МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ
УНІВЕРСИТЕТ ІМЕНІ ІВАНА ПУЛЮЯ

Кафедра комп’ютерних систем та мереж

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни
„Комп’ютерна логіка”
(3 семестр)
для студентів денної форми навчання
за напрямом
6.050102 „Комп’ютерна інженерія”

Тернопіль, 2016

Конспект лекцій розроблений у відповідності з навчальним планом напрямку 6.050102 „Комп’ютерна інженерія ”

Укладач: к.т.н. Тиш Є.В.

Рецензент:

Відповідальний за випуск: зав. каф. КС, к.т.н., доц.Осухівська Г.М.

Затверджено на засіданні кафедри комп’ютерних систем та мереж, протокол №____ від «___» _____2016 р.

Схвалено та рекомендовано до друку методичною комісією факультету комп’ютерно-інформаційних систем і програмної інженерії Тернопільського національного технічного університету імені Івана Пулюя, протокол №____ від «___» _____2016р.

Конспект лекцій складений з врахуванням методичних розробок інших вищих закладів освіти, а також матеріалів літературних джерел, перелічених в списку.

ЗМІСТ

ПЕРЕДМОВА

Конспект лекцій з курсу «Комп'ютерна логіка» призначений для допомоги студентам, що навчаються за напрямом «Комп'ютерна інженерія», в опануванні нормативної дисципліни «Комп'ютерна логіка».

Метою викладання дисципліни є:

- ознайомлення студентів з основними поняттями та моделями, що використовуються при вивченні дискретних обчислювальних, керуючих та вимірювальних пристроїв;

- навчання методам проектування (опис, синтез, аналіз, моделювання, діагностика) комбінаційних схем, синхронних та асинхронних автоматів із пам'яттю, що є моделями дискретних обчислювальних, керуючих та вимірювальних пристроїв.

Навчальний посібник складається з десяти розділів, в яких у логічній послідовності розкрито теми, що є базовими для дисципліни «Комп'ютерна логіка». Усі розділи книги логічно взаємопов'язані. Автори дотримувалися певної схеми викладання матеріалу, яка дає змогу побудувати вивчення предмета за зростанням складності. Усі твердження супроводжуються прикладами, що дадуть змогу студентові легко опанувати теоретичний матеріал. Кожен розділ закінчується висновками, що акцентують увагу читача на основних моментах матеріалу, контрольних запитань та набором ретельно підібраних задач, що спонукатимуть студента до перевірки отриманих знань.

У результаті вивчення матеріалу студенти отримують відомості про:

- представлення інформації в цифрових автоматах (арифметичні та логічні операції над числами у двійковій та двійково-десятковій системах числення);

- основні поняття та моделі дискретних автоматів;

- загальні та спеціальні методи синтезу та аналізу комбінаційних схем у різних елементних базисах, критерії та алгоритми оптимізації схем;

- методи опису та синтезу синхронних та асинхронних автоматів із пам'яттю;

- методи контролю та діагностики, навики побудови перевіряючих та діагностуючих тестів для схем дискретних пристроїв.

Студенти в процесі роботи над матеріалами посібника "Прикладна теорія цифрових автоматів" отримують такі навики:

- розроблення опису алгоритмів роботи комбінаційних схем, синхронних та асинхронних автоматів з пам'яттю на кількох мовах (система булевих функцій, граф автомата, таблиця переходів та виходів) із врахуванням оптимальності вибору необхідного опису;

- виконання основних етапів синтезу схем із врахуванням заданого елементного базису проектування;

- забезпечення перевірки правильності результатів синтезу автоматів шляхом їх аналізу, синхронним та асинхронним моделюванням.

ЛЕКЦІЯ 1

КОМП'ЮТЕРНА ЛОГІКА ТА ТЕОРІЯ ЦИФРОВИХ АВТОМАТІВ

Комп'ютерна логіка – це навчальна дисципліна, що охоплює логічні, математичні та технічні основи, базові принципи, поняття та моделі обчислювальних та управляючих цифрових систем.

Фундаментальною логічною моделлю комп'ютерної системи, що вивчається у рамках предмету «Комп'ютерна логіка» є поняття автомату. Термін «автомат» має декілька смислових аспектів. По-перше, автомат – це пристрій (система), який без участі людини здійснює відбір, опрацювання, передавання та зберігання інформації (даних, повідомлень, сигналів) відповідно до закладеного у нього алгоритму. З іншого боку, автомат – це математична модель реальної технічної системи (наприклад, комп'ютера чи певного його вузла, елемента). У цьому випадку автомат розглядається як деяка «чорна скринька», що має скінченну кількість входів та виходів, а також деяку множину внутрішніх станів. У цьому контексті, *теорія цифрових автоматів* вивчає математичні моделі перетворювачів (автоматів) дискретної інформації, яка подана у цифровій формі. З певної точки зору такими перетворювачами є як реальні пристрої (обчислювальні машини, живі організми), так й абстрактні системи (наприклад, *формальна система* – це сукупність абстрактних об'єктів, не пов'язаних із зовнішнім світом, в яких подано правила оперування множиною символів у строго синтаксичному трактуванні без урахування смислового навантаження, тобто семантики).

З практичної точки зору, *автомат* – це деякий пристрій, призначений для перетворення інформації. У цьому сенсі автомат можна вважати інформаційною системою. Отже, поняття «автомат» суттєво пов'язане з поняттям «інформація».

Термін «інформація» має багато означень. У широкому сенсі *інформація* – відображення реального світу. Існує означення інформації й у вузькому сенсі: *інформація* – будь-які відомості, що є об'єктом відбору, передавання, перетворення та зберігання.

Важливе питання теорії інформації – встановлення міри, кількості та якості інформації. Інформаційні міри, переважно, розглядаються у трьох аспектах: структурному, статистичному та семантичному.

У *структурному аспекті* розглядається будова масивів інформації та їх зміна простим підрахунком інформаційних елементів або комбінаторним методом. Структурний підхід застосовується для оцінювання можливостей інформаційних систем незалежно від умов їх використання.

При *статистичному підході* використовується поняття ентропії як міри невизначеності, що враховує ймовірність появи того чи іншого повідомлення.

Семантичний підхід дає змогу виділити корисність або цінність інформаційного повідомлення.

Інформація надходить у систему передавання даних у формі повідомлень. Під *повідомленням* розуміють сукупність знаків або первинних сигналів, що містять інформацію. Джерело повідомлень, у загальному випадку, утворює

сукупність джерела інформації (досліджуваного або спостережуваного об’єкта) й первинного перетворювача (давача, людини-оператора тощо), що сприймає інформацію про процес, що протікає в ньому.

Розрізняють дискретні та неперервні повідомлення.

Дискретні повідомлення формуються в результаті послідовного видавання джерелом повідомлень окремих елементів – *знаків, символів, букв*. Множину різних знаків називають *алфавітом* джерела повідомлення, а число знаків – *об’ємом алфавіту*. Якщо повідомлення є дискретними, то пристрій обробки такої інформації називають дискретним пристроєм або *дискретним автоматом*.

Неперервні повідомлення не поділені на елементи. Вони описуються функціями часу, що означені на континуальних множинах.

Для передавання повідомлення каналом зв’язку йому ставлять у відповідність певний сигнал. Під *сигналом* розуміють фізичний процес, що відображає, переносить повідомлення.

Перетворення повідомлення в сигнал, зручний для передавання каналом зв’язку, називають *кодуванням* у широкому сенсі. Операцію відновлення повідомлення по прийнятому сигналу називають *декодуванням*.

Закодовані таким чином сигнали називають *цифровими сигналами*. Подання сигналу в цифровій формі практично завжди дає суттєву перевагу при передаванні, зберіганні та опрацюванні інформації.

У сучасних дискретних автоматах прийнято ототожнювати букви довільних алфавітів з цифрами тієї або іншої системи числення. Тому дискретні автомати прийнято називати *цифровими автоматами*.

Як правило, на практиці вдаються до операції представлення початкових знаків в іншому алфавіті з меншим числом знаків. Пристрій, що виконує таку операцію, називають таким, що кодує, або *кодером*. У такому разі кожному знаку відповідає деяка послідовність символів, яку називають *кодовою комбінацією*.

Кількість символів у кодовій комбінації називають її *значущістю*, кількість ненульових символів – *вагою*.

Для операції зіставлення символів зі знаками початкового алфавіту використовують термін "декодування". Технічна реалізація цієї операції здійснюється декодуючим пристроєм, або *декодером*.

Найтісніше теорія автоматів пов’язана з *теорією алгоритмів*. Це пояснюється тим, що автомат перетворює дискретну інформацію по кроках у дискретні моменти часу та формує результуючу інформацію по кроках заданого алгоритму. Ці перетворення можливі за допомогою технічних та/або програмних засобів. При аналізі автоматів вивчають їх поведінку при різних збурюючих впливах та мінімізують число станів автомата для роботи згідно з заданим алгоритмом. Такий автомат називають *абстрактним*, оскільки абстрагуються від реальних фізичних вхідних та вихідних сигналів, розглядаючи їх просто як букви деякого алфавіту. При синтезі автоматів формують систему з елементарних автоматів, що є еквівалентною заданому абстрактному автомату. Такий автомат називають *структурним*.

Особливе місце в теорії автоматів займає поняття скінченного автомата. *Скінченний автомат* – математична абстракція, що дає змогу описувати шляхи зміни стану об'єкта залежно від його поточного стану та вхідних даних, за умови, що загальна можлива кількість станів та множина вхідних сигналів скінченні. Скінченний автомат є частковим випадком абстрактного автомата.

До складу цифрових автоматів обов'язково входять запам'ятовуючі елементи (елементи пам'яті). Вихідні сигнали в таких автоматах формуються залежно не тільки від вхідних сигналів, але й від деякої передісторії – *станів автомата*, тобто сигналів, які надійшли на входи автомата раніше.

Ще однією особливістю цифрових автоматів є властивість *стрибкоподібного переходу* автомата з одного стану в інший. Такий перехід можна вважати миттєвим, хоча для будь-якого реально існуючого автомата має місце скінченна тривалість перехідних процесів. Інше припущення, яке стосується роботи цифрового автомата, полягає в тому, що після переходу автомата в довільний стан перехід у наступний стан може бути здійсненим не раніше, ніж через деякий фіксований проміжок часу $\Delta t > 0$, який називають *інтервалом дискретності автомата*. Таке припущення дає змогу розглядати функціонування цифрового автомата в *дискретному часі*. При побудові автоматів з дискретним автоматним часом розрізняють синхронні та асинхронні автомати.

У *синхронних автоматах* моменти часу, в які необхідна зміна стану автомата, визначаються спеціальним пристроєм – генератором синхронізуючих імпульсів. Сусідні моменти часу при цьому поділені рівними часовими проміжками.

В *асинхронних автоматах* моменти переходів з одного стану в інший заздалегідь не визначаються та можуть здійснюватися через нерівні між собою проміжки часу.

Як вже зазначалося, зміни станів цифрового автомата здійснюються під впливом вхідних сигналів, які виникають поза автоматом і передаються в автомат скінченною кількістю вхідних каналів. Вхідний сигнал розглядається як миттєвий, хоча насправді він має скінченну тривалість. Зазначимо, що реальний фізичний вхідний сигнал, що викликає зміну стану автомата, у момент часу t може закінчитися до настання цього моменту, однак він відноситься саме до поточного моменту часу t, а не до попереднього $t-1$.

Результатом роботи цифрового автомата є генерування вихідних сигналів, які генеруються по скінченній кількості вихідних каналів. Реальний фізичний вихідний сигнал $w(t)$, що належить до моменту часу t, з'являється завжди після вхідного сигналу $z(t)$, що відповідає цьому ж моменту часу. Стосовно моменту часу t переходу автомата зі стану $a(t-1)$ у стан $a(t)$, то сигнал $w(t)$ може фактично з'являтися або раніше, або пізніше цього моменту.

У першому випадку приймається, що вихідний сигнал $w(t)$ однозначно визначається вхідним сигналом $z(t)$ та станом $a(t-1)$ автомата в попередній момент часу, в другому випадку сигнал $w(t)$ однозначно визначається парою

$(z(t), a(t))$. Вважають, що для будь-якого моменту часу завжди має місце лише одна із цих можливостей (одночасно для всіх переходів).

Цифрові автомати, в яких вихідний сигнал $w(t)$ визначається парою $(z(t), a(t-1))$, називають *автоматами першого роду*, а автомати, в яких сигнал $w(t)$ визначається парою $(z(t), a(t))$ – *автоматами другого роду*.

Цифровий автомат (першого або другого роду) називається *правильним*, якщо вихідний сигнал $w(t)$ визначається одним лише його станом ($a(t-1)$ або $a(t)$) та не залежить явно від вхідного сигналу $z(t)$.

Автомати першого роду зазвичай називають *автоматами Мілі*, а автомати другого роду – *автоматами Мура*.

Загальна теорія цифрових автоматів за наведених вище припущень розбивається на дві великі частини: *абстрактну теорію цифрових автоматів* та *структурну теорію цифрових автоматів*. Відмінність між ними полягає в тому, що в абстрактній теорії не враховується структура як самого автомата, так і структури його вхідних та вихідних сигналів. Вхідні та вихідні сигнали розглядаються при цьому просто як букви двох фіксованих для цього автомата алфавітів: вхідного та вихідного. Не цікавлячись способом побудови автомата, абстрактна теорія вивчає лише ті переходи, які зазнає автомат під впливом вхідних сигналів, та ті вихідні сигнали, які він при цьому видає.

На противагу абстрактній теорії, структурна теорія автоматів враховує структуру автомата та його вхідних та вихідних сигналів. У структурній теорії вивчаються способи побудови автоматів з кількох елементарних автоматів, способи кодування вхідних та вихідних сигналів елементарними сигналами, що передаються по реальних вхідних та вихідних каналах. Структурна теорія автоматів є продовженням та подальшим розвитком абстрактної теорії.

Контрольні запитання

1. Дайте означення терміну «комп'ютерна логіка».
2. Дайте означення цифрового автомату.
3. Наведіть задачі, які вивчає теорія цифрових автоматів. З яких основних частин вона складається?
4. Наведіть основні означення понять теорії інформації – «інформація», «повідомлення», «знак», «символ», «сигнал».
5. Яка операція опрацювання інформації називається кодуванням та декодуванням?
6. Наведіть класифікацію цифрових автоматів.

ЛЕКЦІЯ 2

ОСНОВНІ ПОНЯТТЯ АЛГЕБРИ ЛОГІКИ

В алгебрі логіки, що використовується для опису цифрових пристроїв, які працюють у двійковому представленні інформації, основними поняттями є *логічна* (*булева*) *змінна* та *логічна функція* (*функція алгебри логіки*).

Логічна (*булева*) *змінна* – така величина *x*, яка може приймати тільки два значення $x \in \{0,1\}$. Самі значення 0 та 1 булевих змінних називають *булевими константами*.

Логічна функція (*функція алгебри логіки*) – функція $f(x_1,...,x_n)$, область значень якої є двоелементною множиною $\{0,1\}$ та яка залежать від булевих змінних $x_1,...,x_n$, що набувають значення із цієї ж двоелементної множини. Логічні функції також називають *функціями перемикання* або *перемикальними функціями*.

Логічні функції можуть бути задані трьома способами:

1. За допомогою *таблиці істинності* – таблиці, в якій кожній інтерпретації (тобто набору аргументів) функції поставлено у відповідність її значення (наприклад, таблиця 2.1 – таблиця істинності логічної функції від трьох змінних). В таблиці істинності кожній змінній та значенню самої функції відповідає по одному стовпчику, а кожній інтерпретації – по одному рядку. Кількість рядків у таблиці відповідає кількості різних інтерпретацій логічної функції.

Таблиця 2.1. Таблиця істинності логічної функції від трьох змінних

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

2. *Порядковим номером* логічної функції. Кожній логічній функції ставиться у відповідність її порядковий номер у вигляді натурального числа, двійковий код якого зображує стовпчик значень функції у таблиці істинності. Молодшим розрядом вважається найнижчий рядок (значення функції на інтерпретації (1,1,…,1)), а старшим – самий верхній (значення функції на інтерпретації (0,0,…,0)). Вказаний порядковий номер функції, як двійковий, так й десятковий, однозначно задає булеву функцію. Кожній інтерпретації булевої функції також ставиться у відповідність свій номер – значення двійкового коду, який зображує інтерпретація. Інтерпретації, що записана у верхньому рядку таблиці істинності, ставиться у відповідність номер 0, потім йде інтерпретація

номер 1 тощо. У найнижчому рядку розташована інтерпретація за номером $2^n - 1$, де n – кількість змінних, від яких залежить булева функція.

3. *Аналітично* (у вигляді формули). *Формула* – це вираз, що містить булеві функції та їхні суперпозиції. *Суперпозицією* називають спосіб отримання нових функцій, шляхом підстановки одних функцій замість аргументів інших функцій. Точніше, суперпозицією булевої функції f_0 та функцій $g_i(x_1,\ldots,x_m), i \in \{1,\ldots,k\}$ називають складну функцію

$$f(x_1,\ldots,x_m) = f_0(g_1(x_1,\ldots,x_m),\ldots,g_k(x_1,\ldots,x_m)),$$

при цьому деякі з функцій g_i можуть тотожно співпадати з однією зі змінних x_j, $j \in \{1,\ldots,m\}$.

Формули, що зображують одну й ту ж функцію, називають *еквівалентними* або *рівносильними*. Еквівалентність формул позначається знаком рівності «=».

Окрім *еквівалентності* «=» в алгебрі логіки означено три операції:

- диз'юнкція (логічне додавання, функція АБО), яка позначається знаком «+» або «$\cup$» (наприклад, $f(x,y) = x + y = x \cup y$);

- кон'юнкція (логічне множення, функція І), яка позначається крапкою, яку можна опускати, а також знаками «$\cap$» або «&» (наприклад, $f(x,y) = x \cdot y = xy = x \cap y = x \,\&\, y$);

- інверсія (заперечення, функція НІ), що позначається рискою над логічними змінними або над константами 0 та 1 (наприклад, $\bar{x}$, $\bar{1}$, $\bar{0}$).

Операція еквівалентності задовольняє такі властивості:

- $x = x$ – *рефлексивність*;

- якщо $x = y$, то $y = x$ – *симетричність*;

- якщо $x = y$ та $y = z$, то $x = z$ – *транзитивність*.

Із відношення еквівалентності випливає *принцип підстановки*: якщо $x = y$, то в будь-якій формулі, що вміщує x, замість x можна підставити y, й буде отримана еквівалентна формула.

Алгебра логіки визначається такою системою аксіом

$$\begin{cases} x = 0, & якщо \quad x \neq 1, \\ x = 1, & якщо \quad x \neq 0. \end{cases} \tag{2.1}$$

$$\begin{cases} 1+1=1, \\ 0 \cdot 0 = 0. \end{cases} \tag{2.2}$$

$$\begin{cases} 0+0=0, \\ 1 \cdot 1 = 1. \end{cases} \tag{2.3}$$

$$\begin{cases} 0+1=1+0=1, \\ 1 \cdot 0 = 0 \cdot 1 = 0. \end{cases} \tag{2.4}$$

$$\begin{cases} \bar{0} = 1, \\ \bar{1} = 0. \end{cases} \tag{2.5}$$

Аксіома (2.1) стверджує, що в алгебрі логіки розглядаються лише двійкові змінні, аксіоми (2.2)–(2.4) означують операції диз'юнкції та кон'юнкції, а аксіома (2.5) – операцію заперечення.

Для алгебри логіки особливе значення мають логічні функції від двох змінних. Справа у тому, що відповідно до принципу замкненості відносно операції суперпозиції задання логічних функцій лише для двох змінних дає змогу визначити усі логічні функції для будь-якої кількості двійкових аргументів.

Кількість логічних функцій двох змінних дорівнює $M = 2^{2^2} = 16$. Упорядкована послідовність наборів та відповідних логічних функцій наведена в таблиці 2.2.

Таблиця 2.2 Логічні функції від двох змінних

Аналітичний запис	Назва	x_1x_2				Функція
		00	01	10	11	
$y_0 = 0$	константа 0	0	0	0	0	y_0
$y_1 = x_1 \cdot x_2$	кон'юнкція	0	0	0	1	y_1
$y_2 = x_1 \Delta x_2 = x_1 \cdot \bar{x}_2$	заборона за x_2	0	0	1	0	y_2
$y_3 = x_1 = x_1\bar{x}_2 + x_1x_2$	змінна x_1	0	0	1	1	y_3
$y_4 = x_2 \Delta x_1 = \bar{x}_1 \cdot x_2$	заборона за x_1	0	1	0	0	y_4
$y_5 = x_2 = \bar{x}_1x_2 + x_1x_2$	змінна x_2	0	1	0	1	y_5
$y_6 = x_1 \oplus x_2$	сума за модулем 2 (нееквівалентність)	0	1	1	0	y_6
$y_7 = x_1 + x_2$	диз'юнкція	0	1	1	1	y_7
$y_8 = x_1 \downarrow x_2 = \overline{x_1 + x_2}$	стрілка Пірса (заперечення диз'юнкції)	1	0	0	0	y_8
$y_9 = x_1 \equiv x_2$	еквівалентність	1	0	0	1	y_9
$y_{10} = \bar{x}_2 = \bar{x}_1\bar{x}_2 + x_1\bar{x}_2$	інверсія x_2	1	0	1	0	y_{10}
$y_{11} = x_2 \rightarrow x_1 = x_1 \cdot \bar{x}_2$	імплікація від x_2 до x_1	1	0	1	1	y_{11}
$y_{12} = \bar{x}_1 = \bar{x}_1\bar{x}_2 + \bar{x}_1x_2$	інверсія x_1	1	1	0	0	y_{12}
$y_{13} = x_1 \rightarrow x_2 = \bar{x}_1 \cdot x_2$	імплікація від x_1 до x_2	1	1	0	1	y_{13}
$y_{14} = x_1 / x_2 = \overline{x_1x_2}$	штрих Шеффера (заперечення кон'юнкції)	1	1	1	0	y_{14}
$y_{15} = 1$	константа 1	1	1	1	1	y_{15}

Із шістнадцяти перелічених в таблиці 2.2 функцій перемикання шість є такими, що або зовсім не залежать від двох змінних, або залежать тільки від

однієї змінної. Це так звані *вироджені функції перемикання*: y_0, y_3, y_5, y_{10}, y_{12}, y_{15}. Для решти невироджених функцій перемикання застосовуються спеціальні позначення, що наведені в таблиці 2.2.

За допомогою аксіом алгебри логіки можна довести цілий ряд теорем та тотожностей. Одним із універсальних методів доведення теорем алгебри логіки є *метод перебору* всіх значень логічних змінних. Базуючись на аксіомах алгебри логіки, методом перебору легко переконатися у справедливості теорем, які називають *законами алгебри логіки* (таблиця 2.3).

Таблиця 2.3. Закони алгебри логіки

Назва закону	Формули	Номер
Закони ідемпотентності	$x + x = x$ $x \cdot x = x$	(2.6)
Закони комутативності	$x + y = y + x$ $x \cdot y = y \cdot x$	(2.7)
Закони асоціативності	$(x + y) + z = x + (y + z)$ $(x \cdot y) \cdot z = x \cdot (y \cdot z)$	(2.8)
Закон дистрибутивності для кон'юнкції відносно диз'юнкції	$x \cdot (y + z) = x \cdot y + x \cdot z$	(2.9)
Закон дистрибутивності для диз'юнкції відносно кон'юнкції	$x + y \cdot z = (x + y) \cdot (x + z)$	(2.10)
Закон виключення третього	$x + \bar{x} = 1$	(2.11)
Закон протиріччя	$x \cdot \bar{x} = 0$	(2.12)
Співвідношення для констант	$0 + x = x$ $1 \cdot x = x$ $1 + x = 1$ $0 \cdot x = 0$	(2.13)
Закони подвійності (теореми де Моргана)	$\overline{x + y} = \bar{x} \cdot \bar{y}$ $\overline{x \cdot y} = \bar{x} + \bar{y}$	(2.14)
Закон подвійного заперечення	$\overline{(\bar{x})} = \bar{\bar{x}} = x$	(2.15)
Закони поглинання	$x + x \cdot y = x$ $x \cdot (x + y) = x$	(2.16)
Закон склеювання	$x \cdot y + x \cdot \bar{y} = x$ $(x + y) \cdot (x + \bar{y}) = x$	(2.17)

Якщо в логічний вираз входять операції диз'юнкції та кон'юнкції, то потрібно зберігати порядок (пріоритет) виконання операцій: спочатку виконується операція кон'юнкції, а потім – операція диз'юнкції. У складних

логічних виразах для задавання порядку виконання операцій використовуються дужки.

В основі алгебри логіки лежить принцип двоїстості. Функцію $f^*(x_1, x_2, \ldots, x_n)$ називають *двоїстою* до функції $f(x_1, x_2, \ldots, x_n)$, якщо $f^*(x_1, x_2, \ldots, x_n) = \bar{f}(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)$. Функція, двоїста сама собі, тобто $f = f^*$, називається *самодвоїстою функцією*. Якщо дві функції рівні, то й двоїсті їм функції також рівні: якщо $f_1 = f_2$, то $f_1^* = f_2^*$.

Оскільки відношення двоїстості симетричне, то можна сказати, що кон'юнкція двоїста диз'юнкції, диз'юнкція двоїста кон'юнкції, функція «0» двоїста функції «1», та навпаки, а заперечення – самодвоїста функція.

Звідси виходить так званий *принцип двоїстості*, що вказує на правило отримання двоїстих формул у алгебрі логіки: для того, щоб отримати двоїсту формулу, необхідно замінити в ній всі кон'юнкції на диз'юнкції, диз'юнкції на кон'юнкції, 0 на 1, 1 на 0 та використовувати дужки, де необхідно, щоб порядок операцій залишався попереднім.

Контрольні запитання

1. Назвіть способи задавання булевих функцій. Охарактеризуйте кожен спосіб.
2. Назвіть основні аксіоми та закони булевої алгебри.
3. Дайте означення суперпозиції булевих функцій.
4. Який пріоритет визначений для операцій алгебри логіки? Для якої цілі служить пріоритет операцій?
5. Які формули називають еквівалентними?
6. Яким чином здійснюється перехід від формули до таблиці істинності функції?
7. Дайте означення двоїстості функції. Яким чином формується таблиця істинності двоїстої функції?

ЛЕКЦІЯ 3

АНАЛІТИЧНЕ ПОДАННЯ ФУНКЦІЙ АЛГЕБРИ ЛОГІКИ

Розглянемо фіксований набір змінних $\{x_1, x_2, \ldots, x_n\}$, на якому задана функція алгебри логіки. Оскільки будь-яка змінна $x_i = \{0,1\}$, то набір значень цих змінних фактично є деяким двійковим числом. Нехай номером набору $(x_1, x_2, \ldots, x_n)$ буде число i, що отримане таким чином:

$$i = 2^{n-1}x_1 + 2^{n-2}x_2 + \ldots + x_n. \qquad (3.1)$$

Функцію $\Phi_i(x_1, x_2, \ldots, x_n)$, яку задано як:

$$\Phi_i = \begin{cases} 0, \textit{ якщо номер набору } (x_1, x_2, \ldots, x_n) \textit{ дорівнює } i, \\ 1, \textit{ якщо номер набору } (x_1, x_2, \ldots, x_n) \textit{ не дорівнює } i; \end{cases}$$

називають *термом*.

Диз'юнктивний терм (*макстерм*, *конституента нуля*) – терм, що пов'язує всі змінні, які подано в прямій або інверсній формі, знаком диз'юнкції. Наприклад, $\Phi_2 = x_1 + \bar{x}_2 + x_3$, $\Phi_5 = x_1 + \bar{x}_2 + \bar{x}_3 + x_4$.

Кон'юнктивний терм (*мінтерм*, *конституента одиниці*) – терм, що пов'язує всі змінні, які подано в прямій або інверсній формі, знаком кон'юнкції. Позначається мінтерм наступним чином:

$$F_i = \begin{cases} 1, \textit{ якщо номер набору } (x_1, x_2, \ldots, x_n) \textit{ дорівнює } i, \\ 0, \textit{ якщо номер набору } (x_1, x_2, \ldots, x_n) \textit{ не дорівнює } i. \end{cases}$$

Наприклад, $F_2 = \bar{x}_1\bar{x}_2\bar{x}_3x_4$, $F_4 = \bar{x}_1x_2\bar{x}_3$.

Ранг терма r визначається кількістю логічних змінних, що входять у даний терм. Наприклад, для мінтерма $F_4 = \bar{x}_1\bar{x}_2$ ранг $r = 2$, для макстерма $\Phi_6 = x_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_4$ ранг $r = 4$.

На основі сказаного вище, можна сформулювати наступні теореми.

Теорема 3.1. Будь-яка таблично задана функція алгебри логіки може бути подана у вигляді

$$f(x_1, x_2, \ldots, x_n) = F_1 \cup F_2 \cup \ldots \cup F_n = \bigcup_1 F_i, \qquad (3.2)$$

де i – номер наборів, на яких функція дорівнює 1, $\bigcup_1$ – знак диз'юнкції, що об'єднує всі терми F_i, які дорівнюють одиниці.

Формула, що зображена у вигляді диз'юнкції мінтермів різних рангів, називається *диз'юнктивною нормальною формою* (*ДНФ*). Термін «нормальна» означає, що в даному виразі відсутні групові інверсії, тобто інверсії над кількома змінними одразу. Наприклад, $f(x_1, x_2, x_3) = \bar{x}_1x_2\bar{x}_3 + \bar{x}_2x_3$.

Теорема 3.2. Будь-яка таблично задана функція алгебри логіки може бути подана у вигляді

$$f(x_1, x_2, \ldots, x_n) = \Phi_1 \cap \Phi_2 \cap \ldots \cap \Phi_k, \qquad (3.3)$$

де k – кількість двійкових наборів, для яких $\Phi_i = 0$.

Формула, що зображена у вигляді кон'юнкції макстермів різних рангів (диз'юнктивних термів), називається *кон'юнктивною нормальною формою* (*КНФ*). Наприклад, $f(x_1, x_2, x_3, x_4) = (x_1 + x_2 + \bar{x}_3)(\bar{x}_1 + \bar{x}_4)$.

Нормальні кон'юнктивна та диз'юнктивна форми не дають однозначного подання функції. Таке подання можна отримати застосовуючи досконалі нормальні форми (ДНФ). ДНФ функції f має регламентовану логічну структуру та однозначно визначається за функцією f, а її побудова заснована на рекурентному застосуванні теорем про спеціальні розклади логічних функцій за змінними.

Введемо позначення $x^0 = \bar{x}$, $x^1 = x$.

Тоді в загальному вигляді змінна може бути задана як деяка функція

$$x^{\alpha} = \begin{cases} x, & якщо\ \alpha = 1, \\ \bar{x}, & якщо\ \alpha = 0, \end{cases} \tag{3.4}$$

причому

$$x^{\alpha} = \alpha x + \bar{\alpha}\,\bar{x}\ , \tag{3.5}$$

де α – двійкова змінна.

Теорема 3.3. Про диз'юнктивний розклад функції алгебри логіки $f(x_1, x_2, \ldots, x_n)$ *за* k *змінними.* Будь-яку функцію алгебри логіки $f(x_1, x_2, \ldots, x_n)$ можна зобразити у такій формі:

$$f(x_1, \ldots, x_k, x_{k+1}, \ldots, x_n) = \bigcup_{(\alpha_1, \alpha_2, \ldots, \alpha_k)} x_1^{\alpha_1} x_2^{\alpha_2} x_k^{\alpha_k} f(\alpha_1, \alpha_2, \ldots, \alpha_k, x_{k+1}, \ldots, x_n). \tag{3.6}$$

Запис $\bigcup_{(\alpha_1, \alpha_2, \ldots, \alpha_k)}$ означає багатократну диз'юнкцію, яка береться за всіма можливими наборами значень $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ при будь-якому k $(1 \le k \le n)$.

Розглянемо наслідки з теореми 3.3.

Наслідок 3.1 (до теореми 3.3). Диз'юнктивний розклад функції алгебри логіки $f(x_1, x_2, \ldots, x_n)$ *за одною змінною.* Будь-яку функцію алгебри логіки $f(x_1, x_2, \ldots, x_n)$ *можна* зобразити у такій формі:

$$f(x_1, x_2, \ldots, x_n) = \bigcup_{\alpha_i} x_i^{\alpha_i} f(x_1, x_2, \ldots, x_{i-1}, \alpha_i, x_{i+1}, \ldots, x_n). \tag{3.7}$$

Запис означає, що диз'юнкція береться за всіма значеннями α_i, тобто 0 та 1. Запис $f(x_1, x_2, \ldots, x_{i-1}, \alpha_i, x_{i+1}, \ldots, x_n)$ позначає значення функції на наборі $x_1, x_2, \ldots, x_n$, де замість значення змінної x_i підставлено α_i.

Наслідок 3.2 (до теореми 3.3). Диз'юнктивний розклад функції алгебри логіки $f(x_1, x_2, \ldots, x_n)$ *за всіма* n *змінними.* Будь-яку функцію алгебри логіки $f(x_1, x_2, \ldots, x_n) \ne 0$ можна зобразити у такій формі:

$$f(x_1, x_2, \ldots, x_n) = \bigcup_{\substack{(\alpha_1, \alpha_2, \ldots, \alpha_n) \\ f(\alpha_1, \alpha_2, \ldots, \alpha_n) = 1}} x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}. \tag{3.8}$$

Запис $\bigcup_{\substack{(\alpha_1,\alpha_2,\ldots,\alpha_n)\\ f(\alpha_1,\alpha_2,\ldots,\alpha_n)=1}}$ означає, що диз'юнкція береться за всіма наборами значень $(\alpha_1,\alpha_2,\ldots,\alpha_n)$, на яких $f(\alpha_1,\alpha_2,\ldots,\alpha_n)=1$.

Досконалою диз'юнктивною нормальною формою (*ДДНФ*) функції алгебри логіки називається формула, що зображена у вигляді диз'юнкції конституент одиниці даної функції. ДДНФ функції є результатом диз'юнктивного розкладу функції за всіма змінними та відповідає формулі (3.8). Як видно з означення, ДДНФ функції містить тільки операції диз'юнкції, кон'юнкції та заперечення, отже, застосувавши до ДДНФ принцип двоїстості, можна отримати двоїсте зображення, яке називається *кон'юнктивним розкладом*.

Теорема 3.4. Про кон'юнктивний розклад функції алгебри логіки $f(x_1, x_2, \ldots, x_n)$ *за* k *змінними.* Будь-яку функцію алгебри логіки $f(x_1, x_2, \ldots, x_n)$ можна зобразити у такій формі:

$$f(x_1, \ldots, x_k, x_{k+1}, \ldots, x_n) = \bigcap_{(\alpha_1,\alpha_2,\ldots,\alpha_k)} x_1^{\alpha_1} + x_2^{\alpha_2} + x_k^{\alpha_k} + f(\alpha_1,\alpha_2,\ldots,\alpha_k, x_{k+1}, \ldots, x_n). \quad (3.9)$$

Запис $\bigcap_{(\alpha_1,\alpha_2,\ldots,\alpha_k)}$ означає, що кон'юнкція береться за всіма можливими наборами значень $(\alpha_1,\alpha_2,\ldots,\alpha_k)$.

Розглянемо наслідки з теореми 3.4.

Наслідок 3.3 (до теореми 3.4). Кон'юнктивний розклад функції алгебри логіки $f(x_1, x_2, \ldots, x_n)$ *за одною змінною.* Будь-яку функцію алгебри логіки $f(x_1, x_2, \ldots, x_n)$ *можна* зобразити у такій формі:

$$f(x_1, x_2, \ldots, x_n) = \bigcap_{\alpha_i} x_i^{\overline{\alpha}_i} + f(x_1, x_2, \ldots, x_{i-1}, \alpha_i, x_{i+1}, \ldots, x_n). \quad (3.10)$$

Запис $\bigcap_{\alpha_i}$ означає, що кон'юнкція береться за двома можливими значеннями α_i, тобто 0 та 1. Індекс i є фіксованим.

Наслідок 3.4 (до теореми 3.4). Кон'юнктивний розклад функції алгебри логіки $f(x_1, x_2, \ldots, x_n)$ *за всіма* n *змінними.* Будь-яку функцію алгебри логіки $f(x_1, x_2, \ldots, x_n) \neq 0$ можна зобразити у такій формі:

$$f(x_1, x_2, \ldots, x_n) = \bigcap_{\substack{(\alpha_1,\alpha_2,\ldots,\alpha_n)\\ f(\alpha_1,\alpha_2,\ldots,\alpha_n)=0}} x_1^{\overline{\alpha}_1} + x_2^{\overline{\alpha}_2} + \ldots + x_n^{\overline{\alpha}_n}. \quad (3.11)$$

У правій частині (3.11) кон'юнкція береться за всіма наборами значень $(\alpha_1,\alpha_2,\ldots,\alpha_n)$, на яких $f(\alpha_1,\alpha_2,\ldots,\alpha_n)=0$.

Досконалою кон'юнктивною нормальною формою (*ДКНФ*) функції алгебри логіки називається формула, що зображена у вигляді кон'юнкції конституент нуля даної функції.

З аналізу формул (3.8) та (3.11), відповідних ДДНФ та ДКНФ логічної функції, можна зробити такі висновки:

1. Для кожної функції алгебри логіки $f(x_1, x_2, \ldots, x_n)$, що не є конституентою нуля, існує зображення у вигляді ДДНФ.

2. Для кожної функції алгебри логіки $f(x_1, x_2, \ldots, x_n)$, що не є конституентою одиниці, існує зображення у вигляді ДКНФ.

3. Дві різні функції алгебри логіки не можуть мати однакових ДДНФ та ДКНФ.

4. Для кожної функції алгебри логіки $f(x_1, x_2, \ldots, x_n)$ існує зображення у вигляді формули алгебри логіки, що містить тільки операції диз'юнкції, кон'юнкції та заперечення.

Перехід від табличної форми задавання логічних функцій до їх аналітичних форм записування

Для отримання ДДНФ та ДКНФ логічної функції, виходячи з її таблиці істинності, сформулюємо відповідні алгоритми переходу.

Алгоритм переходу від таблиці істинності логічної функції до ДДНФ полягає в наступному:

1. Виділити всі інтерпретації $(\alpha_1, \alpha_2, \ldots, \alpha_n)$, на яких значення функції дорівнює одиниці.

2. Записати конституенти одиниці виду $x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}$, що стосуються відповідних інтерпретацій.

3. Отримати ДДНФ функції за допомогою об'єднання записаних конституент одиниці операцією диз'юнкції.

Алгоритм переходу від таблиці істинності логічної функції до ДКНФ полягає в наступному:

1. Виділити всі інтерпретації $(\alpha_1, \alpha_2, \ldots, \alpha_n)$, на яких значення функції дорівнюють нулю.

2. Записати конституенти одиниці виду $x_1^{\overline{\alpha}_1} + x_2^{\overline{\alpha}_2} + \ldots + x_n^{\overline{\alpha}_n}$, що стосуються відповідних інтерпретацій.

3. Отримати ДКНФ функції за допомогою об'єднання записаних конституент нуля операцією кон'юнкції.

Алгоритм переходу від довільних формул алгебри логіки до ДДНФ та ДКНФ

Алгоритм переходу від довільної формули алгебри логіки до ДДНФ полягає в наступному:

1. Виключити константи, використовуючи закони дій з константами.

2. Опустити знаки заперечення безпосередньо на змінні, використовуючи закони де Моргана.

3. Використовуючи дистрибутивний закон, розкрити дужки. До отриманих елементарних кон'юнкцій застосувати закони ідемпотентності й протиріччя, спростити їх та звести подібні змінні. Результатом виконання вказаних дій є отримання ДНФ булевої функції.

4. Побудувати конституенти одиниці логічної функції, шляхом введення у кожну елементарну кон'юнкцію відсутніх змінних, використовуючи закон виключення третього.

5. За допомогою дистрибутивного закону розкрити дужки та звести подібні члени, використовуючи закон ідемпотентності. Отримана формула відповідає ДДНФ функції.

Алгоритм переходу від довільної формули алгебри логіки до ДКНФ можна сформулювати таким чином:

1. Виключити константи, використовуючи закони дій з константами.

2. Опустити знаки заперечення безпосередньо на змінні, використовуючи закони де Моргана.

3. За допомогою використання дистрибутивного закону звести функцію до виду кон'юнкції елементарних диз'юнкцій. До отриманих елементарних диз'юнкцій застосувати закони ідемпотентності й виключення третього, спростити їх та звести подібні змінні. Результатом виконання вказаних дій є отримання КНФ булевої функції.

4. Побудувати конституенти нуля функції, шляхом введення у кожну елементарну диз'юнкцію відсутніх змінних, використовуючи закон протиріччя.

5. За допомогою дистрибутивного закону звести функцію до виду кон'юнкції конституент нуля та спростити формулу, використовуючи закон ідемпотентності. Отримана формула є ДКНФ функції.

Контрольні запитання

1. Назвіть способи задавання булевих функцій. Охарактеризуйте кожен спосіб.

2. Назвіть основні аксіоми та закони булевої алгебри.

3. Сформулюйте означення понять нормальних та досконалих нормальних форм булевих функцій. Поясніть їх зв'язок з диз'юнктивним та кон'юнктивним розкладом булевих функцій.

4. Дайте означення таких понять: елементарна кон'юнкція, елементарна диз'юнкція, конституента одиниці, конституента нуля. Яким чином ці поняття пов'язані з диз'юнктивним та кон'юнктивним розкладом булевих функцій?

5. Назвіть властивості конституенти одиниці та конституенти нуля.

6. Сформулюйте означення понять нормальних та досконалих нормальних форм булевих функцій. Поясніть їх зв'язок з диз'юнктивним та кон'юнктивним розкладом булевих функцій.

7. Опишіть алгоритми переходу від таблиці істинності булевої функції до ДДНФ та ДКНФ.

8. Поясніть алгоритм переходу від ДДНФ булевої функції до таблиці істинності.

9. Опишіть алгоритм переходу від ДКНФ булевої функції до таблиці істинності.

10. Дайте порівняльну характеристику алгоритмів переходу від довільної формули булевої функції до ДДНФ та ДКНФ.

ЛЕКЦІЯ 4

ГРАФІЧНІ МЕТОДИ МІНІМІЗАЦІЇ ЛОГІЧНИХ ФУНКЦІЙ

4.1 Карти Карно

Карта Карно показана на рисунку 4.1.

Чотири квадрати (1, 2, 3, 4) відповідають чотирьом можливим комбінаціям x_1 та x_2 у таблиці істинності функції з двома змінними. При такому зображенні квадрат 1 відповідає добутку $\bar{x}_1\bar{x}_2$, квадрат 2 – $\bar{x}_1 x_2$ тощо.

Нехай потрібно скласти карту Карно для функції перемикання, що записана у ДДНФ, $f(x_1, x_2) = x_1\bar{x}_2 + \bar{x}_1 x_2 + x_1 x_2$.

Розташуємо логічні одиниці в усіх квадратах, яким відповідають добутки у вихідній функції перемикання на рисунку 4.2. Заповнена таким чином карта Карно тепер готова для побудови. Сусідні одиниці об'єднуються в один контур групами по дві, чотири або вісім одиниць. Кожен контур – це новий член спрощеної функції перемикання. Відзначимо, що на рисунку 4.3 отримано тільки два контури. Це означає, що нова, спрощена функція перемикання буде складатися тільки з двох членів, що пов'язані функцією АБО.

x_1	x_2	$f(x_1, x_2)$
0	0	1
0	1	2
1	0	3
1	1	4

	$\bar{x}_2$	x_2
$\bar{x}_1$	1	2
x_1	3	4

Рисунок 4.1. Позначення квадратів на карті Карно

	$\bar{x}_2$	x_2
$\bar{x}_1$		1
x_1	1	1

Рисунок 4.2. Нанесення одиниць на карту Карно

	$\bar{x}_2$	x_2
$\bar{x}_1$		1
x_1	1	1

Рисунок 4.3. Об'єднання одиниць групами в один контур

Спростимо функцію перемикання, беручи до уваги два контури на рисунку 4.3. Взявши спочатку нижній контур, зауважимо, що x_1 тут зустрічається у комбінації з x_2 та $\bar{x}_2$. У відповідності з правилами алгебри логіки x_2 та $\bar{x}_2$ доповнюють один одного та їх можна опустити. Тоді в

нижньому контурі залишається тільки x_1. Аналогічно цьому вертикально розташований контур вміщує x_1 та $\bar{x}_1$, які також можна опустити, залишивши тільки x_2. Елементи x_1 та x_2, що залишилися, об'єднуються функцією АБО, що призводить до спрощеної функції перемикання $f(x_1, x_2) = x_1 + x_2$. Алгоритм мінімізації функції перемикання записується таким чином:

1. Переведення функції перемикання в ДДНФ.
2. Нанесення одиниць на карту Карно.
3. Об'єднання сусідніх одиниць контурами, що охоплюють два, чотири або вісім квадратів.
4. Проведення спрощення, виключаючи елементи, які доповнюють один одного всередині контуру.
5. Об'єднання елементів, що залишилися (по одному у кожному контурі), функцією АБО.
6. Запис мінімізованої функції перемикання в ДНФ.

Приклад 4.1. Мінімізувати за допомогою карт Карно функцію перемикання трьох змінних $f(x_1, x_2, x_3) = x_1\bar{x}_2\bar{x}_3 + x_1\bar{x}_2x_3 + \bar{x}_1\bar{x}_2x_3 + \bar{x}_1x_2x_3$.

Карта Карно для цієї функції показана на рисунку 4.4. Нижній контур вміщує x_3 та $\bar{x}_3$, їх можна опустити. Після цього у складі нижнього контуру залишається тільки добуток $x_1\bar{x}_2$. У верхній контур входять x_2 та $\bar{x}_2$, які теж опускаються, після чого залишається тільки добуток $\bar{x}_1x_3$. Результуюча функція перемикання має вигляд $f(x_1, x_2, x_3) = \bar{x}_1x_3 + x_1\bar{x}_2$.

Приклад 4.2. Мінімізувати за допомогою карт Карно функцію перемикання

$$f(x_1, x_2, x_3, x_4) = x_1\bar{x}_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2\bar{x}_3x_4 + \bar{x}_1\bar{x}_2\bar{x}_3x_4 + \bar{x}_1x_2\bar{x}_3x_4 + \\ + x_1x_2x_3\bar{x}_4 + \bar{x}_1\bar{x}_2x_3x_4 + \bar{x}_1x_2x_3x_4.$$

Карта Карно для функції перемикання з чотирма змінними допускає 16 можливих комбінацій x_1, x_2, x_3 та x_4 (рисунок 4.5). Ці комбінації подані відповідно 16-ма квадратами карти. Нанесемо на карту 7 одиниць, які відповідають 7-ми добуткам у заданій функції перемикання. Групи із двох та чотирьох одиниць об'єднані контурами. Нижній контур із двох одиниць дає можливість опустити x_4 та $\bar{x}_4$. Після цього в ньому залишається добуток

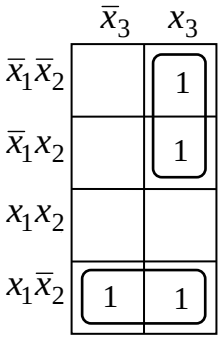


Рисунок 4.4. Спрощення функції перемикання трьох змінних

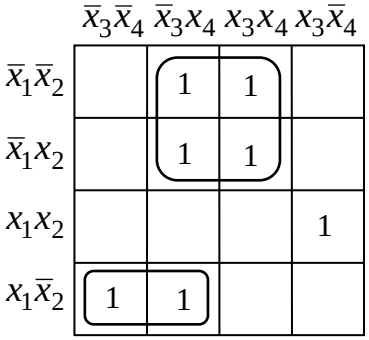


Рисунок 4.5. Мінімізація функції перемикання з чотирма змінними

$x_1\bar{x}_2\bar{x}_3$. Далі у верхньому контурі із чотирьох одиниць попарно опускаються x_3 та $\bar{x}_3$, x_4 та $\bar{x}_4$, так, що в результаті цього верхній контур дає добуток $\bar{x}_1 x_4$. Зауважимо, що добуток $x_1x_2x_3\bar{x}_4$ не був внесений у жоден із контурів, тому у мінімізованій функції перемикання він запишеться без змін. Спрощена функція перемикання має вигляд $f(x_1, x_2, x_3, x_4) = x_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_4 + x_1x_2x_3\bar{x}_4$.

Приклад 4.3. Мінімізувати за допомогою карт Карно функцію перемикання для п'яти змінних:

$$f(x_1, x_2, x_3, x_4, x_5) = x_1\bar{x}_2\bar{x}_3\bar{x}_4\bar{x}_5 + x_1x_2x_3\bar{x}_4\bar{x}_5 + \bar{x}_1x_2x_3\bar{x}_4\bar{x}_5 + \bar{x}_1x_2x_3x_4\bar{x}_5 +$$

$$+ \bar{x}_1x_2\bar{x}_3x_4\bar{x}_5 + \bar{x}_1\bar{x}_2\bar{x}_3x_4\bar{x}_5 + \bar{x}_1x_2x_3x_4\bar{x}_5 + \bar{x}_1x_2x_3\bar{x}_4x_5 + x_1x_2x_3\bar{x}_4x_5 + x_1\bar{x}_2\bar{x}_3\bar{x}_4x_5.$$

Нанесемо одиниці на карту, як це показано на рисунку 4.6. Зробимо два об'єднання по чотири одиниці та одне по дві. В результаті операції поглинання над відповідними змінними отримаємо таку МДНФ:

$$f(x_1, x_2, x_3, x_4, x_5) = x_1\bar{x}_2\bar{x}_3\bar{x}_4 + x_2x_3\bar{x}_4 + \bar{x}_1x_4\bar{x}_5.$$

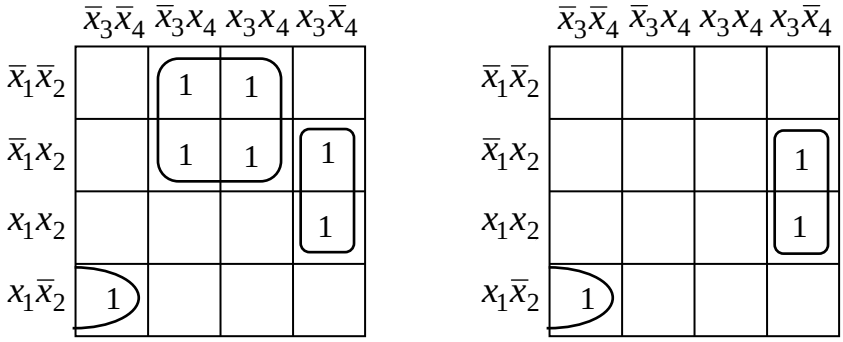


Рисунок 4.6 . Мінімізація функції перемикання для п'яти змінних за допомогою карт Карно

4.2 Діаграми Вейча

Для функції перемикання двох змінних діаграма Вейча має вигляд згідно з рисунком 4.7.

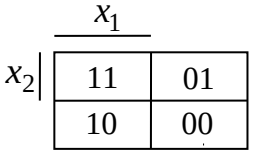


Рисунок 4.7. Діаграма Вейча для функції перемикання двох змінних

Кожна клітинка діаграми відповідає набору змінних функції перемикання в її таблиці істинності. В клітинці діаграми Вейча ставиться одиниця, якщо функція набуває одиничне значення на відповідному наборі. Нульові значення функції перемикання в діаграмі Вейча не ставляться. Для функції перемикання

трьох змінних діаграма Вейча зображена на рисунку 4.8. Додавання до неї ще такої ж таблиці дає діаграму для функції чотирьох змінних (рисунок 4.9).

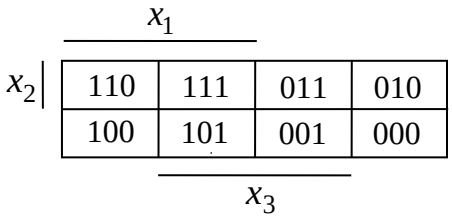


Рисунок 4.8. Діаграма Вейча для функції перемикання трьох змінних

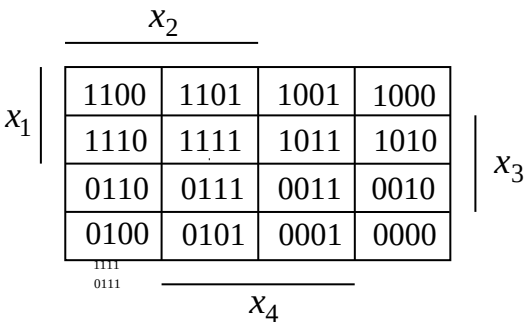


Рисунок 4.9. Діаграма Вейча для функції перемикання чотирьох змінних

Мінімізація функції перемикання полягає в знаходженні мінімального покриття всіх одиниць діаграми Вейча блоками з одиниць (вказаної конфігурації), розташованих у сусідніх клітинках діаграми. При цьому завжди вважають, що лівий край діаграми Вейча чотирьох змінних примикає до її правого краю, а верхній край діаграми примикає до нижнього її краю. Після отримання мінімального покриття всіх одиниць діаграми Вейча МДНФ булевої функції записується як диз'юнкція елементарних кон'юнкцій, що відповідають виділеним блокам одиниць у діаграмі.

