефіцієнт об'єднання по виходу *U* (коефіцієнт розгалуження) і затримка сигналу $\Delta\tau$ в логічному елементі.

Система функцій, реалізована обраною для синтезу схем сукупністю логічних елементів, завжди повинна бути функціонально повною, тобто допускати реалізацію будь-якої булевої функції на основі принципу суперпозиції. Якщо для реалізації системи функцій обрані функції І, АБО, НЕ, то вважають, що реалізований *булевий базис*. Проектування КС в булевом базисі найбільш просте, так як методи мінімізації булевих функцій в основному орієнтовані на нього. Тому, як правило, на першому етапі КС проектується в булевому базисі з подальшим переходом в заданий базис. Якщо обрані функції І-НІ або АБО-НІ, то вважають, що реалізується універсальний або монофункціональний базис. Для зручності проектування в різних системах елементів можлива реалізація і змішаного базису.

Конструктивно логічні елементи об'єднуються в єдині корпуси інтегральних мікросхем (ІМС). У загальному випадку, під інтегральною мікросхемою розуміється мікроелектронний виріб, що має високу щільність упаковки елементів і з'єднань між ними; при цьому всі елементи виконані нероздільно і електрично з'єднані між собою таким чином, що з точки зору специфікації, випробувань, поставки і експлуатації виріб розглядається як єдине ціле.

Число логічних елементів, що об'єднуються в один корпус ІМС, характеризує ступінь інтеграції логічних елементів. Ступінь інтеграції впливає на надійність, габаритні розміри, енергоспоживання проектованих КС. Розрізняють ІМС малого, середнього, великого та надвеликого ступеня інтеграції.

Закон Му́ра - емпіричне спостереження, спочатку зроблене Гордоном Муром, згідно з яким кількість транзисторів, що розміщуються на кристалі інтегральної схеми, подвоюється кожні 24 місяці.

В даний час використовують оцінки: до 100 000 вентилів - ВІС, більш 100000 вентилів - СВІС.

Коефіцієнт об'єднання *I* по входу логічного елемента ІМС задає максимальне число логічних елементів, виходи яких можуть бути об'єднані на вході даного елемента.

Коефіцієнт об'єднання *U* по виходу (коефіцієнт розгалуження (разветвления)) логічного елемента ІМС задає максимальне число входів логічних елементів, які можуть бути з'єднані з виходом даного логічного елемента без порушення режиму його роботи.

Якщо деякий логічний елемент КС виявився перевантаженим по виходу (після закінчення проектування КС), то необхідно провести еквівалентне

перетворення структури КС з метою його розвантаження. Це перетворення зводиться або до введення в КС спеціальних підсилювачів-формувачів, або до дублювання даного логічного елемента.

Затримка $\Delta\tau$ логічного елемента характеризує проміжок часу між моментами встановлення сигналів на входах і виходах логічного елемента. Поширення сигналу по КС залежно від затримок логічних елементів, через які він проходить, характеризує швидкодію КС. Проходження сигналів по різних шляхах в КС викликає появу різних затримок, що може послужити причиною нестійкого функціонування КС.

Сучасні засоби обчислювальної техніки збираються з ІМС, типових як по фізичних принципах функціонування, так і по виконуваних логічних функціях.

Основні вимоги до комплекту ІМС наступні:

1) ІМС комплекту повинні забезпечувати можливість побудови різних пристроїв і систем обробки цифрової інформації;

2) число різних типів ІМС має бути оптимальним, щоб забезпечувалася простота експлуатації складних систем і взаємозамінність їх частин;

3) в комплекті повинні бути передбачені ІМС, які не виконують логічних функцій, а узгоджують навантажувальні характеристики логічних елементів і забезпечують формування електричних сигналів;

4) ІМС комплекту повинні бути технологічними у виготовленні і зручними для перевірки їх електричних параметрів;

5) комплект ІМС повинен бути функціонально повним;

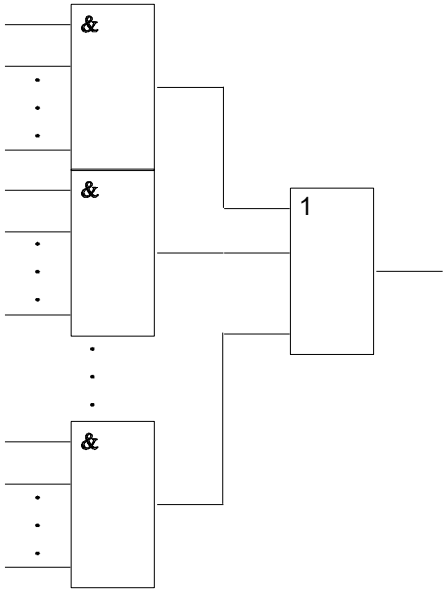
6) комплект ІМС повинен містити спеціальні ІМС, призначені для побудови керуючих ланцюгів, запам'ятовуючих пристроїв, ланцюгів зв'язку запам'ятовуючих і логічних пристроїв, узгодження електромеханічних пристроїв (реле, перемикачів, механізмів друку) та логічних пристроїв, зв'язку різних пристроїв з пристроями введення-виведення інформації, індикації інформаційних станів і генерації високостабільних тактових сигналів.

У процесі реалізації конкретних схем вирішуються завдання забезпечення необхідних характеристик надійності. У загальному випадку ці характеристики можуть бути розраховані, виходячи з надійнісних характеристик елементів і конкретної схеми (це стосується не тільки комбінаційних схем). У тих випадках, коли розрахункова надійність не задовольняє вихідним вимогам, застосовуються спеціальні методи підвищення надійності. Серед них найбільш цікавими, з точки зору теорії цифрових автоматів, є методи контролю роботи схем з використанням завадостійких кодів.

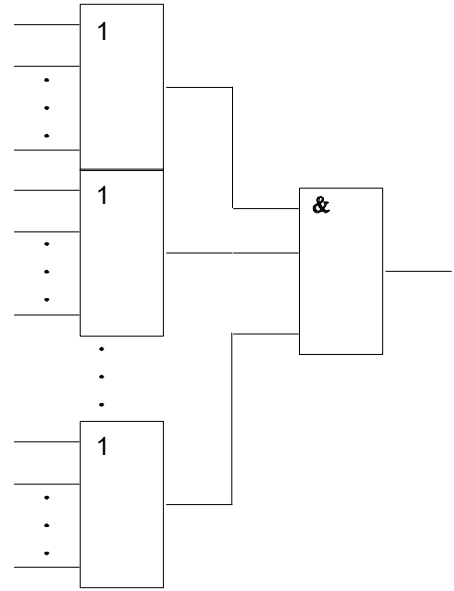
Таким чином, на етапі структурного синтезу вирішується завдання побудови комбінаційної схеми, що реалізує задану сукупність булевих функцій і задовольняє заданим вимогам швидкодії та надійності.

7.2 Проектування комбінаційних схем в булевому і монофункціональному базисах

При проектуванні КС на логічних елементах І, АБО, НІ і відсутності обмежень на число входів елементів користуються викладеними раніше методами мінімізації булевих функцій. При наявності обмежень найбільш простим методом є застосування спеціальних ІМС, званих розширювачами і наявних у комплектах ІМС. Розширювачі дозволяють збільшити, в разі необхідності, число входів логічного елемента шляхом включення додаткового (точно такого ж) логічного елемента на один з входів основного.



а)



б)

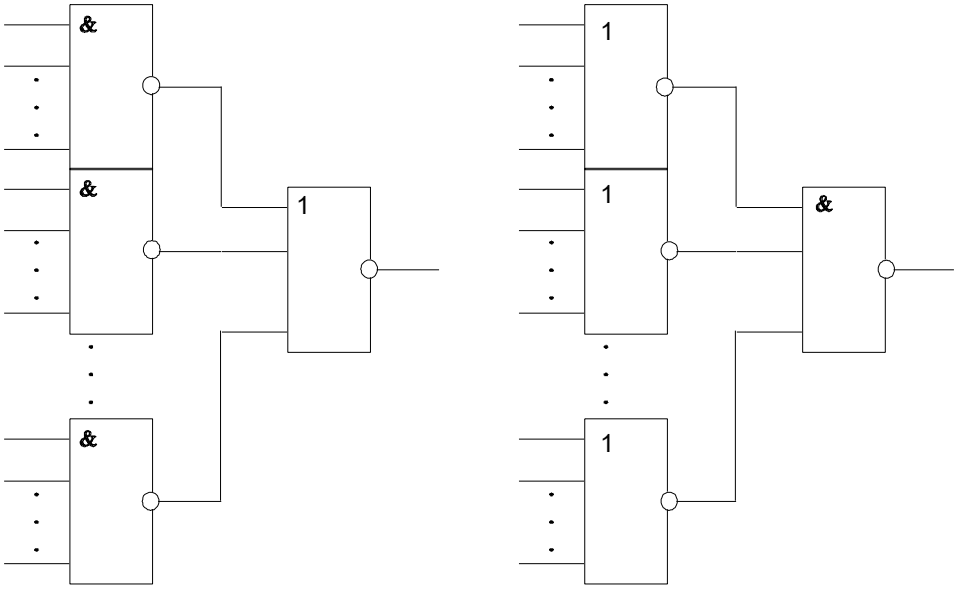


Рис. 7.5

Розглянемо перехід від реалізації булевої функції в булевому базисі, тобто на логічних елементах І, АБО, НІ до схем в монофункціональному базисі, тобто реалізованих на логічних елементах АБО - НІ чи І - НІ. Такі логічні елементи широко використовуються в наявних на практиці комплектах ІМС.

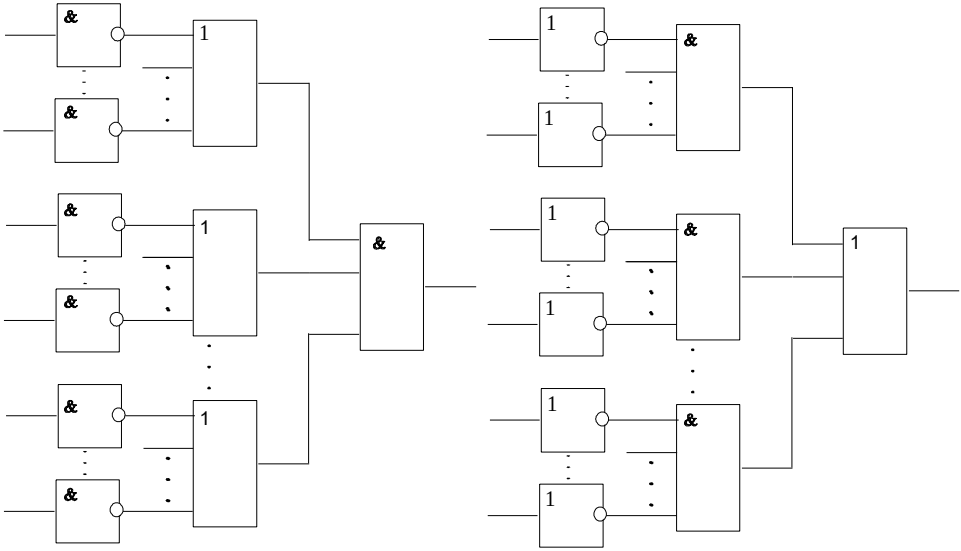
Зауважимо, що якщо булева функція в базисі І, АБО, НІ реалізована дворівневою КС відповідно до рис. 7.5, а. б, то перехід до реалізації в базисі І - НІ або АБО-НІ може бути здійснений заміною всіх елементів КС (рис. 7.5 а) на логічні елементи АБО-НІ, та елементів КС (рис. 7.5.б) на логічні елементи АБО-НІ із збереженням як змінних, поданих на входи елементів, так і зв'язків між ними. Перетворені КС представлені на рис. 7.5. в., г. У наведених на малюнках схемах допускається, що на входи КС змінні надходять як з запереченням, так і без заперечення, тобто елемент НІ на входах КС не враховується.

Однак використовувати в якості зовнішніх входів КС змінні Xi і $\overline{X}i$ не завжди вдається. У цьому випадку КС, реалізована в булевом базисі, може бути представлена рис.7.6 а, б і є трирівневою. Відповідні КС, реалізовані в монофункціональному базисі, також будуть трирівневими.

Існує вельми простий спосіб переходу від реалізації КС в базисі І - НІ до реалізації КС в базисі І - АБО. Спосіб заснований на застосуванні правил де

Моргана і дозволяє за допомогою нескладного алгоритму відразу з реалізації КС в базисі І - НЕ отримати реалізацію КС в базисі І - АБО. Якщо у вихідній КС відсутні елементи І-НІ, що виконують функцію інвертора, то перетворена КС буде містити рівно стільки логічних елементів І, АБО, скільки їх є у вихідній КС. Якщо у вихідній КС інвертори є, то в перетвореній КС число логічних елементів (порівняно з вихідною КС) буде зменшено рівно на число інверторів.

Перетворення складних аналітичних виразів з булевого базису в базис АБО - НІ або І - НІ може бути зроблено за допомогою методу, заснованого на послідовному застосуванні теорем де Моргана. Метод дозволяє здійснювати перехід від довільної за формою булевої функції, реалізованої на елементах І, АБО, НІ, до форми, яка реалізується на елементах І-НІ, АБО-НІ, зокрема від мінімальної ДНФ або КНФ до мінімальних (в точності до однієї літери) найкоротших форм в базисі І-НІ чи АБО-НІ.



а) б)

Рис. 7.6

Приклад. Реалізувати булеву функцію

f= $\overline{X}_1\overline{X}_2 \vee \overline{X}_1X_4 \vee X_2X_3X_4 \vee X_3X_4$

в монофункціональних базисах І-НІ, АБО-НІ.

Функція задана в булевому базисі. Застосувавши правило де Моргана, перетворимо функцію в монофункціональний базис.

$$f=\overline{\overline{X}_1\overline{X}_2 \vee \overline{X}_1X_4 \vee X_2X_3X_4 \vee X_3X_4}=$$

$$\overline{\overline{X_1\overline{X}_2} \wedge \overline{\overline{X}_1X_4} \wedge \overline{X_2X_3X_4} \wedge \overline{X_3X_4}}$$ - перетворення в базис І-НІ.

$$f=\overline{\overline{\overline{X}_1\overline{X}_2} \vee \overline{\overline{X}_1X_4} \vee \overline{X_2X_3X_4} \vee \overline{X_3X_4}}=$$

$$\overline{X_1 \vee X_2} \vee \overline{X_1 \vee \overline{X}_4} \vee \overline{\overline{X}_2 \vee \overline{X}_3 \vee \overline{X}_4} \vee \overline{\overline{X}_3 \vee \overline{X}_4}$$ - перетворення в базис АБО-НІ.

КС відповідні даними реалізаціям, представлені на рис.7.7 а, б, відповідно.

На закінчення нагадаємо, що отримання мінімальних форм булевих функцій в монофункціональному базисі можна представити таким чином:

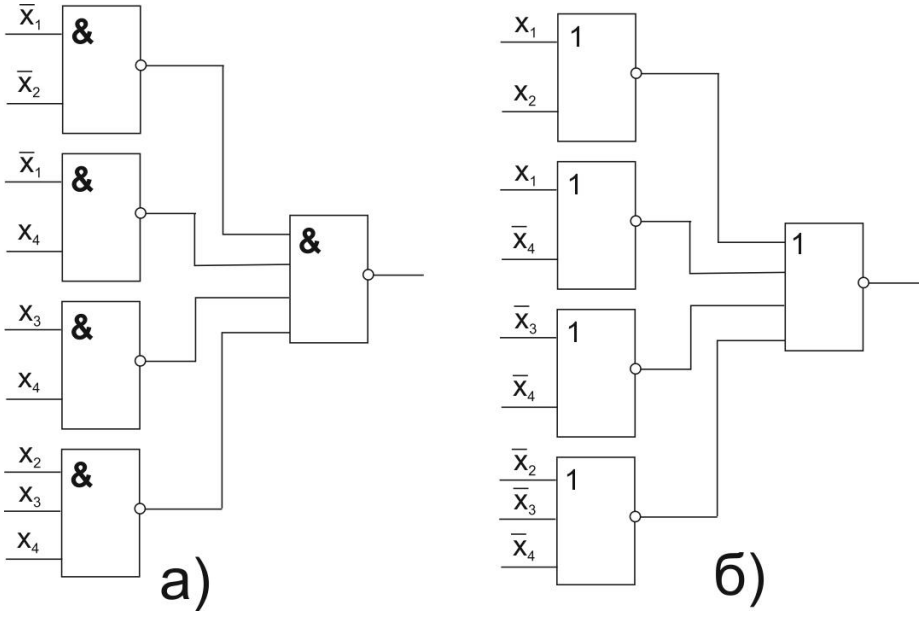


Рис. 7.7. а) - реалізація функції в базисі І-НІ;
б) - реалізація функції в базисі АБО-НІ.

1) отримання ДДНФ булевої функції;

2) отримання мінімальної ДНФ булевої функції на основі її ДДНФ за допомогою будь-якого відомого методу мінімізації булевих функцій;

3) переведення мінімальної ДНФ в монофункціональний базис із застосуванням теорем де Моргана в будь-якій послідовності.

Останнє справедливо, в силу того, що застосування теорем де Моргана не змінює числа букв у виразі.

7.3 Проектування комбінаційних схем з урахуванням коефіцієнтів об'єднання по входу і виходу

Допустима величина коефіцієнта об'єднання по входу (*I*) в реальних умовах проектування КС робить істотний вплив на вибір її структури. Припустимо, булева функція аналітично представлена виразом:

$$f= \overline{X}_1\overline{X}_2 \vee \overline{X}_1X_4 \vee X_2X_3X_4 \vee X_3X_4$$

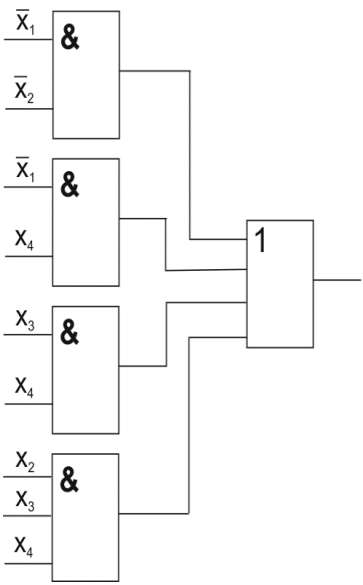


Рис. 7.8

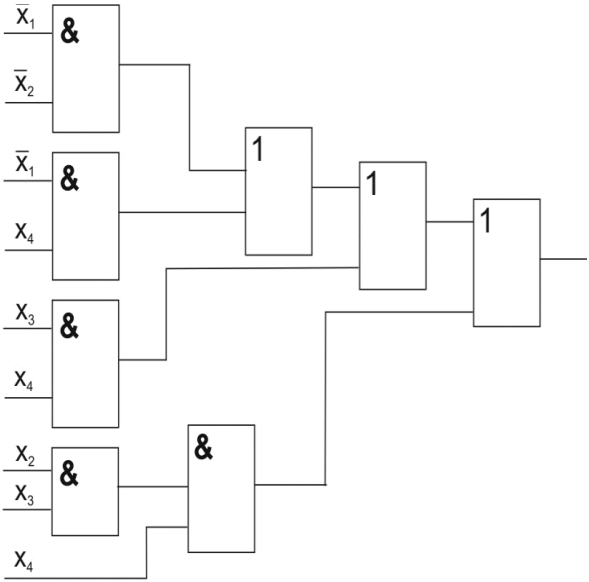


Рис. 7.9

Якщо не враховувати коефіцієнт об'єднання по входу, то така булева функція може бути реалізована дворівневою КС (рис. 7.8), що є оптимальним варіантом за

швидкодією. Однак, якщо коефіцієнт об'єднання по входу $I = 2$, то запропонована реалізація повинна бути видозмінена. Перетворення КС в булевом базисі зводиться до простого поділу змінних на елементах І і АБО на основі асоціативності операцій кон'юнкції і диз'юнкції. Перетворена КС представлена на рис. 7.9. Як випливає з рис.7.8, 7.9, виконане перетворення призводить до необхідності використання чотирирівневої КС, що знижує швидкодію схеми. У загальному випадку, бажано проектувати КС мінімальної глибини при виконанні вимог на величину коефіцієнта І для використовуваного комплекту ІМС. Глибину КС можна зменшити, якщо від мінімальної ДНФ перейти до дужкової форми, виносячи загальні члени ДНФ за дужки.

У ряді випадків зменшити глибину КС при збереженні величини коефіцієнта I можна, використовуючи перехід до змішаного базису, якщо в комплекті ІМС, використовуваному для проектування КС, є відповідні елементи.

Коефіцієнт об'єднання по входу U (коефіцієнт розгалуження) деякого логічного елемента характеризує максимально можливу кількість елементів схеми, входи яких можуть бути підключені до його виходу. Коефіцієнт U є однією з технічних характеристик комплекту ІМС і не пов'язаний з логікою роботи ІМС. Якщо деякий логічний елемент α з коефіцієнтом розгалуження $U\alpha$, підключений до входів n α > $U\alpha$ логічних елементів, то вважається, що елемент α перевантажений. У цьому випадку необхідно так структурно перетворити КС, можливо шляхом введення в неї деяких додаткових елементів, щоб число навантажень на елемент було менше $U\alpha$. У коректно побудованої КС для всіх її елементів α_i повинно виконуватися умова відсутності перевантажень $n\ \alpha_i \le U\alpha_i$ Таким чином, розрахунок КС за коефіцієнтом розгалуження зводиться до визначення перевантажених елементів і усунення перевантажень.

При побудові КС з урахуванням коефіцієнта об'єднання по виходу найбільшого поширення набули два способи усунення перевантажень:

1) **введення розв'язуючих підсилювачів;**

2) дублювання перевантажених елементів.

Розглянемо їх застосування на конкретних прикладах. Нехай задана КС (рис. 7.10) в монофункціональному базисі І - НІ з коефіцієнтом об'єднання по виходу U = 2. Елемент 2 у схемі перевантажений, так як його вихід підключений до входів трьох логічних елементів КС. Установка розв'язуючого підсилювача, що складається з двох послідовно з'єднаних елементів І-НІ (рис. 7.11), на виході елемента 2 усуває перевантаження.

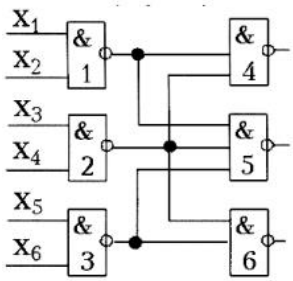


Рис. 7.10

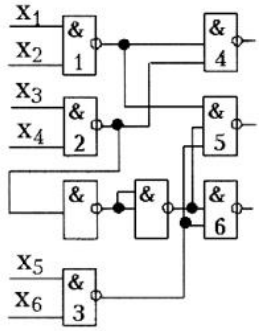


Рис. 7.11

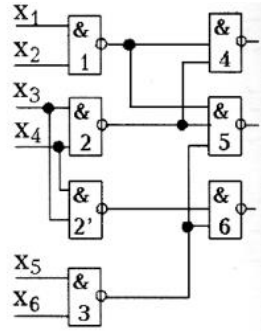


Рис. 7.12

Інший спосіб усунення перевантажень елемента 2 зводиться до його дублювання (рис. 7.12). Однак дублювання логічного елемента підвищує навантаження на його входи, що може призвести до перевантаження логічних елементів, що з'єднуються з входами дубльованого. Її усунення пов'язано з введенням нових дубльованих елементів і т. д. Необхідно відзначити, що дублювання елементів не вносить додаткових затримок в КС і може виявитися корисним при організації контролю правильності її функціонування. Введення підсилювачів завжди вносить в КС додаткову затримку поширення сигналу. У тих випадках, коли потрібно зберегти задану швидкодію КС і отримати оптимальний варіант за кількістю елементів, що вводяться, застосовують комбінований метод, тобто вводять підсилювачі там, де можливо (при збереженні заданої швидкодії КС), а елементи, які залишилися перевантажені, дублюють.

ЧАСТИНА 2

8 АБСТРАКТНІ ЦИФРОВІ АВТОМАТИ

8.1 Основні поняття

Цифровий автомат можна трактувати як пристрій, що здійснює прийом, зберігання і перетворення дискретної інформації по деякому алгоритму. З певної точки зору до автоматів можна віднести як реальні пристрої (ЕОМ), так і абстрактні системи (математичні моделі).

Автоматом називається дискретний перетворювач інформації, здатний приймати різні стани, переходити під впливом вхідних сигналів з одного стану в інший і видавати вихідні сигнали.

Загальна теорія автоматів підрозділяється на абстрактну і структурну.

Абстрактна теорія автоматів, відволікаючись від структури автомата (тобто не цікавлячись способом його побудови), вивчає лише поведінку автомата щодо зовнішнього середовища. Абстрактна теорія автоматів близька до теорії алгоритмів, будучи по суті її подальшою реалізацією.

На противагу абстрактної теорії автоматів, структурна теорія автоматів цікавиться як структурою самого автомата, так і структурою вхідних впливів і реакцій автомата на них. У структурній теорії вивчаються способи побудови автоматів, способи кодування вхідних впливів і реакцій автомата на них. Структурна теорія автоматів є продовженням і розвитком абстрактної теорії. Спираючись на апарат булевих функцій і на абстрактну теорію автоматів, структурна теорія дає ефективні рекомендації з розробки реальних пристроїв обчислювальної техніки.

Абстрактний цифровий автомат (ЦА) є математичною моделлю дискретного керуючого пристрою. Абстрактний ЦА визначається множиною, що складається з шести елементів:

$S=\{ X, A, У, \delta, \lambda, a_o\}$, де:

$X=\{x_1, x_2,... x_n\}$ - множина вхідних сигналів (вхідний алфавіт);

$У=\{y_1, y_2, ... y_m\}$ - множина вихідних сигналів (вихідний алфавіт);

$A=\{ a_0, a_1, a_2,... a_N\}$ - множина станів (алфавіт станів);

a_o- початковий стан ($a_o \in A$);

δ - функція переходів автомата, яка задає відображення (ХхА) $\rightarrow$ А, тобто ставить у відповідність будь-якій парі елементів декартового добутку (ХхА) елемент множини А;

λ - функція виходів автомата, яка задає або відображення (ХхА) $\rightarrow$ У, або відображення А $\rightarrow$ У.

Іншими словами, функція переходів δ показує, що автомат S, знаходячись в деякому стані $a_i \in A$, при появі вхідного сигналу $x_j \in X$ переходить в деякий стан $a_p \in A$. Це можна записати:

$a_p = \delta\ (a_i, x_j)$.

Функція виходів λ показує, що автомат S, який знаходиться в деякому стані $a_i \in A$, при появі вхідного сигналу $x_j \in X$ видає вихідний сигнал $y_k \in Y$. Це можна записати: $y_k = \lambda\ (a_i, x_j)$.

Автомат (рис. 8-1) має один вхід и один вихід. Автомат працює в дискретному часі, який приймає цілі невід'ємні значення t = 0, 1, 2,.... В кожен момент t дискретного часу автомат знаходиться в деякому стані a (t) із множини станів автомата, причому в початковий момент t = 0 він завжди знаходиться в початковому стані a (0) = a_0. В момент t, будучи в стані a (t), автомат здатний сприйняти на вході букву вхідного алфавіту x (t) ∈ X. У відповідності з функцією виходів він видасть в той же момент часу t букву вихідного алфавіту y (t) = λ(a (t), x(t)) та у відповідності з функцією переходів перейде в наступний стан a (t+ 1) = δ(a(t), x (t)); a (t) ∈ A, y (t) ∈ У. Зміст поняття абстрактного автомата полягає в тому, що він реалізує деяке відображення множини слів вхідного алфавіту Х в множину слів вихідного алфавіту У. Інакше кажучи, якщо на вхід автомата, встановленого в початковий стан a_1 подавати буква за буквою деяку послідовність літер вхідного алфавіту x (0), x (1), x (2), ... - вхідне слово, то на виході автомата будуть послідовно з'являтися літери вихідного алфавіту y (0), y (1), y (2), … - вихідне слово. Відносячи до кожного вхідного слова відповідне йому вихідне слово, ми отримаємо відображення φ, індуковане абстрактним автоматом.

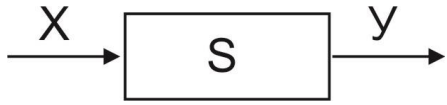


Рис. 8.1. Абстрактний автомат

Таким чином, на рівні абстрактної теорії поняття «робота автомата» розуміється як перетворення вхідних слів у вихідні слова. Можна сказати, що в

абстрактному автоматі ми відволікаємося від його структури - вмісту прямокутника на рис 8-1, розглядаючи його як «чорний ящик» (прийнятий в кібернетиці підхід, коли основна увага приділяється поведінці системи щодо зовнішнього середовища). Спочатку ми будемо розглядати автомат - не більше ніж перетворювач слів, і лише в подальшому ми займемося начинкою цього ящика - питанням, як реалізувати поведінку, задану на рівні абстрактного автомата, за допомогою наявних в нашому розпорядженні елементів. Виявляється, однак, що таке абстрагування від структури пристрою дозволяє вирішити низку складних проблем, які на структурному рівні ховаються за множиною деталей, несуттєвих з точки зору поведінки всієї системи в цілому. У якомусь сенсі поняття абстрактного автомата в теорії дискретних пристроїв втілило в собі системний підхід, при якому предмет або явище розглядається як щось цілісне і основний акцент у дослідженні робиться на виявленні різноманіття зв'язків і відносин із зовнішнім середовищем, на відміну від структурного підходу, при якому властивості об'єкта визначаються властивостями елементів, з яких він складається.

