

ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ

ФАКУЛТЕТ ПО КОМПЮТЪРНИ СИСТЕМИ И УПРАВЛЕНИЕ

гл.ас. Диана В. Григорова, доц. Валентин С. Моллов

АНАЛИЗ И СИНТЕЗ НА ЛОГИЧЕСКИ СХЕМИ

Издателство на Техническия университет - София

2009

СЪДЪРЖАНИЕ

ТЕМИ	стр.
ПРЕДГОВОР	7
Т Е М А 1	
БУЛЕВИ ФУНКЦИИ. ЗАДАВАНЕ НА ЛОГИЧЕСКИ ФУНКЦИИ. КАНОНИЧНИ ФОРМИ НА ЛОГИЧЕСКИ ФУНКЦИИ	9
1.1 Булеви функции. Основни понятия, означения, логически операции	9
1.2 Функции на една и две променливи	10
1.3 Правила и закони на булевата алгебра	12
1.4 Задаване на логически функции	15
1.5 Канонични форми на логически функции	17
1.6 Преминаване от нормални към канонични форми на логически функции	20
Т Е М А 2	
МЕТОД НА КУАЙН-МАК КЛАСКИ ЗА МИНИМИЗАЦИЯ НА ЛОГИЧЕСКИ ФУНКЦИИ. МИНИМИЗАЦИЯ НА ЛОГИЧЕСКИ ФУНКЦИИ С КАРТИ НА ВЕЙЧ. МИНИМИЗАЦИЯ НА СИСТЕМА ЛОГИЧЕСКИ ФУНКЦИИ	23
2.1 Метод на Куайн-Мак Класки	23
2.2 Минимизация с карти на Вейч	26
2.3 Минимизация на непълно определени логически функции	28
2.4 Минимизация на система логически функции	28
2.4.1 Минимизация на система логически функции с обща подфункция	28
2.4.2 Систематичен подход за минимизация на система логически функции	30
2.4.3 Минимизация на система функции с базова функция	33

ТЕМА 3		
	СТАТИЧЕН И ДИНАМИЧЕН АНАЛИЗ НА КОМБИНАЦИОННИ ЛОГИЧЕСКИ СХЕМИ. ВИДОВЕ СЪСТЕЗАНИЯ НА СИГНАЛИТЕ. ОТКРИВАНЕ И ОТСТРАНЯВАНЕ НА СЪСТЕЗАНИЯ	37
3.1	Статичен анализ	37
3.2	Динамичен анализ	38
3.3	Видове състезания	40
3.4	Откриване и отстраняване на състезанията	40

ТЕМА 4		
	ДЕШИФРАТОРИ И ДЕМУЛТИПЛЕКСОРИ. МУЛТИПЛЕКСОРИ. ПОСТОЯННИ ПАМЕТИ, ПРОГРАМИРУЕМИ ЛОГИЧЕСКИ МАТРИЦИ. ШИФРАТОРИ. КОМПАРАТОРИ. КОДОВИ ПРЕОБРАЗОВАТЕЛИ. СУМАТОРИ	45
4.1	Дешифратори и демултиплексори. Реализация на логически функции чрез дешифратори	45
4.2	Мултиплексори	49
4.3	Постоянни памет. Програмируеми логически матрици. Реализация на логически функции с ПЛМ	52
4.4	Шифратори	58
4.5	Цифрови компаратори	58
4.6	Кодови преобразуватели	59
4.7	Суматори	61

ТЕМА 5		
	ПОСЛЕДОВАТЕЛНОСТНИ СХЕМИ. СТРУКТУРНИ МОДЕЛИ. ЕЛЕМЕНТИ ПАМЕТ. СИНТЕЗ И АНАЛИЗ НА КРАЙНИ АВТОМАТИ С ПАМЕТ	65
5.1	Последователностни схеми: структура, особености, автоматни модели	65
5.2	Задаване на последователностни схеми	67

5.2.1	Таблично задаване	67
5.2.2	Графично задаване	68
5.3	Структурни модели на ПС	69
5.3.1	Структурен модел без елементи памет	69
5.3.2	Структурен модел с единичен блок памет	71
5.3.3	Структурен модел с двоен блок памет	72
5.4	Елементарни автомати	73
5.5	Синтез на автомати с памет	76
5.5.1	Структурен синтез на автомати със синхронизирани ЕП	76
5.5.2	Особености на синтеза на частични автомати с памет	78
5.6	Анализ на автомати със синхронизирани елементи памет	81

ТЕМА 6		
	БРОЯЧИ – ФУНКЦИИ, ВИДОВЕ, СИНТЕЗ НА ПЪЛНИ И ЧАСТИЧНИ БРОЯЧИ. БРОЯЧИ В ИНТЕГРАЛНО ИЗПЪЛНЕНИЕ. РЕГИСТРИ - ФУНКЦИИ, ВИДОВЕ, СИНТЕЗ. РЕГИСТРИ В ИНТЕГРАЛНО ИЗПЪЛНЕНИЕ	85
6.1	Броячи	85
6.1.1	Начин на функциониране	85
6.1.2	Видове броячи	85
6.1.3	Синтез на броячи	86
6.1.4	Броячи в интегрално изпълнение	91
6.2	Регистри	92
6.2.1	Начин на функциониране и видове	92
6.2.2	Синтез на регистри	93
6.2.3	Регистри в интегрално изпълнение	96

ТЕМА 7		
	СИНТЕЗ НА МИКРОПРОГРАМНИ АВТОМАТИ	99
7.1	Управляващи и операционни автомати	99
7.2	Задаване и синтез на микропрограмни автомати	100
7.3	Замяна на входните променливи	103

7.4	Декомпозиция на матриците на микропрограмен автомат	104
7.5	Декомпозиция на микропрограмни автомати	106
	Приложение – карти на Вейч на 2,3,4,5 и 6 променливи	113
	Литература	115

ПРЕДГОВОР

Учебното пособие е предназначено за студенти и специалисти, обучаващи се и работещи в областта на компютърните науки и технологии. То има за цел да даде базови знания и умения при решаване на широк кръг задачи с използване на логически елементи и базирани на тях разнообразни комбинационни структури, както и при анализа и синтеза на крайни автоматни устройства.

В изданието се разглеждат елементи от теорията на булевата алгебра и приложението ѝ при описание работата на логически елементи и схеми. Дискутират се методите за минимизация на пълно и непълно зададени логически функции на 2,3,4 и повече променливи, както и на система от логически функции. Показани са техниките за преминаване от нормални към канонични форми на логически функции. Описан е методът на Куайн-Мак Класки за минимизация на функции.

В книгата са представени основите на статичния и динамичния анализ на схемни структури с логически елементи, явлението „състезание на сигнали“ и методите за отстраняването им. Представени са най-често използваните комбинационни схемни структури в инженерната практика – дешифратори, мултиплексори, демултиплексори, преобразуватели на код, шифратори, цифрови компаратори и др. Разгледани са структурата и особеностите на постоянни запомнящи устройства и програмируеми логически матрици (ПЛМ) в аспект на използването им за реализация на сложни логически структури с елементно излишество. Дискутирани са структурните особености на всяко комбинационно устройство и спецификите в приложението му.

Специално внимание е отделено на методите за синтез на последователностни схеми с елементи памет. Дадени са основните модели на този клас схеми и основните видове елементарни автомати (тригери). Подробно са описани синтезът и анализът на синхронни крайни автомати с елементи памет.

Разглеждат се най-често използваните електронни устройства, изградени с тригери – броячи и регистри. Дадени са техните функционални особености, начини за управление, видове, интегрално изпълнение. Дискутират се методите за синтез на такива устройства.

Книгата дава базови знания за същността, задаването и синтеза на микропрограмни автомати като пример на крайни инженерни устройства

с използване на крайни автоматни устройства с микропрограмно управление.

Авторите изказват своята благодарност на рецензента – доц. д-р инж. Николай Г. Николов за неограничената му помощ при изготвяне на ръкописа и множеството полезни препоръки.

Материалът в това учебно пособие е изготвен от съавторите, както следва:

- Глави 2,3,5,6,7 – от инж. Диана В. Григорова
- Глави 1,4, предговор – от д-р инж. Валентин С. Моллов.

Т Е М А 1

БУЛЕВИ ФУНКЦИИ. ЗАДАВАНЕ НА ЛОГИЧЕСКИ ФУНКЦИИ. КАНОНИЧНИ ФОРМИ НА ЛОГИЧЕСКИ ФУНКЦИИ

1.1 Булеви функции. Основни понятия, означения, логически операции

Булевите (логически) функции¹, както и съответните променливи и константи, които ги определят, могат да заемат само две фиксирани стойности – „0“ и „1“, и представляват двете възможни състояния, които може да заема един цифров сигнал с две нива на кодиране – ниско/високо, истина/неистина (true/false). Поради това дефиниращите логическите функции и техните аргументи се наричат още *двоични*.

Както при всяка алгебрична система, така и в булевата алгебра се изисква задаване на множеството от математически обекти (константи, променливи, изрази) и съответните правила, операции и закони, с които тези обекти ще се преобразуват. Математическият апарат, с който работи булевата алгебра, дефинира своите специфични означения, операции, правила и закони, които дават възможност за представяне на логическите функции в различна форма, както и за преобразуването им от една форма в друга.

Прието е променливите да се означават с малки букви (със или без индекс): x_1, x_5, y_1, z_3 и т.н. или с първите главни букви от латинската азбука: A, B, C, D, докато логическите функции – най-често с букви F, Y, Z или φ .

Съвкупността от всички възможни комбинации от стойности на аргументите на дадена логическа функция се наричат *набори*. Ако една функция има n аргумента, то броят на наборите ѝ е $N=2^n$, а общият брой на логическите функции на n променливи е $M=2^N=2^{2^n}$. Така например, ако Y е функция на четири аргумента: x_1, x_2, x_3 и x_4 , то броят на наборите ѝ е $N=2^4=16$, като всеки набор се записва директно в двоичен вид като последователност от нули и единици. Прието е номерът на съответния набор да се указва в скоби като десетично число, напр. 0010 (2), 0110 (6), 1110 (13) и т.н. Определянето на десетичния номер K на поредния набор става, като се отчете двоичното „тегло“ на съответния аргумент в зависимост от мястото му в записа и неговата стойност. За случай на n аргумента, представени в двоичен запис като $x_{n-1}, x_{n-2}, \dots, x_1, x_0$ десетичният еквивалент се получава чрез $K = x_{n-1} \cdot 2^{n-1} + x_{n-2} \cdot 2^{n-2} + \dots + x_1 \cdot 2^1 + x_0 \cdot 2^0$. Прието е също така най-дясно стоящият в двоичния запис

¹ Математическият апарат на булевата алгебра е формулиран от George Boole (1815-1864г.) през 1849 г. и намира за пръв път практическо инженерно приложение през 1938г. от Claude Shannon при проектиране на телефонни комутационни вериги.

аргумент (бит) да е този с най-малко тегло и да се дефинира като "младши" или с "най-малко значимост" (*the least significant bit*, LSB), докато най-ляво разположеният, с най-голямо тегло – като "старши" или с "най-голяма значимост" (*the most significant bit*, MSB).

Една логическа функция може да бъде пълно или непълно определена. Казва се, че функцията е *пълно определена*, ако тя има фиксирана стойност 0 или 1 за всички свои набори. Тя е *непълно определена*, ако за част от наборите си стойността ѝ е недефинирана. В тези случаи стойността ѝ се означава с X или H. В общия случай, при непълно определена функция, по отношение на входните набори различаваме три непресичащи се подмножества: подмножество от наборите, за които функцията има стойност 1, такова, за което тя приема стойност 0, и подмножество от наборите, за което функцията е неопределена.

Основните операции в булевата алгебра са: *дизюнкция* (логическо събиране, функция ИЛИ), *конюнкция* (логическо умножение, функция И) и *инверсия* (логическо отрицание, функция НЕ). Тези основни логически операции, съответните им означения и схемни елементи, които ги представят (за случая на две входни променливи x и y), са дадени в Табл.1.1:

Основни логически операции	Означения	Наименование	Схемен елемент
ИЛИ (OR)	$Y = x \vee y, Y = x + y$	дизюнкция (логическо събиране)	
И (AND)	$Y = x \wedge y, Y = x \cdot y$ $Y = x \& y$	конюнкция (логическо умножение)	
НЕ (NOT)	$Y = \bar{x}, Y = \sim x, Y = \neg x$	инверсия / инвертиране (логическо отрицание)	

Табл.1.1

1.2 Функции на една и две променливи

При използване апарата на булевата алгебра в практиката голямо значение имат функциите на една и две променливи – т.нар. *елементарни логически функции*. Поради важността им е прието те да се означават по определен начин и да се именуват. В Табл.1.2 са дадени функциите (f_0 до f_3) на една променлива ($n=1$), както споменахме по-горе – общо $M=2^{2^1}=4$.

Броят функции на два аргумента ($n=2$) е общо $M=2^{2^2}=2^4=16$. Както може да се види от Табл.1.3, всяка от тези логически функции (f_0 до f_{15}) има по една инверсна на себе си: функциите f_0 до f_7 и функциите f_8 до f_{15} са взаимноинверсни. Поради важността им всяка от тях носи специфично наименование, а за някои от тях има и съпоставен съответен схемен логически елемент. Особена важност при анализа и синтеза на логически схеми и устройства имат (освен посочените в Табл.1.1. операции конюнкция, дизюнкция и инверсия) логическите функции ИЛИ-НЕ /стрелка на Пирс/, И-НЕ /щрих на Шефер/, сума по модул две и логическа равнозначност.

Всяка съвкупност от логически функции, чрез които може да се представи произволна логическа функция, представлява т.нар. *функционално пълна система функции* или *логически базис* (или само *базис*). Доказва се, че трите функции И, ИЛИ, НЕ образуват функционално пълна система. Логически базис също така образуват всяка една от функциите ИЛИ-НЕ и И-НЕ. Това именно обяснява широкото използване на съответните логически елементи.