Приклад 4.4. Знайти за допомогою діаграми Вейча МДНФ функції перемикання $f(x_1, x_2, x_3) = x_1x_2\bar{x}_3 + x_1x_2x_3 + x_1\bar{x}_2x_3 + \bar{x}_1\bar{x}_2x_3 + \bar{x}_1\bar{x}_2\bar{x}_3$.

Діаграма Вейча, що відповідає цій функції, зображена на рисунку 4.10. Мінімальне покриття всіх одиниць діаграми можливе тільки блоками по дві одиниці. Кожному такому блоку відповідає своя кон'юнкція, як показано на рисунку 4.10. Таким чином, МДНФ даної функції має вигляд

$$f(x_1, x_2, x_3) = x_1x_2 + \bar{x}_1\bar{x}_2 + x_1x_3.$$

Приклад 4.5. Знайти за допомогою діаграми Вейча (рисунок 4.11) МДНФ функції перемикання

$$f(x_1, x_2, x_3, x_4) = x_1x_2\bar{x}_3\bar{x}_4 + x_1x_2x_3\bar{x}_4 + \bar{x}_1x_2\bar{x}_3\bar{x}_4 + x_1x_2x_3x_4 +$$
$$+ \bar{x}_1x_2x_3x_4 + x_1\bar{x}_2x_3x_4 + \bar{x}_1\bar{x}_2x_3x_4 + x_1\bar{x}_2\bar{x}_3\bar{x}_4 + \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4.$$

МДНФ цієї функції має вигляд $f(x_1, x_2, x_3, x_4) = x_3x_4 + \bar{x}_3\bar{x}_4 + x_1x_2x_3$.

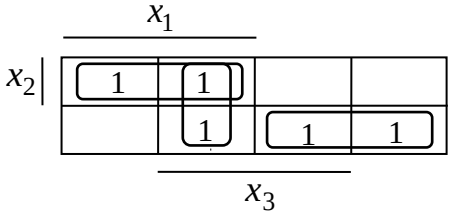


Рисунок 4.10. Мінімізація функції перемикання трьох змінних за діаграмою Вейча

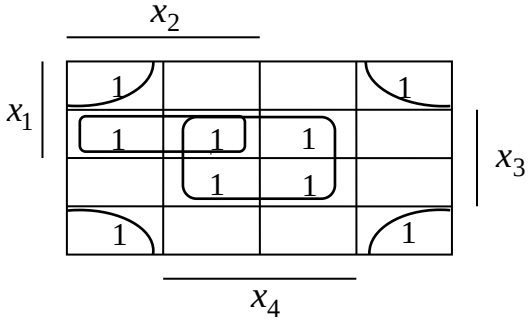


Рисунок 4.11. Мінімізація функції перемикання з чотирма змінними за діаграмою Вейча

Приклад 4.6. Знайти за допомогою діаграми Вейча МДНФ функції перемикання

$$f(x_1, x_2, x_3, x_4, x_5) = x_1x_2x_3\bar{x}_4\bar{x}_5 + \bar{x}_1x_2x_3\bar{x}_4\bar{x}_5 + x_1x_2x_3x_4\bar{x}_5 +$$
$$+ \bar{x}_1x_2x_3x_4\bar{x}_5 + x_1\bar{x}_2\bar{x}_3x_4\bar{x}_5 + \bar{x}_1\bar{x}_2\bar{x}_3x_4\bar{x}_5 + x_1\bar{x}_2\bar{x}_3x_4x_5 + \bar{x}_1\bar{x}_2\bar{x}_3x_4x_5 +$$
$$+ x_1x_2x_3x_4x_5 + \bar{x}_1x_2x_3x_4x_5 + x_1x_2x_3\bar{x}_4x_5 + \bar{x}_1x_2x_3\bar{x}_4x_5.$$

Діаграму Вейча для п’яти змінних зобразимо, як це показано на рисунку 4.12. Об’єднавши один контур по чотири одиниці та один по вісім одиниць, запишемо МДНФ цієї функції: $f(x_1, x_2, x_3, x_4, x_5) = x_2x_3 + \bar{x}_2\bar{x}_3x_4$.

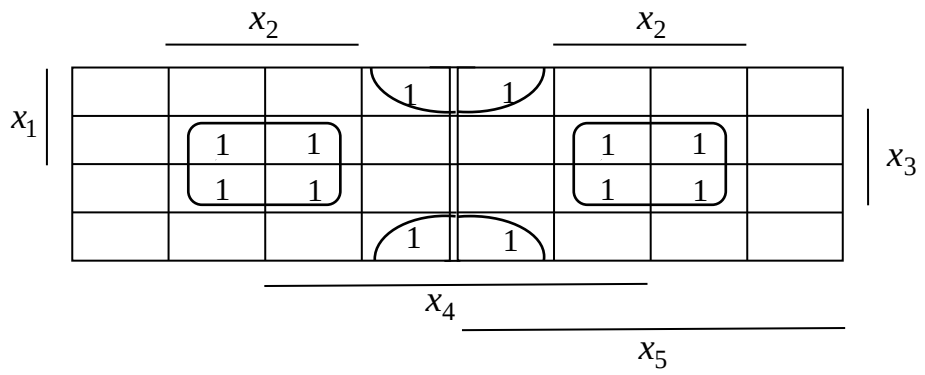


Рисунок 4.12. Мінімізація функції перемикання від п’яти змінних за допомогою діаграм Вейча

Контрольні запитання

1. Назвіть способи задавання булевих функцій. Охарактеризуйте кожен спосіб.
2. Назвіть основні аксіоми та закони булевої алгебри.
3. Сформулюйте означення понять нормальних та досконалих нормальних форм булевих функцій. Поясніть їх зв’язок з диз’юнктивним та кон’юнктивним розкладом булевих функцій.
4. Опишіть алгоритми переходу від таблиці істинності булевої функції до ДДНФ та ДКНФ.
5. Поясніть алгоритм переходу від ДДНФ булевої функції до таблиці істинності.
6. Опишіть алгоритм переходу від ДКНФ булевої функції до таблиці істинності.
7. Запишіть формули операцій диз’юнктивного склеювання та поглинання.
8. Запишіть формули операцій кон’юнктивного склеювання та поглинання.
9. Сформулюйте правило записування мінімальної ДНФ за методом карт Карно та діаграм Вейча.

ЛЕКЦІЯ 5

АНАЛІТИЧНІ МЕТОДИ МІНІМІЗАЦІЇ

5.1. Основні поняття

При проектуванні цифрових автоматів, цифрових логічних схем широко використовують методи мінімізації логічних функцій, що дають змогу отримати рекомендації для побудови оптимальних схем цифрових автоматів. Загальна задача мінімізації функцій алгебри логіки може бути сформульована наступним чином: знайти найпростішу, згідно з обраним *критерієм мінімізації*, формулу. Критерії можуть бути різними, наприклад: кількість змінних у формулі, кількість знаків кон'юнкції та диз'юнкції або комбінація подібних критеріїв.

Розглянемо мінімізацію на множині ДНФ та КНФ кількості символів змінних та операцій, що містяться у формулі. Така задача називається *канонічною задачею мінімізації.* Мінімальні форми, які отримуються в результаті розв'язання цієї задачі, називають мінімальними ДНФ та КНФ.

Наведемо декілька означень.

Імплікантою деякої логічної функції f називається функція g, така, що на всіх інтерпретаціях, на яких g дорівнює одиниці, f теж дорівнює одиниці. Мінтерми функції є її імплікантами, елементарні кон'юнкції, що входять до складу ДНФ функції, також є її імплікантами.

Множина S, що складається з імплікант функції f, називається *покриттям* (або *повною системою імплікант*) функції f, якщо кожне одиничне значення функції f покривається одиницею хоча б однієї імпліканти з множини S. Набір імплікант, складових ДНФ функції, є її покриттям. Набір усіх конституент одиниці функції, що входять до її ДДНФ, є покриттям даної функції.

Будь-яку елементарну кон'юнкцію *A*, що входить до складу елементарної кон'юнкції *B* та містить менше змінних, ніж кон'юнкція *B*, називають *власною частиною* кон'юнкції *B* та кажуть, що кон'юнкція *A покриває* кон'юнкцію *B*.

Простою імплікантою функції f називається така кон'юнкція-імпліканта, що ніяка її власна частина не є імплікантою даної функції.

Множина всіх простих імплікант функції становить *покриття* даної функції.

Диз'юнкція всіх простих імплікант функції називається її *скороченою ДНФ*.

Диз'юнктивним ядром булевої функції f називається така множина її простих імплікант, яка створює покриття f, але після видалення будь-якої імпліканти втрачає цю властивість, тобто перестає бути повною системою імплікант.

Тупиковою ДНФ називається ДНФ даної булевої функції, що складається тільки з простих імплікант.

На відміну від скороченої ДНФ, тупикова ДНФ може не містити деякі з простих імплікант функції. Кожна булева функція має єдину скорочену ДНФ та може мати кілька тупикових ДНФ. У кожну з тупикових ДНФ входять всі імпліканти диз'юнктивного ядра даної функції.

Мінімальною ДНФ (*МДНФ*) даної булевої функції називається одна з її тупикових ДНФ, якій відповідає найменше значення критерію мінімізації ДНФ.

Для знаходження множини простих імплікант функції, що задана ДДНФ, використовуються такі операції:

- операція *неповного диз'юнктивного склеювання* $Ax + A\bar{x} = A + Ax + A\bar{x}$, де A – деяка елементарна кон'юнкція логічних змінних, x – булева змінна;

- операція *диз'юнктивного поглинання* $A + Ax = A$;

- операція *повного диз'юнктивного склеювання* $Ax + A\bar{x} = A$.

Для аналізу різних зображень булевої функції через КНФ та отримання мінімальних КНФ природно трансформувати викладені вище поняття та операції склеювання за принципом двоїстості. Двоїсті поняття відповідно називаються термінами: *імпліцента*, *проста імпліцента*, *повна система імпліцент*, *скорочена КНФ*, *тупикова КНФ*, *мінімальна КНФ*.

Імпліцентою деякої функції f називається функція g, така, що на всіх інтерпретаціях, на яких g дорівнює нулеві, f теж дорівнює нулеві.

Кон'юнкція усіх імпліцент деякої функції f є *повною системою імпліцент*.

Простою імпліцентою функції f називається така диз'юнкція-імпліцента, що ніяка її власна частина не є імпліцентою даної функції.

Кон'юнкція всіх простих імпліцент функції називається її *скороченою КНФ*.

Тупиковою КНФ називається КНФ даної булевої функції, що складається тільки з простих імпліцент.

Мінімальною КНФ (*МКНФ*) даної булевої функції називається одна з її тупикових КНФ, якій відповідає найменше значення критерію мінімізації КНФ.

Операції склеювання, що використовуються для мінімізації КНФ, мають такий вигляд:

- операція *неповного кон'юнктивного склеювання* $(A + x)(A + \bar{x}) = A(A + x)(A + \bar{x})$, де A – деяка елементарна диз'юнкція змінних, x – булева змінна;

- операція *кон'юнктивного поглинання* $A(A + x) = A$;

- операція *повного кон'юнктивного склеювання* $(A + x)(A + \bar{x}) = A$.

5.2 Метод мінімізації Квайна

Метод мінімізації Квайна реалізує перехід від ДДНФ до мінімальної ДНФ із використанням операцій склеювання та поглинання.

Алгоритм Квайна складається з таких кроків:

1. Записати ДДНФ заданої логічної функції.

2. Виконати всі можливі операції неповного диз'юнктивного склеювання. Результуюча формула є диз'юнкцією всіх можливих імплікант даної функції.

3. Виконати всі можливі операції диз'юнктивного поглинання. Результуюча формула є скороченою ДНФ даної функції.

4. Скласти імплікантну таблицю та знайти диз'юнктивне ядро.

5. Спростити імплікантну таблицю за допомогою видалення рядків, що відповідають імплікантам диз'юнктивного ядра, та стовпців, що відповідають тим конституентам одиниці, які покриваються імплікантами ядра.

6. Знайти всі тупикові ДНФ даної функції.

7. Знайти мінімальну ДНФ.

Якщо при мінімізації деякої функції виходить, що спрощена імплікантна таблиця порожня, то тупикова ДНФ цієї функції є мінімальною та складається тільки з імплікант диз'юнктивного ядра. Якщо ж спрощена імплікантна таблиця не порожня, то в кожній тупиковій ДНФ функції, крім імплікант ядра, присутні й деякі прості імпліканти, що не входять до диз'юнктивного ядра. Набір даних імплікант визначає різниці між тупиковими ДНФ.

Приклад 5.1. Знайдемо мінімальну ДНФ функції

$$f(x_1, x_2, x_3) = x_1x_2x_3 + x_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2x_3 + \bar{x}_1\bar{x}_2x_3 + \bar{x}_1\bar{x}_2\bar{x}_3$$

за допомогою методу мінімізації Квайна.

Виконаємо всі можливі операції диз'юнктивного склеювання та поглинання:

$$x_1x_2x_3 + \bar{x}_1x_2x_3 = x_2x_3,$$
$$x_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2\bar{x}_3 = \bar{x}_2\bar{x}_3,$$
$$\bar{x}_1x_2x_3 + \bar{x}_1\bar{x}_2x_3 = \bar{x}_1x_3,$$
$$\bar{x}_1\bar{x}_2x_3 + \bar{x}_1\bar{x}_2\bar{x}_3 = \bar{x}_1\bar{x}_2.$$

Отримаємо формулу:

$$f(x_1, x_2, x_3) = x_1x_2x_3 + x_1\bar{x}_2\bar{x}_3 + \bar{x}_1x_2x_3 + \bar{x}_1\bar{x}_2x_3 + \bar{x}_1\bar{x}_2\bar{x}_3 =$$
$$= x_2x_3 + \bar{x}_2\bar{x}_3 + \bar{x}_1x_3 + \bar{x}_1\bar{x}_2.$$

Зробивши попарне порівняння всіх елементарних кон'юнкцій, що входять у дану формулу, робимо висновок, що отримати інші елементарні кон'юнкції за допомогою операції склеювання в даній формулі неможливо. Таким чином, отримано диз'юнкцію всіх можливих імплікант даної функції

$$f(x_1, x_2, x_3) = x_2x_3 + \bar{x}_2\bar{x}_3 + \bar{x}_1x_3 + \bar{x}_1\bar{x}_2.$$

Отримана формула є скороченою ДНФ даної функції. Складемо імплікантну таблицю (таблиця 5.1). Її рядки задаються простими імплікантами, а стовпці – конституантами одиниці функції. Зірочкою відзначається кожна клітинка таблиці, для якої імпліканта з рядка є власною частиною конституенти із стовпця.

Таблиця 5.1. Імплікантна таблиця функції $f(x_1, x_2, x_3)$

	$x_1x_2x_3$	$x_1\bar{x}_2\bar{x}_3$	$\bar{x}_1x_2x_3$	$\bar{x}_1\bar{x}_2x_3$	$\bar{x}_1\bar{x}_2\bar{x}_3$
x_2x_3	*		*		
$\bar{x}_2\bar{x}_3$		*			*
$\bar{x}_1x_3$			*	*	
$\bar{x}_1\bar{x}_2$				*	*

Знайдемо диз'юнктивне ядро. До нього входить кожна проста імпліканта, яка є єдиною у покритті будь-якої конституенти одиниці. В імплікантній таблиці по одному знаку «*» містять стовпці, що відповідають конституентам одиниці $x_1x_2x_3$ та $x_1\bar{x}_2\bar{x}_3$, навпроти простих імплікант x_2x_3 та $\bar{x}_2\bar{x}_3$. Ці прості імпліканти складають диз'юнктивне ядро. Складемо спрощену імплікантну таблицю. Для цього з імплікантної таблиці викреслюємо рядки, що відповідають імплікантам диз'юнктивного ядра, та стовпці, що відповідають конституентам одиниці, які покриті імплікантами ядра (таблиця 5.2).

У даному випадку імпліканти ядра покривають всі конституенти одиниці, крім однієї. Спрощена імплікантна таблиця має вигляд згідно з таблицею 5.3.

Таблиця 5.2. Отримання спрощеної імплікантної таблиці

	$x_1x_2x_3$	$x_1\bar{x}_2\bar{x}_3$	$\bar{x}_1x_2x_3$	$\bar{x}_1\bar{x}_2x_3$	$\bar{x}_1\bar{x}_2\bar{x}_3$
x_2x_3	*		*		
$\bar{x}_2\bar{x}_3$		*			*
$\bar{x}_1x_3$			*	*	
$\bar{x}_1\bar{x}_2$				*	*

Таблиця 5.3. Спрощена імплікантна таблиця

	$\bar{x}_1\bar{x}_2x_3$
$\bar{x}_1x_3$	*
$\bar{x}_1\bar{x}_2$	*

Зі спрощеної імплікантної таблиці знаходимо, що тупикові ДНФ даної функції, крім диз'юнктивного ядра, включають імпліканту $\bar{x}_1x_3$ або $\bar{x}_1\bar{x}_2$. Таким чином, отримуємо дві тупикові ДНФ для даної функції

ДНФ 1: $f(x_1, x_2, x_3) = x_2x_3 + \bar{x}_2\bar{x}_3 + \bar{x}_1x_3$,

ДНФ 2: $f(x_1, x_2, x_3) = x_2x_3 + \bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2$.

Вказані ДНФ містять по 6 символів змінних, а також однакову кількість знаків операцій диз'юнкції (2) і кон'юнкції (3), тому виберемо як мінімальну ДНФ 1, що містить менше знаків операцій заперечення.

5.3 Метод мінімізації Квайна–Мак-Класкі

В методі Квайна для функції більшого числа змінних перелічення варіантів склеювання та множини тупикових ДНФ є значною складністю. Удосконалення, що введене Мак-Класкі, спрощує записування мінімізаційної формули. Згідно з його пропозицією конституенти одиниці записуються у вигляді двійкового коду – номери даної конституенти. Вказаний номер відповідає інтерпретації, на якій конституента дорівнює одиниці. Всі останні імпліканти, отримані в результаті застосування операцій неповного склеювання, також записуються у вигляді двійкових кодів. У позиціях коду, що відповідають реальним змінним імпліканти, вміщується набір значень змінних, який обертає імпліканту на одиницю. У позиціях фіктивних змінних імпліканти ставиться прочерк. Таким чином, операції склеювання й поглинання проводяться не з самими імплікантами, а з їх двійковими кодами, що робить алгоритм простим у програмній реалізації.

Крім того, множина кодів імплікант поділяється на групи за кількістю одиниць, що в них утримуються. Оскільки операції неповного склеювання застосовні до імплікант, коди яких відрізняються значенням тільки в одній позиції, в склеюванні можуть брати участь тільки імпліканти, що належать до сусідніх груп, номери яких відрізняються на одиницю. Крім того, виконання операцій склеювання здійснюється покроково. На першому кроці здійснюються всі можливі склеювання конституент одиниці, на другому – склеювання на множині імплікант, що отримані на першому кроці, тощо. Ділення на групи та кроки дозволяє істотно скоротити кількість перевірок пар імплікант на можливість здійснення склеювання та робить алгоритм ефективнішим.

При виконанні операцій поглинання з формули видаляються всі імпліканти, які не є простими. Результуюча формула зображає диз'юнкцію простих імплікант функції – її скорочену ДНФ. Імпліканти, що брали участь хоча б в одному склеюванні, не є простими. Отже, замість операцій поглинання можна видалити з формули всі імпліканти, що брали участь хоча б в одному склеюванні, з однаковим результатом. Тому при виконанні операцій склеювання отримані імпліканти позначаються. Після виконання всіх можливих склеювань позначені імпліканти видаляються. Це дає той же результат, що і при виконанні операцій поглинання.

Формальний метод знаходження множини всіх тупикових ДНФ функції запропонував С. Петрик. У цьому методі кожній із простих імплікант функції, що не входять до диз'юнктивного ядра, приписується буквене позначення. Пошук простих імплікант проводиться у спрощеній імплікантній таблиці. За допомогою цієї таблиці записується логічна формула покриття конституент одиниці простими імплікантами. Для цього для кожної конституенти одиниці складається диз'юнкція літерних позначень всіх простих імплікант, які її покривають. Далі записується формула, що зображує кон'юнкцію отриманих елементарних диз'юнкцій. У формулі розкриваються всі дужки, після чого вона набуває вигляд у диз'юнкції елементарних кон'юнкцій. Кожна елементарна кон'юнкція в отриманій формулі відповідає одній з тупикових ДНФ функції.

Для побудови такої ДНФ необхідно записати імпліканти, що відповідають літерним позначенням, які входять до складу даної елементарної кон'юнкції, й імпліканти диз'юнктивного ядра функції.

Приклад 5.2. Знайти за допомогою методу Квайна–Мак-Класкі мінімальну ДНФ функції $f(x_1, x_2, x_3, x_4)$, що задана ДДНФ

$$f(x_1, x_2, x_3, x_4) = \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 + \bar{x}_1\bar{x}_2\bar{x}_3x_4 + \bar{x}_1\bar{x}_2x_3\bar{x}_4 + \bar{x}_1\bar{x}_2x_3x_4 + \bar{x}_1x_2\bar{x}_3x_4 +$$

$$+ \bar{x}_1x_2x_3x_4 + x_1\bar{x}_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2x_3\bar{x}_4 + x_1x_2\bar{x}_3\bar{x}_4 + x_1\bar{x}_2x_3x_4 + x_1x_2\bar{x}_3x_4.$$

У таблиці 5.4 запишемо конституенти одиниці даної функції у вигляді двійкових кодів у другий стовпчик. У першому стовпці запишемо десяткові номери інтерпретацій.

Таблиця 5.4. Двійкові коди конституент одиниці функції $f(x_1, x_2, x_3, x_4)$

Десяткові номери інтерпретацій	Двійкові коди конституент одиниці	Конституенти одиниці функції $f(x_1, x_2, x_3, x_4)$
0	0000	$\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4$
1	0001	$\bar{x}_1\bar{x}_2\bar{x}_3x_4$
2	0010	$\bar{x}_1\bar{x}_2x_3\bar{x}_4$
3	0011	$\bar{x}_1\bar{x}_2x_3x_4$
5	0101	$\bar{x}_1x_2\bar{x}_3x_4$
7	0111	$\bar{x}_1x_2x_3x_4$
8	1000	$x_1\bar{x}_2\bar{x}_3\bar{x}_4$
10	1010	$x_1\bar{x}_2x_3\bar{x}_4$
11	1011	$x_1\bar{x}_2x_3x_4$
12	1100	$x_1x_2\bar{x}_3\bar{x}_4$
13	1101	$x_1x_2\bar{x}_3x_4$

Відповідно до методу здійснимо такі кроки:

1. Згрупуємо двійкові коди імплікант з однаковою кількістю одиниць. Назвемо число одиниць m індексом групи. Упорядкуємо групи у порядку зростання m.

2. Починаючи з $m = 2$, зробимо порівняння кожного двійкового коду в групі з індексом m з кожним кодом з групи з індексом $m+1$. Якщо порівнювані двійкові коди відрізняються тільки в одному розряді, то у наступний стовпчик таблиці запишемо відповідний їм двійковий код з порожньою позначкою «-» на місці зазначеного розряду. Навпроти кожного нового коду запишемо номери кодів двох імплікант, що брали участь у порівнянні, та у наступному стовпчику ці імпліканти позначимо знаком «V», оскільки вони не є простими імплікантами. Всі коди, які залишилися непозначеними знаком «V», відповідають простим імплікантам, тому позначимо їх знаком «X».

3. Якщо серед знов отриманих імплікант є однакові, то з них для подальшого використання залишаємо тільки одну.

4. Повторюємо кроки 1-3 доти, доки існує можливість отримувати нові коди імплікант.

Результат виконання описаних кроків наведено у таблиці 5.5. Спочатку заповнимо три стовпці нульового циклу: m (десятковий індекс групи), двійковий код імпліканти, номер імпліканти. Імпліканти поділяємо на групи за значенням m.

Потім здійснимо порівняння першої імпліканти з групи $m=0$ (імпліканта 0000) з першою імплікантою з групи $m=1$ (імпліканта 0001). Вони відрізняються тільки в одному розряді, тому робимо їх склеювання та отримуємо нову імпліканту з кодом 000-. Записуємо даний код у стовпчик коду імпліканти циклу 1, а навпроти імплікант 0000 і 0001 ставимо позначки «V», оскільки вони не є простими. Їх номери вказуємо поряд з кодом 000-. Потім порівнюємо імпліканту 0000 *з* другою імплікантою з групи $m=1$ – імплікантою 0010, тощо.

Таблиця 5.5. Отримання простих імплікант функції $f(x_1, x_2, x_3, x_4)$ методом Квайна–Мак-Класкі

Цикл 0				Цикл 1				Цикл 2			
m	Код імпл.	№ імпл.	Вид імпл.	m	Код імпл.	№ імпл.	Вид імпл.	m	Код імпл.	№ імпл.	Вид імпл.
0	0000	0	V	0	000-	0, 1	V	0	00--	(0, 1), (2, 3)	X
1	0001	1	V		00-0	0, 2	V		00--	(0, 2), (1, 3)	
	0010	2	V		-000	0, 8	V		-0-0	(0, 2), (8, 10)	X
	1000	8	V	1					-0-0	(0, 8), (2, 10)	
2	0011	3	V		00-1	1, 3	V	1	0--1	(1, 3), (5, 7)	X
	0101	5	V		0-01	1, 5	V		0--1	(1, 5), (3, 7)	
	1010	10	V		001-	2, 3	V		-01-	(2, 3), (10, 11)	X
	1100	12	V		-010	2, 10	V		-01-	(2, 10), (3, 11)	
3	0111	7	V		10-0	8, 10	V				
	1011	11	V		1-00	8, 12	X				
	1101	13	V	2	0-11	3, 7	V				
					-011	3, 11	V				
					01-1	5, 7	V				
					-101	5, 13	X				
					101-	10, 11	V				
					110-	12,13	X				

Коли порівняння всіх кодів імплікант стовпця «цикл 0» завершено, аналізуємо стовпчик «вид імпліканти». Навпроти всіх імплікант ставимо позначення «V», оскільки в циклі 0 кожна імпліканта припускає склеювання та не є простою. Далі розділяємо на групи $m=0$, $m=1$ та $m=2$ коди імплікант циклу 1. Потім попарно порівнюємо ці імпліканти й отримуємо стовпчик коду імплікант «цикл 2». Коди, що містять знаки «-», можуть утворювати нові імпліканти, якщо вони містять знаки «-» в одних і тих же розрядах. Закінчивши порівняння, виділяємо прості імпліканти у стовпці «вид» «цикл 1», які не мають

позначки «V», і позначаємо їх символом «X». У стовпці «код» «цикл 2» є однакові імпліканти, тому довільно викреслюємо один з однакових рядків, щоб кожна імпліканта зустрічалася тільки один раз. Порівнюючи коди імплікант у «цикл 2», доходимо висновку, що методом склеювання з них неможливо отримати нові імпліканти. Всі коди циклу 2 відповідають *простим імплікантам,* отже побудову таблиці завершено.

Для знаходження тупикових ДНФ побудуємо імплікантну таблицю, в рядках якої розташовуємо двійкові коди, що відповідають простим імплікантам, а в стовпцях – коди, що відповідають конституентам одиниці. Якщо двійковий код рядка є частиною коду стовпця (позиції зі знаком «-» не порівнюються), то у відповідну клітинку таблиці записуємо знак «∗» (таблиця 5.6).

Таблиця 5.6. Імплікантна таблиця функції $f(x_1, x_2, x_3, x_4)$

	0000	0001	0010	0011	0101	0111	1000	1010	1011	1100	1101
1-00							∗			∗	
-101					∗						∗
110-										∗	∗
00--	∗	∗	∗	∗							
-0-0	∗		∗				∗	∗			
0--1		∗		∗	∗	∗					
-01-			∗	∗				∗	∗		

Відзначимо стовпці таблиці, які містять по одному знаку «∗». Відповідні їм прості імпліканти є *диз'юнктивними ядрами.* Викреслюємо рядки таблиці, що відповідають ядрам, та стовпці, покриті ядрами, як показано у таблиці 5.7, та отримаємо спрощену імплікантну таблицю (таблиця 5.8).

Таблиця 5.7. Отримання спрощеної імплікантної таблиці

	0000	0001	0010	0011	0101	0111	1000	1010	1011	1100	1101
1-00							∗			∗	
-101					∗						∗
110-										∗	∗
00--	∗	∗	∗	∗							
-0-0	∗		∗				∗	∗			
0--1		∗		∗	∗	∗					
-01-			∗	∗				∗	∗		

Тепер знайдемо всі тупикові ДНФ функції $f(x_1, x_2, x_3, x_4)$ та оберемо з них мінімальну. Згідно з методом Петрика кожній з імплікант таблиці приписуємо літерне позначення та записуємо формулу покриття конституент одиниці простими імплікантами. Конституента одиниці з кодом 0000 може бути покрита імплікантою D або імплікантою E, тобто диз'юнкції: $D+E$. Конституента одиниці з кодом 1000 може бути покрита імплікантою A або

імплікантою E: $A+E$. Аналогічно 1100 може бути покрита $A+C$, а 1101 – диз'юнкцією $B+C$. Таким чином, покриття конституент одиниці спрощеної імплікантної таблиці функції $f(x_1, x_2, x_3, x_4)$ простими імплікантами можна записати у вигляді формули покриття $(D+E)(A+E)(A+C)(B+C)$.

Таблиця 5.8. Спрощена імплікантна таблиця

		0000	1000	1100	1101
A	1-00		*	*	
B	-101				*
C	110-			*	*
D	00--	*			
E	-0-0	*	*		

Якщо в цій формулі розкрити дужки, то отримаємо символьне зображення всіх можливих наборів простих імплікант для тупикових ДНФ, що не включає диз'юнктивні ядра,

$$(D+E)(A+E)(A+C)(B+C) = (DA+E)(AB+C) =$$
$$= DAB + EAB + DAC + EC.$$

В отриманій формулі кожна кон'юнкція буквених позначень відповідає набору імплікант у деякій тупиковій ДНФ, до якої обов'язково входять також диз'юнктивні ядра. Для того, щоб отримати мінімальну ДНФ, необхідно вибрати набір з мінімальною кількістю імплікант (у даному прикладі це EC) та додати імпліканти ядра (0--1, -01-). Отримуємо мінімальну ДНФ вихідної функції

$$f(x_1, x_2, x_3, x_4) = \bar{x}_2\bar{x}_4 + x_1x_2\bar{x}_3 + \bar{x}_1x_4 + \bar{x}_2x_3.$$

Таким чином, за допомогою методу Квайна–Мак-Класкі отримано мінімальну ДНФ функцію $f(x_1, x_2, x_3, x_4)$, яка містить усього лише 4 елементарні кон'юнкції замість 11 конституент одиниці ДДНФ.

Контрольні запитання

1. Назвіть способи задавання булевих функцій. Охарактеризуйте кожен спосіб.

2. Назвіть основні аксіоми та закони булевої алгебри.

3. Дайте означення суперпозиції булевих функцій.

4. Який пріоритет визначений для операцій алгебри логіки? Для якої цілі служить пріоритет операцій?

5. Яким чином здійснюється перехід від формули до таблиці істинності функції?

6. Дайте означення таких понять: елементарна кон'юнкція, елементарна диз'юнкція, конституента одиниці, конституента нуля. Яким чином ці поняття пов'язані з диз'юнктивним та кон'юнктивним розкладом булевих функцій?

7. Назвіть властивості конституенти одиниці та конституенти нуля.

8.Сформулюйте означення понять нормальних та досконалих нормальних форм булевих функцій. Поясніть їх зв’язок з диз’юнктивним та кон’юнктивним розкладом булевих функцій.

9.Опишіть алгоритми переходу від таблиці істинності булевої функції до ДДНФ та ДКНФ.

10. Поясніть алгоритм переходу від ДДНФ булевої функції до таблиці істинності.

11. Опишіть алгоритм переходу від ДКНФ булевої функції до таблиці істинності.

12. Назвіть, у чому полягає задача мінімізації булевих функцій. У чому особливість її канонічної форми?

13. Дайте означення поняттю імпліканти булевої функції. Що зображує повна система імплікант? Яка імпліканта називається простою?

14. Дайте означення скороченої, тупикової та мінімальної диз’юнктивних нормальних форм.

15. Запишіть формули операцій диз’юнктивного склеювання та поглинання.

16. Запишіть формули операцій кон’юнктивного склеювання та поглинання.

17. Назвіть основні кроки алгоритму мінімізації Квайна.

18. Назвіть основні кроки алгоритму мінімізації Квайна–Мак-Класкі.

19. Сформулюйте суть та призначення алгоритму Петрика.

ЛЕКЦІЯ 6

МЕТОД НЕВИЗНАЧЕНИХ КОЕФІЦІЄНТІВ. МЕТОД БЛЕЙКА-ПОРЕЦЬКОГО. МЕТОД НЕЛЬСОНА.

6.1 Метод невизначених коефіцієнтів

Даний метод можна застосовувати для мінімізації функцій перемикання від будь-якої кількості аргументів. Розглянемо принцип методу на прикладі логічних функції від трьох змінних.

ДНФ довільної функцій перемикання від трьох змінних може бути записана у такому вигляді:

$$\begin{aligned} f(x_1,x_2,x_3) &= K_1^1 x_1 + K_1^0 \bar{x}_1 + K_2^1 x_2 + K_2^0 \bar{x}_2 + K_3^1 x_3 + K_3^0 \bar{x}_3 + \\ &+ K_{12}^{11} x_1 x_2 + K_{12}^{10} x_1 \bar{x}_2 + K_{12}^{01} \bar{x}_1 x_2 + K_{12}^{00} \bar{x}_1 \bar{x}_2 + K_{13}^{11} x_1 x_3 + K_{13}^{10} x_1 \bar{x}_3 + \\ &+ K_{13}^{01} \bar{x}_1 x_3 + K_{13}^{00} \bar{x}_1 \bar{x}_3 + K_{23}^{11} x_2 x_3 + K_{23}^{10} x_2 \bar{x}_3 + K_{23}^{01} \bar{x}_2 x_3 + K_{23}^{00} \bar{x}_2 \bar{x}_3 + \\ &+ K_{123}^{000} \bar{x}_1 \bar{x}_2 \bar{x}_3 + K_{123}^{011} \bar{x}_1 x_2 x_3 + K_{123}^{010} \bar{x}_1 x_2 \bar{x}_3 + K_{123}^{111} x_1 x_2 x_3. \end{aligned} \quad (6.1)$$

У даному випадку фіксуємо усі можливі кон'юнктивні терми. Коефіцієнти K з різними індексами називають *невизначеними.* Їх треба підібрати таким чином, щоб отримана ДНФ була мінімальною. Якщо ж набори функції перемикання задані будь-яким способом, то отримуємо систему з 2^n рівнянь, з яких можна знайти значення коефіцієнтів K. При цьому необхідно користуватися основними тотожностями булевої алгебри. Наприклад, якщо функція на даному наборі дорівнює нулеві, та й усі терми відповідного рівняння також дорівнюють нулеві разом із відповідними невизначеними коефіцієнтами.

Розглянувши усі набори, на яких функція дорівнює нулеві, отримуємо нульові коефіцієнти K, які вилучаємо з наборів, що дорівнюють одиниці. З решти коефіцієнтів на наборах, що дорівнюють одиниці, обираємо ті, які мають найнижчий ранг (довільна диз'юнкція дорівнює одиниці, коли хоча б один з її доданків дорівнює одиниці). Ця операція послідовно застосовується до усіх одиничних наборів. Таким чином отримаємо МДНФ.

Приклад 6.1. Мінімізувати ДДНФ, яка задана таблицею істинності (таблиця 6.1)

Таблиця 6.1. Таблиця істинності до прикладу 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	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Для усіх восьми наборів даної функції перемикання складаємо систему відповідних рівнянь з невизначеними коефіцієнтами:

$$\begin{cases} f(0,0,0) = K_1^0 + K_2^0 + K_3^0 + K_{12}^{00} + K_{13}^{00} + K_{23}^{00} + K_{123}^{000} = 0, \\ f(0,0,1) = K_1^0 + K_2^0 + K_3^1 + K_{12}^{00} + K_{13}^{01} + K_{23}^{01} + K_{123}^{001} = 1, \\ f(0,1,0) = K_1^0 + K_2^1 + K_3^0 + K_{12}^{01} + K_{13}^{00} + K_{23}^{10} + K_{123}^{010} = 0, \\ f(0,1,1) = K_1^0 + K_2^1 + K_3^1 + K_{12}^{01} + K_{13}^{01} + K_{23}^{11} + K_{123}^{011} = 1, \\ f(1,0,0) = K_1^1 + K_2^0 + K_3^0 + K_{12}^{10} + K_{13}^{10} + K_{23}^{00} + K_{123}^{100} = 0, \\ f(1,0,1) = K_1^1 + K_2^0 + K_3^1 + K_{12}^{10} + K_{13}^{11} + K_{23}^{01} + K_{123}^{101} = 1, \\ f(1,1,0) = K_1^1 + K_2^1 + K_3^0 + K_{12}^{11} + K_{13}^{10} + K_{23}^{10} + K_{123}^{110} = 0, \\ f(1,1,1) = K_1^1 + K_2^1 + K_3^1 + K_{12}^{11} + K_{13}^{11} + K_{23}^{11} + K_{123}^{111} = 1. \end{cases} \quad (6.2)$$

Із першого, третього, п’ятого та сьомого рівнянь системи (6.2), згідно з основними тотожностями булевої алгебри, отримаємо нульові коефіцієнти K, які вилучимо з наборів, на яких функція дорівнює одиниці:

$$K_1^0 + K_1^1 + K_2^0 + K_2^1 + K_3^0 + K_{12}^{00} + K_{12}^{01} + K_{12}^{10} + K_{12}^{11} + K_{13}^{00} + K_{13}^{10} + \\ + K_{23}^{00} + K_{23}^{10} + K_{123}^{000} + K_{123}^{010} + K_{123}^{100} + K_{123}^{110} = 0. \quad (6.3)$$

Після цього система рівнянь (10.2) набуде такого вигляду:

$$\begin{cases} K_3^1 + K_{13}^{01} + K_{23}^{01} + K_{123}^{001} = 1, \\ K_3^1 + K_{13}^{01} + K_{23}^{11} + K_{123}^{011} = 1, \\ K_3^1 + K_{13}^{11} + K_{23}^{01} + K_{123}^{101} = 1, \\ K_3^1 + K_{13}^{11} + K_{23}^{11} + K_{123}^{111} = 1. \end{cases} \quad (6.4)$$

У кожному рівнянні системи (6.4) обираємо кон'юнкцію найменшого рангу, а всі інші коефіцієнти прирівнюємо до нуля, тобто

$$K_{13}^{01} + K_{23}^{01} + K_{23}^{11} + K_{123}^{001} + K_{123}^{011} + K_{123}^{101} + K_{123}^{111} = 0.$$

Таким чином, система рівнянь (6.4) набуде вигляду:

$$\begin{cases} K_3^1 = 1, \\ K_3^1 = 1, \\ K_3^1 = 1, \\ K_3^1 = 1. \end{cases} \quad (6.5)$$

Виходячи з системи (6.5) складаємо МДНФ даної функції перемикання. Невизначений коефіцієнт K_3^1 відповідає простій імпліканті x_3, яка покриває чотири рівняння системи (6.5). Отже, МДНФ запишемо у такому вигляді:

$$f(x_1, x_2, x_3) = x_3.$$

Таким чином, можна відзначити, що метод невизначених коефіцієнтів дуже простий у використанні. Тут також для пошуку МДНФ серед тупикових ДНФ можна застосовувати імплікантну таблицю. Істотним недоліком даного методу є його громіздкість, яка зростає зі зростанням кількості аргументів.

6.2 Метод Блейка-Порецького

Всі розглянуті вище методи мінімізації функцій перемикання ґрунтувалися на ДДНФ. Але у деяких випадках можна проводити мінімізацію функцій перемикання, які задані безпосередньо у ДНФ. У цьому випадку використовується *метод Блейка-Порецького,* який ґрунтується на операції *узагальненого диз'юнктивного склеювання:*

$$Ax + B\bar{x} = Ax + B\bar{x} + AB. \qquad (6.6)$$

Принцип даного методу полягає в тому, що у довільній ДНФ виконуються усі можливі операції *узагальненого диз'юнктивного склеювання* та *елементарного поглинання,* унаслідок чого отримується СДНФ без гарантії одержання МДНФ.

Покажемо роботу методу на прикладі.

Приклад 6.2. Знайти скорочену ДНФ функції, що подана у вигляді довільної ДНФ:

$$f = x_1\bar{x}_2 + x_1x_2\bar{x}_3 + x_2x_3.$$

Перший та другий терми можуть склеюватися як за x_1, так і за x_2, але результат буде нульовий. Нетривіальне узагальнене диз'юнктивне склеювання можливе лише для першого та третього термів ДНФ. Виконавши цю операцію, отримаємо:

$$f = x_1\bar{x}_2 + \bar{x}_1x_2x_3 + x_2x_3 + x_1x_3.$$

У даному виразі нетривіальне узагальнене склеювання можливе для першого та другого, першого та третього, другого та четвертого термів. Але ці склеювання нових термів не дають. Тобто більше операцію узагальненого диз'юнктивного склеювання застосовувати не можна. Після операції елементарного поглинання отримаємо СДНФ:

$$f = x_1\bar{x}_2 + x_2x_3 + x_1x_3.$$

Приклад 6.3. Знайти методом Блейка-Порецького мінімальну ДНФ функції, що задана довільною ДНФ:

$$f(x_1, x_2, x_3, x_4) = \bar{x}_1x_2 + x_1\bar{x}_3x_4 + \bar{x}_1x_3x_4 + x_1x_3x_4.$$

Проводимо узагальнене склеювання. Другий та четвертий елемент заданої ДНФ допускають узагальнене склеювання за змінною x_3. В результаті склеювання отримаємо

$$f(x_1, x_2, x_3, x_4) = x_1\bar{x}_3x_4 + x_1x_3x_4 + x_2x_4.$$

Очевидно, що ніякі інші елементи заданої ДДНФ не допускають узагальнене склеювання за іншими змінними.

Виконавши останнє узагальнене склеювання, переходимо до ДНФ:

$$f(x_1, x_2, x_3, x_4) = \bar{x}_1x_2 + x_1\bar{x}_3x_4 + \bar{x}_1x_3x_4 + x_1x_3x_4 + x_2x_4.$$

Після виконання поглинань отримуємо

$$f(x_1, x_2, x_3, x_4) = \bar{x}_1x_2 + x_2x_4 + \bar{x}_1x_3x_4.$$

Як і у першому прикладі подальше використання операції узагальненого склеювання та поглинання не дають результату, мінімальну ДНФ знаходять за допомогою імплікантної матриці, як у методі Квайна.

Складемо імплікантну таблицю (таблиця 6.2). Зірочкою відзначимо кожну клітинку таблиці, для якої імпліканта з рядка є власною частиною конституенти із стовпця.

Таблиця 6.2. Імплікантна таблиця функції $f(x_1, x_2, x_3, x_4)$

	$\bar{x}_1 x_2$	$x_1 \bar{x}_3 x_4$	$\bar{x}_1 x_3 x_4$	$x_2 x_3 x_4$
$\bar{x}_1 x_2$	*			
$x_2 x_4$				*
$\bar{x}_1 x_3 x_4$			*	

Знайдемо диз'юнктивне ядро. В імплікантній таблиці по одному знаку «*» містять всі стовпці. Ці прості імпліканти складають диз'юнктивне ядро.

Тупикова ДНФ даної функції буде єдиною та включати всі диз'юнктивні ядра. Таким чином, мінімальна ДНФ функції буде дорівнювати тупиковій ДНФ:

$$f(x_1, x_2, x_3, x_4) = \bar{x}_1 x_2 + x_2 x_4 + \bar{x}_1 x_3 x_4 .$$

6.3 Метод Нельсона

Даний метод застосовується до КНФ довільної функції перемикання та ґрунтується на наступній теоремі.

Теорема 6.1. Якщо у довільній КНФ розкрити відповідно до розподільного закону усі дужки, виконати усі елементарні поглинання та звести подібні, то отримаємо СДНФ.

Метод Нельсона є єдиним методом, який відштовхується від КНФ. У ряді випадків це може суттєво спростити процес мінімізації.

Приклад 6.4. Мінімізувати функцію перемикання, що задана у КНФ:

$$f = (x_1 + \bar{x}_2)(\bar{x}_1 + x_3)(x_1 + x_2 + \bar{x}_3).$$

Спочатку розкриваємо дужки:

$$f = (x_1 x_3 + x_3 \bar{x}_2 + \bar{x}_1 \bar{x}_2)(x_1 + x_2 + \bar{x}_3) = x_1 x_3 + x_1 x_2 x_3 + \bar{x}_1 \bar{x}_2 \bar{x}_3 + x_1 \bar{x}_2 x_3 .$$

Після застосування операції елементарного поглинання та зведення подібних, знаходимо СДНФ без гарантії отримання мінімальної форми:

$$f = x_1 x_3 + \bar{x}_1 \bar{x}_2 \bar{x}_3 .$$

Контрольні запитання

1. Назвіть способи задавання булевих функцій. Охарактеризуйте кожен спосіб.

2. Назвіть основні аксіоми та закони булевої алгебри.

3. Дайте означення суперпозиції булевих функцій.

4.Який пріоритет визначений для операцій алгебри логіки? Для якої цілі служить пріоритет операцій?

5.Які формули називають еквівалентними?

6.Яким чином здійснюється перехід від формули до таблиці істинності функції?

7.Дайте означення таких понять: елементарна кон'юнкція, елементарна диз'юнкція, конституента одиниці, конституента нуля. Яким чином ці поняття пов'язані з диз'юнктивним та кон'юнктивним розкладом булевих функцій?

8.Назвіть властивості конституенти одиниці та конституенти нуля.

9.Сформулюйте означення понять нормальних та досконалих нормальних форм булевих функцій. Поясніть їх зв'язок з диз'юнктивним та кон'юнктивним розкладом булевих функцій.

10. Опишіть алгоритми переходу від таблиці істинності булевої функції до ДДНФ та ДКНФ.

11. Поясніть алгоритм переходу від ДДНФ булевої функції до таблиці істинності.

12. Опишіть алгоритм переходу від ДКНФ булевої функції до таблиці істинності.

13. Дайте порівняльну характеристику алгоритмів переходу від довільної формули булевої функції до ДДНФ та ДКНФ.

14. Назвіть, у чому полягає задача мінімізації булевих функцій. У чому особливість її канонічної форми?

15. Дайте означення поняттю імпліканти булевої функції. Що зображує повна система імплікант? Яка імпліканта називається простою?

16. Дайте означення скороченої, тупикової та мінімальної диз'юнктивних нормальних форм.

17. Запишіть формули операцій диз'юнктивного склеювання та поглинання.

18. Запишіть формули операцій кон'юнктивного склеювання та поглинання.

19. Сформулюйте суть та призначення алгоритму Петрика.

20. Сформулюйте суть методу невизначених коефіцієнтів.

21. Охарактеризуйте метод Блейка-Порецького.

22. Наведіть відмінності алгоритму Нельсона для ДНФ та КНФ.

ЛЕКЦІЯ 7

МІНІМІЗАЦІЯ КОН'ЮНКТИВНИХ НОРМАЛЬНИХ ФОРМ. МІНІМІЗАЦІЯ ЧАСТКОВО ВИЗНАЧЕНИХ ФУНКЦІЙ.

7.1 Мінімізація кон'юнктивних нормальних форм

Мінімізація КНФ виконується аналогічно розглянутим методам мінімізації ДНФ булевих функцій, тому зупинимося лише на основних положеннях такої мінімізації. Задачею мінімізації КНФ є визначення мінімальної КНФ. Ця задача розв'язується в два етапи – пошук скороченої КНФ та знаходження мінімальної КНФ.

Перший етап мінімізації полягає у пошуку всіх простих імпліцент. Практично всі методи мінімізації ДНФ мають свої аналоги для КНФ. Зокрема, метод Нельсона у застосуванні до задачі мінімізації КНФ полягає у розкритті дужок в довільній ДНФ функції та виконанні операції кон'юнктивного поглинання. Такі дії приводять до появи скороченої КНФ.

Другий етап мінімізації виконується за допомогою таблиці Квайна (в цьому випадку вона називається *імпліцентною таблицею*) аналогічно, як й при пошуку мінімальної ДНФ. Оскільки можливо тільки два варіанти: або дана проста імпліцента поглинає дану конституенту нуля, або ні – згідно зі співвідношенням поглинання.

Приклад 7.1. Отримати скорочену КНФ функції, що задана в ДНФ, методом Нельсона:

$$f = x_1\bar{x}_2 + x_2\bar{x}_3.$$

Послідовно застосовуємо розподільний закон булевої алгебри та виконуємо операції поглинання:

$$f(x_1, x_2, x_3) = x_1\bar{x}_2 + x_2\bar{x}_3 = (x_1 + x_2)(x_1 + \bar{x}_3)(\bar{x}_2 + x_2)(\bar{x}_2 + \bar{x}_3) =$$

$$= (x_1 + x_2)(x_1 + \bar{x}_3)(\bar{x}_2 + \bar{x}_3).$$

Отже, скорочена ДКФ функції буде мати вигляд:

$$f(x_1, x_2, x_3) = (x_1 + x_2)(x_1 + \bar{x}_3)(\bar{x}_2 + \bar{x}_3).$$

Приклад 7.2. Методом Квайна знайти мінімальну КНФ функції.

$$f(x_1, x_2, x_3, x_4) = (\bar{x}_1 + x_2 + \bar{x}_3 + \bar{x}_4)(\bar{x}_1 + x_2 + \bar{x}_3 + x_4)(x_1 + \bar{x}_2 + \bar{x}_3 + x_4) \times$$

$$\times (x_1 + \bar{x}_2 + x_3 + x_4)(x_1 + x_2 + \bar{x}_3 + \bar{x}_4)(x_1 + x_2 + \bar{x}_3 + x_4).$$

Спочатку виконаємо всі можливі операції склеювання:

$$(\bar{x}_1 + x_2 + \bar{x}_3 + \bar{x}_4)(\bar{x}_1 + x_2 + \bar{x}_3 + x_4) = \bar{x}_1 + x_2 + \bar{x}_3,$$

$$(\bar{x}_1 + x_2 + \bar{x}_3 + \bar{x}_4)(x_1 + x_2 + \bar{x}_3 + \bar{x}_4) = x_2 + \bar{x}_3 + \bar{x}_4,$$

$$(\bar{x}_1 + x_2 + \bar{x}_3 + x_4)(x_1 + x_2 + \bar{x}_3 + x_4) = x_2 + \bar{x}_3 + x_4,$$

$$(x_1 + \bar{x}_2 + \bar{x}_3 + x_4)(x_1 + \bar{x}_2 + x_3 + x_4) = x_1 + \bar{x}_2 + x_4,$$

$$(x_1 + \bar{x}_2 + \bar{x}_3 + x_4)(x_1 + x_2 + \bar{x}_3 + x_4) = x_1 + \bar{x}_3 + x_4.$$

Отримаємо в результаті формулу:

$$f(x_1, x_2, x_3, x_4) = (\bar{x}_1 + x_2 + \bar{x}_3)(x_2 + \bar{x}_3 + \bar{x}_4)(x_2 + \bar{x}_3 + x_4) \times$$
$$\times (x_1 + \bar{x}_2 + x_4)(x_1 + \bar{x}_3 + x_4).$$

Попарно порівнюємо всі елементарні диз'юнкцій, що входять у дану формулу, та робимо висновок, що можна отримати інші елементарні диз'юнкції за допомогою операції склеювання в даній формулі:

$$(x_2 + \bar{x}_3 + \bar{x}_4)(x_2 + \bar{x}_3 + x_4) = x_2 + \bar{x}_3.$$

Таким чином, отримаємо функцію

$$f(x_1, x_2, x_3, x_4) = (\bar{x}_1 + x_2 + \bar{x}_3)(x_2 + \bar{x}_3)(x_1 + \bar{x}_2 + x_4)(x_1 + \bar{x}_3 + x_4).$$

Застосуємо до першої та другої диз'юнкції операцію поглинання та отримаємо остаточний вигляд функції, що відповідає скороченій КНФ даної функції:

$$f(x_1, x_2, x_3, x_4) = (x_2 + \bar{x}_3)(x_1 + \bar{x}_2 + x_4)(x_1 + \bar{x}_3 + x_4).$$

Складемо імпліцентну таблицю (таблиця 7.1). Її рядки задаються простими імпліцентами, а стовпці – конституантами нуля функції. Зірочкою відзначається кожна клітинка таблиці, для якої імпліцента з рядка є власною частиною конституенти зі стовпця.