Поняття стану у визначенні автомата було введено у зв'язку з тим, що часто виникає необхідність в описі поведінки систем, виходи яких залежать не тільки від стану входів в даний момент часу, але і від деякої передісторії, тобто від сигналів, які надходили на входи системи раніше. Стан якраз і відповідає деякій пам'яті про минуле, дозволяючи усунути час як явну змінну і виразити вихідні сигнали як функцію станів і входів в даний момент часу.

Основну увагу будемо приділяти вивченню автоматів, число станів яких не менше двох, тобто автоматів з пам'яттю. Відомий клас автоматів, у яких вихід не залежить від передісторії і в кожен момент часу визначається лише вхідним сигналом в цей же момент часу - так звані комбінаційні схеми. Цей клас автоматів можна розглядати на абстрактному рівні як автомат з одним станом. Очевидно, що комбінаційну схему також можна представити у вигляді рис. 8.1 і описувати трійкою $S = (X, У, \lambda)$, де X та У - вхідний та вихідний алфавіти відповідно, а λ: $X \rightarrow У$ - функція виходів. Комбінаційні схеми ми іноді будемо називати функціональними перетворювачами і зображати їх так, як це показано на рисунку 8.2.

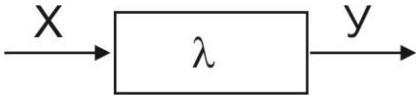


Рис. 8.2. Функціональний перетворювач

Абстрактний автомат функціонує в дискретному автоматному часі t=0,1,2,... і переходи із стану в наступний стан здійснюються миттєво. В кожен момент t дискретного часу автомат знаходиться в певному стані a(t) із множини A станів автомата, причому в початковий момент часу t=0 він завжди знаходиться в початковому стані a_0. В момент часу t, будучи в стані a(t), автомат здатний сприйняти на вхідному каналі сигнал x(t) $\in$ X, та видати на вихідному каналі сигнал y(t)= λ(a(t), x(t)), переходячи в стан a(t+1)= δ (a(t),x(t)). Залежність вихідного сигналу від вхідного і від стану означає, що в його складі є пам'ять.

8.2 Типи абстрактних автоматів

За способом формування функції виходів виділяють три типи абстрактних автоматів: автомат Мілі, автомат Мура, С-автомат.

На практиці найбільшого поширення набули два класи автоматів - автомати Мілі та Мура, названі по імені американських вчених G. H. Mealy і E. F. Moore, які вперше досліджували ці моделі. Закон функціонування автомата Мілі задається системою рівнянь:

y(t)= λ (a(t), x(t));

a(t+1)= δ(a(t), x(t)). (8-1)

Автомат Мура - системою рівнянь:

y(t)= λ (a(t));

a(t+1)= δ(a(t), x(t)). (8-2)

С-автомат - системою рівнянь:

y= y1 ∪ y2

y1(t)= λ_1(a(t),x(t));

У2(t)= λ_2 (a(t));

a(t+1)=δ(a(t),x(t)).

Довільний абстрактний автомат Мілі або Мура (рис. 8.3.) має один вхідний і один вихідний канали. Довільний абстрактний С-автомат має один вхідний і два вихідних канали (Рисунок. 8.4.).

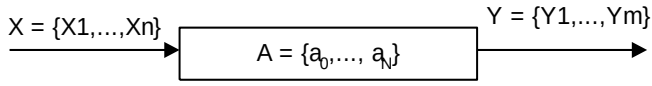


Рис. 8.3- Абстрактний автомат

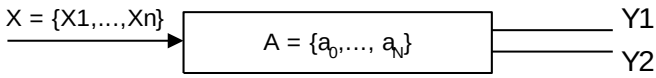


Рис.. 8.4 Абстрактний С-автомат

Якщо на вхід абстрактного автомата Мілі або Мура, встановленого в початковий стан a_o, подавати літера за літерою деяку послідовність літер вхідного алфавіту x(0), x(1), ...- вхідне слово, то на виході автомата будуть послідовно з'являтися літери вихідного алфавіту y(0), y(1),...- вихідне слово. Для випадку С-автомата на його виходах будуть з'являтися дві послідовності: $y_1(0)$, $y_1(1)$,... та $y_2(0)$, $y_2(1)$,.... В абстрактному С- автоматі вихідний сигнал $y_2(t)= \lambda_2 (a(t))$ видається весь час, поки автомат знаходиться в стані a(t). Вихідний сигнал $y_1(t)=\lambda_1(a(t),x(t))$ видається під час дії вхідного сигналу x (t) при перебуванні С-автомата в стані a(t).

Таким чином, на рівні абстрактної теорії функціонування цифрового автомата розуміється як перетворення вхідних слів у вихідні слова.

Автомат називається кінцевим, якщо кінцеві множини А, Х і У. В подальшому будуть розглядатися тільки кінцеві автомати і термін «кінцевий» майже завжди буде опускатися.

Виділяють повністю визначені і часткові автомати.

Повністю визначеним називається абстрактний цифровий автомат, у якого функція переходів або функція виходів, або обидві ці функції визначені для всіх пар переходів (x_i,a_j).

Частковим називається абстрактний цифровий автомат, у якого функція переходів або функція виходів, або обидві ці функції визначені не для всіх пар переходів (x_i,a_j).

Абстрактний цифровий автомат називається **ініціальним**, якщо на множині його станів А виділяється початковий стан a_o.

8.3 Способи завдання абстрактних автоматів

Щоб задати кінцевий автомат S, необхідно описати всі елементи множини: $S=\{ X, A, У, \delta, \lambda, a_o\}$, тобто вхідний і вихідний алфавіти і алфавіт станів, а також функції переходів і виходів. Серед множини станів необхідно виділити стан a_1, в якому автомат знаходиться в момент t = 0.

Існує кілька способів завдання роботи автомата, але найбільш часто використовується табличний (матричний), графічний.

При табличному способі автомат задається двома таблицями: таблицею переходів і таблицею виходів, або матрицею з'єднань. Таблиця переходів довільного повністю визначеного автомата будується таким чином: рядки таблиці позначаються буквами вхідного алфавіту автомата, а стовпчики таблиці - літерами алфавіту станів автомата, причому крайній лівий стовпець позначений початковим станом a_1. клітині таблиці переходів, що знаходиться на перетині рядка, зазначеного вхідним сигналом x_i, стовпчика зазначеного станом a_j, ставиться стан a_k, що є результатом переходу автомата зі стану a_j під дією вхідного сигналу x_i, що визначається виразом $a_k = \delta(a_j, x_j)$.

Таблиця 8.1. Таблиця переходів автомата

Стани / Вхідні сигнали	a_1	a_2	…	a_k
x_1	$\delta(a_1,x_1)$	$\delta(a_2, x_1)$	...	$\delta(a_к, x_1)$
…				
x_j	$\delta(a_1,x_j)$	$\delta(a_2, x_j)$	⋯	$\delta(a_к, x_j)$

Приклад заповнення таблиці переходів деякого абстрактного повністю визначеного автомата з вхідним алфавітом $X=\{x_1, x_2\}$- та алфавітом станів $A=\{a_1, a_2, a_3\}$ представлений в табл. 8.2.

Таблиця 8.2.

A / X	a_1	a_2	a_3
x_1	a_2	a_3	a_2
x_2	a_3	a_2	a_1

Якщо абстрактний автомат частковий, то в клітині таблиці його переходів, що знаходиться, на перетині рядка, відміченого вхідним сигналом і стовпчика відміченого відповідним станом (за умови, що перехід в цей стан під дією даного вхідного сигналу не визначений) ставиться прочерк, і будь-яке вхідне слово, що приводить до зазначеного переходу є забороненим.

Заповнення інших клітин аналогічно випадку повністю визначеного автомата. Вид таблиці переходів не залежить від типу автомата, що задається

автомата (автомат Мілі, Мура, С-автомат). Таблиці виходів автоматів Мілі, Мура, С-автомата мають відмінності.

Таблиця виходів повністю визначеного автомата Мілі будується таким чином: ідентифікація стовпців і рядків, а також формат таблиці відповідають таблиці переходів повністю визначеного автомата. У клітинці таблиці виходів, що знаходиться на перетині рядка, зазначеного вхідним сигналом x_j, та стовпчика, відзначеного станом $a_к$, ставиться вихідний сигнал y_m, який автомат видає, перебуваючи в стані a_k при наявності вхідного сигналу x_j, що визначається виразом $y_m=\lambda(a_k, x_j)$.

Таблиця 8.3. Таблиця виходів абстрактного автомата Мілі.

Стани / Вхідні сигнали	a_1	a_2		a_k
x_1	$\lambda(a_1,x_1)$	$\lambda(a_2, x_1)$		$\lambda(a_k, x_1)$
…		….	...	
x_j	$\lambda(a_1,x_j)$	$\lambda(a_2, x_j)$		$\lambda(a_k, x_j)$

Приклад заповнення таблиці виходів деякого абстрактного повністю визначеного автомата з вхідним алфавітом $X=\{x_1, x_2\}$, алфавітом станів $A=\{a_1, a_2,a_3\}$ та вихідним алфавітом $У=\{y_1, y_2, y_3\}$- представлено в табл. 8.4.

Таблиця 8.4

A / X	a_1	a_2	a_3
x_1	y_1	y_3	y_3
x_2	y_2	y_1	y_1

Таблиця виходів повністю визначеного автомата Мура будується більш просто: кожному стану автомата ставиться у відповідність свій вихідний сигнал. Приклад таблиці виходів автомата Мура з алфавітом станів $A=\{a_1, a_2, a_3, a_4, a_5\}$, вхідним алфавітом $X=\{x_1, x_2\}$ та вихідним алфавітом $У=\{y_1, y_2, y_3\}$ представлено в таблиці 8.5.

Таблиця 8.5.

A	a_1	a_2	a_3	a_4	a_5
У	y_1	y_1	y_3	y_2	y_3

Так як в автоматі Мура вихідний сигнал залежить тільки від стану, автомат Мура задається однією відміченою таблицею переходів, в якій кожному її стовпцю приписаний крім стану a_m ще і вихідний сигнал $y_g = \lambda (a_m)$, який відповідає цьому стану. Приклад табличного завдання автомата Мура представлено в таблиці 8.6.

Таблиця 8.6.

У	y_1	y_1	y_3	y_2	y_3
A	a_1	a_2	a_3	a_4	a_5
x_1	a_2	a_5	a_5	a_3	a_3
x_2	a_4	a_2	a_2	a_1	a_1

Очевидно, абстрактний повністю визначений С-автомат задається двома таблицями виходів, перша з яких є таблиця виходів автомата Мілі, а друга - Мура. Якщо автомат частковий, то в деяких клітинах його таблиці може стояти прочерк, що означає відсутність вихідного сигналу.

Іноді при завданні автоматів Мілі використовують одну поєднану таблицю переходів і виходів, в якій на перетині стовпця a_m та рядка x_f записується наступний стан і вихідний сигнал, який формується на даному переході: a_s/y_g ; $a_s = \delta(a_m, x_f)$, $y_g = \lambda(a_m, x_f)$.

Приклади відмічених таблиць переходів представлені в таблицях 8.7. -8.9. (Загальний вигляд відміченої таблиці переходів - таблиця 8.7., відмічена таблиця переходів автомата Мілі - таблиця 8.8., відмічена таблиця переходів автомата Мура - таблиця 8.9.)

Крім розглянутих вище таблиць переходів і виходів довільний абстрактний автомат може бути заданий матрицею з'єднань.

Матриця з'єднань є квадратною і містить стільки стовпців (рядків), скільки різних станів містить алфавіт станів даного автомата. Кожен стовпець (рядок) матриці з'єднань позначається буквою стану автомата. У клітинці, що знаходиться на перетині рядка, позначеного буквою a_j і рядка, позначеного буквою a_s автомата, ставиться вхідний сигнал (або диз'юнкція вхідних сигналів), під впливом якого (яких) здійснюється даний перехід.

Для абстрактного автомата Мілі в клітинці поруч із станом проставляється також вихідний сигнал, який автомат видає в результаті даного переходу (таблиця

8.10.) Для автомата Мура вихідний сигнал проставляється в рядку поруч із станом (ці стани відповідають вихідним станам автомата).

Таблиця 8.7

Стани / Вхідні сигнали	a_1	a_2	…	a_k
x_1	$\delta(a_1,x_1)$	$\delta(a_2, x_1)$	…	$\delta(a_k, x_1)$
	$\lambda(a_1,x_1)$	$\lambda(a_2, x_1)$	…	$\lambda(a_k, x_1)$
…	…	…	…	…
x_j	$\delta(a_1,x_j)$	$\delta(a_2,x_j)$	…	$\delta(a_k, x_j)$
	$\lambda(a_1,x_j)$	$\lambda(a_2, x_j)$	…	$\lambda(a_k, x_j)$

Таблиця 8.8.

A / X	a_1	a_2	a_3
x_1	a_2/y_2	a_3/y_3	a_1/y_3
x_2	a_1/y_3	a_1/y_1	a_2/y_2

Таблиця 8.9.

Y	y_1	y_2	y_3
A / X	a_1	a_2	a_3
x_1	a_2	a_3	a_1
x_2	a_1	a_1	a_2

Таблиця 8.10.

	a_1		a_n
a_1	$x_j(y_k)$		
a_n			$x_j(y_m)$

При графічному способі завдання абстрактні автомати представляються орієнтованими графами.

Графом автомата називається орієнтований зв'язний граф, вершини якого відповідають станам автомата, а дуги між ними - переходам між станами.

Дві вершини a_m та a_s (початковий стан і стан переходу) з'єднуються дугою, спрямованою від a_m до a_s в тому випадку, якщо в автоматі є перехід зі стану a_m в стан a_s, тобто, якщо $a_s = \delta(a_m, x_f)$ при деякому $x_f. \in X$. Дузі (a_m, a_s) графа автомата приписується вхідний сигнал x_f і вихідний сигнал $y_g = \lambda(a_m, x_f)$, якщо він

визначений, і ставиться прочерк в іншому випадку. Якщо перехід автомата зі стану a_m в стан a_s відбувається під дією декількох вхідних сигналів, то дузі (a_m, a_s) приписуються всі ці вхідні і відповідні вихідні сигнали. При описанні автомата Мура у вигляді графа вихідний сигнал $y_g = \lambda (a_m)$ записується всередині вершини a_m або поряд з нею.

І так, для автомата Мілі вхідний і вихідний сигнали проставляються на дузі, яка відповідає даному переходу (Рисунок 8.5.). Для автомата Мура вхідний сигнал проставляється на дузі, а вихідний - поруч з вершиною, що відповідає даному стану (Рисунок 8.6.).

Тут розглядаються тільки детерміновані автомати, у яких виконуються умови однозначності переходів: автомат, що знаходиться в деякому стані, під дією будь-якого вхідного сигналу не може перейти більш ніж в один стан.

Автомат, заданий таблицею переходів, завжди детермінований, оскільки на перетині стовпця a_m і рядка x_f записується тільки один стан $a_s = \delta (a_m, x_f)$, якщо перехід визначений, і ставиться прочерк, якщо функція δ на парі (a_m, x_f), не визначена.

Стосовно до графічного способу завдання автомата умова однозначності означає, що в графі автомата з будь-якої вершини не можуть виходити дві і більше дуги, відмічені одним і тим же вхідним сигналом.

Виділення в множині станів початкового стану пояснюється чисто практичними міркуваннями, пов'язаними з виникаючою часто необхідністю фіксувати умови початку роботи дискретного пристрою. Багато ж завдань на рівні абстрактного автомата можна вирішувати, описуючи автомат $S = (A, X, Y, \delta, \lambda)$.

Автомат $S = (A, X, Y, \delta, \lambda, a_0)$, представляється шісткою, тобто, з виділеним початковим станом, називається ініціальним.

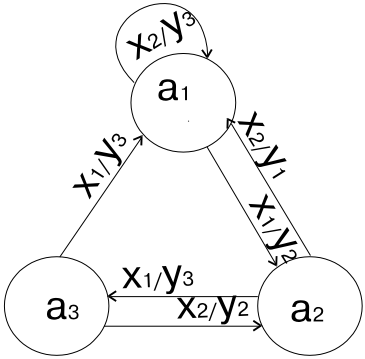


Рис. 8.5-Граф автомата Мілі, (заданий таблицею 8.8.)

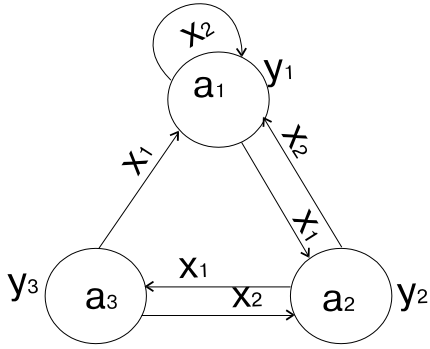


Рис. 8.6-Граф автомата Мура S2 (заданий таблицею 8.9.)

При аналітичному способі завдання абстрактні автомати задаються четвіркою об'єктів:

S={A, X, У, F}, де F задає для кожного стану a_j автомата відображення (XxA) → (AxY). Іншими словами, при аналітичному способі завдання для кожного стану автомата a_j вказується відображення Fa_i, яке представляє собою множину всіх трійок (a_p, x_m, y_k), причому таких, що під впливом вхідного сигналу x_m автомат переходить із стану a_j в стан a_p, видаючи при цьому вихідний сигнал y_k. Стосовно до загального визначенням абстрактного автомата, останнє рівнозначно опису функцій δ і λ у відповідності з виразом: $a_p = \delta(a_i, x_m)$, $y_к = \lambda(a_i, x_m)$.

Відображення Fa_i записується в такий спосіб:

$Fa_i\{a_p(x_m/y_k), a_i(x_f/y_z)\ldots\}$.

Для абстрактного автомата Мілі (табл. 8.7.) Аналітичне завдання має наступний вигляд:

S={ X,A,Y,F}, $X=\{x_1,x_2\}$, $A=\{a_1, a_2, a_3\}$, $Y=\{y_1, y_2, y_3\}$,

$Fa_1=\{a_2(x_1/y_2), a_1(x_2/ y_3)\}$,

$Fa_2=\{a3(x_1/y_3), a_1(x_2/y_1)\}$,

$Fa_3=\{a1(x_1/ y_3), a_2(x_2/y_2)\}$.

Слід зазначити, що функція **Fa_i завжди записується для вихідного (початкового) стану**.

Розширимо функції δ и λ, визначивши їх на множині пар стан - вхідне слово. Нехай $\xi = xi_1 \ldots xi_k$ - вхідне слово довжини k, E - множина всіх кінцевих вхідних слів ненульовий довжини, e - вхідне слово довжини нуль (пусте слово). Тоді функцію переходів стану $\tilde{\delta}(a_m, \xi)$ визначимо індуктивно в множині A × (E ∪ i {e}) наступним чином:

$\tilde{\delta}(a_m, e) = a_m$ для всіх $a_m \in A$;

$$\tilde{\delta}(a_m, \xi) = \tilde{\delta}(a_m, x_{i1} \ldots x_{ik}) =$$

$$= \begin{cases} \underbrace{\delta(\tilde{\delta}(a_m, x_{i1} \ldots x_{ik-1}), x_{ik})}_{a_{ik}} = \delta(a_{ik}, x_{ik}), \text{если } \delta(a_{ij}, x_{ij})\ a_{i1} = a_m \\ \qquad\qquad \text{визначена для всіх } j = 1, \ldots, k; \\ \text{не визначена в противному випадку} \end{cases}$$

Як приклад розглянемо частковий автомат, заданий суміщеної таблицею 8.11. знайдемо $\tilde{\delta}(a_2, \xi)$ при $\xi = x_1\ x_1\ x_2\ x_1\ x_2$:

Таблиця 8.11.

A / X	a_1	a_2	a_3	a_4
x_1	a_2/y_1	a_3/y_3	a_4/y_3	-
x_2	a_3/y_2	-	$a_2/-$	a_2/y_2

Знайдемо $\tilde{\delta}(a_2, \xi)$ при $\xi = x_1\, x_1\, x_2\, x_1\, x_2$

$\tilde{\delta}(a_2, x_1\, x_1\, x_2\, x_1\, x_2) = \tilde{\delta}(\delta(a_2, x_1), x_1\, x_2\, x_1\, x_2) = \tilde{\delta}(a_3, x_1\, x_2\, x_1\, x_2) =$

$= \tilde{\delta}(\delta(a_3, x_1),\ x_2\, x_1\, x_2) = \tilde{\delta}(a_4,\ x_2\, x_1\, x_2) = \tilde{\delta}(\delta(a_4, x_2), x_1\, x_2) =$

$= \tilde{\delta}(a_2, x_1\, x_2) = \tilde{\delta}(\delta(a_2, x_1),\ x_2) = \tilde{\delta}(a_3, x_2) = \delta(a_3, x_2) = a_2.$

Однак, для того автомату $\tilde{\delta}(a_2, x_1\, x_1\, x_2\, x_1\, x_2) = \tilde{\delta}(\delta(a_2, x_1), x_1\, x_2\, x_2\, x_2) =$

$\tilde{\delta}(\delta(a_3, x_1),\ x_2\, x_2\, x_2) = \tilde{\delta}(\delta(a_4, x_2),\ x_2\, x_2) = \tilde{\delta}(\delta(a_2, x_2), x_2)$ не визначена, так як не визначена функція переходів $\delta(a_2, x_2)$.

Таким чином, $\tilde{\delta}(a_m, \xi)$ - заключний стан, в який переходить автомат зі стану a_m під дією вхідного слова ξ.

Функція заключного виходу $\tilde{\lambda}(a_m, \xi)$ в моделі Мура визначена в множині $A \times (E \cup \{e\})$ наступним чином:

$\tilde{\lambda}(a_m, \xi) = \lambda(a_m);$

$$\tilde{\lambda}(a_m, \xi) = \begin{cases} \lambda\left(\tilde{\delta}(a_m, \xi)\right) \text{ для } \xi \in E, \text{якщо } \tilde{\delta}(a_m, \xi) \text{ визначена} \\ \text{не визначена, если не визначена } \tilde{\delta}(a_m, \xi) \end{cases}$$

В моделі Мілі:

$$\tilde{\lambda}(a_m, \xi) = \begin{cases} \lambda\left(\tilde{\delta}(a_m, \xi'),\ x_{i_k}\right) \text{ для } \xi = \xi' x_{i_k}, \text{ якщо } \tilde{\delta}(a_m, \xi') \text{ визначена;} \\ \text{не визначена, если не визначена } \tilde{\delta}(a_m,\ \xi') \end{cases}$$

Тут $\xi = x_{i_1} \ldots x_{i_{k-1}}, x_{i_k},\ \xi' = x_{i_1} \ldots x_{i_{k-1}}$

Таким чином, в моделі Мура $\tilde{\lambda}(a_m, \xi)$ - вихідний сигнал, що відзначає заключний стан. У моделі Мілі $\tilde{\lambda}(a_m, \xi)$ – вихідний сигнал, що видається на останньому переході (перехід в заключний стан). Він з'являється на виході в той же момент часу, коли приходить остання буква вхідного слова ξ.

Знайдемо, наприклад, $\tilde{\lambda}(a_1,\ x_1 x_2\, x_1\, x_1)$ для автомата Мілі, заданого таблицею 8.12:

Таблиця 8. 12

X \ A	a_1	a_2	a_3
x_1	a_2/y_1	a_3/y_3	$a_2/y3$
x_2	a_3/y_2	a_2/y_1	a1/y1

$\tilde{\delta}(a_1, x_1\, x_2\, x_1\, x_1) = \tilde{\delta}(\delta(a_1, x_1), x_2\, x_1\, x_1) = \tilde{\delta}(\delta(a_2, x_2),\ x_1\, x_1) =$

$= \tilde{\delta}(\delta(a_2, x_1), x_1) = \tilde{\delta}(a_3, x_1) = \delta(a_3, x_1) = a_2.$

Останнім у цій послідовності переходів є перехід (a_3, x_1), а тому:

$\tilde{\lambda}(a_1, x_1 x_2\, x_1\, x_1) = \lambda(a_3, x_1) = y_3.$

Функцію $\omega(a_m, \xi)$ - реакцію автомата в стані a_m на вхідне слово $\xi = x_{i_1} \ldots x_{i_{k-1}}, x_{i_k}$ визначимо в множині A × (E ∪ {e}) наступним чином.

Модель Мілі: $\omega(a_m, \xi)$ не визначена, якщо $\tilde{\delta}(a_m, \xi')$ не визначена; тут $\xi' = x_{i_1} \ldots x_{i_{k-1}}$, тобто $\xi = \xi' x_{i_k}$. Інакше, якщо $\tilde{\delta}(a_m, \xi')$ визначена, то

$\omega(a_m, \xi) = \lambda(a_m, x_{i_1})\, \omega(\delta(a_m, x_{i_1}), x_{i_2} \ldots x_{i_k}) =$

$= \lambda(a_m, x_{i_1})\, \lambda(a_{i_2}, x_{i_2}) \ldots \lambda(a_{i_k}, x_{i_k}) = y_{i_1} \ldots y_{i_k}.$

Тут $a_{i_2} = \delta(a_m, x_i)$; $y_{i_j} = \lambda(a_{i_j}, x_{i_j})$; j=2, 3, …, k; $y_{i_1} = \lambda(a_m, x_{i_1})$.

Наприклад, для автомата (таблиця 8.12):

$\omega(a_1,\ x_1\, x_2\, x_1\, x_1) = \lambda(a_1,\ x_1)\, \omega(\delta(a_1,\ x_1), x_2\, x_1\, x_1) =$

$= y_1 \omega(a_2,\ x_2\, x_1\, x_1) = y_1 \lambda(a_2,\ x_2)\, \omega(\delta(a_2, x_2)\, x_1\, x_1) = y_1 y_1 \omega(a_2, x_1\, x_1) =$

$= y_1 y_1 \lambda(a_2,\ x_1)\, \omega(\delta(a_2,\ x_1), x_1) = y_1 y_1 y_2 \lambda(a_3,\ x_1) = y_1 y_1 y_3 y_3$

Неважко бачити, що для часткового автомата Мілі (таблиця 8.11) $\omega(a_2,\ x_1\, x_1\, x_2\, x_2) = y_3 y_3 y_2 y_3 -$; прочерк на останньому місці означає, що в реакції останній вихідний сигнал не визначений. Сама ж реакція визначена, так як $\tilde{\delta}(a_2, x_1\, x_1\, x_2\, x_1)$ визначена.

Однак, для того ж автомата $\omega(a_2,\ x_1\, x_1\, x_2\, x_2\, x_2)$ не визначена, так як не визначена функція $\tilde{\delta}(a_2, x_1\, x_1\, x_2\, x_2)$.

Можна визначити реакцію автомата Мілі в стані a_m на вхідне слово ξ і через заключний вихід:

$\omega(a_m, \xi) = \tilde{\lambda}(a_m, x_{i_1})\, \tilde{\lambda}(a_m, x_{i_1} x_i) \ldots \tilde{\lambda}(a_m, x_{i_1} \ldots x_{i_{1k}})$.

Модель Мура: $\omega(a_m, \xi)$ не визначена, якщо $\tilde{\delta}(a_m, \xi)$ не визначена. Інакше, якщо $\tilde{\delta}(a_m, \xi)$ визначена, то

$\omega(a_m, \xi) = \lambda(\delta(a_m, x_{i_1}))\, \omega(\delta(a_m, x_{i_1}), x_{i_2} \ldots x_{i_k}) =$

$= \lambda(a_{i_2})\, \lambda(a_{i_3}) \ldots \lambda(a_{i_{k+1}}) = y_{i_2}\, y_{i_3} \ldots y_{i_{k+1}}$

Тут $a_{i_2} = \delta(a_m, x_{i_1})$; $y_{i_j} = \lambda(a_{i_j})$, j=2, …, k+1.

Визначимо, наприклад, реакцію $\omega(a_1, x_2 x_1 x_1 x_2)$ автомата Мура, заданого зазначеною таблицею переходів (таблиця 8.13.)