$\begin{matrix} x_1 \\ x_2 \\ f_i \end{matrix}$	0	0	1	1	Наименование	Логическа функция	Логически елемент
f_0	0	0	0	0	Константа 0	0	-
f_1	0	0	0	1	Конюнкция (логическо умножение)	$x_1 \cdot x_2$	
f_2	0	0	1	0	Забрана по x_2	$x_1 \cdot \bar{x}_2$ $\overline{x_1 \rightarrow x_2}$	
f_3	0	0	1	1	Променлива x_1	x_1	-
f_4	0	1	0	0	Забрана по x_1	$\bar{x}_1 \cdot x_2$ $\overline{x_1 \leftarrow x_2}$	
f_5	0	1	0	1	Променлива x_2	x_2	-
f_6	0	1	1	0	Сума по модул 2	$x_1 \bar{x}_2 \vee \bar{x}_1 x_2$ $x_1 \oplus x_2$	

Табл.1.2

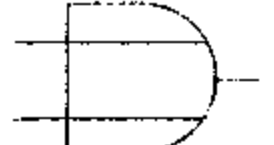
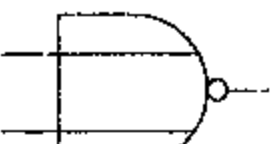
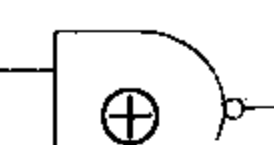
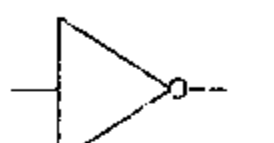

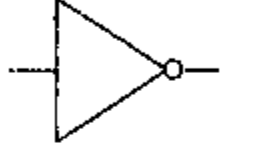
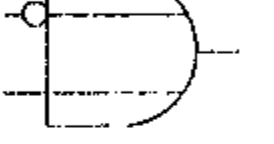
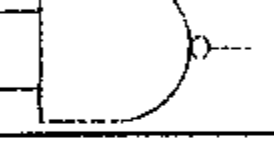
f ₇	0	1	1	1	Дизюнкция (логическо събиране)	$x_1 \vee x_2$	
f ₈	1	0	0	0	ИЛИ-НЕ (стрелка на Пирс)	$\overline{x_1 \vee x_2} \quad x_1 \downarrow x_2$	
f ₉	1	0	0	1	Логическа равнозначност	$\overline{x_1} \overline{x_2} \vee x_1 x_2$ $x_1 \sim x_2$	
f ₁₀	1	0	1	0	Инверсия на x ₂	$\overline{x_2}$	
f ₁₁	1	0	1	1	Импликация на x ₂ към x ₁	$x_1 \vee \overline{x_2}$ $x_1 \leftarrow x_2$	
f ₁₂	1	1	0	0	Инверсия на x ₁	$\overline{x_1}$	
f ₁₃	1	1	0	1	Импликация на x ₁ към x ₂	$\overline{x_1} \vee x_2$ $x_1 \rightarrow x_2$	
f ₁₄	1	1	1	0	И-НЕ (щрих на Шефер)	$\overline{x_1 \cdot x_2} \quad x_1 \mid x_2$	
f ₁₅	1	1	1	1	Константа 1	1	-

Табл.1.3

1.3 Правила и закони на булевата алгебра

Правилата за извършване на основните операции над булеви променливи се задават чрез *постулати*, определящи изпълнението им по отношение на логическите нула и единица – Табл.1.4:

$0 \vee 0 = 0$	$0 \cdot 0 = 0$	$0 = \overline{1}$
$0 \vee 1 = 1$	$0 \cdot 1 = 0$	$1 = \overline{0}$
$1 \vee 0 = 1$	$1 \cdot 0 = 0$	
$1 \vee 1 = 1$	$1 \cdot 1 = 1$	

Табл.1.4

При извършване на логическите операции, разгледани по-горе, булевата алгебра определя т.нар. *приоритет*, т.е. ред за извършването им, както следва:

логическо отрицание (1) →
логическо умножение (2) →
логическо събиране (3)

Въз основа на дадените по-горе постулати могат да се изведат следните обобщени зависимости (*свойства*) за произволна логическа променлива. Те представят действието на основните

$x \vee 0 = x$	$x \cdot 0 = 0$	$\overline{\overline{x}} = x$
$x \vee 1 = 1$	$x \cdot 1 = x$	
$x \vee x = x$	$x \cdot x = x$	
$x \vee \overline{x} = 1$	$x \cdot \overline{x} = 0$	

Табл.1.5

логически операции за произволна булева променлива по отношение на логическите нула и единица, както и по отношение на самата себе си и инверсната си стойност (в дуалните си форми И, ИЛИ) – Табл.1.5.

Законите на булевата алгебра са: *комутативен* (за разместване), *асоциативен* (за съчетаване) и *дистрибутивен* (за разпределение), закон за *слепване*, закон за *поглъщане*, закон(и)/правила на *Де Морган* (за инвертирането). Те са представени, за случаите на две (респ. три) променливи в Табл.1.6.

Наименование	Представяне	
	ИЛИ-форма	И-форма
Комутативен закон	$x_1 \vee x_2 = x_2 \vee x_1$	$x_1 \cdot x_2 = x_2 \cdot x_1$
Асоциативен закон	$x_1 \vee (x_2 \vee x_3) = (x_1 \vee x_2) \vee x_3$ $= x_3 \vee (x_1 \vee x_2)$	$x_1 \cdot (x_2 \cdot x_3) = (x_1 \cdot x_2) \cdot x_3$ $= x_3 \cdot (x_1 \cdot x_2)$
Дистрибутивен закон	$x_1 \vee x_2 x_3 = (x_1 \vee x_2)(x_1 \vee x_3)$	$x_1(x_2 \vee x_3) = x_1 x_2 \vee x_1 x_3$
Закон за слепване	$x_1 x_2 \vee x_1 \overline{x_2} = x_1$	$(x_1 \vee x_2)(x_1 \vee \overline{x_2}) = x_1$
Закон за поглъщане	$x_1 \vee x_1 x_2 = x_1$	$x_1(x_1 \vee x_2) = x_1$
Закон на Де Морган	$\overline{x_1 \vee x_2} = \overline{x_1} \cdot \overline{x_2}$	$\overline{x_1 \cdot x_2} = \overline{x_1} \vee \overline{x_2}$

Табл.1.6

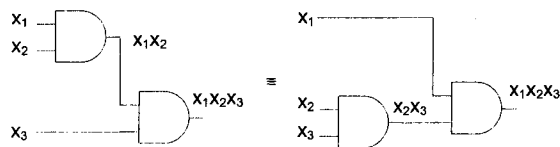
Както може да се види от Табл.1.4, голяма част от законите на булевата алгебра съответстват директно на законите и правилата от обикновената алгебра. Друга част са специфични, уникални и както ще видим по-нататък – имат съществена роля при извършване на преобразувания и опростяване на булеви изрази.



фиг.1.1 Схемна интерпретация на комутативен закон в булевата алгебра (за случай на двуходов елемент И-НЕ)

Комутативният закон представлява познатото ни от небулевата алгебра правило за равнозначност на израза при разместване местата на аргументите, независимо от техния брой. Специфичният смисъл на този закон при изразяването му с помощта на логически елементи е, че стойността на функцията в изхода на елемента (високо или ниско ниво) не зависи от това кой аргумент (сигнал) на кой конкретен вход е подаден – фиг.1.1.

Асоциативният закон интерпретира равнозначността на крайния резултат при различно съчетаване (присвояване) на резултата от междинните логически операции. Схемната му интерпретация е дадена на фиг.1.2.



фиг.1.2 Представяне на асоциативен закон (И-форма) в булевата алгебра с помощта на логически елементи

Дистрибутивният закон изразява “разпределението” на една променлива (в случая x_1) върху другите променливи в израза (в случая x_2 и x_3). Както може да се забележи, този закон е представен в смесен базис (И/ИЛИ) и в двете му форми. Вижда се, че лявата форма (условно наречена ИЛИ) на закона няма аналог в небулевата алгебра.

Законите за слепване и поглъщане се доказват по следния начин:

- закон за слепване: $x_1 x_2 \vee x_1 \bar{x}_2 = x_1 (x_2 \vee \bar{x}_2) = x_1 \cdot 1 = x_1$ (вж. Табл.1.5);

$$(x_1 \vee x_2)(x_1 \vee \bar{x}_2) = x_1 x_1 \vee x_1 \bar{x}_2 \vee x_2 x_1 \vee x_2 \bar{x}_2 = \\ = x_1 \vee x_1 (\bar{x}_2 \vee x_2) \vee 0 = x_1 \vee x_1 \cdot 1 = x_1;$$

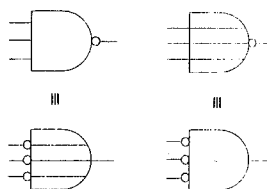
- закон за поглъщането: $x_1 \vee x_1 x_2 = x_1 (1 \vee x_2) = x_1 \cdot 1 = x_1$ (вж. Табл.1.5);

$$x_1 (x_1 \vee x_2) = x_1 x_1 \vee x_1 x_2 = x_1 \vee x_1 x_2 = \\ = x_1 (1 \vee x_2) = x_1$$

Законът (законите) на Де Морган са особено важни в булевата алгебра и третират преобразуването на инверсни изрази от аргументи, свързани помежду си само с операция И или само с операция ИЛИ. Общият вид на законите (в двете им форми) е следният:

$$\overline{x_1 \vee x_2 \vee \dots \vee x_n} = \bar{x}_1 \cdot \bar{x}_2 \cdot \dots \cdot \bar{x}_n$$

$$\overline{x_1 \cdot x_2 \cdot \dots \cdot x_n} = \bar{x}_1 \vee \bar{x}_2 \vee \dots \vee \bar{x}_n$$



фиг.1.3 Представяне дуалността на законите на Де Морган чрез логически елементи

Накратко: законът на Де Морган определя, че инверсен израз от аргументи се преобразува в израз от инверсни аргументи, като се сменя типът на логическата операция, т.е. инверсен израз от конюнкции се преобразува в дизюнкция от инверсните му аргументи и обратно - инверсен израз от

дизюнкции се преобразува в конюнкция от инверсните участващи аргументи.

Законът на Де Морган изразява принципа на дуалност, а именно че функциите И и ИЛИ могат взаимно да се заместят, при положение че входните променливи се инвертират – фиг.1.3.

По-долу са представени примери, в които се прилагат постулатите, свойствата и законите на булевата алгебра при опростяване на логически изрази:

- Да се опрости логическият израз: $F = (x \vee \bar{y} \vee \bar{z})(x \vee \bar{y} \cdot z)$.

Решение:

$$F = (x \vee \bar{y} \vee \bar{z})(x \vee \bar{y} \cdot z) = x \cdot x \vee x \cdot \bar{y} \cdot z \vee \bar{y} \cdot x \vee \bar{y} \cdot \bar{y} \cdot z \vee \bar{z} \cdot x \vee \bar{z} \cdot \bar{y} \cdot z = \\ = x \vee x \cdot \bar{y} \cdot z \vee \bar{y} \cdot x \vee \bar{y} \cdot z \vee \bar{z} \cdot x = x(1 \vee \bar{y} \cdot z) \vee x \cdot \bar{y} \vee \bar{y} \cdot z \vee \bar{z} \cdot x = \\ = x(1 \vee \bar{y} \vee \bar{z}) \vee \bar{y} \cdot z = x \vee \bar{y} \cdot z.$$

- Прилагайки законите на булевата алгебра, да се опрости следната логическа функция: $(\bar{x} \vee y \cdot z \vee \bar{y} \cdot \bar{z}) x (y \cdot \bar{z} \vee y \vee \bar{z})$

Решение:

$$(\bar{x} \vee y \cdot z \vee \bar{y} \cdot \bar{z}) x (y \cdot \bar{z} \vee y \vee \bar{z}) = \overline{\bar{x} \vee y \cdot z \vee \bar{y} \cdot \bar{z}} \vee x (y \cdot \bar{z} \vee y \vee \bar{z}) = \\ = x(y \cdot z \vee \bar{y} \cdot \bar{z}) \vee xy \cdot \bar{z} \vee x \cdot y \vee \bar{z} = xyz \vee x \bar{y} \cdot \bar{z} \vee xy \cdot \bar{z} \vee x \cdot \bar{y} \cdot z = \\ = xy(z \vee \bar{z}) \vee x \bar{y}(\bar{z} \vee z) = x(y \vee \bar{y}) = x.$$

- Да се докаже тъждеството $x_1 x_2 \vee x_1 \bar{x}_2 x_3 = x_1 x_2 \vee x_1 x_3$, прилагайки дистрибутивния закон:

Решение: $x_1 x_2 \vee x_1 \bar{x}_2 x_3 = x_1 (x_2 \vee \bar{x}_2 x_3) =$

$$= x_1 (x_2 \vee \bar{x}_2)(x_2 \vee x_3) = x_1 (x_2 \vee x_3) = x_1 x_2 \vee x_1 x_3.$$

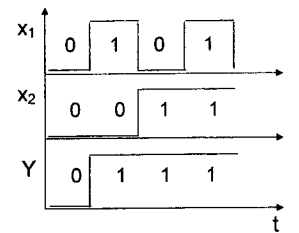
1.4 Задаване на логически функции

Логическите функции могат да бъдат задавани по различни начини. Независимо от възприетия начин на задаване на функцията е необходимо стойността на булевата функция да се представи със стойността си за всички комбинации (набори) от стойности на определящите я входни променливи. По-долу са дадени най-често ползваните начини за задаване (изобразяване) на логически функции:

x_1	x_2	x_{n-1}	x_n	Y
0	0		0	0	1
0	0		0	1	0
0	0		1	0	0
1	1		1	0	1
1	1		1	1	0

фиг.1.4 Представяне на булева функция на n аргумента чрез таблица на истинност

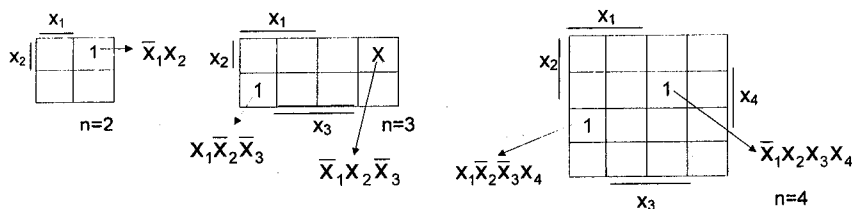
- текстово – това е например представяне от типа: “функцията Y има стойност единица само когато всички нейни аргументи имат стойност единица”, което описва логическа функция от типа И на n на брой аргументи: $Y = x_1 \cdot x_2 \cdot \dots \cdot x_{n-1} \cdot x_n$. Очевидно е, че този начин за описание на булеви функции не е универсален и е удобен само за представяне на относително прости функции;



фиг.1.5 Задаване на логическата функция ИЛИ чрез времедиаграма

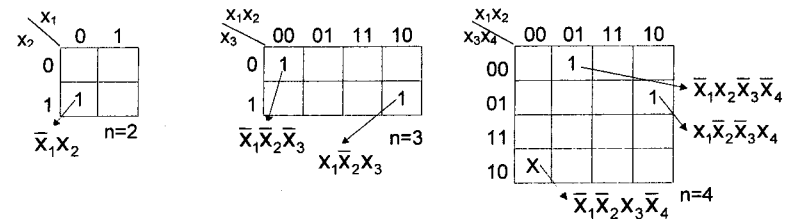
- аналитично – чрез логически израз, даващ зависимостта на булевата функция от своите аргументи, свързани по между си чрез съответните логически операции;
- таблично – посредством *таблица на истинност* (ТИ). Това представяне е универсално в булевата алгебра и е най-често използвано – фиг.1.4. В левите колони на ТИ се разполагат наборите на променливите, от които зависи функцията, а в крайната дясна колона се записва нейната стойност за конкретния набор;
- чрез представяне от тип *времедиаграма* – при това представяне по оста X (използвана обичайно за нанасяне на времето) се разполагат всички набори на входните променливи, а стойността на самата функция за всеки набор се изобразява с ниво нула, единица или неопределена стойност за съответния “времеви” участък – фиг.1.5 (представя функцията ИЛИ на два аргумента x_1 и x_2);
- графично – чрез таблици (карти) на Вейч или Карно: представлява двумерно изображение на стойностите на логическата функция в зависимост от стойността на аргументите.