Таблиця 7.1. Імпліцентна таблиця функції $f(x_1, x_2, x_3, x_4)$

	$\bar{x}_1 + x_2 + \bar{x}_3 + \bar{x}_4$	$\bar{x}_1 + x_2 + \bar{x}_3 + x_4$	$x_1 + \bar{x}_2 + \bar{x}_3 + x_4$	$x_1 + \bar{x}_2 + x_3 + x_4$	$x_1 + x_2 + \bar{x}_3 + \bar{x}_4$	$x_1 + x_2 + \bar{x}_3 + x_4$
$x_2 + \bar{x}_3$	*	*			*	*
$x_1 + \bar{x}_2 + x_4$			*	*		
$x_1 + \bar{x}_3 + x_4$			*			*

Знайдемо кон'юнктивне ядро. До нього входить кожна проста імпліцента, яка є єдиною у покритті будь-якої конституенти нуля. В імпліцентній таблиці по одному знаку «*» містять чотири стовпці, що відповідають конституентам нуля $\bar{x}_1 + x_2 + \bar{x}_3 + \bar{x}_4$, $\bar{x}_1 + x_2 + \bar{x}_3 + x_4$, $x_1 + \bar{x}_2 + x_3 + x_4$ та $x_1 + x_2 + \bar{x}_3 + \bar{x}_4$, навпроти простих імпліцент $x_2 + \bar{x}_3$ та $x_1 + \bar{x}_2 + x_4$. Ці прості імпліценти складають кон'юнктивне ядро. З імпліцентної таблиці викреслюємо рядки, що відповідають імпліцентам кон'юнктивного ядра, та стовпці, що відповідають конституентам нуля, які покриті імпліцентами ядра (таблиця 7.2).

Таблиця 7.2. Викреслювання рядків та стовпців імпліцентної таблиці

	$\bar{x}_1 + x_2 + \bar{x}_3 + \bar{x}_4$	$\bar{x}_1 + x_2 + \bar{x}_3 + x_4$	$x_1 + \bar{x}_2 + \bar{x}_3 + x_4$	$x_1 + \bar{x}_2 + x_3 + x_4$	$x_1 + x_2 + \bar{x}_3 + \bar{x}_4$	$x_1 + x_2 + \bar{x}_3 + x_4$
$x_2 + \bar{x}_3$	*	*			*	*
$x_1 + \bar{x}_2 + x_4$			*	*		
$x_1 + \bar{x}_3 + x_4$			*			*

У даному випадку імпліценти ядра покривають всі конституенти нуля. Таким чином, тупикова КНФ даної функції буде єдиною, а отже, й мінімальною КНФ, та включати тільки кон'юнктивні ядра:

$$f(x_1, x_2, x_3, x_4) = (x_2 + \bar{x}_3)(x_1 + \bar{x}_2 + x_4).$$

Графічний метод мінімізації за допомогою карт Карно або діаграм Вейча здійснюється аналогічно, як й у випадку ДНФ. Відмінність полягає тільки в тому, що аналізуються нульові набори та змінні виписуються з інверсіями.

Приклад 7.3. Знайти мінімальну КНФ логічної функції, що задана картою Карно (рисунок 7.1):

	$\bar{x}_3\bar{x}_4$	$\bar{x}_3 x_4$	$x_3 x_4$	$x_3\bar{x}_4$
$\bar{x}_1\bar{x}_2$	0	0	0	0
$\bar{x}_1 x_2$	1	1	0	1
$x_1 x_2$	1	1	1	1
$x_1\bar{x}_2$	0	0	1	1

Рисунок 7.1. Карта Карно до прикладу 7.3

Об'єднаємо дві групи із двох та одну з чотирьох нулів контурами так, як це показано на рисунку 7.3. Випишемо змінні, змінюючи інверсії змінних на протилежні. Результуюча мінімальна КНФ буде мати вигляд:

$$f(x_1, x_2, x_3, x_4) = (x_1 + x_2)(x_1 + \bar{x}_3 + \bar{x}_4)(\bar{x}_1 + x_2 + x_3).$$

7.2 Мінімізація частково визначених функцій

У реальних системах можливі випадки, коли не всі набори змінних можуть подаватися на входи комбінаційної схеми, тобто існують заборонені вхідні комбінації змінних або ці вхідні змінні ніяким чином не впливають на роботу цифрового автомата.

На таких наборах функція вважається *невизначеною*. У таблиці істинності значення функції на таких наборах позначаються символом «Х» або прочерком «-». Довизначення функції на заборонених наборах роблять таким чином, щоб забезпечити найефективнішу мінімізацію.

При використанні для мінімізації, наприклад, ДДНФ методом діаграм Вейча прочерки розглядають як одиниці в тих випадках, коли це призводить до збільшення розміру прямокутника, що відповідає імпліканті. У протилежному випадку вони розглядаються як нулі.

Приклад 7.4. Знайти МДНФ функції, що задана діаграмою Вейча (рисунок 7.2).

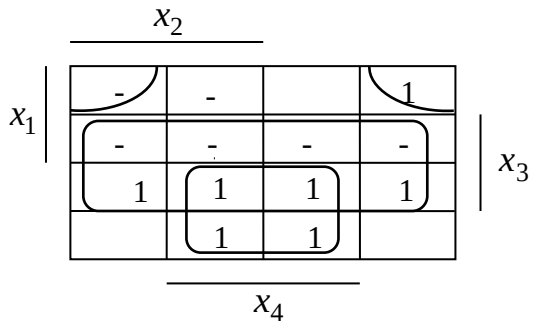


Рисунок 7.2. Діаграма Вейча

МДНФ функції має вигляд $f(x_1, x_2, x_3, x_4) = \bar{x}_1 x_4 + x_3 + x_1 \bar{x}_4$.

При використанні аналітичних методів мінімізації функцій у її ДДНФ вводять усі конституенти заборонених наборів, але в таблицю покриттів дані конституенти не включаються.

Контрольні запитання

1. Назвіть способи задавання булевих функцій. Охарактеризуйте кожен спосіб.
2. Назвіть основні аксіоми та закони булевої алгебри.
3. Яким чином здійснюється перехід від формули до таблиці істинності функції?
4. Дайте означення таких понять: елементарна кон'юнкція, елементарна диз'юнкція, конституента одиниці, конституента нуля. Яким чином ці поняття пов'язані з диз'юнктивним та кон'юнктивним розкладом булевих функцій?
5. Назвіть властивості конституенти одиниці та конституенти нуля.
6. Сформулюйте означення понять нормальних та досконалих нормальних форм булевих функцій. Поясніть їх зв'язок з диз'юнктивним та кон'юнктивним розкладом булевих функцій.
7. Опишіть алгоритми переходу від таблиці істинності булевої функції до ДДНФ та ДКНФ.
8. Поясніть алгоритм переходу від ДДНФ булевої функції до таблиці істинності.
9. Опишіть алгоритм переходу від ДКНФ булевої функції до таблиці істинності.
10. Назвіть, у чому полягає задача мінімізації булевих функцій. У чому особливість її канонічної форми?
11. Дайте означення поняттю імпліканти булевої функції. Що зображує повна система імплікант? Яка імпліканта називається простою?
12. Дайте означення скороченої, тупикової та мінімальної диз'юнктивних нормальних форм.
13. Запишіть формули операцій диз'юнктивного склеювання та поглинання.

14. Запишіть формули операцій кон’юнктивного склеювання та поглинання.

15. Сформулюйте правило записування мінімальної КНФ за методом карт Карно та діаграм Вейча.

16. Назвіть основні кроки алгоритму мінімізації ДКНФ методом Квайна.

17. Назвіть основні кроки алгоритму мінімізації ДКНФ методом Квайна–Мак-Класкі.

18. Сформулюйте суть та призначення алгоритму Петрика.

19. Сформулюйте суть методу невизначених коефіцієнтів для мінімізації ДКНФ.

20. Охарактеризуйте метод Блейка-Порецького при мінімізації ДКНФ.

21. Наведіть відмінності алгоритму Нельсона для ДНФ та КНФ.

22. Охарактеризуйте мінімізацію частково визначених функцій.

ЛЕКЦІЯ 8

КОМБІНАЦІЙНІ СХЕМИ, ЇХ ХАРАКТЕРИСТИКИ. ЛОГІЧНІ ЕЛЕМЕНТИ

8.1. Поняття комбінаційної схеми

Комбінаційною схемою (КС) називають схему, що є технічною реалізацією булевої функції або сукупності булевих функцій. У загальному випадку КС можна зобразити у вигляді «чорної скриньки» (див. рисунок 8.1), де $x_1, x_2, \ldots, x_m$ – логічні входи КС, а $y_1, y_2, \ldots, y_n$ – її логічні виходи.

Якщо КС реалізує одну булеву функцію, то її називають одновиходовою КС. Якщо КС реалізує сукупність булевих функцій – багатовиходовою КС.

Розглянемо комбінаційну схему S, що має m входів та n виходів (рисунок 8.1). На її входи подаються набори значень вхідних логічних змінних $x_i \in \{0,1\}, i = \overline{1,m}$, а на виходах формуються вихідні логічні змінні $y_j \in \{0,1\}, j = \overline{1,n}$.

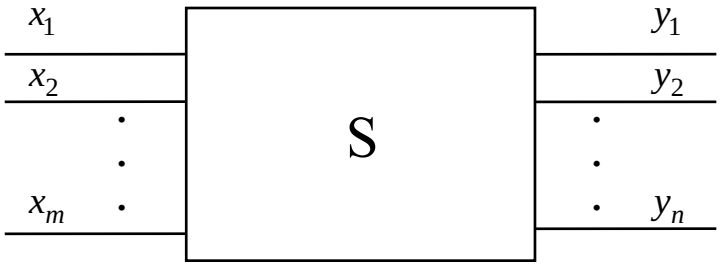


Рисунок 8.1. Модель комбінаційної схеми

КС описується за допомогою системи рівнянь

$$\begin{cases} y_1 = F_1(x_1, x_2, \ldots, x_m), \\ y_2 = F_2(x_1, x_2, \ldots, x_m), \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ y_n = F_n(x_1, x_2, \ldots, x_m), \end{cases}$$

де $\{F_i(x_1, x_2, \ldots, x_m), i = \overline{1,n}\}$ – множина булевих функцій.

Структурно КС може бути подана як система елементарних логічних схем – логічних елементів. Логічні елементи виконують над вхідними змінними елементарні логічні операції типу І, АБО, І-НІ, АБО-НІ тощо. Кількість входів логічного елемента відповідає кількості аргументів відтворюваної ним булевої функції. Графічне зображення КС, у якому показані зв'язки між її різними логічними елементами, які зображені умовними позначеннями, називають *функціональною схемою*.

У ході розроблення КС доводиться вирішувати задачі її аналізу та синтезу.

Задача аналізу КС полягає у визначенні її статичних та динамічних властивостей. У статиці визначаються булеві функції, які реалізуються КС за

відомою її структурою (функціональною схемою). У динаміці розглядається здатність надійного функціонування КС у перехідних процесах, які зумовлені зміною значень логічних змінних на її входах, а саме, досліджується наявність на виходах КС можливих небажаних імпульсних сигналів.

Задача синтезу полягає в побудові КС, що реалізує задану систему булевих функцій із заданого набору логічних елементів. Розв'язок задачі синтезу не є однозначним, оскільки можна запропонувати різні варіанти КС, що реалізують одну й ту ж систему булевих функцій, але відрізняються за рядом інших параметрів. Розробник КС з цієї множини варіантів вибирає один, виходячи з додаткових критеріїв: мінімальної кількості логічних елементів, необхідних для реалізації КС, максимальної швидкодії функціонування КС тощо.

8.2 Логічні елементи

Логічний елемент – це технічний пристрій, що реалізує одну з логічних функцій. У таблиці 8.1 зображені основні логічні елементи, що використовуються у цифрових пристроях.

Кількість входів у логічних елементах різного призначення може бути різною, але входи кожного елемента рівнозначні. Деякі з них при роботі в конкретних схемах можуть не використовуватися. Входи, які не використовуються в схемах І, І-НІ, з'єднують із напругою живлення, а в схемах АБО, АБО-НІ, суматора за модулем 2 – із 0В.

Основними параметрами логічних елементів є:

- напруги живлення та сигналів, які відповідають логічному 0 та логічній 1 (напруга живлення складає +5В, рівень логічної одиниці 2.4В – 5В, рівень логічного нуля – 0В – 0.4В);

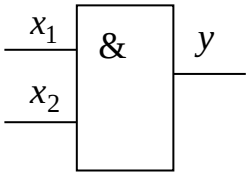
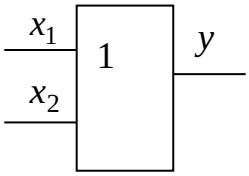
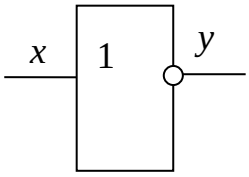
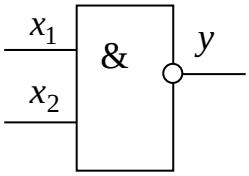
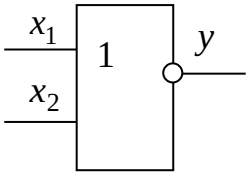
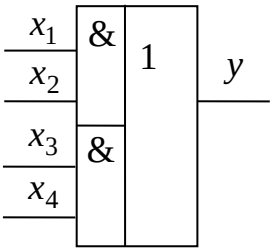
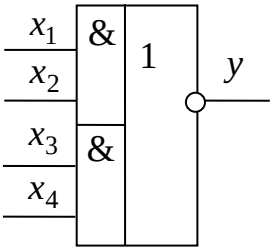
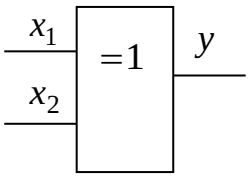
- коефіцієнти об'єднання за виходом $K_{об}$ (визначає максимально можливу кількість входів логічного елемента або, іншими словами, функцію скількох змінних може реалізувати цей елемент. Звичайно $K_{об}$ набуває значення від 2 до 4, рідше $K_{об}$=8. Збільшення кількості входів пов'язано з ускладненням схеми елементів та призводить до погіршення інших параметрів – завадостійкості, швидкодії й т.д.);

- коефіцієнт розгалуження за виходом $K_{роз}$ (показує на скільки логічних входів може бути одночасно навантажений вихід даного логічного елемента. Як правило, у типових логічних елементах $K_{роз}$=10. Іноді замість $K_{роз}$ задається граничне допустиме значення вихідного струму логічного елемента в стані 0 або 1;

- розсіювана потужність;

- завадостійкість – це здатність елемента правильно функціонувати за наявності завад. Вона визначається максимально допустимою напругою завади, за якої не відбувається збою роботи логічного елемента. Переважно ця напруга складає 0.6В – 0.9В;

Таблиця 8.1. Основні логічні елементи

Графічне позначення логічного елемента	Назва логічного елемента	Функція, що виконується логічним елементом
	Елемент І (кон’юнктор)	$y = x_1 x_2$
	Елемент АБО (диз’юнктор)	$y = x_1 + x_2$
	Елемент НІ (інвертор 1)	$y = \bar{x}$
	Елемент І-НІ	$y = \overline{x_1 x_2}$
	Елемент АБО-НІ	$y = \overline{x_1 + x_2}$
	Елемент І-АБО	$y = x_1 x_2 + x_3 x_4$
	Елемент І-АБО-НІ	$y = \overline{x_1 x_2 + x_3 x_4}$
	Суматор за модулем 2	$y = \bar{x}_1 x_2 + x_1 \bar{x}_2$

- швидкодія (характеризується часом затримки розповсюдження сигналу через логічний елемент. Цей параметр істотно залежить від технології виготовлення мікросхем та знаходиться в діапазоні від одиниць до сотень наносекунд).

8.3 Характеристики комбінаційних схем

До основних характеристик комбінаційних схем належать їх складність та швидкодія.

Складність комбінаційної схеми оцінюється кількістю логічних елементів, що входять до її складу. При розробленні КС на основі конкретної елементної бази кількість елементів, як правило, визначається числом корпусів (модулів) інтегральних мікросхем, що використовуються в схемі. У теоретичних розробках КС орієнтуються на довільну елементну базу й тому для оцінювання витрат устаткування використовується оцінка складності комбінаційних схем за Квайном.

Складність (ціна) за Квайном визначається сумарною кількістю входів логічних елементів у складі схеми. При такому оцінюванні одиниця складності – один вхід логічного елемента. Ціна інверсного входу, як правило, дорівнює двом. Такий підхід до оцінювання складності виправданий з таких причин:

- складність КС легко обчислюється за булевими функціями, на основі яких будується схема: для ДНФ складність схеми дорівнює сумі кількості символів (символу зі знаком заперечення відповідає ціна 2) та кількості знаків диз'юнкції, збільшеної на 1 для кожного диз'юнктивного логічного виразу;
- всі класичні методи мінімізації булевих функцій забезпечують мінімальність схеми саме в значенні ціни за Квайном.

Практика показує, що схема з мінімальною ціною за Квайном, як правило, реалізується якнайменшою кількістю конструктивних елементів – корпусів інтегральних мікросхем.

Швидкодія комбінаційної схеми оцінюється максимальною затримкою сигналу при проходженні його від входу схеми до виходу, тобто визначається проміжком часу від моменту надходження вхідних сигналів до моменту встановлення відповідних значень вихідних сигналів КС. Затримка сигналу кратна кількості елементів, через які проходить сигнал від входу до виходу КС. Тому швидкодія схеми характеризується значенням $r\tau$, де τ – затримка сигналу на одному елементі, r – кількість рівнів (глибина) КС. Кількість рівнів КС розраховується таким чином:

- входам КС приписується нульовий рівень;
- логічні елементи, які пов'язані тільки з входами схеми, відносяться до першого рівня;
- елемент відноситься до рівня k, якщо він пов'язаний зі входами елементів рівнів k-1, k-2 й т.д.;
- максимальний рівень елементів r визначає кількість рівнів КС та називається *рангом схеми*.

Контрольні запитання

1. Дайте означення комбінаційної схеми.
2. Яка схема називається функціональною?
3. Дайте означення логічного елемента. Наведіть основні параметри логічних елементів.
4. Наведіть характеристики комбінаційних схем.
5. Дайте означення поняттю «ранг схеми». Наведіть приклад визначення рангу довільної комбінаційної схеми.
6. Обґрунтуйте задачі синтезу та аналізу комбінаційних схем.
7. Охарактеризуйте канонічний метод синтезу комбінаційних схем.
8. Обґрунтуйте синтез комбінаційних схем із врахуванням обмежень на коефіцієнт розгалуження за виходом.
9. Обґрунтуйте синтез комбінаційних схем із врахуванням обмежень на коефіцієнт об’єднання за входом.

ЛЕКЦІЯ 9

МЕТОДИ АНАЛІЗУ ТА СИНТЕЗУ КОМБІНАЦІЙНИХ СХЕМ

9.1 Канонічний метод синтезу комбінаційних схем

Алгоритм канонічного методу синтезу КС полягає в наступному:

1. Булева функція, яку повинна реалізовувати синтезована КС, записується у вигляді ДДНФ.

2. З використанням методів мінімізації визначається мінімальна ДНФ та мінімальна КНФ. З отриманих двох мінімальних форм вибирається простіша.

3. Булеву функцію в мінімальній формі згідно з п.2 подають у заданому (або вибраному розробником) базисі логічних елементів.

4. За поданням логічної функції в заданому базисі будують КС.

Необхідно зазначити, що булева функція $F(x_1, x_2, ..., x_m)$ може бути задана не на всіх можливих наборах аргументів $x_1, x_2, ..., x_m$. На тих наборах, де функція невизначена, її довизначають так, щоб у результаті мінімізації отримати простішу мінімальну ДНФ або мінімальну КНФ. При цьому спроститься й сама КС. Крім того, досить часто з метою отримання ще більш простого подання логічної функції у МДНФ, отримана в п.2 функція зображається в так званій дужковій формі, у якій виносяться за дужки загальні частини імплікант МДНФ.

9.2 Методи аналізу комбінаційних схем

Задачі аналізу КС виникають за необхідності перевірити правильність синтезу (на етапі проектування) або визначити булеву функцію, яку реалізує КС (при аналізі або ремонті схем). Усі існуючі методи аналізу поділяються на прямі та непрямі.

У результаті аналізу КС *прямим методом* виходить множина наборів вхідних змінних, які забезпечують задане значення на виході, що дає змогу записати у вигляді алгебри логічних функцій реалізовану схему. До прямих методів належить метод π - алгоритму.

Використання *непрямих методів* дає можливість визначити реакцію схеми на заданий набір вхідних змінних у статиці або проаналізувати перехідний процес зміни одного вхідного набору на інший. Прикладами непрямих методів аналізу є методи синхронного та асинхронного моделювання.

9.2.1 Аналіз комбінаційних схем методом π -алгоритму

Застосовуючи даний метод, шукають набори вхідних змінних, що забезпечують задане значення на виході КС. Набори, що забезпечують на виході КС логічну 1, утворюють *одиничне покриття* C^1. Аналогічно, вхідні набори, що забезпечують на виході КС логічний 0, утворюють *нульове покриття* C^0. Розглянемо покриття C^0 та C^1 для найпростішого логічного

елемента 2І (рисунок 9.1), що виконує функцію $y = x_1 x_2$. Таблиця істинності для цієї функції записана в таблиці 9.1.

Таблиця 9.1. Істинність функції $y = x_1 x_2$

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

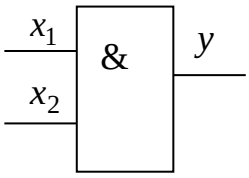


Рисунок 9.1. Логічний елемент 2І

Як бачимо з наведеної таблиці 9.1, тільки при єдиному наборі $x_1 = 1$ та $x_2 = 1$ на виході логічного елемента буде 1, тобто одиничне покриття включає тільки один набір $C^1 = \{11\}$. На виході логічного елемента буде 0 для трьох наборів, що утворюють нульове покриття:

$$C^0 = \begin{bmatrix} 00 \\ 01 \\ 10 \end{bmatrix}.$$

Це покриття можна спростити, якщо перший набір склеїти з другим та третім, тобто

$$C^0 = \begin{bmatrix} 0X \\ X0 \end{bmatrix}.$$

Для логічного елемента 2І можна сказати, що 1 на його виході буде тільки за наявності обох одиниць на входах, а для забезпечення 0 на виході достатньо подати хоча б на один вхід 0. Міркуючи аналогічно, отримаємо таблицю покриттів C^0 та C^1 для деяких логічних елементів (таблиця 9.2).

Аналізуючи схеми за методом π-алгоритму, задавшись певним значенням на виході, замінюють його відповідним покриттям елемента, що формує вихідний сигнал. У результаті цього визначаються, які повинні бути сигнали на виходах елементів, підключених до вихідного логічного елемента. У свою чергу, сигнали на виходах цих елементів можна замінити відповідними покриттями, тобто визначити значення вихідних сигналів для інших логічних елементів й т.д. Цей процес продовжується до тих пір, доки не отримуються покриття, що складаються тільки з вхідних змінних, що називаються *опорними змінними*. Сукупність таких покриттів дає відповідне покриття схеми.

9.2.2 Аналіз комбінаційних схем методом синхронного моделювання

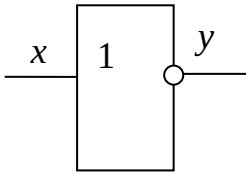
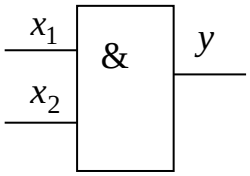
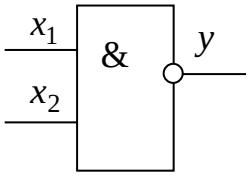
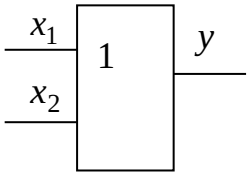
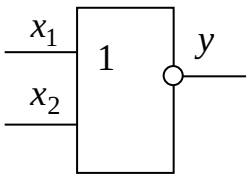
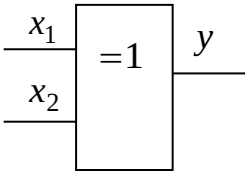
У даному методі вважається, що всі логічні елементи перемикаються одночасно, без затримки. У результаті використання методу визначається стале значення сигналу на виході схеми.

Приклад 9.1. Методом синхронного моделювання проаналізувати схему, зображену на рисунку 9.2.

Спочатку схема розбивається на рівні та записується в порядку зростання рівня рівнянь, що описують функціонування логічних елементів (таблиця 9.3).

Проаналізуємо схему при подаванні на її вхід набору $X = \{010101\}$. Для цього розв'яжемо записані в таблиці 9.3 рівняння $y_1 = 0 \cdot 1 \cdot 0 = 0$, $y_2 = \overline{0 \cdot 1} = 1$, $y_3 = 0 + 1 + 0 = 1$, $Y = y_4 = 1 \cdot 1 = 1$.

Таблиця 9.2. Покриття для деяких логічних елементів

Графічне позначення логічного елемента	Назва логічного елемента	Оператор підстановки	C^0	C^1
	НІ	$П_1$	1	0
	2І	$П_2$	0 X X 0	1 1
	2І-НІ	$П_3$	1 1	0 X X 0
	2АБО	$П_4$	0 0	1 X X 1
	2АБО-НІ	$П_5$	1 X X 1	0 0
	суматор за модулем 2	$П_6$	0 0 1 1	0 1 1 0

Таблиця 9.3. До прикладу 9.1

№ рівня	№ логічного елемента	Функція, що реалізує логічний елемент
1	1 2	$y_1 = x_1 x_2 x_3$ $y_2 = \overline{x_3 x_6}$
2	3	$y_3 = y_1 + x_4 + x_5$
3	4	$Y = y_4 = y_3 y_2$

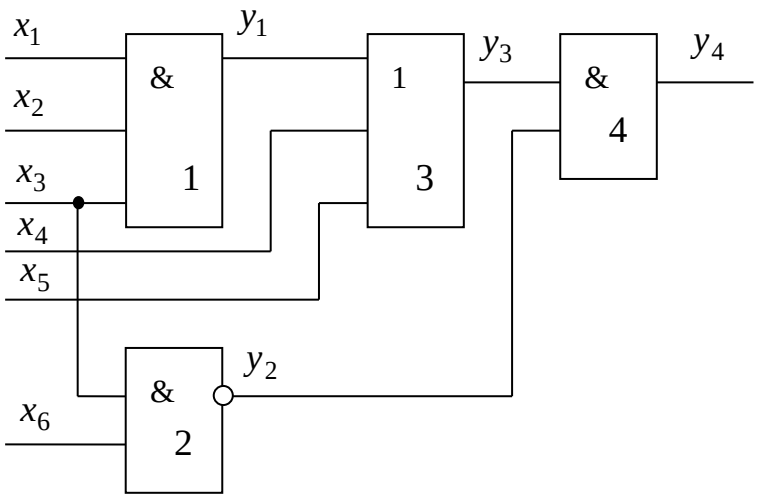


Рисунок 9.2. КС для аналізу методом синхронного моделювання

Проаналізуємо схему при подаванні на її вхід набору $X = \{010101\}$. Для цього розв'яжемо записані в таблиці 9.3 рівняння

$$y_1 = 0 \cdot 1 \cdot 0 = 0,$$
$$y_2 = \overline{0 \cdot 1} = 1,$$
$$y_3 = 0 + 1 + 0 = 1,$$
$$Y = y_4 = 1 \cdot 1 = 1.$$

Отже, при подаванні на вхід набору $X = \{010101\}$, на виході буде $Y = 1$. Аналогічно можна змоделювати роботу КС при поданні на її вхід будь-якого іншого набору.

9.2.3 Аналіз комбінаційних схем методом асинхронного моделювання

Реальний логічний елемент перемикається за деякий скінченний час, що залежить від технології його виготовлення, умов експлуатації, ємностей навантаження й т.д. Проходження сигналу послідовно через кілька логічних елементів призводить до накопичення часу затримки та виникнення зсуву в часі вихідного сигналу відносно вхідного. Наявність затримки та породжуваного

нею часового зсуву сигналів може призвести до появи на виході окремих логічних елементів та всієї схеми в цілому короткотермінових сигналів, які не передбачені булевою функцією, на основі якої реалізована КС.

Таке явище називається *ризиком збою*. Розрізняють статистичний та динамічний ризики збою.

При *статичному ризику* збою до перехідного процесу та після нього стан вихідного сигналу один і той самий, а під час перехідного процесу можлива короткочасна поява протилежного сигналу.

При *динамічному ризику* збою до та після перехідного процесу стани вихідного сигналу протилежні, але в перехідному процесі вихідний сигнал кілька разів змінює своє значення.

Для аналізу процесу перемикання КС при зміні вхідних наборів та виявлення ризиків збою використовується метод *асинхронного моделювання*. У цьому методі вважається, що кожний елемент перемикається з однаковою затримкою. Алгоритм аналізу полягає в наступному.

1. Кожному елементу схеми ставиться у відповідність певний рівень, причому рівень 1 мають елементи, всі входи яких є незалежними входами комбінаційної схеми.

2. Записуються рівняння, що описують кожен логічний елемент у порядку спадання рівня.

3. Для початкового вхідного набору $A(x_1 x_2 \ldots x_n)$ визначаються значення сигналів на виходах усіх логічних елементів схеми. Нехай даний набір A замінюється набором $B(x_1 x_2 \ldots x_n)$.

4. Позначаються ті рівняння, в правій частині яких хоча б одна зі змінних змінила своє значення.

5. Розв’язуються позначені рівняння в порядку їх запису в системі. Після розв’язання рівняння вважається непоміченим.

6. Якщо після розв’язання всіх рівнянь системи змінні, що входять у ліві частини рівнянь, змінили свої значення, то знову позначаються ті рівняння, в правій частини яких входять ці змінні. Потім здійснюється перехід до п.5. У іншому разі, моделювання даного вхідного набору вважається закінченим. Виконання п.5 називається *тактом моделювання.*

Контрольні запитання

1. Дайте означення комбінаційної схеми.
2. Яка схема називається функціональною?
3. Дайте означення логічного елемента. Наведіть основні параметри логічних елементів.
4. Наведіть характеристики комбінаційних схем.
5. Дайте означення поняттю «ранг схеми». Наведіть приклад визначення рангу довільної комбінаційної схеми.
6. Обґрунтуйте задачі синтезу та аналізу комбінаційних схем.
7. Охарактеризуйте канонічний метод синтезу комбінаційних схем.

8. Обґрунтуйте синтез комбінаційних схем із врахуванням обмежень на коефіцієнт розгалуження за виходом.

9. Обґрунтуйте синтез комбінаційних схем із врахуванням обмежень на коефіцієнт об’єднання за входом.

10. Охарактеризуйте принцип аналізу комбінаційних схем методом π − алгоритму.

11. Наведіть принцип аналізу комбінаційних схем методом синхронного моделювання.

12. Обґрунтуйте принцип аналізу комбінаційних схем методом асинхронного моделювання. Дайте означення ризику збою та його різновидів.

ЛЕКЦІЯ 10

ТИПОВІ КОМБІНАЦІЙНІ СХЕМИ: СУМАТОРИ, ШИФРАТОРИ, ДЕШИФРАТОРИ

10.1 Суматори

Суматором називається комбінаційна схема, яка виконує арифметичне (на противагу логічному) додавання та віднімання чисел.

Арифметичні суматори є складовими частинами арифметико-логічних пристроїв (АЛП) мікропроцесорів, де вони використовуються як для додавання двійкових чисел, так і для формування фізичної адреси комірки пам'яті.

Суматори класифікують за різними ознаками. Залежно від системи числення розрізняють: двійкові, двійково-десяткові, десяткові, інші (наприклад, амплітудні) суматори. За кількістю розрядів доданків, що обробляються одночасно, суматори поділяються на: однорозрядні та багаторозрядні.

На практиці використовуються однорозрядні суматори наступних типів: чвертьсуматори, напівсуматори та повні суматори.

Чвертьсуматори (елементи «сума за модулем 2») характеризуються наявністю двох входів, на які подаються два однорозрядні числа, та одним виходом, на якому реалізується їх арифметична сума.

Двійковий напівсуматор є найпростішою комбінаційною схемою та виконує додавання двох однорозрядних двійкових чисел. Він має два входи для доданків: A і B та два виходи: суми S та переносу P. Таблицю функціонування двійкового напівсуматора наведено в таблиці 10.1.

Таблиця 10.1

A	B	S	P	Примітка
0	0	0	0	0+0=0
0	1	1	0	0+1=1
1	0	1	0	1+0=1
1	1	0	1	1+1=0(P=1)

З таблиці 10.1 запишемо булеві функції для S та P:

$$S = \overline{A}B + A\overline{B} = A \oplus B,$$

$$P = AB.$$

Логічну схему та умовне позначення напівсуматора наведено на рисунку 10.1. *Повний двійковий суматор* виконує додавання трьох однорозрядних двійкових чисел. Він має три входи: для двох доданків A, B та перенесення з попереднього молодшого розряду p та два виходи: суми S та переносу в наступний старший розряд P. Таблицю функціонування повного двійкового суматора наведено в таблиці 10.2.

Проведемо мінімізацію функцій S та P за допомогою карт Карно (рисунки 10.2 та 10.2).

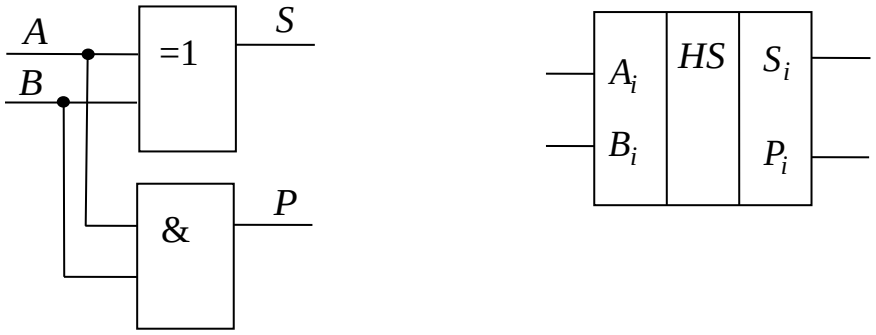


Рисунок 10.1. Схема та умовне позначення напівсуматора

Таблиця 10.2. Таблиця функціонування повного двійкового суматора

A	B	p	S	P
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

	$\overline{p}$	p
$\overline{A}\overline{B}$	0	1
$\overline{A}B$	1	0
AB	0	1
$A\overline{B}$	1	0

Рисунок 10.2. Карта Карно для S

	$\overline{p}$	p
$\overline{A}\overline{B}$	0	0
$\overline{A}B$	0	1
AB	1	1
$A\overline{B}$	0	1

Рисунок 10.3. Карта Карно для P

Як видно з рисунку 10.2, функція S не мінімізується, тобто

$$S = \overline{A}\overline{B}p + \overline{A}B\overline{p} + ABp + A\overline{B}\overline{p}. \quad (10.1)$$

Функція P може бути змінімізована, що дає формулу

$$P = Ap + Bp + AB. \quad (10.2)$$

Застосування законів подвійної інверсії та де Моргана до виразів (10.1) - (10.2) приводить до канонічних рівнянь схеми повного суматора:

$$S = \overline{\overline{A}\overline{B}p \cdot \overline{A}B\overline{p} \cdot \overline{ABp} \cdot \overline{A\overline{B}\overline{p}}},$$

$$P = \overline{\overline{Ap} \cdot \overline{Bp} \cdot \overline{AB}},$$

за якими далі будуємо канонічну схему однорозрядного повного суматора (рисунок 10.4).

Розглянемо інший метод синтезу суматора, який використовується на практиці.

З аналізу таблиці 10.2 випливає, що $S = 1$, якщо кількість одиниць у вхідному наборі непарне, тобто

$$S = A \oplus B \oplus p. \quad (10.3)$$

З таблиці 10.2 випишемо ДДНФ функції P та перетворимо її таким чином:

$$P = \overline{A}Bp + A\overline{B}p + AB\overline{p} + ABp = p\left(\overline{A}B + A\overline{B}\right) + AB = p\left(A \oplus B\right) + AB. \quad (10.4)$$

Отже, схема однорозрядного повного суматора, що побудована на основі виразів (10.3) та (10.4) буде складатися з двох напівсуматорів (рисунок 10.4).

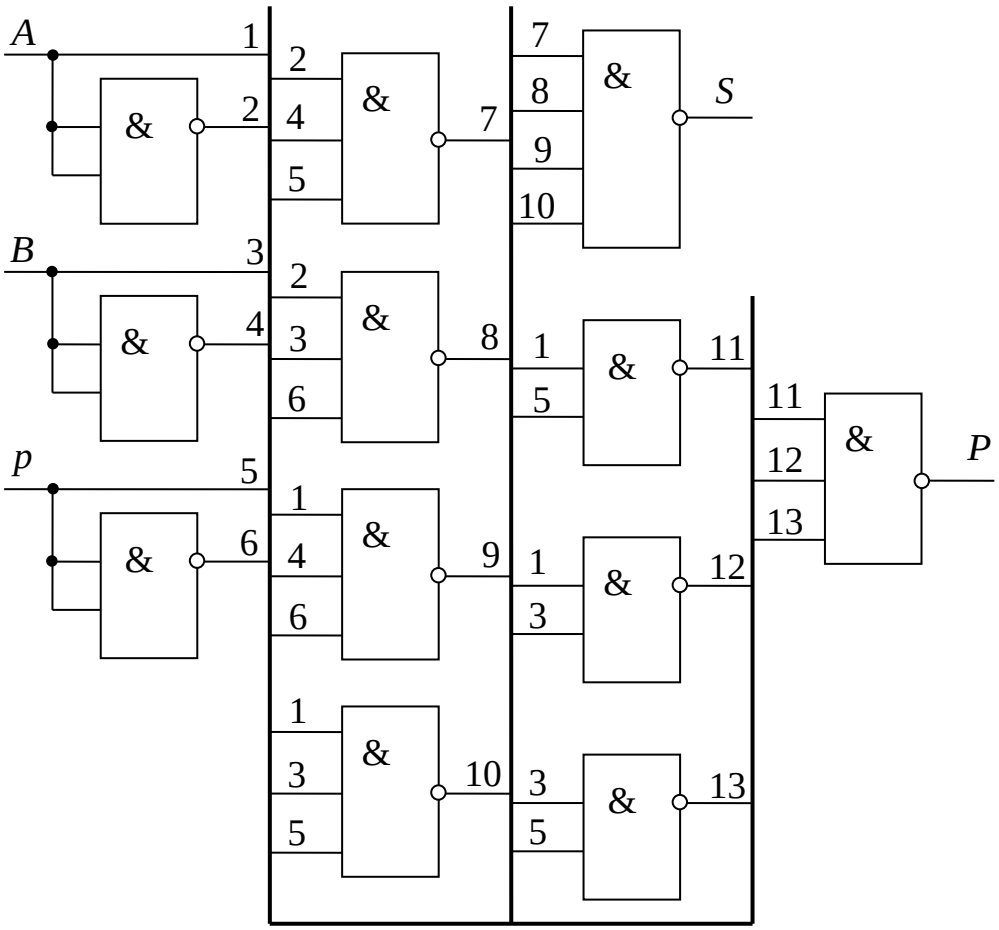


Рисунок 10.3. Канонічна схема однорозрядного повного суматора

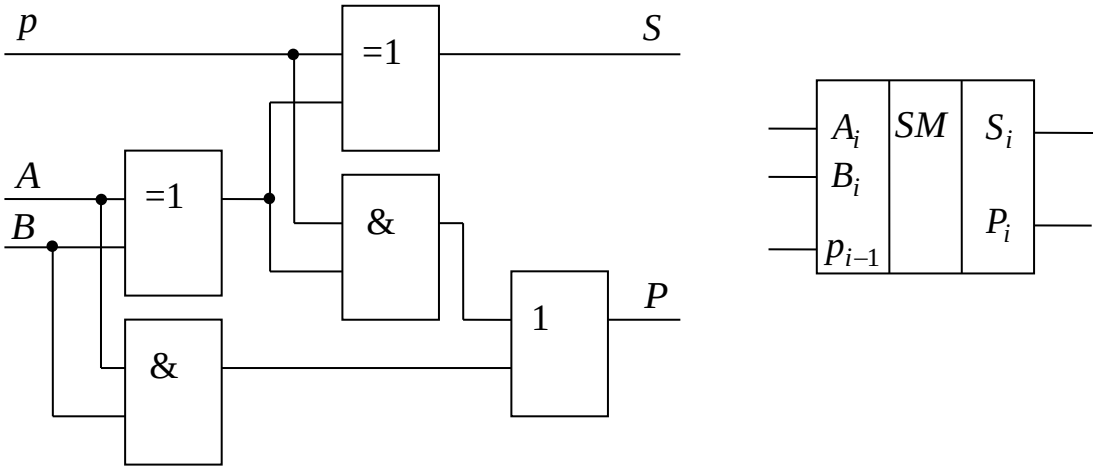


Рисунок 10.4. Схема однорозрядного повного суматора та його умовне позначення

Багаторозрядні суматори становлять собою комбінацію однорозрядних суматорів з трьома входами кожен. Кількість входів та виходів суматора визначається розрядністю доданків. За організацією переносу розрізняють суматори з послідовним переносом та паралельним переносом.

10.2 Дешифратори та шифратори

Дешифратор (декодер) – це типова логічна комбінаційна схема з n інформаційними входами та 2^n виходами. Тобто це схема призначена для реалізації конституент одиниці.

Розрізняють *повні* та *неповні* дешифратори. Повні дешифратори реалізують 2^n конституент, де n – це кількість інформаційних входів. Неповні дешифратори реалізують менше ніж 2^n конституент.

Функціонування повного дешифратора описується системою логічних виразів вигляду:

$$F_0 = \overline{X}_{n-1}\overline{X}_{n-2}\ldots\overline{X}_1\overline{X}_0;$$
$$F_1 = \overline{X}_{n-1}\overline{X}_{n-2}\ldots\overline{X}_1 X_0;$$
$$\ldots$$
$$F_{2^n-1} = X_{n-1}X_{n-2}\ldots X_1 X_0,$$

де $X_{n-1}X_{n-2}\ldots X_0$ – вхідні двійкові змінні; F_0, F_1, ..., F_{2^n-1} – вихідні логічні функції, що є мінтермами. Якщо дешифратор неповний, то кількість виходів m менше ніж 2^n.

Таблиця істинності функцій повного дешифратора для $n=3$ має вигляд згідно таблиці 10.3.

Таблиця 10.3. Таблиця істинності повного дешифратора з трьома інформаційними входами

X_2	X_1	X_0	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Індекс функції F_i визначає номер обраного виходу та відповідає десятковому еквіваленту вхідного коду. Вихід, на якому з'являється керуючий сигнал, називають *активним*. Якщо значення сигналу на активному виході відображається логічною 1 (H), то на решті пасивних виходів встановлюється логічний 0 (L). Двійковий код, який завжди містить тільки одну одиницю,

решта – нулі, називається *унітарним*. Тому дешифратори є перетворювачами вхідного позиційного коду в унітарний вихідний. Зауважимо, що крім позначень H та L для логічних входів та виходів можуть вживатись позначення X (байдужий – 0 чи 1) та Z, що відповідає буферу виходу з Z станом.

Комбінаційну схему дешифратора для $n = 3$ показано на рисунку 10.5.

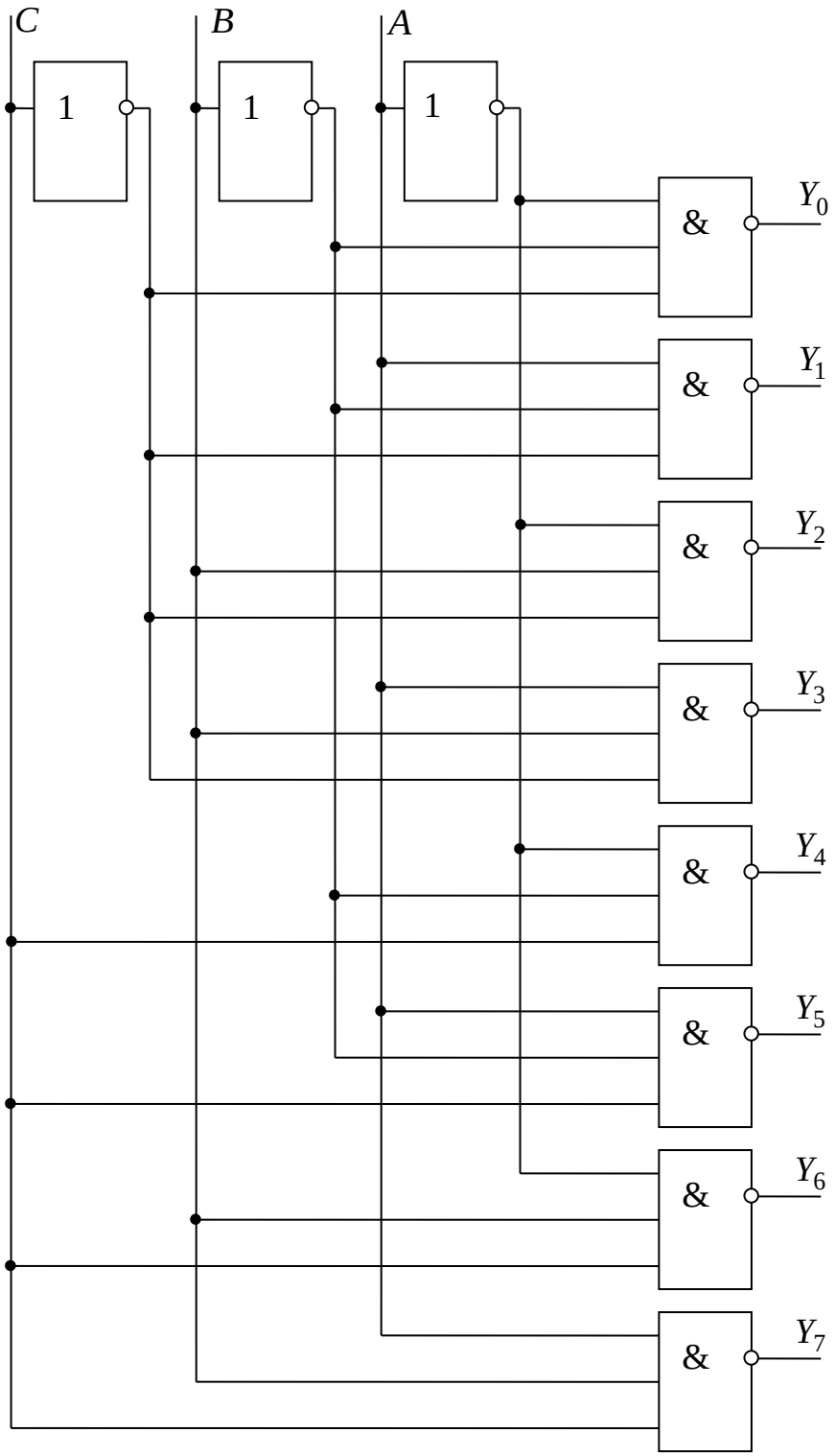


Рисунок 10.5. Комбінаційна схема дешифратора

Схема має 8 виходів, на одному з яких формується низький потенціал (логічний 0), на інших – високий (логічна 1). Номер цього єдиного виходу, на якому формується активний (нульовий) рівень, відповідає числу *N*, яке визначається станом входів *C, B, A* згідно формули $N = C \cdot 2^2 + B \cdot 2^1 + A \cdot 2^0$. Наприклад, якщо на вході подано комбінацію логічних рівнів 101, то із восьми виходів схеми (Y0, Y1, …, Y7) тільки на виході з номером *N*=5 встановиться нульовий рівень сигналу (Y5=0), а всі інші виходи будуть мати рівень логічної одиниці. Цей принцип формування вихідного сигналу можна описати формулою:

$$Y_i = \begin{cases} 0, & якщо \quad i = k; \\ 1, & якщо \quad i \neq k, \end{cases}$$

де $k = C \cdot 2^2 + B \cdot 2^1 + A \cdot 2^0$.

Для цього випадку вихідні сигнали для виходів дешифратора формуються згідно логічних виразів:

$$Y0 = \overline{\overline{C} \cdot \overline{B} \cdot \overline{A}},\ Y1 = \overline{\overline{C} \cdot \overline{B} \cdot A},\ Y2 = \overline{\overline{C} \cdot B \cdot \overline{A}},\ Y3 = \overline{\overline{C} \cdot B \cdot A},$$
$$Y4 = \overline{C \cdot \overline{B} \cdot \overline{A}},\ Y5 = \overline{C \cdot \overline{B} \cdot A},\ Y6 = \overline{C \cdot B \cdot \overline{A}},\ Y7 = \overline{C \cdot B \cdot A}.$$

Крім інформаційних входів *A, B, C* дешифратори мають додаткові входи керування *G*. Сигнали на цих входах, наприклад, дозволяють функціонувати дешифратору або переводять його у пасивний стан, при якому, незалежно від сигналів на інформаційних входах, на всіх виходах встановлюється рівень логічної одиниці.

Вхід дозволу дешифратора може бути прямим або інверсним. У дешифраторів з прямим входом дозволу активним рівнем є рівень логічної одиниці, а у дешифраторів з інверсним входом – рівень логічного нуля.

Шифратор – це типова комбінаційна схема, призначена для перетворення просторового унітарного коду в двійковий код з природним порядком ваг.

Унітарний код, як уже зазначалось, має в своєму записі одну одиницю. З урахуванням цього можна вважати, що шифратор виконує функцію зворотну функції дешифратора, хоча в загальному випадку вихідний код може відрізнятися від коду з природним порядком ваг.

Таблиця істинності шифратора для трирозрядного вхідного коду з природним порядком ваг наведено у таблиці 10.5.

Шифратори та дешифратори використовуються для:

1. Перетворення десяткових цифр у двійкову систему числення.
2. Перетворення кодів.
3. Реалізації логічних функцій за допомогою дешифратора.

Таблиця 10.5. Таблиця істинності шифратора для трирозрядного вхідного коду з природним порядком ваг

X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0	F_2	F_1	F_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Контрольні запитання

1. Які типові схеми називають комбінаційними?
2. Які різновиди входів можуть бути в типових комбінаційних схемах та яке їх призначення?
3. Дайте означення суматора та наведіть класифікацію суматорів.
4. Охарактеризуйте різні методи побудови схем однорозрядних повних суматорів.
5. Дайте означення багаторозрядного суматора. Охарактеризуйте методи побудови багато розрядних суматорів з паралельним та послідовним переносом.
6. Наведіть означення дешифратора. Охарактеризуйте повні та неповні дешифратори.
7. Що таке шифратор та як його можна використовувати для організації функціонування клавіатури?

ЛЕКЦІЯ 11

ТИПОВІ КОМБІНАЦІЙНІ СХЕМИ: МУЛЬТИПЛЕКСОРИ, ДЕМУЛЬТИПЛЕКСОРИ, КОДОПЕРЕТВОРЮВАЧІ, ПРИСТРОЇ ПОРІВНЯННЯ (КОМПАРАТОРИ)

11.1. Мультиплексори та демультиплексори

Мультиплексор – комбінаційна логічна схема, що є керованим перемикачем, який під'єднує до виходу один із інформаційних входів даних. Номер під'єднаного входу дорівнює числу (адресі), яке визначається комбінацією логічних значень на входах керування. Крім інформаційних входів та входів керування, схема мультиплексора містить вхід дозволу, при подачі на який активного рівня, мультиплексор переходить в пасивний стан, при якому сигнал на виході зберігає постійне значення незалежно від значення інформаційних та керуючих сигналів. Число інформаційних входів у мультиплексора, як правило, дорівнює 2, 4, 8 або 16. Якщо, наприклад, у мультиплексора 16 входів та до кожного із них під'єднано 16 джерел числових сигналів – генераторів упорядкованих цифрових слів, то байти із будь-якого із них можна передавати на єдиний вихід. Для вибору будь-якого із 16 каналів необхідно мати 4 входи вибору (2^4=16), на які подається двійкова адреса каналу. Так, для передачі даних з каналу номер 10 на входи вибору необхідно подати код 1010.