Таблиця 8.13

Y	y_1	y_1	y_3	y_2	y_3
A / X	a_1	a_2	a_3	a_4	a_5
x_1	a_2	a_5	a_5	a_3	a_3
x_2	a_4	a_2	a_2	a_1	a_1

$\omega(a_1, x_2x_1x_1x_2) = \lambda(\delta(a_1, x_2))\ \omega(\delta(a_1, x_2), x_1x_1x_2) = \lambda(a_1)\ \omega(a_4, x_1x_1x_2) =$
$= y_2\lambda(\delta(a_4, x_1))\ \omega(\delta(a_4, x_1), x_1x_2) = y_2\lambda(a_2)\ \omega(a_2, x_1x_2) =$
$= y_2y_3\lambda(\delta(a_3, x_1))\ \omega(\delta(a_3, x_1), x_2) = y_2y_3\lambda(a_5)\ \omega(a_5, x_2) =$
$= y_2y_3y_3\lambda(\delta(a_5, x_2)) = y_2y_3y_3\lambda(a_1) = y_2y_3y_3y_1.$

Очевидно, що в автоматі Мура вихідний сигнал $y_{i_1} = \lambda(a_m)$ в момент часу i_1 не залежить від вхідного сигналу x_{i_1}, а визначається тільки станом a_m. Таким чином, цей сигнал y_{i_1} ніяк не пов'язаний з вхідним словом, що надходить на вхід автомата, починаючи з моменту i_1. У зв'язку з цим під реакцією автомата Мура в стані a_m на вхідне слово $\xi = x_{i_1} x_{i_2} \dots x_{i_k}$ розуміється вихідне слово тієї ж довжини $\omega(a_m, \xi) = y_{i_2} y_{i_3} \dots y_{i_{k+1}}$, але зсунуте на один такт автоматного часу.

Зведемо разом три функції:

Модель Мілі

Вхідне слово	x_{i_1}	x_{i_2}	…	x_{i_k}	
Послідовність станів	am	$a_{i_2} == \delta(a_m, x_{i_1})$	…	$a_{i_k} == (\delta(a_{i_{k-1}} x_{i_{k-1}})$,	$\delta(a_m, \xi)$ $a_{i_{k+1}} == \delta(a_{i_k}, x_{i_k})$
Реакція	$y_{i_1} = \lambda(a_m, x_{i_1})$	$y_{i_2} = \lambda(a_{i_2}, x_{i_2})$	…	$y_{i_k} = \lambda(a_{i_k}, x_{i_k})$	
				$\tilde{\lambda}(a_m, \xi)$	
	$\omega(a_m, \xi)$				

Модель Мура

Вхідне слово	x_{i_1}	x_{i_2}	…	x_{i_k}	
Послідовність станів	am	$a_{i_2} == \delta(a_m, x_{i_1})$	…	$a_{i_k} == (\delta(a_{i_{k-1}}\ x_{i_{k-1}}),$	$\tilde{\delta}(a_m, \xi)$ $a_{i_{k+1}} == \delta(a_{i_k}, x_{i_k})$
Реакція	$y_{i_1} = \lambda(a_m)$	$y_{i_2} = \lambda(a_{i_2})$	…	$y_{i_k} = \lambda(a_{i_k})$	$y_{i_{k+1}} = \lambda(a_{i_{k+1}})$
					$\tilde{\lambda}(a_m, \xi)$
		$\omega(a_m, \xi)$			

Поширимо функцію переходів $\delta: A \times X \to A$ на множину $\Psi \times X$, де Ψ — множина всіх підмножин множини A. Визначимо $\delta(B, x_f)$ $(B \subseteq A, x_f \in X)$ наступним чином:

$$\delta(B, x_f) = \{a_s \mid a_s = \delta(a_m, x_f), a_m \in B\}$$

Інакше:

$$\delta(B, x_f) = \bigcup_{a_m \in B} \{\delta(a_m, x_f)\}$$

Таким чином, $\delta(B, x_f)$ - множина станів, у які є перехід з усіх станів, що входять до B, під дією вхідного сигналу x_f.

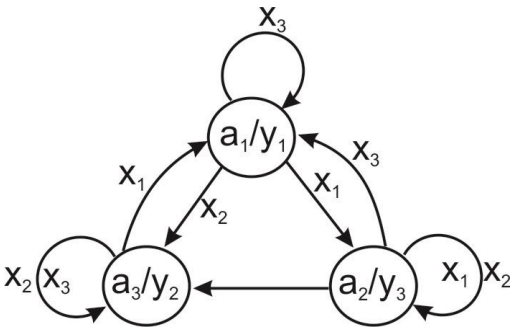


Рис. 8.7. Граф асинхронного автомата Мура.

Наприклад, для автомата, представленого таблицею 8.13

$B = \{a_1, a_4, a_5\}$

Тоді

$\delta(B, x_1) = \{a_2, a_3\}; \delta(B, x_2) = \{a_1, a_4\}.$

Визначимо синхронні і асинхронні автомати.

Стан a_s автомата S називається стійким, якщо для будь-якого входу $x_f \in X$ такого, що $\delta(a_m, x_f) = a_s$ має місце $\delta(a_m, x_f) = a_s$. Це означає, що якщо автомат перейшов у якийсь стан під дією вхідного сигналу x_f, то вийти з цього стану він може тільки при надходженні на його вхід іншого, відмінного від x_f вхідного сигналу. Автомат S називається асинхронним, якщо кожен його стан $a_s \in A$ стійкий. В іншому випадку автомат S буде синхронним.

Необхідно зауважити, що всі побудовані на практиці автомати - асинхронні, і стійкість їх станів забезпечується тим чи іншим способом, наприклад введенням сигналів синхронізації. Однак на рівні абстрактної теорії, коли автомат є лише математичною модель, яка не відображає багатьох конкретних особливостей його можливої реалізації, часто виявляється більш зручним оперувати з синхронними автоматами, що ми і будемо робити. Приклад асинхронного автомата Мура (граф автомата - рис. 8.7). Неважко бачити, що всі його стани стійкі.

Очевидно, що якщо в таблиці переходів асинхронного автомата деякий стан a_s записано на перетині рядка x_f и стовпчика $a_m (m \neq s)$, цей стан обов'язково має зустрітися у цьому ж рядку в стовпці a_s. В графі асинхронного автомата, якщо в якийсь стан є переходи з інших станів під дією якихось сигналів, то в вершині a_s повинна бути петля, зазначена символами тих же вхідних сигналів. Аналіз таблиць 8.11, 8.12, 8.13 показує, що ці автомати є синхронними.

Часто поняття синхронності або асинхронності автоматів пов'язують з наявністю або відсутністю синхронізуючих імпульсів у схемі автомата. Можна показати, що і при наявності синхронізуючих імпульсів автомат (правда, вже структурний, який буде розглянуто далі) може бути і, як правило, є асинхронним в сенсі введеного визначення. З абстрактним автоматом синхронізація взагалі ніяк не пов'язана. У той же час наведене визначення синхронних і асинхронних автоматів легко переноситься на структурний рівень.

8.4 Зв'язок між моделями Мілі та Мура

Як вже зазначалося, абстрактний автомат працює як перетворювач слів вхідного алфавіту в слова вихідного.

Нехай абстрактний автомат Мілі S4 задано графом рис. 8.8.

На вхід цього автомата, встановленого в початковий стан, надходить вхідне слово $X=x_1, x_1, x_2, x_1, x_2, x_2$.

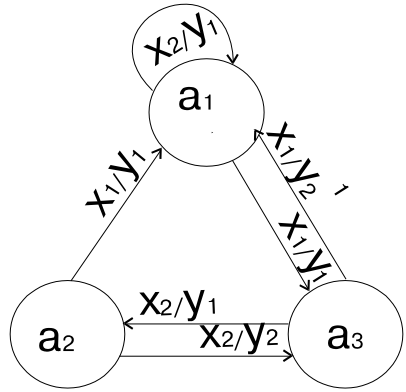


Рис. 8.8- Граф автомата Мілі

Так як $\delta(a_1, x_1) = a_3$, а $\lambda(a_1, x_1) = y_1$, то під дією першої літери слова X вхідного сигналу x_1 автомат перейде в стан a_3 і на виході його появиться сигнал y_1. Далі, $\delta(a_3, x_1) = a_1$, а $\lambda(a_3, x_1) = y_2$, тому при приході другого сигналу x_1 автомат опиниться в стані a_1, а на виході його з'явиться сигнал y_2. Простеживши безпосередньо по графу або за таблицями переходів і виходів подальшу поведінку автомата, опишемо його трьома рядками, перший з яких відповідає вхідному слову X, другий - послідовності станів, які проходить автомат під дією букв слова X, третій - вихідному слову У, яке з'являється на виході автомата:

x_1	x_1	x_2	x_1	x_2	x_2	
a_1	a_3	a_1	a_1	a_3	a_2	a_3
y_1	y2	y_1	y_1	y_1	y_2	

Назвемо $y = \lambda(a_1, X)$ реакцією автомата Мілі в стані a_1 на вхідне слово X. Як видно з прикладу, у відповідь на вхідне слово довжини k автомат Мілі видає

послідовність станів довжини k + 1 і вихідне слово довжини k. У загальному вигляді поведінку автомата Мілі, встановленого в стан a_m, можна описати таким чином:

Вхідне слово	x_{i1}	x_{i2}	x_{i3}
Послідовність станів	a_m	$a_{i2}= \delta (a_m,x_{i1})$	$a_{i3}= \delta (a_{i2},x_{i2})$.
Вихідне слово	$y_{i1}= \lambda (a_m,x_{i1})$	$y_{i2}= \lambda (a_{i2},x_{i2})$	$y_{i3}= \lambda (a_{i3},x_{i3})$

Точно так само можна описати поведінку автомата Мура, що знаходиться в стані a_m, при приході вхідного слова x_{i1}, x_{i2}, ..., x_{ik}. Нагадаємо, що відповідно до (8-2) вихідний сигнал в автоматі Мура в момент часу t (У(t)) залежить лише від стану, в якому знаходиться автомат в момент t (a(t)):

Вхідне слово	x_{i1}	x_{i2}	x_{i3}	
Послідовність станів	a_m	$a_{i2}= \delta (a_m,x_{i1})$	$a_{i3}= \delta (a_{i2},x_{i2})$.	$a_{i4}= \delta (a_{i3},x_{i3})$
Вихідне слово	$y_{i1}= \lambda (a_m,x_{i1})$	$y_{i2}= \lambda (a_{i2},x_{i2})$	$y_{i3}= \lambda (a_{i3},x_{i3})$	$y_{i4}= \lambda (a_{i4})$

Очевидно, що вихідний сигнал $y_{i1}=\lambda(_{am})$ в момент часу i_1 не залежить від вхідного сигналу x_{i1}, а визначається тільки станом a_m. Таким чином, цей сигнал y_{i1} ніяк не пов'язаний з вхідним словом, що надходить на вхід автомата, починаючи з моменту i_1. У зв'язку з цим під реакцією автомата Мура, встановленого в стан a_m на вхідне слово $X=x_{i1}, x_{i2}, ..., x_{ik}$ будемо розуміти вихідне слово тієї ж довжини

$y= \lambda (a_m, X)=y_{i2}, y_{i3} , ... , y_{ik+1}$.

Як приклад розглянемо автомат Мура S5, граф якого зображений на малюнку 8-9, і знайдемо його реакцію в початковому стані на те ж саме вхідне слово яке ми використовували при аналізі поведінки автомата Мілі S4 (малюнок 8.8):

Вхідне слово	x_1	x_1	x_2	x_1	x_2	x_2	
Послідовність станів	a_1	a_4	a_1	a_1	a_4	a_3	a_5
Вихідне слово	y_1	y_1	y_2	y_1	y_1	y_1	y_2

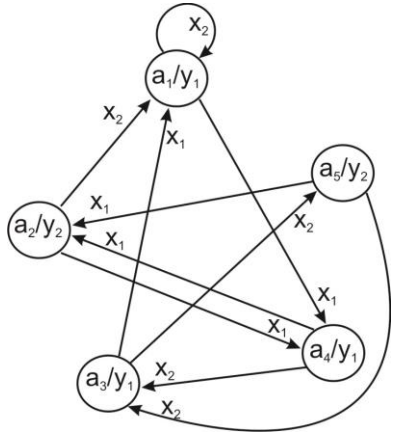


Рис. 8-9- Граф автомата Мура

Як видно з цього і попереднього прикладів, реакції автоматів S5 і S4 в початковому стані на вхідне слово X з точністю до зсуву на 1 такт збігаються (реакція автомата Мура обведена лінією). Дамо тепер суворе визначення еквівалентності повністю визначених автоматів.

Два автомата S_A і S_B з однаковими вхідними та вихідними алфавітами називаються еквівалентними, якщо після встановлення їх в початкові стани їх реакції на будь-яке вхідне слово збігаються.

8.5 Еквівалентні автомати. Еквівалентні перетворення автоматів

Можна показати, що для будь-якого автомата Мілі існує еквівалентний йому автомат Мура і, навпаки, для будь-якого автомата Мура існує еквівалентний йому автомат Мілі.

При описі алгоритмів взаємної трансформації автоматів Мілі та Мура відповідно до викладеного вище ми будемо нехтувати в автоматах Мура вихідним сигналом, пов'язаним з початковим станом ($\lambda(a_1)$).

Розглянемо спочатку перетворення автомата Мура в автомат Мілі.

Нехай дано автомат Мура: $S_A=\{ X_A, A_A, У_A, \delta_A, \lambda_A, a_{oA}\}$, де:

$X_A=\{x_1, x_2, \ldots x_n\}$; $У=\{y_1, y_2, \ldots y_m\}$; $A=\{ a_0, a_1, a_2, \ldots a_N\}$; $a_{0A} = a_0$ – початковий стан ($a_{0A} \in A$); δ_A - функція переходів автомата, яка задає відображення $(X_A \times A_A) \rightarrow A_A$; λ_A - функція виходів автомата, яка задає відображення $A_A \rightarrow У_A$.

Побудуємо автомат Мілі: $S_B=\{ X_B, A_B, У_B, \delta_B, \lambda_B, a_{0B}\}$, у якого $A_B=A_A$; $X_B=X_A$; $У_B=У_A$; $\delta_B= \delta_A$; $a_{0B}=a_{0A}$. Функцію виходів λ_B визначимо наступним чином. Якщо в автоматі Мура $\delta_A(a_m, x_f)= a_s$ і $\lambda_A (a_s)=y_g$, то в автоматі Мілі $\lambda_B(a_m, x_f)=y_g$.

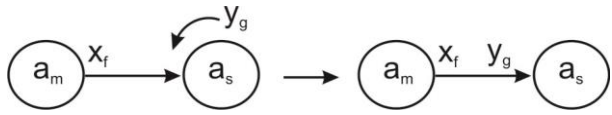


Рис. 8.10- Ілюстрація переходу від моделі Мура до моделі Мілі

Перехід від автомата Мура до автомату Мілі при графічному способі завдання ілюструється рисунком 8-10: вихідний сигнал y_g записаний поряд з вершиною (a_s), переноситься **на всі дуги, що входять** в цю вершину.

При табличному способі завдання автомата таблиця переходів автомата Мілі збігається з таблицею переходів початкового автомата Мура, а таблиця виходів виходить з таблиці переходів заміною символу a_s, що стоїть на перетині рядка x_f і стовпчика a_m, символом вихідного сигналу y_g який позначає стовпець a_s в таблиці переходів автомата S^A.

З самого способу побудови автомата Мілі S_B очевидно, що він еквівалентний автомату Мура S_A. За індукції неважко показати, що будь-яке вхідне слово кінцевої довжини, подане на входи автоматів S_A і S_B, встановлених в стан a_m, викличе появу однакових вихідних слів і, отже, автомати S_A і S_B еквівалентні.

Перш ніж розглянути трансформацію автомата Мілі в автомат Мура, накладемо на автомат Мілі наступне обмеження: у автомата не повинно бути переходящих станів. Під переходящим будемо розуміти стан, в який при представленні автомата у вигляді графа не входить жодна дуга, але який має принаймні одну дугу, яка виходить з нього. Отже, нехай заданий автомат Мілі:

$S_A=\{ X_A, A_A, У_A, \delta_A, \lambda_A, a_{0A}\}$, де:

$XA=\{x_1, x_2, ... x_n\}$; $У=\{y_1, y_2, ... y_m\}$; $A=\{ a_0, a_1, a_2, ... a_N\}$; $a_{0A} = a_0$- початковий стан ($a_{0A} \in A$); δ_A - функція переходів автомата, яка задає відображення $(X_A x A_A) \rightarrow A_A$; λ_A - функція виходів автомата, яка задає відображення $(X_A \times A_A) \rightarrow У_A$.

Побудуємо автомат Мура: $S_B=\{X_B, A_B, У_B, \delta_B, \lambda_B, a_{0B}\}$, у якого $X_B=X_A$; $У_B=У_A$.

Для визначення A_B кожному стану $a_s \in A_A$ поставимо у відповідність множину A_s всіх можливих пар виду (a_s, y_g)

Функції виходів λ_B і переходів δ_B визначимо наступним чином. Кожному стану автомата Мура S_B, який представляє собою пару виду (a_s,y_g), поставимо у відповідність вихідний сигнал y_g. Якщо в автоматі Мілі S_A був перехід $\delta_A(a_m, x_f)= a_s$ і при цьому видавався вихідний сигнал λ_A $(a_m,x_f)=y_k$, то в S_B буде перехід із множини станів A_m, що породжуються a_m, в стан (a_s ,y_k) під дією вхідного сигналу x_f.

В якості початкового стану a_{0B} можна взяти будь-який із станів множини A_0, який породжується початковим станом a_0 автомата S_A. При порівнянні реакцій автоматів S_A и S_B на всі можливі вхідні слова не повинен враховуватися вихідний сигнал в момент часу t = 0, пов'язаний зі станом a_{0B} автомата S_B.

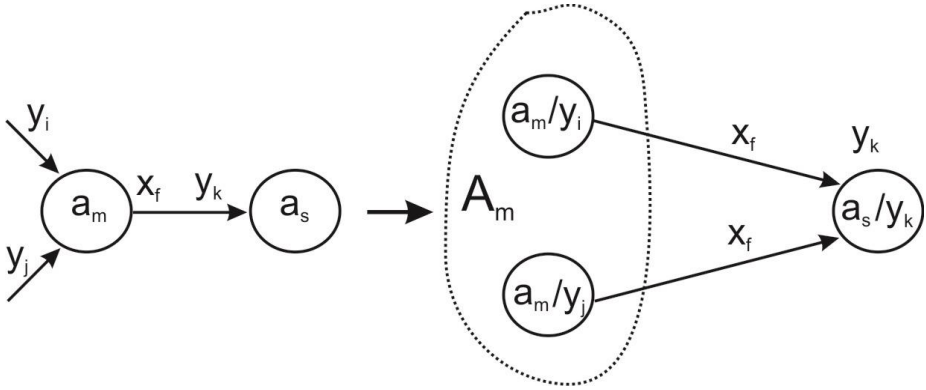


Рис. 8.11- Ілюстрація переходу від моделі Мілі до моделі Мура.

Розглянемо приклад. Нехай заданий автомат Мілі (табл. 8.14.)

Таблиця 8.14

A	**x_1**	**x_2**
a_0	**a_2/y_1**	**a_0/y_1**
a_1	**a_0/y_1**	**a_2/y_2**
a_2	**a_0/y_2**	**a_1/y_1**

Таблиця 8.15

X A	x_1	x_2
a_0 b_0	a_2/y_1 b_{01}	a_0/y_1 b_{02}
a_1	a_0/y_1 b_{11}	a_2/y_2 b_{12}
a_2	a_0/y_2 b_{21}	a_1/y_1 b_{22}

Поставимо у відповідність кожній парі a_i/x_k стан b_{ik} (i-номер стану, k-номер вхідного сигналу), з урахуванням b_0. Отримаємо таблицю 8.15.

Складемо таблицю переходів автомата Мура, керуючись такими правилами:

1) Випишемо з таблиці 8.15 стани автомата Мілі та відповідні кожному з них множини станів автомата Мура (b_{ik}):

a_0= {b_0, b_{02}, b_{11}, b_{21}}; a_1= {b_{22}}; a_2= {b_{01}, b_{12}};

2) Якщо стан автомата Мура b_{ik} входить в множину, до якої відноситься стан a_p автомата Мілі, то в рядок таблиці переходів автомата Мура для стану b_{ik} слід записати рядок з таблиці переходів автомата Мілі, який відповідає стану a_p (із таблиці 8.14.).

3) Функцію виходів автомата Мура визначимо наступним чином: $\lambda_B(b_{ik})= \lambda_A(a_i, x_k)$. Для початкового стану b_0 значення вихідного сигналу може бути вибраним довільно, але він повинен бути породжуваним початковим станом a_0 (з урахуванням поняття еквівалентності станів). Результуюча таблиця переходів і виходів автомата Мура еквівалентного автомату Мілі, заданого таблицею 8.14. представлена в таблиці 8.16.

Таблиця 8.16.

	x_1	x_2	y
b_0	b_{01}	b_{02}	y_1
b_{01}	b_{21}	b_{22}	y_1
b_{02}	b_{01}	b_{02}	y_1
b_{11}	b_{01}	b_{02}	y_1
b_{12}	b_{21}	b_{22}	y_2
b_{21}	b_{01}	b_{02}	y_2
b_{22}	b_{11}	b_{12}	y_1

4) Знайдемо в таблиці 8.16. еквівалентні стани і видалимо їх (замінимо на представника класу еквівалентності).

Якщо вихідний сигнал біля b_0 довизначити y_1, то виявиться, що в даній таблиці переходів знаходиться 3 еквівалентних стани (b_0,b_{11},b_{02}), $b_0 \equiv b_{11} \equiv b_{02}$. Замінивши клас еквівалентності одним представником (b_0), отримаємо остаточну таблицю переходів (табл. 8.17)

Таблиця 8.17.

	x_1	x_2	У
b_0	b_{01}	b_0	y_1
b_{01}	b_{21}	b_{22}	y_1
b_{12}	b_{21}	b_{22}	y_2
b_{21}	b_{01}	b_0	y_2
b_{22}	b_0	b_{12}	y_1

Викладені методи взаємної трансформації автоматів Мілі та Мура показують, що при переході від автомата Мура до автомата Мілі число станів автомата не змінюється, тоді як при зворотному переході число станів в автоматі Мура, як правило, зростає (воно не зростає, якщо кожен стан автомата Мілі породжує тільки один стан автомата Мура).

Таким чином, еквівалентні між собою автомати можуть мати різне число станів, у зв'язку з чим виникає завдання знаходження мінімального (з мінімальним числом станів) автомата в класі еквівалентних між собою автоматів. Існування для будь-якого абстрактного автомата S еквівалентного йому абстрактного автомата S_{min} з мінімальним числом внутрішніх станів вперше було доведено американським вченим Муром.

8.6 Мінімізація числа внутрішніх станів автомата. Алгоритм Ауфенкампа-Хона

В основу методу мінімізації станів автомата покладена ідея розбиття всіх станів вихідного абстрактного автомата на класи еквівалентних станів, які попарно не перетинаються і заміні кожного класу еквівалентності одним станом (представником даного класу). Утворений в результаті цих перетворень мінімальний автомат має стільки ж станів, на скільки класів еквівалентності розбивається множина вихідних станів.

Два стани автомата a_m та a_s називаються еквівалентними ($a_m \equiv a_s$), якщо $\lambda(a_m,X) = \lambda(a_s,X)$ для всіх можливих вхідних слів довжини X, тобто реакції автомата в цих станах на всілякі вхідні слова збігаються.

Якщо a_m та a_s не еквівалентні, вони різні. Більш слабкою еквівалентністю є K-еквівалентність. Стани a_m та a_s K- еквівалентні, якщо $\lambda(a_m, X_k) = \lambda(a_s,X_k)$ для всіх можливих вхідних слів довжини K.

Два автомати S={A, X, У, δ, λ, a_1} та S'={A', X, У, δ', λ', a_1'} одного і того ж типу (Милі або Мура) еквівалентні, якщо і тільки якщо для кожного стану $a_m \in A$ еквівалентне йому $a_s' \in A'$ і, навпаки для кожного $a_s' \in A'$ існує $a_m \in A$, йому еквівалентне.

Автомат S – мінімальний, якщо і тільки якщо з $a_m \equiv a_s$ випливає, що $a_m = a_s$.

При мінімізації числа внутрішніх станів автомата Мілі S={X, У, A, δ, λ, a_0} використовується алгоритм Ауфенкампа-Хона:

1. Знаходять послідовні розбиття π_1, π_2,…, π_k, π_{k+1}, множини A на класи одно-, дво-, ..., К-, (К+1) - еквівалентних станів до тих пір, поки на якомусь (К + 1) кроці не виявиться, що $\pi_k = \pi_{k+1}$. У цьому випадку К-еквівалентні стани є еквівалентними. Число кроків К, при $\pi_k = \pi_{k+1}$ не перевищує N-1, де N- число внутрішніх станів автомата.

2. У кожному класі еквівалентності π вибирають по одному елементу (представнику класу), які утворюють множини A' станів мінімального абстрактного автомата S'.

3. Функцію переходів δ' та виходів λ' автомата S' визначають на множині A'xX. Для цього в таблицях переходів і виходів викреслюють стовпці, що відповідають тим, які не ввійшли в множину станів A', а у решті стовпцях таблиці переходів усі стани замінюються на еквівалентні з множини A', (на представників класу).

4. В якості a'_0 вибирається один зі станів, еквівалентний стану a_0. Зокрема, зручно прийняти сам стан a_0.

При мінімізації повністю визначених автоматів Мура вводиться поняття 0-еквівалентності станів і розбиття множини станів на 0-класи: 0-еквівалентними називаються будь-які, однаково відмічені вихідними сигналами, стани автомата Мура. Якщо два 0-еквівалентних стани будь-яким вхідним сигналом переводяться в два 0-еквівалентних стани, то вони називаються 1-еквівалентними. Всі подальші класи еквівалентності станів для автомата Мура визначаються так само, як і для автомата Мілі.

Як приклад розглянемо мінімізацію автомата Мура, заданого таблицею переходів і виходів (Таблиця 8.18).

Таблиця 8.18

У	y_1	y_1	y_3	y_3	y_3	y_2	y_3	y_1	y_2	y_2	y_2	y_2
A	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}
x_1	a_{10}	a_{12}	a_5	a_7	a_3	a_7	a_3	a_{10}	a_7	a_1	a_5	a_2
x_2	a_5	a_7	a_6	a_{11}	a_9	a_{11}	a_6	a_4	a_6	a_8	a_9	a_8

Виконаємо розбиття π_0:

$\pi_0=\{B_1,B_2,B_3\}$;

$B_1=\{a_1, a_2, a_8\}$, $B_2=\{a_6, a_9, a_{10}, a_{11}, a_{12}\}$, $B_3=\{a_3, a_4, a_5, a_7\}$.

Побудуємо таблицю розбиття $\pi 0$:

У	**B_1**			B_2					B_3			
A	**a_1**	a_2	a_8	a_6	a_9	a_{10}	a_{11}	a_{12}	a_3	a_4	a_5	a_7
x_1	B_2	*B_2*	B_2	B_3	B_3	B_1	B_3	B_1	B_3	B_3	B_3	B_3
x_2	B_3	*B_3*	B_3	B_2	B_2	B_1	B_2	B_1	B_2	B_2	B_2	B_2

Виконаємо розбиття π_1:

$\pi_1=\{C_1, C_2, C_3, C_4\}$;

$C_1=\{a_1, a_2, a_8\}$, $C_2=\{a_6, a_9, a_{11}\}$, $C_3=\{ a_{10}, a_{12}\}$, $C_4=\{a_3, a_4, a_5, a_7\}$.

Побудуємо таблицю розбиття $\pi 1$:

У	C_1			C_2			C_3		C_4			
A	a_1	a_2	a_8	a_6	a_9	a_{11}	a_{10}	a_{12}	a_3	a_4	a_5	a_7
x_1	C_3	C_3	C_3	C_4	C_4	C_4	C_1	C_1	C_4	C_4	C_4	C_4
x_2	C_4	C_4	C_4	C_2	C_2	C_2	C_1	C_1	C_2	C_2	C_2	C_2

Виконаємо розбиття π_2.

$\pi_2=\{D_1, D_2, D_3, D_4\}$;

$D_1=\{a_1, a_2, a_8\}$, $D_2=\{a_6, a_9, a_{11}\}$, $D_3=\{ a_{10}, a_{12}\}$, $D_4=\{a_3, a_4, a_5, a_7\}$.

Розбиття π_2 повторює розбиття π_1 - процедуру розбиття завершено.

Виберемо довільно з кожного класу еквівалентності D_1, D_2, D_3, D_4 по одному представнику - в даному випадку за мінімальним номером:

$A'=\{a_1, a_3, a_6, a_{10}\}$.

Видаляючи з вихідної таблиці переходів "зайві" стани, визначаємо мінімальний автомат Мура (табл. 1.19.)

Таблиця 1.19.

У	y_1	y_3	y_2	y_2
A	a_1	a_3	a_6	a_{10}
x_1	a_{10}	a_3	a_3	a_1
x_2	a_3	a_6	a_6	a_1

Розглянемо приклад.