Картите представляват квадрат или правоъгълник, по страните на който се записват стойностите на аргументите в прав или инверсен вид, така че да се получат всички възможни набори. В самата клетка се записва стойността на логическата функция (1, 0 или X) за съответния набор (фиг.1.6, фиг.1.7), като обикновено 0 не се нанася.



фиг.1.6 Графично представяне на логически функции на 2, 3 и 4 променливи чрез карти на Вейч

На всеки набор съответства само една точно определена клетка. При съставянето на самите карти е търсено представяне, при което всяка



фиг.1.7 Графично представяне на логически функции на 2, 3 и 4 променливи с карти на Карно

клетка да намира своите геометрично “съседни” клетки по отношение на всички участващи променливи – т.е. ако клетката отговаря например на набор x_1, x_2, x_3 , то съседните ѝ клетки следва да съответстват на набори: $\bar{x}_1 x_2 x_3, x_1 \bar{x}_2 x_3, x_1 x_2 \bar{x}_3$. Това представяне е в пряка връзка с търсене на т.нар. “минимални форми” на представяне на логическите функции и ще бъде подробно разгледано в гл.2;

- чрез запис на номерата на наборите, за които функцията има стойност 1 (или 0) - за пълно определена функция или чрез номерата на наборите на логическата функция, при които тя има стойност 1, и неопределена стойност (или 0 и неопределена стойност) - за непълно определена функция. Например, пълно определена функция f на четири променливи може да се зададе по един от следните начини:

$$f^1 = (0, 4, 7, 11, 13) \text{ или } f^0 = (1, 2, 3, 5, 6, 8, 9, 10, 12, 14, 15).$$

Непълно определената функция ϕ на три променливи също може да се представи по следните два начина:

$$\phi^1 = (1, 3, 4, 5), \phi^H = (0, 2) \text{ или } \phi^0 = (6, 7), \phi^H = (0, 2).$$

- чрез логическия елемент (елементи) или логическата схема, които реализират съответната функция.

1.5 Канонични форми на логически функции

Много често в практиката се изисква една логическа функция да бъде представена в базис И-НЕ или в базис ИЛИ-НЕ. Такова представяне може да бъде лесно получено, ако имаме запис на функцията чрез елементарни логически функции И и ИЛИ. Всяко от тези представяния е удобно с оглед схемната реализация на функцията, тъй като логически елементи от тези видове се предлагат от всички производители на интегрални логически елементи.

Съществуват две основни нормални форми на запис на една логическа функция: като сума от произведения (*sum of products*, SOP) - *дизюнктивна нормална форма* (ДНФ), и като произведение от суми (*product of sums*, POS) - *конюнктивна нормална форма* (КНФ). На всяка една от тези форми на представяне съответства двустъпална схемна реализация на логическата функция.

Представянето на логическите функции в т.нар. *канонични* (съвършени) форми изисква въвеждането на понятията *минчлен* и *максчлен*:

- *минчлен* (конституента на единицата) е такава логическа функция, която приема стойност 1 за един-единствен набор. За всички останали набори тя приема стойност 0. Прието е минчленовете да се означават с m_i , където i означава номера на единичния за съответния минчлен набор. В Табл.1.7 е дадена таблицата на истинност за минчленовете с номера 2,3 и 7 за функция на три променливи;
- *максчлен* (конституента на нулата) е тази логическа функция, приемаща стойност 0 само за един-единствен набор. За всички останали набори тя има стойност 1. Максчленовете се означават с M_i , където i означава номера на нулевия за съответния максчлен набор. Табл.1.8 представя таблицата на истинност за максчленове с номера 1,4 и 6.

x_1	x_2	x_3	m_2	m_3	m_7
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	0	0	1

Табл.1.7

x_1	x_2	x_3	M_1	M_4	M_6
0	0	0	1	1	1
0	0	1	0	1	1
0	1	0	1	1	1
0	1	1	1	1	1
1	0	0	1	0	1
1	0	1	1	1	1
1	1	0	1	1	0
1	1	1	1	1	1

Табл.1.8

Броят на минчленовете, както и този на максчленовете за една функция на n променливи е равен на 2^n . Записът на минчленовете се извършва с помощта на логическата операция конюнкция, а този на максчленовете – чрез операцията дизюнкция. За примерите от Табл.1.7 и Табл.1.8 тези записи имат съответно вида:

$$m_2 = \bar{x}_1 \cdot x_2 \cdot \bar{x}_3, \quad m_3 = \bar{x}_1 \cdot x_2 \cdot x_3, \quad m_7 = x_1 \cdot x_2 \cdot x_3$$

$$M_1 = x_1 \vee x_2 \vee \bar{x}_3, \quad M_4 = \bar{x}_1 \vee x_2 \vee x_3, \quad M_6 = \bar{x}_1 \vee \bar{x}_2 \vee x_3.$$

Нормалната И-ИЛИ форма на една логическа функция, в която всички членове са минчленове, се нарича *канонична* (съвършена, СДНФ) И-ИЛИ форма – т.е. *дизюнкция от минчленовете*, за които функцията има стойност 1. Тази форма на представяне е единствена за

съответната логическа функция. За примера от Табл.1.9 (пълно определена функция Z на четири променливи) каноничната СДНФ е следната:

$$Z = m_2 \vee m_4 \vee m_5 \vee m_{10} \vee m_{13} \vee m_{14} =$$

$$= \bar{x}_1 \bar{x}_2 x_3 \bar{x}_4 \vee \bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 \vee \bar{x}_1 x_2 \bar{x}_3 x_4 \vee x_1 \bar{x}_2 x_3 \bar{x}_4 \vee x_1 x_2 \bar{x}_3 x_4 \vee x_1 x_2 x_3 \bar{x}_4.$$

Нормална ИЛИ-И форма на логическа функция, в която всички членове са максчленове, се нарича *канонична* (съвършена, СКНФ) ИЛИ-И форма – т.е. *конюнкция от максчленовете*, за които функцията има стойност 0. Представянето от този тип е единствено за всяка логическа функция. За функцията от Табл.1.9 СКНФ има вида:

$$Z = M_0 M_1 M_3 M_6 M_7 M_8 M_9 M_{11} M_{12} M_{15} =$$

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

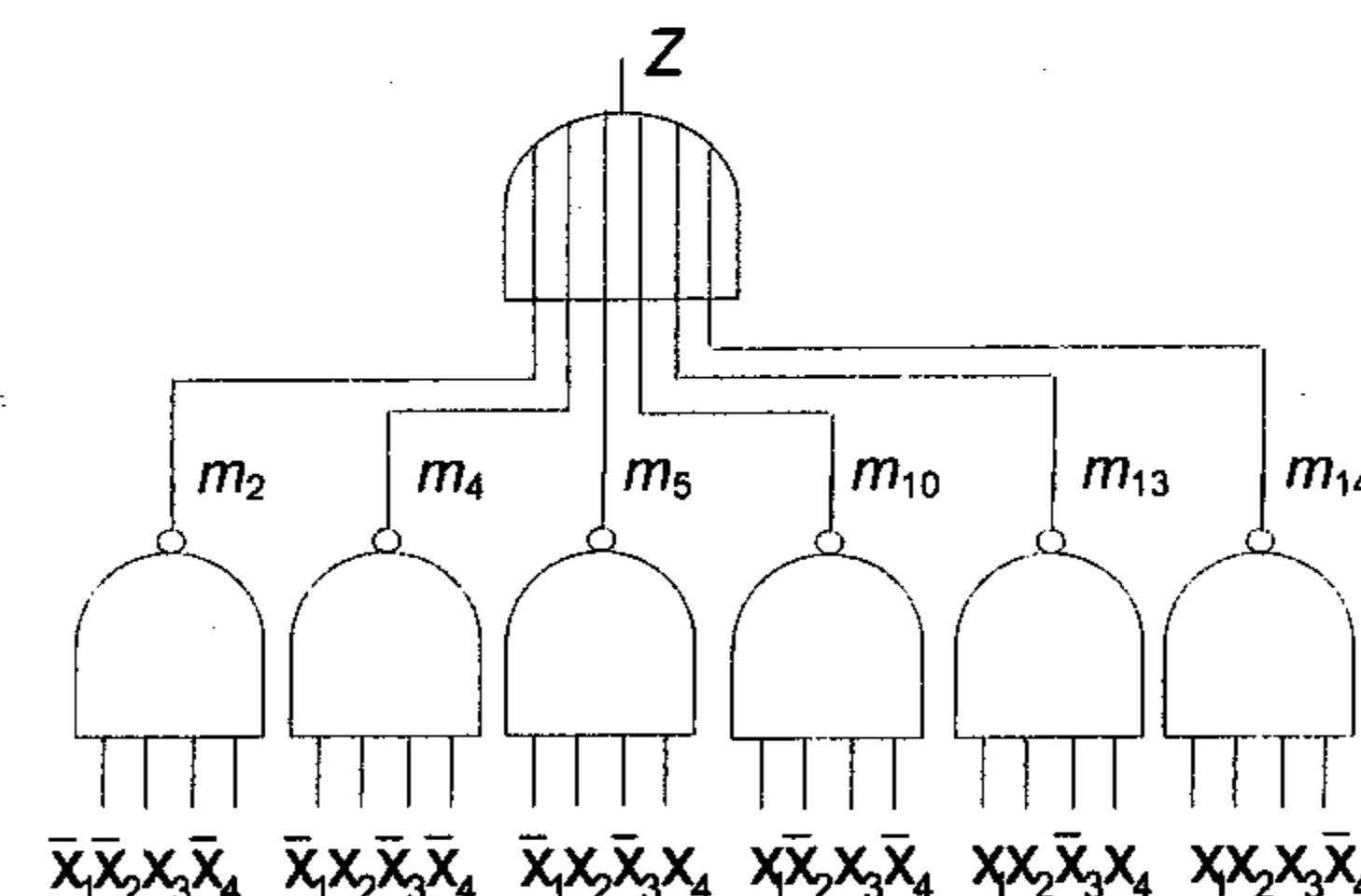
$$\cdot (x_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4)(\bar{x}_1 \vee x_2 \vee x_3 \vee x_4)(\bar{x}_1 \vee x_2 \vee x_3 \vee \bar{x}_4)(\bar{x}_1 \vee x_2 \vee \bar{x}_3 \vee \bar{x}_4).$$

$$\cdot (\bar{x}_1 \vee \bar{x}_2 \vee x_3 \vee x_4)(\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4).$$

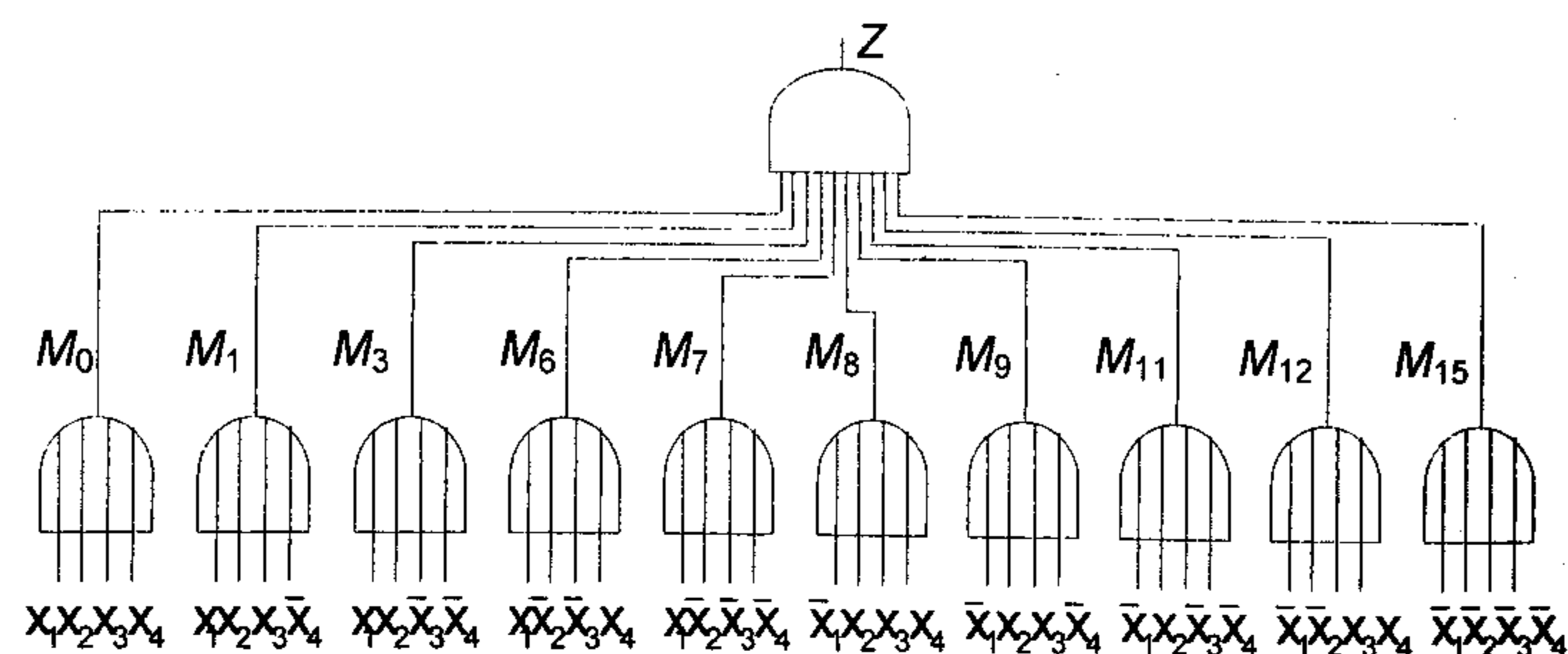
Схемната реализация чрез логически елементи, съответстващо на запис на функцията Z в канонична И-ИЛИ и ИЛИ-И форма са представени на фиг.1.8 а,б.

x_1	x_2	x_3	x_4	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Табл.1.9



фиг.1.8 а Канонична И-ИЛИ форма на функцията Z



фиг.1.8 б Канонична ИЛИ-И форма на функцията Z

1.6 Премаване от нормални към канонични форми на логически функции

Премаването към канонична форма на една логическа функция може да стане от произволен аналитичен запис на функцията чрез прилагане на законите на булевата алгебра и съответните преобразувания. Преобразуването от нормална конюнктивна или дизюнктивна форма към съответните канонични форми се извършва чрез следните преобразувания:

- преминаване от ДНФ към СДНФ. Нека p е елементарна конюнкция, в която липсва аргументът x_i . Добавянето му, така че да не се промени стойността на функцията, се извършва въз основа на следното равенство: $p = p \cdot 1 = p(x_i \vee \bar{x}_i) = px_i \vee p\bar{x}_i$. Ако в получените елементарни конюнкции px_i и $p\bar{x}_i$ все още има липсващи аргументи, описаната операция се повтаря. Процесът продължава, докато се включат всички липсващи аргументи на функцията;
- преминаване от КНФ към СКНФ. Нека q е елементарна дизюнкция, в която липсва променливата x_i . Добавянето на тази променлива, така че да не се измени стойността на функцията, се извършва с прилагане на зависимостта: $q = q \vee 0 = q \vee x_i\bar{x}_i = (q \vee x_i)(q \vee \bar{x}_i)$. При положение, че в получените елементарни дизюнкции $q \vee x_i$ и $q \vee \bar{x}_i$ има още липсващи аргументи, процедурата се повтаря до включването на всички останали аргументи.