Як приклад, розглянемо мультиплексор з чотирма інформаційними входами, символьне зображення якого подано на рисунку 11.1. Даний мультиплексор має чотири інформаційні входи D_0, D_1, D_2, D_3, два адресні входи A_0, A_1, вхід для подачі стробуючого сигналу C та один вихід Q. Кожному інформаційному входу мультиплексора приписується номер, що називається адресою. При подачі стробуючого сигналу на вхід C мультиплексор вибирає один з входів, адреса якого задається двійковим кодом на адресних входах, та з'єднує його з виходом. Таким чином, подаючи на адресні входи адреси різних інформаційних входів, можна передавати цифрові сигнали з цих входів на вихід Q. Кількість інформаційних входів $n_{інф}$ та кількість адресних входів $n_{адр}$ пов'язані співвідношенням $n_{інф} = 2^{n_{адр}}$.

Таблиця 11.1. Функціонування мульплексора

A_1	A_0	C	Q
*	*	0	0
0	0	1	D_0
0	1	1	D_1
1	0	1	D_2
1	1	1	D_3

Функціонування мульплексора визначається таблицею 11.1.

При відсутності стробуючого сигналу ($C = 0$) зв'язок між інформаційними входами та виходами відсутній ($Q = 0$). При подачі стробуючого сигналу ($C = 1$) на вихід передається логічний рівень того з інформаційних

входів D_i, номер якого i в двійковій формі заданий на адресних входах. Так, при заданні адреси $A_1A_0 = 11_2 = 3_{10}$ на вихід Q буде передаватися сигнал інформаційного входу з адресою 3_{10}, тобто D_3.

За таблицею 11.1 можна записати такий логічний вираз для виходу Q:

$$Q = \left(D_0\overline{A}_1\overline{A}_0 + D_1\overline{A}_1A_0 + D_2A_1\overline{A}_0 + D_3A_1A_0\right)\cdot C.$$

Побудована за цим виразом комбінаційна схема подана на рисунку 11.1.

Одне з основних використань мультиплексорів – синтез логічних функцій. При цьому кількість елементів, що буде використовуватися в комбінаційній схемі, побудованій за цією функцією, значно зменшується.

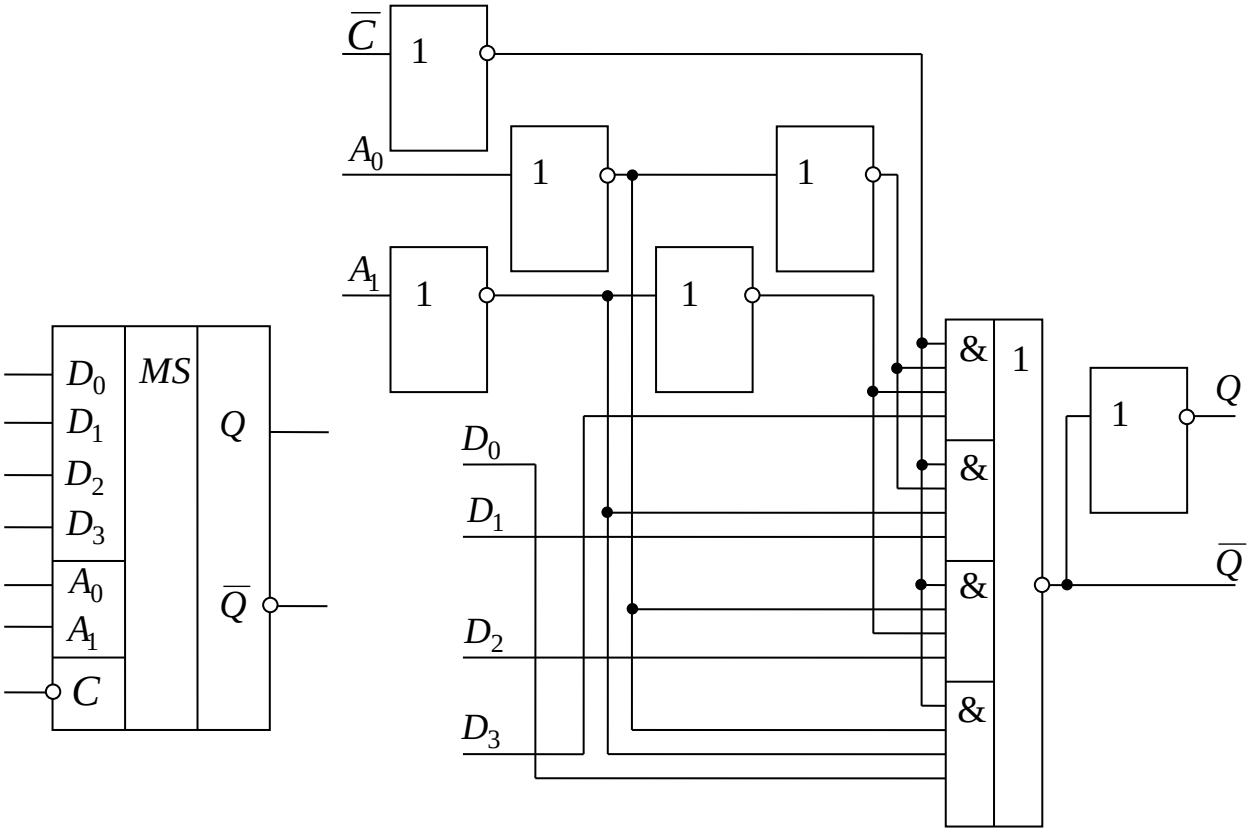


Рисунок 11.1. Умовне позначення та схема мультиплексора з 4 інформаційними входами

Мультиплексори також застосовуються, наприклад, у мікропроцесорах для видачі на одні й ті самі виводи адреси та даних, що дає можливість істотно скоротити загальну кількість виводів мікросхеми; у мікропроцесорних системах керування мультиплексори встановлюються на віддалених об’єктах для можливості передачі інформації по одній лінії від декількох встановлених на них сенсорів.

Демультиплексори у функціональному відношенні протилежні мультиплексорам. За їх допомогою сигнали з одного інформаційного входу розподіляються у потрібній послідовності на декілька виходів. Вибір потрібного виходу, як й в мультиплексорі, забезпечується встановленням відповідного коду на адресних входах.

Принципи роботи демультиплексора пояснимо за допомогою схеми на рисунку 11.2, на якій позначено: *X* – інформаційний вхід , *A* – вхід адреси, *Y0*, *Y*1 – виходи. Легко бачити, що при *A*=0 сигнал інформаційного каналу передається на вихід *Y0*, а при *A*=1 – на вихід *Y*1.

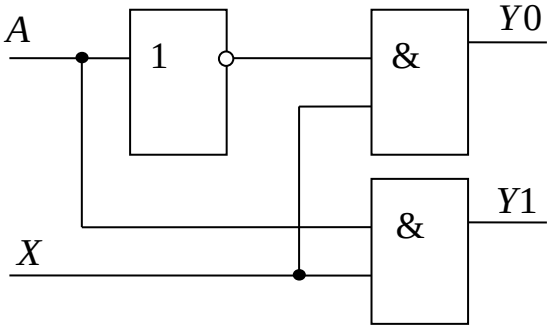


Рисунок 11.2.Схема демультиплексора

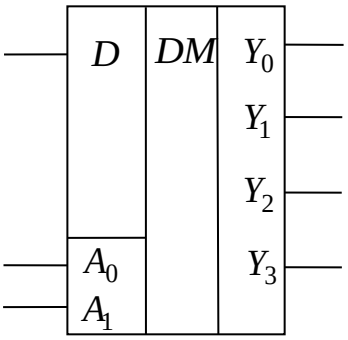


Рисунок 11.3. Умовне позначення демультиплексора

На рисунку 11.3 показано умовне позначення демультиплексора з чотирма виходами. Функціонування цього демультиплексора визначається таблицею 11.2. Якщо на вхід демультиплексора подати константу $D=1$, то на вибраному відповідно із заданою адресою виході буде логічна 1, на інших виходах – логічний 0.

Таблиця 11.2. Функціонування демультиплексора

A_1	A_0	Y_0	Y_1	Y_2	Y_3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D

Використання демультиплексора може суттєво спростити побудову логічного пристрою, що має декілька виходів, на яких формуються різні логічні функції одних й тих самих змінних. За необхідності отримати більшу кількість виходів може бути побудоване демультиплексорне дерево.

11.2 Кодоперетворювачі

Кодоперетворювач – комбінаційна схема, яка перетворює n-елементний код в m-елементний код та використовується для шифрації та дешифрації цифрової інформації. Співвідношення між числами m и n можуть бути різними ($m=n,\ m>n, m<n$).

Кодоперетворювачі можна побудувати двома методами:

1) кодоперетворювач реалізується як система булевих функцій групи аргументів. Схему перетворювача кодів в такому випадку будують шляхом синтезу схеми з декількома виходами.

2) кодоперетворювач подається у вигляді пари дешифратор-шифратор. При цьому зручно використовувати скорочену форму закону функціонування: двійкові коди заміняють їх десятковими еквівалентами. Кількість входів

дешифратора дорівнює кількості входів перетворювача кодів, кількість виходів шифратора дорівнює кількості виходів перетворювача кодів.

11.3 Пристрої порівняння

Пристрій порівняння (*цифровий компаратор*) – комбінаційна схема, що призначена для порівняння двох багаторозрядних двійкових чисел та формування результату порівняння у вигляді цифрових сигналів. Розрізняють порівняння на рівність та на нерівність.

У найпростішому випадку необхідно встановити факт рівності чисел A та B. Така задача виникає, наприклад, при порівнянні постійного числа A з числом B, яке в кожен наступний такт змінює своє значення на 1 (збільшується або зменшується).

Для визначення моменту, коли $A = B$, проводиться порозрядне додавання за модулем 2. При n-розрядних числах пристрій складається з n суматорів за модулем 2, виходи яких з'єднані з елементом АБО. Тільки при співпадінні значень всіх розрядів чисел A та B на виходах всіх суматорів буде 0. Якщо числа відрізняються хоча б в одному розряді, на виході відповідного суматора та на загальному виході буде 1.

При застосуванні елемента АБО, навпаки, рівності чисел відповідає вихідний сигнал 1 (рисунок 11.4).

Схема порівняння на нерівність реалізується у вигляді мікросхеми та має окрім інформаційних входів три виходи для стикування з попередньою мікросхемою та три виходи, які є виходами всієї багаторозрядної схеми порівняння або використовуються для формування вхідних сигналів наступної мікросхеми. Умовне позначення компаратора чотирирозрядних двійкових чисел, який реалізується у вигляді мікросхеми в багатьох серіях елементів, подано на рисунку 11.5.

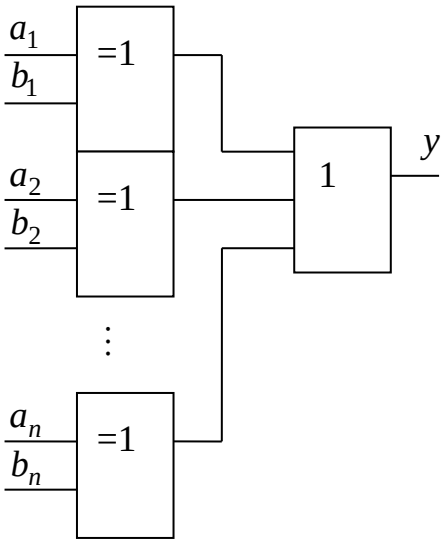


Рисунок 11.4 Схема порівняння на рівність

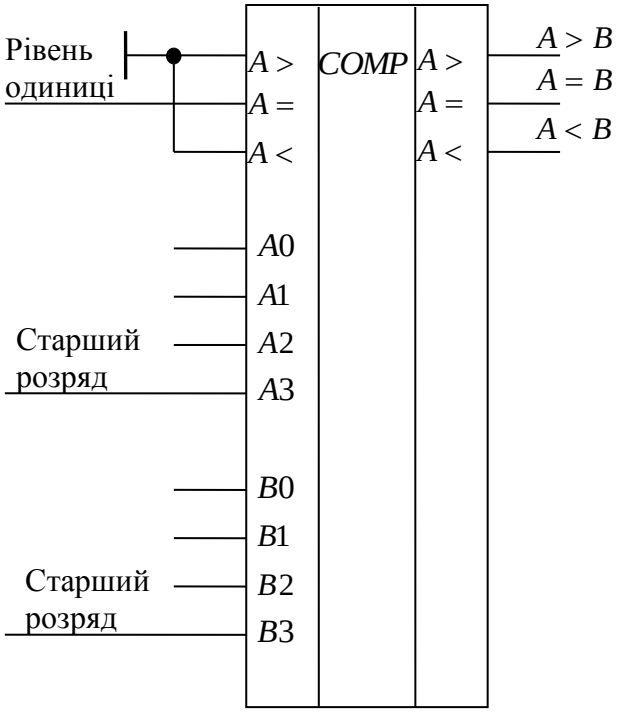


Рисунок 11.5. Умовне позначення чотирирозрядного компаратора кодів

Порівняння багаторозрядних чисел виконують, починаючи зі старших розрядів вхідних чисел A та B. Припускається рівність попередніх, відсутніх старших розрядів, тому на вхід «$A=$» подається активний сигнал (одиниця), а на входи «$A>$» та «$A<$» подається пасивний сигнал, в нашому випадку логічний рівень. Якщо порівнюються числа розрядністю більше чотирьох, то виходи компаратора старших розрядів з'єднуються з однойменними входами молодших розрядів чисел, що порівнюються. Виходами всього багаторозрядного компаратора кодів є виходи компаратора наймолодших розрядів, що порівнюються.

Контрольні запитання

1. Які типові схеми називають комбінаційними?
2. Дайте означення мультиплексора та наведіть принцип його функціонування.
3. Для чого призначені мультиплексори.
4. Охарактеризуйте призначення мультиплексорів для синтезу логічних функцій.
5. Поясніть принцип та мету побудови мультиплексорного дерева.
6. Що таке демультиплексор? Наведіть принцип його функціонування.
7. Дайте означення кодоперетворювачів та наведіть методи їх побудови.
8. Яка комбінаційна схема називається компаратором? Охарактеризуйте два типи компараторів.

ЛЕКЦІЯ 12

ТРИГЕРИ

12.1 Загальні відомості про тригери

Комбінаційна схема зі зворотними зв'язками, що має два стійких стани та призначена для записування та зберігання одного біта інформації, називається *елементарним автоматом* або *тригером*.

Під дією вхідних сигналів тригер переходить з одного стійкого стану в інший. При цьому напруга на його виході стрибкоподібно змінюється. Для коректної роботи цифрових автоматів необхідно виключити вплив перехідних процесів в тригерах та КС на зміну станів цифрового автомата та на вихідний сигнал. Ця вимога виконується при використанні складної багатофазної системи синхронізуючих сигналів для блока пам'яті та вихідної КС.

Як правило, тригер має два виходи – прямий та інверсний. Кількість входів залежить від структури та функцій, що виконуються тригером.

За логічним функціонуванням розрізняють тригери типів RS, D, T, JK та ін. Окрім того, використовуються комбіновані тригери, в яких поєднуються одночасно декілька типів, та тригери зі складною вхідною логікою (групами входів, пов'язаних між собою логічними залежностями).

За способом записування інформації тригери поділяють на асинхронні та синхронні. В *асинхронних тригерах* інформація може записуватися неперервно та визначається інформаційними сигналами, які діють на входах у даний момент часу. Якщо інформація записується в тригер тільки в момент дії синхронізуючого сигналу, то такий тригер називають *синхронним*. Окрім інформаційних входів, синхронізовані тригери мають вхід синхронізації (тактовий вхід). За способом сприйняття тактових сигналів тригери поділяються на такі, що *керуються рівнем*, та такі, що *керуються фронтом*. Керування рівнем означає, що при одному рівні тактового сигналу тригер сприймає вхідні сигнали та реагує на них, а при іншому не сприймає та лишається в незмінному стані. При керуванні фронтом дозвіл на перемикання дається тільки у момент перепаду тактового сигналу (на його фронті або спаді). В інший час незалежно від рівня тактового сигналу тригер не сприймає вхідні сигнали та залишається в незмінному стані. Тригери, що керуються фронтом, називаються тригерами з *динамічним керуванням*.

Динамічний вхід може бути прямим або інверсним. Пряме динамічне керування означає дозвіл на перемикання при зміні тактового сигналу з нульового значення на одиничне, інверсне – при зміні тактового сигналу з одиничного на нульовий.

За характером процесу переключення тригери поділяються на *одноступеневі* та *двоступеневі*. В одноступеневому тригері переключення в новий стан відбувається одразу, а в двоступеневому – за етапами. Двоступеневі тригери складаються з вхідної та вихідної сходинки. Перехід в новий стан відбувається в обох сходинках одночасно. Один із рівнів тактового сигналу

дозволяє приймати інформацію у вхідну сходинку при незмінному стані вихідної сходинки. Другий рівень тактового сигналу дозволяє передачу нового стану з вхідної сходинки у вихідну. Двоступеневі тригери позначаються двома буквами T.

У цифровій техніці прийняті такі позначення входів та виходів тригерів:

- Q – прямий вихід тригера;
- $\overline{Q}$ – інверсний вихід тригера;
- S – роздільний вхід встановлення в одиничний стан (напруга високого рівня на прямому виході Q);
- R – роздільний вхід встановлення в нульовий стан (напруга низького рівня на прямому виході Q);
- D – інформаційний вхід (на нього подається інформація, яка призначена для занесення в тригер);
- C – вхід синхронізації;
- T – лічильний вхід.

Найбільше розповсюдження в цифрових пристроях отримали RS-тригер з двома стійкими входами, D-тригер, лічильний T-тригер та JK-тригер. Розглянемо функціональні можливості кожного з них.

12.2 *RS*-тригер

Асинхронний *RS*-тригер. Даний тригер має два вхідних канали *R* та *S*: вхід *S* (set) називається входом встановлення в одиницю, вхід *R* (reset) – входом встановлення в нуль. Таблиця переходів такого тригера подана в таблиці 12.1.

У наведеній таблиці переходів *RS*-тригера прийняті такі позначення: R_t, S_t, Q_t значення логічних змінних у момент часу t на входах R, S та виході Q; Q_{t+1} – стан тригера після перемикання; "-" – невизначений стан на тих наборах, де вхідні сигнали R_t та S_t одночасно набувають значення 1 (заборонена комбінація сигналів).

Таблиця 12.1. Таблиця переходів асинхронного *RS*-тригера

R_t	S_t	Q_t	Q_{t+1}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	-
1	1	1	-

Таблиці переходів асинхронного *RS*-тригера відповідає неповністю визначена карта Карно (рисунок 12.1). Замінивши символи "-" спочатку на 0 (рисунок 12.2), а потім на 1 (рисунок 12.3) та виконавши склеювання, дістанемо логічні рівняння асинхронного *RS*-тригера:

$$Q_{t+1} = \overline{R}_t (S_t \vee Q_t), \qquad (12.1)$$

$$Q_{t+1} = S_t \vee \overline{R}_t Q_t. \qquad (12.2)$$

Логічні вирази (12.1) та (12.2) визначають новий стан тригера Q_{t+1} залежно від стану Q_t та вхідних сигналів R_t

та S_t. У подальшому для спрощення індекс t у правій частині логічного виразу будемо опускати.

	$\overline{Q}_t$	Q_t
$\overline{R}\overline{S}$	0	1
$\overline{R}S$	1	1
RS	-	-
$R\overline{S}$	0	0

Рисунок 12.1. Карта Карно асинхронного *RS*-тригера

	$\overline{Q}_t$	Q_t
$\overline{R}\overline{S}$	0	1
$\overline{R}S$	1	1
RS	0	0
$R\overline{S}$	0	0

Рисунок 12.2. Заміна невизначеностей на карті Карно нулями

	$\overline{Q}_t$	Q_t
$\overline{R}\overline{S}$	0	1
$\overline{R}S$	1	1
RS	1	1
$R\overline{S}$	0	0

Рисунок 12.3. Заміна невизначеностей на карті Карно одиницями

Залежно від логічної структури розрізняють *RS*-тригери з прямими та інверсними входами. Тригери таких типів побудовані на двох логічних елементах: 2АБО-НІ або 2І-НІ. Вихід кожного з елементів з'єднується із одним зі входів іншого елемента, що забезпечує тригеру два стійких стани.

Реалізація асинхронного *RS*-тригера з прямими входами здійснюється на елементах АБО-НІ. Для цього логічний вираз (12.1) перепишемо у вигляді, який зручно реалізовувати на елементах АБО-НІ:

$$Q_{t+1} = \overline{\overline{\overline{R}(S_t \vee Q_t)}} = \overline{R \vee \overline{(S_t \vee Q_t)}}. \qquad (12.3)$$

Схема асинхронного RS-тригера на двох елементах АБО-НІ з логічними зв'язками на основі виразу (12.1) показана на рисунку 12.4.

Перетворимо логічний вираз (12.2) до вигляду, який зручно реалізовувати на елементах І-НІ:

$$Q_{t+1} = \overline{\overline{S \vee \overline{R}Q}} = \overline{\overline{S} \cdot \overline{\overline{R} \cdot Q}}. \qquad (12.4)$$

Схема асинхронного *RS*-тригера з інверсними входами на двох елементах І-НІ з логічними зв'язками на основі виразу (12.2) показана на рисунку 12.5.

У таблицях 12.2 та 12.3 наведені таблиці переходів для розглянутих вище тригерів. У таблицях 12.2 та 12.3 символом Q^t позначені рівні, які були на виході тригера до подачі на його входи так званих активних рівнів. *Активним* називають логічний рівень, що діє на вході логічного елемента та однозначно визначає логічний рівень вихідного сигналу (незалежно від логічних рівнів, що діють на інших входах). Для елементів АБО-НІ за активний рівень приймають високий рівень, а для елементів І-НІ – низький рівень. Рівні, подача яких на один із входів не призводить до модифікації логічного рівня на виході елемента, називають *пасивними*. Рівні Q^{t+1} позначають логічні рівні на виході тригера після подачі інформації на його входи. Для тригера з прямими входами $Q^{t+1} = 1$ при S=1 та R=0; $Q^{t+1} = 0$ при S=0 та R=1; $Q^{t+1} = Q^t$ при S=0 та R=0.

При R=S=1 стан тригера буде невизначеним (в таблиці 12.2 та 12.3 цей стан позначено прочерком), бо під час дії інформаційних сигналів логічні рівні на виході тригера однакові $Q^{t+1} = \overline{Q}^{t+1}$, а після закінчення їхньої дії тригер може рівноймовірно прийняти будь-який зі стійких станів. Тому така комбінація є забороненою.

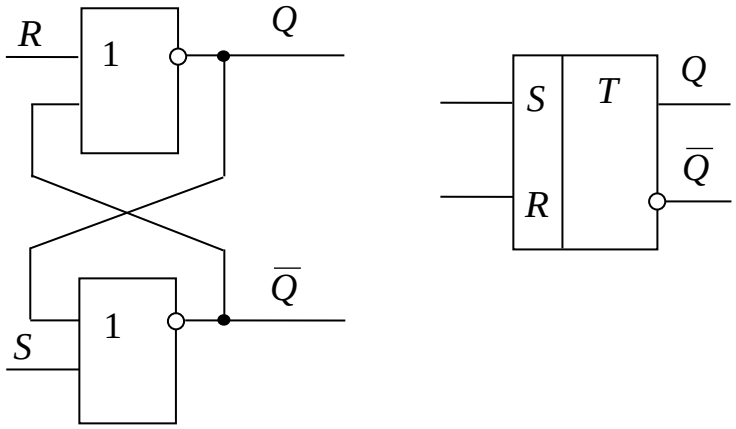


Рисунок 12.4. Схема та умовне позначення *RS*-тригера з прямими входами

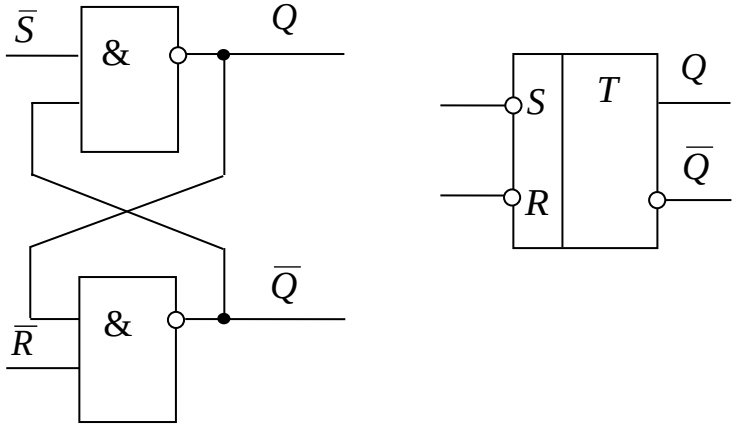


Рисунок 12.5. Схема та умовне позначення *RS*-тригера з інверсними входами

Таблиця 12.2. Таблиця переходів асинхронного *RS*-тригер з прямими входами

S	R	Q^t	Q^{t+1}
0	1	0	0
1	0	0	1
0	0	0	0
1	1	0	-
0	1	1	0
1	0	1	1
0	0	1	1
1	1	1	-

Таблиця 12.3. Таблиця переходів асинхронного *RS*-тригер з інверсними входами

$\overline{S}$	$\overline{R}$	Q^t	Q^{t+1}
0	1	0	1
1	0	0	0
0	0	0	-
1	1	0	0
0	1	1	1
1	0	1	0
0	0	1	-
1	1	1	0

Режим S=1, R=0 називають режимом записування 1 (бо $Q^{t+1}=1$); режим S=0 та R=1 – режимом записування 0. Режим S=0, R=0 називають режимом зберігання інформації, бо інформація на виході залишається незмінною. Для тригера з інверсними входами режим записування логічної 1 реалізується при S=0, R=1, режим записування логічного 0 – при S=1, R=0. При S=R=0 забезпечується зберігання інформації. Комбінація S=R=1 є забороненою.

Синхронний *RS*-тригер. Схема *RS*-тригера дає змогу запам'ятовувати стани комбінаційної схеми, але оскільки в початковий момент часу може виникати перехідний процес (у цифрових схемах цей процес називається гонки), то запам'ятовувати стани комбінаційної схеми потрібно тільки у визначені моменти часу, коли всі перехідні процеси завершені. Це означає, що більшість цифрових схем потребують сигналу синхронізації (тактового сигналу). Всі перехідні процеси в комбінаційній схемі повинні закінчуватися за час періоду синхросигналу, що подається на входи тригерів.

Для побудови синхронного *RS*-тригера на елементах АБО-НІ потрібно замінити в логічному виразі (12.3) S та R на добутки $\overline{\overline{CS}}$ та $\overline{\overline{CR}}$:

$$Q_{t+1} = = \overline{\overline{\overline{CR}} \vee \overline{(\overline{\overline{CS}} \vee Q)}} = \overline{\overline{C} \vee \overline{R} \vee \overline{(\overline{C} \vee \overline{S} \vee Q)}}\,. \qquad (12.5)$$

Схему такого тригера наведено на рисунку 12.6. Елементи 1 та 2 складають схему керування з інверсними входами, а елементи 3 та 4 утворюють фіксатор – елемент пам'яті тригера, що будується на двох інверторах, пов'язаних один з іншим «нахрест», так що вихід одного з'єднаний з входом іншого. При значенні сигналів $\overline{C}=0$ та $\overline{S}=0$ на виході елемента 2 встановлюється логічна 1 (тобто $CS=1$), та тригер переходить в стан 1. При значенні сигналів $\overline{C}=0$ та $\overline{R}=0$ на виході елемента 1 встановлюється логічна 1 (тобто $CR=1$), та тригер переходить в стан 0. Комбінація сигналів $\overline{C}=\overline{S}=\overline{R}=0$, заборонена, тому що призводить до невизначеного стану тригера.

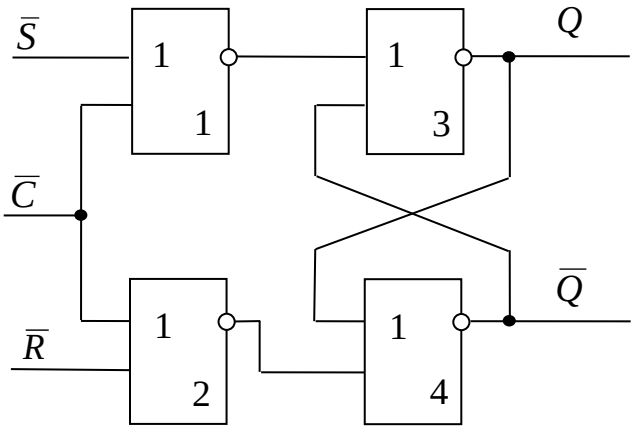


Рисунок 12.6. Схема синхронного *RS*-тригера на елементах АБО-НІ

Для побудови синхронного *RS*-тригера на елементах І-НІ потрібно замінити в логічному виразі (12.4) S та R на добутки CS та CR:

$$Q_{t+1} = \overline{\overline{CS} \cdot \overline{\overline{CR} \cdot Q}}\,. \qquad (12.6)$$

Схему *RS*-тригера, реалізація якого здійснена на елементах І-НІ**,** наведено на рисунку 12.7. При $C = 0$ на виходах елементів 1 та 2 діють одиничні сигнали, та фіксатор (елементи 3 та 4) зберігає незмінний стан. Якщо $C = 1$, то для сигналів S та R елементи 1, 2 стають інверторами, та схема фіксатора отримує нульовий сигнал встановлення або скидання від входу, на якому діє одиничний сигнал. Таким чином, перемикання дозволяється тільки при $C = 1$.

У таблиці 12.4 наведена таблиця переходів синхронного *RS*-тригера.

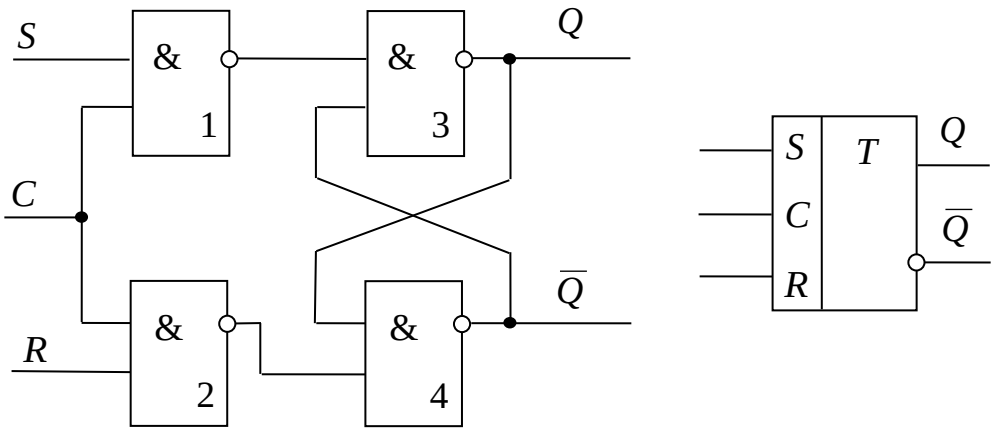


Рисунок 12.7. Схема та умовне позначення синхронного *RS*-тригера на елементах І-НІ

Робота тригера може описуватися таблицею функцій входів (таблиця 12.5), в якій вказується, які набори керуючих сигналів у поточний момент автоматного часу необхідно подати на входи тригера, щоб у наступний момент автоматного часу він перейшов з поточного вказаного стану в новий вказаний стан. Таблиця функцій входів будується за таблицею переходів для тригера.

Таблиця 12.4. Таблиця переходів синхронного *RS*-тригера

C	R	S	Q^t	Q^{t+1}	Примітка
0	X	X	0	0	Режим зберігання інформації
0	X	X	1	1	
1	0	0	0	0	Режим зберігання інформації
1	0	0	1	1	
1	0	1	0	1	Режим встановлення одиниці $S = 1$
1	0	1	1	1	
1	1	0	0	0	Режим записування нуля $R = 1$
1	1	0	1	0	
1	1	1	0	-	$R = S = 1$ заборонена комбінація
1	1	1	1	-	

Таблиця 12.5. Таблиця функцій входів синхронного *RS*-тригера

Q^t	Q^{t+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Із таблиці 12.5 можна отримати функцію збудження пам'яті автомата при синтезі на базі асинхронного *RS*-тригера з прямими входами. Наприклад, якщо деякий автомат, що містить у своєму складі три *RS*-тригери, переходить зі стану $a_i = 010$ у стан $a_j = 110$, то для забезпечення такого переходу функції збудження повинні бути:

- для першого тригера при переході з 0 в 1 – $R_1 = 0$, $S_1 = 1$;
- для другого тригера при переході з 1 в 1 – $R_2 = 0$, $S_2 = X$;
- для третього тригера при переході з 0 в 0 – $R_3 = X$, $S_3 = 0$.

12.3 *D*-тригер

Тригером типу D називається запам'ятовуючий елемент з двома стійкими станами, що має один інформаційний вихід та один вхід синхронізації. Логіка функціонування *D*-тригера описується логічним рівнянням:

$$Q_{t+1} = C_t \cdot D_t. \qquad (12.7)$$

Це рівняння показує, що після перемикання стан *D*-тригера повторює значення сигналу на *D*-вході в тактові моменти часу. Тому в літературі *D*-тригери часто називають тригерами затримки (від Delay – затримка).

Одна із можливих структурних схем однотактного *D*-тригера та його умовне позначення наведені на рисунку 12.8.

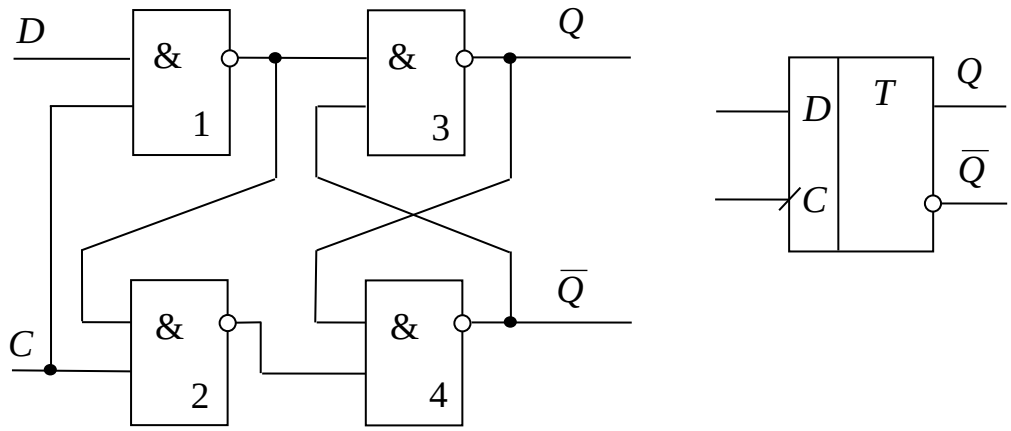


Рисунок 12.8. Схема та умовне позначення тактового *D*-тригера

Якщо рівень сигналу на вході *C*=0, стан тригера стійкий та не залежить від рівня сигналу на інформаційному вході. При цьому на входи *RS*-тригера з інверсними входами (елементи 3 та 4) надходять пасивні рівні (*S*=*R*=1). При подачі на вхід синхронізації рівня *C*=1 інформація на прямому виході буде

повторювати інформацію, що подається на вхід *D*. Таким чином, при *C*=0 $Q^{t+1} = Q^t$, а при *C*=1 $Q^{t+1} = D$ (таблиця 12.6).

Таблиця 12.6. Таблиця переходів тактового *D*-тригера

D	Q^t	Q^{t+1}
0	0	0
0	1	0
1	0	1
1	1	1

Таблиця 12.7. Таблиця функції входів тактового *D*-тригера

Q^t	Q^{t+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

В таблиці 12.6 символ Q^t означає логічний рівень на прямому виході до подачі імпульсу синхронізації, а Q^{t+1} – логічний рівень на цьому ж виході після подачі імпульсу синхронізації.

У такому тригері відбувається затримка сигналу на виході відносно сигналу, поданого на вхід, під час паузи між синхросигналами. Для стійкої роботи тригера необхідно, щоб протягом синхроімпульсу інформація на вході була незмінною.

З наведеної таблиці можна отримати таблицю функцій входів *D*-тригера (таблиця 12.7). Як бачимо з таблиці 12.7, стан, в який переходить тригер (середній стовпець), співпадає з сигналом *D* (правий стовпець), що надійшов на його вхід. У зв'язку з цим таблиця функцій збудження пам'яті автомата, що синтезується, з використанням *D*-тригерів повністю співпадатиме з кодованою таблицею переходів цього автомата.

12.4 Лічильний *T*-тригер

Тригером типу T називається запам'ятовуючий елемент з двома стійкими станами та одним інформаційним *T*-входом (рисунок 12.9). Інформація на виході асинхронного *T*-тригера змінює свій знак на протилежний за кожного позитивного (або за кожного негативного) перепаду напруги на вході. Синхронний *T*-тригер змінює свій стан тільки тоді, коли $T = 1$ та $C = 1$. Логіка функціонування асинхронного лічильного тригера подана таблицею переходів (таблиця 12.8) та таблицею функцій входів (таблиця 12.9) та описується логічним рівнянням:

$$Q_{t+1} = \overline{T}_t \cdot Q \vee T_t \cdot \overline{Q}_t . \qquad (12.8)$$

У серіях мікросхем, що випускаються, *T*-тригерів, як правило, немає. Але тригер такого типу може бути побудований на базі тактового *D*-тригера, якщо його інверсний вихід з'єднати з інформаційним входом (рисунок 12.10). Частота сигналу на виході *T*-тригера в два разу нижче частоти сигналу на вході, тому такий тригер можна використовувати як подільник частоти та двійковий лічильник.

З таблиці 12.9 можна отримати функцію збудження елементів пам'яті при синтезі автомата на базі *T*-тригера. Наприклад, якщо автомат перейшов зі стану $a_i = 010$ у стан $a_j = 110$, то для забезпечення цього переходу функції збудження для трьох *T*-тригерів, що входять до складу цього автомата, повинні бути:

- для першого тригера при переході з 0 в 1 – $T_1 = 1$;
- для другого тригера при переході з 1 в 1 – $T_2 = 0$;
- для третього тригера при переході з 0 в 0 – $T_3 = 0$.

Таблиця 12.8. Таблиця переходів *T*-тригера

T	Q^t	Q^{t+1}
0	0	0
0	1	1
1	0	1
1	1	0

Таблиця 12.9. Таблиця функції входів Т-тригера

Q^t	Q^{t+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

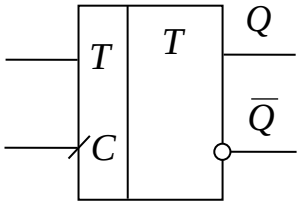


Рисунок 12.9. Умовне позначення *T*-тригера

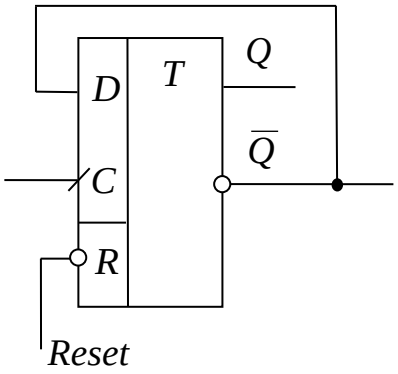


Рисунок 12.10. Схема *T*-тригера, що побудована на основі *D*-тригера

12.5 *JK*-тригер

Тригером типу JK називається запам'ятовуючий елемент з двома стійкими станами та інформаційними входами *J* (аналог *S*) та *K* (аналог *R*), які забезпечують відповідно роздільну установку станів '1' та '0'. Даний тригер функціонує подібно до *RS*-тригера, але при збігу сигналів *JK*=1 перемикається в протилежний стан, тобто реалізує додавання сигналів за модулем два. Таким чином, *JK*-тригер не має заборонених комбінацій вхідних сигналів. Тригер типу *JK* є універсальним, оскільки може виконувати функції *RS*-тригера (при роздільному надходженні сигналів *J* та *K*), *T*-тригера (при одночасній подачі сигналів *J* та *K*), *D*-тригера (при подачі сигналу від входу *J* через інвертор на вхід *K*). Таблиця переходів *JK* -тригера наведена в таблиці 12.10.

Таблиця 12.10. Таблиця переходів *JK*-тригера

C	K	J	Q^t	Q^{t+1}	Примітка
0	X	X	0	0	Режим зберігання інформації
0	X	X	1	1	
1	0	0	0	0	Режим зберігання інформації
1	0	0	1	1	
1	0	1	0	1	Режим встановлення одиниці $J=1$
1	0	1	1	1	
1	1	0	0	0	Режим записування нуля $K=1$
1	1	0	1	0	
1	1	1	0	1	$K=J=1$ лічильний режим тригера
1	1	1	1	0	

Як випливає з таблиці переходів (таблиця 12.10), для комбінацій вхідних сигналів $JK=00\div10$ тригер поводиться як *RS*-тригер, а при комбінації *JK* = 11 – як *T*-тригер.

За допомогою карти Карно (рисунок 12.11) легко отримати наступне логічне рівняння роботи тригера: $Q_{t+1}=\overline{K}\cdot Q\vee J\cdot\overline{Q}$.

Аналізуючи таблицю переходів (таблиця 12.10), зазначимо, що перехід тригера, наприклад, з 0 в 1 вимагає подачі вхідних сигналів *J*=1, *K*=0 або *J*=1, *K*=1, тобто *J*=1, *K*=X. Аналогічно міркуючи стосовно інших переходів, отримаємо таблицю функцій входів *JK*-тригера (таблиця 12.11).

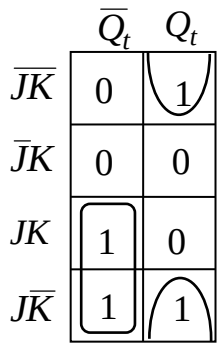


Рисунок 12.11. Карта Карно для *JK*-тригера

Таблиця 12.11. Таблиця функцій входів *JK* –тригера

Q^t	Q^{t+1}	J	K
0	0	X	0
0	1	1	X
1	0	X	1
1	1	0	X

Із таблиці 12.11 можна отримати функцію збудження елементів пам'яті при синтезі автомата на *JK*-тригерах. Наприклад, під час переходу деякого автомата, що містить у своєму складі три *JK*-тригери, зі стану $a_i=010$ у стан $a_j=110$, функції збудження повинні бути:

- для першого тригера при переході з 0 в 1 – *J1* = 1, *K1* = X;
- для другого тригера при переході з 1 в 1 – *J2* = X, *K2* = 0;
- для третього тригера при переході з 0 в 0 – *J3* = 0, *K3* = X.

Для побудови одноступеневого синхронного *JK*-тригера на елементах І-НІ потрібно замінити в рівнянні роботи тригера змінні *J* та *K* на сполучення *CK* та *CJ*, після чого виконати перетворення на основі правил подвійного заперечення та закону де Моргана:

$$Q_{t+1} = \overline{\overline{C \cdot K} \cdot Q \vee C \cdot J \cdot \overline{Q}} = \overline{\overline{C \cdot K} \cdot Q} \cdot \overline{C \cdot J \cdot \overline{Q}}\,. \qquad (12.9)$$

Схема одноступеневого *JK*-тригера з логічними зв'язками на основі рівняння роботи тригера та його умовне позначення подані на рисунку 12.12.

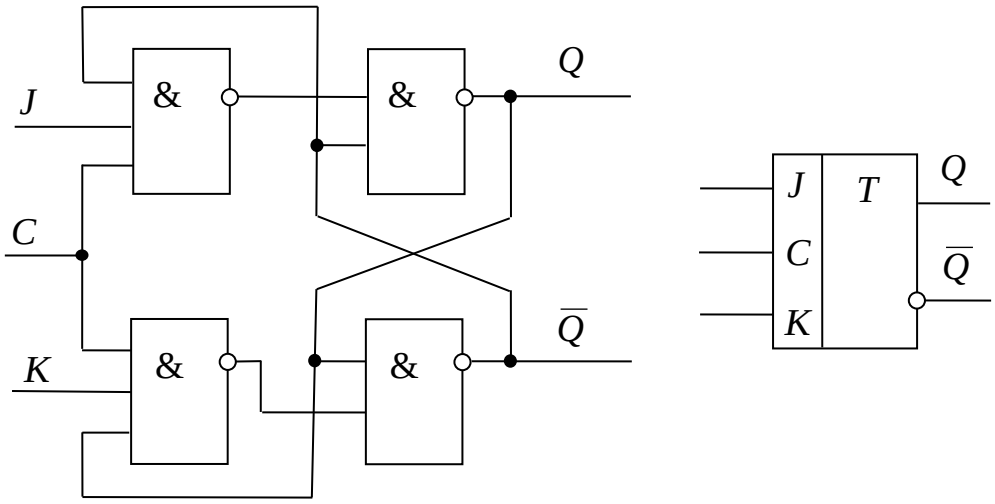


Рисунок 12.12. Схема та умовне позначення одноступеневого *JK*-тригера

Контрольні запитання

1. Наведіть означення та класифікацію тригерів.
2. Охарактеризуйте *RS*-тригери.
3. Охарактеризуйте *D*-тригери.
4. Охарактеризуйте *T*-тригери.
5. Дайте означення та наведіть особливості *JK*-тригерів.

ЛЕКЦІЯ 13

АБСТРАКТНА ТЕОРІЯ ЦИФРОВИХ ПРИСТРОЇВ

13.1 Основні поняття теорії абстрактних автоматів (пристроїв)

Цифровий автомат (пристрій) – це дискретний перетворювач інформації, що здатний приймати різні внутрішні стани, переходити під впливом вхідних сигналів або команд закладеної в нього програми розв'язання задачі з одного стану в інший та видавати вихідні сигнали. Перехід автомата з одного стану в інший здійснюється в певний дискретний момент часу.

Математичною моделлю цифрового автомата є так званий *абстрактний автомат*, означений як 6-компонентний об'єкт

$$\mathbf{S} = \{\mathbf{A}, \mathbf{Z}, \mathbf{W}, \delta, \lambda, a_1\}, \tag{13.1}$$

де $\mathbf{A} = \{a_m, m = \overline{1, M}\}$ – множина станів (внутрішній алфавіт);

$\mathbf{Z} = \{z_f, f = \overline{1, F}\}$ – вхідний алфавіт, що є множиною вхідних сигналів (букв, символів);

$\mathbf{W} = \{w_g, g = \overline{1, G}\}$ – вихідний алфавіт, що є множиною вихідних сигналів (букв, символів);

$\delta : \mathbf{A} \times \mathbf{Z} \to \mathbf{A}$ – функція переходів, що реалізує відображення декартового добутку $\mathbf{A} \times \mathbf{Z}$ в $\mathbf{A}$ (іншими словами функція δ деяким парам стан–вхідний сигнал (a_m, z_f) ставить у відповідність стани автомата $a_s = \delta(a_m, z_f),\ a_s \in \mathbf{A}$);

$\lambda : \mathbf{A} \times \mathbf{Z} \to \mathbf{W}$ – функція виходу, що реалізовує відображення декартового добутку $\mathbf{A} \times \mathbf{Z}$ в $\mathbf{W}$ (функція λ парам стан–вхідний сигнал (a_m, z_f) ставить у відповідність вихідні сигнали автомата $w_g = \lambda(a_m, z_f),\ w_g \in \mathbf{W}$);

$a_1 \in \mathbf{A}$ – початковий стан автомата.

Абстрактний автомат (рисунок 13.1) має один вхід та один вихід. Його функціонування відбувається в дискретному часі, що подається цілими додатними значеннями $t = 0,1,2,\ldots$ В кожен момент t дискретного часу автомат знаходиться в деякому стані $a(t)$ з множини можливих станів автомата, причому в початковий момент $t = 0$ він завжди знаходиться в початковому стані $a(0) = a_1$. У момент t, будучи в стані $a(t)$, автомат здатний сприйняти на вході букву вхідного алфавіту $z(t) \in \mathbf{Z}$. Відповідно до функції виходів цифровий автомат видає в той же момент часу t букву вихідного алфавіту $w(t) = \lambda(a(t), z(t))$ та відповідно до функції переходів перейде в наступний стан $a(t+1) = \delta(a(t), z(t)),\ a(t) \in \mathbf{A},\ w(t) \in \mathbf{W}$.

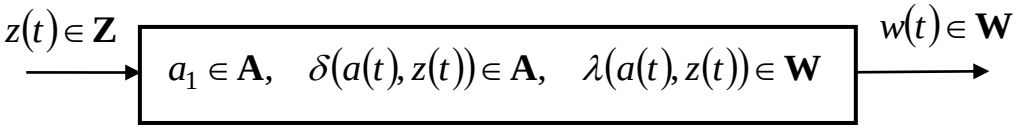


Рисунок 13.1. Абстрактний автомат

Таким чином, з математичної точки зору, цифровий автомат реалізує деяке відображення множини слів (послідовності символів) вхідного алфавіту $\mathbf{Z}$ у множину слів вихідного алфавіту $\mathbf{W}$. Тобто, якщо на вхід автомата, встановленого в початковий стан a_1, подавати деяке вхідне слово $z(0), z(1), \ldots$, то на виході автомата з'явиться вихідне слово $w(0), w(1), \ldots$.

Теорія абстрактних цифрових автоматів розглядає автомат без вивчення його структури, розглядаючи його як "чорну скриньку", тобто основна увага приділяється дослідженню поведінки автомата стосовно зовнішнього середовища.

Поняття стану в означенні автомата введено у зв'язку з тим, що часто виникає необхідність в описуванні поведінки систем, виходи яких залежать не тільки від стану входів у даний момент часу, але й від деякої передісторії, тобто від сигналів, які надійшли на входи системи раніше. Стани якраз й відповідають деякій пам'яті про минуле, даючи змогу усунути час як явну змінну та виразити вихідний сигнал як функцію стану та входу в даний момент часу.

На практиці найбільше поширення набули два класи абстрактних автоматів – автомати Мілі (Mealy) та Мура (Moore).

Закон функціонування автомата Мілі задається рівняннями

$$a(t+1) = \delta(a(t), z(t)); \tag{13.2}$$

$$w(t) = \lambda(a(t), z(t)),\ t = 0,1,2\ldots \tag{13.3}$$

Закон функціонування автомата Мура задається рівняннями

$$a(t+1) = \delta(a(t), z(t)); \tag{13.4}$$

$$w(t) = \lambda(a(t)),\ t = 0,1,2\ldots \tag{13.5}$$

Порівнюючи закони функціонування бачимо, що, на відміну від автомата Мілі, вихідний сигнал в автоматі Мура залежить тільки від поточного стану автомата та в явному вигляді не залежить від вхідного сигналу. Для повного задавання автомата Мілі або Мура додатково до законів функціонування, необхідно вказати початковий стан.

Окрім автоматів Мілі та Мура іноді виявляється зручно користуватися суміщеною моделлю цих автоматів – так званим C-автоматом.

Під *абстрактним C-автоматом* розуміють математичну модель дискретного пристрою, яка задається восьмикомпонентним об'єктом

$$\mathbf{S} = \{\mathbf{A}, \mathbf{Z}, \mathbf{W}, \mathbf{U}, \delta, \lambda_1, \lambda_2, a_1\}, \tag{13.6}$$

де $\mathbf{A} = \{a_m,\ m = \overline{1, M}\}$ – множина станів;

$\mathbf{Z} = \{z_f,\ f = \overline{1, F}\}$ – вхідний алфавіт;

$\mathbf{W} = \{w_g,\ g = \overline{1, G}\}$ – вихідний алфавіт типу 1;

$\mathbf{U} = \{u_h,\ h = \overline{1, H}\}$ – вихідний алфавіт типу 2;

$\delta : \mathbf{A} \times \mathbf{Z} \to \mathbf{A}$ – функція переходів, що реалізовує відображення $\mathbf{A} \times \mathbf{Z}$ в $\mathbf{A}$;

$\lambda_1 : \mathbf{A} \times \mathbf{Z} \to \mathbf{W}$ – функція виходів, що реалізовує відображення $\mathbf{A} \times \mathbf{Z}$ в $\mathbf{W}$;

$\lambda_2 : \mathbf{A} \to \mathbf{U}$ – функція виходів, що реалізовує відображення $\mathbf{A}$ в $\mathbf{U}$;

$a_1 \in \mathbf{A}$ – початковий стан автомата.