Мінімізувати число внутрішніх станів автомата Мілі, заданого таблицею переходів і виходів (Таблиця 1.20.), використовуючи алгоритм Ауфенкампа-Хона.

Таблиця 1.20

A	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
x_1	a_1/y_2	a_0/y_1	a_2/ y_4	a_3/ y_2	a_5/ y_1	a_0/ y_1	a_1/ y_2	a_7/ y_4	a_5/ y_1
x_2	a_3/ y_3	a_5/ y_2	a_3/ y_1	a_8/ y_3	a_3/ y_2	a_2/ y_2	a_3/ y_3	a_3/ y_1	a_3/ y_2
x_3	a_7/ y_4	a_6/ y_4	a_4/ y_2	a_4/ y_1	a_8/ y_3	a_8/ y_4	a_2/ y_4	a_8/ y_2	a_4/ y_3

Виконаємо розбиття π_1:

$\pi_1=\{B_1, B_2, B_3, B_4, B_5\}$;

$B_1=\{a_0, a_6\}$, $B_2=\{a_1, a_5\}$, $B_3=\{a_2, a_7\}$, $B_4=\{a_3\}$, $B_5=\{a_4, a_8\}$.

Побудуємо таблицю розбиття π_1:

	B_1		B_2		B_3		B_4	B_5	
A	a_0	a_6	a_1	a_5	a_2	a_7	a_3	a_4	a_8
x_1	B_2	B_2	B_1	B_1	B_3	B_3	B_4	B_2	B_2
x_2	B_4	B_4	B_2	B_3	B_4	B_4	B_5	B_4	B_4
x_3	B_3	B_3	B_3	B_5	B_5	B_5	B_5	B_5	B_5

Виконаємо розбиття π_2:

$\pi_2=\{C_1, C_2, C_3, C_4, C_5, C6\}$;

$C_1=\{a0, a_6\}$, $C_2=\{a_1\}$, $C_3=\{ a_5\}$, $C_4=\{a_2, a_7\}$, $C_5=\{a_3\}$, $C_6=\{a_4, a_8\}$

Побудуємо таблицю розбиття π 2:

	C_1		C_2	C_3	C_4		C_5	C_6	
A	a_0	a_6	a_1	a_5	a_2	a_7	a_3	a_4	a_8
x_1	C_2	C_2	C_1	C_1	C_4	C_4	C_5	C_3	C_3
x_2	C_5	C_5	C_3	C_4	C_5	C_5	C_6	C_5	C_5
x_3	C_4	C_4	C_1	C_6	C_6	C_6	C_6	C_6	C_6

Виконаємо розбиття π_3:

$\pi_3=\{D_1, D_2, D_3, D_4, D_5, D_6\}$;

$D_1=\{a_0, a_6\}$, $D_2=\{a_1\}$, $D_3=\{ a_5\}$, $D_4=\{a_2, a_7\}$, $D_5=\{a_3\}$, $D_6=\{a_4, a_8\}$

Розбиття π_3 повторює розбиття π_2 – процедуру розбиття завершено.

Можна зробити висновок: $a_0 \equiv a_6$; $a_2 \equiv a_7$; $a_4 \equiv a_8$.

Виберемо довільно з кожного класу еквівалентності D_1, D_2, D_3, D_4, D_5, D_6 по одному представнику - в даному випадку з мінімальним номером:

A'= $\{a_0, a_1, a_2, a_3, a_4, a_5\}$.

Видаляючи з вихідної таблиці переходів "зайві" стани, визначаємо мінімальний автомат Мілі:

A	a_0	a_1	a_2	a_3	a_4	a_5
x_1	a_1/y_2	a_0/y_1	a_2/y_4	a_3/y_2	a_5/y_1	a_0/y_1
x_2	a_3/ y_3	a_5/ y_2	a_3/ y_1	a_4/ y_3	a_3/ y_2	a_2/ y_2
x_3	a_2/ y_4	a_0/ y_4	a_4/ y_2	a_4/ y_1	a_4/ y_3	a_4/ y_4

9 СТРУКТУРНІ АВТОМАТИ

9.1 Основні поняття. Канонічний метод структурного синтезу автоматів. Теорема Глушкова про структурну повноту

Слідом за етапом абстрактного синтезу автоматів, що закінчується мінімізацією числа внутрішніх станів, слідує етап структурного синтезу, метою якого є побудова схеми, що реалізує автомат з логічних елементів заданого типу.

Якщо абстрактний автомат був лише математичною моделлю дискретної системи, то в структурному автоматі враховується структура вхідних і вихідних сигналів автомата, а також його внутрішня побудова на рівні структурних схем. Структурним синтезом займається структурна теорія автоматів, основним завданням якої є знаходження загальних прийомів побудови структурних схем автоматів на основі композиції елементарних автоматів, що належать до заздалегідь визначеного кінцевого числа типів.

Під композицією елементарних автоматів в загальному випадку розуміється наступне.

Нехай задані елементарні автомати $S_1,...,S_k$. Виконаємо об'єднання елементарних автоматів в систему спільно працюючих автоматів. Введемо в розгляд деяку кінцеву множину вузлів, званих зовнішніми вхідними і зовнішніми вихідними вузлами. Ці вузли відрізняються від вузлів розглянутих елементарних автоматів, які носять назву внутрішніх. Композиція автомата полягає в тому, що в отриманій системі елементарних автоматів $S_1,...,S_k$ і зовнішніх вузлів проводиться ототожнення деяких вузлів (як зовнішніх, так і внутрішніх). З точки зору спільної роботи системи автоматів зміст операції ототожнення вузлів полягає в тому, що елементарний сигнал, що потрапляє на один з вузлів, що входять в множину тих вузлів, які ототожнюються між собою, потрапляє тим самим на всі вузли цієї множини. Після проведених ототожнювань вузлів система автоматів перетворюється в так звану схему (мережу) автоматів. Будемо вважати, що автомати, що входять в схему автоматів, працюють спільно, якщо в кожен момент автоматного часу на всі зовнішні вхідні вузли подається набір вхідних сигналів (структурний вхідний сигнал схеми) і з усіх зовнішніх вихідних вузлів знімається набір вихідних сигналів (структурний вихідний сигнал).

Вважають, що у структурній теорії як вхідні так і вихідні канали складаються з елементарних вхідних (вихідних) каналів. По всіх елементарних вхідних (вихідних) каналах можуть передаватися тільки елементарні сигнали.

Рис. 9.1- Структурний автомат

Набір можливих значень сигналів, що подаються на один зовнішній вхідний (вихідний) вузол, називається структурним вхідним (вихідним) алфавітом автомата. Алфавіт повинен бути **кінцевим**.

Вхідний і вихідний сигнали задаються кінцевими впорядкованими наборами елементарних сигналів, які називаються **векторами**, а елементарні сигнали з яких вони складаються - компоненти векторів. Число компонентів вектора - це розмірність алфавіту.

Наприклад, $X=\{x_1, x_2, x_3, x_4, x_5\}$ - вхідний алфавіт абстрактного автомата.

Структурний вхідний алфавіт, розмірність якого дорівнює трьом:

$x_1=000$, $x_2=001$, $x_3=010$, $x_4=011$, $x_5= 100$.

Векторне представлення вхідних і вихідних сигналів називається структурним вхідним вихідним сигналом, відповідно.

Передбачається, що всі автомати, які входять в композицію мають один і той же структурний алфавіт і працюють в одному і тому ж автоматному часі. В даний час найбільш поширеним структурним алфавітом є двійковий, що пояснюється простотою його представлення в сучасних елементах і приладах. Крім того, для двійкового алфавіту найбільш розроблений апарат булевих функцій, що дозволяє проводити багато операцій над схемою формально. Тому надалі при вирішенні завдань структурного синтезу автоматів буде використовуватися в основному двійковий структурний алфавіт.

Припустимо, що в кожен момент автоматного часу структурний вихідний сигнал схеми однозначно визначається:

- кінцевою послідовністю структурних вхідних сигналів, що надійшла до цього часу на входи автомата;
- початковими станами автоматів, що входять у схему;
- зробленими при побудові схеми ототожненнями вузлів.

У цьому випадку побудовану схему будемо розглядати як деякий автомат S, а саму схему назвемо структурною схемою цього автомата.

Побудований таким чином автомат S є результатом композиції елементарних автоматів $S_1,...,S_k$. На відміну від абстрактного С-автомата, що має один вхідний і два вихідних канали, на які надходять сигнали у вхідному і вихідних алфавітах автомата, структурний автомат має вхідні і вихідні канали, на яких з'являються сигнали в структурному алфавіті автомата. У разі двійкового алфавіту кожен вхідний і вихідні сигнали абстрактного автомата можуть бути закодовані векторами різної довжини відповідно, компоненти яких можуть приймати два значення - нуль і одиницю.

На етапі структурного синтезу попередньо вибираються елементарні автомати, з яких потім шляхом їх композиції будується структурна схема отриманого на етапі абстрактного синтезу автомата Мілі, Мура або С-автомата. Якщо рішення задачі структурного синтезу існує, кажуть, що задана система автоматів структурно повна.

В даний час немає скільки-небудь ефективних методів (істотно більш простих, ніж метод перебору всіх варіантів) вирішення основного завдання структурного синтезу при будь-якому наборі структурно повних систем елементарних автоматів. Тому, зазвичай, застосовується так званий **канонічний метод структурного синтезу**, при якому використовуються елементарні автомати деякого спеціального виду: автомати з пам'яттю, що мають більш одного стану, і автомати без пам'яті - з одним станом. Автомати першого класу носять назву елементів пам'яті, а автомати другого класу - комбінаційних або логічних елементів.

Теоретичним обґрунтуванням канонічного методу структурного синтезу автоматів є доведена теорема про структурну повноту (теорема Глушкова):

Всяка система елементарних автоматів, яка містить автомат Мура з нетривіальною пам'яттю, що має повну систему переходів і повну систему виходів, і яку-небудь функціонально повну систему логічних елементів, є структурно повною.

Існує загальний конструктивний прийом (канонічний метод структурного синтезу), що дозволяє в розглянутому випадку звести задачу структурного синтезу довільних автоматів до задачі синтезу комбінаційних схем.

Результатом канонічного методу структурного синтезу є система логічних рівнянь, що виражає залежність вихідних сигналів автомата (функції виходів автомата) і сигналів, що подаються на входи елементів пам’яті, від сигналів, що приходять на вхід всього автомата в цілому, і сигналів, що знімаються з виходу

елементів пам'яті (функції збудження елементів пам'яті автомата). **Ці рівняння називаються канонічними.**

Для правильної роботи схем, очевидно, не можна дозволяти, щоб сигнали на вході елементів пам'яті безпосередньо приймали участь в утворенні вихідних сигналів, які по ланцюгах зворотного зв'язку подавалися б у той же самий момент часу на ці входи. У зв'язку з цим елементами пам'яті повинні бути не автомати Мілі, а автомати Мура (дивись рівняння функціонування цих автоматів).

Таким чином, структурно повна система елементарних автоматів повинна містити хоча б один автомат Мура. У той же час для синтезу будь-яких автоматів з мінімальним числом елементів пам'яті необхідно в якості таких елементів вибирати автомати Мура, що мають повну систему переходів і повну систему виходів - так звані повні автомати.

Розглянемо повноту автоматів пам'яті на прикладі автомата Мура. (табл. 9.1.) ***Повнота системи переходів означає***, що для будь-якої пари станів (a_m,…,a_s) автомата знайдеться вхідний сигнал, що переводить перший елемент цієї пари a_m в другий - a_s тобто, у такому автоматі в кожному стовпчику таблиці переходів повинні зустрічатися усі стани автомата. ***Повнота системи виходів*** автомата Мура полягає в тому, що кожному стану автомата поставлений у відповідність свій особливий вихідний сигнал, відмінний від вихідних сигналів інших станів.

Таблиця 9.1.

У	y_1	y_2	y_3
A\X	x_1	x_2	x_3
a_1	a_1	a_2	a_3
a_2	a_3	a_1	a_2
a_3	a_2	a_3	a_1

Очевидно, що в такому автоматі число вихідних сигналів дорівнює числу станів автомата. Разом з попереднім твердженням це призводить до того, що в автоматі Мура з повною системою виходів можна ототожнити стани автомата з його вихідними сигналами. У зв'язку з цим в автоматах пам'яті ми будемо використовувати одні й ті ж позначення і для станів, і для вихідних сигналів, тобто, зазначена таблиця переходів в автоматах Мура з повною системою виходів перетворюється просто в таблицю переходів. Автомат Мура в табл. 9.1. задовольняє умовам повноти системи переходів і виходів.

Наявність функціонально повної системи логічних елементів дозволяє реалізувати булеву функцію будь-якого ступеня складності.

Після вибору елементів пам'яті і кодування станів автомата синтез структурного автомата зводиться до синтезу комбінаційної схеми, яка реалізує канонічні рівняння.

Автомат пам'яті також можна розглядати на абстрактному і структурному рівнях. Автомат пам'яті має один вхідний і один вихідний канали. При переході від абстрактного автомата до структурного автомата вхідні і вихідні сигнали повинні бути закодовані відповідними двійковими наборами сигналів на вхідних і вихідних каналах.

9.2 *Основні етапи канонічного методу структурного синтезу*

У канонічному методі структурного синтезу можна виділити кілька основних етапів:

1. Кодування алфавітів автомата.
2. Вибір елементів пам'яті.
3. Вибір функціонально повної системи логічних елементів.
4. Запис і мінімізація канонічних рівнянь.
5. Побудова функціональної схеми автомата.

Вихідними даними для початку роботи даного методу є абстрактний цифровий автомат з пам'яттю, заданий таблицею переходів і виходів. Розглянемо докладніше кожен з етапів канонічного методу.

9.2.1 Кодування алфавітів автомата

На структурному рівні кожна літера вхідного алфавіту $x \in X$, кожна літера вихідного алфавіту $y \in У$ і кожна літера алфавіту станів $a \in A$ кодується двійковими векторами (двійковими наборами), число компонент яких визначається наступним чином:

$K_{вх} \geq \text{int}(\log_2|X|)$; $K_{вих} \geq \text{int}(\log_2|Y|)$; $K_{сост} \geq \text{int}(\log_2|A|)$;

де int - найближче більше ціле число.

|X|, |У|, |A| - потужність алфавіту вхідного, вихідного і алфавіту станів, відповідно.

Потужністю алфавіту називається кількість різних символів, що входять в цей алфавіт.

Процес заміни літер алфавіту (X, У, A) абстрактного автомата двійковими векторами носить назву кодування і описується таблицями кодування. Таблиця

кодування має наступний вигляд: в лівій частині перераховуються всі символи абстрактного алфавіту, а в правій - відповідні їм двійкові вектори.

Розглянемо приклад.

Автомат Мілі заданий таблицею переходів і виходів (табл. 9.2.). Виконати кодування алфавітів автомата.

Таблиця 9.2

A\ X	x_1	x_2
a_1	a_2/y_1	a_3/y_3
a_2	a_1/y_2	a_2/y_1
a_3	a_2/y_2	a_1/y_1

Випишемо алфавіти автомата і визначимо довжини векторів кодів алфавітів:

$X=\{x_1,x_2\}$; $K_{вх} \geq int(\log_2|2|)=1$,

$У=\{y_1,y_2,y_3\}$; $K_{вих} \geq int(\log_2|3|)=2$,

$A=\{a_1,a_2,a_3\}$; $K_{сост} \geq int(\log_2|3|)=2$.

00 01 10 11

Заповнимо таблиці кодування:

Таблиця 9.3

X	
x_1	0
x_2	1

Таблиця 9.4

У	
y_1	00
y_2	01
y_3	10

Таблиця 9.5

A	
a_1	00
a_2	01
a_3	10

Результуюча таблиця - структурна таблиця переходів і виходів автомата (табл. 9.6.) Одержанням структурної таблиці переходів-виходів автомата закінчується етап кодування.

Таблиця 9.6.

A\ X	0	1
00	01/00	10/10
01	00/01	01/00
10	01/01	00/00

У загальному випадку, кожній літері, що кодується може бути присвоєний довільний двійковий вектор, але обов'язково дві різні літери одного алфавіту повинні кодуватися різними векторами. Коди, що відповідають символам різних алфавітів можуть збігатися, в розглянутому прикладі - коди вихідних сигналів і станів.

На даному етапі доцільно зазначити, що спосіб кодування станів в значній мірі визначають складність реалізації комбінаційних схем. Існують спеціальні способи кодування, що забезпечують отримання економічних схем. Вони будуть розглянуті нижче.

9.2.2 Вибір елементів пам'яті автомата

Заміна таблиці переходів абстрактного автомата на структурну таблицю переходів призводить до того, що функція переходів $\delta(a_i,x_j)$ автомата стає векторною. Іншими словами, аргументами такої функції є всі можливі пари двійкових векторів (a_i,x_j), а сама функція приймає значення з множини А двійкових векторів станів автомата. У відповідності зі структурною таблицею переходів автомата його векторна функція переходів кожній парі двійкових векторів ставить у відповідність певний двійковий вектор a_k, що на абстрактному рівні визначається співвідношенням $a_k = \delta(a_i,x_j)$. З цього випливає, що структурний автомат повинен запам'ятовувати двійковий вектор кожного чергового стану автомата, для чого служать елементи пам'яті (запам'ятовуючі елементи, тригери).

При канонічному методі структурного синтезу автоматів в якості елементів пам'яті використовуються елементарні автомати Мура з двома станами, що мають повну систему переходів і виходів.

Повнота системи переходів автомата в загальному випадку означає, що для будь-якої пари станів автомата існує вхідний сигнал, що переводить елементарний автомат з одного стану в інший. Таблиця переходів елементарного автомата з повною системою переходів повинна містити в кожному своєму рядку всі можливі стани.

Повнота системи виходів означає, що різним станам автомата відповідають різні вихідні сигнали. Зазвичай нульовому стану елементарного автомата відповідає нульовий вихідний сигнал, одиничному - одиничний.

Очевидно, що ***число елементів пам'яті структурного автомата дорівнює числу компонент вектора його станів***.

9.2.3 Вибір функціонально-повної системи логічних елементів

Функціонування структурного автомата в часі припускає управління перемиканням кожного елементарного автомата його пам'яті у відповідності зі структурною таблицею переходів синтезованого автомата. Останнє здійснюється за допомогою спеціальної комбінаційної схеми, яка підключається до інформаційних входів елементарного автомата пам'яті і реалізує булеві функції, які управляють його перемиканням.

Такі булеві функції називаються **функціями збудження елемента пам'яті** і, в загальному випадку, різних функцій збудження стільки, скільки різних інформаційних входів є у елементарних автоматів пам'яті в синтезованому структурному автоматі.

Функція збудження будь-якого елемента пам'яті є довільною булевою функцією і для її реалізації комбінаційної схемою необхідно використовувати будь-яку функціонально-повну систему логічних елементів. Теоретичним фундаментом канонічного методу структурного синтезу автоматів з пам'яттю є теорема про структурну повноту, з якої випливає, що для побудови структурного автомата необхідно крім елементів пам'яті мати комбінаційну схему, що реалізує булеві функції збудження елементів пам'яті автомата, а для вироблення вихідних сигналів структурного автомата - спеціальну комбінаційну схему формування вихідних сигналів автомата.

Функція збудження структурного автомата є векторною. Її аргументами є пари двійкових векторів (a_i, x_j) .- а значенням функції - двійковий вектор, кожна і-а компонента якого є значення булевої функції збудження і-го елемента пам'яті автомата, що визначає той двійковий сигнал, який необхідно подати на вхід і- го елемента пам'яті для забезпечення його перемикання відповідно до вимог структурної таблиці переходів.

Якщо векторна функція переходів задає перехід з одного вектора стану структурного автомата в інший вектор стану під впливом двійкового вектора вхідного сигналу, то векторна функція збудження автомата задає двійковий вектор, який потрібно подати на входи елементів пам'яті автомата, щоб забезпечити необхідний перехід (відповідно до векторної функції переходів автомата).

Останнє означає, що змінними, від яких залежить векторна функція збудження, є ті ж змінні, що і для векторної функції переходів автомата, тобто виходи всіх елементів пам'яті автомата і входи автомата. Тому структурний автомат Мура, в загальному випадку, може бути представлений структурною схемою (малюнок 9.2), а структурний автомат Мілі - структурною схемою малюнок 9.3).

9.2.4 Побудова рівнянь булевих функцій збудження та виходів автомата

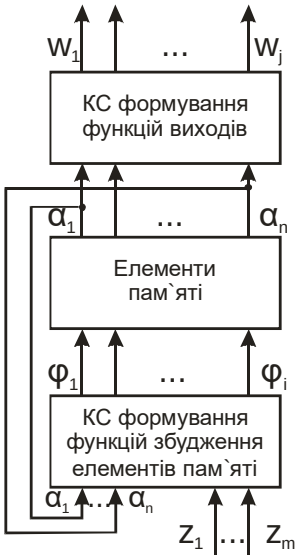


Рис. 9.2 - Структурна схема автомата Мура

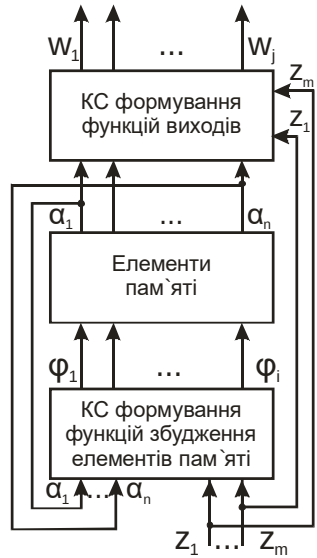


Рис. 9.3 - Структурна схема автомата Мілі

Літерами $\alpha_1,... \alpha_\varphi$ на малюнках позначені виходи елементів пам'яті. Літерами $\varphi_1,\ldots, \varphi_j$ позначені булеві функції збудження елементів пам'яті. Для простоти припустимо, що кожен елемент пам'яті структурного автомата має один інформаційний вхід. Літерами $w_1,...,w_j$ позначені вихідні канали автомата, де j - число вихідних каналів автомата. Літерами $z_1,\ldots,z_m$ – вхідні канали.

Кодування і вибір системи елементів однозначно визначають комбінаційну частину автомата: спочатку будується таблиця істинності функцій збудження елементів пам'яті автомата, що отримала назву таблиці функцій збудження; канонічні рівняння функцій збудження виписуються виходячи з побудованої таблиці. Отриманий аналітичний запис булевої функції збудження (для кожного елемента пам'яті автомата) може бути мінімізованим будь-яким з відомих методів. Вихідними даними для побудови таблиці функцій збудження є структурна таблиця переходів автомата і таблиця переходів елемента пам'яті.

Обрамлення таблиці функцій збудження, тобто ідентифікація її рядків і стовпців повністю збігається з обрамленням структурної таблиці переходів синтезованого автомата. Клітини, розташовані всередині таблиці функцій збудження, заповнюються спеціальним чином.

Отримання канонічних рівнянь булевих функцій виходів структурного автомата простіше і може бути зроблено безпосередньо по структурній таблиці виходів автомата. Структурна таблиця виходів автомата є таблиця істинності булевих функцій виходів автомата. Іншими словами, рівняння булевих функцій виходів автомата не залежить від типу використовуваних елементів пам'яті, проте залежать від їх кількості.

Якщо синтезований автомат є автоматом Мура, то завдання побудови рівнянь булевих функцій збудження вирішується так само. Рівняння булевих функцій виходів автомата Мілі будуються трохи інакше. Останнє пов'язано з відмінностями в способах побудови структурних таблиць виходів автомата Мілі і Мура. Після проведення етапу кодування станів автомата та кодування вихідних сигналів отримуємо структурну таблицю виходів автомата, яка є таблицею істинності булевих функцій виходів автомата.

9.2.5 Побудова функціональної схеми автомата

На підставі отриманих виразів для булевих функцій збудження елементів пам'яті автомата і булевих функцій виходів автомата будуються комбінаційна схема функцій збудження і комбінаційна схема формування вихідних сигналів автомата. Елементи пам'яті підключаються до побудованих комбінаційних схем. Функціональна схема автомата Мура відрізняється тільки комбінаційної схемою формування вихідних сигналів, яка будується на підставі рівнянь для булевих функцій виходів. Відзначимо, що реалізація комбінаційних схем може бути виконана в будь-якому функціонально-повному базисі.

9.3 Приклад канонічного методу структурного синтезу

Нехай задано абстрактний автомат Мілі таблицею переходів і виходів (табл. 9.7.). Використовуючи логічні елементи І, АБО, НЕ і елементарний автомат Мура (елемент пам'яті), заданий табл. 9.8.

Таблиця 9.7.

A\ X	x_1	x_2	x_3
a_1	a_2/y_3	a_3/y_4	a_2/y_2
a_2	-	a_1/y_3	a_3/y_1
a_3	a_1/y_2	-	a_3/y_3

Таблиця 9.8

	r_1	r_2
b_1	b_1	b_2
b_2	b_2	b_1

Виконаємо кодування алфавітів автомата (табл. 9.7.)
Випишемо алфавіти автомата і визначимо довжини векторів кодів алфавітів:

$X = \{x_1, x_2, x_3\}$; $K_{вх} \geq int(\log_2|3|)= 2$,
$Y = \{y_1, y_2, y_3, y_4\}$; $K_{вых} \geq int(\log_2|4|)= 2$,
$A = \{a_1, a_2, a_3\}$; $K_{сост} \geq int(\log_2|3|)= 2$.

Заповнимо таблиці кодування (табл. 9.9 – 9.11):

Таблиця 9.9.

X	z_1	z_2
x_1	0	0
x_2	0	1
x_3	1	0

Таблиця 9.10.

Y	w_1	w_2
y_1	1	0
y_2	0	0
y_3	1	1
y_4	0	1

Таблиця 9.11.

A	α_1	α_2
a_1	0	0
a_2	0	1
a_3	1	1

Кожен розряд вектора коду позначимо символом з відповідним номером.
Вхідні - z, вихідні - w, стани – α.

Таблиця 9.12

	z_1z_2		
$\alpha_1\alpha_2$	00	01	10
00	01/11	11/01	01/00
01	-	00/11	11/10
11	00/00	-	11/11

В результаті отримаємо таблицю переходів і виходів структурного автомата (табл. 9.12.).

Виконаємо кодування елементарного автомата Мура (табл. 9.8.):

- випишемо алфавіти автомата і визначимо довжини векторів кодів алфавітів,

$R = \{r_1, r_2\}$; $K_{вх} \geq int(\log_2|2|)=1$,

$B = \{b_1, b_2\}$; $K_{сост} \geq int(\log_2|2|)= 1$;

- заповнимо таблиці кодування:

Таблиця 9.13.

R	φ
r_1	0
r_2	1

Таблиця 9.14.

B	β
b_1	0
b_2	1

Структурна таблиця переходів елементарного автомата Мура має вигляд (табл. 9.15.):

Таблиця 9.15.

	0	1
0	0	1
1	1	0

Так як абстрактний автомат має три стани, кожен з яких кодується двома розрядами, то структурний автомат буде містити два запам'ятовуючих елементи.

Тепер завдання зводиться до синтезу комбінаційної схеми, яка реалізує канонічні рівняння:

$w_1=\lambda(\alpha_1, \alpha_2, z_1, z_2)$, $w_2= \lambda(\alpha_1, \alpha_2, z_1, z_2)$ - функції виходів автомата

$\varphi_1= f(\alpha_1, \alpha_2, z_1, z_2)$, $\varphi_2= f (\alpha_1, \alpha_2, z_1, z_2)$ - функції збудження елементів пам'яті автомата.

Функції w_1, w_2 можна отримати безпосередньо із зазначеної таблиці переходів структурного автомата як диз'юнкцію кон'юнкцій, що відповідають тим наборам $(\alpha_1,\alpha_2,z_1,z_2)$, на яких ці функції приймають значення 1. Але більш зручно користуватися так званою таблицею формування функцій збудження і функцій виходів автомата, в якій в табличній формі задана система булевих функцій (табл. 9.16). Заповнимо цю таблицю, використовуючи коди, що відповідають алфавітам і таблицю переходів і виходів абстрактного автомата. Для заповнення φ_1, φ_2 необхідно скористатися ще й таблицею елемента пам'яті (табл. 9.15).

Таблиця 9.16.

Початковий стан		Вхідний сигнал		Стан переходу		Функції збудження елементів пам'яті		Вихідний сигнал	
α_1	α_2	z_1	z_2	α_1	α_2	φ_1	φ_2	w_1	w_2
0	0	0	0	0	1	0	1	1	1
0	0	0	1	1	1	1	1	0	1
0	0	1	0	0	1	0	1	0	0
0	1	0	1	0	0	0	1	1	1
0	1	1	0	1	1	1	0	1	0
1	1	0	0	0	0	1	1	0	0
1	1	1	0	1	1	0	0	1	1

Для заповнення функцій збудження елементів пам'яті автомата розглядається перехід з початкового стану (α_1 α_2) в стан переходу (α_1 α_2). За перший розряд α_1 відповідає перший елемент пам'яті (його функція φ_1), за другий α_2 - другий елемент пам'яті (його функція φ_2).