Примерите, дадени по-долу, показват техниките за преминаване към дизюнктивна и конюнктивна канонични форми на представяне на логически функции:

- Да се представи функцията $\varphi = x_1x_2x_3 \vee \bar{x}_1x_3 \vee x_1x_2 \vee \bar{x}_1\bar{x}_2x_3$ в канонична дизюнктивна нормална форма (СДНФ) и да се запише с номерата на минтермите си:

Решение:

$$\begin{aligned} \varphi &= x_1x_2x_3 \vee \bar{x}_1x_3 \vee x_1x_2 \vee \bar{x}_1\bar{x}_2x_3 = x_1x_2x_3 \vee \bar{x}_1x_3(x_2 \vee \bar{x}_2) \vee \\ &\vee x_1x_2(x_3 \vee \bar{x}_3) \vee \bar{x}_1\bar{x}_2x_3 = \underline{x_1x_2x_3} \vee \bar{x}_1x_2x_3 \vee \underline{\bar{x}_1\bar{x}_2x_3} \vee \\ &\vee \underline{x_1x_2x_3} \vee x_1x_2\bar{x}_3 \vee \underline{\bar{x}_1\bar{x}_2x_3} = x_1x_2x_3 \vee \bar{x}_1\bar{x}_2x_3 \vee \bar{x}_1x_2x_3 \vee x_1x_2\bar{x}_3. \end{aligned}$$

$$111 \quad 001 \quad 011 \quad 110$$

$$\varphi = \vee m(1,3,6,7).$$

Забележка: Отдолу е указан бинарният запис на всеки минтерм в каноничната форма на функцията.

- Функцията $f = (x \vee \bar{z})(y \vee \bar{x})$ да се запише в канонична конюнктивна нормална форма (СКНФ) и чрез номерата на макстермите си:

Решение:

$$\begin{aligned} f &= (x \vee \bar{z})(y \vee \bar{x}) = (x \vee y \cdot \bar{y} \vee \bar{z})(\bar{x} \vee y \vee z \cdot \bar{z}) = \\ &= (x \vee y \vee \bar{z})(x \vee \bar{y} \vee \bar{z})(\bar{x} \vee y \vee z)(\bar{x} \vee y \vee \bar{z}). \end{aligned}$$

$$1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0$$

$$f = \wedge M(2,3,4,6).$$

Задание

1. Опростете зададените изрази, като използвате законите и правилата на булевата алгебра.
2. Докажете представените твърдения.
3. Представете дадените логически функции в канонична форма:
 - в СДНФ;
 - в СКНФ.

Контролни въпроси

1. Какво представлява набор на една логическа функция. Как се представя наборът в двоичен и десетичен вид?
2. Как се определя броят набори и възможните стойности на произволна булева функция на n аргумента?
3. Какви видове логически функции съществуват по отношение на определеността си?
4. Как се дефинират понятията младши и старши бит от съвкупността аргументи на една двоична функция?

5. Дефинирайте постулатите и свойствата на булевата алгебра.
6. Какви са основните закони на булевата алгебра? Кой от законите е уникален само за булевата алгебра?
7. Как се дефинира законът на Де Морган? Представете го в двете му форми: И-НЕ и ИЛИ-НЕ. Начертайте представянето му чрез логически елементи.
8. Какви начини на задаване (представяне) на логически функции познавате? Какво представляват картите на Вейч и Карно за 2, 3 и 4 променливи.
9. Какво представляват каноничните форми (СДНФ, СКНФ) за представяне на логически функции? Като изрази от какви конституенти се представя всяка от тези форми?

ТЕМА 2

МЕТОД НА КУАЙН - МАК КЛАСКИ ЗА МИНИМИЗАЦИЯ НА ЛОГИЧЕСКИ ФУНКЦИИ. МИНИМИЗАЦИЯ НА ЛОГИЧЕСКИ ФУНКЦИИ С КАРТИ НА ВЕЙЧ. МИНИМИЗАЦИЯ НА СИСТЕМА ЛОГИЧЕСКИ ФУНКЦИИ

Основна задача при синтеза на логическите функции е да се намери форма на представяне, която да осигури реализация с минимален брой логически елементи. Тази задача се нарича минимизация, а резултатът – минимална форма на функцията. Съществуващите методи за минимизация са разработени в базис, включващ функциите конюнкция, дизюнкция и отрицание. Получената минимална форма е дизюнктивна нормална форма с възможно най-малък брой елементарни конюнкции, като всяка от тях е възможно най-кратка, т.е. с най-малък брой променливи.

2.1 Метод на Куайн - Мак Класки

Ако са дадени 2 логически функции f и ϕ , които зависят от едни и същи аргументи и е изпълнено условието единиците на ϕ да са подмножество на единиците на f , се казва, че ϕ е импликанта на f .

x_1	x_2	f	ϕ_1	ϕ_2
0	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	1	0	1

Фиг. 2.1 Функция f , нейна импликанта ϕ_1 и не-импликанта ϕ_2 .

От примера на фиг. 2.1 се вижда, че ϕ_1 е импликанта на f , но ϕ_2 не е. Една импликанта се нарича проста, ако никоя нейна съставна част не е импликанта на същата функция. Това означава, че простите импликанти съдържат минимален брой от променливите на функцията. Нека представим функцията f от фиг. 2.1 в СДНФ: $f = \bar{x}_1\bar{x}_2 \vee x_1\bar{x}_2 \vee x_1x_2$. Нито една от импликантите (в случая импликантите са и минтерми) не е проста. От $\bar{x}_1\bar{x}_2$ може да отпадне \bar{x}_1 , от $x_1\bar{x}_2$ - x_1 или \bar{x}_2 , а от x_1x_2 - x_2 . Функцията f има две прости импликанти - x_1 и \bar{x}_2 . Първата се получава от слепването на импликантите $x_1\bar{x}_2$ и x_1x_2 , а втората - от слепването на импликантите $x_1\bar{x}_2$ и $\bar{x}_1\bar{x}_2$. Казва се, че простата импликанта *покрива* минтермите, от които е получена. Целта на минимизацията се свежда до търсене на покритие на функцията от минимален брой прости импликанти (ПИ).

Минимизацията по метода на Куайн-Мак Класки протича в следния ред:

1. Функцията се представя в СДНФ – дизюнкция от конституенти на единицата (минтерми), съответстващи на наборите, за които функцията има стойност 1.

2. Тези набори се разделят на групи според броя на единиците в тях и се записват в колона.

3. Всеки набор от всяка група се проверява за възможност за слепване с всеки набор от съседната група¹. Ако е възможно слепване, резултатът се записва в нова колонка, като на мястото на отпадналата променлива се пише тире. Слепените набори се отбелязват със звездичка.

4. Получените импликанти се подреждат в групи според позицията, в която се е извършило слепването. Търсят се всички възможности за слепване между членовете на всяка група. Слепените импликанти се отбелязват, а резултатът се записва в нова колона.

5. Процесът продължава, докато се получат импликанти, които повече не могат да се слепят. Ако дадена импликанта има k на брой тирета (отпаднали са k променливи), тя трябва да се получи 2^{k-1} пъти, т.е. от 2^{k-1} различни слепвания. Неотбелязаните набори заедно с неотбелязаните импликанти от стъпки 3 и 4 представляват всички прости импликанти на функцията. Тяхната дизюнкция покрива единиците на функцията, но в общия случай това не е минималната ѝ форма.

Преди да продължим нататък, нека илюстрираме казаното дотук с един пример. Задачата е да се минимизира функцията $f = \sum m(0,1,2,3,4,5,7,10,12,13,15)$ по метода на Куайн-Мак Класки.

Функцията е представена в числов вид. Преминаваме към групирането на наборите според броя на единиците в тях и търсене на простите импликанти чрез осъществяване на всички възможности за слепване.

(0)	0000 *	000-	-010	-10-	-10-
(1)	0001 *	00-0	-100 *	-1-1	-1-1
(2)	0010 *	0-00	-101 *	0-0-	0-0-
(4)	0100 *	00-1	-111 *	0--1	0--1
(3)	0011 *	0-01	0-00 *	00--	00--
(5)	0101 *	001-	0-01 *	0-1	
(10)	1010 *	-010	0-11 *	1-1	
(12)	1100 *	010-	00-0 *	00--	
(7)	0111 *	-100	00-1 *	0-0-	
(13)	1101 *	0-11	01-1 *	10--	
(15)	1111 *	01-1	11-1 *		
		-101	000- *		
		110-	001- *		
		-111	010- *		
		11-1	110- *		

¹ За краткост ще казваме, че се слепват наборите, но всъщност се слепват минтермите, съответстващи на наборите, за които функцията има стойност 1.

Намерените ПИ са: $\bar{x}_2 x_3 \bar{x}_4$, $x_2 \bar{x}_3$, $x_2 x_4$, $\bar{x}_1 \bar{x}_3$, $\bar{x}_1 x_4$, $\bar{x}_1 \bar{x}_2$

6. Търсенето на минимално покритие се осъществява с помощта на таблица на покритията (импликантна таблица). Колони са наборите, за които функцията има стойност 1, а редове – простите импликанти. В клетката, получена при пресичане на i -тия ред с j -тия стълб се поставя отметка, ако ПИ с номер i покрива набор² с номер j . По този начин таблицата се попълва с отметки, които онагледяват кои ПИ кои набори покриват. От тук нататък целта е да се подбере минимален брой ПИ, които съвместно покриват всички единици на функцията.

Построяваме таблицата на покритията на функцията – табл.2.1.

	0000	0001	0010	0011	0100	0101	0111	1010	1100	1101	1111	
_010			*					⊕				A
10					*	*			⊕	*		B
_1_1						*	*			*	⊕	C
0_0_	*	*			*	*						D
0_1		*		*		*	*					E
00_	*	*	*	*								F

Табл.2.1

Възможни са два подхода за намиране минималното покритие на функцията: евристичен и систематичен.

Евристичен подход

1. Определят се задължителните ПИ. Това са ПИ, които единствено покриват някои от минтермите. В колоните, съответстващи на тези набори има само една отметка и съответната ПИ задължително трябва да участва в окончателния израз, представящ функцията. От таблица 2.1 се вижда, че задължителните ПИ са 3: $\bar{x}_2 x_3 \bar{x}_4$, $x_2 \bar{x}_3$, $x_2 x_4$.

2. Таблицата се съкращава, като отпадат всички колони, покрити от задължителните ПИ – получава се табл.2.2.

	0000	0001	0011
0-0-	*	*	
0--1		*	*
00--	*	*	*

Табл.2.2

² За краткост ще казваме, че простата импликанта покрива наборите, но всъщност тя покрива минтермите, съответстващи на наборите.

3. Търси се покритие от най-малък брой най-кратки ПИ. От табл. 2.2 се вижда, че $\bar{x}_1\bar{x}_2$ е простата импликанта, която едновременно покрива останалите 3 набора.

4. Минималната форма на функцията е дизюнкция от простите импликанти, определени в т.1 и т.3, а именно: $f = \bar{x}_2x_3\bar{x}_4 \vee x_2\bar{x}_3 \vee x_2x_4 \vee \bar{x}_1\bar{x}_2$.

Систематичен подход

Съставя се т.нар. функция на покритията. За целта всеки ред от таблицата се именува с допълнителна променлива. Условието за покритие на всяка колона е дизюнкция от допълнителните променливи, съответстващи на простите импликанти, покриващи минтерма. Функцията на покритията е конюнкция от условията за покритие на всяка от колоните. Получената КНФ се преобразува в ДНФ чрез прилагане на закона за поглъщането и чрез разкриване на скобите. Всеки конюнктивен член от тази ДНФ представлява една несъкратима форма на функцията.