Абстрактний C-автомат можна подати у вигляді пристрою з одним входом, на який надходять символи з вхідного алфавіту $\mathbf{Z}$ та двома виходами, на яких з'являються символи з алфавітів $\mathbf{W}$ та $\mathbf{U}$. Відмінність C-автомата від автоматів Мілі та Мура полягає в тому, що він одночасно реалізує дві функції виходів λ_1 та λ_2, кожна з яких характерна для цих автоматів окремо. *Функціонування C-автомата* можна описати такими рівняннями:

$$a(t+1) = \delta(a(t), z(t)); \tag{13.7}$$

$$w(t) = \lambda_1(a(t), z(t)); \tag{13.8}$$

$$u(t) = \lambda_2(a(t)); \; t = 0,1,2\ldots \tag{13.9}$$

Вихідний символ $u_h = \lambda_2(a_m)$ видається весь час, поки автомат знаходиться в стані a_m. Вихідний сигнал $w_g = \lambda_1(a_m, z_f)$ видається під час дії вхідного сигналу z_f при знаходженні автомата в стані a_m.

Розглянуті вище абстрактні автомати поділяють на:

1) повністю визначені та частково визначені;
2) детерміновані та ймовірнісні;
3) синхронні та асинхронні.

Повністю визначеним абстрактним автоматом називають абстрактний цифровий автомат, у якого функція переходів та функція виходів визначені для всіх пар (a_m, z_f) декартового добутку $\mathbf{A} \times \mathbf{Z}$.

Частково визначеним абстрактним автоматом називаєють абстрактний автомат, у якого функція переходів або функція виходів, або обидві ці функції визначені не для всіх пар (a_m, z_f) декартового добутку $\mathbf{A} \times \mathbf{Z}$.

До *детермінованих абстрактних автоматів* належать автомати, у яких виконується умова однозначності переходів, а саме, автомат, що знаходиться в деякому стані a_m, під дією вхідного сигналу z_f обов'язково (з ймовірністю 1) перейде в деякий стан a_n.

Ймовірнісний автомат – це автомат, у якому при заданому стані a_m та заданому вхідному сигналі z_f можливий перехід із заданими ймовірностями в різні стани.

Для означення *синхронних* та *асинхронних* автоматів вводиться поняття стійкого стану. Стан a_s автомата називається стійким, якщо для будь-якого стану a_i та вхідного сигналу z_j таких, що $\delta(a_i, z_j) = a_s$ має місце $\delta(a_s, z_j) = a_s$, тобто стан стійкий, якщо потрапивши в цей стан під дією деякого сигналу z_j, автомат вийде з нього тільки під дією іншого сигналу z_k, відмінного від z_j.

Автомат, у якого всі стани стійкі, називають *асинхронним*, у протилежному випадку автомат називається *синхронним*.

Абстрактний автомат називають *скінченним автоматом*, якщо множини $\mathbf{A} = \{a_m, m = \overline{1, M}\}$, $\mathbf{Z} = \{z_f, f = \overline{1, F}\}$, $\mathbf{W} = \{w_g, g = \overline{1, G}\}$ є скінченними.

Автомат називають *ініціальним автоматом*, якщо в ньому виділений початковий стан a_1.

13.2 Методи описування та задавання автоматів

Для того, щоб задати скінченний автомат, необхідно описати всі елементи множини $\mathbf{S} = \{\mathbf{A, Z, W}, \delta, \lambda, a_1\}$. Множини **A, Z, W** описуються та задаються простим переліком своїх елементів. Існує кілька різних способів задавання функцій δ та λ (отже, й всього автомата в цілому): табличний, матричний та графічний.

При *табличному методі* задавання автомат Мілі описується за допомогою двох таблиць. Одна з них (таблиця переходів) задає функцію δ, тобто $a(t+1) = \delta(a(t), z(t))$ (таблиця 13.1), друга (таблиця виходів) – функцію λ, тобто $w(t) = \lambda(a(t), z(t))$ (таблиця 13.2).

Кожному стовпцю з наведених таблиць поставлено у відповідність один стан із множини **A**, кожному рядку – один вхідний символ з множини **Z**. На перетині стовпця a_m та рядка z_f таблиці 13.1 записується стан a_s, в який повинен перейти автомат зі стану a_m під дією вхідного символу z_f, тобто $a_s = \delta(a_m, z_f)$. На перетині стовпця a_m та рядка z_f таблиці 13.2 записується вихідний символ w_g, виданий автоматом у стані a_m під час надходження на вхід символу z_f, тобто $w_g = \lambda(a_m, z_f)$.

Таблиця 13.1. Таблиця переходів автомата Мілі

	a_1	a_2	a_3	a_4
z_1	a_2	a_1	a_2	a_3
z_2	a_3	a_4	a_1	a_1

Таблиця 13.2. Таблиця виходів автомата Мілі

	a_1	a_2	a_3	a_4
z_1	w_1	w_2	w_1	w_2
z_2	w_2	w_3	w_4	w_5

Автомат Мілі може бути заданий однією суміщеною таблицею переходів-виходів (таблиця 13.3), в якій кожен елемент a_s / w_g записаний на перетині стовпця a_m та рядка z_f.

Автомат Мура задається однією таблицею переходів (таблиця 13.4), в якій кожному стовпцю приписується не тільки стан a_m, але й вихідний символ $w_g = \lambda(a_m)$, що відповідає цьому стану.

Для часткових автоматів Мілі та Мура в розглянутих таблицях на місці невизначених станів та вихідних символів ставиться прочерк. У таких автоматах вихідний сигнал на будь-якому переході завжди невизначений, якщо невизначеним є стан переходу. Крім того, вихідний сигнал може бути невизначеним й для деяких існуючих переходів.

Для задавання C-автоматів також використовується табличний метод. У цьому випадку таблиця переходів (таблиця 13.5) аналогічна таблиці переходів

автомата Мілі, а в таблиці виходів кожен стан відзначений відповідним вихідним сигналом u_h вихідного алфавіту типу 2 (таблиця 13.6).

Таблиця 13.3. Суміщена таблиця переходів-виходів для автомата Мілі

	a_1	a_2	a_3	a_4
z_1	a_2 / w_1	a_1 / w_2	a_2 / w_1	a_3 / w_2
z_2	a_3 / w_2	a_4 / w_3	a_1 / w_4	a_1 / w_5

Таблиця 13.4. Таблиця переходів автомата Мура

	w_1	w_2	w_3	w_4
	a_1	a_2	a_3	a_4
z_1	a_1	a_2	a_2	a_3
z_2	a_2	a_3	a_4	a_1

Таблиця 13.5. Таблиця переходів C-автомата

	a_1	a_2	a_3	a_4
z_1	a_1	a_2	a_2	a_3
z_2	a_3	a_4	a_1	a_2

Таблиця 13.6. Таблиця виходів C-автомата

	u_1	u_2	u_3	u_4
	a_1	a_2	a_3	a_4
z_1	w_1	w_4	w_1	w_2
z_2	w_3	w_2	w_1	w_3

Матричний метод задавання абстрактних автоматів полягає в поданні автомата у вигляді матриці з'єднань. Матриця з'єднань довільного абстрактного автомата є квадратною та має стільки стовпців (рядків), скільки різних станів має автомат, що розглядається. Кожен стовпець (рядок) матриці з'єднань помічається буквою стану автомата. Якщо автомат ініціальний, то перший зліва стовпець та перший зверху рядок матриці його з'єднань помічаються буквою початкового стану автомата. В клітинці матриці з'єднань, що знаходиться на перетині стовпця, поміченого буквою стану a_i, та рядка, поміченого буквою стану a_j автомата, ставиться вхідний сигнал z_f (або диз'юнкція вхідних сигналів), під дією якого здійснюється перехід автомата зі стану a_j в стан a_i. Якщо матрицею з'єднань задається абстрактний автомат Мілі, то поруч з буквою вхідного сигналу z_f в дужках вказується буква вихідного сигналу w_g, який автомат Мілі видає, здійснюючи перехід $a_j = \delta(a_i, z_f)$. Матриця з'єднань автомата Мілі, поданого таблицею переходів (таблиця 13.1), наведена в таблиці 13.7.

Таблиця 13.7. Матриця з'єднань автомата Мілі

	a_1	a_2	a_3	a_4
a_1	-	$z_1(w_1)$	$z_2(w_2)$	-
a_2	$z_1(w_2)$	-	-	$z_2(w_3)$
a_3	$z_2(w_4)$	$z_1(w_1)$	-	-
a_4	$z_2(w_5)$	-	$z_1(w_2)$	-

Якщо матрицею з'єднань задається абстрактний автомат Мура, то вихідними сигналами помічаються стани автомата, що ідентифікують рядки матриці з'єднань.

При *графічному методі* автомат задається у вигляді орієнтованого графа, вершини якого відповідають станам, а дуги – переходам між ними. Дуга, направлена з вершини a_m, задає перехід в автоматі зі стану a_m в стан a_s. На початку цієї дуги записується вхідний символ $z_f \in \mathbf{Z}$, що викликає даний перехід $a_s = \delta(a_m, z_f)$. Для графа автомата Мілі вихідний сигнал $w_g \in \mathbf{W}$, формований на переході, записується в кінці дуги, а для автомата Мура – поряд з вершиною a_m, відзначеною станом a_m, в якому він формується. Якщо перехід в автоматі зі стану a_m в стан a_s проводиться під дією кількох вхідних символів, то дузі графа, що направлена з a_m в a_s, приписуються всі ці вхідні та відповідні вихідні символи. Граф C-автомата містить вихідні символи двох типів, що позначаються на графі як на графах відповідних автоматів. Графи автоматів, що задані своїми таблицями переходів та виходів (таблиці 13.1, 13.2, 13.4–13.6) наведено на рисунках 13.2–13.4.

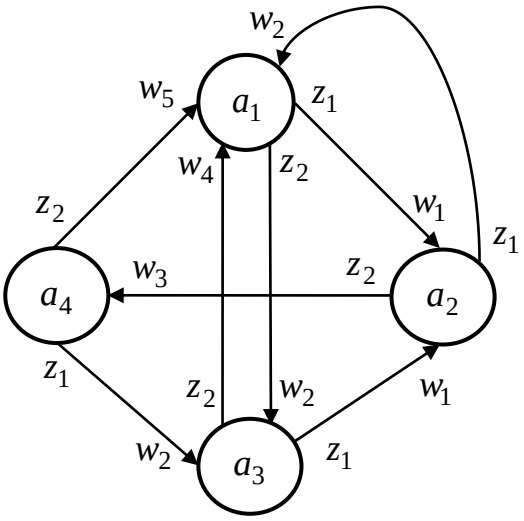


Рисунок 13.2. Граф автомата Мілі

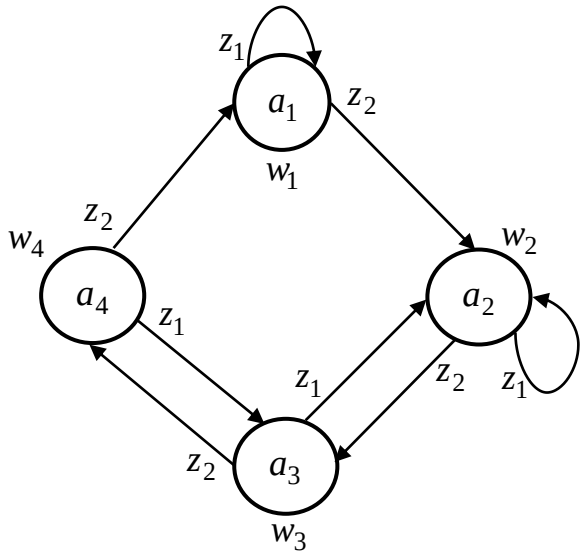


Рисунок 13.3. Граф автомата Мура

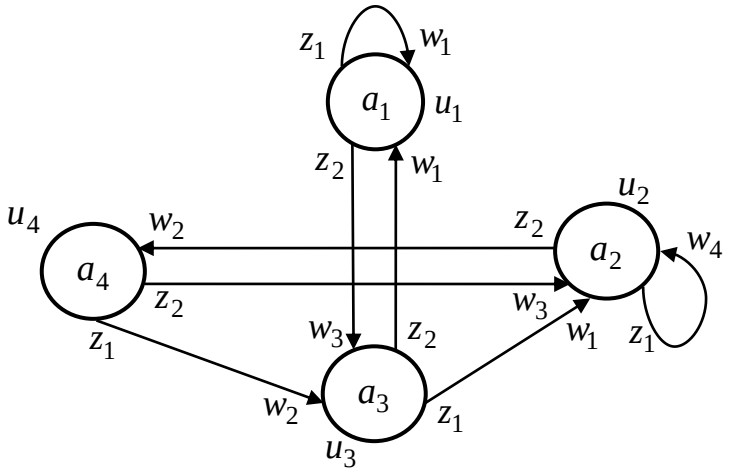


Рисунок 13.4. Граф *C*-автомата

13.3. Зв'язок між автоматами Мілі та Мура

Розглянемо автомат Мілі, який задано таблицями переходів та виходів (таблиці 13.8 та 13.9). Подамо на вхід автомата, встановленого в стан a_1, вхідне слово $\xi = z_1z_2z_2z_1z_2z_2$. Оскільки $\delta(a_1, z_1) = a_2$, $\lambda(a_1, z_1) = w_2$, то під впливом вхідного символу z_1 автомат перейде в стан a_2 та видасть на переході вихідний символ w_2. Потім, знаходячись у стані a_2, під впливом символу z_2 автомат перейде в стан $a_1 = \delta(a_2, z_2)$ та видасть символ $w_1 = \lambda(a_2, z_2)$ й т.д. У таблиці 13.10 наведено послідовність станів, які автомат проходить, сприймаючи вхідне слово ξ, та вихідні символи, що видаються автоматом на цих переходах.

Таблиця 13.8. Таблиця переходів автомата Мілі

	a_1	a_2	a_3
z_1	a_2	a_3	a_3
z_2	a_3	a_1	a_1

Таблиця 13.9. Таблиця виходів автомата Мілі

	a_1	a_2	a_3
z_1	w_2	w_1	w_2
z_2	w_2	w_1	w_1

Таблиця 13.10. Реакція автомата Мілі

Послідовність станів	a_1	a_2	a_1	a_3	a_3	a_1	a_3
Вхідне слово ξ	z_1	z_2	z_2	z_1	z_2	z_2	
Вихідне слово ω	w_2	w_1	w_2	w_2	w_1	w_2	

Вихідне слово $\omega = \lambda(a_1, \xi)$ називається *реакцією* автомата Мілі в стані a_1 на вхідне слово ξ. У нашому випадку $\omega = w_2w_1w_2w_2w_1w_2$.

Як бачимо з наведеного прикладу, у відповідь на вхідне слово довжиною k автомат Мілі видасть послідовність станів довжиною $k+1$ та вихідне слово довжиною k.

У загальному вигляді поведінку автомата Мілі, встановленого в стан a_m, можна записати у вигляді таблиці 13.11.

Таблиця 13.11. Реакція автомата Мілі в загальному вигляді

Вхідне слово	z_{i1}	z_{i2}	z_{i3}
Послідовність станів	a_m	$a_{i2}=\delta(a_m,z_{i1})$	$a_{i3}=\delta(a_{i2},z_{i2})$
Вихідне слово	$w_{i1}=\lambda(a_m,z_{i1})$	$w_{i2}=\lambda(a_{i2},z_{i2})$	$w_{i3}=\lambda(a_{i3},z_{i3})$

Аналогічно можна описати поведінку автомата Мура, що знаходиться в стані a_1, при приході вхідного слова $\xi=z_{i1},z_{i2},\ldots,z_{ik}$, враховуючи, що $w(t)=\lambda(a(t))$ (таблиця 12.12).

Таблиця 13.12. Реакція автомата Мура в загальному вигляді

Вхідне слово	z_{i1}	z_{i2}	z_{i3}	z_{i4}
Послідовність станів	a_m	$a_{i2}=\delta(a_m,z_{i1})$	$a_{i3}=\delta(a_{i2},z_{i2})$	$a_{i4}=\delta(a_{i3},z_{i3})$
Вихідне слово	$w_{i1}=\lambda(a_m,z_{i1})$	$w_{i2}=\lambda(a_{i2},z_{i2})$	$w_{i3}=\lambda(a_{i3},z_{i3})$	$w_{i4}=\lambda(a_{i4})$

Очевидно, що для автомата Мура вихідний символ $w_{i1}=\lambda(a_m)$ у момент часу i_1 не залежить від вхідного символу z_{i1} та визначається тільки станом a_m. Отже, символ w_{i1} ніяк не пов'язаний з вхідним словом ξ.

У зв'язку з цим під реакцією автомата Мура, встановленого в стан a_m, на вхідне слово $\xi=z_{i1},z_{i2},\ldots,z_{ik}$ розуміють вихідне слово тієї ж довжини $\omega=\lambda(a_m,\xi)=w_{i2}w_{i3}\ldots w_{ik+1}$, зсунуте відносно ξ на один такт.

Розглянемо приклад. Нехай заданий автомат Мура (таблиця 13.13). Подамо на вхід цього автомата ту ж послідовність, що й для автомата Мілі: $\xi=z_1z_2z_2z_1z_2z_2$. Послідовність зміни станів і вихідних символів, що видаються автоматом, наведено в таблиці 13.14.

Таблиця 13.13. Автомат Мура

	w_1	w_2	w_3	w_4
	a_1	a_2	a_3	a_4
z_1	a_2	a_3	a_4	a_4
z_2	a_4	a_1	a_1	a_1

Таблиця 13.14. Реакція автомата Мура

Послідовність станів	a_1 a_2 a_1 a_4 a_4 a_1 a_4
Вхідне слово ξ	z_1 z_2 z_2 z_1 z_2 z_2
Вихідне слово ω	w_1 w_2 w_1 w_2 w_2 w_1 w_2

Порівнюючи реакції автомата Мілі (таблиця 13.10) та автомата Мура (таблиця 13.14), можна зауважити, що ці реакції на одне й те саме слово ξ співпадають. Отже, автомати Мілі та Мура реалізують одне й те саме перетворення слів вхідного алфавіту. Такі автомати називають еквівалентними. Таким чином, два автомати з однаковими вхідними та вихідними алфавітами називають *еквівалентними*, якщо після встановлення їх у початковий стан їх реакції на будь-яке вхідне слово співпадають.

Твердження 13.1. Для кожного автомата Мілі A може бути побудований еквівалентний йому автомат Мура B, зокрема у випадку скінченності автомата A автомат B також може бути вибраний скінченним із кількістю станів, що дорівнює $(m+1)n$, *де* m *– кількість вхідних сигналів, а* n *– кількість станів в автоматі A.*

Перехід від автомата Мура до еквівалентного йому автомата Мілі тривіальний та легко здійснюється при графічному методі подання автомата. Для отримання графа автомата Мілі необхідно вихідний сигнал w_g, записаний поряд з вершиною a_s початкового автомата Мура, перенести на всі дуги, що входять у цю вершину. На рисунку 13.5 наведено граф автомата Мілі, еквівалентного автомату Мура (рисунок 13.3).

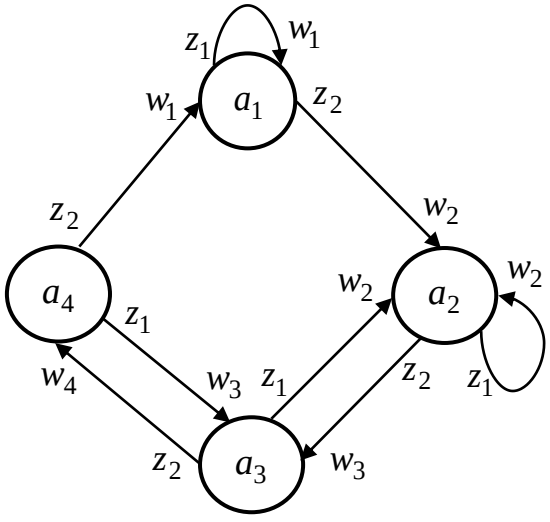


Рисунок 13.5. Автомат Мілі, еквівалентний автомату Мура (рисунок 13.3)

Легко переконатися, що отриманий автомат Мілі справді еквівалентний початковому автомату Мура. Для цього достатньо розглянути реакцію обох

автоматів на довільну вхідну послідовність. Необхідно також відзначити, що в еквівалентному автоматі Мілі кількість станів така ж, як й в початковому автоматі Мура.

Перехід від автомата Мілі до еквівалентного йому автомата Мура складніший. Це пов'язано з тим, що в автоматі Мура в кожному стані видається тільки один вихідний символ. Як й у попередньому випадку, перехід найбільш наочно можна зробити при графічному методі задавання автомата. У цьому випадку кожен стан a_i початкового автомата Мілі породжує стільки станів автомата Мура, скільки різних вихідних сигналів генерується в початковому автоматі при попаданні в стан a_i. Розглянемо перехід від автомата Мілі S_a до автомата Мура S_b на прикладі автомата, зображеного на рисунку 13.2.

Як випливає з рисунка 13.2, для автомата S_a при потраплянні його в стан a_1 формуються вихідні символи w_2, w_4, w_5, при попаданні в стан $a_2 - w_1$, w_2, $a_3 - w_2$, $a_4 - w_3$. Кожній парі (a_i, w_j), яка формується при попаданні в стан a_i, поставимо у відповідність стан b_k еквівалентного автомата Мура S_b з тим самим вихідним сигналом $w_j : b_1 = (a_1, w_2)$, $b_2 = (a_1, w_4)$, $b_3 = (a_1, w_5)$, $b_4 = (a_2, w_1)$, $b_5 = (a_2, w_2)$, $b_6 = (a_3, w_2)$, $b_7 = (a_4, w_3)$, тобто кожен стан a_i автомата Мілі породжує деяку множину $\mathbf{A}_i$ станів еквівалентного автомата Мура: $\mathbf{A}_1 = \{b_1, b_2, b_3\}$, $\mathbf{A}_2 = \{b_4, b_5\}$, $\mathbf{A}_3 = \{b_6\}$, $\mathbf{A}_4 = \{b_7\}$. Як бачимо, в еквівалентному автоматі Мура кількість станів 7. Граф S_b будують таким чином. Оскільки в автоматі S_a є перехід зі стану a_1 в стан a_2 під дією символу z_1 з видачею символу w_1, то з множини станів $\mathbf{A}_1 = \{b_1, b_2, b_3\}$, що породжуються станом a_1 автомата S_a в автоматі S_b має бути перехід у стан $b_4 = (a_2, w_1)$ під дією символу z_1 й т.д. Граф еквівалентного автомата Мура зображено на рисунку 13.6.

Якщо від автомата Мура S_b, еквівалентного автомата Мілі S_a (рисунок 13.6), знову перейти до автомата Мілі, то отримаємо автомат Мілі S_1 (рисунок 13.7).

Внаслідок транзитивності відношення еквівалентності два автомати Мілі S_1 та S_a також будуть еквівалентними, але в останнього будуть на 3 стани більше. Тобто, еквівалентні між собою автомати можуть мати різну кількість станів, у зв'язку з чим виникає задача знаходження мінімального (тобто з мінімальною кількістю станів) автомата в класі еквівалентних між собою автоматів.

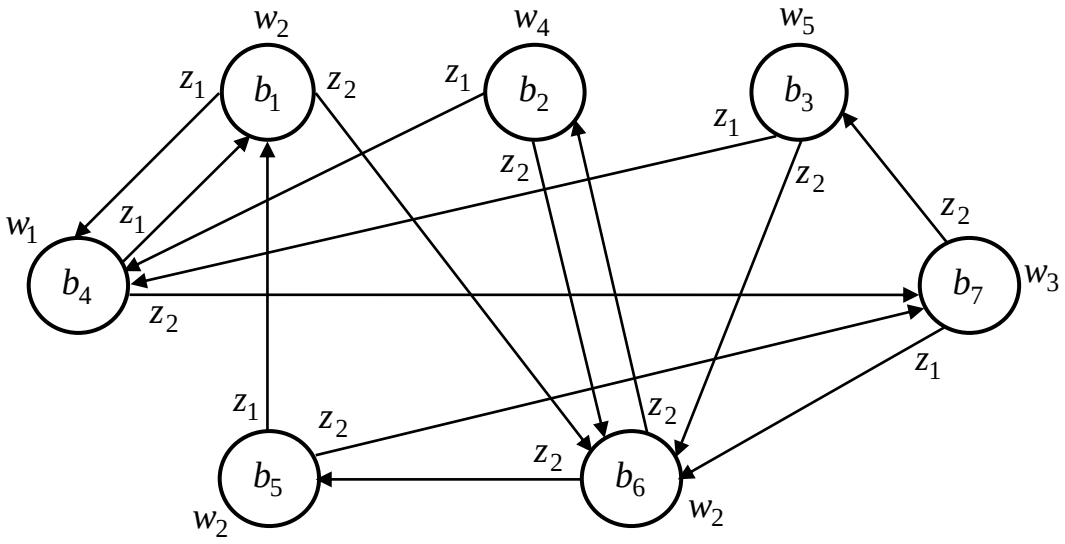


Рисунок 13.6. Автомат Мура S_b, що еквівалентний автомату Мілі S_a

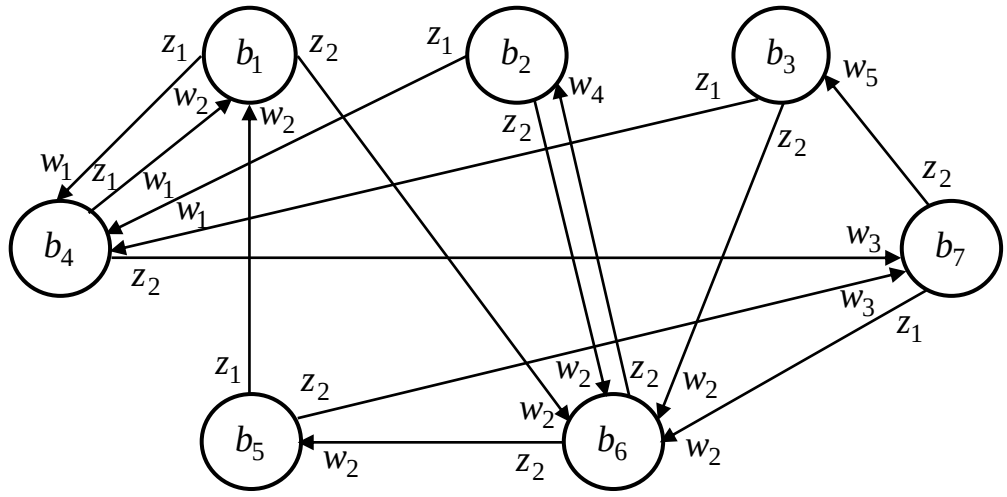


Рисунок 13.7. Автомат Мілі S_1, що еквівалентний автомату Мура S_b

13.4 Мінімізація кількості внутрішніх станів абстрактних автоматів

13.4.1 Мінімізація кількості внутрішніх станів повністю визначених автоматів

Розглянемо метод мінімізації повністю визначених автоматів, запропонований Ауфенкампом та Хоном. Ідея методу мінімізації повністю визначених автоматів полягає в розбитті всіх станів початкового абстрактного автомата на попарно непересічні класи еквівалентних станів та заміні кожного класу еквівалентності одним станом. Тобто мінімальний автомат, що виходить у результаті, має стільки станів, на скільки класів еквівалентності розбиваються стани початкового автомата.

Для початку дамо кілька означень.

Два стани абстрактного автомата називають *одноеквівалентними* в тому випадку, якщо реакції автомата в цих станах на будь-які вхідні слова співпадають. Об'єднання всіх одноеквівалентних станів абстрактного автомата утворює *перший клас еквівалентності*.

Одноеквівалентні стани автомата називають *двоеквівалентними*, якщо вони переводяться будь-яким вхідним сигналом також в одноеквівалентні стани. Об'єднання всіх двоеквівалентних станів утворює *другий клас еквівалентності*.

За індукцією, можна дати означення *i-еквівалентних* станів та *i-класів еквівалентності.*

Якщо для деякого i розбиття станів автомата на $(i+1)$ - класи співпадає з розбиттям на i-класи, то воно є розбиттям i на ∞-класи еквівалентності.

Розбиття множини внутрішніх станів автомата на ∞-класи є необхідним розбиттям на класи еквівалентності, при цьому таке розбиття може бути отримано за скінченну кількість кроків.

Все викладене вище безпосередньо стосується мінімізації автомата Мілі.

При мінімізації повністю визначених автоматів Мура вводиться поняття *0-еквівалентності станів* та розбиття множини станів на *0-еквівалентні класи*: до такого класу відносяться однаково відзначені стани автомата Мура.

Якщо два 0-еквівалентні стани будь-яким вхідним символом переводяться в два 0-еквівалентні стани, то їх називають одноеквівалентними. Всі подальші класи еквівалентності станів для автомата Мура означуються аналогічно наведеному для автоматів Мілі.

Розглянемо приклад мінімізації автомата Мілі, заданого таблицями переходів та виходів (таблиці 13.15 та 13.16).

Таблиця 13.15. Таблиця переходів автомата Мілі

	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_3	a_4	a_3	a_4	a_5	a_6
z_2	a_5	a_6	a_5	a_6	a_1	a_2

Таблиця 13.16. Таблиця виходів автомата Мілі

	a_1	a_2	a_3	a_4	a_5	a_6
z_1	w_1	w_1	w_1	w_1	w_1	w_1
z_2	w_1	w_1	w_2	w_2	w_1	w_1

З таблиці виходів (таблиця 13.16) отримуємо розбиття на 1-класи еквівалентності π_1, об'єднуючи в еквівалентні класи b_i стани з однаковими значеннями w_i у стовпцях таблиці:

$$\pi_1 = \{b_1, b_2\}; b_2 = \{a_1, a_2, a_5, a_6\}; b_2 = \{a_3, a_4\}.$$

Для отримання 2-еквівалентних станів будуємо таблицю 1-розбиття (таблиця 13.17), замінюючи в таблиці переходів стану a_1 відповідними класами еквівалентності b_1 або b_2.

З таблиці (таблиця 13.17) отримуємо 2-класи еквівалентності c_i та розбиття $\pi_2 = \{c_1, c_2, c_3\}$, де $c_1 = \{a_1, a_2\}$, $c_2 = \{a_5, a_6\}$, $c_3 = \{a_3, a_4\}$. Порівнюючи π_2 та π_1, відзначаємо, що це розбиття відрізняється одне від одного. Тому

аналогічно будуємо таблицю 2-розбиття (таблиця 13.18), знову замінюючи в таблиці переходів стану a_i відповідними класами еквівалентності c_i.

Таблиця 13.17. 1-розбиття

	b_1				b_2	
	a_1	a_2	a_5	a_6	a_3	a_4
z_1	b_2	b_2	b_1	b_1	b_2	b_2
z_2	b_1	b_1	b_1	b_1	b_1	b_1

Таблиця 13.18. 2-розбиття

	c_1		c_2		c_3	
	a_1	a_2	a_5	a_6	a_3	a_4
z_1	c_3	c_3	c_2	c_2	c_3	c_3
z_2	c_2	c_2	c_1	c_1	c_2	c_2

З отриманої таблиці 2-розбиття знаходимо 3-класи еквівалентності d_i та розбиття $\pi_3 = \{d_1, d_2, d_3\}$, де $d_1 = \{a_1, a_2\}$, $d_2 = \{a_5, a_6\}$, $d_3 = \{a_3, a_4\}$.

Порівнюючи π_3 та π_2, помічаємо, що $d_1 = c_1$, $d_2 = c_2$, $d_3 = c_3$, тобто $\pi_3 = \pi_2$. Отже, отримано розбиття на ∞-еквівалентні класи. Оскільки було отримано всього три класи, то мінімальний автомат міститиме всього три стани. Вибираємо з кожного класу d_i по одному стану та отримуємо множину станів $\mathbf{A}'$ мінімального автомата. Нехай, наприклад, $\mathbf{A}' = \{a_1, a_3, a_5\}$. Для отримання мінімального автомата з первинних таблиць переходів та виходів (таблиці 13.16 та 13.17) викреслюємо стовпці, які відповідають "зайвим станам" a_2, a_4, a_6. У результаті виходить мінімальний автомат Мілі (таблиці 13.19 та 13.20), який еквівалентний початковому автомату.

Таблиця 13.19. Таблиця переходів мінімального автомата Мілі

	a_1	a_3	a_5
z_1	a_3	a_3	a_5
z_2	a_5	a_5	a_1

Таблиця 13.20. Таблиця виходів мінімального автомата Мілі

	a_1	a_3	a_5
z_1	w_1	w_1	w_1
z_2	w_1	w_2	w_1

13.4.2 Мінімізація кількості внутрішніх станів частково визначених автоматів

Задачу мінімізації часткового автомата S можна сформулювати як задачу пошуку автомата S', який серед всіх автоматів, що покривають S, має найменшу кількість станів.

Задачу мінімізації кількості внутрішніх станів частково визначених автоматів розглянемо на основі роботи М.Полла та С.Ангера. Процес мінімізації часткових автоматів включає три основні етапи:

1. Знаходження всіх максимальних класів сумісності.
2. Знаходження мінімального замкнутого покриття.
3. Отримання за мінімальним замкнутим покриттям нового автомата.

Для початку дамо декілька визначень.

Вхідне слово $\xi = z_{i_1}...z_{i_k}$ називається *допустимим* для автомата S в стані a_m, якщо:

1) функція переходів $\delta(a_{i_j}, z_{i_j})$ визначена для всіх $j = \overline{1, k-1}$; $a_{i_j} = a_m$, $a_{i_{j+1}} = \delta(a_{i_j}, z_{i_j})$;

2) визначено заключний вихід $\lambda(a_m, \xi) = \lambda(a_{i_k}, z_{i_k})$.

Таким чином, відповідна допустимому вхідному слову послідрвність станів, що визначається функцією переходів, не містить невизначеного стану. Наприклад, для деякого автомата Мілі S_1, що заданий таблицями переходів та виходів (таблиці 13.21 та 13.22), вхідне слово $z_3z_2z_3z_1z_4$ допустиме в стані a_1 (див. таблицю 13.23).

Вхідні слова $z_3z_4z_3z_2z_1$ та $z_4z_1z_2z_2z_3$ в стані a_1 не є допустимими, оскільки при надходженні першого з них не визначено послідовність станів (див. таблицю 13.24), а при надходженні другого – не визначено вихід (див. таблицю 13.25).

Таблиця 13.21. Таблиця переходів автомата S_1

	a_1	a_2	a_3	a_4	a_5
z_1	a_2	a_3	a_3	-	-
z_2	-	a_5	a_4	a_1	-
z_3	a_3	a_2	-	a_2	a_1
z_4	a_2	-	a_5	-	-

Таблиця 13.22. Таблиця виходів автомата S_1

	a_1	a_2	a_3	a_4	a_5
z_1	w_1	w_1	w_1	-	-
z_2	w_2	w_2	w_2	w_2	-
z_3	-	w_1	-	-	w_2
z_4	w_1	-	w_1	-	-

Таблиця 13. 23. Реакція автомата S_1 на вхідне слово $z_3z_2z_3z_1z_4$

Вхідне слово	z_3	z_2	z_3	z_1	z_4	
Послідовність станів	a_1	a_3	a_4	a_2	a_3	a_5
Вихідне слово	-	w_2	-	w_1	w_1	

Таблиця 13. 24. Реакція автомата S_1 на вхідне слово $z_3z_4z_3z_2z_1$

Вхідне слово	z_3	z_4	z_3	z_2	z_1
Послідовність станів	a_1	a_3	a_5	a_1	-
Вихідне слово	-	w_1	w_2	w_2	-

Таблиця 13. 25. Реакція автомата S_1 на вхідне слово $z_4z_1z_2z_2z_3$

Вхідне слово	z_4	z_1	z_2	z_2	z_3	
Послідовність станів	a_1	a_2	a_3	a_4	a_1	a_3
Вихідне слово	w_1	w_1	w_2	w_2	-	

Стан a'_m автомата S' *покриває стан* a_m автомата S, якщо будь-яке вхідне слово, допустиме для S в стані a_m, допустиме й для S' в стані a'_m та при застосуванні до S в a_m та до S' в a'_m викликає в обох випадках однакові реакції всюди, де вихідні сигнали автомата S визначені. Наприклад, стан b_1 автомата S_2 (див. таблиці 13.26 та 13.27) покриває стани a_1 та a_2, а стан b_3 – стани a_2 та a_3 автомата S_1.

Таблиця 13.26. Таблиця переходів автомата S_2

	b_1	b_2	b_3	b_4
z_1	b_3	b_1	b_3	b_3
z_2	b_4	b_1	b_4	b_2
z_3	b_3	b_3	b_1	b_1
z_4	b_1	b_1	b_4	b_4

Таблиця 13.27. Таблиця виходів автомата S_2

	b_1	b_2	b_3	b_4
z_1	w_1	w_1	w_1	w_1
z_2	w_2	w_2	w_2	w_2
z_3	w_1	-	w_1	w_2
z_4	w_1	w_1	w_1	w_1

Відношення покриття рефлексивне, транзитивне, але не симетричне. Також очевидно, що автомат S', що покриває автомат S, «працює» так само, як автомат S, в умовах роботи S. Кажуть, що автомат S' «робить так само та, можливо, й більше», ніж автомат S'.

Автомат S' *покриває автомат* S, якщо для кожного стану a_m автомата S існує хоча б один стан a'_m автомата S', що покриває стан a_m. Так стан a_1 автомата S_1 покривається станами b_1 та b_2 автомата S_2, a_2 покривається станами b_1 та b_3, a_3 покривається станами b_3 та b_4, a_4 покривається станами b_2 та b_4, a_5 покривається станом b_4. Отже, автомат S_2 покриває автомат S_1.

Два стани a_m та a_s називаються *сумісними* та позначаються як $a_m \sim a_s$, якщо при подачі будь-якого допустимого в a_m та a_s вхідного слова отримані вихідні слова *несуперечливі*, тобто співпадають там, де обидва визначені. У протилежному випадку стани a_m та a_s є *несумісними* та позначаються як $a_m \nsim a_s$.

Наведемо ще одне визначення сумісності. Два стани a_m та a_s називаються *сумісними*, якщо:

1) виконується умова *сумісності по виходу* – для кожного входу $z_f \in Z$, для якого обидва виходи визначені, функції переходів рівні, тобто

$$\lambda(a_m, z_f) = \lambda(a_s, z_f);$$

2) для кожного входу $z_f \in Z$, для якого обидва наступних стани визначені, функції виходів сумісні

$$\delta(a_m, z_f) \sim \delta(a_s, z_f),$$

тобто обидва наступних за a_m та a_s стани також мають бути сумісні.

Наведене визначення відноситься до автомата Мілі. У випадку автомата Мура умову (1) необхідно замінити так: якщо обидва виходи $\lambda(a_m)$ та $\lambda(a_s)$ визначені, то $\lambda(a_m) = \lambda(a_s)$.

Очевидно, що умова сумісності рефлексивна, симетрична, але не транзитивна.

Множина станів C $(C \subseteq A)$ називається *класом сумісності*, якщо всі стани в C попарно сумісні. Клас сумісності C називається *максимальним класом сумісності*, якщо він не міститься в жодному іншому класі сумісності.

Нехай $B \subseteq A$ – підмножина множини станів автомата. Множина станів C слідує за множиною B по входу z_f, тоді й тільки тоді якщо:

$$C = \delta(B, z_f) = \{a_s \mid a_s = \delta(a_m, z_f),\ a_m \in B\}.$$

З поняття сумісності ясно, що множина станів, що слідує за деяким класом сумісності, також є класом сумісності.

Сумісні пари станів можна отримати на основі *трикутної таблиці*, яку будують за таблицями переходів та виходів автомата. Таблиця є трикутною, оскільки її вигляд обумовлений рефлексивністю та симетричністю відношення сумісності станів.

Побудуємо трикутну таблицю для автомата S_1 (таблиці 13.21 та 13.22). Для спрощення запису в цій таблиці замість a_i будемо писати i. Трикутну таблицю (таблиця 13.28) заповнюємо таким чином. На перетині рядка m та стовпця s проставляємо:

- позначку «×», якщо стани a_m та a_s несумісні по входу, наприклад, a_2 та a_5 (в таблиці 13.22 в рядку z_3 в стовпцях a_2 та a_5 знаходяться різні вихідні сигнали – w_1 та w_2 відповідно);
- множину всіх пар станів, що слідують за (a_m, a_s). Наприклад, для сумісності (a_1, a_3) необхідна сумісність (a_2, a_3) та (a_2, a_5), оскільки (a_2, a_3) та (a_2, a_5) слідують за множиною (a_1, a_3) по входах z_1 та z_4 відповідно (див. таблицю 13.21);

Таблиця 13.28. Трикутна таблиця для автомата S_1

2	2,3			
3	2,3 2,5	4,5		
4	2,3	1,5	1,4	
5	1,3	×	V	1,2
	1	2	3	4

- позначку «V», якщо за (a_m, a_s) не слідують пари, що відрізняються від (a_m, a_s), тобто пара (a_m, a_s) сумісна без додаткових умов, що накладаються на сумісність інших пар. В нашому прикладі це пара (a_3, a_5).

Для знаходження несумісних пар станів та, відповідно, також й сумісних пар, трикутна матриця розглядається по стовпцях. Знаходимо першу комірку матриці, що позначена знаком «×». Нехай це буде комірка (i, j), а в нашому випадку $(2,5)$. Тоді у всіх комірках, де є пара (i, j), ставиться «×», а вже розглянута комірка (i, j) позначається ще раз позначкою «×». Ця процедура проводиться для всіх комірок (з новими включно), що помічені один раз «×», та завершується, коли таких комірок не залишиться. Очевидно, що в такому випадку комірки без позначки «×» відповідають сумісним парам станів, а комірки з позначками – несумісним парам станів. Для розглянутого прикладу, отримаємо таблицю 13.29, з якої можна записати, що:

$$1\sim 2,\quad 1\sim 4,\quad 2\sim 3,\quad 3\sim 4,\quad 3\sim 5,\quad 4\sim 5$$
$$1\approx 3,\quad 1\approx 5,\quad 2\approx 4,\quad 2\approx 5.$$

Таблиця 13.29. Трикутна таблиця сумісних пар, що побудована за таблицею 13.28

2	2,3			
3	2,3 × 2,5 ×	4,5		
4	2,3	1,5 × ×	1,4	
5	1,3 × ×	× ×	V	1,2
	1	2	3	4

Наступним кроком знаходимо всі максимальні класи сумісності. Будемо вважати, що *множина станів* $B\,(B \subseteq A)$ *сумісна* з деяким станом $a_m \in A$ (позначимо як $a_m \sim B$), тоді та тільки тоді якщо $a_m \sim a_s$ для будь-якого $a_s \sim B$. У протилежному випадку $a_s \approx B$.

Алгоритм знаходження всіх класів сумісності полягає в наступному:

1. Починаємо складання списку Ф класів сумісності із сумісних пар в крайньому правому стовпчику трикутної матриці, що містить хоча б одну комірку без позначки «×». Отримаємо $\Phi = \{\overline{4,5}\}$.

2. Просуваємося вліво, виписуючи для кожного i-того стовпця множину станів $A_i \sim a_i$, тобто множину всіх станів, що відповідають коміркам без позначки «×» в i-тому стовпчику таблиці. Для нашого прикладу: $3 \sim \overline{4,5}$ (в третьому стовпчику немає позначки «×» в рядках 4 та 5).

Кожне A_i перетинаємо з кожним елементом поточного списку Ф. В нашому прикладі: $A_3 = \overline{4,5}$, список Ф також має один елемент $\overline{4,5}$, тому $\overline{4,5} \cap \overline{4,5} = \overline{4,5}$.

Якщо такий перетин містить більше одного стану, то дабавляємо в список об'єднання $\{a_i\}$ з результатом перетину: $\overline{3} \cup \overline{4,5} = \overline{3,4,5}$; $\Phi = \{\overline{4,5}, \overline{3,4,5}\}$.

Далі проводиться *максимізація* отриманої *множини* Ф – видалення всіх повторень множин в поточному списку та видалення всіх множин, що містяться в інших множинах списку: $\Phi = \{\overline{3,4,5}\}$.

Добавляємо в список всі пари, що складаються з a_i та різних станів з A_i, та не є підмножинами інших елементів списку Ф (при $i = 3$ таких добавлень немає).

Приведемо результат застосування другого кроку до всіх стовпців:

$$\Phi = \{\overline{4,5}\};$$
$$\overline{3} \sim \overline{4,5}, \quad \Phi = \{\overline{3,4,5}\};$$
$$\overline{2} \sim \overline{3}, \quad \Phi = \{\overline{3,4,5}, \overline{2,3}\};$$
$$\overline{1} \sim \overline{2,4}, \quad \Phi = \{\overline{3,4,5}, \overline{2,3}, \overline{1,2}, \overline{1,4}\}.$$

3. В список Ф, отриманий після другого кроку, додаємо всі одно елементарні множини, що складаються з станів, що не включені ні в які інші максимальні класи сумісності. В нашому прикладі таких немає.

Отже, для автомата S_1 список всіх максимальних класів сумісності рівний: $\Phi = \{\overline{3,4,5}, \overline{2,3}, \overline{1,2}, \overline{1,4}\}$.

Другим кроком мінімізації частинних автоматів є знаходження мінімального замкнутого покриття. Дамо декілька теоретичних аспектів до цього кроку.

Ланцюгом B, що породжений множиною станів C, називається система множин, що визначається наступним чином:

1. C вважається елементом ланцюга.

2. Якщо Q – елемент ланцюга, то всі множини, що слідують за Q та містять більше одного елемента, вважаються елементами ланцюга.

Множина D класів сумісності *замкнута*, якщо для будь-якого класу $C \in D$ кожен наступний за ним клас міститься хоча б в одному класі множини D. Очевидно, що ланцюг, що породжений деяким класом сумісності, є замкнутою множиною.

Якщо множина D класів сумісності включає в свої класи всі стани множини A, то говорять, що D покриває множину станів початкового автомата, а сама множина D буде називатися *покриттям* (*групировкою*).

Множина D класів сумісності називається *замкнутим покриттям* (*правильною групировкою*), якщо D – покриття (групировка) та воно замкнуте.

При перевірці на замкнутість деякого покриття, що містить деякий одноелементний клас сумісності, немає необхідності перевіряти замкнутість цього класу в даному покритті, оскільки будь-яка множина слідуючих за ним станів або порожня, або також одноелементна та, відповідно, завжди міститься в деякому класі покриття.

Очевидно, що множина всіх максимальних сумісних класів завжди є замкнуте покриття для даного автомата.

В роботах М.Полла та С.Ангера показано, що кожному автомату S' з M' станами, який покриває автомат $S = \{A, Z, W, \delta, \lambda, a_1\}$, відповідає в S замкнуте покриття потужності M' та навпаки, кожному замкнутому покриттю потужності M' в автоматі S відповідає автомат S', що покриває автомат S. Таким чином, задачу мінімізації кількості станів автомата можна звести до задачі знаходження замкнутої множини сумісних класів, що покриває множину станів початкового автомата та такої, що має мінімальну кількість класів сумісності, тобто до задачі знаходження мінімального замкнутого покриття.

Процедура отримання автомата $S' = \{A', Z', W', \delta', \lambda', a_1'\}$, що представляється знайденим для початкового автомата S замкнутим покриттям D, достатньо проста. Кожному класу сумісності $C_i \in D$ ставиться у відповідність стан a_i' нового автомата S'. Для кожного класу $C_i \in D$ та кожного $z_f = Z$ обчислюється множина слідуючих за C_i станів $T_{if} = \delta(C_i, z_f)$.

Функції переходів та виходів визначаються наступним чином:

$$\delta'(a_i', z_f) = \begin{cases} \text{не визначена, якщо } T_{if} = \varnothing; \\ a_j', \text{ де } a_j' - \text{стан, що відповідає одному із класів} \\ \quad \text{сумісності } C_j \in D, \text{ що містить } T_{if} = \varnothing. \end{cases}$$

$$\lambda'(a_i', z_f) = \begin{cases} \lambda(a_i, z_f), \text{ якщо існує } a_{i_q} \in C_i, \text{ таке,} \\ \quad \text{що } \lambda(a_i, z_f) \text{ визначена;} \\ \text{не визначена у протилежному випадку.} \end{cases}$$

За початковий стан a_1' автомата S' вибирається будь-який стан із множини A', що відповідає деякому класу сумісності, що містить початковий стан a_1 автомата S.

На основі викладеного вище методу можна за автоматом S_1 (таблиці 13.21 та 13.22) побудувати автомат S_2 (таблиці 13.26 та 13.27). За вихідне замкнуте покриття для нього була взята множина максимальних класів сумісності $\Phi = \{\overline{1,2}, \overline{1,4}, \overline{2,3}, \overline{3,4,5}\} = \{b_1, b_2, b_3, b_4\}$.

Зауважимо, що автоматів за даним замкнутим покриттям D можна побудувати декілька. Це пояснюється тим, що класи сумісності в D можуть перетинатися та для деяких множин слідуючих станів T_{if} може існувати декілька класів $C_j \in D$, що містять T_{if}, а тому й декілька варіантів визначення переходу $\delta'(a_i', z_f)$.

13.5 Декомпозиція абстрактних автоматів

Під *декомпозицією*, у загальному випадку, розуміють подання абстрактного автомата сукупністю кількох простіших автоматів. Декомпозиція зазвичай здійснюється паралельним, послідовним або змішаним поєднанням простіших автоматів. Тому можна розглядати кілька видів декомпозиції:

паралельну, послідовну та змішану. Таких автоматів, які розкладаються тільки в паралельну, послідовну чи змішану схему автоматів – незначна кількість відносно множини автоматів, які не можна подати паралельною, послідовною або змішаною декомпозицією. У зв'язку з цим вводиться поняття загальної декомпозиції абстрактного автомата, яку розуміють як подання абстрактного автомата сумісної роботи простіших автоматів зі зв'язками між ними. Розглянемо три основних види об'єднання автоматів: паралельне, послідовне та зі зворотним зв'язком.

Паралельне об'єднання двох автоматів $\mathbf{S}_1 = \{\mathbf{A}_1, \mathbf{Z}_1, \mathbf{W}_1, \delta_1, \lambda_1\}$ та $\mathbf{S}_2 = \{\mathbf{A}_2, \mathbf{Z}_2, \mathbf{W}_2, \delta_2, \lambda_2\}$ подано на рисунку 13.8. Вхідний алфавіт в обох автоматів є спільний. Виходи автоматів $\mathbf{S}_1$ та $\mathbf{S}_2$ з'єднані з функціональним перетворювачем φ (автоматом без пам'яті), що реалізує відображення $\varphi: \mathbf{W}_1 \times \mathbf{W}_2 \to \mathbf{W}$.

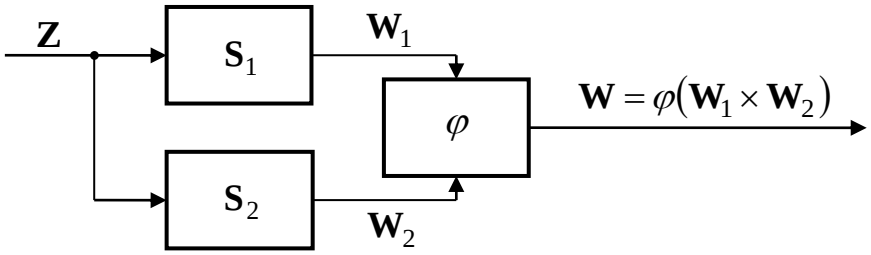


Рисунок 13.8. Паралельне об'єднання двох автоматів

Результуючим автоматом паралельного об'єднання двох автоматів $\mathbf{S}_1$ та $\mathbf{S}_2$ будемо вважати автомат $\mathbf{S} = \{\mathbf{A}, \mathbf{Z}, \mathbf{W}, \delta, \lambda\}$, в якого:

1. Множина станів $\mathbf{A} = \mathbf{A}_1 \times \mathbf{A}_2$ – множина всіх можливих пар, перші та другі компоненти яких є, відповідно, стани автоматів $\mathbf{S}_1$ та $\mathbf{S}_2$: $\mathbf{A} = \left\{ a_m = \left(a_{m_1}, a_{m_2}\right) \middle| a_{m_1} \in \mathbf{A}_1, a_{m_2} \in \mathbf{A}_2 \right\}$.
2. Вхідним алфавітом є вхідний алфавіт $\mathbf{Z}$ автоматів $\mathbf{S}_1$ та $\mathbf{S}_2$.
3. Вихідний алфавіт $\mathbf{W} = \varphi(\mathbf{W}_1 \times \mathbf{W}_2)$, де φ – задане відображення.
4. Функція переходів $\delta: \mathbf{A} \times \mathbf{Z} \to \mathbf{A}$ визначається наступним чином: $\delta(a_m, z_f) = (\delta_1(a_{m_1}, z_f), \delta_2(a_{m_2}, z_f))$, $z_f \in \mathbf{Z}$ або може бути записана у такий спосіб: $\delta(\mathbf{A} \times \mathbf{Z}) = (\delta_1(\mathbf{A}_1 \times \mathbf{Z}), \delta_2(\mathbf{A}_2 \times \mathbf{Z}))$.
5. Функція виходів $\lambda: \mathbf{A} \times \mathbf{Z} \to \mathbf{W}$ визначається наступним чином: $\lambda(a_m, z_f) = (\lambda_1(a_{m_1}, z_f), \lambda_2(a_{m_2}, z_f))$, $z_f \in \mathbf{Z}$ або може записана так: $\lambda(\mathbf{A} \times \mathbf{Z}) = \varphi(\lambda_1(\mathbf{A}_1 \times \mathbf{Z}), \lambda_2(\mathbf{A}_2 \times \mathbf{Z}))$.