В таблиці проставляється значення вхідного сигналу, який забезпечує відповідний перехід. Кількість функцій збудження елементів пам'яті автомата залежить від кількості розрядів вектора коду стану і від кількості інформаційних входів самого запам'ятовуючого елемента. Розглянемо, наприклад, що буде зі структурним автоматом, якщо він знаходиться в стані 01, і на його вхід надійшов сигнал 10.

Як видно з таблиці переходів структурний автомат перейде в стан 11. Цей перехід складається з двох переходів елементів пам'яті: 1-й з 0 в 1, 2-й з 1 в 1. По таблиці 9.15. визначимо вхідні сигнали елемента пам'яті, що забезпечують ці переходи: це 0 і 1., і т.д. У клітинку відповідного переходу запишемо вектор функції збудження, що викликає даний перехід.

По таблиці 9.16 запишемо аналітичні вирази канонічних рівнянь:

$$\varphi_1 = \bar{a}_1 \bar{a}_2 \bar{z}_1 z_2 \vee \bar{a}_1 a_2 z_1 \bar{z}_2 \vee a_1 a_2 \bar{z}_1 \bar{z}_2 = 1 \vee 6 \vee 12$$

$$\varphi_2 = \bar{a}_1\bar{a}_2\bar{z}_1\bar{z}_2 \vee \bar{a}_1\bar{a}_2\bar{z}_1 z_2 \vee \bar{a}_1\bar{a}_2 z_1\bar{z}_2 \vee \bar{a}_1 a_2\bar{z}_1 z_2 \vee a_1 a_2\bar{z}_1\bar{z}_2 = 0 \vee 1 \vee 2 \vee 5 \vee 12$$

$$w_1 = \bar{a}_1\bar{a}_2\bar{z}_1\bar{z}_2 \vee \bar{a}_1 a_2\bar{z}_1 z_2 \vee \bar{a}_1 a_2 z_1\bar{z}_2 \vee a_1 a_2 z_1\bar{z}_2 = 0 \vee 5 \vee 6 \vee 14$$

$$w_2 = \bar{a}_1\bar{a}_2\bar{z}_1\bar{z}_2 \vee \bar{a}_1\bar{a}_2\bar{z}_1 z_2 \vee \bar{a}_1 a_2\bar{z}_1 z_2 \vee a_1 a_2 z_1\bar{z}_2 = 0 \vee 1 \vee 5 \vee 14$$

Структурна схема даного автомата наведена на рис. 9.4.

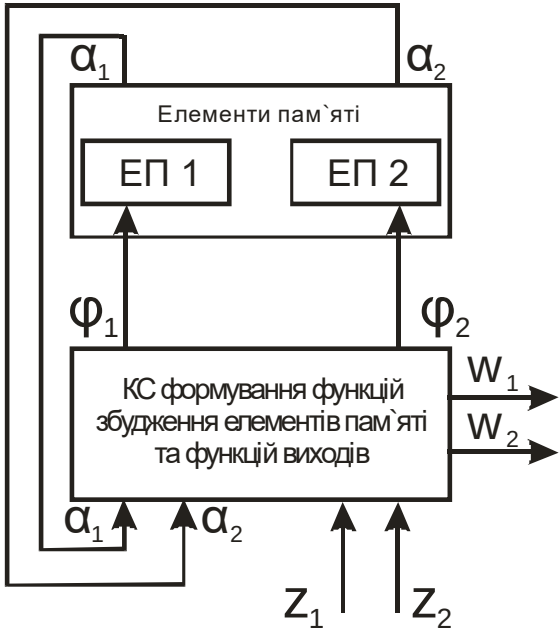


Рис. 9.4 - Структурна схема автомата

Не займаючись мінімізацією канонічних рівнянь, побудуємо схему електричну функціональну (рис. 9.5).

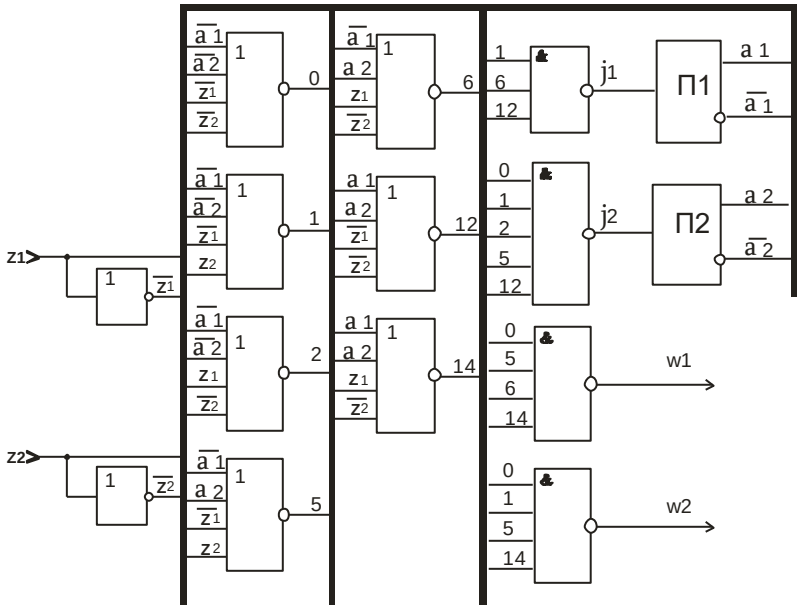


Рис. 9.5 - Функціональна схема автомата

Розглянемо деякі з етапів канонічного методу більш докладно, із застосуванням спеціальних методів.

9.4 Елементи пам'яті

В якості елементів пам'яті структурного автомата зазвичай використовуються тригери. Як вже було сказано, з точки зору прикладної теорії цифрових автоматів, тригер - це елементарний автомат Мура, що володіє повною системою переходів і повною системою виходів.

Тригер характеризується числом інформаційних входів, внутрішніх станів, числом вихідних сигналів і т.д. вихідні сигнали тригера ототожнюються з його внутрішніми станами, саме тому таблиця переходів збігається з таблицею виходів і тригер задається тільки однієї з них (таблицею переходів). Як правило, тригер формує як прямий сигнал, так і інверсний.

9.4.1 Елементи пам'яті з одним інформаційним входом

Існує тільки 4 типа запам'ятовуючих елементів з одним інформаційним входом, що мають повну систему переходів та виходів: D-тригер, Т-тригер, $\overline{D}$, -тригер, $\overline{T}$ -тригер. Таблиці їх переходів представлені табл. 2.17 - 2.20. відповідно, а умовні графічні зображення тригерів представлені на рис. 2.6. входи D, T, , $\overline{D}$, $\overline{T}$ називаються інформаційними.

Таблиці переходів тригерів складаються лише для інформаційних входів. Решта входів є допоміжними. Зокрема, вхід С - вхід для підключення синхросерії (про що буде сказано нижче). Кожен з тригерів має два виходи. Поява одиничного сигналу на виході, позначеному на малюнках символом q, означає, що тригер знаходиться в одиничному стані. Поява одиничного сигналу на виході $\overline{q}$ говорить про нульовий стан.

D-тригер - (елемент затримки) має один вхід та один вихід і здійснює затримку сигналу, який надійшов на його вхід, на один такт. Як видно з табл. 9.17, стан, в який переходить D-тригер (Q(t+1)), збігається з сигналом, який надходить на його вхід (D). У зв'язку з цим таблиця функцій збудження пам'яті синтезованого автомата S буде повністю співпадати з таблицею переходів цього автомата, якщо в якості елемента пам'яті буде обраний D-тригер. Дійсно, рядки і стовпці цих таблиць відзначені однаково, а на перетині і-го рядка і j-го стовпця в таблиці переходів записується код стану, в який переходить автомат зі стану, який відзначає j-й стовпець, під дією вхідного сигналу, який відзначає і-й рядок. Але при використанні D-тригера код стану, в який здійснюється перехід, збігається з сигналами, що надійшли на входи елементів пам'яті.

Тригер з лічильним входом (Т-тригер) має 1 вхід і 1 вихід. Його функція переходів наведена в таблиці 9.18.

Як видно з таблиці 9.18, вхідний сигнал Т, записаний в першому стовпці, дорівнює сумі по модулю 2 коду початкового стану (Q(t)) і стану переходу (Q(t+1)). На підставі цього при синтезі автомата на Т-тригерах таблицю функцій збудження можна отримати з таблиці переходів покомпонентною сумою за модулем 2 кодів початкового стану і стану переходу.

Таблиця 9.17.

D	Q(t)	Q(t+1)
0	0	0
0	1	0
1	0	1
1	1	1

Таблиця 9.18

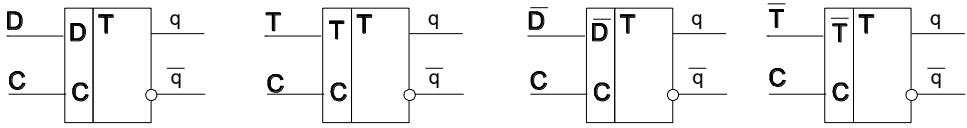
T	Q(t)	Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

Таблиця 9.19

$\overline{D}$	Q(t)	Q(t+1)
0	0	1
0	1	1
1	0	0
1	1	0

Таблиця 9.20.

$\overline{T}$	Q(t)	Q(t+1)
0	0	1
0	1	0
1	0	0
1	1	1



а б в г

Рис. 9.6- Умовне графічне позначення тригерів:

а) D-тригер; б) T-тригер; в) $\overline{D}$ - тригер; г) $\overline{T}$ -тригер.

У таблицях переходів тригерів з одним інформаційним входом дві перші колонки однакові - в них перераховані всі можливі комбінації вхідного сигналу і стану елемента пам'яті. Для того, щоб елементарний автомат мав повну систему переходів, колонку Q(t+1) слід заповнити таким чином, щоб у другій і третій колонках зустрічалися всі можливі типи переходів (00, 01, 10, 11). Для тригера типу D колонка Q(t+1) та D збігаються, так як вихідний сигнал ототожнюється зі станом, то це означає, що даний елемент є елементом затримки на 1 такт. Його характеристичне рівняння має вигляд: Q(t+1)=δ(Q(t), D(t))= DQ v D $\overline{Q}$ = D. Тригер типу T змінює свій стан тільки при подачі на його вхід «1». Це тригер з лічильним входом. Його характеристичне рівняння має вигляд:

Q(t+1)= δ(Q(t), T(t))= $\overline{T}$ **Q v T** $\overline{Q}$ **.**

9.4.2 Елементи пам'яті з двома інформаційними входами

Тригери з двома інформаційними входами мають різну побудова залежно від режимів використання наявних входів. Основними, найбільш поширеними двохвходовими тригерами є RS-тригер, JK-тригер, синхронізований D-тригер. Розглянемо докладніше кожен з них.

RS-тригер

Назва цього елемента походить від англійських слів «set-reset» - «установка-скидання (інше значення – переустановлення)». Він має два установочних входи: S-установка в 1, R - установка в нуль (скидання (*рос. сброс*)). Робота описується таблицею переходів (табл. 9.21.). На 6 і 7 наборах функція не визначена, тому вважається, що немає необхідності встановлювати даний тригер в положення «1» і «0» одночасно. Таким чином, вхідна комбінація 11 для RS-тригера є забороненою і не повинна виникати в реальних умовах роботи.

Характеристичне рівняння після його перетворення і мінімізації має вигляд:

$Q(t+1)=\delta(Q(t), R(t), S(t))= \overline{R}\, Q \vee S.$

Це співвідношення показує, що при нульовому сигналі на вході «установка в нуль» (R=0) RS-тригер є діз'юнктором накопичувального типу. Він здійснює логічне додавання вмісту тригера Q (t) і сигналу S (t), після чого результат операції записується замість першого доданка. В окремому випадку, при обнуленому тригері, здійснюється запис в тригер тієї інформації, яка надійшла на вхід S. Умовне графічне позначення RS-тригера представлено на рис.9.7.а).

Таблиця 9.21.

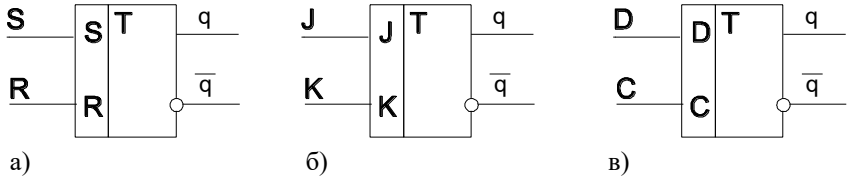
№	S	R	Q(t)	Q(t+1)
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	-
7	1	1	1	-

Таблиця 9.22.

№	J	K	Q(t)	Q(t+1)
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	1
7	1	1	1	0

Таблиця 9.23.

№	D	C	Q(t)	Q(t+1)
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1



а) б) в)

Рис. 9.7- Умовне графічне позначення тригерів:
а) RS-тригер; б) JK-тригер; в) синхронізований D-тригер.

Функціональна схема асинхронного RS-тригера, побудованого на елементах АБО-НІ, показана на рис.9.8.

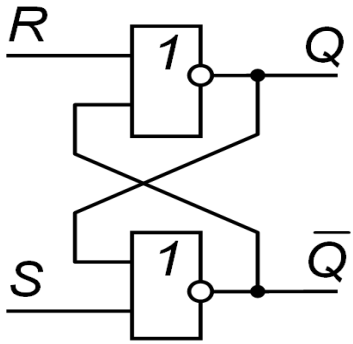


Рис. 9.8- Функціональна схема асинхронного RS-тригера

J-K-тригер

Він має два установочних входи: J - установка в 1, К - установка в нуль. Робота описується таблицею переходів (табл. 9.22.). Для нього не існує заборонених наборів вхідних сигналів.

Характеристичне рівняння JK-тригера після його перетворення і мінімізації має вигляд:

Q(t+1 $)=\delta$ (Q(t), J(t), K(t))= $\overline{K}Q \vee \overline{Q}J$

З цього співвідношення випливає, що JK-тригер є універсальним, який має два режими роботи.

1) Режим запису і зберігання інформації, що прийшла по входу J, якщо тригер заздалегідь був обнулений, оскільки робота обнуленого JK-тригера описується рівнянням RS-тригера. Даний режим називається RS-режимом.

2) Режим рахування, який виникає при забезпеченні однакових сигналів на обох входах. Оскільки такий режим описується рівнянням, аналогічним рівнянню Т-тригера, то його можна назвати Т-режимом. Умовне графічне позначення JK-тригера представлено на рис. 9.7.б.

D-тригер

Тригер має також 2 входи: інформаційний (D) і синхронізуючий (C). Функція на виході тригера приймає значення інформаційного сигналу, якщо є дозволяючий сигнал по входу C (C = 1). При відсутності дозволяючого сигналу (C = 0), значення функції заморожується, тобто залишається рівним вмісту тригера на попередньому такті. Робота описується таблицею переходів (табл. 9.23.).

Характеристичне рівняння тригера має вигляд:

$Q(t+1)=\delta(Q(t), D(t), C(t))= \overline{C}\, Q \vee DC.$

З чого випливає, що D-тригер є перемикачем накопичувального типу: він пропускає на вихід або сигнал, що приходить за умовною шиною D, або сигнал, що приходить за умовною шиною Q (t), залежно від значення керуючого сигналу C. Умовне графічне позначення тригера представлено на мал. 9.7.в.

9.4.3 Матриця переходів елемента пам'яті

Елемент пам'яті (тригер) може бути заданий одним з декількох способів: скороченою таблицею переходів, повною таблицею переходів, характеристичним рівнянням, матрицею переходів. Розглянемо останній спосіб.

Для кожного з 4-х можливих переходів елементарного автомата (00, 01, 10, 11) завжди знайдеться значення вхідного сигналу, рівне 0 або 1, яке викликає даний перехід. Якщо елементарний автомат має 2 або більше входів, то на деякі переходи значення вхідних сигналів, що діють на одному або іншому вході, виявляються несуттєвими.

Опишемо закон функціонування елементарного автомата, що має m входів, матрицею переходів:

	x_1	x_2	…	x_k	…	x_m
00	b_{00}^{1}	b_{00}^{2}	…	b_{00}^{k}	…	b_{00}^{m}
01	b_{01}^{1}	b_{01}^{2}	…	b_{01}^{k}	…	b_{01}^{m}
10	b_{10}^{1}	b_{10}^{2}	…	b_{10}^{k}	…	b_{10}^{m}
11	b_{11}^{1}	b_{11}^{2}	…	b_{11}^{k}	…	b_{11}^{m}

Кількість рядків матриці завжди дорівнює 4 (за кількістю можливих переходів), кількість стовпців дорівнює числу вхідних сигналів. Елемент матриці $\mathbf{b_{is}^{k}}$ являє собою значення вхідного сигналу x_k під дією якого елементарний автомат

переходить із стану i в стан s. При цьому b_{is}^{k} завжди дорівнює 0 або 1, або не визначений, якщо він не впливає на даний перехід.

Матриця переходів елементарного автомата складається за таблицею переходів.

Розглянемо приклад побудови матриці переходів тригера.

Нехай тригер, в загальному випадку, заданий скороченою таблицею переходів (табл. 9.24.). Побудувати повну таблицю переходів тригера і матрицю переходів.

Таблиця 9.24.

t		(t+1)
X	У	Q
0	0	0
0	1	Q(t)
1	0	$\overline{Q(t)}$
1	1	1

Таблиця 9.25

t			t+1
X	У	Q(t)	Q
0	0	0	0
0	0	1	0
0	1	**0**	0
0	1	**1**	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Таблиця 9.26.

Q(t)	Q(t+1)	X	У
0	0	0	0
		0	1
0	1	1	0
		1	1
1	0	0	0
		1	0
1	1	0	1
		1	1

Повна таблиця переходів тригера, побудована за скороченою, представлена в таблиці 9.25. Користуючись повною таблицею переходів запишемо комбінації вхідних сигналів, що відповідають усім можливим переходам (табл. 9.26.) Таким чином:

1. Для переходу "0-0" X=0, У може дорівнювати 0 або 1
2. Для переходу "0-1" X=1, У може дорівнювати 0 або 1
3. Для переходу "1-0" X може дорівнювати 0 або 1, а У=0.
4. Для переходу "1-1" X може дорівнювати 0 або 1, а У=1.

Тоді матриця переходів тригера запишеться у вигляді:

0-0	0	b_1
0-1	1	b_2
1-0	b_3	0
1-1	b_4	1

де b1,b2,b3,b4 - довільні сигнали (0 або 1).

Як правило, значення двох різних коефіцієнтів b_i, та b_s з одного рядка матриці є взаємозалежними один від одного і знаходження цієї залежності з ростом числа інформаційних входів ускладнюється. Встановлення точної взаємозалежності між вхідними змінними тригера є обов'язковою умовою, що забезпечує можливість максимального спрощення схем з пам'яттю. В основі методики лежить таблиця, в якій представлені можливі поєднання вхідних змінних і відповідні їм описи (табл. 9.27.).

Таблиця 9.27. Таблиця довизначення вхідних сигналів тригерів.

		Номер варіанту														
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Вхідні сигнали		X1	0	0	1	1	00	01	01	01	01	11	001	001	011	011	0011
		X2	0	1	0	1	01	00	01	10	11	01	010	011	001	101	0101
Опис взаємо-залеж-ності	I	X1	0	0	1	1	-	b1	b1	b1	b1	-	b1	b1	b1	b1	b1
	II	X2	0	1	0	1	-	0	b1	$\overline{b1}$	1	-	$\overline{b1}\wedge b2$	b1∨b2	b1∨b2	$\overline{b1}\vee b2$	b2
		X1	0	0	1	1	0	-	b1	$\overline{b1}$	-	1	$\overline{b1}\wedge b2$	b1∨b2	b1∨b2	$\overline{b1}\vee b2$	b2
		X2	0	1	0	1	b1	-	b1	b1	-	b1	b1	b1	b1	b1	b1

З урахуванням довизначень вхідних змінних матриці переходів деяких стандартних тригерів мають вигляд (таб. 9.28.)

Таблиця 9.28. Матриці переходів стандартних тригерів

Тригер-D			Тригер-T			Тригер-RS				Тригер-JK			
Q(t)	Q(t+1)	D	Q(t)	Q(t+1)	T	Q(t)	Q(t+1)	R	S	Q(t)	Q(t+1)	K	J
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	1	1	0	1	0	1	0	1	0	1
1	0	0	1	0	1	1	0	1	0	1	0	1	0
1	1	1	1	1	0	1	1	0	0	1	1	0	0

9.5 Кодування станів та виходів автомата і складність комбінаційних схем

Аналіз канонічного методу структурного синтезу показав, що різні варіанти кодування станів автомата приводять до різних виразів для функцій збудження і функцій виходів.

У розглянутому раніше прикладі коди станів автомата приймали значення: a_1=00, a_2=01, a_3=11. Якщо прийняти коди: a_1=01, a_2=01, a_3=00, то таблиця переходів структурного автомата матиме вигляд (табл. 9.29).

Якщо в якості запам'ятовуючого елемента застосувати D-тригер, то таблиця формування функцій збудження буде збігатися з цією таблицею. Тоді функції приймуть вид:

Таблиця 9.29

	Z_iZ_2		
$\alpha_1\alpha_2$	00	01	10
01	10	00	10
10	-	01	00
00	01	-	00

$$\varphi_1 = \bar{a}_1 a_2 \bar{z}_1 \bar{z}_2 \vee \bar{a}_1 a_2 z_1 \bar{z}_2;$$

$$\varphi_2 = \bar{a}_1 \bar{a}_2 \bar{z}_1 \bar{z}_2 \vee a_1 \bar{a}_2 \bar{z}_1 z_2;$$

Якщо порівняти з попереднім результатом, то неважко зробити висновок, що реалізація цих виразів комбінаційною схемою буде простішою.

В даному випадку при кодуванні станів був використаний **ваговий** метод, який може бути використаний і для кодування вихідних сигналів.

Алгоритм вагового методу кодування:

1. Кожному вихідному сигналу y_i, ставиться у відповідність ціле число P_i, яке дорівнює числу появ сигналу y_i в таблиці виходів автомата.

2. Числа P_i сортуються за зменшенням.

3. Вихідний сигнал y_i, який має найбільший ваговий коефіцієнт (P_i max) кодуються кодом, що містить всі 0 (00...00).

4. Наступні L вихідних сигналів (де L- число розрядів в двійковому векторі вихідного сигналу) за списком зменшення вагових коефіцієнтів (див. п. 2) кодуються кодами, що містять одну 1 (00...01, 00...10,... ,10...00).

5. Для кодування наступних вихідних сигналів за списком зменшення вагових коефіцієнтів використовуються всі коди, що містять дві одиниці, потім три одиниці і т.д., поки не будуть закодовані всі вихідні сигнали.

В результаті отримуємо таке кодування, при якому чим частіше зустрічається вихідний сигнал в таблиці виходів, тим менше одиниць міститься в його коді.

Кодування внутрішніх станів двійковими символами має істотний вплив на вартість комбінаційної частини схеми автомата. Оптимальне кодування дасть мінімальну вартість, проте алгоритм ефективного кодування невідомий. Тим не менш, при кодуванні внутрішніх станів автомата використовуються різні евристичні алгоритми, що дозволяють, принаймні в деяких випадках отримати кодування, близьке до оптимального.

Д.А. Мороз запропонував **евристичний алгоритм кодування** внутрішніх станів автоматів, заснований на мінімізації сумарного числа змін станів елементів пам'яті автомата на всіх переходах автомата.

При такому критерії зменшується складність схем, що реалізують диз'юнкції на виходах елементів пам'яті, тобто мінімізується комбінаційна схема автомата. Для оцінки кодування вводиться коефіцієнт ефективності кодування:

$K_{еф}$= W/P; де P - загальна кількість переходів автомата,

W - вагова функція: **$W=\Sigma t_{ij}$**

де t_{ij}- відстань Хеммінга між кодами станів a_i та a_j, рівна числу елементів пам'яті, що міняють свій стан на даному переході.

Відзначимо, що при визначенні вагової функції підсумовування проводиться по всіх переходах автомата. Коефіцієнт ефективності дозволяє оцінити складність комбінаційної схеми автомата: чим менше його значення, тим менше складність комбінаційної схеми, і оптимальний варіант – $K_{еф}=1$.

Алгоритм складається з наступних кроків:

1. Побудувати матрицю M, складену з усіх пар номерів (a_r, b_r) переходів автомата.

2. Переставити рядки в матриці таким чином, щоб у кожному наступному рядку містився хоча б один елемент з попередніх рядків.

3. Закодуємо стани з першого рядка матриці M наступним чином:

Ka_1=00...00, Kb_1=00...01.

4. Викреслимо з матриці M перший рядок з закодованими станами. Отримаємо матрицю M'.

5. У першому рядку матриці M' закодовано один елемент. Виберемо із першого рядка матриці M' незакодований елемент та позначимо його γ.

6. Побудуємо матрицю Mγ, вибравши з матриці M' рядки, які містять γ. Нехай Bγ={γ_1, γ_2,..., γ_f,..., γ_F} - множина елементів з матриці Mγ, які вже закодовані. Їх коди позначимо через K γ_1,K γ_2,... ,K γ_f,...,K γ_F відповідно.

7. Для кожного Kγ_f (f=1, 2,...,F) знайдемо $C^1\gamma_f$ - множина кодів, віддалених від Kγ_f на відстань Хеммінга, рівну 1 і ще не зайнятих для кодування станів автомата. Побудуємо множину $D_\gamma^1 = \bigcup_{f=1}^{F} C_{\gamma f}^1$.

Якщо $D^1\gamma$ =0, то будуємо нову множину $D_\gamma^2 = \bigcup_{f=1}^{F} C_{\gamma f}^2$, де $C_{\gamma f}^2$ - множина кодів, для яких кодова відстань із кодом K γ_f дорівнює 2. Якщо і $D^2\gamma$ =0, будуємо $D^3\gamma$ і т.д., поки не знайдемо $D_\gamma^n \neq 0$.

Нехай $D_\gamma^n = \{K_{\delta 1},...,K_{\delta g},...,K_{\delta G}\}$.

8. Для кожного $K_{\delta g}$ знаходимо $w_{gf}=|K_{\delta g}- K_{\gamma f}|^2$ – відстань Хеммінга між $K_{\delta g}$ і всіма використаними кодами $K_{\gamma f}$ (f=1, 2, ...,F).

9. Знаходимо $w_g = \sum_{f=1}^{F} w_{gf}, (g = 1,...,G)$.

10. Із D_γ^n вибираємо K_γ, у якого $W_g=W_{g\ min}$. Елементу γ буде присвоєно код K_γ.

11. Із матриці M' викреслюємо рядки, в яких обидва елементи закодовані, в результаті чого отримуємо нову матрицю, яку також позначаємо M'. Якщо в матриці M' не залишилося жодного рядка, переходимо до пункту 12, інакше до пункту 5.

12. Обчислити функцію $W = \sum t_{ij}$, де $t_{ij}=|K_i-K_j|^2$.

13. Кінець.

Оцінкою якості кодування по розглянутому алгоритму може служити число K=W/P, де P - число переходів в автоматі. Очевидно, що K≥1, причому, чим менше значення K, тим кращий результат кодування.

Розглянемо приклад кодування автомата, заданого таблицею переходів і виходів (табл. 9.30.).

Таблиця 9.30.

У	y_1	y_2	y_2	y_3
А	a_1	a_2	a_3	a_4
x_1	3	4	2	3
x_2	4	3	1	2

1. Будуємо матрицю М:

M=	M'=	M_γ=
1-3		
1-4	1-4	
2-4	2-4	**1-4**
2-3	2-3	2-4
3-2	3-2	**4-3**
3-1	4-3	4-2
4-3	4-2	
4-2		

Кодуємо перший рядок: K_1=00; K_3=01;

2. Викреслюємо з матриці М 1-й рядок (1-3) і 6-й рядок (3-1). Отримуємо матрицю М', в першому рядку якої не закодований елемент 4. Позначимо γ=4 і запишемо матрицю M_γ,. В матриці M_γ закодовані 1 і 3:

$B_\gamma = B_4 = \{1,3\} = \{00,01\}$ $C_{41}^1 = \{10\}; C_{43}^1 = \{11\}; D_4^1 = \{10,11\}$.

Знайдемо W:

$W_{10} = \begin{vmatrix} 00 \\ 10 \end{vmatrix} + \begin{vmatrix} 10 \\ 01 \end{vmatrix} = 3$; $W_{11} = \begin{vmatrix} 00 \\ 11 \end{vmatrix} + \begin{vmatrix} 11 \\ 01 \end{vmatrix} = 3$

Виберемо K_4=10.

Викреслимо із матриці М' рядок (1-4). Отримаємо нову матрицю М', в першому рядку якої не закодований елемент 2. Позначимо γ =2 і запишемо матрицю M_2. В матриці M_2 закодовані елементи 3, 4:

$B\gamma = B_2 = \{3,4\} = \{01,10\}$ $C_{23}^1 = \{11\}$; $C_{24}^1 = \{11\}$; $D_2^1 = \{11\}$.