За примера от табл.2.1 функцията на покритията има вида

$$f_{\text{покр.}} = (D \vee F)(D \vee E \vee F)(A \vee F)(E \vee F)(B \vee D)(B \vee C \vee D \vee E)(C \vee E)A.B.(B \vee C).C$$

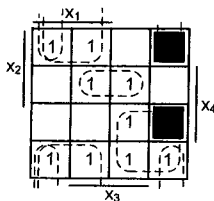
Прилагаме закона за поглъщането. Отпадат дизюнкциите, в които участват A, B или C .

$$f_{\text{покр.}} = A.B.C.(D \vee F)(D \vee E \vee F)(E \vee F) = A.B.C.(D \vee F)(E \vee F) = (A.B.C.D \vee A.B.C.F)(E \vee F) =$$

$$= A.B.C.D.E \vee A.B.C.E.F \vee A.B.C.D.F \vee A.B.C.F = A.B.C.D.E \vee A.B.C.F \vee A.B.C.D.F$$

Очевидно, най-кратката форма съответства на произведението $A.B.C.F$

2.2 Минимизация с карти на Вейч



Фиг. 2.2
Минимизация на
функция с карта
на Вейч

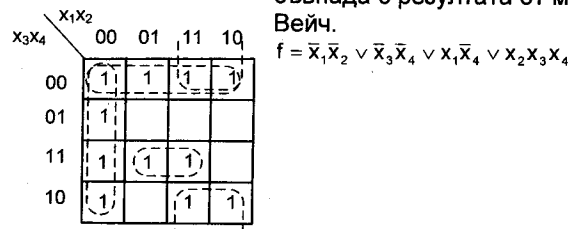
Картата на Вейч представлява квадрат или правоъгълник, разделен на клетки, чийто брой съвпада с броя на наборите на функцията. Функция на n аргумента се нанася в карта с 2^n на брой клетки. На всеки аргумент се съпоставя половината от клетките в картата, образуващи компактна група от редове или стълбове. На инверсната стойност на аргумента се съпоставя другата половина от картата. Разположението на аргументите върху картата задава съответствието клетка – номер на набор. В клетката се нанася стойността на логическата функция за съответния набор. Специфичното разположение на аргументите върху

картата задава такова разположение на наборите, което поставя тези минтерми, между които е възможно слепване в т.нар. съседни клетки. Съседни са клетките, които са една до друга (имат обща стена) или една срещу друга в краищата на даден ред или стълб (при карти за 2, 3, 4 аргумента). Картите за 5, 6, 7 и повече аргументи се състоят от съседни

карти за функции на 4 променливи. Условието за съседство в тях са като във всяка от съставлящите карти, но се добавя и още едно – съседни са и клетките, които се намират на едно и също място в съседните карти. Условието за съседство на картите са същите като условията за съседство на клетките. В приложението на стр. 113 са показани карти на Вейч за 2, 3, 4, 5, 6 и 7 аргумента и е посочено съответствието между наборите от аргументи и клетките в картите. На фиг. 2.2 е показана функцията $f = \vee m(0,1,2,3,4,7,8,10,12,14,15)^1$, нанесена в карта на Вейч за 4 аргумента.

Минимизацията с карта на Вейч се свежда до слепване, което се извършва директно върху картата. Търсят се и се ограждат максимално големи групи от съседни единици, които образуват квадрат или правоъгълник (фиг. 2.2). Целта е всички единици на функцията да бъдат покрити с най-малък брой фигури от съседни единици, като всяка фигура съдържа възможно най-голям брой клетки. Някои от единиците могат да участват в повече от едно слепване, а други – в едно-единствено. Всяко слепване трябва да включва поне една единица, която не участва в друго слепване. Дизюнкцията от логическите произведения, описващи слепените единици, е минималната форма на функцията. Винаги се слепват 2^k на брой единици, като буквите, с които се описват, са $n-k$. Минималната форма на функцията от фиг. 2.2 е $f = \bar{x}_1\bar{x}_2 \vee \bar{x}_3\bar{x}_4 \vee x_1\bar{x}_4 \vee x_2x_3x_4$. Задължителните ПИ са $\bar{x}_1\bar{x}_2$ и $\bar{x}_3\bar{x}_4$, тъй като те покриват единици, които не могат да бъдат покрити по друг начин (защрихованите).

Освен карти на Вейч в литературата се срещат и т.нар. карти или матрици на Карно. Те са подобни на картите на Вейч и принципите за минимизация са същите. Различават се по разположението на аргументите върху картата. На фиг. 2.3 е показана карта на Карно за 4 аргумента, в която е нанесена функцията от фиг. 2.2. Резултатът от минимизацията съвпада с резултата от минимизацията от картата на Вейч.

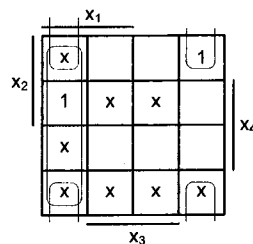


Фиг. 2.3 Минимизация на
функция с карта на Карно

2.3 Минимизация на непълно определени логически функции

При минимизиране по метода на Куайн – Мак Класки се постъпва по следния начин: Когато се търсят всички възможни сцепвания, неопределените набори се доопределят като единици. По този начин се получават възможно най-кратки и възможно най-голям брой конюнкции. При построяването на импликантната таблица неопределените набори се доопределят с нули, т.е. търси се покритие само на наборите, за които функцията задължително има стойност 1.

При минимизиране с карти на Вейч с единица се доопределят само тези неопределени набори, които допълват група от съседни единици до по-голяма такава група. Останалите неопределени набори се доопределят с нули. На фиг. 2.4 е показана карта на Вейч, в която е нанесена една непълно определена функция. Нейната минимална форма, определена в съответствие със споменатото правило, е: $f = x_1 \bar{x}_3 \vee \bar{x}_3 \bar{x}_4$.



Фиг. 2.4
Минимизация
на НОФ

2.4 Минимизация на система логически функции

Разгледаните дотук методи за минимизация се отнасят за самостоятелно минимизиране на отделна логическа функция. Синтезът на цифрови устройства с повече от един изход изисква синтез и минимизация на система логически функции, зависещи от едни и същи входни променливи.

Възможно е всяка от функциите да се минимизира самостоятелно, но по-добри резултати обикновено се получават, когато в процеса на минимизация се търсят

обща части между функциите, които се реализират еднократно.

2.4.1 Минимизация на система логически функции с обща подфункция

Обща подфункция е тази функция ϕ , която има стойност 1 за наборите, за които всички функции от системата имат стойност 1. Тя се минимизира самостоятелно (обикновено с карта на Вейч). Всяка функция f_i , ($i = 1 + n$, n – брой функции) от системата се разглежда като дизюнкция от общата подфункция ϕ и допълваща функция f_i^* . f_i^* има стойност 1 за наборите, за които f_i има стойност 1, с изключение на наборите на общата подфункция – за тях функциите f_i са неопределени. f_i^* се минимизират самостоятелно.

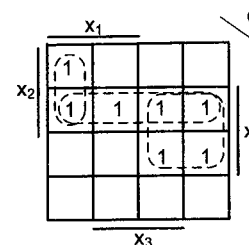
Нека зададените функции f_1 , f_2 и f_3 да се минимизират чрез обща подфункция:

$$f_1 = \vee m(1,3,5,7,8,12,13,14,15)^1$$

$$f_2 = \vee m(1,3,4,5,6,7,12,13,15)^1$$

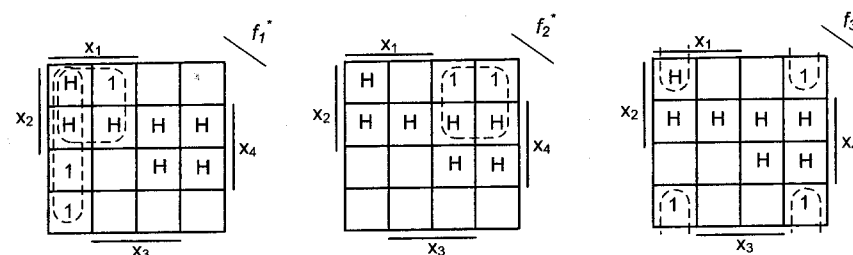
$$f_3 = \vee m(0,1,3,4,5,7,8,12,13,15)^1$$

Определяме подфункцията $\phi = \vee m(1,3,5,7,12,13,15)^1$ и я нанасяме в карта на Вейч (фиг. 2.5). След минимизацията получаваме $\phi = x_2 x_4 \vee \bar{x}_1 x_4 \vee x_1 x_2 \bar{x}_3$.



Фиг. 2.5 Минимизация на подфункцията ϕ , нанесена в карта на Вейч

Нанасяме f_1 , f_2 и f_3 в карти на Вейч, като за наборите на ϕ поставяме знак за неопределеност (фиг. 2.6).

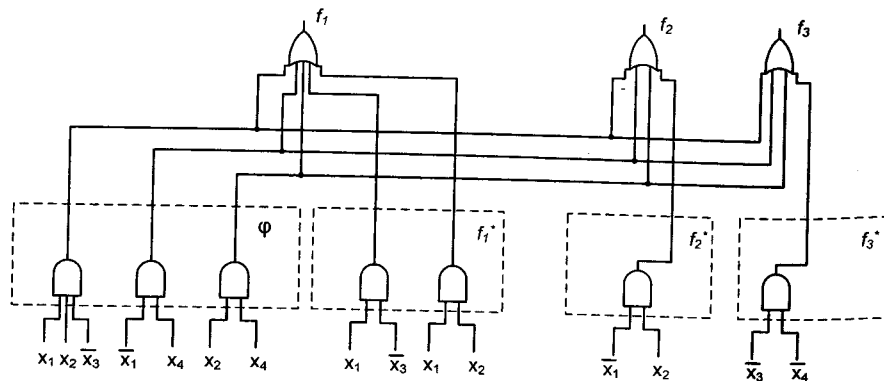


Фиг. 2.6 Функциите f_1^* , f_2^* и f_3^*

От фиг. 2.6. определяме: $f_1^* = x_1 x_2 \vee x_1 \bar{x}_3$, $f_2^* = \bar{x}_1 x_2$, $f_3^* = \bar{x}_3 \bar{x}_4$.

За f_1 , f_2 и f_3 получаваме съответно: $f_1 = \phi \vee f_1^*$, $f_2 = \phi \vee f_2^*$, $f_3 = \phi \vee f_3^*$.

Схемата е показана на фиг. 2.7.



Фиг. 2.7 Структурна схема на системата функции, минимизирана чрез обща подфункция

2.4.2 Систематичен подход за минимизация на система логически функции

Този подход използва метода на Мак Класки за съвместно минимизиране на функциите от системата. При търсенето на възможности за слепване участват всички набори, за които поне една от функциите има стойност 1. Всеки набор е съпроводен от признакова част, която е „п“-разрядно двоично число, където п е броят на функциите. В i-тия разряд на признаковата част има 1, ако i-тата функция има стойност 1 за него.

Търсенето на всички ПИ протича по известния вече начин, но с една особеност: при слепване се отбелязват само тези импликанти, които освен че са участвали в операцията, имат признакова част, съпадаща с признаковата част на резултата. Той се получава чрез поразрядна конюнкция на признаковите части на импликантите, участвали в слепването. По този начин се получават всички прости импликанти на системата.

Следва построяване на таблица на покритието на системата, представена като една хипотетична функция, която има стойност 1 за наборите, за които поне една от функциите има стойност 1. Простите импликанти в тази таблица са намерените ПИ на системата. Те участват заедно със своята признакова част. Колони са наборите, за които някоя от функциите има стойност 1. Всяка колона се разделя на подколони, означени с номера на съответната функция, на която той принадлежи. По този начин се маркира принадлежността на набора към съответната функция. При поставянето на отметките, отбелязващи коя ПИ кой набор покрива, се взема предвид принадлежността на набора. Отметка се

поставя при две условия: 1) ПИ покрива набора и 2) признаковата ѝ част показва, че тя и наборът принадлежат на една и съща функция. Например, ПИ с признакова част 101 може да покрива набори от първата и третата функция, но не от втората.

След като се определят ПИ на хипотетичната функция, следва построяване на таблици на покритието на всяка отделна функция от системата. В построяването на съответните таблици участват тези от току-що намерените ПИ, които принадлежат на съответната функция. Това се определя от признаковата част на всяка от простите импликанти. Колони са наборите, за които дадената функция има стойност 1.

Ще разгледаме този метод за минимизация с помощта на системата функции от следващия пример.

$$f_1 = \vee m(4,6,11)^1, \vee m(3,5,7,15)^H$$

$$f_2 = \vee m(0,3,6,11)^1, \vee m(4)^H$$

$$f_3 = \vee m(0,11)^1, \vee m(1,3,9)^H$$

1. Търсим простите импликанти на системата, като доопределяме всички неопределени набори с единици - фиг. 2.8.

2. Построяваме таблицата на покритието. Неопределените набори доопределяме с нули. За удобство именуваме простите импликанти.

0000 (011)	000- (001)	-001 (001)*	-0-1 (001)	-0-1 (001)
0001 (001) *	0-00 (010)	-011 (111)	--11 (100)	--11 (100)
0100 (110) *	00-1 (001)	-111 (100) *	--11 (100)	01-- (100)
0011 (111) *	0-01 (000)	0-00 (010)	0--1 (000)	
0101 (100) *	-001 (001)	0-11 (100) *	0-11 (001)	
0110 (110) *	010- (100)	1-11 (100) *	01-- (100)	
1001 (001) *	01-0 (110)	00-1 (001) *	0-0- (000)	
0111 (100) *	0-11 (100)	01-0 (110)	01-- (100)	
1011 (111) *	-011 (111)	01-1 (100) *		
1111 (100) *	01-1 (100)	10-1 (001) *		
	011- (100)	000- (001)		
	10-1 (001)	010- (100) *		
	-111 (100)	011- (100) *		
	1-11 (100)			

Фиг. 2.8 Прости импликанти на системата функции

	0000		0011	0100	0110		1011		
	2	3	2	1	1	2	1	2	3
0000 (011) a	*	*							
-011 (111) b			⊕				*	⊕	*
0-00 (010) c	*								
01-0 (110) d				*	*	⊕			
000- (001) e		*							
-0-1 (001) f									*
--11 (100) g							*		
01-- (100) h				*	*				

Табл. 2.3

Простите импликанти, покриващи системата, са b, d и a.

Построяваме таблици на покритието на всяка отделна функция – фиг. 2.9. В таблицата на първата функция не участва a, а само b и d, тъй като a не ѝ принадлежи. Предвид съображението за принадлежност са построени таблиците на покритието и на другите две функции. В конкретния случай всички ПИ от таблиците са задължителни за съответните функции. Минималните форми на функциите се получават като дизюнкция от задължителните ПИ.

	0100	0110	1011
-011 b			*
01-0 d	*	*	

$$f_1 = b \vee d$$

	0000	0011	0110	1011
0000 a	*			
-011 b		*		*
01-0 d			*	

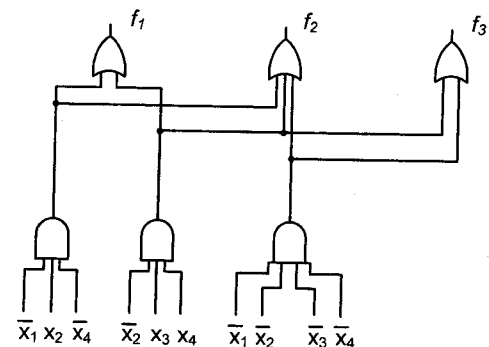
$$f_2 = a \vee b \vee d$$

	0000	1011
0000 a	*	
-011 b		*

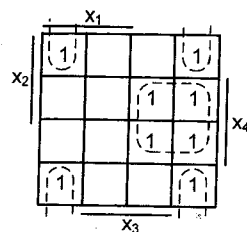
$$f_3 = a \vee b$$

Фиг. 2.9 Таблицы на покритието на отделните функции и техните минимални форми

Схемата на системата функции, минимизирана по систематичния подход, е показана на фиг. 2.10.