Розглянемо приклад паралельного об'єднання двох автоматів $\mathbf{S}_1 = \{\mathbf{B}, \mathbf{Z}, \mathbf{U}, \delta_1, \lambda_1\}$, що поданий згідно таблиць 13.30 та 13.31, та $\mathbf{S}_2 = \{\mathbf{C}, \mathbf{Z}, \mathbf{V}, \delta_2, \lambda_2\}$, що представлений таблицями 13.32 та 13.33. Функція $\varphi: \mathbf{U} \times \mathbf{V} \to \mathbf{W}$ визначена в таблиці 13.34.

Таблиця 13.30. Таблиця функції переходів $\delta_1 : \mathbf{B} \times \mathbf{Z} \to \mathbf{B}$

	b_1	b_2	b_3
z_1	b_3	b_3	b_2
z_2	b_1	b_2	b_1

Таблиця 13.31. Таблиця функції виходів $\lambda_1 : \mathbf{B} \times \mathbf{Z} \to \mathbf{U}$

	b_1	b_2	b_3
z_1	u_2	u_1	u_1
z_2	u_1	u_2	u_1

Таблиця 13.32. Таблиця функції переходів $\delta_2 : \mathbf{C} \times \mathbf{Z} \to \mathbf{C}$

	c_1	c_2
z_1	c_1	c_2
z_2	c_2	c_1

Таблиця 13.33. Таблиця функції виходів $\lambda_2 : \mathbf{C} \times \mathbf{Z} \to \mathbf{V}$

	c_1	c_2
z_1	v_2	v_1
z_2	v_1	v_2

Таблиця 13.34. Таблиця функції $\varphi : \mathbf{U} \times \mathbf{V} \to \mathbf{W}$

	u_1	u_2
v_1	w_1	w_4
v_2	w_3	w_2

Результуючим автоматом такого об'єднання буде автомат $\mathbf{S} = \{\mathbf{A}, \mathbf{Z}, \mathbf{W}, \delta, \lambda\}$, у якого:

1. $\mathbf{A} = \mathbf{B} \times \mathbf{C} = \{(b_1, c_1), (b_1, c_2), (b_2, c_1), (b_2, c_2), (b_3, c_1), (b_3, c_2)\} =$
$= \{a_1, a_2, a_3, a_4, a_5, a_6\}.$
2. $\mathbf{Z} = \{z_1, z_2\}.$
3. $\mathbf{W} = \{w_1, w_2, w_3, w_4\}.$
4. Функція переходів $\delta : \mathbf{A} \times \mathbf{Z} \to \mathbf{A}$ визначена в таблиці 13.35. Тут, наприклад, $\delta(a_4, z_2) = \delta((b_2, c_2), z_2) = (\delta_1(b_2, z_2), \delta_2(c_2, z_2)) = (b_2, c_1) = a_3$.
5. Функція виходів $\lambda : \mathbf{A} \times \mathbf{Z} \to \mathbf{W}$ визначена в таблиці 13.36. Тут, наприклад, $\lambda(a_4, z_2) = \lambda((b_2, c_2), z_2) = \varphi(\lambda_1(b_2, z_2), \lambda_2(c_2, z_2)) = \varphi(u_2, v_2) = w_2$.

Таблиця 13.35. Таблиця функції переходів $\delta : \mathbf{A} \times \mathbf{Z} \to \mathbf{A}$

	a_1	a_2	a_3	a_4	a_5	a_6
	b_1c_1	b_1c_2	b_2c_1	b_2c_2	b_3c_1	b_3c_2
z_1	b_3c_1	b_3c_2	b_3c_1	b_3c_2	b_2c_1	b_2c_2
z_2	b_1c_2	b_1c_1	b_2c_2	b_2c_1	b_1c_2	b_1c_1

Таблиця 13.36. Таблиця функції виходів $\lambda : \mathbf{A} \times \mathbf{Z} \to \mathbf{W}$

	a_1	a_2	a_3	a_4	a_5	a_6
	b_1c_1	b_1c_2	b_2c_1	b_2c_2	b_3c_1	b_3c_2
z_1	w_2	w_4	w_3	w_1	w_3	w_1
z_2	w_1	w_3	w_4	w_2	w_1	w_3

Послідовне об'єднання двох автоматів $\mathbf{S}_1 = \{\mathbf{A}_1, \mathbf{Z}, \mathbf{W}_1, \delta_1, \lambda_1\}$ та $\mathbf{S}_2 = \{\mathbf{A}_2, \mathbf{W}_1, \mathbf{W}_2, \delta_2, \lambda_2\}$ подано на рисунку 13.9. Вихідний алфавіт автомата $\mathbf{S}_1$ є вхідним алфавітом автомата $\mathbf{S}_2$.

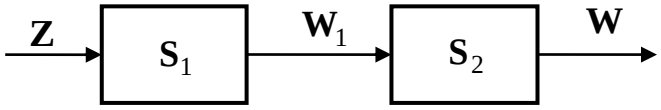


Рисунок 13.9. Послідовне об'єднання двох автоматів

Результуючим автоматом послідовно об'єднаних двох автоматів $\mathbf{S}_1$ та $\mathbf{S}_2$ будемо називати автомат $\mathbf{S} = \{\mathbf{A}, \mathbf{Z}, \mathbf{W}, \delta, \lambda\}$, у якого:

1. $\mathbf{A} = \mathbf{A}_1 \times \mathbf{A}_2$, інакше $\mathbf{A} = \left\{ a_m = (a_{m_1}, a_{m_2}) \middle| a_{m_1} \in \mathbf{A}_1, a_{m_2} \in \mathbf{A}_2 \right\}$.
2. $\mathbf{Z} = \mathbf{Z}$.
3. $\mathbf{W} = \mathbf{W}$.
4. Функція переходів $\delta : \mathbf{A} \times \mathbf{Z} \to \mathbf{A}$ визначається як:

$$\delta(a_m, z_f) = (\delta_1(a_{m_1}, z_f), \delta_2(a_{m_2}, \lambda_1(a_{m_1}, z_f))),$$

інакше

$$\delta(\mathbf{A} \times \mathbf{Z}) = (\delta_1(\mathbf{A}_1 \times \mathbf{Z}), \delta_2(\mathbf{A}_2 \times \lambda_1(\mathbf{A}_1 \times \mathbf{Z}))).$$

5. Функція виходів $\lambda : \mathbf{A} \times \mathbf{Z} \to \mathbf{W}$ визначається як:

$$\lambda(a_m, z_f) = \lambda_2(a_{m_2}, \lambda_1(a_{m_1}, z_f)),$$

інакше

$$\lambda(\mathbf{A} \times \mathbf{Z}) = \lambda_2(\mathbf{A}_2 \times \lambda_1(\mathbf{A}_1 \times \mathbf{Z})).$$

Розглянемо приклад послідовного об'єднання двох автоматів $\mathbf{S}_1 = \{\mathbf{B}, \mathbf{Z}, \mathbf{U}, \delta_1, \lambda_1\}$, що поданий згідно таблиць 13.30 та 13.31, та $\mathbf{S}_2 = \{\mathbf{C}, \mathbf{U}, \mathbf{V}, \delta_2, \lambda_2\}$, робота якого описується таблицями 13.37 та 13.38.

Результуючим автоматом такого об'єднання буде автомат $\mathbf{S} = \{\mathbf{A}, \mathbf{Z}, \mathbf{W}, \delta, \lambda\}$, у якого:

1. $\mathbf{A} = \mathbf{B} \times \mathbf{C} = \{(b_1, c_1), (b_1, c_2), (b_2, c_1), (b_2, c_2), (b_3, c_1), (b_3, c_2)\} =$
$= \{a_1, a_2, a_3, a_4, a_5, a_6\}$.
2. $\mathbf{Z} = \{z_1, z_2\}$.
3. $\mathbf{W} = \{w_1, w_2\}$.
4. Функція переходів $\delta : \mathbf{A} \times \mathbf{Z} \to \mathbf{A}$ визначена в таблиці 13.39. Тут, наприклад, $\delta(a_4, z_2) = \delta((b_2, c_2), z_2) = (\delta_1(b_2, z_2), \delta_2(c_2, \lambda_1(b_2, z_2))) =$
$= (b_2, \delta_2(c_2, u_2)) = (b_2, c_1) = a_3$.
5. Функція виходів $\lambda : \mathbf{A} \times \mathbf{Z} \to \mathbf{W}$ визначена в таблиці 13.40. Тут, наприклад, $\lambda(a_4, z_2) = \lambda((b_2, c_2), z_2) = \lambda_2(c_2, \lambda_1(b_2, z_2)) = \lambda_2(c_2, u_2) = w_1$.

Таблиця 13.37. Таблиця функції переходів $\delta_2 : \mathbf{C} \times \mathbf{U} \to \mathbf{C}$

	c_1	c_2
u_1	c_1	c_2
u_2	c_2	c_1

Таблиця 13.38. Таблиця функції виходів $\lambda_2 : \mathbf{C} \times \mathbf{U} \to \mathbf{W}$

	c_1	c_2
u_1	w_1	w_2
u_2	w_2	w_1

Таблиця 13.39. Таблиця функції переходів $\delta : \mathbf{A} \times \mathbf{Z} \to \mathbf{A}$

	a_1	a_2	a_3	a_4	a_5	a_6
	b_1c_1	b_1c_2	b_2c_1	b_2c_2	b_3c_1	b_3c_2
z_1	b_3c_2	b_3c_1	b_3c_1	b_3c_2	b_2c_1	b_2c_2
z_2	b_1c_1	b_1c_2	b_2c_2	b_2c_1	b_1c_1	b_1c_2

Таблиця 13.40. Таблиця функції виходів $\lambda : \mathbf{A} \times \mathbf{Z} \to \mathbf{W}$

	a_1	a_2	a_3	a_4	a_5	a_6
	b_1c_1	b_1c_2	b_2c_1	b_2c_2	b_3c_1	b_3c_2
z_1	w_2	w_1	w_1	w_2	w_1	w_2
z_2	w_1	w_2	w_2	w_1	w_1	w_2

Об'єднання із зворотнім зв'язком двох автоматів $\mathbf{S}_1 = \{\mathbf{A}_1, \mathbf{Z}_1, \mathbf{W}_1, \delta_1, \lambda_1\}$ та $\mathbf{S}_2 = \{\mathbf{A}_2, \mathbf{Z}_2, \mathbf{W}_2, \delta_2, \lambda_2\}$ подано на рисунку 13.10. функціональний перетворювач γ (автомат без пам'яті) реалізує відображення $\gamma : \mathbf{Z} \times \mathbf{W}_2 \to \mathbf{Z}_1$.

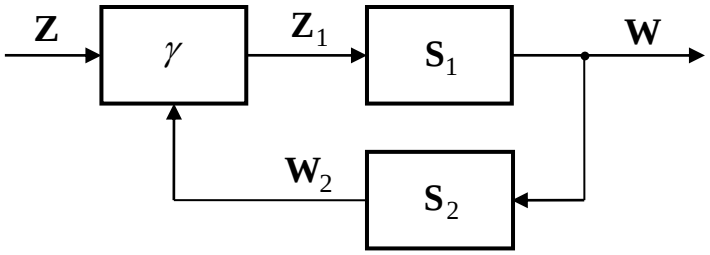


Рисунок 13.10. Об'єднання двох автоматів із зворотнім зв'язком

Покажемо, що у випадку об'єднання із зворотнім зв'язком хоча б один із автоматів ($\mathbf{S}_1$ або $\mathbf{S}_2$) повинен бути автоматом Мура. Доводити будемо від протилежного, тобто що $\mathbf{S}_1$ та $\mathbf{S}_2$ є автоматами Мілі. Тоді $\mathbf{W} = \lambda_2(\mathbf{A}_2 \times \mathbf{W})$, $\mathbf{W} = \lambda_1(\mathbf{A}_1 \times \mathbf{Z}_1)$.

Вхідний сигнал першого автомата буде визначатися згідно виразу:

$$\mathbf{Z}_1 = \gamma(\mathbf{Z} \times \mathbf{W}_2) = \gamma(\mathbf{Z} \times \lambda_2(\mathbf{A}_2 \times \mathbf{W})) = \gamma(\mathbf{Z} \times \lambda_2(\mathbf{A}_2 \times \lambda_1(\mathbf{A}_1 \times \mathbf{Z}_1))).$$

Таким чином, виявилося, що $\mathbf{Z}_1$ в даний момент часу залежить від $\mathbf{Z}_1$ у той самий момент часу, що призведе до нестабільності системи при такому

об’єднанні. Якщо ж один з автоматів, наприклад $\mathbf{S}_1$ є автоматом Мура, то $\mathbf{W} = \lambda_1(\mathbf{A}_1)$ та

$$\mathbf{Z}_1 = \gamma(\mathbf{Z} \times \mathbf{W}_2) = \gamma(\mathbf{Z} \times \lambda_2(\mathbf{A}_2 \times \mathbf{W})) = \gamma(\mathbf{Z} \times \lambda_2(\mathbf{A}_2 \times \lambda_1(\mathbf{A}_1))).$$

У цьому випадку стабільність системи не порушується.

Нехай автомат $\mathbf{S}_2$ є автоматом Мура ($\mathbf{W}_2 = \lambda_2(\mathbf{A}_2)$). Результуючим автоматом при об’єднанні із зворотнім двох автоматів $\mathbf{S}_1$ та $\mathbf{S}_2$ буде автомат $\mathbf{S} = \{\mathbf{A}, \mathbf{Z}, \mathbf{W}, \delta, \lambda\}$, у якого:

1. $\mathbf{A} = \mathbf{A}_1 \times \mathbf{A}_2$, інакше $\mathbf{A} = \{a_m = (a_{m_1}, a_{m_2}) \mid a_{m_1} \in \mathbf{A}_1, a_{m_2} \in \mathbf{A}_2\}$.
2. $\mathbf{Z} = \mathbf{Z}$.
3. $\mathbf{W} = \mathbf{W}$.
4. Функція переходів $\delta : \mathbf{A} \times \mathbf{Z} \to \mathbf{A}$ визначається як:

$$\delta(a_m, z_f) = (\delta_1(a_{m_1}, \gamma(z_f, \lambda_2(a_{m_2}))), \delta_2(a_{m_2}, \lambda_1(a_{m_1}, \gamma(z_f, \lambda_2(a_{m_2}))))),$$

інакше

$$\delta(\mathbf{A} \times \mathbf{Z}) = (\delta_1(\mathbf{A}_1 \times \gamma(\mathbf{Z} \times \lambda_2(\mathbf{A}_2))), \delta_2(\mathbf{A}_2 \times \lambda_1(\mathbf{A}_1 \times \gamma(\mathbf{Z} \times \lambda_2(\mathbf{A}_2))))).$$

5. Функція виходів $\lambda : \mathbf{A} \times \mathbf{Z} \to \mathbf{W}$ визначається як:

$$\lambda(a_m, z_f) = \lambda_1(a_{m_1}, \gamma(z_f, \lambda_2(a_{m_2}))),$$

інакше

$$\lambda(\mathbf{A} \times \mathbf{Z}) = \lambda_1(\mathbf{A}_1 \times \gamma(\mathbf{Z} \times \lambda_2(\mathbf{A}_2))).$$

Розглянемо приклад об’єднання із зворотнім зв’язком (рисунок 13.11) автомата $\mathbf{S}_1 = \{\mathbf{B}, \mathbf{R}, \mathbf{W}, \delta_1, \lambda_1\}$, робота якого задається таблицями 13.41 та 13.42, та автомата $\mathbf{S}_2 = \{\mathbf{C}, \mathbf{W}, \mathbf{V}, \delta_2, \lambda_2\}$, що заданий таблицею 13.43. Функція $\gamma : \mathbf{Z} \times \mathbf{V} \to \mathbf{R}$ визначена в таблиці 13.44.

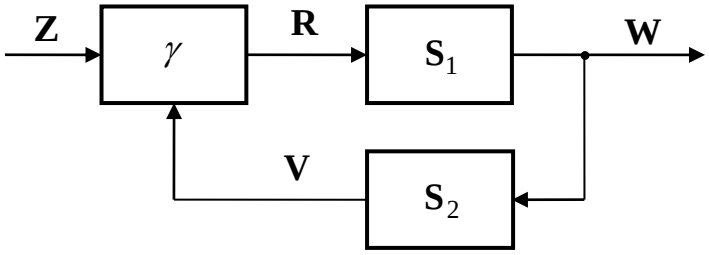


Рисунок 13.11. Приклад об’єднання двох автоматів із зворотнім зв’язком

Результуючим автоматом такого об’єднання буде автомат $\mathbf{S} = \{\mathbf{A}, \mathbf{Z}, \mathbf{W}, \delta, \lambda\}$, у якого:

1. $\mathbf{A} = \mathbf{B} \times \mathbf{C} = \{(b_1, c_1), (b_1, c_2), (b_2, c_1), (b_2, c_2), (b_3, c_1), (b_3, c_2)\} =$
   $= \{a_1, a_2, a_3, a_4, a_5, a_6\}$.
2. $\mathbf{Z} = \{z_1, z_2, z_3\}$.
3. $\mathbf{W} = \{w_1, w_2, w_3\}$.

4. Функція переходів $\delta : \mathbf{A} \times \mathbf{Z} \to \mathbf{A}$ визначена в таблиці 13.45. Тут, наприклад,

$\delta(a_1, z_1) = \delta((b_1, c_1), z_1) = (\delta_1(b_1, \gamma(z_1, \lambda_2(c_1))), \delta_2(c_1, \lambda_1(b_1, \gamma(z_1, \lambda_2(c_1))))) =$

$= (\delta_1(b_1, \gamma(z_1, v_1)), \delta_2(c_1, \lambda_1(b_1, \gamma(z_1, v_1)))) = (\delta_1(b_1, r_1), \delta_2(c_1, \lambda_1(b_1, r_1))) =$

$= (b_2, \delta_2(c_1, w_1)) = (b_2, c_2) = a_4.$

6. Функція виходів $\lambda : \mathbf{A} \times \mathbf{Z} \to \mathbf{W}$ визначена в таблиці 13.46. Тут, наприклад,

$\lambda(a_1, z_1) = \lambda((b_1, c_1), z_1) = \lambda_1(b_1, \gamma(z_1, \lambda_2(c_1))) = \lambda_1(b_1, \gamma(z_1, v_2)) = \lambda_1(b_1, r_2) = w_2.$

Таблиця 13.41. Таблиця функції переходів $\delta_1 : \mathbf{B} \times \mathbf{R} \to \mathbf{B}$

	b_1	b_2	b_3
r_1	b_2	b_3	b_1
r_2	b_3	b_1	b_2

Таблиця 13.42. Таблиця функції виходів $\lambda_1 : \mathbf{B} \times \mathbf{R} \to \mathbf{W}$

	b_1	b_2	b_3
r_1	w_1	w_2	w_3
r_2	w_2	w_3	w_1

Таблиця 13.43. Таблиця функцій $\delta_2 : \mathbf{C} \times \mathbf{W} \to \mathbf{C}$ та $\lambda_2 : \mathbf{C} \to \mathbf{V}$

	v_1	v_2
	c_1	c_2
w_1	c_2	c_2
w_2	c_1	c_2
w_3	c_2	c_1

Таблиця 13.44. Таблиця функції $\gamma : \mathbf{Z} \times \mathbf{V} \to \mathbf{R}$

	z_1	z_2	z_3
v_1	r_1	r_2	r_2
v_2	r_2	r_1	r_2

Таблиця 13.45. Таблиця функції переходів $\delta : \mathbf{A} \times \mathbf{Z} \to \mathbf{A}$

	a_1	a_2	a_3	a_4	a_5	a_6
	b_1c_1	b_1c_2	b_2c_1	b_2c_2	b_3c_1	b_3c_2
z_1	b_2c_2	b_3c_2	b_3c_1	b_1c_1	b_1c_2	b_2c_2
z_2	b_3c_1	b_2c_2	b_1c_2	b_3c_2	b_2c_1	b_1c_1
z_3	b_3c_1	b_3c_2	b_1c_2	b_1c_1	b_2c_2	b_2c_2

Таблиця 13.46. Таблиця функції виходів $\lambda : \mathbf{A} \times \mathbf{Z} \to \mathbf{W}$

	a_1	a_2	a_3	a_4	a_5	a_6
	b_1c_1	b_1c_2	b_2c_1	b_2c_2	b_3c_1	b_3c_2
z_1	w_1	w_2	w_2	w_3	w_3	w_1
z_2	w_2	w_1	w_3	w_2	w_1	w_3
z_3	w_2	w_2	w_3	w_3	w_1	w_1

Контрольні запитання

1. Дайте означення абстрактного автомата. Наведіть методи задавання абстрактних автоматів.
2. Охарактеризуйте абстрактні автомати Мілі та Мура, *C*-автомат.
3. Опишіть принцип переходу від автомата Мілі до автомата Мура та навпаки.
4. Наведіть етапи мінімізації внутрішніх станів цілком визначених автоматів.
5. Охарактеризуйте принцип мінімізації внутрішніх станів частково визначених автоматів.
6. Дайте означення декомпозиції абстрактних автоматів. Наведіть та охарактеризуйте види декомпозицій.

ЛЕКЦІЯ 14

СТРУКТУРНИЙ СИНТЕЗ ЦИФРОВИХ АВТОМАТІВ

14.1 Основна задача теорії структурного синтезу автоматів

У теорії автоматів виділяють абстрактну теорію автоматів і структурну теорію автоматів. Головна відмінність структурної теорії автоматів полягає в тому, що на відміну від абстрактної теорії вона враховує структуру вхідних і вихідних сигналів автомата, а також його внутрішню структуру на рівні так званих структурних схем, що забезпечують задане перетворення дискретної інформації. Основним завданням структурної теорії автоматів є вивчення композиції автоматів, тобто методів побудови складних автоматів із простіших автоматів.

Точніше ця задача може бути сформульована таким чином.

Нехай задана деяка скінченна множина автоматів в одному й тому ж структурному алфавіті Z. Назвемо ці автомати елементарними автоматами та припустимо, що таких елементарних автоматів є необмежена кількість екземплярів. Нехай також заданий довільний скінченний автомат A в тому ж самому структурному алфавіті Z. Необхідно знайти алгоритм, що дає змогу за заданим автоматом A побудувати деяку композицію елементарних автоматів так, щоб отриманий у результаті композиції автомат індукував відображення, яке тотожне відображенню, що індукується автоматом A.

Слід зазначити, що далеко не за будь-якого вибору системи елементарних автоматів ця задача має розв'язок. Однак, якщо розв'язок є (для довільного скінченного автомата A), то вважають, що задана система елементарних автоматів структурно повна або може мати ослаблену структурну повноту.

Система елементарних автоматів *ослаблено структурно повна*, якщо для будь-якого відображення φ, яке індукується скінченним автоматом, можна знайти таку композицію елементарних автоматів, що отриманий у результаті цієї композиції автомат індукує відображення, яке продовжує або саме відображення φ, або відображення, отримане з відображення φ у результаті застосування до нього операції вирівнювання довжин слів. При цьому передбачається, що автомат, який індукує відображення φ, заданий у тому ж структурному алфавіті, що й всі елементарні автомати. Зауважимо також, що операція вирівнювання довжин слів, що застосована до відображення φ, яке є автоматним відображенням, полягає в дописуванні рівної кількості порожніх символів справа до вхідних слів та зліва до вихідних слів відображення φ.

Насьогодні майже немає ефективних методів (істотно простіших, ніж метод перебору всіх варіантів) розв'язання основної задачі структурного синтезу за довільного вибору структурно повних систем елементарних автоматів. Найчастіше застосовують так званий канонічний метод структурного

синтезу автоматів, який коректно використовується для структурно повних систем елементарних автоматів деякого спеціального виду.

Канонічний метод структурного синтезу оперує з елементарними автоматами, які поділяють на два великі класи. Перший клас складають елементарні автомати з пам'яттю (тобто автомати, що мають більше одного внутрішнього стану); такі автомати називаються елементами пам'яті або запам'ятовуючими елементами. Другий клас складають автомати без пам'яті (тобто автомати з одним внутрішнім станом), які прийнято називати комбінаційними, або логічними елементами.

Суть канонічного методу структурного синтезу автоматів полягає в тому, щоб звести завдання структурного синтезу довільних автоматів до завдання структурного синтезу КС (автоматів без пам'яті). При цьому спеціальним чином робиться вибір елементів, що запам'ятовують. Як запам'ятовуючі елементи вибираються автомати Мура, які мають повну систему переходів та повну систему виходів.

Повнота системи переходів в автоматі Мура означає, що для будь-якої впорядкованої пари станів цього автомата знайдеться вхідний символ, що переводить перший елемент цієї пари в другий. Інакше кажучи, якщо $\delta(a,z)$ – функція переходів автомата, то для повноти системи переходів у цьому автоматі необхідно та достатньо, щоб для будь-якої пари (a_i, a_j) його станів рівняння $a_j = \delta(a_i, z)$ мало б розв'язок відносно вхідного сигналу z.

Повнота системи виходів в автоматі Мура означає, що кожному стану автомата відповідає свій власний вихідний символ, який відмінний від вихідного символу, що відповідає будь-якому іншому стану. Інакше кажучи, для повноти системи виходів автомата Мура необхідно й достатньо, щоб його зсунута функція виходів $w = \lambda(a)$ здійснювала взаємно однозначне відображення множини станів автомата на множину всіх його вихідних символів.

У автоматі Мура з повною системою виходів можна ототожнити внутрішній стан з вихідними символами автомата. Тоді один і той самий (структурний) алфавіт застосовується не лише для позначення (кодування) вхідних та вихідних символів, але й для кодування внутрішніх станів цього автомата.

Для автоматів із повною системою переходів доцільно ввести наступне означення.

Функцією входів автомата A з повною системою переходів та заданою функцією переходів $\delta(a,z)$ називається функція $z = \mu(a_i, a_j)$ (зазвичай неоднозначна), задана на впорядкованих парах станів автомата *A*. Для кожної такої пари (a_i, a_j) як значення функції входів $\mu(a_i, a_j)$ вибирається не порожня множина всіх тих вхідних сигналів z, для яких $\delta(a_i, z) = a_j$.

Функції входів скінченних автоматів задаються таблицями входів. У таблиці входів на перетині a_i-го рядка a_j-го стовпця вміщається множина вхідних символів, що викликають перехід автомата зі стану a_i в стан a_j.

Таким чином, у реальних цифрових автоматах, як правило, використовуються запам'ятовуючі елементи, що є автоматами Мура з повною системою переходів та з повною системою виходів. При цьому обмеження, що вводяться канонічним методом структурного синтезу, з практичної точки зору є несуттєвими.

14.2 Теорема про структурну повноту

Розглянемо справедливість наступного твердження, яке називають *теоремою про структурну повноту*.

Будь-яка система елементарних автоматів вважається структурно-повною системою, якщо в цій системі є автомат Мура з нетривіальною пам'яттю, який має повну систему переходів й повну систему виходів та будь-яку функціонально повну систему логічних елементів. Існує загальний конструктивний прийом (канонічний метод структурного синтезу), що дає змогу в даному випадку звести завдання структурного синтезу довільних скінченних автоматів до завдання структурного синтезу КС.

Для доведення цієї теореми синтезуємо довільний скінченний автомат A. Виберемо запам'ятовуючі елементи $A_1,\ldots,A_R$, які мають повні системи переходів та виходів, причому такі, що добуток кількості їх станів не менший, ніж кількість станів автомата A. Кожному стану a автомата A поставимо у відповідність скінченну послідовність $(a_1,\ldots,a_r)$ станів автоматів $A_1,\ldots,A_R$ так, що різним станам автомата A ставляться у відповідність різні послідовності. Назвемо цей процес кодуванням станів автомата A.

Після кодування станів (виконуваного довільним чином) виникають структурні стани автомата A.

Враховуючи те, що в автоматі Мура з повною системою виходів можна ототожнити внутрішній стан із вихідними сигналами автомата, вважаємо ці структурні стани векторами в структурному алфавіті; при цьому компонента a_i в структурному стані $(a_1,\ldots,a_r)$ замінюється груповою компонентою структурного вихідного сигналу Q_i запам'ятовуючого елемента A, що відповідає стану a_i.

Позначимо структурний стан $(Q_1,\ldots,Q_p)$ через Q. Структурний стан Q можна розглядати ще як структурний вихідний сигнал об'єднання автоматів $A_1,\ldots,A_R$. Через U позначимо структурний вхідний сигнал цього об'єднання, який назвемо структурним вхідним сигналом пам'яті автомата A.

Кодування станів автомата A дає змогу функцію переходів та закон функціонування автомата подати у векторному вигляді

$$\mathbf{Q}(t+1)=\delta(\mathbf{Q}(t),\mathbf{x}(t)). \qquad (14.1)$$

Перехід автомата A зі стану $\mathbf{Q}(t)$ в стан $\mathbf{Q}(t+1)$ складається з відповідних переходів автоматів $A_1,\ldots,A_R$. Кожний наступний перехід відбувається під дією структурного сигналу $\mathbf{U}_i(t)$, що подається на вхід автомата A_i та визначається за допомогою функції входів μ_i цього автомата

$(i=1,\ldots,r)$. Структурні сигнали $\mathbf{U}_i(t)$ об'єднуються в структурний вхідний сигнал $\mathbf{U}(t)$ пам'яті автомата A, а з функцій μ_i будується об'єднуюча їх (що містить їх як свої компоненти) векторна функція

$$\mu(\mathbf{Q}(t),\mathbf{Q}(t+1))=\mathbf{U}(t). \qquad (14.2)$$

Підставляючи у формулу (8.2) значення $\mathbf{Q}(t+1)$ з формули (14.1), отримаємо

$$\mathbf{Q}(t)=\mu(\mathbf{Q}(t),\delta(\mathbf{Q}(t),\mathbf{x}(t)))=\zeta(\mathbf{Q}(t),\mathbf{x}(t)). \qquad (14.3)$$

Векторна функція $\zeta(\mathbf{Q},\mathbf{x})$, що визначається формулою (14.3), називається (векторною) *функцією збудження* автомата A або *функцією входів його пам'яті*. Вона визначає, який структурний вхідний сигнал $\mathbf{U}(t)$ у будь-який момент часу t має бути поданий на елементи пам'яті автомата A, що знаходиться в стані $\mathbf{Q}(t)$, якщо на зовнішні вхідні вузли автомата подається в той самий момент часу структурний вхідний сигнал $\mathbf{x}(t)$.

Завдяки збігу усіх розглянутих сигналів у часі, сигнал $\mathbf{U}(t)$ можна розглядати як структурний вихідний сигнал деякої КС, який відрізняється тим, що її структурний вхідний сигнал формується в результаті об'єднання структурного вхідного сигналу $\mathbf{x}(t)$ автомата A та структурного вихідного сигналу $\mathbf{Q}(t)$ його пам'яті. Реалізовуючи цю схему у вигляді композиції заданих логічних елементів, очевидні всі ті переходи, які передбачаються функцією переходів автомата A.

Побудована КС називається *схемою зворотних зв'язків* автомата A, а рівняння

$$\mathbf{U}=\zeta(\mathbf{Q},\mathbf{x}),$$

яке визначає векторну функцію виходів, що реалізовується цією схемою, – канонічним (векторним) рівнянням автомата A. При переході до компонентів, це рівняння розпадається на *систему канонічних рівнянь* автомата A.

Зауважимо, що функція збудження $\zeta(\mathbf{Q},\mathbf{x})$ автомата є неоднозначною функцією, оскільки неоднозначними, в загальному випадку, є функції входів μ_i запам'ятовуючих елементів $A_i(i=1,\ldots,R)$. Неоднозначністю цієї функції можна скористатися для отримання якомога простішої схеми зворотних зв'язків в автоматі.

Значні можливості спрощення схем зворотних зв'язків і схем виходів пов'язані з вибором раціонального способу кодування станів автомата, що синтезується. У виборі раціонального, з вказаної точки зору, кодування станів автомата полягає так звана проблема кодування, яка є однією з складних проблем структурної теорії автоматів. У наш час вирішення цієї проблеми у кожному конкретному випадку пов'язане, як правило, з перебором великої кількості різних варіантів кодування станів.

14.3 Синтез структурної схеми автомата

При побудові схеми зворотних зв'язків необхідно враховувати те, що в кожен момент часу t відповідно до закону функціонування автомата A

забезпечується утворення структурного вихідного сигналу $\mathbf{y}(t)$. Залежно від того, чи є заданий автомат A автоматом Мілі або Мура, утворення його вихідного сигналу визначатиметься або законом $\mathbf{y}(t)=\lambda(\mathbf{Q}(t),\mathbf{x}(t))$, або законом $\mathbf{y}(t)=\lambda(\mathbf{Q}(t))$. І у першому, й у другому випадках необхідний структурний вихідний сигнал може бути забезпечений КС, яку також називають *схемою виходів* початкового автомата A. Схема, побудована на автоматах Мілі, реалізує векторну функцію $\mathbf{y}(t)=\lambda(\mathbf{Q},\mathbf{x})$, а на автоматах Мура – векторну функцію $\mathbf{y}(t)=\lambda(\mathbf{Q})$. В обох випадках схема виходів автомата може бути отримана в результаті композиції заданих логічних елементів тільки на основі припущення про функціональну повноту системи цих елементів.

При структурному синтезі автомата обидві побудовані КС (схема зворотних зв'язків та схема виходів) об'єднуються в одну загальну КС, яку також називають комбінаційною частиною автомата, а частина пам'яті автомата є об'єднанням усіх запам'ятовуючих елементів.

Структурна схема автомата, синтезованого відповідно до канонічного методу структурного синтезу, зображена на рисунку 14.1.

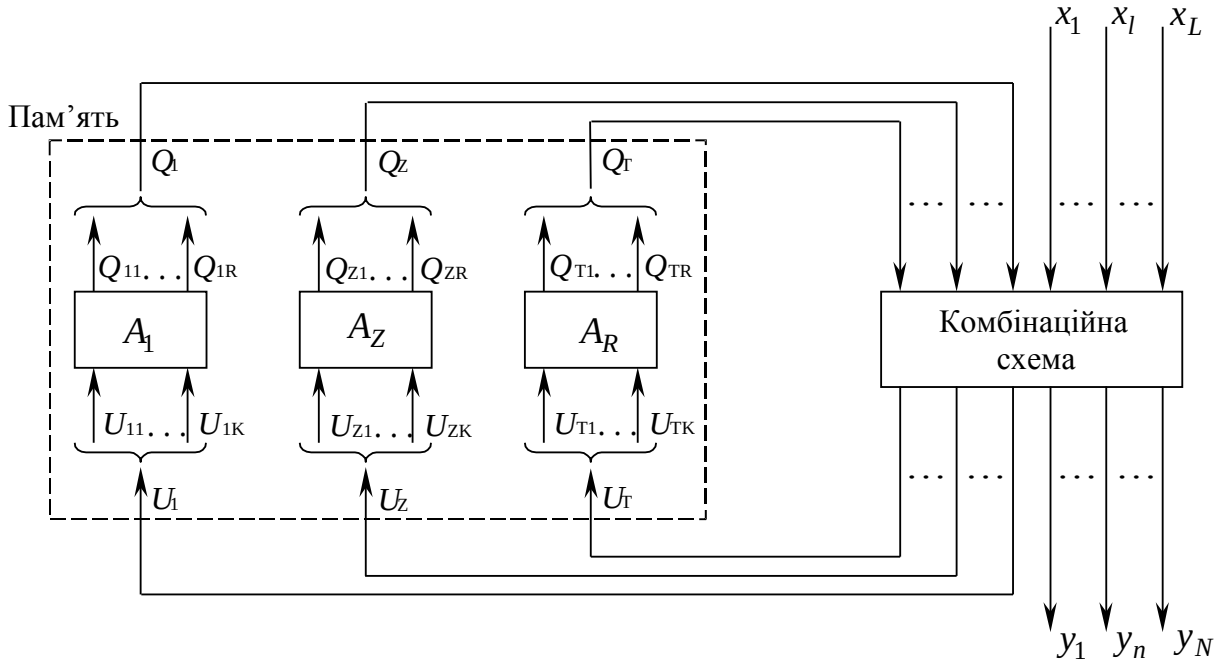


Рисунок 14.1. Структурна схема автомата, синтезованого у відповідності з канонічним методом структурного синтезу

Пам'ять складається з елементарних автоматів Мура $A_1,\ldots,A_z,\ldots,A_R$. Після вибору елементів пам'яті кожен стан синтезованого автомата A кодується набором їх станів. Якщо всі автомати $A_1,\ldots,A_R$ однакові, що в загальному випадку не є обов'язковим, то їх кількість задовольняє умову

$$R \geq \log_b M,$$

де M – кількість станів синтезованого автомата, а b – кількість станів елементарного автомата пам'яті. Звичайно для елементарного автомата $b=2$, тоді $R \geq \log_2 M$.

Переходи автоматів пам'яті, які відповідають переходам в автоматі A, відбуваються під дією сигналів збудження пам'яті, що надходять із виходу КС на вхід пам'яті автомата. На рисунку (14.1) $\mathbf{x} = \{x_1, \ldots, x_L\}$ та $\mathbf{y} = \{y_1, \ldots, y_N\}$ – векторні структурні вхідні та вихідні сигнали автомата, $\mathbf{U} = \{U_1, \ldots, U_T\}$ – векторна функція збудження пам'яті, $\mathbf{Q} = \{Q_1, \ldots, Q_T\}$ – вектор вихідного сигналу зворотного зв'язку від елементів пам'яті автомата.

Очевидно, що структурна схема (рисунок 14.1) буде коректно побудованою або правильною, якщо коректно побудована або відповідно правильна його комбінаційна частина.

Таким чином, канонічний метод синтезу довільних скінченних автоматів повністю зводиться до синтезу автоматів без пам'яті. Результатом канонічного методу синтезу є система логічних рівнянь, яка виражає залежність вихідних сигналів автомата та сигналів, що подаються на входи запам'ятовуючих елементів, від сигналів, які надходять на вхід всього автомата вцілому, та сигналів, які знімаються з виходів елементів пам'яті.

14.4 Синхронізація в цифрових автоматах

Зміна станів у синхронних автоматах відбувається у визначені моменти часу, що задаються в ланцюгах синхронізації зовнішнім тактовим генератором. Зміна станів у реальних цифрових автоматах супроводжується перехідними процесами, що мають цілком визначені тривалості для вибраної елементної бази. Вихідні сигнали по ланцюгах зворотного зв'язку (рисунок 8.1) надходять на вхід цифрового автомата, а далі на вхід його блока пам'яті. Через невизначеності сигналів зворотного зв'язку під час протікання перехідних процесів невизначеними будуть й сигнали збудження блоку пам'яті. Тому передбачаються спеціальні заходи, що гарантують перехід автомата в необхідний стан. У синхронізованих цифрових автоматах стійкість автомата забезпечується спеціальними ланцюгами синхронізації, що розривають ланцюги зворотного зв'язку під час протікання перехідних процесів. Така синхронізація вимагає забезпечення деякого проміжку часу на протікання перехідних процесів й, відповідно, зменшує швидкодію цифрових автоматів та вимагає виконання визначених часових співвідношень при зміні значень вхідних сигналів.

Схема цифрового автомата, яка побудована з елементарних автоматів, є коректно побудованою, якщо в кожному її вузлі відсутня неоднозначність сигналів у будь-який суттєвий інтервал часу. Суттєвим інтервалом часу є інтервал, протягом якого інформація зчитується з будь-якого вузла цієї схеми. Коректна схема автомата Мура зображена на рисунку 8.2. Комбінаційна схема $КС_{вх}$ служить для формування коду, що відповідає новому стану автомата у відповідності з функцією переходів

$$a(t+1) = \delta(a(t), z(t)),$$

схема $КС_{вих}$ – для генерування сигналів, що відповідають функції виходів

$$w(t) = \lambda(a(t)).$$

Блок пам’яті складається з двох однакових блоків БП1 та БП2. Передача сигналів від блока до блока здійснюється через кон’юнктори, що керуються тактовими сигналами τ_1 та τ_2. Код, що відповідає новому стану, формується $КС_{вх}$ та при появі імпульсу τ_1 встановлює в необхідний стан БП1. В інтервалі часу τ_1 БП2 зберігає код свого попереднього стану на входах КС1. Протягом часу, що виділений сірим відтінком (рисунок 14.3), зміна вхідних сигналів заборонена, тому невизначеність на виходах КС1 не виникає. Виконання обмежень на часові співвідношення при формуванні вхідних сигналів забезпечується в джерелі вхідних сигналів X. Імпульсом τ_2 інформація переноситься в БП2 та цим закінчується такт роботи цифрового автомата. На зміну стану цифрового автомата таким чином витрачається час T’. Оскільки у автомата Мура функції виходів не залежать від вхідних сигналів, то порядок їх зміни може бути довільним, тільки б на інтервалі часу τ_1 вхідні сигнали були незмінні.

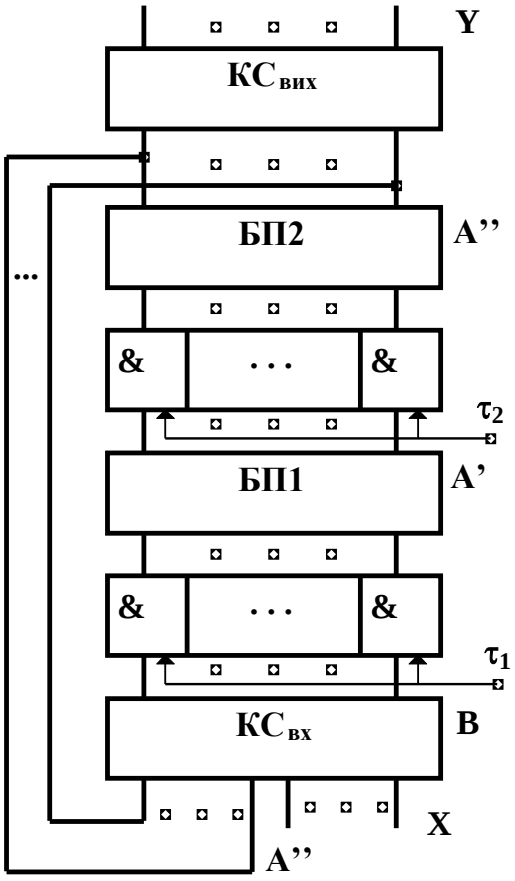


Рисунок 14.2. Схема автомата Мура з двотактною пам’яттю

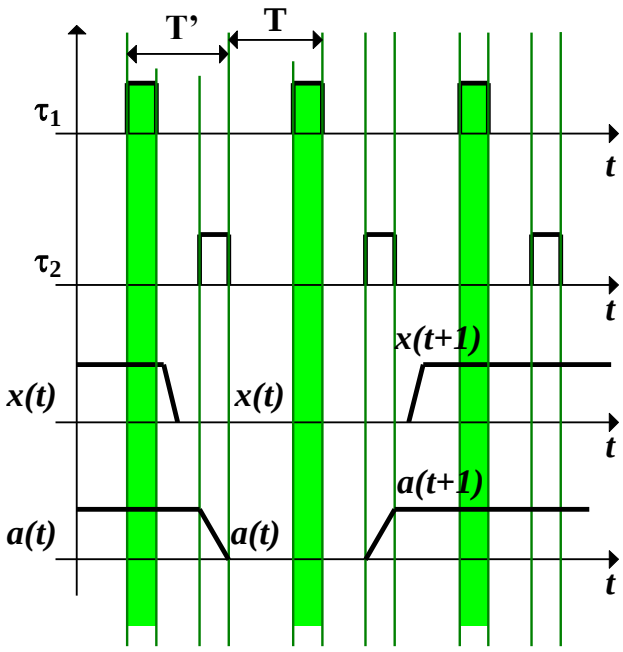


Рисунок 14.3. Часова діаграма цифрового автомата Мура з двотактною пам’яттю

Таким чином, коректність роботи цифрового автомата Мура забезпечується введенням двотактного синхронізованого блока пам’яті. Логіка роботи двотактної пам’яті практично реалізується в двоступеневих тригерах структури О-В (або M-S), що мають вбудовану двотактну систему синхронізації.

Двотактна синхронізована пам’ять забезпечує коректність й автомата Мілі. Але тільки одного цього недостатньо, оскільки одним із аргументів функцій виходів автомата Мілі є код вхідного сигналу $x(t)$, тобто

$$w(t) = \lambda(a(t), z(t)).$$

Для синхронного цифрового автомата є неприпустимим випадковий характер зміни коду вихідного сигналу у відповідності зі змінами коду вхідного сигналу, але накладати жорсткі обмеження на часові співвідношення у вхідному сигналі також недопустимо. Ця суперечність розв’язується шляхом введення стробування вихідної комбінаційної схеми $КС_{вих}$ синхросигналом τ_1. Вихідні сигнали цифрового автомата Мілі при цьому стають імпульсними. Для цифрового автомата Мура вихідні сигнали вважаються потенціальними, оскільки нема необхідності в стробуванні вихідної комбінаційної схеми.

Структурна схема коректного цифрового автомата Мілі та часова діаграма його роботи наведені на рисунках 14.4 та 14.5.

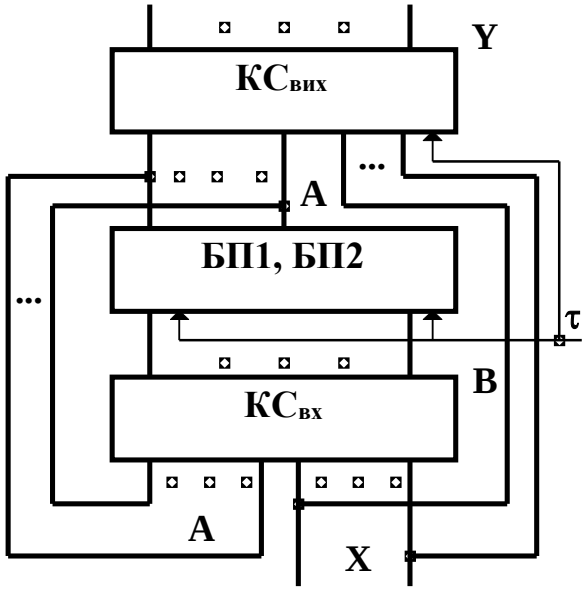


Рисунок 14.4. Структурна схема коректного цифрового автомата Мілі з двотактною пам’яттю

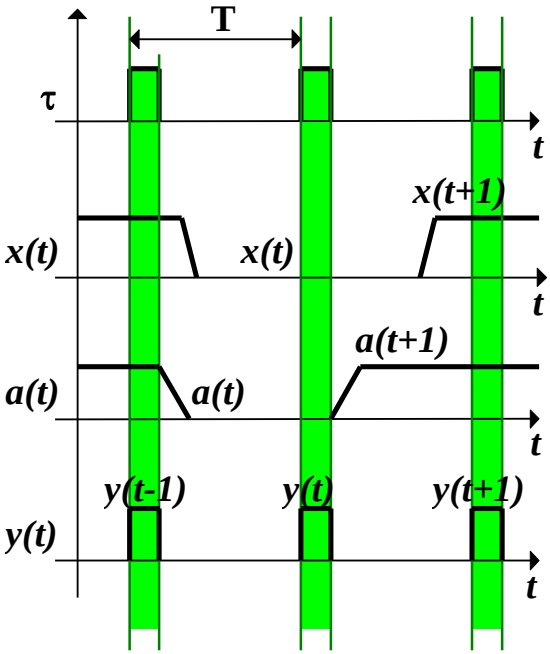


Рисунок 14.5. Часова діаграма цифрового автомата Мілі з двотактною пам’яттю

За часовою діаграмою рисунку 8.5 бачимо, що зміна станів блока пам’яті повинна відбуватися по спаду синхроімпульсу. У протилежному випадку в цифровому автоматі Мілі буде реалізована функція виходів, що описується рівнянням

$$w(t) = \lambda(a(t), z(t)).$$

14.5 Канонічний метод структурного синтезу цифрових автоматів

Алгоритм канонічного методу структурного синтезу цифрових автоматів полягає в наступному.

1. *Кодування станів абстрактного автомата*. У процесі структурного синтезу різним станам заданого абстрактного автомата a_i ставляться у відповідність різні впорядковані послідовності станів елементарних автоматів $Q_1, Q_2, \ldots, Q_p$. Цей процес називається кодуванням станів автомата.

Результатом кодування станів є виникнення структурних станів автомата $Q^l = Q_1^l, Q_2^l, \ldots, Q_R^l$, де $l = 0,1,2,\ldots, M(M+1)$ – кількість станів абстрактного автомата, $R = \log_2 M$. Ці стани можна ототожнювати зі структурними вихідними сигналами запам'ятовуючої частини автомата.

У зв'язку з тим, що кожен структурний вихідний сигнал пам'яті автомата є сукупністю елементарних сигналів, що надходять по окремих каналах, будемо вважати структурний вихідний сигнал пам'яті векторним сигналом $\mathbf{Q}^l$, в якому кожна компонента ототожнюється з буквою структурного алфавіту стану $Q = \{Q_1, Q_2, \ldots, Q_P\}$, тобто $Q_i^l \in Q$.

2. *Кодування абстрактних вхідних та вихідних сигналів*. Абстрактним вхідним та вихідним сигналам $z_i \in Z$ та $w_i \in W$, де Z та W – вхідні та вихідні абстрактні алфавіти, ставляться у відповідність зовнішні структурні вхідні та вихідні сигнали автомата, що позначаються відповідно $x^j = x_1^j x_2^j \ldots x_L^j$ та $y^k = y_1^k y_2^k \ldots y_N^k$, де $j = 1,2,\ldots, F$, F – кількість символів вхідного абстрактного алфавіту, $k = 1,2,\ldots, G$, G – кількість символів вихідного абстрактного алфавіту, $L = \log_2 F$ та $N = \log_2 G$. Сигнали $\mathbf{x}^j$ та $\mathbf{y}^k$ є векторними сигналами, компоненти яких x_i^j та y_i^k – відповідно елементарні вхідні та вихідні сигнали на кожному елементарному вхідному або вихідному каналі, тобто $x_i^j \in X = \{x_1, x_2, \ldots, x_n\}$, а $y_i^k \in Y = \{y_1, y_2, \ldots, y_r\}$, де X та Y – відповідно структурні вхідні та вихідні алфавіти.

3. *Складання кодованих таблиць переходів-виходів структурного автомата*. У процесі синтезу необхідно забезпечити, щоб переходи з однієї послідовності станів елементарних автоматів в іншу проходили у повній відповідності з функцією переходів заданого абстрактного автомата, а значення структурних вихідних сигналів формувалися відповідно до заданої послідовності абстрактних вхідних сигналів. Таким чином, повинні бути забезпечені відповідні закони функціонування для структурних автоматів Мура та Мілі.

Закон функціонування структурного автомата може бути описаний за допомогою кодованих таблиць переходів-виходів, які формуються на основі таблиць переходів-виходів абстрактного автомата та отриманих згідно з першими двома пунктами алгоритму структурних значень станів та сигналів автомата. У клітинках цих таблиць замість символів, що позначають абстрактні стани та сигнали, записуються коди відповідних до них структурних станів та сигналів.

4. *Формування таблиці функцій збудження структурного автомата*. Функції збудження задаються за допомогою таблиці, що сформована на базі

структурної таблиці переходів проектованого автомата, й таблиці переходів заданого елементарного автомата (тригера). У клітинках таблиці функцій збудження записуються значення структурних вхідних сигналів вибраних елементарних автоматів (тригерів), що забезпечують переходи їх станів відповідно до кодованої таблиці переходів.

5. *Отримання логічних виразів функцій збудження та вихідних сигналів автомата*. Для отримання логічних виразів функцій збудження та вихідних сигналів необхідно скористатися відповідно таблицею функцій збудження та кодованою таблицею виходів у ролі таблиць істинності. Записані за цими таблицями логічні вирази є ДДНФ, які необхідно мінімізувати, зокрема, за допомогою карт Карно.