$W_{11} = \begin{vmatrix} 11 \\ 10 \end{vmatrix} + \begin{vmatrix} 11 \\ 01 \end{vmatrix} + \begin{vmatrix} 01 \\ 11 \end{vmatrix} + \begin{vmatrix} 10 \\ 11 \end{vmatrix} = 4;$

У зв'язку із відсутністю інших варіантів кодів вибираємо K_2=11.

Таким чином: K_1=00; K_2=11; K_3=01; K_4=10.

Обчислимо функцію:

$$W = \left|\begin{matrix}00\\01\end{matrix}\right| + \left|\begin{matrix}00\\10\end{matrix}\right| + \left|\begin{matrix}11\\10\end{matrix}\right| + \left|\begin{matrix}11\\01\end{matrix}\right| + \left|\begin{matrix}01\\11\end{matrix}\right| + \left|\begin{matrix}01\\00\end{matrix}\right| + \left|\begin{matrix}10\\01\end{matrix}\right| + \left|\begin{matrix}10\\11\end{matrix}\right| = 9$$

Обчислимо K=W/P=9/8=1, 125.

9.6 Забезпечення стійкості функціонування цифрових автоматів. Гонки в автоматах

Одне з головних завдань, що вирішуються на етапі структурного синтезу синхронних цифрових автоматів з пам'яттю, полягає в забезпеченні стійкості їх функціонування. Поняття стійкості пов'язане з розробкою такої принципової електричної схеми автомата, яка забезпечувала б його функціонування відповідно до таблиці переходів і виходів автомата.

Неправильне функціонування автомата (нестійка його робота) пов'язане з особливостями фізичної реалізації логічних елементів і елементів пам'яті його схеми, а також різними величинами затримок розповсюдження сигналів в елементах і комбінаційних схемах.

Розглянемо процес забезпечення стійкості функціонування автомата більш докладно. Після надходження чергового вхідного сигналу і формування сигналів збудження на входах елементів пам'яті автомат переходить в новий стан. При цьому відбувається формування нових сигналів збудження по ланцюгах зворотних зв'язків (з виходів елементів пам'яті через логічні елементи на входи елементів пам'яті), і автомат переходить в новий стан і т. д.

Таким чином, автомат, в загальному випадку, не може зупинитися в якомусь певному стані і починає функціонувати в режимі *генератора станів.* Для усунення такого ефекту використовують синхросерії - послідовність спеціальних (зазвичай прямокутних) сигналів, що подаються на входи елементів пам'яті і дозволяють надходження чергових сигналів збудження на входи елементів пам'яті тільки з приходом чергового синхросигналу. При відсутності синхросигналу сигнал збудження не надходить на вхід елемента пам'яті, і елемент пам'яті цифрового автомата не здійснює перехід, тобто залишається в якомусь стані.

Практично підключення синхросерії здійснюється до спеціальних входів елемента пам'яті, що позначається символом С на малюнках і називається синхровхід. Введення синхросерії, однак, не забезпечить стабільного функціонування автомата, якщо не враховувати деякі особливості. Якщо при переході з одного стану в інший повинні змінювати свої стани відразу кілька

елементів пам'яті, то між ними починаються **змагання.** Той елемент, який виграє змагання, по ланцюгах зворотного зв'язку може змінити сигнали на входах інших елементів пам'яті до того, як вони змінять свої стани. Це може призвести до переходу автомата в стан, не передбачений графом.

Якщо в процесі переходу зі стану a_m в стан a_s під дією вхідного сигналу x_i автомат може опинитися в деякому проміжному стані (a_i, a_k) в залежності від того, який з елементів пам'яті виграв змагання, але, потім, при дії цього ж вхідного сигналу перейде в стан a_s, то такі змагання називаються **допустимими**, або не критичними. Але якщо відбудеться перехід в стан a_k (не передбачений графом переходів автомата) і правильність роботи автомата порушується, то такі змагання називаються неприпустимими (критичними) або ***гонками***. Нехай автомат повинен виконати переходи, зображені на рис. 9.9.

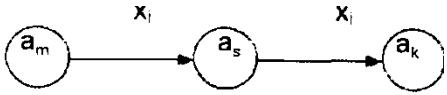


Рис. 9.9

Можливі наступні ситуації:

а) якщо черговий синхросигнал на входи елементів пам'яті автомата надходить раніше, ніж скінчилися перехідні процеси в його комбінаційній схемі збудження і елементах пам'яті після надходження вхідного сигналу x_i, то можливо неправильне формування сигналу збудження на вході одного або декількох елементів пам'яті автомата, тобто автомат може замість переходу зі стану a_m в стан a_s по вхідному сигналу x_i здійснити помилковий перехід в деякий стан a_t по тому ж самому вхідному сигналу x_i;

б) якщо тривалість вхідного сигналу x_i перевищує тривалість переходу автомата зі стану a_m в стан a_s, то автомат може (при надходженні чергового синхросигналу) проскочити стан a_s і потрапити відразу в стан a_k за рахунок подвійного спрацьовування автомата по вхідному сигналу x_i. Іншими словами, стан a_s може виявитися нестійким.

9.6.1 Методи усунення гонок в автоматах

Для забезпечення сталого функціонування автомата потрібно рознести в часі момент подачі інформації на входи його елементів пам'яті і момент зняття інформації з виходів елементів пам'яті. При такому рознесенні формування чергового сигналу збудження будь-якого елементу пам'яті в момент появи

синхросигналу здійснюється тільки за значеннями станів елементів пам'яті в попередній момент часу, а перехідні процеси в елементах пам'яті не впливають на формування сигналу збудження (виходи елементів пам'яті відключені).

Природно, що період проходження синхросигналів при цьому повинен вибиратися, виходячи з урахування закінчення перехідних процесів, пов'язаних із затримками поширення вхідного для автомата сигналу по логічним елементам комбінаційної схеми збудження. В результаті стійкість функціонування цифрового автомата може бути забезпечена, наприклад, з використанням **двоповерхової** (**дворівневої**) пам'яті. У цьому випадку кожен елемент пам'яті дублюється, і перезапис інформації з нижнього елемента пам'яті в верхній здійснюється по відсутності синхросигналу (рис. 9.9.).

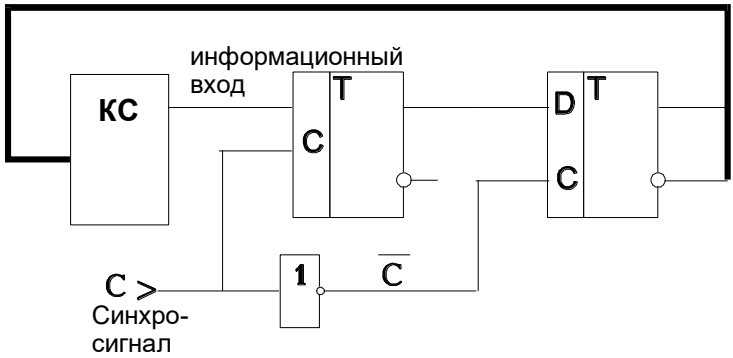


Рис. 9.10

Сигнали зворотних зв'язків, що використовуються для формування функцій збудження, і сигнали виходів автомата знімаються з виходів елемента пам'яті верхнього (другого) ярусу. При такій організації пам'яті автомата відсутня небезпека формування повторного сигналу збудження по одному і тому ж синхросигналу і переходу автомата в новий стан. Останнє пов'язано з тим, що перехід в робочий стан автомата завершується після закінчення дії синхросигналу.

Проте використання подвійний пам'яті автомата призводить до уповільнення роботи автомата. Якщо зазвичай період синхросигналів вибирається з розрахунку, що сигнал збудження елемента пам'яті встигне пройти по найдовшому ланцюжку логічних елементів і перемкнути елемент пам'яті, то тут період потрібно подовжити принаймні на 3t де t- затримка розповсюдження сигналу в логічному елементі (1t- на інвертор і 2t - на другий елемент пам'яті).

У випадках, коли з міркувань швидкодії **двоповерхову** пам'ять використовувати не можна, вдаються до багатофазної системи тактування вхідних сигналів автомата. Так, для випадку двофазної синхронізації синхросерії СС1 та СС2 замість одного вхідного сигналу x_i, (рис. 9.10.) використовуються два різних: $x_i \cdot CC1$ та $x_i \cdot CC2$ (рис. 9.11.)

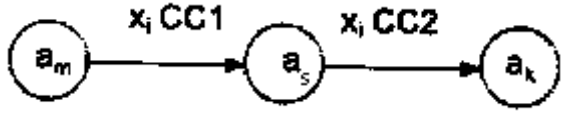


Рис. 9.11

Таким способом стійкість функціонування автомата забезпечується автоматично. При двофазній синхронізації необхідно, щоб всі дуги графа переходів автомата можна було б розмітити символами СС1 і СС2 так, що для будь-якої вершини графа усі, дуги, що виходять з неї, відзначалися б символом однієї синхросерії (наприклад, СС1, а всі дуги, що заходять в ту ж вершину графа переходів автомата, - символом іншої синхросерії (наприклад, СС2). Якщо граф переходів автомата містить контур непарної довжини, то така розмітка неможлива.

Однак її можна зробити, перетворивши контур непарної довжини в контур парної довжини, додавши додаткову вершину або стан автомата з порожнім вихідним сигналом. Завдання перетворення довільного графа з непарними контурами до графа з парними контурами вирішується методами теорії графів, зокрема з використанням поняття цикломатичного числа графа і методу побудови матриці фундаментальних циклів графа.

Крім описаних вище випадків, стійкість функціонування цифрового автомата з пам'яттю може бути частково забезпечена за допомогою спеціальних заходів, прийнятих щодо усунення в схемі автомата ефекту гонок. Це пов'язано з тим, що елементи пам'яті мають різні часи спрацьовування. Різні також затримки сигналів збудження, що надходять на входи елементів пам'яті по ланцюжках логічних елементів різної довжини.

Якщо при переході автомата з одного стану в інший, повинні переключитися відразу кілька елементів пам'яті, то між ними починаються гонки, (змагання), що може призвести до неправильної роботи автомата. Справді, при переході автомата зі стану a_i; в стан a_j на деякий, хоча і дуже короткий, проміжок часу може виникнути проміжний стан автомата, відмінний від a_i та a_j. Наприклад, при переході із a_i (0110) в a_j (1010) змінюють свій стан два перших елемента пам'яті. Через

змагання може виникнути стан 1110 (або 0010), який може привести до зміни стану третього або четвертого елемента пам'яті.

В останньому випадку після закінчення перехідних процесів автомат вже не потрапляє в стан a_j. При використанні двоповерхової пам'яті гонки в автоматі не виникають, так як зміна стану автомата відбувається в той час, коли синхросигнал відсутній.

Слід зауважити, що сучасна елементна база включає в себе елементи пам'яті з уже вбудованою двоповерховою пам'яттю (наприклад, JK-тригер). Існує ще один спосіб усунення гонок в автоматах, пов'язаний зі спеціальним кодуванням станів автомата, яке називається **протигоночним кодуванням**.

Окремим випадком протигоночного кодування є сусіднє кодування.

Умова відсутності гонок при сусідньому кодуванні завжди виконується. Суть цього кодування полягає в тому, що два стани, пов'язані дугою графа, кодуються наборами, що відрізняються лише в 1 елементі пам'яті, тобто мають відмінність тільки в одному розряді. Існує декілька алгоритмів, але вони не завжди піддаються формалізації.

Сусіднє кодування проте не завжди можливо. Вимоги до графа автомата, що допускає сусіднє кодування такі:

1) У графі не повинно бути циклів з непарними числом вершин. (Рисунок 9.12).

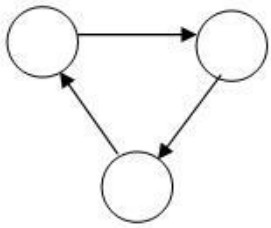


Рис. 9.12

2) Два сусідніх стани не повинні мати більше двох станів, що лежать між ними (Рисунок 9.13)

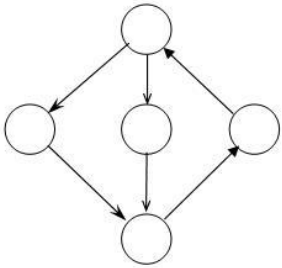


Рис. 9.13

Відзначимо, що проблема змагань і деякі інші питання забезпечення стійкості роботи автоматів виникли лише з появою потенційної елементної бази. Використовувана раніше (в ЕОМ другого покоління) імпульсно-потенційна елементна база передбачала застосування статичних тригерів з вбудованою затримкою.

При цьому величина затримки вибиралася більша, ніж тривалість імпульсного сигналу, що надходить на вхід тригера. Тим самим перехідні процеси формування сигналів на виходах елементів пам'яті автомата починалися лише після закінчення вхідного імпульсного сигналу і, отже, не чинили вплив на входи цих же елементів пам'яті по ланцюгах зворотного зв'язку. Щось подібне здійснюється тепер у потенційній елементній базі введенням серій синхронізуючих сигналів, що не збігаються в часі, особливо при організації двоповерхової пам'яті.

10 МІКРОПРОГРАМНІ АВТОМАТИ

10.1 Основні поняття. Принцип мікропрограмного управління

Мікропрограмні автомати являють собою наступний крок на шляху ускладнення інтелекту цифрових схем. На основі мікропрограмних автоматів можна будувати пристрої, які працюють за досить складними алгоритмами, виконують різні функції, які визначаються вхідними сигналами, видають складні послідовності вихідних сигналів. При цьому алгоритм роботи мікропрограмного автомата може бути легко змінений заміною прошивки ПЗУ (постійний запам'ятовуючий пристрій).

В машине для инженерных расчетов МИР1 было использовано ступенчатое микропрограммное управление. В 1967 году на выставке в Лондоне, где демонстрировалась МИР1, она была куплена американской фирмой IBM - крупнейшей в США, являющейся поставщиком почти 80% вычислительной техники для всего мира.

На відміну від пристроїв на "жорсткій" логіці, що розглядалися раніше і принцип роботи яких однозначно визначається використовуваними елементами і способом їх з'єднання, мікропрограмні автомати за допомогою однієї і тієї ж схеми можуть виконувати самі різні функції. Тобто вони набагато гнучкіші, ніж схеми на "жорсткій" логіці. До того ж проектувати мікропрограмні автомати з точки зору схемотехніки досить просто. Недоліком будь-якого мікропрограмного автомата в порівнянні зі схемами на "жорсткій" логіці є менша гранична швидкодія й необхідність складання карти прошивки ПЗУ з мікропрограмами, часто досить складними.

Розглянуті методи і прийоми синтезу дискретних пристроїв (ДП) використовували їх представлення у вигляді сукупності двох основних блоків: комбінаційного логічного і блоку елементів пам'яті. Такий підхід характеризується універсальністю і забезпечує гарні результати при побудові відносно нескладних ДП. Однак, отримані на його основі процедури синтезу ДП виявляються надмірно громіздкими і трудомісткими при побудові пристроїв середньої і великої складності, що мають важливе практичне значення. Робота таких пристроїв, зазвичай, полягає в реалізації деякого алгоритму обробки інформації, тобто у виконанні впорядкованої послідовності певних операцій над даними, які надходять. ***При побудові таких ДП доцільно використовувати принцип мікропрограмного управління, що полягає в наступному:***

1) будь-яка операція, реалізована пристроєм, розглядається як складна дія, яка поділяється на послідовність елементарних дій, званих **мікрооперaціями**;

2) для управління порядком проходження мікрооперацій використовуються логічні умови x_i, що приймають залежно від результатів виконання мікрооперацій значення 1 або 0;

3) процес виконання операцій в пристрої описується у формі алгоритму, представленого в термінах мікрооперацій і логічних умов і називається **мікропрограмою**;

4) мікропрограма використовується як форма представлення функції пристрою, на основі якої визначаються його структура та порядок функціонування.

Все сказане можна розглядати, як змістовний опис принципу мікропрограмного управління.

10.2 Концепція управляючого та операційного автоматів

При використанні описаного принципу прийнято ділити дискретні пристрої на дві частини: операційний автомат (ОА) і управляючий автомат (УА) (Рисунок. 10.1).

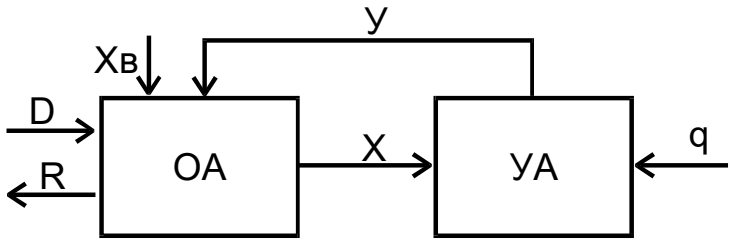


Рис. 10.1- Узагальнена структурна схема мікропрограмного дискретного пристрою

ОА призначений для зберігання інформації, що надходить на вхід D, видачі результатів виконання операцій R, виконання заданого набору мікрооперацій, вироблення значень логічних умов $X=(x_0,x_1,..., x_m)$, які є для управляючого автомата тими, що сповіщають. УА генерує послідовність управляючих сигналів $Y=(y_1,y_2,...,y_n)$ відповідно до заданої мікропрограми і зі значеннями логічних умов X. Кожен управляючий сигнал ініціює виконання відповідної мікрооперації в ОА.

У загальному випадку ДП призначається для виконання ряду мікропрограм, і на УА подається зовнішній сигнал q, відповідно до якого починається виконання тієї чи іншої мікропрограми. Якщо ДП є частиною системи обробки інформації, то

він може також обмінюватися спеціальними сигналами логічних умов X_B та управління Y_B з іншими блоками системи.

До складу ОА входять головним чином типові функціональні вузли: регістри, лічильники, суматори, дешифратори, шифратори, арифметико-логічні пристрої (АЛП), схеми порівняння, блоки пам'яті, схеми пересилання даних і т. п. Число елементів пам'яті (ЕП), що містяться в ОА, визначається розрядністю оброблюваних даних nД, яка може бути досить великою. Однак трудомісткість і складність проектування ОА, як правило, слабо залежать від nД в силу широкого використання стандартних вузлів. Таким чином, ОА є виконавчою частиною пристрою; його склад і структура можуть бути однаковими для реалізації багатьох алгоритмів одного класу.

Елементарний неподільний акт обробки інформації в операційному автоматі, що відбувається протягом одного моменту автоматного часу (одного такту роботи автомата), називається ***мікрооперацією***. Прикладами мікроопераций можуть служити «Зсув інформації», «+1», «Інверсія змінної» і т. д.

Якщо в операційному автоматі одночасно реалізується декілька мікроопераций, то така множина мікроопераций називається ***мікрокомандою***. Не виключений випадок, коли множина мікроопераций, що утворюють мікрокоманду, є порожньою. Реалізація такої мікрокоманди в операційному автоматі рівносильна відсутності виконання будь-яких елементарних операцій. У разі синхронних дискретних пристроїв порожня мікрокоманда інтерпретується як пропуск такту, коли ніякі сигнали від управляючого автомата на операційний автомат не надходять.

Мікрооперації побуджуються вихідними сигналами управляючого автомата, а їх послідовність у часі визначається функціями переходу управляючого автомата.

Сукупність мікрокоманд і функцій переходу утворює ***мікропрограму***. Таким чином, для опису мікропрограми необхідно задати множину мікрокоманд і функцій переходу, що визначають порядок їх виконання. Для опису мікропрограм зручно використовувати мову граф-схем алгоритмів (ГСА).

10.3 Управляючі автомати з жорсткою і програмуємою логікою

Обсяг обладнання УА залежить від складності реалізованого алгоритму і від структури цього автомата, яку можна виконати в трьох варіантах.

1. УА з жорсткою (схемною, довільною) логікою, при якій перемикальні функції, необхідні для формування заданої послідовності управляючих сигналів У, реалізуються за допомогою логічних елементів з довільними зв'язками (зазвичай із

застосуванням схем з малою і середньою ступенями інтеграції). Тут використовується апаратний підхід до реалізації пристрою.

2. УА із збереженою в пам'яті (гнучкою, програмною) логікою, при якій сигнали У виробляються на основі сукупності управляючих слів, що зберігаються в пам'яті автомата. У цьому випадку написані мікропрограми використовуються в явній формі і зазвичай записуються в постійні запам'ятовуючі пристрої (ПЗУ), виконані на основі напівпровідникових БІС великої ємності, що дозволяє забезпечити регулярність структури УА і його компактність; тут використовується апаратно-програмний підхід до реалізації пристрою.

3. УА на основі програмованих логічних матриць (ПЛМ), в яких задані функції реалізуються за допомогою БІС ПЛМ, що дозволяє поєднувати чимало достоїнств перших двох.

Таким чином, використання принципу мікропрограмного управління дозволяє впорядкувати та спростити процедуру логічного проектування ДП, забезпечити регулярність їх структури, а також відкриває можливість широкого застосування сучасних БІС. Принцип мікропрограмування застосовується при створенні мікропроцесорів і пристроїв на їх основі. Це не тільки дозволяє упорядкувати управління, але і дає можливість формувати систему команд мікропроцесорів на свій розсуд, виходячи з наявної системи мікрокоманд.

Розглянемо порядок проектування мікропрограмного ДП, який складається з наступних основних етапів:

Запис алгоритму. За описом окремих алгоритмів, що реалізуються пристроєм, складається їх формалізований запис у вигляді граф-схем алгоритмів (ГСА). Для цього складається список необхідних мікрооперацій y_j, і відповідних їм управляючих сигналів y_j, а також логічних умов x_i;. далі при необхідності, проводиться мінімізація числа вершин ГСА і складається об'єднана ГСА, що є формою побудови ДП для виконання наступних етапів.

Побудова ОА. У загальному випадку ОА може бути побудований за канонічною схемою автомата і містить три основні частини: блок елементів пам'яті для зберігання операндів, а також проміжних і кінцевих результатів; комбінаційну схему, що реалізує набір мікрооперацій; комбінаційну схему, що виробляє значення логічних умов. Як вже зазначалося, при побудові ОА доцільно застосовувати типові вузли, а також прагнути використовувати окремі вузли для виконання декількох мікрооперацій.

Побудова УА. Спочатку вибирають варіант структури УА, враховуючи вимоги швидкодії, допустимий обсяг апаратури й інші обмеження. Далі здійснюється синтез УА відповідно до процедури, яка залежить від прийнятої структури автомата.

В результаті виконання цих етапів складають структурні схеми ОА і УА і переходять до технічного проектування, яке включає питання практичної реалізації схеми пристрою на обраній елементній базі, введення необхідних розв'язуючих, підсилюючих та формуючих каскадів, компоновку деталей на платах, складання монтажних схем і видачу технічної документації.

10.3.1 Форми завдання алгоритмів мікропрограмних автоматів.

Один з улюблених методів зображення якого-небудь процесу обчислень або управління у дослідників - картинка-діаграма, яка компактна, наочна і легко розуміється будь-яким користувачем. Вже давно намітилося прагнення до стандартизації подібних способів опису функціонування різних пристроїв.

Розглянемо можливі способи такої стандартизації.

Процес функціонування мікропрограмного автомата зобразимо за допомогою графів. Кожній вершині графа співставимо деякий оператор, який реалізується в цій вершині. Оператором може бути обчислення деякого виразу, або формування деякого набору управляючих сигналів, або яка-небудь інша дія, яку необхідно виконати при реалізації управління об'єктом. Дугам, що з'єднують вершини, будемо надавати зміст вказівника послідовності виконання операторів.

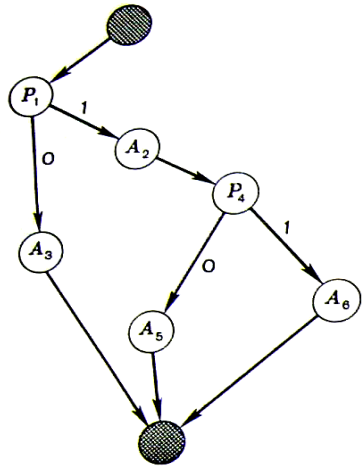


Рисунок 10.2 Приклад графічного зображення процесу функціонування мікропрограмного автомата.

Для того щоб можна було встановити початок і кінець цієї послідовності, виділимо спеціальний ***початковий оператор***, в який не входить ні однієї дуги, а виходить тільки одна дуга, і ***кінцеві оператори***, які можуть мати кілька вхідних дуг, але не мають жодної, яка виходить. Всі інші оператори діляться на два класи: ***функціональні оператори*** (що позначаються A_i) та ***логічні оператори*** (що позначаються P_i). Кожен оператор A_i має єдину дугу, яка виходить з нього. Тому незалежно від результату дії функціонального оператора A_i слідом за ним виконується завжди один оператор, в який веде дуга A_i. Кожен оператор P_i має дві дуги, що виходять з нього, відмічені символами 0 і 1 («брехня» і «істина»). логічний оператор P_i завжди перевіряє деяку умову. Якщо умова виконана, то наступним виконується оператор, в який веде дуга з символом 1, якщо умова не виконана, то перехід відбувається по дузі з символом 0.

На рисунку 10.2 показано приклад графічного зображення процесу функціонування мікропрограмного автомата. На цьому малюнку початковий і кінцевий оператори заштриховані. Зображення обчислювального або керуючого процесу у вигляді графа з урахуванням тих умов зображення, які ми тільки що ввели, називається графічною схемою алгоритму.

Логічні схеми алгоритмів

Замість графічних схем, що вимагають зображення у вигляді картинки, можна використовувати більш компактний запис.

Будемо записувати оператори в рядок зліва направо в порядку їхнього природного виконання. При цьому не будемо спеціально виписувати початковий оператор, вважаючи, що він завжди знаходиться на початку цього рядка. Кінцевий оператор будемо позначати буквою S.

Порядок виконання операторів в рядку приймемо наступним. Якщо виконується оператор A_i, то слідом за ним виконується оператор, що стоїть в рядку праворуч від A_i. Якщо ж виконується оператор P_i, то подальше виконання алгоритму залежить від того, чи виконано умову, яка виконується цим оператором. Якщо умова виконана, то наступним виконуваним оператором буде оператор, записаний праворуч від P_i. Якщо ж умова не виконана, то починає виконуватися оператор, до якого йде стрілка від оператора P_i. Вищенаведену графічну схему алгоритму можна записати у вигляді логічної схеми алгоритму (ЛСА)

$$P_1A_2P_4A_6SA_5SA_3S$$

Однак при великому числі логічних операторів зображення переходів у вигляді сукупності стрілок може виявитися не дуже зручним. Тому замість того щоб від кожного оператора P_i вести стрілку до того оператора, який повинен виконуватися, якщо логічна умова, що перевіряється оператором P_i, не виконана, поступають таким чином. Справа від P_i ставиться стрілка вверх, над якою записується номер цього оператора, а перед оператором, до якого повинна була йти ця стрілка, ставиться стрілка кінцем вниз, зазначена тим же номером. При таких позначеннях розглянутий вище рядок набуде вигляду:

$$P_1 \overset{1}{\uparrow} A_2 P_4 \overset{2}{\uparrow} A_6 \, S \underset{2}{\downarrow} A_5 \, S \underset{1}{\downarrow} A_3 S$$

Подібний спосіб запису алгоритмів обчислень і алгоритмів керування отримав назву логічних схем алгоритмів (ЛСА).

Матричні схеми алгоритмів

Запис у вигляді ЛСА при великій кількості стрілок може здатися занадто громіздким. Тому багато розробників віддають перевагу записові у вигляді матричних схем алгоритму (МСА), що представляє по суті лише іншу форму зображення ЛСА.

В МСА використовується матриця розмірності q×q, де q – число всіх операторів, які є в ЛСА. Позначимо цю матрицю через G. Елементи G приймають два значення: 0 та 1. Якщо g_i =1, то це означає, що після оператора з індексом і (оператор, що стоїть в рядку) буде виконуватися оператор з індексом j (оператор, що стоїть в стовпці).

У кожному рядку, співставленому функціональному оператору, є тільки одна одиниця, в кожному рядку, співставленому логічному оператору, є тільки дві одиниці, а в кожному рядку, співставленому оператору S, немає жодної одиниці. Для всіх операторів, крім логічних, забороняється мати на співставленому їм рядку одиницю на головній діагоналі. Одна з двох одиниць, співставлених логічному оператору, виділена жирним шрифтом. Ця одиниця показує перехід при виконанні даної логічної умови. Наведемо приклад запису МСА. Запишемо МСА для розглянутої вище ЛСА. Так як в ЛСА складається з 7 операторів, то МСА представляється матрицею 7×7:

	P_1	P_4	A_2	A_3	A_5	A_6	S
P_1	0	0	**1**	1	0	0	0
P_4	0	0	0	0	1	**1**	0
A_2	0	1	0	0	0	0	0
A_3	0	0	0	0	0	0	1
A_5	0	0	0	0	0	0	1
A_6	0	0	0	0	0	0	1
S	0	0	0	0	0	0	0

Логічна схема алгоритму, яку ми представили у вигляді МСА, була досить простою в порівнянні з цією матрицею.