Фиг. 2.10 Структурна схема на системата функции, минимизирана по систематичния подход



Фиг. 2.11 Функцията f_1 , избрана за базова

2.4.3 Минимизация на система функции с базова функция

При този подход една от функциите (обикновено най-простата) се избира за базова и се минимизира самостоятелно с карта на Вейч. Тя се добавя като допълнителна променлива към аргументите, от които зависят останалите функции, и удвоява наборите им. При попълването на техните таблици на истинност се взема предвид стойността на базовата функция. Ако стойността на базовата функция като част от набора не съвпада с нейната действителна стойност, този набор е нереален. Ако съвпада, наборът е реален и срещу него се записва стойността на съответната функция. Следва нанасяне на функцията в карта на Вейч и търсене на минималната ѝ форма. По този начин се обработват всички функции в системата. Ще разгледаме третия подход с помощта на следващия пример.

Системата функции се състои от следните 3 функции:

$$f_1 = \vee m(0,1,3,4,5,7,8,12)'$$

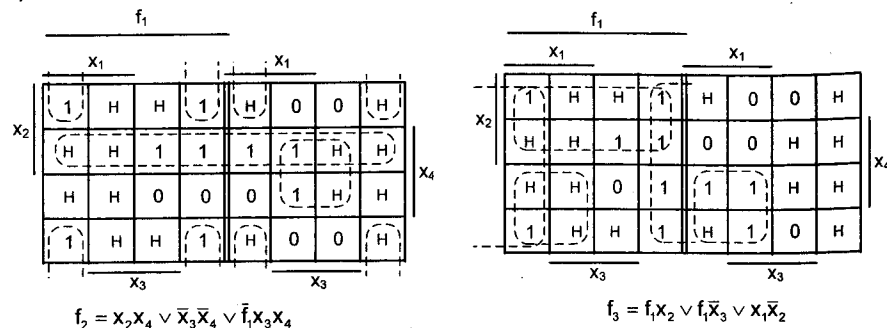
$$f_2 = \vee m(0,4,5,7,8,11,12,13,15)'$$

$$f_3 = \vee m(0,1,4,5,7,8,9,10,11,12)'$$

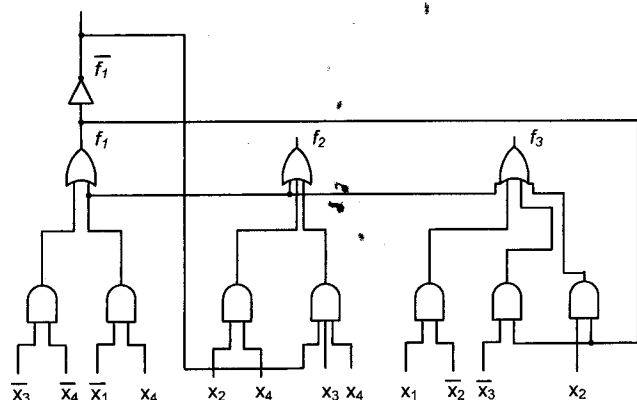
За базова избираме функцията f_1 . Нанасяме я в карта на Вейч и минимизираме. (фиг. 2.11)

$$f_1 = \bar{x}_1 x_4 \vee \bar{x}_3 \bar{x}_4$$

Построяваме таблиците на истинност на функциите f_2 и f_3 по описания начин и ги нанасяме в карти на Вейч. Определяме минималните им форми-фиг. 2.12. Схемата е показана на фиг. 2.13.



Фиг. 2.12 Минимални форми на функциите f_2 и f_3



Фиг. 2.13 Структурна схема на системата функции, минимизирана с базова функция

Задание

1. Да се минимизира по метода на Куайн – Мак Класки зададена непълноопределена функция на 4 променливи.
2. Същата функция да се минимизира с карта на Вейч.
3. Получената минимална форма на функцията да се реализира и да се снесе нейната таблица на истинност.
4. Да се нанесе в карта на Вейч и да се минимизира зададена функция на 5 аргумента, представена в произволна ДНФ.
5. Да се минимизира зададена система от логически функции по трите подхода и да се сравнят резултатите.

Контролни въпроси

1. Какво е импликанта на дадена функция?
2. Какво е проста импликанта (ПИ)?
3. Какво означава една ПИ да покрива даден минтерм?
4. Защо при минимизация по метода на Куайн - Мак Класки се търсят възможности за слепване само между наборите от съседни групи?
5. Как се съставя и попълва таблицата на покритията?
6. Какъв е броят на отметките в един ред от таблицата на покритията за пълноопределена функция? А за непълноопределена?
7. Кой ПИ са задължителни?
8. Кой ПИ могат да отпаднат от окончателния израз за минималната форма на функцията?
9. Кой клетки са съседни в картата на Вейч и защо единиците, които се намират в съседни клетки, могат да се слепват?
10. По какво се отличава търсенето на минимална форма на НОФ?
 - а) по метода на Куайн – Мак Класки;
 - б) с карта на Вейч
11. В какво се състои минимизацията на система логически функции с използването на обща подфункция?
12. Какво показва признаковата част на всеки набор при систематичния подход за минимизация на система логически функции?

13. На какво допълнително условие трябва да отговарят наборите и ПИ, за да бъдат отбелязани при този подход?
14. Как се постъпва с ПИ, която в признаковата си част има само нули?
15. По какъв начин се взема предвид признаковата част на наборите при попълването на импликантната таблица?
16. Задължителни ли са за всички функции от системата получените в импликантната таблица задължителни ПИ?
17. В какво се състои минимизацията на система логически функции с използването на базова функция?

ТЕМА 3

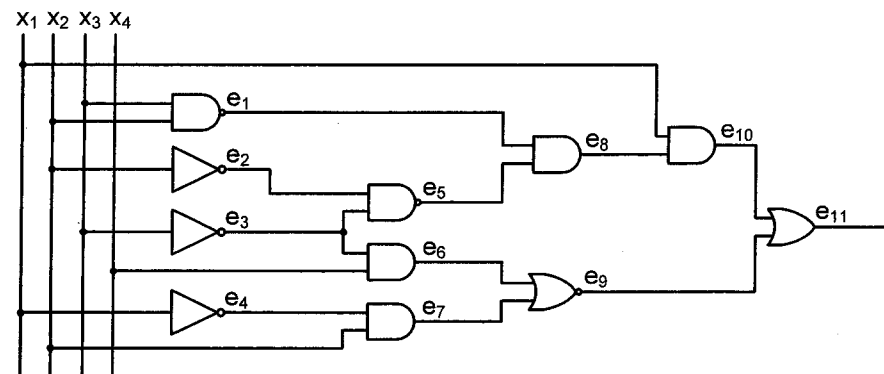
СТАТИЧЕН И ДИНАМИЧЕН АНАЛИЗ НА КОМБИНАЦИОННИ ЛОГИЧЕСКИ СХЕМИ. ВИДОВЕ СЪСТЕЗАНИЯ НА СИГНАЛИТЕ. ОТКРИВАНЕ И ОТСТРАНЯВАНЕ НА СЪСТЕЗАНИЯ

3.1 Статичен анализ

Анализът е процес, който се извършва върху предварително зададена логическа схема. Целите на статичния анализ могат да бъдат изследване на непозната схема или проверка дали определена схема действително реализира зададените при синтеза функции. В резултат на статичния анализ се получават реализираните от схемата функции в аналитичен вид. Статичният анализ се извършва при предположение, че всички сигнали в схемата са идеални и логическите елементи не внасят закъснения. Стъпките за провеждане на статичния анализ са следните:

- 1) Именува се всички входове и изходи на логическите елементи, които съставят схемата.
- 2) Изходните величини се изразяват като функция на входните в съответствие с начина на преобразуване на всеки логически елемент.
- 3) Чрез последователно заместване (суперпозиция) се получава изходната величина като функция на входните променливи.
- 4) Получената форма може да бъде преобразувана в съвършена, ако е необходимо.

Нека проведем статичен анализ на логическата схема, показана на фиг. 3.1, в съответствие с изброените стъпки.



Фиг. 3.1 Примерна логическа схема, използвана за провеждане на анализ

1) Означенията на всички променливи са показани на същата фигура.

$$2) e_1 = \overline{x_2} \cdot x_3 \quad e_2 = \overline{x_2} \quad e_3 = \overline{x_3} \quad e_4 = \overline{x_1} \quad e_5 = e_2 \cdot e_3 \quad e_6 = e_3 \cdot x_4 \\ e_7 = x_2 \cdot e_4 \quad e_8 = e_1 \cdot e_5 \quad e_9 = e_6 \vee e_7 \quad e_{10} = x_1 \cdot e_8 \quad e_{11} = f = e_9 \vee e_{10}$$

3)

$$f = e_9 \vee e_{10} = \overline{e_6} \vee \overline{e_7} \vee x_1 \cdot e_8 = \overline{e_3 \cdot x_4 \vee e_4 \cdot x_2 \vee x_1 \cdot e_1 \cdot e_5} \vee x_1 \cdot \overline{e_3 \cdot x_4 \vee e_4 \cdot x_2 \vee x_1 \cdot e_1 \cdot e_5} = \\ = \overline{x_3 \cdot x_4 \vee \overline{x_1} \cdot x_2 \vee x_1 \cdot x_2 \cdot x_3 \cdot \overline{x_2} \cdot \overline{x_3}} \vee x_1 \cdot \overline{x_3 \cdot x_4 \vee \overline{x_1} \cdot x_2 \vee x_1 \cdot x_2 \cdot x_3 \cdot \overline{x_2} \cdot \overline{x_3}} =$$

$$4) f = \overline{x_3} \cdot x_4 \vee \overline{x_1} \cdot x_2 \vee x_1 \cdot x_2 \cdot x_3 \cdot \overline{x_2} \cdot \overline{x_3} = \overline{x_3} \cdot x_4 \cdot \overline{x_1} \cdot x_2 \vee x_1 \cdot (\overline{x_2} \vee \overline{x_3}) (x_2 \vee x_3) =$$

$$f = (x_3 \vee \overline{x_4}) (x_1 \vee \overline{x_2}) \vee x_1 \cdot (\overline{x_2} \vee \overline{x_3}) (x_2 \vee x_3) =$$

$$= x_1 \cdot x_3 \vee \overline{x_2} \cdot x_3 \vee x_1 \cdot \overline{x_4} \vee \overline{x_2} \cdot \overline{x_4} \vee x_1 \cdot x_2 \cdot \overline{x_3} \vee x_1 \cdot \overline{x_2} \cdot x_3 = x_1 \cdot x_3 \vee \overline{x_2} \cdot x_3 \vee x_1 \cdot \overline{x_4} \vee \overline{x_2} \cdot \overline{x_4} \vee x_1 \cdot x_2 \cdot \overline{x_3}$$

След добавяне на липсващите аргументи и отстраняване на повтарящите се минтерми получаваме СДНФ на функцията:

$$f = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4} \vee \overline{x_1} \cdot \overline{x_2} \cdot x_3 \cdot \overline{x_4} \vee \overline{x_1} \cdot \overline{x_2} \cdot x_3 \cdot x_4 \vee x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4} \vee x_1 \cdot \overline{x_2} \cdot x_3 \cdot \overline{x_4} \vee x_1 \cdot \overline{x_2} \cdot x_3 \cdot x_4 \vee x_1 \cdot x_2 \cdot \overline{x_3} \cdot \overline{x_4} \vee \\ \vee x_1 \cdot x_2 \cdot \overline{x_3} \cdot x_4 \vee x_1 \cdot x_2 \cdot x_3 \cdot \overline{x_4} \vee x_1 \cdot x_2 \cdot x_3 \cdot x_4$$

3.2 Динамичен анализ

Динамичният анализ има за цел да проследи състоянието на изхода на схемата при смяна на един входен набор с друг. Възможно е стойността на изходния сигнал да е една и съща за стария и за новия набор, но в процеса на тяхната смяна на изхода на схемата за кратко време да се получи противоположна стойност на логическия сигнал. Тази погрешна стойност може да повлияе на елементите, свързани към нейния изход, и да доведе до неправилна реакция на цялото цифрово устройство. Първата предпоставка за такова поведение е, че логическите елементи не са идеални, а имат определено време за реакция на входните сигнали, т.е. всеки елемент внася определено времезакъснение. Втората предпоставка е различната дължина на пътищата на сигнала в схемата. Комбинацията от тези две обстоятелства е възможно да доведе до погрешна изходна реакция на изхода на схемата, макар и за кратко време. Тази неправилна изходна реакция се нарича състезание на сигналите. Целта на динамичния анализ е да открие евентуални състезания в схемата. Пълният динамичен анализ предполага проследяване поведението на схемата при смяна на всички възможни двойки набори по 2 пъти.

Динамичният анализ се извършва при следните предположения:

- 1) Времезакъснението на всички логически елементи е еднакво - t .
- 2) Времето за промяна на сигнала от 0 в 1 и обратно е нула, т.е. времето за предния и задния фронт на сигнала е нула.

Динамичният анализ при смяна на един входен набор с друг се извършва в следната последователност:

- 1) Определя се стъпалността на схемата w и се съставя таблица с $w+1$ колони и толкова реда, колкото е броят на входните променливи

плюс броя на логическите елементи, съставляващи схемата. Колоните се номерират от 0 до w , а редовете се именуват с входен сигнал или номер на логически елемент.

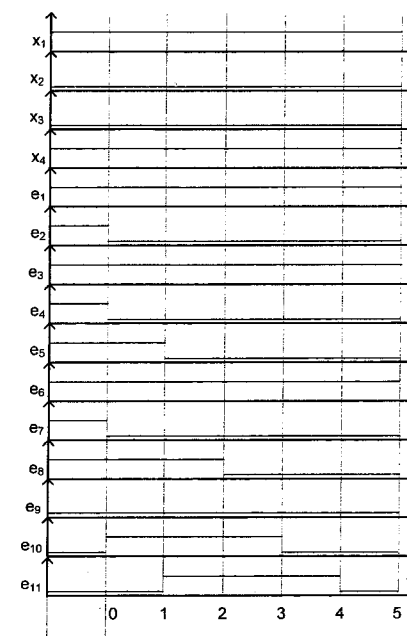
- 2) Намира се реакцията на всеки от логическите елементи за началния входен набор и стойностите се записват на съответните места в най-лявата колона. Вторият зададен набор се записва в съответните места във всички колони.
- 3) Пресмятат се и се записват на съответните места реакции на всеки логически елемент, като при попълване на i -тата колона се ползват стойностите от $(i-1)$ -та.