6. *Побудова функціональної схеми автомата*. На основі мінімізованих логічних виразів будується схема структурного автомата із заданих елементарних автоматів (тригерів) та логічних елементів функціонально повного базису.

Контрольні запитання

1. Сформулюйте основну задачу теорії структурного синтезу цифрових автоматів.
2. Дайте означення поняттям «повнота системи переходів» та «повнота системи виходів» в автоматах Мура.
3. Сформулюйте теорему про структурну повноту.
4. Назвіть, з яких частин складається структурна схема автомата, синтезованого відповідно до канонічного методу структурного синтезу. Дайте характеристику цих частин.

ЛЕКЦІЯ 15

КОДУВАННЯ ВНУТРІШНІХ СТАНІВ ЦИФРОВИХ АВТОМАТІВ ТА ГОНКИ В АВТОМАТАХ

15.1 Основні поняття кодування внутрішніх станів абстрактних автоматів

Процес кодування станів абстрактних автоматів є першим етапом канонічного методу структурного синтезу автоматів.

Кодування полягає в зіставленні кожному стану автомата набору (коду) станів елементів пам'яті**.** При цьому набори для всіх станів повинні мати однакову довжину, а різним станам автомата повинні відповідати різні набори. Якщо елементи пам'яті двійкові, то їх кількість $R \geq \log_2 A$, де A– кількість станів автомата.

Кодування станів автомата можна здійснювати різними способами. Це може бути *довільне кодування*, коли кожному стану ставиться у відповідність довільний набір двійкових символів, кількість яких дорівнює R. Синтезований на основі такого кодування автомат не буде оптимальним, оскільки, по-перше, його КС може мати підвищену складність та, по-друге, за відсутності синхронізації й подвійної пам'яті в процесі функціонування цього автомата можуть виникнути так звані *гонки*.

Це явище виникає внаслідок того, що елементи пам'яті мають різні, хоча й достатньо близькі часи спрацювання. Різні також затримки сигналів збудження, що надходять на вхідні канали елементарних автоматів по логічних ланцюгах неоднакової довжини.

Якщо під час переходу автомата з одного стану в інший повинні змінити свої стани відразу кілька запам'ятовуючих елементів, то між ними починаються гонки. Той елемент, який виграє ці гонки, тобто змінить свій стан раніше, ніж інші елементи, може через ланцюг зворотного зв'язку змінити сигнали на входах деяких запам'ятовуючих елементів до того, як інші, що беруть участь у гонках елементи, змінять свої стани. Це може призвести до переходу автомата в стан, що не передбачений його графом (рисунок 15.1, суцільна лінія). Тому в процесі переходу зі стану a_m у стан a_s під дією вхідного сигналу z_f автомат може виявитися в стані a_k або a_l: (рисунок 15.1, пунктирна лінія). Якщо потім при тому ж вхідному сигналі z_f автомат з a_k або з a_l перейде в a_s, то такі *гонки* є *допустимими* або *некритичними*. Якщо ж у цьому автоматі є перехід, наприклад, з a_k в $a_j \neq a_s$ під дією того самого сигналу z_f, то автомат може перейти в a_j, а не в a_s та правильність його роботи буде порушена (рисунок 15.1, пунктирна лінія). Такі гонки називаються *критичними*, що вказує на необхідність вжиття заходів для їх усунення.

Усунути гонки можна апаратними засобами або використовуючи спеціальні методи кодування. Один зі способів ліквідації гонок полягає в

тактуванні вхідних сигналів автомата імпульсами певної тривалості. Передбачається, що окрім вхідних каналів $x_1,\ldots,x_l$ є ще канал C від генератора синхроімпульсів, по якому надходить сигнал $C=1$ у момент приходу імпульсу та $C=0$ за його відсутності. У зв'язку з цим, вхідним сигналом на переході $(a_m,\ a_s)$ буде не z_f, а Cz_f. Тоді, якщо тривалість імпульсу t_c менше найкоротшого шляху проходження тактового сигналу зворотного зв'язку по КС, то до моменту переходу в проміжний стан a_k сигнал $C=0,\ Cz_f=0$, що виключає гонки. Канал C – це фактично синхровхід тригера. Недолік методу – у складності підбору необхідної тривалості імпульсу.

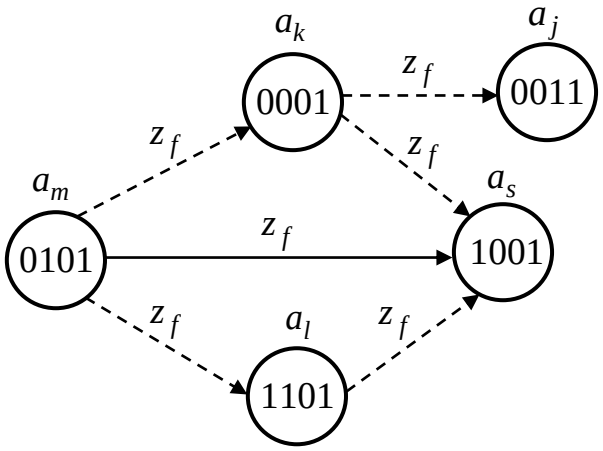


Рисунок 15.1. Гонки в автоматі

Інший метод ліквідації гонок полягає у введенні подвійної пам'яті. В цьому випадку кожен елемент пам'яті дублюється, причому перепис з першого елемента пам'яті в другий відбувається в момент $C=0$ (рисунок 15.2).

Сигнали зворотного зв'язку для отримання функцій збудження та функцій виходів автомата знімаються з виходу другого тригера. Таким чином, гонки можуть виникнути тільки між першими тригерами, сигнали на виходах других тригерів не можуть змінитися до тих пір, поки C не дорівнюватиме 0. Але тоді $Cz_f=0$, перший тригер перестане сприймати інформацію та гонок не буде.

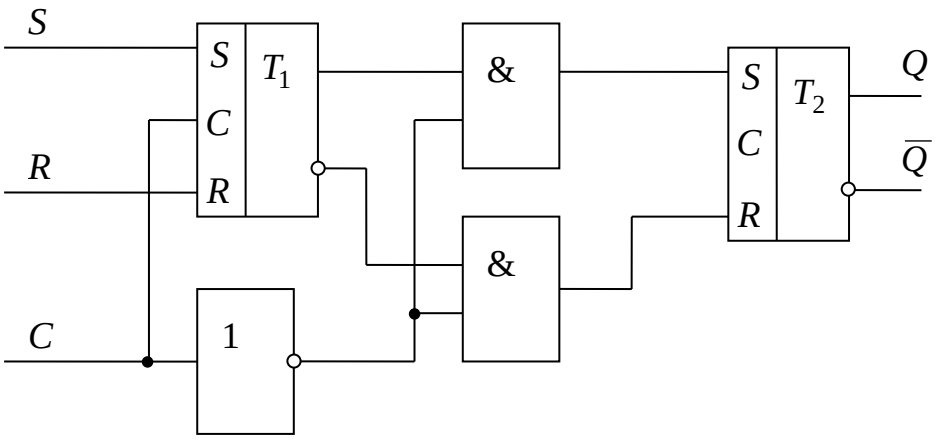


Рисунок 15.2. Елемент з подвійною пам'яттю

Для усунення гонок використовуються спеціальні методи протигоночного кодування, серед яких частіше за все застосовується так зване сусіднє кодування станів автомата, за якого умова відсутності гонок завжди виконана. При сусідньому кодуванні будь-які два стани, які зв'язані дугою на графі автомата, кодуються наборами, що відмінні лише на одну одиницю.

Сусіднє кодування не завжди можливе. Граф автомата, що допускає сусіднє кодування, повинен задовольняти ряд вимог, а саме:

1) у графі автомата не повинно бути циклів з непарною кількістю вершин;

2) два сусідні стани другого порядку не повинні мати більше двох станів, що лежать між ними.

Під станами другого порядку розуміють такі два стани, шлях між якими по графу автомата складається з двох ребер (незалежно від орієнтації). Приклади графів автоматів, що допускають та не допускають сусіднє кодування, зображено на рисунках 15.3. та 15.4 відповідно.

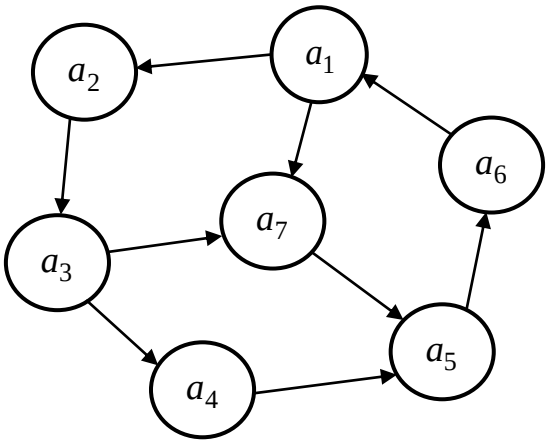


Рисунок 15.3. Граф, який допускає сусіднє кодування

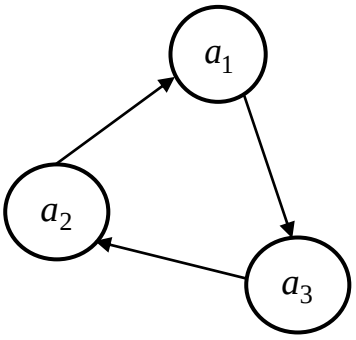


Рисунок 15.4. Граф, який не допускає сусіднє кодування

При сусідньому кодуванні переважно використовують карти Карно. У цьому випадку стани, які зв'язані дугою, розташовують у сусідніх клітинках карти. Наприклад, для графа, зображеного на рисунку 15.3, стани можна закодувати згідно рисунку 15.5.

	00	01	11	10
0	a_1	a_2	a_3	a_7
1	a_6		a_4	a_5

$a_1 = 000$ $a_2 = 010$ $a_3 = 110$ $a_4 = 111$

$a_5 = 101$ $a_6 = 001$ $a_7 = 100$

Рисунок 15.5. Карта Карно для сусіднього кодування

Таким чином, при сусідньому кодуванні на кожному переході перемикається лише один тригер, що принципово усуває гонки.

15.2 Кодування станів та складність комбінаційної схеми автомата

Аналіз канонічного методу структурного синтезу автомата показує, що різні варіанти кодування станів автомата призводять до різних виразів функцій збудження пам'яті та функцій виходів, внаслідок чого складність КС істотно залежить від вибраного кодування. Серед множини існуючих алгоритмів кодування найчастіше зустрічаються наступні:

1) алгоритм кодування абстрактних автоматів, побудованих на D-тригерах;

2) евристичний алгоритм кодування;

3) сусіднє кодування логічно суміжних станів.

15.2.1. Алгоритм кодування абстрактних автоматів, побудованих на D-тригерах

Даний алгорит передбачає наступне:

1. Кожному стану автомата a_m $(m = 1,2,\ldots,M)$ ставиться у відповідність ціле число N_m, яке дорівнює кількості переходів у стан a_m (N_m дорівнює кількості появ a_m у полі таблиці переходів або кількості дуг, що входять в a_m при графічному методі задавання автомата).

2. Числа $N_1, N_2, \ldots, N_m$ впорядковуються у бік зменшення.

3. Стан a_s з найбільшим N_s кодується послідовністю $\underbrace{00\ldots0}_{R}$, де R – кількість елементів пам'яті.

4. Наступні R станів згідно зі списком пункту 2 кодуються послідовністю, що містить тільки одну одиницю: --00-...-01,--00-...-10,...,-01-...-00, … .

5. Для станів, що залишилися, знову в порядку списку п.2. використовують коди з двома одиницями, потім з трьома й так далі, поки не будуть закодовані всі стани.

У результаті отримують код, для якого має місце така закономірність: чим більше є переходів у деякий стан, тим менше одиниць у його коді. Оскільки для D-тригерів функції збудження однозначно визначаються кодом стану переходу, то очевидно, що вирази для функцій збудження будуть простішими. Цей метод особливо ефективний за відсутності мінімізації функцій збудження, що має місце в реальних автоматах з великою кількістю внутрішніх станів та вхідних змінних.

Приклад 15.1. Для автомата, заданого своїми таблицями переходів та виходів (таблиці 15.4 та 15.5), закодувати внутрішні стани, вважаючи, що автомат побудований на базі D-тригерів.

Таблиця 15.4. Таблиця переходів автомата

	a_1	a_2	a_3	a_4	a_5
z_1	a_2	a_3	a_5	a_4	a_5
z_2	a_1	a_5	a_4	a_4	a_1
z_3	a_3	a_1	a_2	a_2	a_3
z_4	a_1	a_3	a_3	a_3	a_5

Таблиця 15.5. Таблиця виходів автомата

	a_1	a_2	a_3	a_4	a_5
z_1	w_1	w_2	w_1	w_1	w_1
z_2	w_1	w_3	w_4	w_2	w_2
z_3	w_2	w_2	w_2	w_1	w_3
z_4	w_2	w_2	w_2	w_1	w_3

Внутрішні стани можна закодувати так:

$a_1 = N_1 = 4$, $a_3 = N_3 = 000$,
$a_2 = N_2 = 3$, $a_1 = N_1 = 001$,
$a_3 = N_3 = 6$, $a_5 = N_5 = 010$,
$a_4 = N_4 = 3$, $a_2 = N_2 = 100$,
$a_5 = N_5 = 4$, $a_4 = N_4 = 011$.

Аналогічно кодуванню внутрішніх станів абстрактних автоматів, побудованих на D-тригерах, можна кодувати вихідні сигнали автоматів, побудованих на будь-яких типах тригерів, тобто чим частіше генерується даний вихідний сигнал w_i, тим менше одиниць в його коді. Для автомата (таблиці 15.4 та 15.5) отримаємо:

$w_1 = N_1 = 6$, $w_1 = N_1 = 00$,
$w_2 = N_2 = 5$, $w_2 = N_2 = 01$,
$w_3 = N_3 = 2$, $w_3 = N_3 = 10$,
$w_4 = N_4 = 2$, $w_4 = N_4 = 11$.

15.2.2 Евристичний алгоритм кодування внутрішніх станів абстрактних автоматів

Даний алгоритм мінімізує сумарну кількість перемикань елементів пам'яті на всіх переходах автомата та використовується для кодування станів автомата при синтезі на базі *T, RS, JK*-тригерів. Для даних типів тригерів (на відміну від *D*-тригерів) на кожному переході, де тригер змінює своє значення на протилежне, одна з функцій збудження обов'язково дорівнює 1. Зменшення кількості перемикань тригерів призводить до зменшення кількості одиниць відповідних функцій збудження, що за відсутності мінімізації однозначно призводить до спрощення комбінаційної схеми автомата.

Дамо деякі означення.

Нехай $G(S)$ – неорієнтований граф переходів автомата S. Вершини графа ототожнюються зі станами автомата. Вершини i та j сполучені ребром, якщо є перехід від a_i до a_j або навпаки.

Позначимо $q(i, j)$ кількість усіх переходів автомата від стану a_i до стану a_j. Кожному ребру (i, j) графа $G(S)$ поставимо у відповідність вагу ребра $p(i, j) = q(i, j) + q(j, i)$.

Введемо функцію $w(i, j) = p(i, j) \cdot d(i, j)$, де $d(i, j)$ – кількість компонентів, якими коди станів a_i та a_j відрізняються один від одного (тобто кодова відстань між кодами a_i та a_j). Функція $w(i, j)$ має простий фізичний зміст. Перехід автомата зі стану a_i у стан a_j (або навпаки) супроводжується перемиканням стількох тригерів, скількома компонентами відрізняються коди цих станів.

Функція $w = \sum_{(i,j) \in G(S)} w(i, j) = \sum_{(i,j) \in G(S)} p(i, j) d(i, j)$ показує, скільки всього перемикається тригерів при проходженні автомата по всіх можливих переходах. Функція w показує, скільки всього одиниць у функції збудження, тобто дає змогу оцінювати складність КС автомата. Функцію w можна розглядати як певну цільову функцію, мінімум якої визначить таке кодування, за якого КС буде найпростішою. До речі, мінімальна кодова відстань між різними станами дорівнює 1 та якщо вдається закодувати всі стани сусіднім кодуванням, то очевидно, що w буде мінімально можливою та дорівнюватиме $w = \sum_{(i,j) \in G(S)} p(i, j)$, тобто сумарній кількості переходів автомата.

Із виразу для w випливає, що перехід з a_i в a_j, для якого $d(i, j) = 0$, не впливає на w (що цілком очевидно, якщо врахувати, що на цьому переході жоден тригер не перемикається).

Розглянемо використання евристичного алгоритму на прикладі.

Приклад 15.2. Закодувати внутрішні стани автомата, заданого таблицями переходів та виходів (таблиці 15.6 та 15.7), використовуючи евристичний алгоритм кодування.

Для даного автомата можна побудувати неорієнтований граф (без урахування петель), зображений на рисунку 15.6. На кожному ребрі вказана його вага.

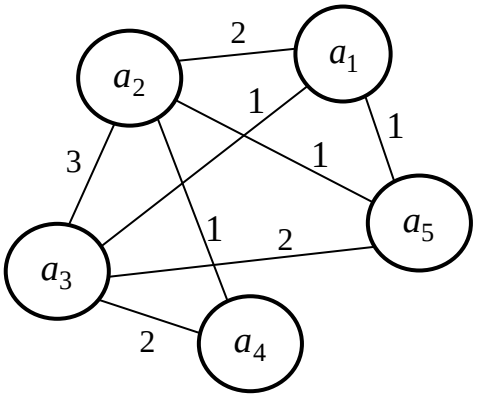


Рисунок 15.6. Граф автомата, заданого таблицями переходів та виходів

у таблицях 15. 6 та 15.7

Евристичний алгоритм кодування буде складатися з наступних кроків:

1. Будуємо матрицю T, що складається зі всіх пар номерів (i, j), для яких $p(i, j) \neq 0$ (тобто в автоматі є перехід з a_i в a_j або навпаки) та $i < j$. Для кожної пари в матриці T вказуємо її вагу $p(i, j)$, що співпадає з вагою ребра, який сполучає a_i та a_j:

$$T = \left\| \begin{array}{ccc} i & j & p(i,j) \\ 1 & 2 & 2 \\ 1 & 3 & 1 \\ 1 & 5 & 1 \\ 2 & 3 & 3 \\ 2 & 4 & 1 \\ 2 & 5 & 1 \\ 3 & 4 & 2 \\ 3 & 5 & 2 \end{array} \right\|.$$

2. Впорядкуємо рядки матриці T. Для цього побудуємо матрицю M таким чином. У перший рядок матриці M запишемо пару (α, β) з найбільшою вагою $p(\alpha, \beta)$. У нашому випадку $(\alpha, \beta) = (2,3)$, $p(2,3) = 3$. Зі всіх пар, що мають спільну компоненту з парою (α, β), вибирається пара (γ, δ) з найбільшою вагою та заноситься в другий рядок матриці M. Зрозуміло, що $(\alpha, \beta) \cdot (\gamma, \delta) \neq 0$. Потім зі всіх пар, які мають спільну компоненту хоча б з однією з внесених вже в матрицю M пар, вибирається пара з найбільшою вагою та заноситься в матрицю M й т.д. У разі рівності ваг пар обчислюються суми ваг компонентів пар (вагою $p(\alpha)$, компонентою α називається кількість появ α в матриці T) та в матрицю M заноситься пара з найбільшою сумою ваг. У даному автоматі на друге місце вслід за парою (2,3) претендують пари (1,2) з $p(1,2) = 2$; (3,4) з $p(3,4) = 2$, (3,5) з $p(3,5) = 2$.

Для визначення того, яка пара займе друге місце в матриці M, знаходимо вагу компонент пар

$p(1) = 3$, $p(2) = 3$, $p(1) + p(2) = 6$,
$p(3) = 4$, $p(4) = 2$, $p(3) + p(4) = 6$,
$p(3) = 4$, $p(5) = 2$, $p(3) + p(5) = 6$.

У даному випадку для всіх пар співпадають як їх вага, так і вага їх компонент. Тому на друге місце матриці M може бути поставлена будь-яка з пар (1,2), (3,4) (3,5). Але тоді на 3-му та 4-му будуть інші дві. Виконавши впорядковування всіх пар, отримаємо матрицю M у вигляді

$$M = \left\| \begin{array}{ccc} i & j & p(i,j) \\ 2 & 3 & 3 \\ 1 & 2 & 2 \\ 3 & 4 & 2 \\ 3 & 5 & 2 \\ 1 & 3 & 1 \\ 1 & 5 & 1 \\ 2 & 4 & 1 \\ 2 & 5 & 1 \end{array} \right\|.$$

3. Визначаємо розрядність коду для кодування станів автомата (кількість елементів пам'яті – тригерів). Усього станів $A = 5$. Тоді

$$R \geq \log_2 A = \log_2 5 = 3.$$

Закодуємо стани з першого рядка матриці таким чином:

$$K_2 = K(a_2) = 000,$$

$$K_3 = K(a_3) = 001.$$

Для зручності кодування ілюструватимемо цей процес картою Карно (рисунок 15.7).

	00	01	11	10
0	a_2	a_3		
1				

Рисунок 15.7. Нанесення станів a_2 та a_3 на карту Карно

4. Викреслимо з матриці M перший рядок, що відповідає закодованим станам a_2 та a_3. Отримаємо матрицю M'.

$$M' = \left\| \begin{array}{ccc} i & j & p(i,j) \\ 1 & 2 & 2 \\ 3 & 4 & 2 \\ 3 & 5 & 2 \\ 1 & 3 & 1 \\ 1 & 5 & 1 \\ 2 & 4 & 1 \\ 2 & 5 & 1 \end{array} \right\|.$$

5. У першому рядку матриці M' закодований один елемент, що відповідає стану a_2. Виберемо з першого рядка незакодований елемент та позначимо його через γ (у нашому випадку $\gamma = 1$).

6. Будуємо матрицю M_γ, вибравши з M' стрічки, що містять γ:

$$M_\gamma = \left\| \begin{array}{cc|c} i & j & p(i,j) \\ 1 & 2 & 2 \\ 1 & 3 & 1 \\ 1 & 5 & 1 \end{array} \right\|.$$

Нехай $B_\gamma = \{\gamma_1 \ldots \gamma_F\}$ – множина елементів з матриці M_γ, які вже закодовані. Їх коди $K_{\gamma_1}, \ldots, K_{\gamma_F}$ відповідно. У нашому випадку

$$B_\gamma = B_1 = \{2,3\}, \qquad K_2 = 000, \quad K_3 = 001.$$

7. Для кожного K_{γ_1}, $f = \overline{1,F}$ знайдемо $C^1_{\gamma_f}$ – множину кодів, сусідніх з K_{γ_f} та ще незайнятих для кодування станів автомата (для сусідніх кодів кодова відстань $d = 1$)

$K_2 = 000$, $C^1_2 = \{100, 010\}$,

$K_3 = 001$, $C^1_3 = \{011, 101\}$.

Побудуємо множину $D^1_\gamma = \bigcup_{f=1}^{F} C^1_{\gamma_f} = C^1_{\gamma_1} \cup C^1_{\gamma_2} \cup \ldots \cup C^1_{\gamma_F}$

$D^1_3 = C^1_2 \cup C^1_3 = \{100,010,011,101\}$

Якщо виявляється, що $D^1_\gamma = \varnothing$, то будуємо нову множину $D^2_\gamma = \bigcup_{f=1}^{F} C^2_{\gamma_f}$, де $C^2_{\gamma_f}$ – множина кодів, у яких кодова відстань до коду K_{γ_f} дорівнює 2 й т.д.

8. Для кожного коду з множини D_γ знаходимо кодову відстань до коду K_{γ_f}

$K_2 = 000$, $K_3 = 001$,

$d(100, 000) = 1$, $d(100, 001) = 2$,

$d(010, 000) = 1$, $d(010, 001) = 2$,

$d(011, 000) = 2$, $d(011, 001) = 1$,

$d(101, 000) = 2$, $d(100, 001) = 1$.

9. Знаходимо значення функції w для кожного коду з множини D_γ

$$w_{100} = d(100,000) \cdot p(1,2) + d(100,001) \cdot p(1,3) = 1 \cdot 2 + 2 \cdot 1 = 4,$$

$$w_{010} = d(010,000) \cdot p(1,2) + d(010,001) \cdot p(1,3) = 1 \cdot 2 + 2 \cdot 1 = 4,$$

$$w_{011} = d(011,000) \cdot p(1,2) + d(011,001) \cdot p(1,3) = 2 \cdot 2 + 2 \cdot 1 = 5,$$

$$w_{101} = d(101,000) \cdot p(1,2) + d(101,001) \cdot p(1,3) = 2 \cdot 2 + 2 \cdot 1 = 5.$$

10. З множини D_γ вибираємо код K_γ, якому відповідає мінімальне значення функції w в п.9. Вибираємо код $K_1 = 100$ для стану a_1 та заносимо його на карту Карно (рисунок 15.8).

	00	01	11	10
0	a_2	a_3		
1	a_1			

Рисунок 15.8. Нанесення стану a_1 на карту Карно

11. З матриці M' викреслюємо рядки, в яких обидва елементи вже закодовано, внаслідок чого отримаємо нову матрицю M''. Якщо в новій матриці M'' не залишилося жодного рядка, то кодування закінчено. Інакше повертаємося до п.5. У нашому випадку маємо

$$M'' = \left\| \begin{array}{ccc} i & j & p(i,j) \\ 3 & 4 & 2 \\ 3 & 5 & 2 \\ 1 & 5 & 1 \\ 2 & 4 & 1 \\ 2 & 5 & 1 \end{array} \right\|,$$

$$\gamma = 4, \qquad \mathrm{M}_\gamma = M_4 = \left\| \begin{array}{cc|c} 3 & 4 & 2 \\ 2 & 4 & 1 \end{array} \right\|, \qquad B_\gamma = B_4 = \{2,3\},$$

$K_2 = 000$, $\quad C_2^1 = \{010\}$,

$K_3 = 001$, $\quad C_3^1 = \{011, 101\}$,

$$D_4^1 = C_2^1 \cup C_3^1 = \{010,011,101\},$$

$K_2 = 000$, $\quad K_3 = 001$,

$d(010, 000) = 1$, $\quad d(010, 001) = 2$,

$d(011, 000) = 2$, $\quad d(011, 001) = 1$,

$d(101, 000) = 2$, $\quad d(101, 001) = 1$,

$$w_{010} = d(010,000) \cdot p(2,4) + d(010,001) \cdot p(3,4) = 1 \cdot 1 + 2 \cdot 2 = 5,$$

$$w_{011} = d(011,000) \cdot p(2,4) + d(011,001) \cdot p(3,4) = 2 \cdot 1 + 1 \cdot 2 = 4,$$

$$w_{101} = d(101,000) \cdot p(2,4) + d(101,001) \cdot p(3,4) = 2 \cdot 1 + 1 \cdot 2 = 4.$$

Вибираємо $K_4 = 101$ та заносимо стан a_4 на карту Карно (рисунок 15.9).

	00	01	11	10
0	a_2	a_3		
1	a_1	a_4		

Рисунок 15.9. Нанесення стану a_4 на карту Карно

$$M''' = \left\| \begin{array}{c|c} 3\,5 & 2 \\ 1\,5 & 1 \\ 2\,5 & 1 \end{array} \right\|, \qquad \gamma = 5, \qquad M_\gamma = M_5 = \left\| \begin{array}{c|c} 3\,5 & 2 \\ 1\,5 & 1 \\ 2\,5 & 1 \end{array} \right\|, \qquad B\gamma = B_5 = \{1,2,3\},$$

$$K_1 = 100, \qquad C_1^1 = \{110\},$$
$$K_2 = 000, \qquad C_2^1 = \{010\},$$
$$K_3 = 001, \qquad C_3^1 = \{011\},$$

$$D_5^1 = C_1^1 \cup C_2^1 \cup C_3^1 = \{110,010,011\},$$

$K_1 = 100$, $K_2 = 000$, $K_3 = 001$,
$d(110, 100) = 1$, $d(110, 000) = 2$, $d(110, 001) = 3$,
$d(010, 100) = 2$, $d(010, 000) = 1$, $d(010, 001) = 2$,
$d(011, 100) = 3$, $d(011, 000) = 2$, $d(011, 001) = 1$,

$$w_{110} = d(110,100) \cdot p(1,5) + d(110,000) \cdot p(2,5) + d(110,001) \cdot p(3,5) = 1 \cdot 1 + 2 \cdot 1 + 3 \cdot 2 = 9,$$
$$w_{010} = d(010,100) \cdot p(1,5) + d(010,000) \cdot p(2,5) + d(010,001) \cdot p(3,5) = 2 \cdot 1 + 1 \cdot 1 + 2 \cdot 2 = 7,$$
$$w_{011} = d(011,100) \cdot p(1,5) + d(011,000) \cdot p(2,5) + d(011,001) \cdot p(3,5) = 3 \cdot 1 + 2 \cdot 1 + 1 \cdot 2 = 7.$$

Вибираємо $K_5 = 011$ та заносимо стан a_5 на карту Карно (рисунок 15.10).

	00	01	11	10
0	a_2	a_3	a_5	
1	a_1	a_4		

Рисунок 15.10. Нанесення стану a_5 на карту Карно

Оскільки всі стани автомата закодовані, то робота алгоритму закінчується. Загальна кількість перемикань тригерів дорівнює

$$w = \sum p(i, j)d(i, j) = p(2,3) \cdot d(2,3) + p(1,2) \cdot d(1,2) + p(3,4) \cdot d(3,4) +$$
$$+ p(3,5) \cdot d(3,5) + p(1,3) \cdot d(1,3) + p(1,5) \cdot d(1,5) + p(2,4) \cdot d(2,4) +$$
$$+ p(2,5) \cdot d(2,5) = 3 \cdot 1 + 2 \cdot 1 + 2 \cdot 1 + 2 \cdot 1 + 1 \cdot 2 + 1 \cdot 3 + 1 \cdot 2 + 1 \cdot 2 = 18.$$

Мінімальна можлива кількість перемикань (якби стани були закодовані сусіднім кодуванням) дорівнює

$$w_{\min} = \sum p(i, j) = 13.$$

Коефіцієнт ефективності кодування визначається

$$K_{еф} = \frac{w}{w_{\min}}.$$

Очевидно, що $K_{еф} \geq 1$, зокрема, чим менше значення $K_{еф}$, тим ближче евристичний алгоритм кодування наближається до сусіднього, при якому $K_{еф} = 1$.

У нашому випадку коефіцієнт ефективності дорівнює

$$K_{еф} = \frac{18}{13} \approx 1{,}38.$$

Розглянутий алгоритм кодування є машино-орієнтованим, існують програми, що реалізовують цей алгоритм.

15.2.3 Сусіднє кодування логічно суміжних станів абстрактних автоматів

Існує інший метод кодування внутрішніх станів абстрактних автоматів, що дає змогу спростити структурну схему. Суть цього методу полягає у використанні двох наступних правил кодування.

Правило 15.1. Ті стани, з яких можливі переходи в одні й ті ж самі стани хоча б для одного значення вхідного сигналу, є логічно суміжними та повинні бути закодовані сусідніми кодами.

Правило 15.2. Логічно суміжними є стани, що слідують для одного й того самого стану. Їх необхідно кодувати сусідніми кодами.

Якщо при використанні цих правил неможливо закодувати сусідніми кодами всі логічно суміжні стани, то пріоритет повинен зберігатися за правилом 15.1.

У таблиці переходів стани, що задовольняють правило 15.1, повинні мати однакові стани переходу в будь-якому рядку. Стани, що задовольняють правило 15.2, знаходяться в одному стовпці таблиці переходів.

Приклад 15.3. Закодувати таблицю переходів автомата (таблиця 15.6) сусіднім кодуванням логічно суміжних станів.

Таблиця 15.6. Таблиця переходів автомата до прикладу 15.3

	a_0	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_3	a_2	a_3	a_5	-	a_1	-
z_2	a_4	a_3	a_5	a_4	a_2	a_5	a_2
z_3	-	a_1	a_1	a_6	a_6	-	a_1

Перед початком операції кодування доцільно зробити всі довизначення, якщо це необхідно. Далі виписуємо групи станів, у яких є однакові елементи у будь-якому рядку (правило 15.1):

(a_0, a_2) – обидва переходять в a_3 за сигналом z_1,

(a_0, a_3) – обидва переходять в a_4 за сигналом z_2,

(a_2, a_5) – обидва переходять в a_5 за сигналом z_2,

(a_4, a_6) – обидва переходять в a_2 за сигналом z_2,

(a_3, a_4) – обидва переходять в a_6 за сигналом z_3,

(a_1, a_2, a_6) – обидва переходять в a_1 за сигналом z_3.

Виписуємо групи станів, що знаходяться в одних й тих самих стовпцях. У нашому прикладі – це пари (a_3, a_4), (a_2, a_1), (a_2, a_3, a_1), (a_3, a_5, a_1), (a_5, a_4, a_6), (a_2, a_6), (a_1, a_5). Усі стани, що знаходяться в кожній зі сформованих груп, повинні бути закодовані сусідніми кодами. Для цього на основі отриманих груп складаємо класи станів, що логічно суміжні з кожним зі станів автомата, зокрема, кожну пару логічно суміжних станів включаємо тільки в один клас. Отримаємо наступні класи станів (таблиця 15.7). Пари станів, які отримані відповідно до правила 15.1, позначені знаком «*» .

Таблиця 15.7. Класи логічно суміжних станів

K_0	K_1	K_2	K_3	K_4	K_5
$*(a_0, a_3)$	$*(a_1, a_2)$	$*(a_2, a_5)$	$*(a_3, a_4)$	(a_4, a_5)	$*(a_5, a_6)$
$*(a_0, a_2)$	$*(a_1, a_6)$	$*(a_2, a_6)$	(a_3, a_5)	$*(a_4, a_6)$	
	(a_1, a_5)	(a_2, a_3)			
	(a_1, a_3)				

Для кодування логічно суміжних станів скористаємося картою Карно, віддаючи пріоритет парам станів, що були отримані за правилом 15.1 (рисунок 15.11).

	00	01	11	10
0	a_1		a_4	a_6
1	a_2	a_0	a_3	a_5

Рисунок 15.11. Нанесення станів на карту Карно

Виявилося, що в даному випадку не вдалося закодувати сусідніми кодами пари логічно суміжних станів (a_2, a_6), (a_2, a_3), (a_4, a_5), (a_1, a_5), (a_1, a_3).

У результаті отримали коди станів $K(a_0) = 101$, $K(a_1) = 000$, $K(a_2) = 100$, $K(a_3) = 111$, $K(a_4) = 011$, $K(a_5) = 110$, $K(a_6) = 010$.

Ефективність кодування визначається за формулою

$$k_{еф} = \frac{m}{n},$$

де m – кількість пар логічно суміжних станів, які вдалося закодувати сусідніми кодами, n – загальна кількість пар станів, сформованих у класах K_1, K_2, …, K_N.

У розглянутому прикладі коефіцієнт ефективності дорівнює

$$k_{еф} = \frac{9}{13} \approx 0{,}69.$$

Ефективним вважається кодування, для якого $k_{еф} \geq 0{,}5$.

Контрольні запитання

1. Дайте означення поняттю «кодування внутрішніх станів автомата».
2. Що таке «гонки» в автоматах. Назвіть методи усунення гонок.
3. Охарактеризуйте принцип сусіднього кодування внутрішніх станів абстрактних автоматів.
4. Наведіть алгоритм кодування абстрактних автоматів, побудованих на D-тригерах.
5. Назвіть етапи евристичного алгоритму кодування внутрішніх станів абстрактних автоматів.
6. Охарактеризуйте сусіднє кодування абстрактних автоматів із визначенням логічно суміжних станів.

ЛЕКЦІЯ 16

КАНОНІЧНИЙ МЕТОД СТРУКТУРНОГО СИНТЕЗУ ЦИФРОВИХ АВТОМАТІВ

Розглянемо застосування канонічного методу структурного синтезу цифрових автоматів на прикладах.

Приклад 16.1. Виконати структурний синтез автомата Мілі, заданого своїми таблицями переходів та виходів (таблиці 16.1 та 16.2). Як елементи пам'яті використати *D*-тригери.

Таблиця 16.1. Таблиця переходів автомата Мілі

	a_1	a_2	a_3	a_4
z_1	a_1	a_3	a_1	a_2
z_2	a_3	a_2	a_4	a_1
z_3	a_2	a_4	a_3	a_2
z_4	a_3	a_4	a_1	a_4

Таблиця 16.2. Таблиця виходів автомата Мілі

	a_1	a_2	a_3	a_4
z_1	w_1	w_4	w_3	w_6
z_2	w_3	w_2	w_2	w_4
z_3	w_5	w_1	w_5	w_1
z_4	w_1	w_4	w_6	w_2

Синтез виконуватимемо в наступному порядку:

1. Визначаємо кількість вхідних, вихідних структурних сигналів та внутрішніх станів автомата.

Кількість вхідних абстрактних сигналів $F = 4$, отже, кількість вхідних структурних сигналів $L \geq \log_2 F = \log_2 4 = 2$, тобто x_1, x_2.

Кількість вихідних абстрактних сигналів $G = 6$, отже кількість вихідних структурних сигналів $N \geq \log_2 G = \log_2 6 = 3$, тобто y_1, y_2 та y_3.

Кількість внутрішніх станів абстрактного автомата $M = 4$, отже кількість двійкових елементів пам'яті (тригерів) $R \geq \log_2 M = \log_2 4 = 2$.

2. Будуємо структурну схему цифрового автомата.

Структурна схема цифрового автомата з урахуванням того, що початковий автомат є автоматом Мілі, як елемент пам'яті використовується *D*-тригер, може бути подана у вигляді рисунка 16.1.

3. Кодуємо вхідні, вихідні сигнали та внутрішні стани автомата.

Кодування вхідних, вихідних сигналів та внутрішніх станів подано в таблицях 16.3–16.5. Кодування, в загальному випадку, здійснюється довільно. Тому, наприклад, кожному з сигналів z_i можна поставити у відповідність будь-яку дворозрядну комбінацію x_1, x_2. Необхідно тільки, щоб різні вихідні сигнали z_i кодувалися різними комбінаціями x_1, x_2. Аналогічно для w_i та a_i.

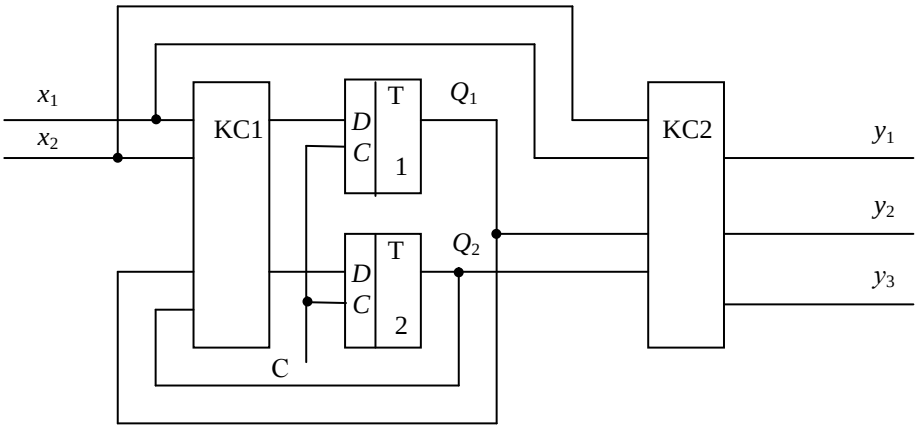


Рисунок 16.1. Структурна схема цифрового автомата Мілі

Таблиця 16.3. Кодування вхідних сигналів

Вхідні сигнали	Код вхідних сигналів	
	x_1	x_2
z_1	0	0
z_2	0	1
z_3	1	0
z_4	1	1

Таблиця 16.4. Кодування станів автомата

Стани автомата	Код стану	
	Q_1	Q_2
a_1	0	0
a_2	0	1
a_3	1	0
a_4	1	1

Таблиця 16.5. Кодування вихідних сигналів

Вихідні сигнали	Код вихідних сигналів		
	y_1	y_2	y_3
w_1	0	0	0
w_2	0	0	1
w_3	0	1	0
w_4	0	1	1
w_5	1	0	0
w_6	1	0	1

4. Будуємо кодовані таблиці переходів та виходів структурного автомата (таблиці 16.6 та 16.7). Для цього в таблицях переходів та виходів початкового абстрактного автомата замість a_i, z_i, w_i ставимо відповідні коди (таблиці 16.3–16.5).

Таблиця 16.6. Закодована таблиця переходів автомата Мілі

Q_1Q_2 / x_1x_2	00	01	10	11
00	00	10	00	01
01	10	01	11	00
10	01	11	10	01
11	10	11	00	11

Таблиця 16.7. Закодована таблиця виходів автомата Мілі

Q_1Q_2 / x_1x_2	00	01	10	11
00	000	011	010	101
01	010	001	001	011
10	100	000	100	000
11	000	011	101	001

5. Знаходимо функції y_1, y_2 та y_3.

Функції y_1, y_2 та y_3 можуть бути безпосередньо отримані з таблиці виходів (таблиця 16.7) у вигляді

$$y_1 = x_1\bar{x}_2\bar{Q}_1\bar{Q}_2 + x_1\bar{x}_2Q_1\bar{Q}_2 + x_1x_2Q_1\bar{Q}_2 + \bar{x}_1\bar{x}_2Q_1Q_2,$$
$$y_2 = \bar{x}_1x_2\bar{Q}_1\bar{Q}_2 + \bar{x}_1\bar{x}_2\bar{Q}_1Q_2 + x_1x_2\bar{Q}_1Q_2 + \bar{x}_1\bar{x}_2Q_1\bar{Q}_2 + \bar{x}_1x_2Q_1Q_2,$$
$$y_3 = \bar{x}_1\bar{x}_2\bar{Q}_1Q_2 + \bar{x}_1x_2\bar{Q}_1Q_2 + x_1x_2\bar{Q}_1Q_2 + \bar{x}_1x_2Q_1\bar{Q}_2 + \bar{x}_1\bar{x}_2Q_1Q_2 +$$
$$+ \bar{x}_1x_2Q_1Q_2 + x_1x_2Q_1\bar{Q}_2 + x_1x_2Q_1Q_2.$$

Вирази для y_1, y_2 та y_3 можна істотно спростити в результаті мінімізації, наприклад, за допомогою карт Карно (рисунки 16.2-16.4).

	$\bar{Q}_1\bar{Q}_2$	$\bar{Q}_1Q_2$	Q_1Q_2	$Q_1\bar{Q}_2$
$\bar{x}_1\bar{x}_2$			1	
$\bar{x}_1x_2$				
x_1x_2				1
$x_1\bar{x}_2$	1			1

Рисунок 16.2. Мінімізація функції y_1

	$\bar{Q}_1\bar{Q}_2$	$\bar{Q}_1Q_2$	Q_1Q_2	$Q_1\bar{Q}_2$
$\bar{x}_1\bar{x}_2$		1		1
$\bar{x}_1x_2$	1		1	
x_1x_2		1		
$x_1\bar{x}_2$				

Рисунок 16.3. Мінімізація функції y_2

	$\bar{Q}_1\bar{Q}_2$	$\bar{Q}_1Q_2$	Q_1Q_2	$Q_1\bar{Q}_2$
$\bar{x}_1\bar{x}_2$		1	1	
$\bar{x}_1x_2$		1	1	1
x_1x_2		1	1	1
$x_1\bar{x}_2$				

Рисунок 16.4. Мінімізація функції y_3

У результаті мінімізації отримаємо

$$y_1 = x_1\bar{x}_2\bar{Q}_2 + x_1Q_1\bar{Q}_2 + \bar{x}_1\bar{x}_2Q_1Q_2,$$
$$y_2 = \bar{x}_1x_2\bar{Q}_1\bar{Q}_2 + \bar{x}_1\bar{x}_2\bar{Q}_1Q_2 + x_1x_2\bar{Q}_1Q_2 + \bar{x}_1\bar{x}_2Q_1\bar{Q}_2 + \bar{x}_1x_2Q_1Q_2,$$
$$y_3 = \bar{x}_1Q_2 + x_2Q_2 + x_2Q_1.$$

6. Знаходимо функції входів для двох тригерів D_1 та D_2.

Для отримання виразів для D_1 та D_2 необхідно отримати таблиці функцій збудження для двох тригерів. Для чого, в загальному випадку, необхідно скористатися таблицею переходів та таблицею функцій входів елементів пам'яті. Знаючи код початкового стану автомата та код стану переходу, на підставі таблиці входів тригера отримуємо необхідне значення функції збудження, що забезпечує заданий перехід. Однак для *D*-тригерів, як наголошувалося раніше, таблиця переходів співпадає з таблицею функції збудження. Тоді безпосередньо із закодованої таблиці переходів автомата Мілі (таблиця 16.6) знаходимо функції D_1 та D_2

$$D_1 = \bar{x}_1 x_2 \overline{Q}_1 \overline{Q}_2 + x_1 x_2 \overline{Q}_1 \overline{Q}_2 + \bar{x}_1 \bar{x}_2 \overline{Q}_1 Q_2 + x_1 \bar{x}_2 \overline{Q}_1 Q_2 + x_1 x_2 \overline{Q}_1 Q_2 + \bar{x}_1 x_2 Q_1 \overline{Q}_2 +$$
$$+ x_1 \bar{x}_2 Q_1 \overline{Q}_2 + x_1 x_2 Q_1 Q_2,$$
$$D_2 = x_1 \bar{x}_2 \overline{Q}_1 \overline{Q}_2 + \bar{x}_1 x_2 \overline{Q}_1 Q_2 + x_1 \bar{x}_2 \overline{Q}_1 Q_2 + x_1 x_2 \overline{Q}_1 Q_2 + \bar{x}_1 x_2 Q_1 \overline{Q}_2 + \bar{x}_1 \bar{x}_2 Q_1 Q_2 +$$
$$+ x_1 \bar{x}_2 Q_1 Q_2 + x_1 x_2 Q_1 Q_2.$$

Використовуємо карти Карно (рисунки 16.5 та 16.6) для мінімізації цих функцій.

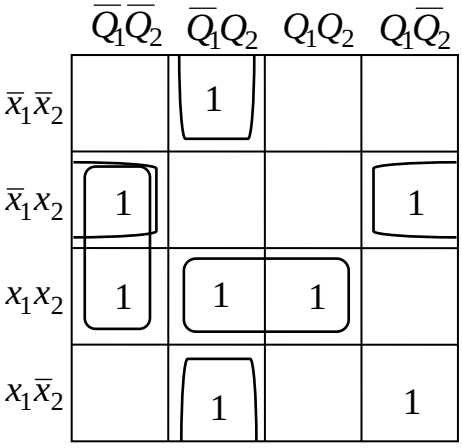


Рисунок 16.5. Мінімізація функції D_1

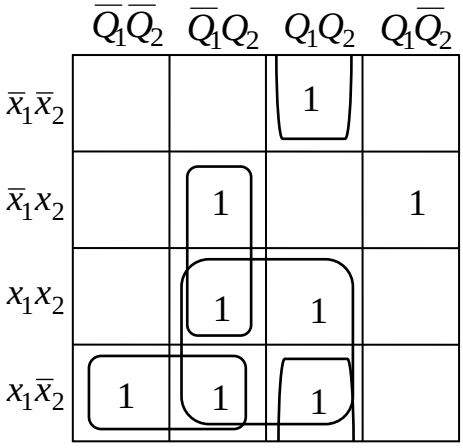


Рисунок 16.6. Мінімізація функції D_2

У результаті мінімізації отримаємо

$$D_1 = x_2 \overline{Q}_1 \overline{Q}_2 + \bar{x}_1 x_2 \overline{Q}_2 + \bar{x}_2 \overline{Q}_1 Q_2 + x_1 x_2 Q_2 + x_1 \bar{x}_2 Q_1 \overline{Q}_2,$$
$$D_2 = x_1 Q_2 + x_2 \overline{Q}_1 Q_2 + x_1 \bar{x}_2 \overline{Q}_1 + \bar{x}_1 x_2 Q_1 \overline{Q}_2 + \bar{x}_2 Q_1 Q_2.$$

7. На підставі отриманих у результаті синтезу виразів для D_1, D_2, y_1, y_2 та y_3 будуємо функціональну схему автомата (рисунок 16.7). Додатково на функціональній схемі показаний сигнал $\overline{\text{Reset}}$, що встановлює автомат у початковий стан (у даному випадку 00).

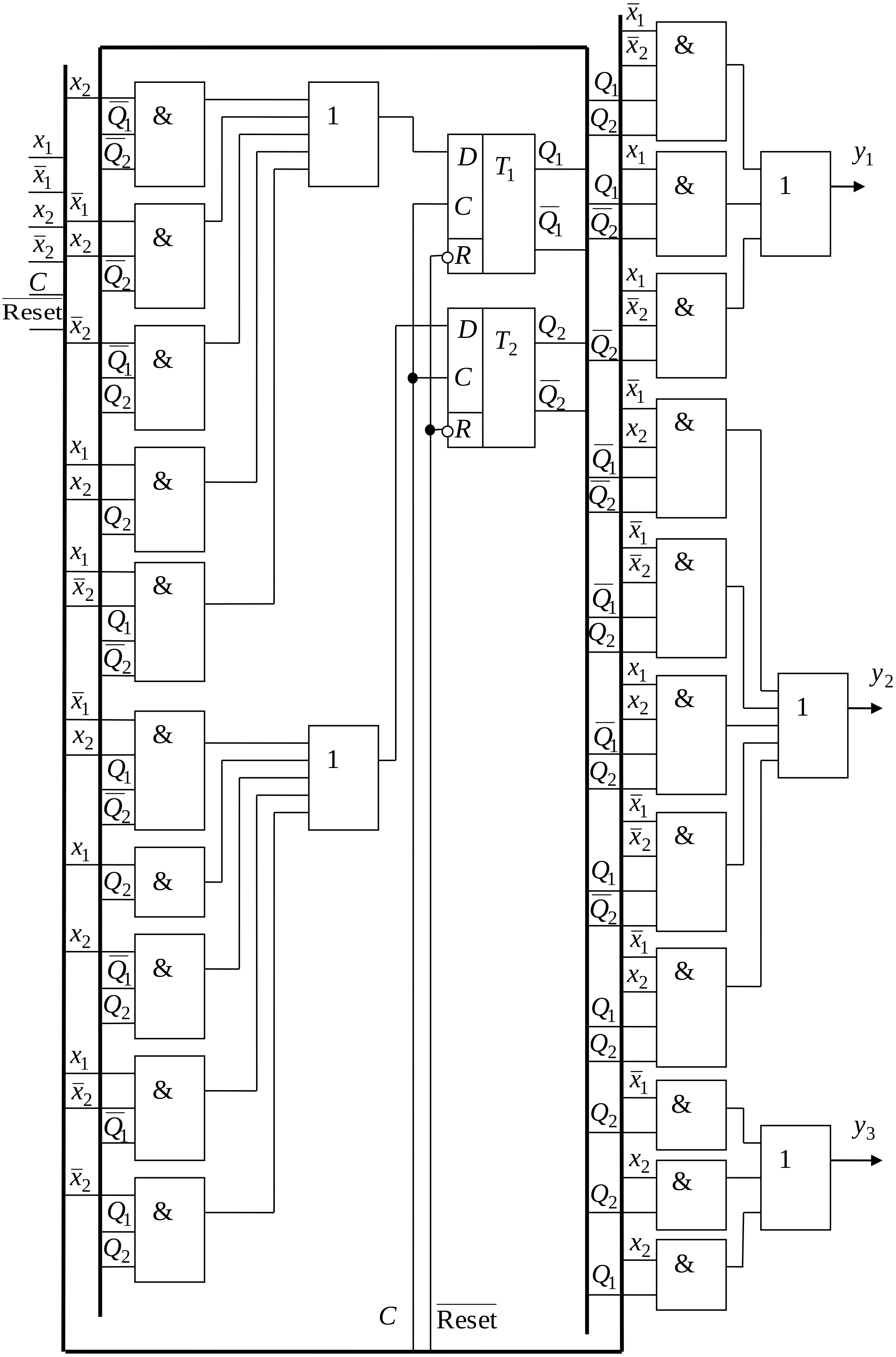


Рисунок 16.7. Функціональна схема автомата

Приклад 16.2. Виконати структурний синтез частково визначеного автомата Мілі, заданого своїми таблицями переходів та виходів (таблиці 16.8 та 16.9). Як елементи пам'яті використати RS-тригери.