10.4 Граф-схеми алгоритмів (ГСА)

ГСА - це орієнтований зв'язний граф, що задає послідовність виконання операцій даного алгоритму і містить ряд операторних і умовних вершин, а також одну початкову та одну кінцеву вершини. Операторною називається вершина, якій співставляється одна або декілька мікрооперацій і відзначається відповідними управляючими сигналами У, а умовною - вершина, якій співставляється деяка логічна умова *X*.

Будь-яка вершина ГСА, крім вершини «Початок», має по одному входу. Вершина «Початок» входів не має. Вершина «Початок» и будь-яка операторна вершина мають по одному виходу. Вершина «Кінець» виходів не має. Кожна умовна вершина має два виходи, які позначаються символами «Так» і «Ні»: Замість цих символів можуть бути використані цифри «1» і «0» відповідно. Зображення вершин «Початок», «Кінець», операторної вершини і умовної вершини ГСА представлено на рисунку 10.3.

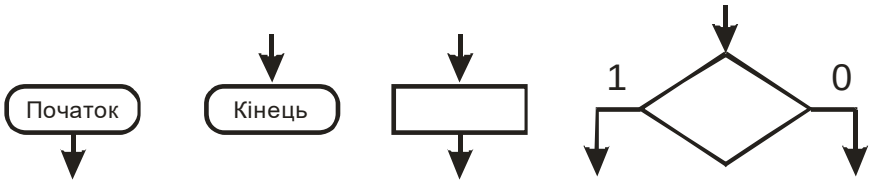


Рис. 10.3-Граф-схеми алгоритмів

ГСА складають так, щоб забезпечити виконання необхідних операцій і перевірку логічних умов відповідно зі словесним описом алгоритму.

На підставі переліку мікрооперацій і функціональних вузлів, що реалізують їх, складається структурна схема ОА. На структурній схемй ОА стрілками показують шини, по яких передається інформація, та управляючі сигнали у, що керують роботою окремих вузлів або передачею інформації по шинах.

ГСА повинна задовольняти наступним умовам:

1. Входи і виходи вершин з'єднуються один з одним за допомогою дуг, спрямованих завжди від виходу до входу.

2. Кожен вихід з'єднаний тільки з одним входом.

3. Будь-який вхід з'єднується, принаймні, з одним виходом.

4. Будь-яка вершина ГСА лежить, принаймні, на одному шляху з вершини «Початок» в вершину «Кінець».

5. Один з виходів умовної вершини може з'єднуватися з її входом, що неприпустимо для операторної вершини. Такі умовні вершини іноді називаються зворотними.

6. У кожній умовній вершині записується логічна умова з множини логічних умов. Дозволяється в різних умовних вершинах записувати однакові логічні умови.

7. У кожній операторної вершині записується оператор, що представляє собою вихідний сигнал або сукупність вихідних сигналів управляючого автомата. Дозволяється в різних операторних вершинах записувати однакові оператори.

10.5 Змістовні граф-схеми алгоритмів

Зазвичай при проектуванні різних пристроїв попередньо складається так звана змістовна ГСА, в якій всередині умовних і операторних вершин записані логічні умови і мікрооперації в змістовних термінах.

Як приклад побудуємо змістовну ГСА пристрою, що обчислює функцію знака числа:

$$S = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases}$$

Відповідна змістовна ГСА представлена на рисунку 10.4.

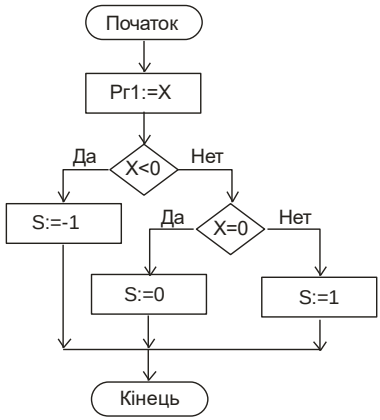


Рис. 10.4 - Змістовна ГСА функції визначення знака числа.

10.6 Синтез управляючого автомата по граф-схемі алгоритму

Кінцевий управляючий автомат, який реалізує мікропрограму роботи дискретного пристрою, прийнято називати мікропрограмним автоматом. Як уже зазначалося, мікропрограма відображається за допомогою ГСА. Розглянемо послідовність етапів синтезу управляючого автомата по його ГСА.

1. Запис словесного алгоритму функціонування операційного автомата (виконуваних операцій) з урахуванням структури операційного автомата.
2. Побудова змістовної ГСА функціонування операційного автомата.
3. Побудова відміченої ГСА з урахуванням типу автомата.
4. Побудова графа переходів автомата або таблиці переходів.
5. Проведення структурного синтезу автомата за його графом переходів з використанням відомих методів, наприклад, за допомогою канонічного методу структурного синтезу.

Побудова відміченої ГСА проводиться по змістовній ГСА. Для автоматів Мілі та Мура процедура розмітки має відмінності.

10.6.1 Побудова відміченої ГСА автомата Мілі

Якщо необхідно побудувати мікропрограмний автомат Мілі, то змістовна ГСА управляючого автомата розмічається відповідно до наступних правил:

1) символом стану a_1 відзначити вхід вершини, наступної за вершиною «Початок», а також вхід вершини «Кінець»;

2) входи всіх вершин, що знаходяться за операторними, повинні бути відзначені символами а з послідовними індексами;

3) якщо вихід вершини відмічається, то тільки одним символом;

4) входи різних вершин, за винятком вершини «Кінець», відзначаються різними символами;

5) змістовні терміни мікрооперацій і логічних умов замінюються їх умовними позначеннями: у кожній операторній вершині послідовно проставляються символи вихідних сигналів, якщо в різних операторних вершинах записані однакові мікрооперації, то дозволяється їх відмічати однаковими символами вихідних сигналів; якщо в різних умовних вершинах записані однакові логічні умови, то дозволяється їх відмічати однаковими символами вхідних сигналів.

Відмічена ГСА для змістовної ГСА (рис. 10.4.) Після розмітки за наведеним алгоритмом представлена на рисунку 10.5.

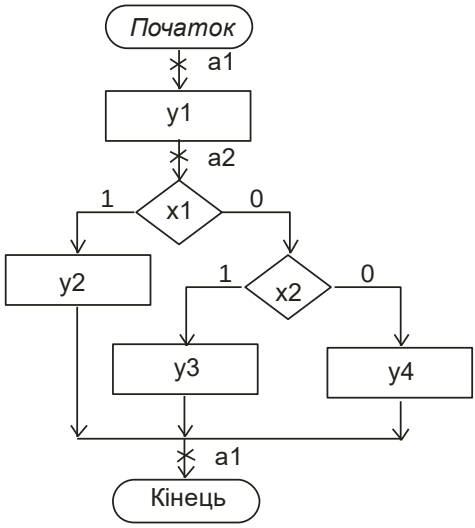


Рис. 10.5- Відмічена ГСА автомата Мілі

10.6.2 Побудова відміченої ГСА автомата Мура

Якщо необхідно побудувати мікропрограмний автомат Мура, то змістовна ГСА управляючого автомата розмічається відповідно до наступних правил:

1) символом a_1 відмічаються вершини «Початок» і «Кінець»;

2) різні операторні вершини відмічаються різними символами;

3) всі операторні вершини повинні бути відмічені.

4) змістовні терміни мікрооперацій і логічних умов замінюються їх умовними позначеннями.

Змістовна ГСА (рис. 10.4.) після розмітки за наведеним алгоритмом представлена на рисунку 10.6.

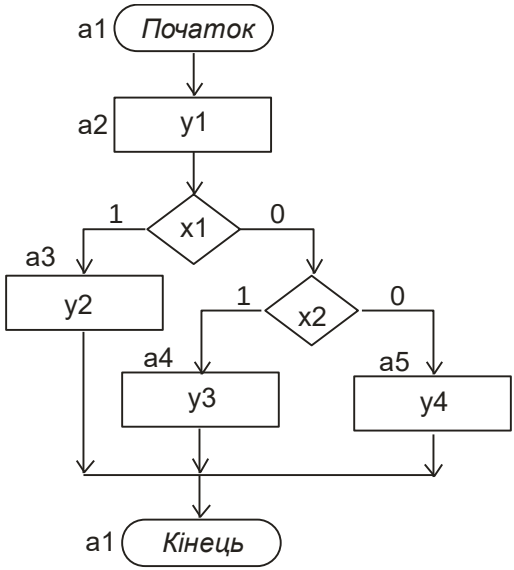


Рис. 10.6- Відмічена ГСА автомата Мура

Після отримання відміченої ГСА будується граф переходів автомата. Він має стільки різних вершин, скільки різних букв a_i з індексами є на відміченій ГСА. Кожна вершина графа переходів автомата відзначається буквою а з відповідним індексом.

Між двома вершинами графа проводиться дуга, якщо на відміченій ГСА між вершинами з мітками a_i і a_k, є шлях. Над дугою ставиться вхідний сигнал, що дорівнює кон'юнкції логічних умов відповідного шляху у відміченій ГСА. При цьому виконанню логічної умови відповідає змінна без заперечення, а невиконанню логічної умови - змінна з запереченням на відповідній дузі графа переходів автомата.

Якщо у відміченій ГСА між згаданими вершинами з мітками a_i і a_k є кілька шляхів, то в графі переходів автомата на дузі, що зв'язує a_i і a_k через символ диз'юнкції перераховуються всі кон'юнкції, що відповідають наявним шляхам.

Якщо будується граф переходів автомата Мура, то символи мікрооперацій (вихідні сигнали управляючого автомата) записуються поряд з відповідними вершинами. Для автомата Мілі символи мікрооперацій записуються на відповідних дугах при кон'юнкції логічних умов, що описують шлях через операційну вершину з розглянутою мікрооперацією.

Якщо у відміченій ГСА є ***безумовний*** перехід між операторними вершинами, тобто є шлях, що не проходить ні через які умовні вершини, то на графі переходів автомата йому відповідає дуга, якій приписується вхідний сигнал «1», що показує, що даний перехід в автоматі здійснюється при надходженні чергового синхросигналу.

Надалі синтез проводиться за допомогою описаного раніше методу структурного синтезу. Підкреслимо, що вхідними сигналами синтезованого структурного автомата є кон'юнкції булевих змінних (або диз'юнкції кон'юнкцій), кожна з яких відображає шлях через відповідні умовні вершини відміченої ГСА, а вихідними сигналами - мікрооперації, що позначають або вершини, або дуги графа переходів автомата, залежно від його типу. Використовуючи канонічний метод структурного синтезу, можна побудувати функціональну схему цифрового автомата.

10.7 Приклад синтезу мікропрограмного автомата

10.7.1 Побудова змістовної ГСА виконання операції додавання і віднімання чисел, представлених в форматі з фіксованою комою

Число X з фіксованою комою представляється правильним двійковим дробом виду $X = \pm, x_1 x_2 \ldots x_n$, де x_i - 0 або 1, (i= 1, 2, . . ., n). Знак числа кодується

двійковими символами 0 і 1. Знак «+» представляється символом 0 і знак «-» - символом 1.

При паралельному способі виконання операції додавання для представлення від'ємних чисел використовуються модифіковані обернені коди. Від'ємне число $X = -,x_1x_2...x_n$ в модифікованому оберненому коді має вигляд $(X)_{обр} = \overline{11},x_1x_2...\overline{x}_n$, де старший розряд цілої частини використовується для представлення знака числа і наступний розряд цілої частини - для контролю переповнення розрядної сітки сумматора. Для додавання двійкових чисел з фіксованою комою при використанні обернених модифікованих кодів застосовується наступний алгоритм:

1) якщо знак операнда додатний, то операнд вступає операцію додавання в прямому модифікованому коді; якщо знак операнда від'ємний, то операнд вступає в операцію у оберненому модифікованому коді;

2) проводиться додавання двійкових кодів операндів за всіма розрядами, включаючи знакові розряди; при цьому спадаюча одиниця зі старшого знакового розряду передається з якості одиниці переносу в молодший розряд суми;

3) якщо значення знакових розрядів суми дорівнює 00, то сума має додатне значення і представлена в прямому коді;

4) якщо значення знакових розрядів суми дорівнює 11, то сума має від'ємне значення і представлена у оберненому коді, для отримання прямого коду суми необхідно інвертувати значення в цифрових розрядах суми;

5) якщо значення в знакових розрядах суми дорівнює 01 або 10, то результат переповнює розрядну сітку машини.

Розглянемо схему операційної частини пристрою, орієнтованого тільки на виконання операції додавання (малюнок. 10.7.). У цьому пристрої для додавання використовується п + 2-розрядний накопичувальний суматор. Суматор охоплений ланцюгом циклічного переносу, по якому сигнал переносу із старшого знакового розряду суматора надходить на вхід молодшого розряду суматора. Передбачається, що перший доданок зберігається на суматорі в прямому коді. Другий доданок зберігається на регістрі Рг. Код операнда $X = x_0x_1x_2...x_n$ надходить по вхідний шині X. Результат операції видається в паралельному коді на вихідній шині У, $Y = y_0y_1y_2...y_n$. Значення y_0 відповідає старшому знаковому розряду коду, що зберігається на суматорі.

Відповідно до прийнятого алгоритму додавання в операційній частині пристрою виконуються наступні мікрооперації:

М1: См:=(Рг)$_{обр}$; *М6: Рг[0]:= Рг[0];*

М2: См:=Рг; *М7: Рг:=Х;*

М3: Додавання на суматорі *М8: Рг:=0.*

М4: См:=0; *М9: ПП:=1.*

М5: См:=(См)$_{обр}$;

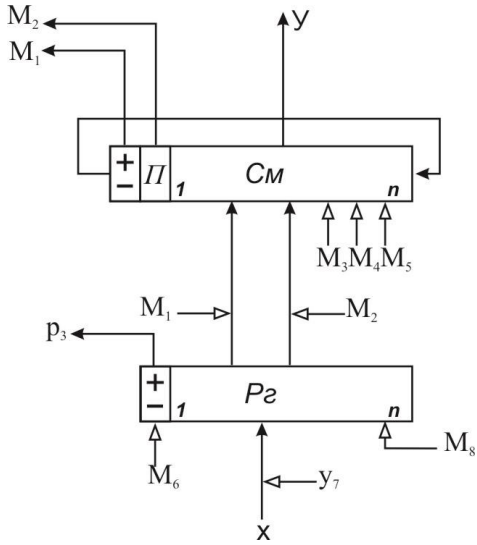


Рис. 10.7. Операційний пристрій для виконання операції додавання.

Передачі оберненого і прямого коду з регістра на суматор виконуються мікрооперaціями М1 і М2. При передачі оберненого коду в знакові розряди суматора заноситься код 11. За сигналом М3 = 1 проводиться включення суматора на додавання. При цьому сигнал М3 = 1 включає ланцюги міжрозрядного переносу, які блокуються, якщо М3 = 0. Для додавання кодів, збережених на суматорі і регістрі, необхідно одночасно виконати наступні дві мікрооперації: М3 = 1, М2: См:=Рг. В результаті цього на суматорі виконується дія См:=См+Рг. Мікрооперація М5 використовується для інвертування цифрових розрядів коду, що зберігається на суматорі. При виконанні мікрооперації М5 значення знакових розрядів суматора не змінюється.

Мікрооперація М6 використовується для інвертування знаку коду, що зберігається на регістрі.

Для управління процесом додавання використовуються сигнали p_1, p_2, p_3, Сигнал p_1 визначає значення знакового розряду суматора sign См, сигнал p_2 - значення розряду переповнення См [П], та сигнал p_3 - значення знакового розряду регістра sign Рг.

Нехай на момент початку операції додавання на суматорі зберігається прямий код першого доданка і на регістрі - прямий код другого доданка. Мікропрограма операції додавання наведена на малюнку. 10.8. У мікропрограмі поруч з операторними вершинами вказані найменування сигналів, що приймають одиничне значення при виконанні відповідних операторів мікропрограми.

Якщо код має від'ємний знак sign См =1, то виконання мікропрограми починається з інвертування коду першого доданка, що зберігається на суматорі. Далі, в залежності від знаку другого доданка sign Рг проводиться додавання коду, що зберігається на суматорі, з прямим кодом (sign Рг = 0) або оберненим кодом (sign Рг =1) другого доданка, що зберігається на регістрі. Якщо знак результату sign См = 1, то проводиться перетворення оберненого коду від'ємного результату в прямий код. Переповненню розрядної сітки суматора відповідає нерівнозначність значень p_1 и p_2, що відповідають знаковим розрядами суматора. Якщо, $\overline{p}_1 p_2 \vee p_1 \overline{p_2} = 1$ то в центральний пристрій управління видається сигнал М9, за яким признаку переповнення (ПП) присвоюється значення 1.

Час виконання мікропрограми, тобто час виконання операції додавання залежить від значень доданків. Якщо перший доданок і результат представляються додатними значеннями і не відбувається переповнення розрядної сітки суматора, то операція додавання виконується за один такт. У найгіршому випадку для виконання операції додавання потрібно чотири такти машинного часу.

Приклад побудови відміченої ГСА автомата Мілі

Проведемо розмітку змістовної ГСА відповідно до наступних правил:

1) символом стану a_1 відмітимо вхід вершини, наступної за вершиною «Початок», а також вхід вершини «Кінець»;

2) входи всіх вершин, наступних за операторними, відзначимо символами a з послідовними індексами;

3) якщо вихід вершини відмічається, то тільки одним символом;

4) входи різних вершин, за винятком вершини «Кінець», відмічаються різними символами;

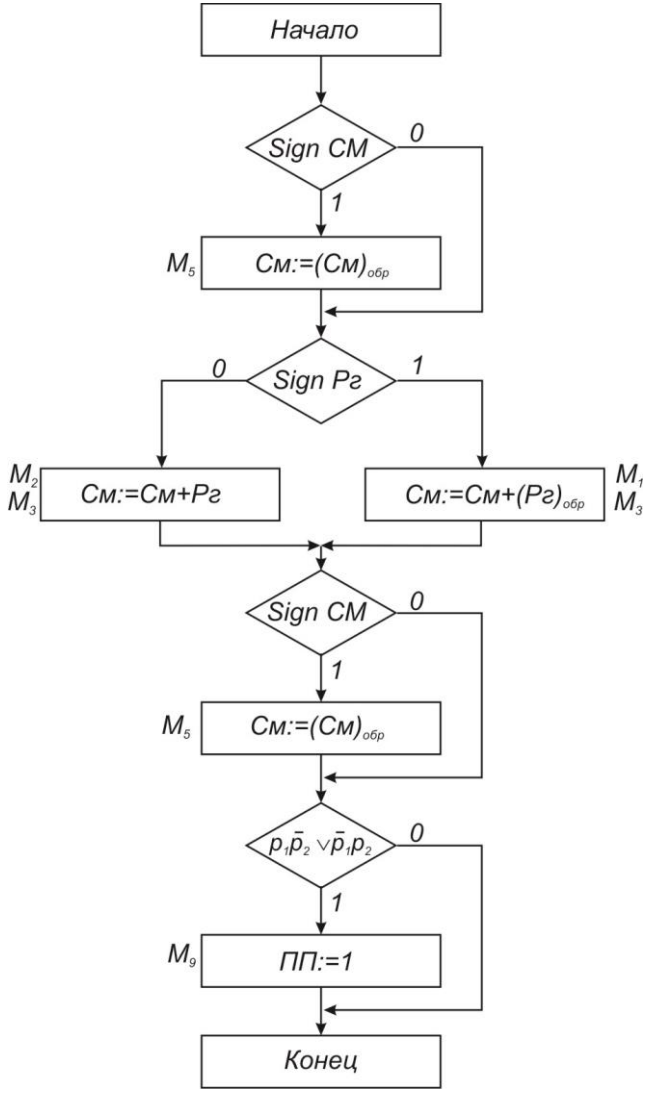


Рис. 10.8. Змістовна граф схема операції додавання чисел з фіксованою комою

5) змістовні терміни мікрооперацій і логічних умов замінюються їхніми умовними позначеннями: у кожній операторній вершині послідовно проставляються символи вихідних сигналів, якщо в різних операторних вершинах

записані однакові мікрооперації, то дозволяється їх відзначати однаковими символами вихідних сигналів; якщо в різних умовних вершинах записані однакові логічні умови, то дозволяється їх відзначати однаковими символами вхідних сигналів.

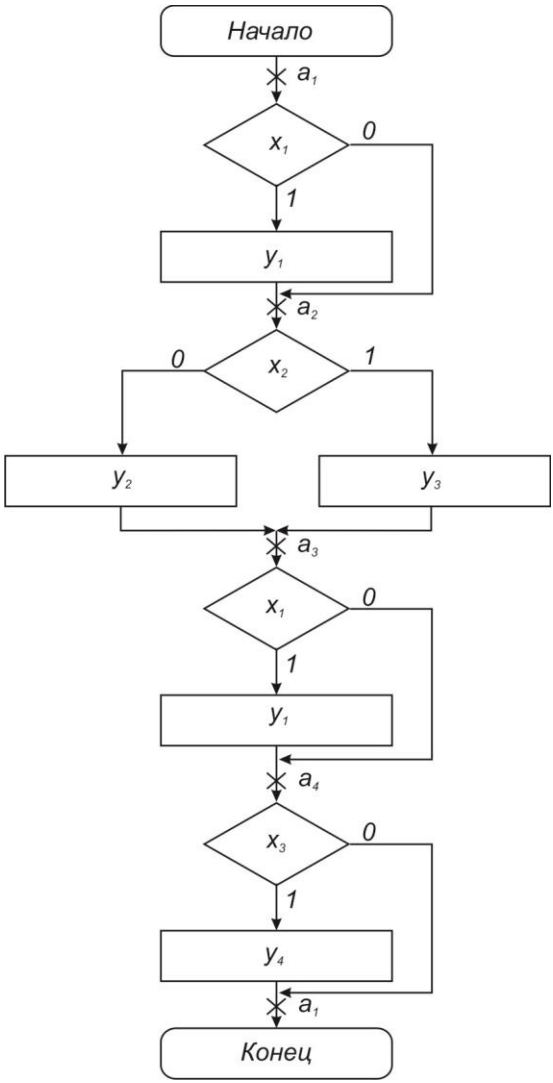


Рис. 10.9.- Відмічена ГСА автомата Мілі

Відмічена ГСА автомата Мілі для змістовної ГСА (Рис. 10.8.) після розмітки за наведеним алгоритмом представлена на рисунку 10.9.

За відміченою ГСА можна побудувати таблицю переходів автомата. (Таблиця 10. 1.)

Таблиця 10.1

Вхідний сигнал / Стан	x_1	$\bar{x}_1$	x_2	$\bar{x}_2$	x_3	$\bar{x}_3$
a_1	a_2/ y_1	$a_2/-$				
a_2			a_3/ y_3	a_3/ y_2		
a_3	a_4/ y_1	$a_4/-$				
a_4					a_1 /y_4	$a_1/-$

На перетині рядка стану, із якого виходить автомат і стовпчика вхідного сигналу проставляється стан, в який переходить автомат під впливом даного вхідного сигналу і вихідний сигнал, що формується в результаті цього переходу.

Також можна побудувати граф автомата Мілі. Він матиме 4 різних вершини, тому що є 4 різних стани на відміченій ГСА.

Між двома вершинами графа проходить дуга, якщо на відміченій ГСА між вершинами з мітками a_i і a_k, проходить шлях. Над дугою ставиться вхідний сигнал, що дорівнює кон'юнкції логічних умов відповідного шляху у відміченій ГСА. При цьому виконанню логічної умови відповідає змінна без заперечення, а невиконанню логічної умови - змінна з запереченням на відповідній дузі графа переходів автомата.

Якщо у відміченій ГСА між згаданими вершинами з мітками a_i та a_k проходять кілька шляхів, то в графі переходів автомата на дузі, що зв'язує a_i та a_k через символ диз'юнкції перераховуються всі кон'юнкції, що відповідають наявним шляхам.

Граф автомата Мілі, побудований по таблиці переходів (таблиця 10.1) представлений на рисунку 10.10.

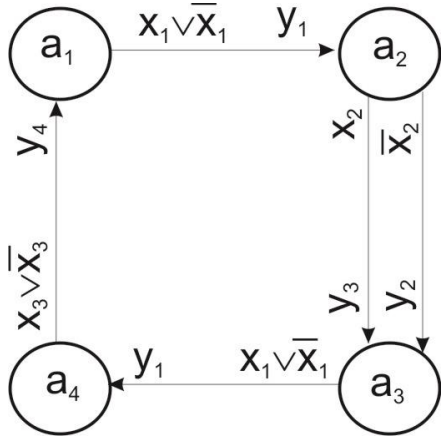


Рис. 10.10.- Граф автомата Мілі.

Надалі синтез автомата може проводитися за допомогою описаного раніше канонічного методу структурного синтезу автомата.

Приклад побудови відміченої ГСА автомата Мура

Якщо необхідно побудувати мікропрограмний автомат Мура, то змістовна ГСА управляючого автомата розмічається відповідно до наступних правил:

1) символом a_1 відмічаються вершини «Початок» та «Кінець»;

2) різні операторні вершини відзначаються різними символами;

3) всі операторні вершини повинні бути відмічені.

4) змістовні терміни мікрооперацій і логічних умов замінюються їх умовними позначеннями.

Відмічена ГСА автомата Мура (по змістовній ГСА рисунок 10.8.) після розмітки за наведеним алгоритмом представлена на рисунку 10.11.

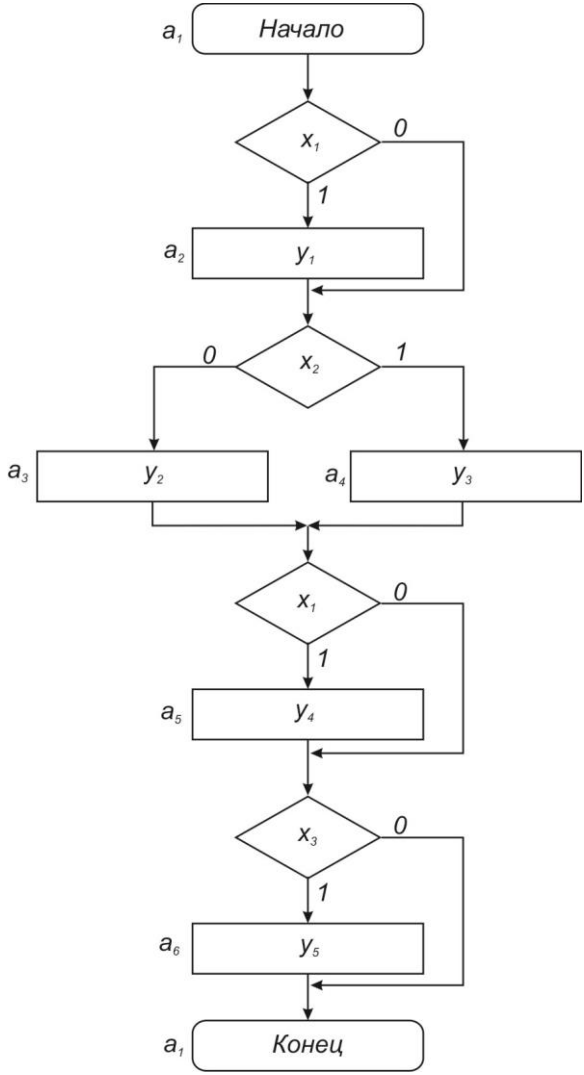


Рис. 10.11.- Відмічена ГСА автомата Мура

За відміченою ГСА автомата Мура можна побудувати таблицю переходів даного автомата. (Таблиця 10.2.)

Таблиця 10.2

Вхідний сигнал / Стан	x_1	$\bar{x}_1\bar{x}_2$	$\bar{x}_1 x_2$	$\bar{x}_2$	x_2	$\bar{x}_1\bar{x}_3$	$\bar{x}_1 x_3$	x_3	$\bar{x}_3$	1	Вихідний сигнал
a_1	a_2	a_3	a_4								-
a_2				a_3	a_4						y_1
a_3	a_5					a_1	a_6				y_2
a_4	a_5					a_1	a_6				y_3
a_5								a_3	a_1		y_4
a_6										a_1	y_5

Після отримання відміченої ГСА будується граф переходів автомата. Він має стільки різних вершин, скільки різних букв a_i з індексами мається на відміченій ГСА. Кожна вершина графа переходів автомата відзначається буквою а з відповідним індексом.

Між двома вершинами графа проходить дуга, якщо на відміченій ГСА між вершинами з мітками a_i і a_k, проходить шлях. Над дугою ставиться вхідний сигнал, що дорівнює кон'юнкції логічних умов відповідного шляху у відміченій ГСА. При цьому виконанню логічної умови відповідає змінна без заперечення, а невиконанню логічної умови - змінна з запереченням на відповідній дузі графа переходів автомата.