Нека проведем динамичен анализ на схемата от фиг. 3.1 при смяна на входен набор 0101 с набор 1001.

Стъпалността на схемата е 5, а броят на входните променливи плюс броят на логическите елементи е 15. Съставяме и попълваме таблицата по описания начин – табл. 3.1.

	0	1	2	3	4	5
x_1	1	1	1	1	1	1
x_2	0	0	0	0	0	0
x_3	0	0	0	0	0	0
x_4	1	1	1	1	1	1
e_1	1	1	1	1	1	1
e_2	0	1	1	1	1	1
e_3	1	1	1	1	1	1
e_4	1	0	0	0	0	0
e_5	1	1	0	0	0	0
e_6	1	1	1	1	1	1
e_7	1	0	0	0	0	0
e_8	1	1	1	0	0	0
e_9	0	0	0	0	0	0
e_{10}	0	1	1	1	0	0
e_{11}	0	0	1	1	1	0

Табл. 3.1



Фиг. 3.2 Времедиаграма на динамичния анализ на схемата от фиг. 3.1 при смяна на набор 0101 с набор 1001

Динамичният анализ показва, че преди установяването на изходния сигнал в ниво 0, има период, в който той е 1. Изходният сигнал може да се счита за установен едва след време, равно на резултата от умножението на времезакъснението на един елемент (τ) и стъпалността на схемата. До същия резултат може да се достигне и с помощта на времедиаграма (фиг. 3.2), която отразява състоянието на изходите на всички логически елементи за периода на установяване на сигналите.

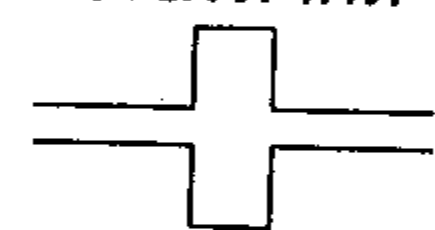
Една от важните задачи при синтеза на логическите схеми е предотвратяването на състезания в схемите. Динамичният анализ е универсален метод за откриване на състезания, но е изключително трудоемък, като се има предвид, че за да бъде пълен, трябва да се приложи за всички двойки набори по два пъти. Ето защо се търсят и други методи за откриване и борба със състезанията.

3.3 Видове състезания

Според броя на позициите, в които се различават двата набора, чиято смяна предизвиква състезанията, биват: 1-позиционни, 2-позиционни, ..., n-позиционни.

Според вида на реакцията на изхода на схемата при двата набора състезанията биват:

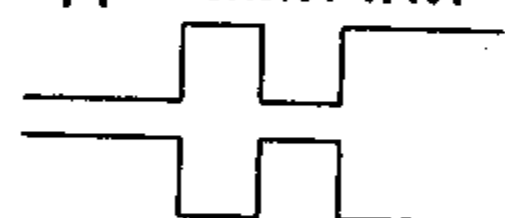
- статични – реакцията на схемата за двата набора е една и съща.



0 статично

1 статично

- динамични – реакцията на схемата за двата набора е различна.



0-1 динамични

1-0 динамични

Най-лесни за откриване и отстраняване са 1-позиционните състезания. Доказва се, че схема, свободна от 1-позиционни статични състезания е свободна и от 1-позиционни динамични състезания.

3.4. Откриване и отстраняване на състезанията

Нека да анализираме реакциите на логическите елементи И и ИЛИ от гледна точка на поява на състезания при промяна на сигналите на входовете им.

От табл. 3.2 се вижда, че състезания са възможни, когато имаме едновременно разнопосочна промяна в стойностите на входните сигнали. За n-входови логически елементи И и ИЛИ условията за състезания ще бъдат две. Първо, поне на два от входовете трябва да има разнопосочни промени, и второ – сигналът на всички останали входове да не държи изхода в постоянно ниво. За логически елемент И това означава на никой

от останалите входове да няма нула, а за логически елемент ИЛИ – единица. Откриването на възможността за поява на състезания в комбинационните схеми се свежда до последователна проверка за изпълнение на тези две условия, като се започне от изходите към входовете на схемата.

Нека разгледаме възможностите за съществуване на едноразположени еднопозиционни състезания в двустъпална схема. Такива състезания ще са възможни при смяна на набори, които са:

- съседни
- за които функцията има стойност 1
- двата набора не се покриват от една и съща проста импликанта.

Оттук се налага изводът, че за да няма състезания в двустъпална схема, тя трябва да се реализира не в минимална форма, а като дизюнкция от всички нейни прости импликанти. Като илюстрация на тези разсъждения нека да разгледаме функцията, зададена с карта на Вейч на фиг. 3.3.

Входни променливи		Изход на лог. елемент И	Изход на лог. елемент ИЛИ
x ₁	x ₂		
0	0	0	0
0	0-1	0	0-1
0	1-0	0	1-0
0	1	0	1
0-1	0	0	0-1
0-1	0-1	0-1	0-1
0-1	1-0	0-1-0	1-0-1
0-1	1	0-1	1
1-0	0	0	1-0
1-0	0-1	0-1-0	1-0-1
1-0	1-0	1-0	1-0
1-0	1	1-0	1
1	0	0	1
1	0-1	0-1	1
1	1-0	1-0	1
1	1	1	1

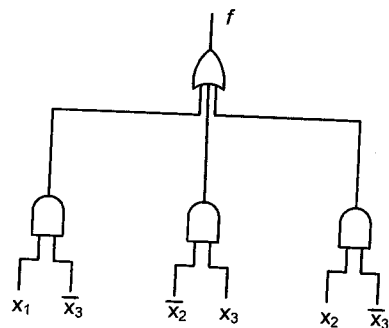
Това са случаите на неправилна изходна реакция

Табл. 3.2 Промяна на състоянието на изходите на логическите елементи при промяна на входните сигнали

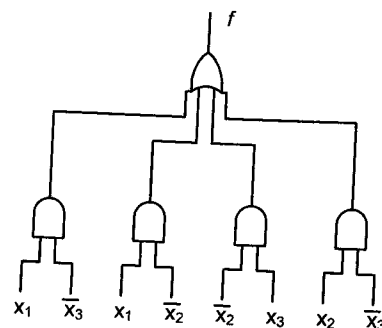
Минималната форма на функцията от фиг.3.3 е $f = x_1\bar{x}_3 \vee \bar{x}_2x_3 \vee \bar{x}_3x_2$. (Простите импликанти, оградени с плътен контур). Ако реализираме схема на тази функция (фиг. 3.4), ще има състезания при смяна на набори 4 и 5. За да елиминираме възможността за едновременно разнопосочна смяна в стойностите на сигналите, които се подават на входовете на логическия елемент ИЛИ, трябва да добавим логически елемент И, който реализира и останалата проста импликанта на функцията $x_1\bar{x}_2$. (ПИ, оградена с прекъснат контур). Схемата е показана на фиг.3.5.

	x_1			
x_2	1	0	0	1
	1	1	1	0
	x_3			

Фиг. 3.3 Логическа функция, нанесена в карта на Вейч с означени всички ПИ



Фиг. 3.4 Схема, в която има условия за 1-позиционни 1-статични състезания



Фиг. 3.5 Схема, свободна от 1-позиционни 1-статични състезания

Доказва се, че двустъпална И-ИЛИ схема, свободна от 1-позиционни 1-статични състезания е свободна от всякакви 1-позиционни състезания (0-статични и динамични), но това не означава, че схемата е свободна въобще от r-позиционни състезания. Например при смяна на набор 2 с набор 4 и в двете схеми има 2-позиционно 1-статично състезание. Като изход от проблема могат да се наложат следните ограничения:

- 1) Входната последователност да се състои само от съседни набори.

- 2) Схемата да е построена като дизюнкция от всички ПИ на функцията.

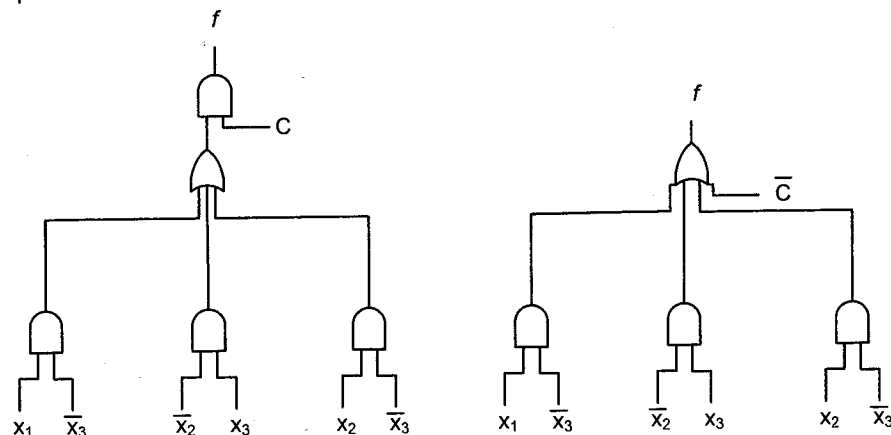
Друг подход за решаване на проблемите, свързани със състезанията, се състои не в тяхното избягване, а в предотвратяване на отрицателните последици за схемите, които се управляват от изходите на схемата със състезания. Налагат се следните ограничения:

- 1) Всеки входен набор се подава достатъчно дълго време, за да се установят правилни стойности на всички изходни сигнали. Изходният сигнал може да се счита за установен едва след време, равно на резултата от умножението на времезакъснението на един елемент (τ) и стъпалността на схемата.

- 2) Изходните сигнали на схемата не се ползват непрекъснато, а само в точно определени моменти от време. Тези моменти се определят от т.нар. тактови или синхронизиращи сигнали, обикновено означавани с "C" или "Clock". При $C = 1$ реакцията на схемата се ползва, а при $C = 0$ – не се ползва. Времето, през което C има стойност 0, трябва да е достатъчно, за да се разпространят сигналите по най-дългия път от вход към изход на схемата.

Такива схеми се наричат синхронизирани. По принцип синхронизираните схеми са по-бавни, но за сметка на това са гарантирано свободни от състезания.

На фиг. 3.6 са показани два варианта на синхронизиране на схемата от фиг. 3.4.



Фиг. 3.6 Два варианта на синхронизиране на схемата от фиг.3.4

Задание

1. Да се направи статичен анализ на зададена схема. Да се състави таблица на истинност на логическата функция според получения булев израз.
2. Експериментално да се снее таблицата на истинност и да се сравнят резултатите.
3. Да се направи динамичен анализ на зададена схема за определени входни набори.
4. Да се предложи вариант за същата схема, но свободна от 1-позиционни 1-статични състезания.
5. Да се предложи вариант за синхронизиране на схемата.

Контролни въпроси

1. Какви са възможните цели на статичния анализ?
2. Какви са стъпките, през които преминава статичният анализ?
3. Каква е целта на динамичния анализ?
4. Какво означава „състезания на сигналите“? Какви са видовете състезания?
5. Какви са стъпките, през които преминава динамичният анализ?
6. Кое е предимството и кой е недостатъкът на метода за откриване на състезания „динамичен анализ“?
7. Какви са условията за състезания в схеми, построени с логически елементи И и ИЛИ?
8. Защо 1-позиционни 1-статични състезания в двустъпална схема се търсят между набори, които са съседни, единични и не се покриват от една ПИ?
9. Схема, свободна от 1-позиционни 1-статични състезания, свободна ли е изобщо от състезания?
10. Кое е предимството и кой е недостатъкът на синхронизираните схеми?

ТЕМА 4

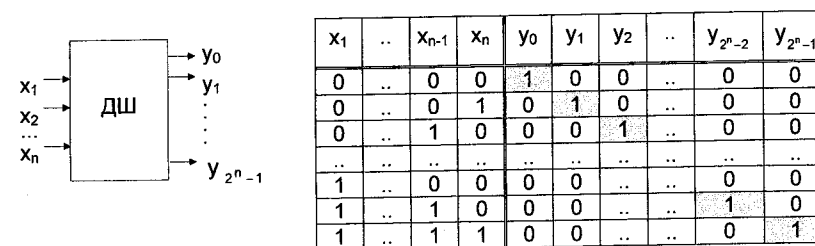
ДЕШИФРАТОРИ И ДЕМУЛТИПЛЕКСОРИ. МУЛТИПЛЕКСОРИ. ПОСТОЯННИ ПАМЕТИ. ПРОГРАМИРУЕМИ ЛОГИЧЕСКИ МАТРИЦИ. ШИФРАТОРИ. КОМПАРАТОРИ. КОДОВИ ПРЕОБРАЗОВАТЕЛИ. СУМАТОРИ

4.1 Дешифратори и демултиплексори. Реализация на логически функции чрез дешифратори

Дешифраторите или декодерите (decoders) принадлежат към класа на комбинационните схеми – структури от логически елементи, при които изходната функция (функции) се получава директно при прилагане на конкретна комбинация от входни сигнали и при които липсва времева задръжка. Към този клас схеми принадлежат също и мултиплексорите, преобразувателите на код, суматорите, схемите за изваждане, за сравнение, схемите за контрол и аритметичните логически устройства (АЛУ).

Комбинационните устройства представляват елементни възли, обикновено със средна степен на интеграция (MSI, Medium Scale Integration, до 10^3 елемента), включващи относително голям брой логически елементи, свързани по между си в многостъпални структури по начин, позволяващ изпълнението на конкретната специфична функционалност.

Дешифраторите (ДШ) са устройства, притежаващи n на брой входа и съответно $m=2^n$ брой изхода (пълни дешифратори) или $m < 2^n$ (непълни или частични дешифратори). Във всеки един момент дешифраторът има един-единствен активен изход, а всички останали са неактивни. При това, за конкретната комбинация от входни сигнали е активен точно определен изход – т.е. дешифраторът разпознава (идентифицира) подадения на входните шини двоичен код. На фиг.4.1 са показани блоковата схема на пълен дешифратор, таблицата на истинност и съответните уравнения в изходите му.