Таблиця 16.8. Таблиця переходів частково визначеного автомата Мілі

	a_1	a_2	a_3	a_4
z_1	a_2	-	a_4	-
z_2	a_3	a_2	-	a_2
z_3	a_4	a_1	a_1	a_3
z_4	-	a_3	a_2	-

Таблиця 16. 9. Таблиця виходів частково визначеного автомата Мілі

	a_1	a_2	a_3	a_4
z_1	w_1	-	w_2	-
z_2	w_3	w_2	-	w_1
z_3	w_4	w_4	w_1	w_2
z_4	-	w_3	w_4	-

Синтез виконуватимемо в наступному порядку:

1. Визначаємо кількість вхідних, вихідних структурних сигналів та внутрішніх станів автомата.

Кількість вхідних абстрактних сигналів $F = 4$, отже кількість вхідних структурних сигналів $L \geq \log_2 F = \log_2 4 = 2$, тобто x_1, x_2.

Кількість вихідних абстрактних сигналів $G = 4$, отже кількість вихідних структурних сигналів $N \geq \log_2 G = \log_2 4 = 2$, тобто y_1, y_2.

Кількість внутрішніх станів абстрактного автомата $M = 4$, отже кількість двійкових елементів пам'яті (тригерів) $R \geq \log_2 M = \log_2 4 = 2$.

2. Будуємо структурну схему цифрового автомата на RS−тригерах (рисунок 16.8).

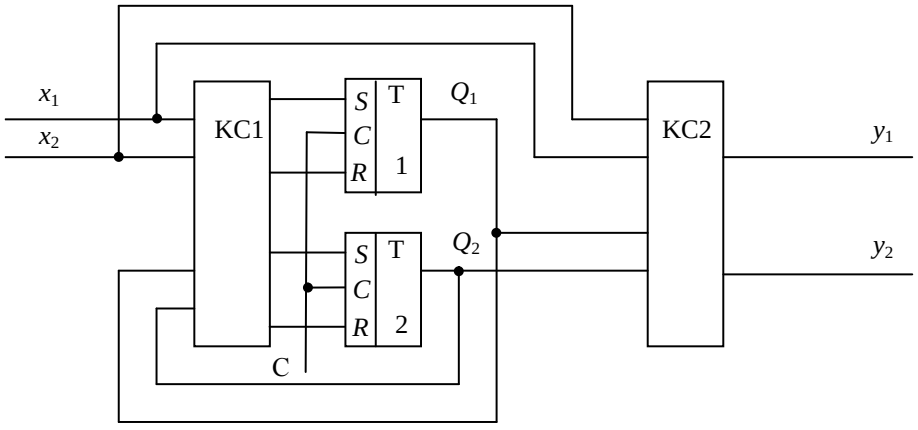


Рисунок 16.8. Структурна схема цифрового автомата Мілі

3. Кодуємо вхідні, вихідні сигнали та внутрішні стани автомата (таблиці 16.10-16.12). Як і в попередньому прикладі кодування здійснюємо довільно.

4. Будуємо кодовані таблиці переходів та виходів структурного автомата (таблиці 16.13 та 16.14). Для цього в таблицях переходів та виходів початкового абстрактного автомата (таблиці 16.8 та 16.9) замість a_i, z_i, w_i ставимо їм відповідні коди (таблиці 16.10-16.12).

Таблиця 16.10. Кодування вхідних сигналів

Вхідні сигнали	Код вхідних сигналів	
	x_1	x_2
z_1	0	0
z_2	0	1
z_3	1	0
z_4	1	1

Таблиця 16.11. Кодування станів автомата

Стани автомата	Код стану	
	Q_1	Q_2
a_1	0	0
a_2	0	1
a_3	1	0
a_4	1	1

Таблиця 16.12. Кодування вихідних сигналів

Вихідні сигнали	Код вихідних сигналів	
	y_1	y_2
w_1	0	0
w_2	0	1
w_3	1	0
w_4	1	1

Таблиця 16.13. Закодована таблиця переходів частково визначеного автомата Мілі

x_1x_2 \ Q_1Q_2	00	01	10	11
00	01	-	11	-
01	10	01	-	01
10	11	00	00	10
11	-	10	01	-

Таблиця 16.14. Закодована таблиця виходів частково визначеного автомата Мілі

x_1x_2 \ Q_1Q_2	00	01	10	11
00	00	-	01	-
01	10	01	-	00
10	11	11	00	01
11	-	10	11	-

5. Знаходимо функції y_1, y_2 з таблиці виходів (таблиця 16.14):

$$y_1 = \bar{x}_1 x_2 \overline{Q}_1 \overline{Q}_2 + x_1 \bar{x}_2 \overline{Q}_1 \overline{Q}_2 + x_1 \bar{x}_2 \overline{Q}_1 Q_2 + x_1 x_2 \overline{Q}_1 Q_2 + x_1 x_2 Q_1 \overline{Q}_2,$$

$$y_2 = x_1 \bar{x}_2 \overline{Q}_1 \overline{Q}_2 + \bar{x}_1 x_2 \overline{Q}_1 Q_2 + x_1 \bar{x}_2 \overline{Q}_1 Q_2 + \bar{x}_1 \bar{x}_2 Q_1 \overline{Q}_2 + x_1 x_2 Q_1 \overline{Q}_2 + x_1 \bar{x}_2 Q_1 Q_2.$$

Вирази для y_1 та y_2 мінімізуємо за допомогою карт Карно (рисунки 16.9 та 16.10).

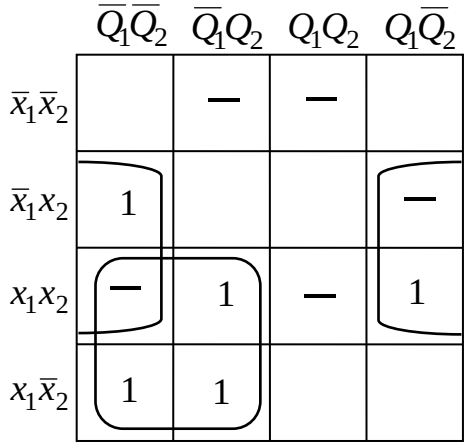


Рисунок 16.9. Мінімізація функції y_1

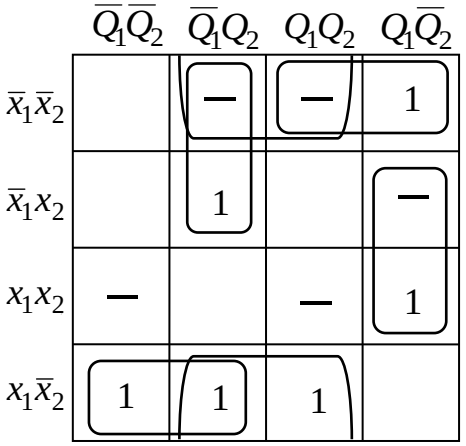


Рисунок 16.10. Мінімізація функції y_2

У результаті мінімізації отримаємо

$$y_1 = x_2\overline{Q}_2 + x_1\overline{Q}_1,$$

$$y_2 = x_1\bar{x}_2\overline{Q}_1 + \bar{x}_1\overline{Q}_1Q_2 + \bar{x}_1\bar{x}_2Q_1 + x_2Q_1\overline{Q}_2 + \bar{x}_2Q_2.$$

6. Знаходимо функції входів RS-тригерів.

Для отримання виразів для R_1, S_1 та R_2, S_2 необхідно отримати таблиці функцій збудження. Для цього необхідно скористатися таблицею переходів та таблицею функцій входів RS-тригерів. Знаючи код початкового стану автомата та код стану переходу, на підставі таблиці входів RS-тригера отримуємо необхідне значення функції збудження, що забезпечує заданий перехід (таблиця 16.15).

Таблиця 16.15. Закодована таблиця функцій збудження RS-тригерів

Q_1Q_2 / x_1x_2	00				01				10				11			
	R_1	S_1	R_2	S_2	R_1	S_1	R_2	S_2	R_1	S_1	R_2	S_2	R_1	S_1	R_2	S_2
00	X	0	0	1	-				0	X	0	1	-			
01	0	1	X	0	X	0	0	X	-				1	0	0	X
10	0	1	0	1	X	0	1	0	1	0	X	0	0	X	1	0
11	-				0	1	1	0	1	0	0	1	-			

Із закодованої таблиці функцій збудження елементів пам'яті (таблиця 16.15) знаходимо функції R_1, S_1 та R_2, S_2:

$$R_1 = x_1\bar{x}_2Q_1\overline{Q}_2 + x_1x_2Q_1\overline{Q}_2 + \bar{x}_1x_2Q_1Q_2,$$

$$S_1 = \bar{x}_1x_2\overline{Q}_1\overline{Q}_2 + x_1\bar{x}_2\overline{Q}_1\overline{Q}_2 + x_1x_2\overline{Q}_1Q_2,$$

$$R_2 = x_1\bar{x}_2\overline{Q}_1Q_2 + x_1x_2\overline{Q}_1Q_2 + x_1\bar{x}_2Q_1Q_2,$$

$$S_2 = \bar{x}_1\bar{x}_2\overline{Q}_1\overline{Q}_2 + x_1\bar{x}_2\overline{Q}_1\overline{Q}_2 + \bar{x}_1\bar{x}_2Q_1\overline{Q}_2 + x_1x_2Q_1\overline{Q}_2.$$

Використаємо карти Карно (рисунки 16.11-16.14) для мінімізації цих функцій.

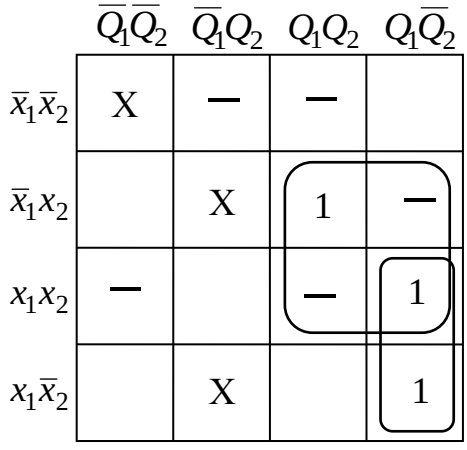


Рисунок 16.16. Мінімізація функції R_1

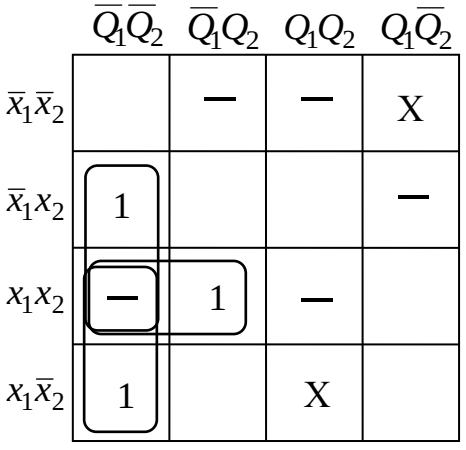


Рисунок 16.17. Мінімізація функції S_1

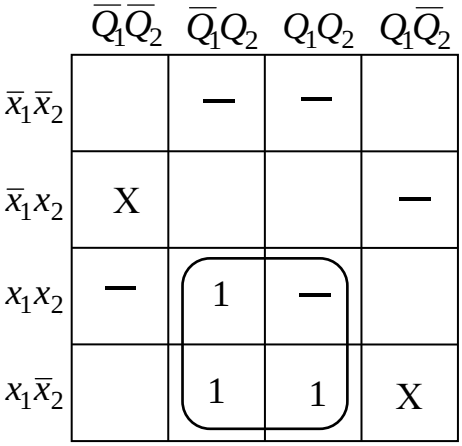


Рисунок 16.13. Мінімізація функції R_2

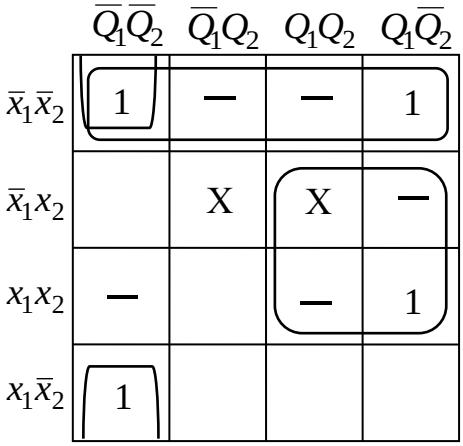


Рисунок 16.14. Мінімізація функції S_2

У результаті мінімізації отримаємо

$$R_1 = x_2Q_1 + x_1Q_1\overline{Q}_2,$$
$$S_1 = x_2\overline{Q}_1\overline{Q}_2 + x_1\overline{Q}_1\overline{Q}_2 + x_1x_2\overline{Q}_1,$$
$$R_2 = x_1Q_2,$$
$$S_2 = \overline{x}_1\overline{x}_2 + \overline{x}_2\overline{Q}_1\overline{Q}_2 + x_2Q_1.$$

7. На підставі отриманих у результаті синтезу виразів R_1, S_1, R_2, S_2 та y_1, y_2 будуємо функціональну схему автомата (рисунок 16.15).

Приклад 16.3. Виконати структурний синтез автомата Мура, заданого таблицею переходів (таблиці 16.16). Як елементи пам'яті використати *T*-тригери.

Таблиця 16.16. Таблиця переходів автомата Мура

	u_1	u_2	u_3	u_4
	a_1	a_2	a_3	a_4
z_1	a_2	a_3	a_2	a_1
z_2	a_1	a_1	a_4	a_2
z_3	a_4	a_4	a_1	a_4

Синтез виконуватимемо в наступному порядку:

1. Визначаємо кількість вхідних, вихідних структурних сигналів та внутрішніх станів автомата.

Кількість вхідних абстрактних сигналів $F = 3$, отже, кількість вхідних структурних сигналів $L \geq \log_2 F = \log_2 3 = 1{,}58$, приймаємо $L = 2$, тобто x_1, x_2.

Кількість вихідних абстрактних сигналів $G = 4$, отже кількість вихідних структурних сигналів $N \geq \log_2 G = \log_2 4 = 2$, тобто g_1, g_2.

Кількість внутрішніх станів абстрактного автомата $M = 4$, отже кількість двійкових елементів пам'яті (*T*-тригерів) $R \geq \log_2 M = \log_2 4 = 2$.

2. Будуємо структурну схему цифрового автомата.

Структурна схема цифрового автомата з урахуванням того, що початковий автомат є автоматом Мура, як елемент пам’яті використовується *T*-тригер, може бути подана у вигляді рисунка 16.16.

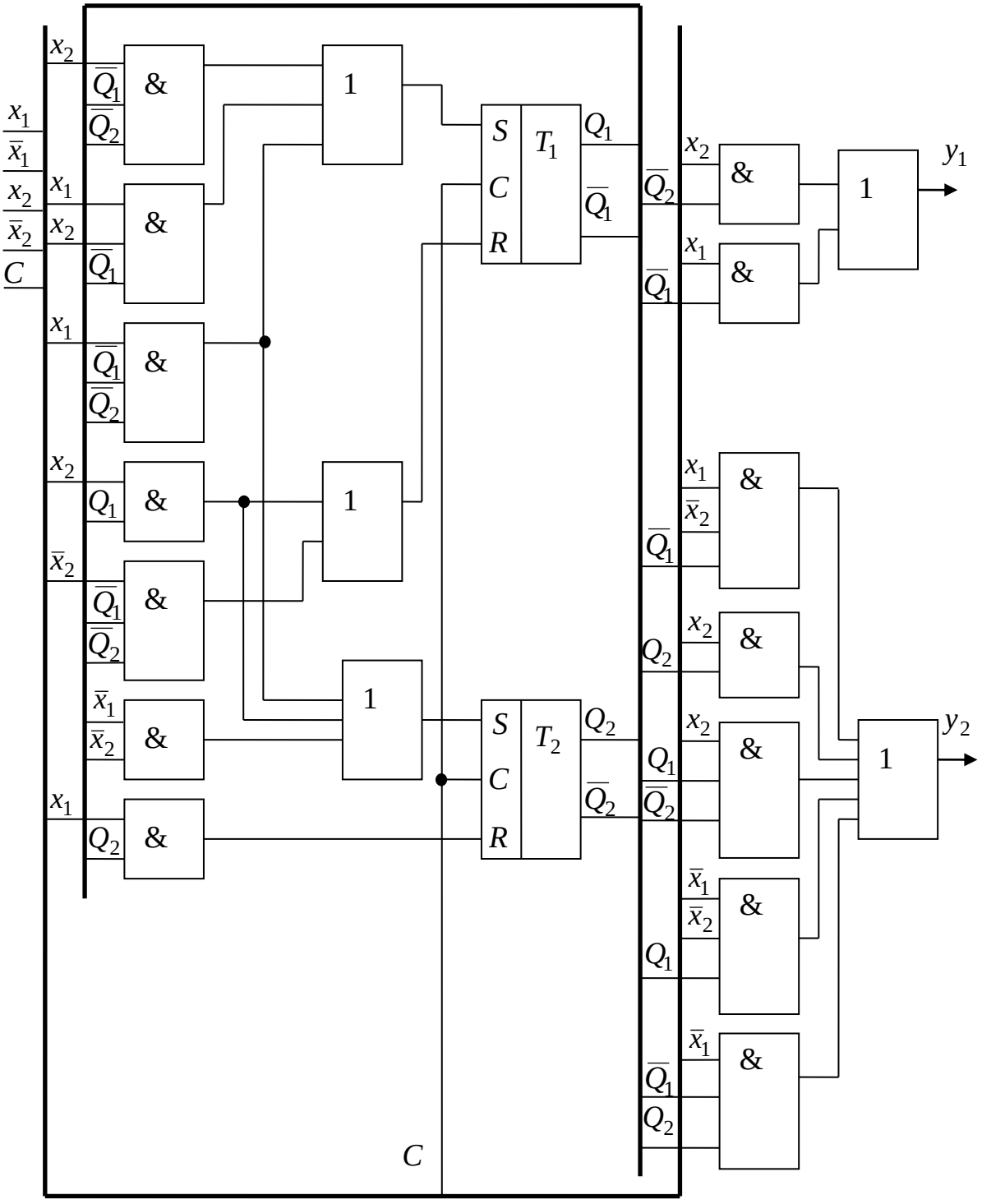


Рисунок 16.15. Функціональна схема автомата

3. Кодуємо вхідні, вихідні сигнали та внутрішні стани автомата.

Кодування вхідних, вихідних сигналів та внутрішніх станів подано в таблицях 16.17–16.19. Кодування, як й в попередніх прикладах, здійснюється довільно.

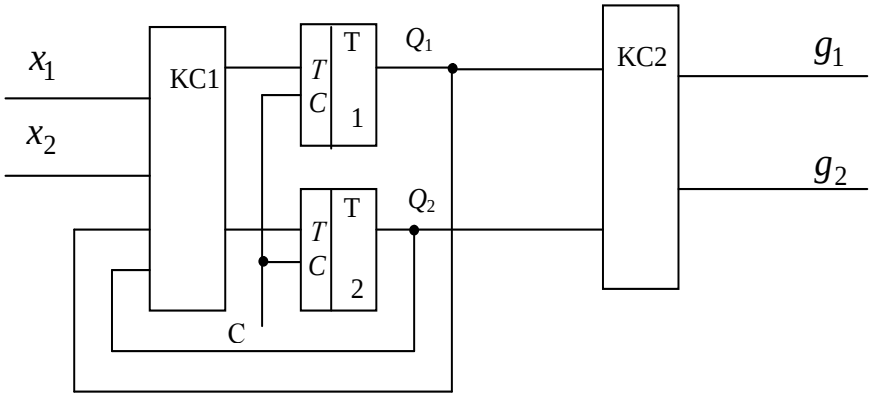


Рисунок 16.16. Структурна схема цифрового автомата Мура

Таблиця 16.17. Кодування вхідних сигналів

Вхідні сигнали	Код вхідних сигналів	
	x_1	x_2
z_1	0	0
z_2	0	1
z_3	1	0

Таблиця 16.18. Кодування станів автомата

Стани автомата	Код стану	
	Q_1	Q_2
a_1	0	0
a_2	0	1
a_3	1	0
a_4	1	1

Таблиця 16.19. Кодування вихідних сигналів

Вихідні сигнали	Код вихідних сигналів	
	g_1	g_2
u_1	0	0
u_2	0	0
u_3	0	1
u_4	0	1

4. Будуємо кодовану таблицю переходів структурного автомата (таблиця 16.20).

Таблиця 16.20. Кодована таблиця переходів автомата Мура

	00	01	10	11
Q_1Q_2 / x_1x_2	00	01	10	11
00	01	10	01	00
01	00	00	11	01
10	11	11	00	11

5. Знаходимо функції g_1 та g_2 з таблиці 16.20:

$$g_1 = Q_1\overline{Q}_2 + Q_1Q_2 = Q_1,$$

$$g_2 = \overline{Q}_1Q_2 + Q_1Q_2 = Q_2.$$

6. Знаходимо функції збудження T_1 та T_2.

Скориставшись таблицею функцій входів T-тригерів, знаючи код початкового стану автомата та код стану переходу, отримуємо необхідне значення функції збудження, що забезпечує заданий перехід (таблиця 16.21).

Таблиця 16.21. Закодована таблиця функцій збудження T-тригерів

x_1x_2 \ Q_1Q_2	00	01	10	11
00	01	11	11	11
01	00	01	01	10
10	11	10	10	00

З таблиці 16.21 записуємо вирази для функцій T_1 та T_2:

$$T_1 = x_1\bar{x}_2\overline{Q_1}\overline{Q_2} + \bar{x}_1\bar{x}_2\overline{Q_1}Q_2 + x_1\bar{x}_2\overline{Q_1}Q_2 + \bar{x}_1\bar{x}_2Q_1\overline{Q_2} + x_1\bar{x}_2Q_1\overline{Q_2} + \bar{x}_1\bar{x}_2Q_1Q_2 + \bar{x}_1x_2Q_1Q_2,$$

$$T_2 = \bar{x}_1\bar{x}_2\overline{Q_1}\overline{Q_2} + x_1\bar{x}_2\overline{Q_1}\overline{Q_2} + \bar{x}_1\bar{x}_2\overline{Q_1}Q_2 + \bar{x}_1x_2\overline{Q_1}Q_2 + \bar{x}_1\bar{x}_2Q_1\overline{Q_2} +$$
$$+ \bar{x}_1x_2Q_1\overline{Q_2} + \bar{x}_1\bar{x}_2Q_1Q_2.$$

Використаємо карти Карно (рисунки 16.17 та 16.18) для мінімізації цих функцій.

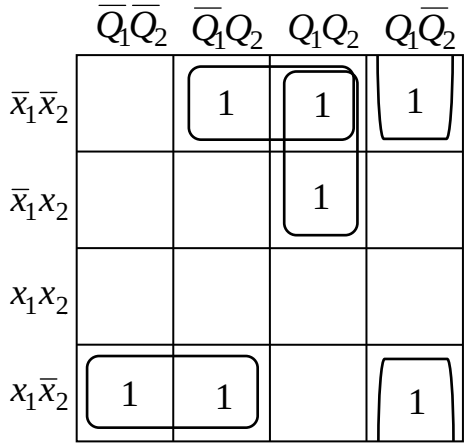


Рисунок 16.17. Мінімізація функції T_1

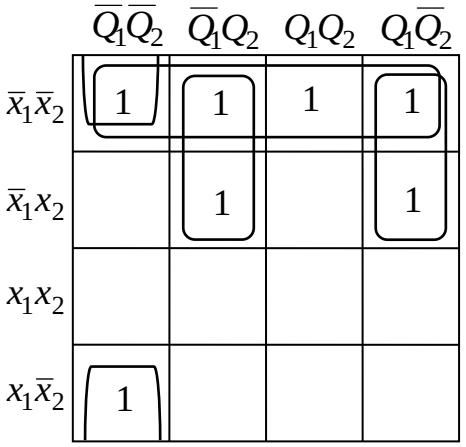


Рисунок 16.18. Мінімізація функції T_2

У результаті мінімізації отримаємо

$$T_1 = x_1\bar{x}_2\overline{Q_1} + \bar{x}_1\bar{x}_2Q_2 + \bar{x}_2Q_1\overline{Q_2} + \bar{x}_1Q_1Q_2,$$

$$T_2 = \bar{x}_2\overline{Q_1}\overline{Q_2} + \bar{x}_1\bar{x}_2 + \bar{x}_1\overline{Q_1}Q_2 + \bar{x}_1Q_1\overline{Q_2} = \bar{x}_2\overline{Q_1}\overline{Q_2} + \bar{x}_1\bar{x}_2 + \bar{x}_1(Q_1 \oplus Q_2).$$

7. На підставі отриманих у результаті синтезу виразів для T_1, T_2, g_1 та g_2 будуємо функціональну схему автомата Мура (рисунок 16.19).

Приклад 16.4. Виконати структурний синтез *C*-автомата S, заданого таблицею переходів (таблиці 16.22) та відміченою таблицею виходів (таблиця 16.23), використовуючи в якості пам'яті автомат R, функція переходів якого подана в таблиці 16.24.

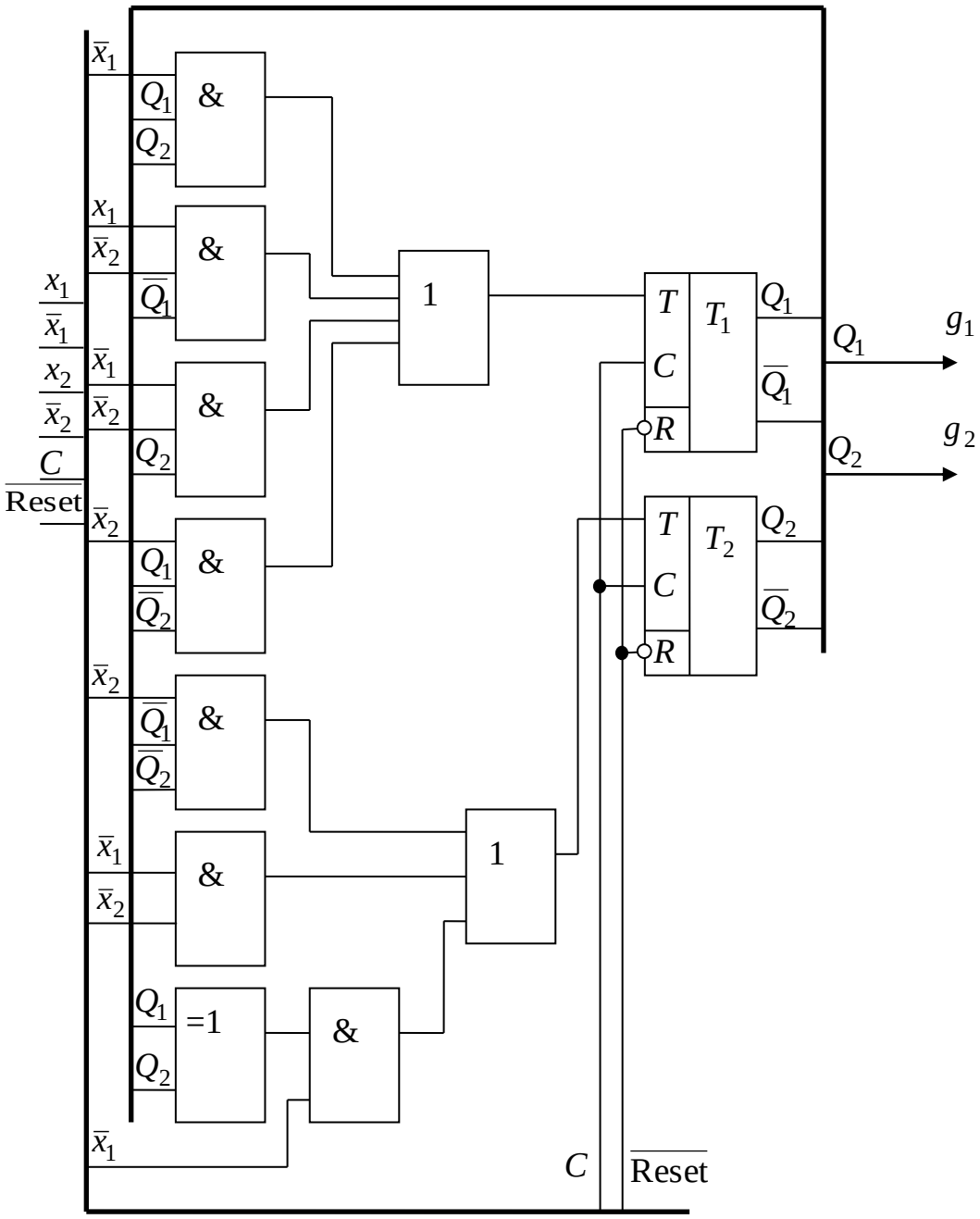


Рисунок 16.19. Функціональна схема автомата

Таблиця 16.22. Таблиця переходів частково визначеного C-автомата

	a_1	a_2	a_3	a_4
z_1	a_2	a_3	-	a_4
z_2	-	a_4	a_2	a_1
z_3	a_3	-	a_1	-

Таблиця 16.23. Відмічена таблиця виходів частково визначеного C-автомата

	u_1	u_2	u_2	u_1
	a_1	a_2	a_3	a_4
z_1	w_1	w_3	-	w_2
z_2	-	w_1	w_3	w_4

z_3	w_2	-	w_2	-

Таблиця 16.24. Таблиця переходів автомата R

	r_1	r_2
p_1	r_1	r_2
p_2	r_2	r_1

Структурний синтез *С*-автомата будемо виконувати так:

1. Визначаємо кількість вхідних та вихідних (двох типів) структурних сигналів, кількість елементів пам'яті автомата S, а також кількість вхідних та вихідних сигналів автомата R.

Кількість вхідних абстрактних сигналів автомата S дорівнює трьом, отже, кількість вхідних структурних сигналів $L = \log_2 3 = 2$, тобто x_1 та x_2.

Кількість вихідних абстрактних сигналів типу 1 автомата S дорівнює чотирьом, отже кількість вихідних структурних сигналів $N = \log_2 4 = 2$, тобто y_1 та y_2. Кількість вихідних абстрактних сигналів типу 2 автомата S дорівнює двом, отже отримаємо один цього типу вихідний структурний сигнал g_1.

Кількість внутрішніх станів абстрактного автомата S $M = 4$, отже кількість елементів пам'яті $R = \log_2 M = \log_2 4 = 2$.

Оскільки в абстрактному автоматі пам'яті R два вхідних та два вихідних сигнали, то в структурному автоматі R буде один вхід та один вихід.

2. Будуємо структурну схему цифрового автомата.

Структурна схема цифрового автомата з урахуванням того, що початковий автомат є *С*-автоматом, який містить вихідні структурні сигнали двох типів та два елементи пам'яті, представлена на рисунку 16.20.

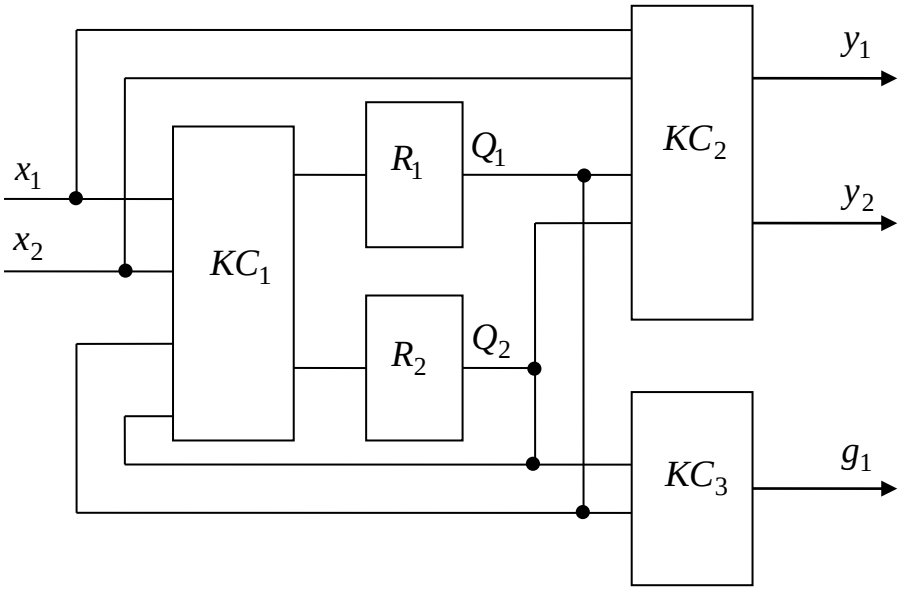


Рисунок 16.20. Структурна схема автомата

3. Кодуємо вхідні, вихідні сигнали та внутрішні стани автомата.

Кодування вхідних, вихідних сигналів та внутрішніх станів абстрактного автомата S подано в таблицях 16.25–16.28. Кодування, як й в попередніх прикладах, здійснюється довільно.

Після кодування вхідних сигналів (таблиця 16.29) та станів (таблиця 16.30) абстрактного автомата R його таблиця переходів перетвориться в таблицю 16.31. Функція входів структурного автомата R подана в таблиці 16.32.

Таблиця 16.25.
Кодування вхідних сигналів автомата S

Вхідні сигнали	Код вхідних сигналів	
	x_1	x_2
z_1	0	0
z_2	0	1
z_3	1	0

Таблиця 16.26.
Кодування станів автомата автомата S

Стани автомата	Код стану	
	Q_1	Q_2
a_1	0	0
a_2	0	1
a_3	1	0
a_4	1	1

Таблиця 16.27.
Кодування вихідних сигналів типу 1 автомата S

Вихідні сигнали	Код вихідних сигналів	
	y_1	y_2
w_1	0	0
w_2	0	1
w_3	1	0
w_4	1	1

Таблиця 16.28.
Кодування вихідних сигналів типу 2 автомата S

Вихідні сигнали	Код вихідних сигналів
	g_1
u_1	0
u_2	1

Таблиця 16.29.
Кодування вхідних сигналів автомата R

Вхідні сигнали	Код вхідних сигналів
	q
p_1	0
p_2	1

Таблиця 16.30.
Кодування станів автомата R

Стани автомата	Код стану
	Q
r_1	0
r_2	1

Таблиця 16.31. Закодована таблиця переходів автомата R

	0	1
0	0	1
1	1	0

Таблиця 16.32. Таблиця функцій входів структурного автомата R

Q^t	q	Q^{t+1}
0	0	0
0	1	1
1	1	0
1	0	1

4. Будуємо закодовану таблицю переходів (таблиця 7.33) та закодовану відмічену таблицю виходів (таблиця 7.34) структурного автомата S.

5. Знаходимо функції y_1, y_2 та g_1 з таблиці 7.34:

$$y_1 = \bar{x}_1\bar{x}_2\overline{Q}_1Q_2 + \bar{x}_1x_2Q_1\overline{Q}_2 + \bar{x}_1x_2Q_1Q_2,$$

$$y_2 = x_1\bar{x}_2\overline{Q}_1\overline{Q}_2 + x_1\bar{x}_2Q_1\overline{Q}_2 + \bar{x}_1\bar{x}_2Q_1Q_2 + \bar{x}_1x_2Q_1Q_2,$$

$$g_1 = \overline{Q}_1Q_2 + Q_1\overline{Q}_2 = Q_1 \oplus Q_2 .$$

Таблиця 16.33. Закодована таблиця переходів автомата S

x_1x_2 \ Q_1Q_2	00	01	10	11
00	01	10	-	11
01	-	11	01	00
10	10	-	00	-

Таблиця 16.34. Закодована відмічена таблиця виходів автомата S

	0	1	1	0
x_1x_2 \ Q_1Q_2	00	01	10	11
00	00	10	-	01
01	-	00	10	11
10	01	-	01	-

Вирази для y_1 та y_2 мінімізуємо за допомогою карт Карно (рисунки 16.21 та 16.22).

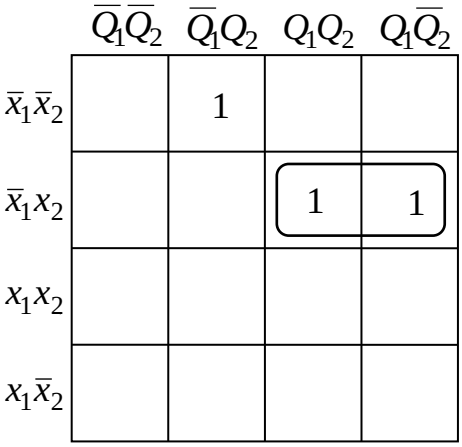


Рисунок 16.21. Мінімізація функції y_1

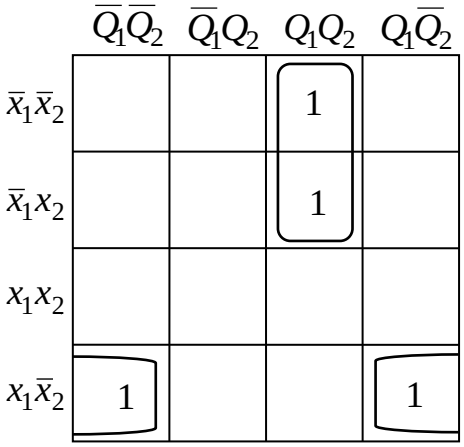


Рисунок 16.22. Мінімізація функції y_2

У результаті мінімізації отримаємо

$$y_1 = \bar{x}_1\bar{x}_2\overline{Q}_1Q_2 + \bar{x}_1x_2Q_1,$$

$$y_2 = x_1\bar{x}_2\overline{Q}_2 + \bar{x}_1Q_1Q_2.$$

6. Знаходимо функції збудження елементів пам’яті R_1 та R_2.

Скориставшись таблицею функцій входів структурного автомата R (таблиця 16.32), знаючи код початкового стану автомата та код стану переходу, отримуємо необхідне значення функції збудження, що забезпечує заданий перехід (таблиця 16.35).

Таблиця 16.35. Таблиця функцій збудження пам’яті автомата S

x_1x_2 \ Q_1Q_2	00	01	10	11
00	01	11	-	00
01	-	10	11	11
10	10	-	10	-

З таблиці 16.35 записуємо вирази для функцій R_1 та R_2:

$$R_1 = x_1\bar{x}_2\overline{Q}_1\overline{Q}_2 + \bar{x}_1\bar{x}_2\overline{Q}_1Q_2 + \bar{x}_1x_2\overline{Q}_1Q_2 + \bar{x}_1x_2Q_1\overline{Q}_2 + x_1\bar{x}_2Q_1\overline{Q}_2 + \bar{x}_1x_2Q_1Q_2,$$

$$R_2 = \bar{x}_1\bar{x}_2\overline{Q}_1\overline{Q}_2 + \bar{x}_1\bar{x}_2\overline{Q}_1Q_2 + \bar{x}_1x_2Q_1\overline{Q}_2 + \bar{x}_1x_2Q_1Q_2.$$

Проведемо мінімізацію отриманих виразів R_1 та R_2 (рисунки 16.23 та 16.24).

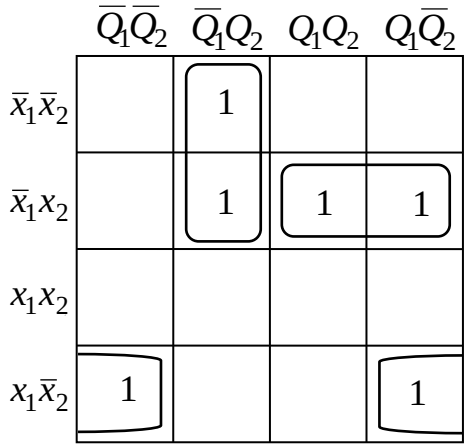


Рисунок 16.23. Мінімізація функції R_1

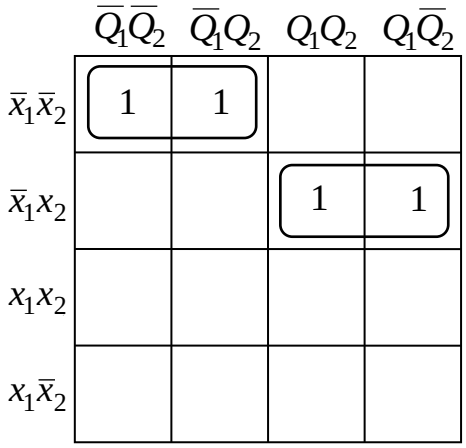


Рисунок 16.24. Мінімізація функції R_2

З наведених карт Карно отримаємо:

$$R_1 = \bar{x}_1\overline{Q}_1Q_2 + \bar{x}_1x_2Q_1 + x_1\bar{x}_2\overline{Q}_2,$$

$$R_2 = \bar{x}_1\bar{x}_2\overline{Q}_1 + \bar{x}_1x_2Q_1.$$

На підставі отриманих у результаті синтезу виразів R_1, R_2, y_1, y_2 та g_1 будуємо функціональну схему автомата (рисунок 16.25).

Контрольні запитання

1. Сформулюйте основну задачу теорії структурного синтезу цифрових автоматів.
2. Дайте означення поняттям «повнота системи переходів» та «повнота системи виходів» в автоматах Мура.
3. Сформулюйте теорему про структурну повноту.
4. Назвіть, з яких частин складається структурна схема автомата, синтезованого відповідно до канонічного методу структурного синтезу. Дайте характеристику цих частин.
5. Обґрунтуйте канонічний метод структурного синтезу цифрового

автомата на *D*-тригерах.

6. Наведіть особливості канонічного методу структурного синтезу цифрового автомата на базі *T, RS, JK* -тригерах.

7. Дайте означення поняттю «кодування внутрішніх станів автомата».

8. Наведіть алгоритм кодування абстрактних автоматів, побудованих на *D* -тригерах.

9. Назвіть етапи евристичного алгоритму кодування внутрішніх станів абстрактних автоматів.

10. Охарактеризуйте сусіднє кодування абстрактних автоматів із визначенням логічно суміжних станів.

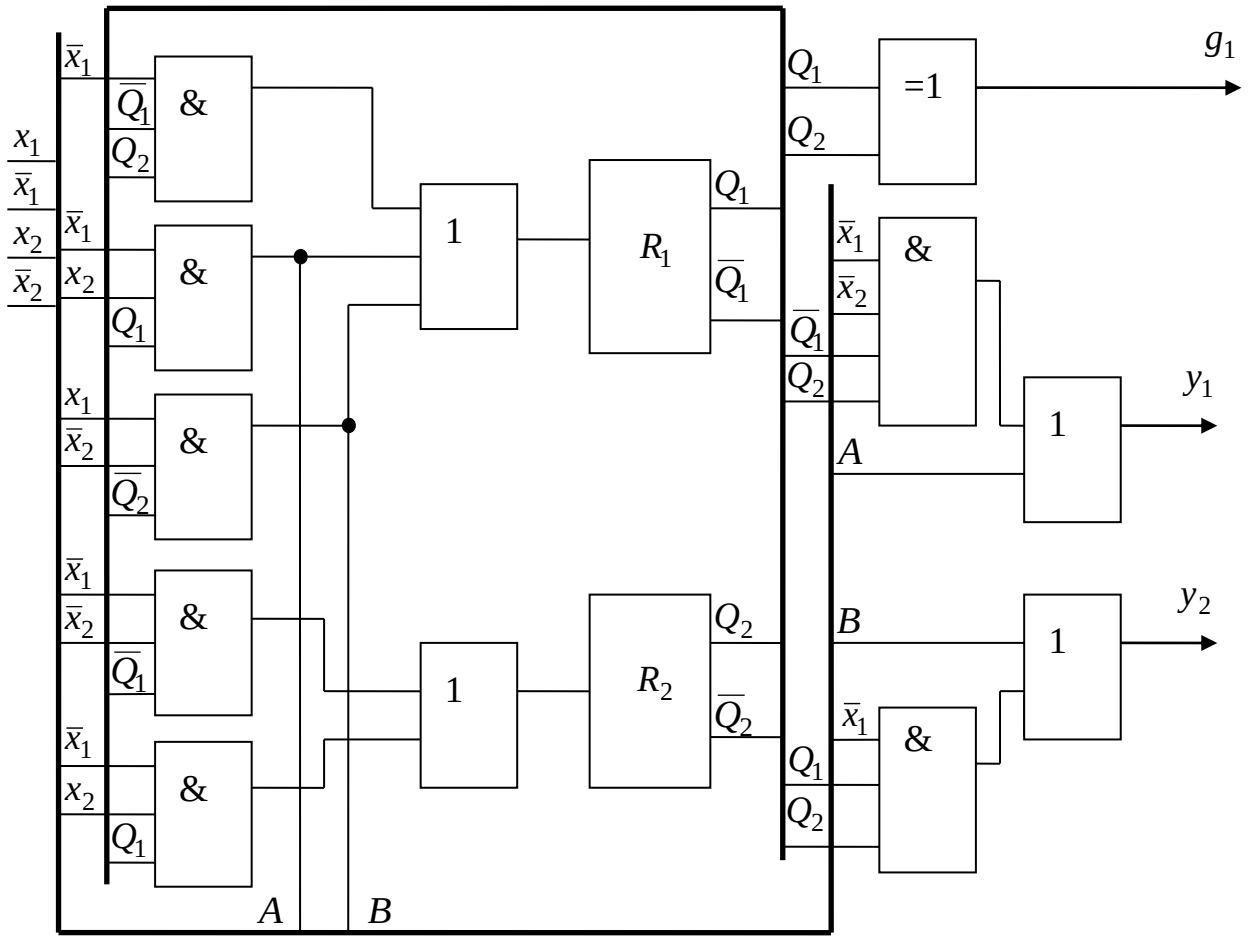


Рисунок 16.25. Функціональна схема автомата

ВИКОРИСТАНІ ЛІТЕРАТУРНІ ДЖЕРЕЛА

1. Баранов, С.И. Синтез микропрограммных автоматов (граф-схемы и автоматы). Издание второе, переработанное и дополненное [Текст] / С.И. Баранов. – Ленинград : Энергия, 1979. – 232 с.
2. Бондаренко, М.Ф. Комп'ютерна дискретна математика: підручник [Текст] / М.Ф. Бондаренко, Н.В. Білоус, А.Г. Руткас. – Харків: Компанія СМІТ, 2004. – 480с.
3. Брауэр, В. Введение в теорию конечных автоматов [Текст] / В. Брауэр; перевод с английского под редакцией Ю.И. Журавлева. – М.: Радио и связь, 1987. – 392с.
4. Воробйова, О.М. Основи схемотехніки: у двох частинах: навч. посібник [Текст] / О.М. Воробйова, В.Д. Іванченко. – Одеса: ОНАЗ ім. О.С. Попова, 2004. – Ч. 2. – 172с.
5. Глушков, В.М. Синтез цифровых автоматов [Текст] / В.М. Глушков. – М.: Государственное издательство физико-математической литературы, 1962.
6. Голдсуорт, Б. Проектирование цифровых логических устройств [Текст] / Б. Голдсуорт. – М.: Машиностроение, 1985. – 288 с.
7. Дмитриев, В.И. Прикладная теория информации [Текст] / В.И. Дмитриев. – М.: Высшая школа, 1989. – 319 с.
8. Жмакин, А.П. Архитектура ЭВМ. СПб.: БХВ-Петербург, 2006. – 320 с.
9. Жураковський, Ю.П. Теорія інформації та кодування: підручник [Текст]/ Ю.П. Жураковський, В.П. Полторак. – К.: Вища шк., 2001. – 255 с.
10. Емеличев, В.А. Лекции по теории графов [Текст] / В.А. Емеличев, О.И. Мельников. – М.: Наука, 1990.
11. Закревский, А.Д. Алгоритмы синтеза дискретных автоматов [Текст] / А.Д. Закревский. – М.: Наука, 1971. – 512 с.
12. Захаров, Н.Г. Синтез цифровых автоматов: учебное пособие [Текст] / Н.Г. Захаров, В.Н. Рогов. – Ульяновск: УлГТУ, 2003.
13. Каган, Б.М. Электронные вычислительные машины и системы: Учеб. пособие для вузов. – 3-е изд., перераб. и доп. [Текст] / Б.М. Каган – М.: энергоатомиздат, 1991. – 592с.
14. Карпов, Е.А. Теория автоматов [Текст] / Е.А. Карпов. – СПб.: Питер, 2003. – 208 с.
15. Кочубуй, О.О., Прикладна теорія цифрових автоматів. Логічні основи [Текст] / О.О. Кочубуй, О.В. Сопільник. – Видавництво Дніпропетровського університету, 2009. - 264 с.
16. Кузьмин, И.В. Основы теории информации и кодирования [Текст] / И.В. Кузьмин, В.А. Кедрус. – К.: Вища шк. Головное изд-во, 1986. – 238 с.

17. Липкин, И.А. Статистическая радиотехника. Теория информации и кодирования [Текст] / И.А. Липкин – М.: «Вузовская книга», 2002. – 216 с.

18. Лазарев, В.Г. Синтез управляющих автоматов [Текст] / В.Г. Лазарев, Е.И. Пийль. – М.: Энергоатомиздат, 1989. – 328 с.

19. Лупал, А.М. Теория автоматов: учеб. пособие [Текст] / А.М. Лупал. – СПб.: СПбГУАП, 2000. – 119 с.

20. Лысиков, Б.Г. Арифметические и логические основы цифровых автоматов [Текст] / Б.Г. Лысиков — Мн.: Выш. школа, 1980. – 336 с.

21. Майоров, С.А. Структура электронных вычислительных машин [Текст] / С.А. Майоров, Г.И. Новиков. – Л.: Машиностроение, Ленингр.отд-ие, 1979.

22. Микросхемы и их применение: Справочное пособие / В.А. Батушев, В.Н. Вениаминов, В.Г. Ковалев, О.Н. Лебедев. А.И. мирошниченко. – 2-е изд., перер. и доп. – М.: Радио и связь, 1983. – 272 с.

23. Наумкина, Л.Г. Цифровая схемотехника. Конспект лекций по дисциплине «Схемотехника» [Текст] / Л.Г. Наумкина. – М.: Издательство «Горная книга», Издательство МГГУ, 2008. – 308 с.

24. Нікольський, Ю.В. Дискретна математика: підручник [Текст] / Ю.В. Нікольський, В.В. Пасічник, Ю.М. Щербина. – Львів: Магнолія 2006, 2007. – 608с.

25. Основы технической диагностики. В 2-х книгах. Кн.1. модели объектов, методы и алгоритмы диагностики. Под ред. П.П. Пархоменко. М., «Энергия», 1976. – 464 с.

26. Поспелов, Д.А. Логические методы анализа и синтеза схем [Текст] / Д.А. Поспелов. – М.: Энергия, 1974.

27. Пухальский, Г.И. Проектирование дискретных устройств на интегральных микросхемах: справочник [Текст] / Г.И. Пухальский, Т.Я. Новосельцева. – М.: Радио и связь, 1990. – 304 с.

28. Пухальский, Г.И. Цифровые устройства: Учебное пособие для вузов. [Текст] / Г.И. Пухальский, Т.Я. Новосельцева. – СПб.:Политехника,1996.– 885с.

29. Савельев, А.Я. Основы информатики: учеб. для вузов [Текст] / А.Я. Савельев. – М.: Изд-во МГТУ им. Н. Э. Баумана, 2001. – 328 с.

30. Савельев, А.Я. Прикладная теория цифровых автоматов: учебник для вузов [Текст] / А.Я. Савельев. – М.: Высшая школа, 1987. – 272с.

31. Самофалов, К.Г. Прикладная теория цифровых автоматов [Текст] / К.Г. Самофалов. – Киев: Высшая школа, 1987.

32. Угрюмов, Е.П. Цифровая схемотехника [Текст] / Е.П. Угрюмов – СПб.: БХВ-Петербург, 2004, - 528 с.

33. Уилкинсон, Б. Основы проектирования цифровых схем [Текст] / Б. Уилкинсон. – М.: Издательский дом «Вильямс», 2004. – 320 с.

34. Уэйкерли, Дж.Ф. Проектирование цифровых устройств: В 2 т., пер. с англ. [Текст] /Дж.Ф. Уэйкерли – М.: Постмаркет, 2002.

35. Фридман, А. Теория и проектирование переключательных схем [Текст] / А. Фридман, П. Менон. – М.: Мир, 1978.

36. Хэмминг, Р.В. Теория кодирования и теория информации [Текст] / Р.В. Хэмминг. – М.: Радио и связь, 1983. – 176 с.

37. Хопкрофт, Д. Введение в теорию автоматов, языков и вычислений, 2-е изд.: пер. с англ. [Текст] / Д. Хопкрофт, Р. Мотванн, Д. Ульман. – М.: Вильямс, 2002. – 528 с.

38. Шульгин, В.И. Основы теории передачи информации. Ч. 2. помехоустойчивое кодирование. Учеб.пособие. [Текст] – Харьков: Нац. Аэрокосм. Ун-т «Харьк. авиац. ин-т», 2003. – 87 с.

39. Якубовский, С.В. Аналоговые и цифровые интегральные схемы [Текст] / С.В. Якубовский. – М.: Советское радио, 1979.