Якщо у відміченій ГСА між згаданими вершинами з мітками a_i і a_k є кілька шляхів, то в графі переходів автомата на дузі, що зв'язує a_i і a_k через символ диз'юнкції перераховуються всі кон'юнкції, що відповідають наявним шляхам.

Якщо будується граф переходів автомата Мура, то символи мікрооперацій (вихідні сигнали управляючого автомата) записуються поряд з відповідними вершин.

Якщо у відміченій ГСА є безумовний перехід між операторними вершинами, тобто шлях, що не проходить ні через які умовні вершини, то на графі переходів автомата йому відповідає дуга, якій приписується вхідний сигнал «1», що показує, що даний перехід в автоматі здійснюється при надходженні чергового синхросигналу.

Надалі синтез проводиться за допомогою описаного раніше методу структурного синтезу. Підкреслимо, що вхідними сигналами синтезованого структурного автомата є кон'юнкції булевих змінних (або диз'юнкції кон'юнкцій), кожна з яких відображає шлях через відповідні умовні вершини відміченої ГСА, а вихідними сигналами - мікрооперації, що позначають або вершини, або дуги графа переходів автомата, залежно від його типу. Використовуючи канонічний метод структурного синтезу, можна побудувати функціональну схему автомата.

10.8 Побудова управляючих автоматів з програмуємою логікою на основі ПЗП

На відміну від управляючого автомата з жорсткою логікою, закон функціонування якого забезпечується належним чином з'єднаними логічними елементами, в автоматах, побудованих на основі ПЗП, задана мікропрограма реалізується в явній формі і зберігається в пам'яті у вигляді послідовності управляючих слів. Управляюче слово визначає порядок роботи пристрою протягом одного такту і називається ***мікрокомандою*** (МК). Вона містить інформацію про мікрооперації, які повинні виконуватися в даному такті, і (або) про адресу наступної мікрокоманди.

Формат мікрокоманди в загальному випадку може мати операційно-адресну структуру.

Операційна	Адресна

Мікрокоманда може містити наступні частини:

У		X		A		P	
1	m	1	L	1	P	1	k

- операційну частину У, що складається з одного або декількох полів, в кожному записується номер вихідного сигналу y_j, що виробляється в даному такті;

- адресну частину, що складається з поля X, в яке записується номер логічної умови x_i (зазвичай єдиної), що перевіряється в даному такті;

- а також з поля А, в яке записується інформація про адресу наступної мікрокоманди;

- службову частину Р, що містить допоміжну управляючу інформацію.

Узагальнена структурна схема УА, виконаного на основі ПЗП, дана на рисунку 10.12.

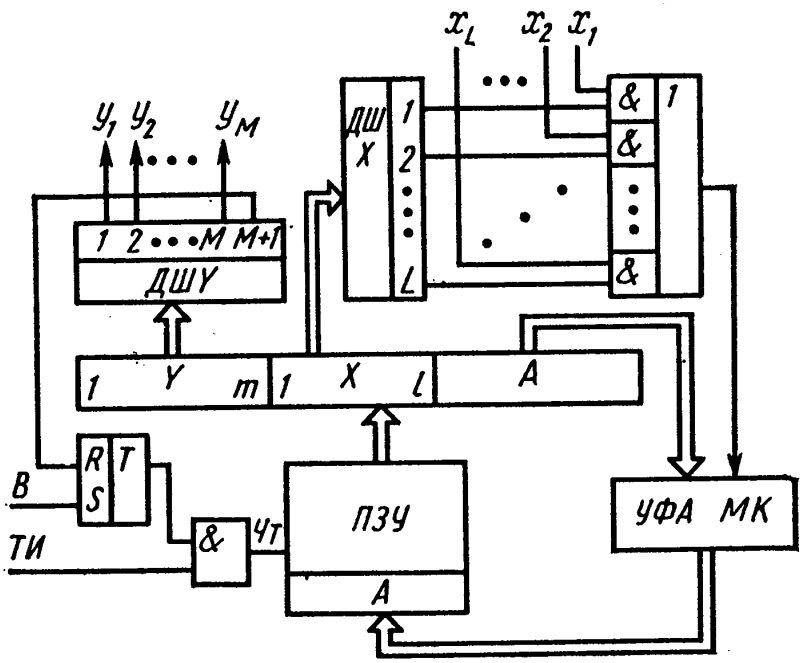


Рис. 10.12. - Узагальнена структурна схема управляючого автомата, побудованого на основі ПЗП

Перед початком роботи на УА подається сигнал СБРОС (RESET), що встановлює всі тригери автомата і регістра мікрокоманд (РМК) в нульовий стан. Цим забезпечується занесення вмісту нульової комірки ПЗП в РМК при надходженні першого тактового імпульсу після подачі стартового сигналу В.

За допомогою дешифратора ДШУ виробляється відповідний вихідний сигнал y_j, а за допомогою ДШХ визначається номер логічної умови x_i, що перевіряється в даному такті. В залежності від значення x_i, що пройшов через схему вибору ЛУ, та інформації, що надходить з адресного поля А, пристрій формування адреси наступної МК (УФ АМК) виробляє адресу комірки ПЗП, вміст якої буде переписано в РМК в наступному такті. На УФ АМК може також надходити зовнішній управляючий сигнал V, що забезпечує, наприклад, вибір певного алгоритму з тих, чиї мікропрограми зберігаються в ПЗП автомата.

Схема управління СУ (у деяких варіантах УА вона може бути відсутньою) дозволяє роботу ДШУ або ДШХ залежно від змісту службової частини Р формату

команд. В останньому такті виконання мікропрограми на виході ДШК виробляється додатковий сигнал $y_{m+}1$, використовуваний як сигнал F, який зупиняє роботу автомата і здійснює обнулення всіх його тригерів.

Таким чином, структура УА з збереженою в ПЗП логікою стандартна, і в цьому полягає одна з переваг розглянутої реалізації автомата. У цьому випадку основні зусилля спрямовуються не на отримання структурної схеми, а на складання кодованої мікропрограми, яка записується в комірки ПЗП, тобто центр ваги при розробці УА зміщується з апаратних на програмні засоби.

Процедура побудови УА з збереженою в ПЗП логікою по наявній ГСА полягає в наступному.

1. Вибирають спосіб адресації і формат мікрокоманд, причому прагнуть скоротити число двійкових розрядів у форматі МК, що, як правило, дозволяє зменшити об'єм обладнання ПЗП. При цьому враховують реальну швидкодію окремих вузлів УА і необхідність забезпечення заданої швидкодії автомата в цілому. При необхідності використовують структурні методи підвищення швидкодії УА.

2. Проводять розмітку ГСА відповідно до правил, які визначаються вибраним способом адресації.

3. Складають кодовану мікропрограму у вигляді таблиці, рядки якої відповідають відміткам на ГСА.

4. Вибирають типи необхідних мікросхем і складають структурну і принципову схеми автомата.

Виконавши перераховані етапи, переходять до технічної реалізації УА, яка багато в чому залежить від способу запису інформації в використовуваний ПЗП.

Розглянемо особливості виконання окремих етапів зазначеної процедури.

При побудові УА використовуються головним чином два види адресації:

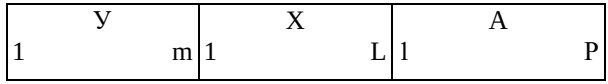
а) примусова (у кожній МК вказується адреса наступної МК);

б) природна (адреса наступної МК в явному вигляді вказується лише в деяких МК, а в інших випадках вона приймається рівною збільшеній на одиницю адресі попередньої мікрокоманди).

Формат МК при примусовій адресації може містити як два адресних поля A0, A1, (Рис.. 10.13. а) так і одне A0 (Рис.. 10.13. б).

У		X		A0		A1	
1	m	1	L	1	P	1	P

а)



б)

Рис. 10.13. Формат МК при примусовій адресації.

У першому випадку адреса наступної МК визначається залежно від значення умови x_i, яка перевіряється в даному такті, наступним чином: в якості адреси використовується вміст поля *A0,* якщо x_i =0, і поля А1 - якщо x_i = 1; безумовні переходи здійснюються за адресою *A0.*

У другому випадку переходи при хі = 0, а також безумовні переходи здійснюються за адресою *A0, а* переходи при x_i = 1 здійснюються до комірки ПЗП з адресою А1= *A0* + 1. Додавання одиниці до *A0* може бути здійснене за допомогою комбінаційної схеми інкрементора в блоці УФАМК.

При використанні двох адресних полів А0 і А1 розмітка ГСА здійснюється наступним чином.

1. Початкова вершина відзначається символом s_0.

2. Кожна операторна вершина, а також кінцева вершина відзначаються символом Si, що відрізняється від інших вершин. Якщо число вихідних сигналів y_j, записаних в деякій вершині, перевищує число операційних полів у форматі команди, то число відміток у такої вершини збільшують відповідним чином.

3. Відзначається також кожна умовна вершина, якщо її вхід пов'язаний з входом іншого умовної вершини; це викликано тим, що в кожному такті аналізується лише одна логічна умова *xi.*

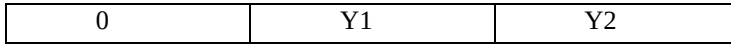
Далі кожній позначці si співставляється комірка ПЗП з тією ж адресою (номером) і таким чином складається таблиця вмісту ПЗП. Ця таблиця є основним результатом логічного проектування автомата наряду з принциповою схемою УА.

Розмітка ГСА при використанні єдиного адресного поля АТ здійснюється за цими ж правилами, до яких додається ще одне:

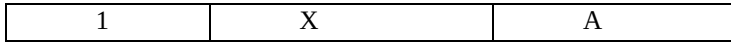
4. Присвоюються додаткові відмітки s', та s", і т.д. кожній умовній вершині, до якої підходить декілька стрілок від інших умовних вершин, так щоб загальна кількість позначок у такої вершини була рівною числу згаданих стрілок.

Необхідність збільшення числа відміток і числа використовуваних комірок ПЗП зумовлена обмеженнями в розташуванні мікрокоманд в комірках ПЗП через взаємний зв'язок адрес *A0* та А1 = А0 + 1. Швидкодія УА дещо знижується в порівнянні з випадком використання двох адресних полів за рахунок витрат часу на роботу інкрементора. Однак виключення поля А, з формату МК дозволяє зменшити розрядність ПЗП.

Подальше скорочення розрядності ПЗП досягається шляхом переходу до природної адресації мікрокоманд, при якій зазвичай використовуються МК двох типів: операційні та управляючі. (Рис.. 10.14, а, *б).* Типи МК розрізняються за значенням однорозрядного поля признака Р: 0, якщо МК операційна і 1, якщо МК управляюча.



а) операційна мікрокоманда



б) управляюча мікрокоманда

Рис. 10.14. Формат МК при природній адресації.

Обчислення адреси наступної МК проводиться за допомогою лічильника мікрокоманд (СМК), який передбачається в структурній схемі УА (мал. 10.15.). Операційна МК задає коди вироблюваних сигналів y_j і після її виконання автомат переходить до наступної МК по порядку їх розташування в комірках ПЗП, тобто здійснює перехід за адресою (СМК)+1, де СМК означає вміст лічильника мікрокоманд.

Управляюча МК, яка містить поле логічної умови Х і адресне поле А, використовується для зміни природного порядку виконання МК, тобто для здійснення умовних і безумовних переходів у відповідності зі значенням умови *Xi,* яка перевіряється. Якщо Xi=1, то перехід здійснюється за адресою, записаною в полі А, для чого його вміст переписується в СМК. Якщо *Xi* = 0 або здійснюється безумовний перехід, то наступну МК вибирають за адресою (СМК)+1. Таким чином, кожен такт роботи УА розділяється на ряд мікротактів, протягом яких виконуються дії пов’язані з формуванням вихідних сигналів y_j і виробленням адреси наступної МК.

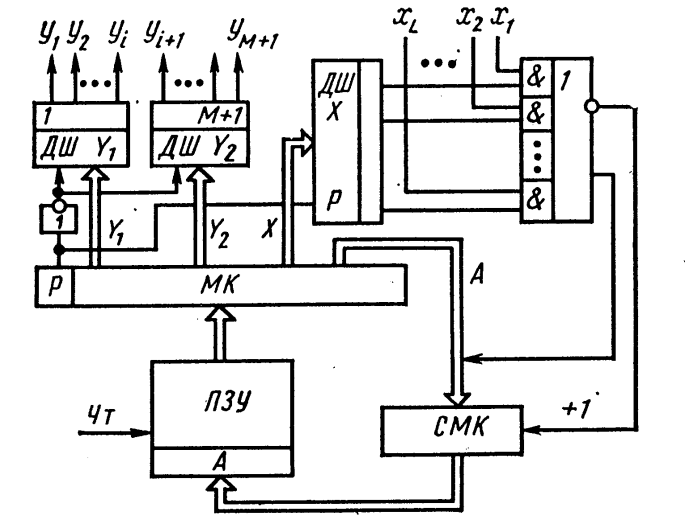


Рис. 10.15. - Структурна схема УА на основі ПЗП при використанні природної адресації

Розмітку ГСА здійснюють за правилами, в яких враховується та обставина, що аналіз Xi і вироблення сигналів y_j відбуваються тепер в різних тактах: початкова вершина відзначається символом s_0, кожна із наступних вершин отримує відмінну від інших позначку s_i; кажна вершина, до якої підходить *h* стрілок, отримує додаткові відмітки s_i^j (1=1,2, ..., h-1) так, щоб загальна кількість відміток у цієї вершини стала рівною *h*.

При складанні мікропрограми основній відмітці кожної операційної вершини ставиться у відповідність операційна МК, а основній відмітці кожної умовної вершини - управляюча МК, що реалізує умовний перехід. Кожній додатковій відмітці s_i^j ставиться у відповідність управляюча МК, що реалізує безумовний перехід в комірку ПЗП, що відповідає основній відмітці s_i.

Складання таблиці починають з позначки s_1 і послідовно розглядають вершини ГСА в напрямку стрілок. При проході через умовну вершину спочатку рухаються у напрямку стрілок, відзначених нулем (оскільки такому руху відповідає природна адресація МК). Адресні поля управляючих МК тимчасово залишаються незаповненими. Дійшовши до кінцевої відмітки, повертаються вгору по таблиці до першої управляючої МК із незаповненим адресним полем і записують в це поле адресу наступної по порядку вільної комірки. Далі продовжують рух по ГСА від

умовної вершини, якій відповідає дана управляюча МК, в напрямку дуги, зазначеної одиницею. Описану процедуру повернення вгору по таблиці повторюють до заповнення адресних полів всіх управляючих МК, забезпечуючи тим самим проходження всіх шляхів на ГСА.

Порівняння розглянутих трьох варіантів реалізації УА на основі ПЗУ з примусовою адресацією і двома адресними полями, з примусовою адресацією і одним адресним полем, з природною адресацією показує, що найменшу розрядність ПЗП забезпечує варіант з використанням природної адресації. При цьому час реалізації заданого алгоритму виявляється найбільшим, в основному через збільшення загального числа виконуваних мікрокоманд. Якщо при обраному способі адресації об'єм обладнання побудованого УА виявляється надмірним або ж швидкодія автомата недостатньою, то можна прийняти деякі додаткові заходи.

Так, для зменшення об'єму ПЗУ знаходять раціональне розбиття повної множини вихідних сигналів y_j на підмножини, кожній з яких виділяється своє операційне поле У так, щоб загальна кількість розрядів операційної частини формату МК була найменшою. Скорочення довжини адресної частини формату МК можна отримати сторінковою організацією (сегментацією) ПЗП.

При цьому ПЗП розбивається на сегменти по 2^q комірок у кожному і адреса кожної формується з двох частин: з адреси (номеру) відповідного сегмента і адреси комірки в ньому. Спеціальною мікрокомандою адреса сегмента, в межах якого здійснюється робота, записується в окремий регістр або лічильник, а в наступних МК вказується лише адреса комірки в сегменті.

Основним засобом підвищення швидкодії УА є організація випереджаючої вибірки мікрокоманд, тобто організація конвеєрного режиму роботи мікропрограмного ДП. При цьому процес вибірки і дешифрування наступної МК поєднується в часі з процесом виконання попередньої МК в ОА.

Інші можливості підвищення швидкодії УА полягають в паралельній вибірці декількох МК, які потім обробляються в порядку, який задається мікропрограмою, а також в організації паралельного аналізу декількох логічних умов при здійсненні складних переходів. Однак, ці заходи вимагають істотного збільшення обсягу обладнання.

1	0	1	0	…….	0	0	0	0	1

1	0	1	1

1	0	1	0	…….	0	0	0	0			1

10.9 Загальна структура мікропроцесорного обчислювального пристрою

При створенні сучасної радіоелектронної апаратури використовуються три основні підходи до реалізації дискретних пристроїв (ДП): апаратний, програмний та апаратно-програмний. При апаратному отримують ДП з традиційною «жорсткою» логікою, що забезпечує найбільшу швидкодію пристроїв, але вимагає трудомісткої розробки індивідуальної структури ДП.

При програмному підході ДП реалізується у вигляді програми для готової універсальної ЕОМ, в якості якої можна використовувати мікроЕОМ, призначену для вбудовування безпосередньо в розроблювані блоки.

Апаратно-програмний підхід передбачає розробку як програмних, так і апаратних засобів. Сюди відноситься реалізація ДП у вигляді автомата з мікропрограмним управлінням та збереженою в ПЗП програмою, а також побудова ДП на основі мікропроцесора (МП). Цей варіант відкриває широкі можливості для застосування сучасних ВІС і дозволяє найбільшою мірою узгодити апаратно-програмні засоби, які розробляються, з особливостями розв'язуваних задач.

Мікропроцесор представляє собою функціонально закінчений цифровий пристрій, виконаний у вигляді однієї або декількох ВІС і призначений для виконання операцій з обробки інформації та управління у відповідності до програми, яка зберігається в пам’яті. Необхідно відзначити, що термін «мікропроцесор», незважаючи на широке поширення, не має строгого визначення. Це зумовлено насамперед наявністю великої кількості типів МП, які сильно розрізняються між собою, а також їх постійним розвитком.

У вузькому сенсі МП збігається з центральним процесорним елементом (ЦПЕ) обчислювального пристрою, виконаним на основі ВІС. ЦПЕ зазвичай використовується в якості основного елемента мікропроцесорного обчислювального пристрою МПОП, схема якого представлена на мал. 10.16.

Мікропроцесорний обчислювальний пристрій мінімальної конфігурації містить ЦПЕ, блоки ПЗП і ОЗП, тактовий генератор ГТІ і блок інтерфейсу (ІФ), через який здійснюється зв'язок із зовнішніми пристроями (ВУ). Будемо вважати, що мікропроцесорний обчислювальний пристрій, що представляє собою спеціалізований обчислювальний пристрій, використовується в апаратурі для виконання деякого заданого алгоритму обробки інформації (або сукупності алгоритмів).

Тому основна програма роботи мікропроцесорного обчислювального пристрою записується в ПЗП, яке служить також для зберігання різних підпрограм, констант, таблиць та інших даних, відомих уже на етапі проектування пристрою.

ОЗП використовується для зберігання даних, що надійшли з ВУ або підготовлених для видачі в ВУ, а також проміжних результатів обчислень і деякої адресної інформації.

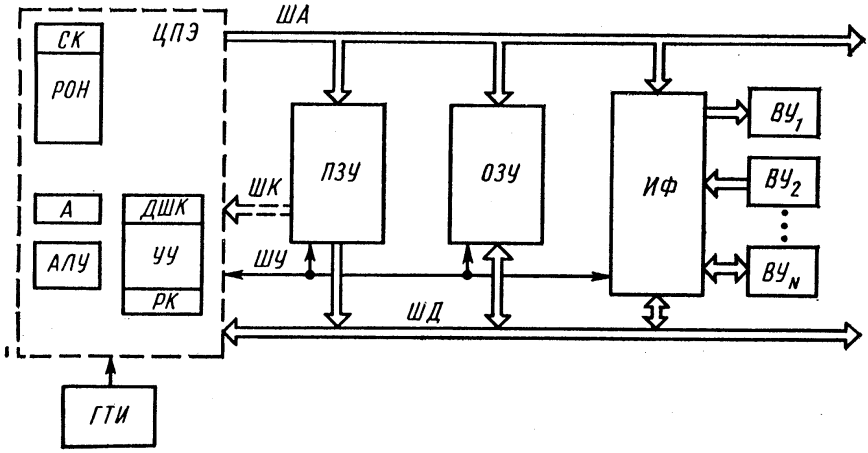


Рис. 10.16. Узагальнена структурна схема мікропроцесорного обчислювального пристрою.

Блок ГТІ, який виконується, як правило, на основі кварцового генератора, призначений для вироблення серій тактових імпульсів і деяких допоміжних сигналів, необхідних для роботи ЦПЕ і синхронізації інших блоків системи.

Інтерфейс являє собою сукупність шин для передачі інформації, електронних схем, спеціальних сигналів і алгоритмів, які управляють обміном інформації. Блок інтерфейсу служить для сполучення сигналів мікропроцесорного обчислювального пристрою і зовнішніх пристроїв по тимчасовим і електричним параметрам, а також в необхідних випадках для перетворення даних і управління обміном.

До основних вузлів ЦПЕ відносяться: керуючий пристрій (КП) з регістром команд (РК) і дешифратором команд (ДШК); арифметико-логічний пристрій (АЛП) з акумулятором (А), який є основним робочим регістром; блок регістрів загального призначення (РОН) з лічильником команд (СК).

Зв'язок між блоками мікропроцесорного обчислювального пристрою здійснюється за допомогою ряду шин: шини адреси (ША), шини даних (ШД), шини управління (ШУ), шини команд (ШК).

Можливі різні варіанти організації шин: використовується одна двонаправлена шина даних, або дві одно направлені (одна з яких є вхідною для ЦПЕ, а інша - вихідною), шина команд може поєднуватися з шиною даних при забезпеченні тимчасового розділення сигналів і т. д.

Узагальнено процес виконання команди в мікропроцесорному обчислювальному пристрої можна розбити на дві фази: фазу вибірки коду команди і фазу її виконання. Фаза вибірки складається з трьох кроків: спочатку адреса команди з СК виставляється на ША, потім відбувається вибірка коду команди з ПЗП і передача його через ШК або ШД в регістр команд ЦПЕ, після чого проводиться дешифрування цього коду в ДШК.

У відповідності з кодом команди УУ починає виробляти послідовність керуючих сигналів, необхідних для її виконання. Фаза виконання команди починається з підготовки операндів (тобто оброблюваних даних), яка полягає у визначенні місця розташування операндів і їх розміщенні в необхідних вузлах, після чого ЦПЕ переходить до виконання операції, заданої кодом команди. У цей час у СК формується адреса наступної команди і вся описана послідовність роботи мікропроцесорного обчислювального пристрою повторюється. Більш детально процес роботи мікропроцесорного обчислювального пристрою розглядається при вивченні конкретних серій мікропроцесорів.

В залежності від вимог реального застосування мікропроцесорного обчислювального пристрою в мінімальну конфігурацію системи можуть бути введені: контролер пріоритетних переривань (КПП); контролер прямого доступу до пам'яті (КПДП); програмований паралельний адаптер (інтерфейс) ППА); програмно-керований зв'язковий інтерфейс (ПСИ); програмований таймер (ПТ), і т. д.

Блок КПП сприяє організації роботи мікропроцесорного обчислювального пристрою в реальному часі тим, що дає можливість використовувати тимчасовий зовнішній пристрій, що викликає переривання. Блок КПДП дозволяє прискорити обмін масивами даних між зовнішніми пристроям та памяттю за рахунок виключення ЦПЕ з ланцюга передачі інформації. Блоки ППІ та ПСІ дозволяють організувати обмін між ЦПЕ і зовнішніми пристроям інформацією, що надається відповідно в паралельному і послідовному кодах. Блок ПТ служить для вироблення тимчасових затримок програмованої тривалості і міток часу, що сприяє організації роботи мікропроцесорного обчислювального пристрою в реальному часі.

Для реалізації цих блоків в багатьох мікропроцесорних комплектах ВІС передбачені відповідні інтегральні схеми. Крім перерахованих типових блоків в мікропроцесорні обчислювальні пристрої можуть вводитися нестандартні блоки, спеціально розроблені для вирішення конкретних задач.

СПИСОК ЛІТЕРАТУРИ

1. Самофалов К.Г., Романкевич А.М., и др. Прикладная теория цифровых автоматов. – Киев: Вища школа, 1987.
2. Пєтух А.М., Войтко В.В. Прикладна теорія цифрових автоматів. Навчальний посібник. - Вінниця, ВДТУ, 2001. -77 с.
3. Кочубей О.О. Прикладна теорія цифрових автоматів. Логічні основи: навч. посіб./ О.О.Кочубей, О.В.Сопільник.-Д.: ДНУ; Вид-во ДНУ,2009.-264с.
4. Жабін В.І., Жуков І.А., Клименко І.А., Ткаченко В.В. Прикладна теорія цифрових автоматів: Навч. Посібник.- К.: Книжкове вид-во НАУ, 2007.-364с.
5. Матвієнко М. П. Комп'ютерна логіка. Навчальний посібник. — К.: Видавництво Ліра-К, 2012. — 288 с.
6. Разживін О.В. Комп’ютерна логіка. Навчальний посібник для студентів денної та заочної форм навчання спеціальності 123 "Комп’ютерна інженерія"/ О.В. Разживін, Єнікєєв О.Ф.: Краматорськ: ДДМА, - 2020.-116с.
7. Комп’ютерна логіка. Курсова робота. [Електронний ресурс] : навч. посібн. для здобувачів ступеня бакалавра за освітньою програмою «Комп’ютерні системи та мережі» спеціальності 123 «Комп’ютерна інженерія» / Укладачі: В. І. Жабін, О. А. Верба; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 1,67 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2022. – 52 с.
8. Теорія цифрових автоматів та формальних мов. Вступний курс : навч. посібник / Гавриленко С. Ю., Клименко А. М., Любченко Н.Ю. та ін. – Харків : НТУ "ХПІ", 2011. – 176 с
9. Лысиков Б.Г. Арифметические и логические основы цифровых автоматов. - Минск: Вышэйшая школа, 1980.
10. Поснов Н.Н. Арифметика вычислительных машин в упражнениях и задачах: системы счисления, коды. –Мн.: Изд-во ”Университетское”, 1984.
11. Глушков, В.М. Синтез цифровых автоматов [Текст] / В.М. Глушков. – М.: Государственное издательство физико-математической литературы, 1962.
12. Верьовкін Л.Л. Цифрові логічні автомати. Методичні рекомендації до практичних занять для здобувачів вищої освіти першого бакалаврського рівня за спеціальністю 153 «Мікро- та наносистемна техніка» освітньопрофесійної програми «Мікро- та наносистемна техніка». Запоріжжя : ЗНУ, 2021. 32 с.

Методичне забезпечення

1. Конспект лекцій дисципліни “Комп’ютерна логіка та цифрові автомати“ для студентів, напрям підготовки 122 "Комп'ютерні науки", 123 "Комп'ютерна інженерія" (електронне видання)) / Розр: О.К. Лифар. – Київ: Вид-во СНУ ім. В. Даля, 2023. – 239 с.
2. Методичні вказівки до практичних занять з дисципліни “Комп’ютерна логіка та цифрові автомати“ для здобувачів першого (бакалаврського) рівня освіти за

спеціальністю F3 – Комп’ютерні науки, F7 – Комп’ютерна інженерія (Електронне видання) / Уклад.: О.К. Лифар. – Київ: вид-во СНУ ім. В. Даля, 2023. - 225 с.

3. Методичні вказівки до виконання контрольних робіт з дисципліни «Комп’ютерна логіка та цифрові автомати» для студентів заочної форми напряму підготовки 122 «Комп’ютерні науки» та 123 «Комп’ютерна інженерія» (електронне видання)) / Розр: О.К. Лифар. – Сєвєродонецьк: Вид-во СНУ ім. В. Даля, 2018. – 54 с.

Навчальне видання

КОНСПЕКТ ЛЕКЦІЙ

дисципліни **«Комп'ютерна логіка та цифрові автомати»**
(для здобувачів вищої освіти денної та заочної форми навчання
галузь знань: F – Інформаційні технології
спеціальності: F3 – Комп'ютерні науки та
F7 – Комп'ютерна інженерія)

Укладач ЛИФАР Олена Костянтинівна

Оригінал - макет О.К. Лифар

Підписано до друку__.__.202__.

Формат 60х84 1/16. Папір типогр. Гарнітура Times.

Друк офсетний. Умов. друк. арк. ___. Обл.-вид. арк.___.

Тираж ___ екз. Вид. № ___. Замов. № ____. Ціна договірна.

Видавництво Східноукраїнського національного університету

імені Володимира Даля

Свідоцтво про реєстрацію: серія ДК № 1620 від 18.12.03 р.
Адреса університета: вул. Іоанна Павла II., 17
м. Київ, 01042, Україна
e-mail: vidavnictvosnu.ua@gmail.com