фиг.4.1 Блокова схема, таблица на истинност и уравнения на изходите на пълен дешифратор с "прави" изходи

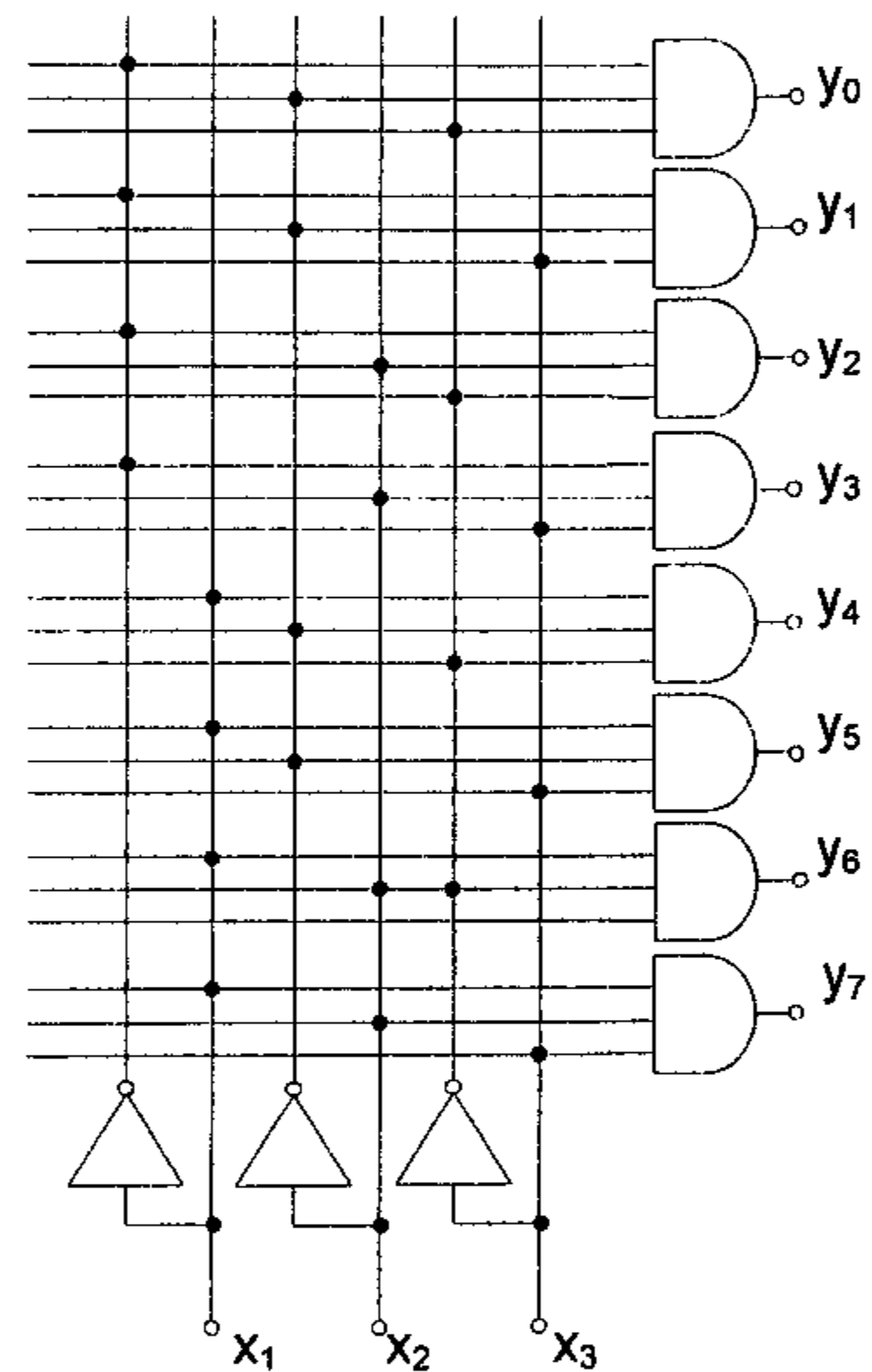
$$\begin{aligned}
y_0 &= \bar{x}_1 \bar{x}_2 \dots \bar{x}_{n-1} \bar{x}_n = \overline{x_1 \vee x_2 \vee \dots \vee x_{n-1} \vee x_n} \\
y_1 &= \bar{x}_1 \bar{x}_2 \dots \bar{x}_{n-1} x_n = \overline{x_1 \vee x_2 \vee \dots \vee x_{n-1} \vee \bar{x}_n} \\
y_2 &= \bar{x}_1 \bar{x}_2 \dots x_{n-1} \bar{x}_n = \overline{x_1 \vee x_2 \vee \dots \vee \bar{x}_{n-1} \vee x_n} \\
&\vdots \\
y_{2^{n-2}} &= x_1 x_2 \dots x_{n-1} \bar{x}_n = \overline{\bar{x}_1 \vee \bar{x}_2 \vee \dots \vee \bar{x}_{n-1} \vee x_n} \\
y_{2^{n-1}} &= x_1 x_2 \dots x_{n-1} x_n = \overline{\bar{x}_1 \vee \bar{x}_2 \vee \dots \vee \bar{x}_{n-1} \vee \bar{x}_n}
\end{aligned}$$

Прието е размерността на дешифраторите да се означава по следния начин: $n \rightarrow 2^n$ (или $n \rightarrow m$ при непълен ДШ), например $2 \rightarrow 4$, $3 \rightarrow 8$, $4 \rightarrow 16$. Фиг.4.2 представя примерна схема с логически елементи на вътрешната структура на ДШ с размерност $3 \rightarrow 8$.

Различаваме ДШ с активно високо ниво в изходите си (единица на "фон" от нули) или с активно ниско ниво (нула на "фон" от единици). Практическите схеми на ДШ обикновено са такива с активно ниско изходно ниво. Двоичното тегло на входните сигнали е прието да се означава с главни букви А, В, С и т.н., като най-ниско е това на вход А.

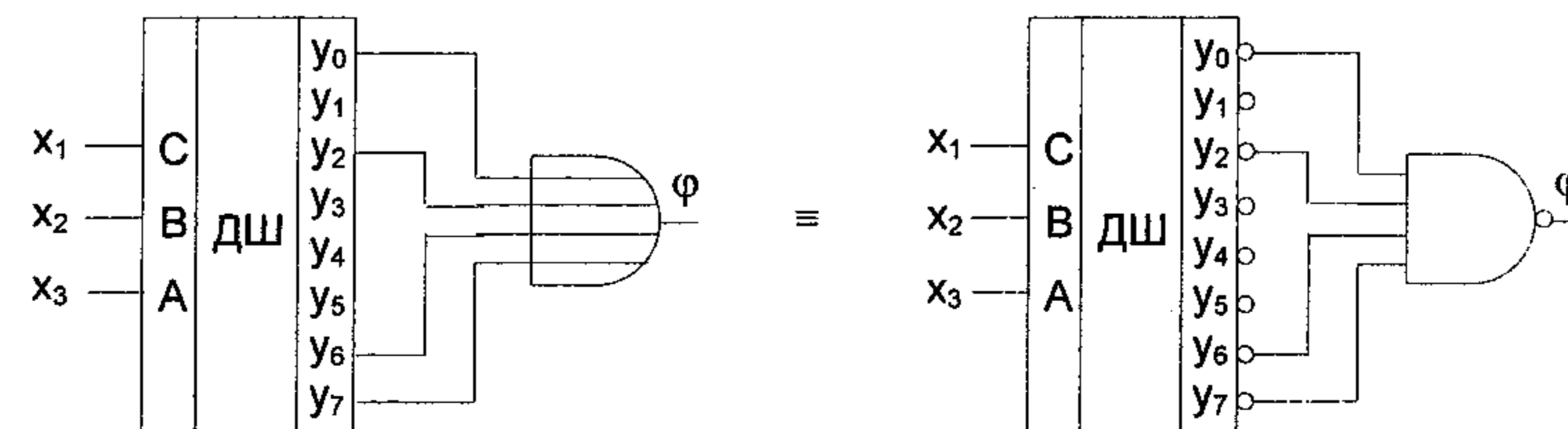
В интегрално изпълнение схемите имат още отделен *разрешаващ вход* Е (enable), чрез който се разрешава използването на цялата схема. Освен това, в рамките на чипа в една интегрална схема (ИС) могат да бъдат разположени повече от една схема на дешифратор. Например, често използваната ИС 74155 включва два ДШ $2 \rightarrow 4$ с общи входове, като всеки от дешифраторите има отделен вход $1G, 2G$, разрешаващ схемите с ниско ниво, и още по един вход $1C, 2C$, разрешаващ всяка от схемите с противоположен по ниво сигнал и така предоставящ възможност за лесно разширяване разрядността на схемата до ДШ $3 \rightarrow 8$.

Както може да се забележи от дадените на фиг.4.1 уравнения на изходите, с помощта на пълен ДШ $n \rightarrow 2^n$ с "прави" изходи и 2^n -входова логическа схема ИЛИ може да се реализира произволна функция на n променливи, представена в СДНФ. Ако функцията не е представена в



фиг.4.2 Вътрешна структура на ДШ с размерност $3 \rightarrow 8$

канонична форма, тя предварително следва да се преобразува в такава (вж. гл.1).



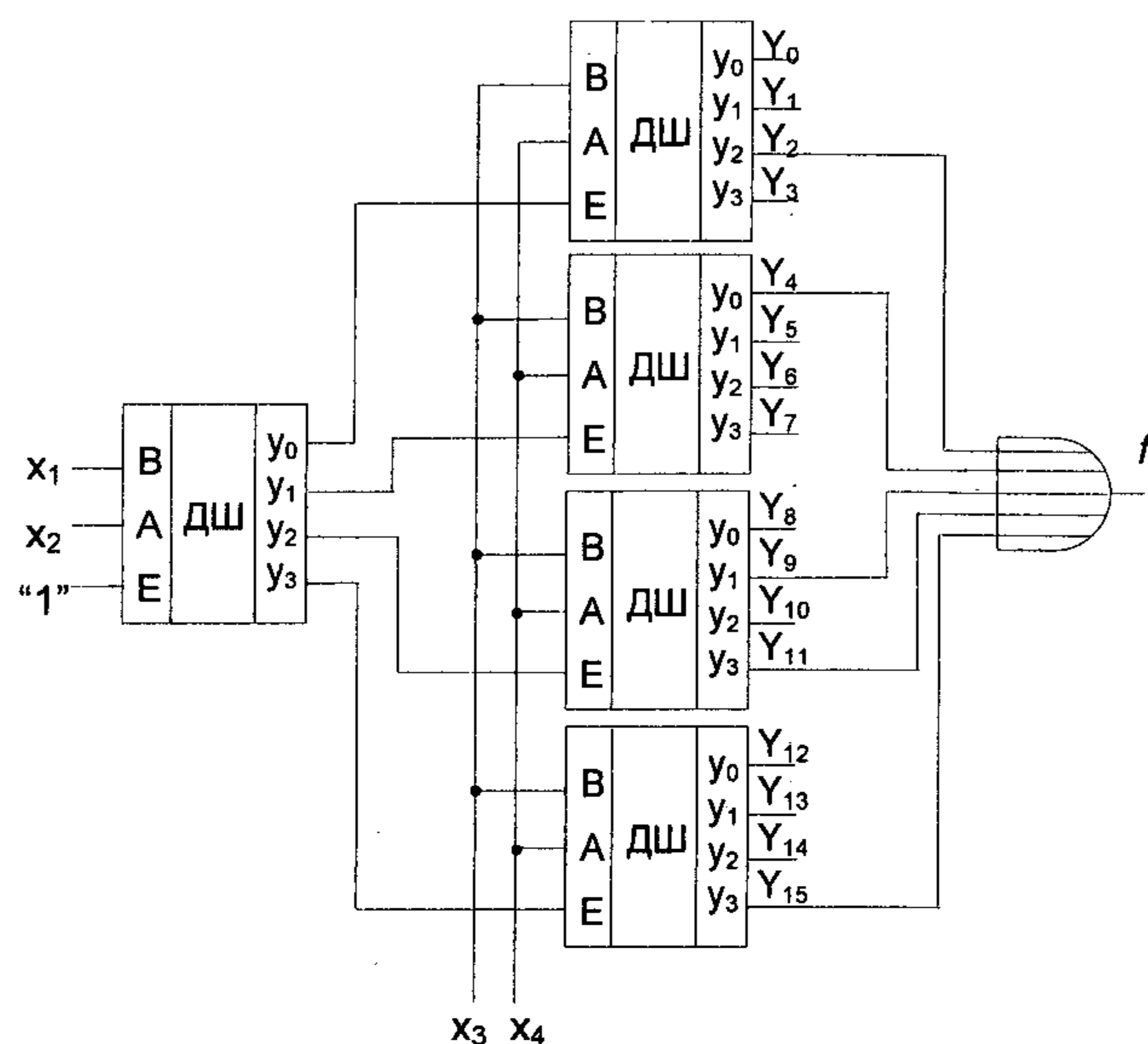
Фиг.4.3 Реализация на логическата функция $\phi = v(0, 2, 6, 7)$ чрез пълни дешифратори $3 \rightarrow 8$ с прави и инверсни изходи и логически схеми ИЛИ, респ.И-НЕ

Същото може да се постигне, ако се разполага с пълен ДШ с размерност n с инверсни изходи и 2^n -входова схема И-НЕ (доказва се чрез инвертиране на дясната част от уравненията на изходите и последващо прилагане на закона на Де Морган). На фиг.4.3 е дадена реализацията на функция на три променливи по описаните по-горе два начина, като функцията е зададена с номерата на единичните си набори в СДНФ.

При положение, че не разполагаме със схеми на дешифратор с необходимата разрядност, то се прилага т.нар. *каскадно свързване* на схеми с по-малка разрядност, като крайната схема на ДШ е двустъпална. Младшите входни сигнали се подават едновременно на входовете на всички схеми от второто стъпало, а старшите – на входовете на дешифратора от първото стъпало. Схемата от първото стъпало чрез своите изходи разрешава през входовете Е (Enable) един от ДШ от второто стъпало.

Фиг.4.4 представя пример на реализация на ДШ с размерност $4 \rightarrow 16$, използвайки изграждащи дешифратори $2 \rightarrow 4$. Тук приемаме, че входът Е разрешава схемите с високо ниво, а всички ДШ имат "прави" изходи. Ако разрешението на схемите е с ниско ниво, то и изходите на участващите схеми следва да са инверсни, а логическата схема в изхода да бъде И-НЕ.

За примера от фиг.4.4 сигналите x_1, x_2, x_3 и x_4 ще съответстват на входовете D, C, В и А по отношение на крайната схема на ДШ $4 \rightarrow 16$, а изходите на изграждащите дешифратори - съответно на изходи $y_2, y_4, y_9, y_{11}, y_{15}$ на крайната схема. Ако в горния пример реализираната функция не включваше нито един изход от даден изграждащ дешифратор (напр. y_4), то и съответната включена схема (в случая – втората от горе надолу) би била излишна.

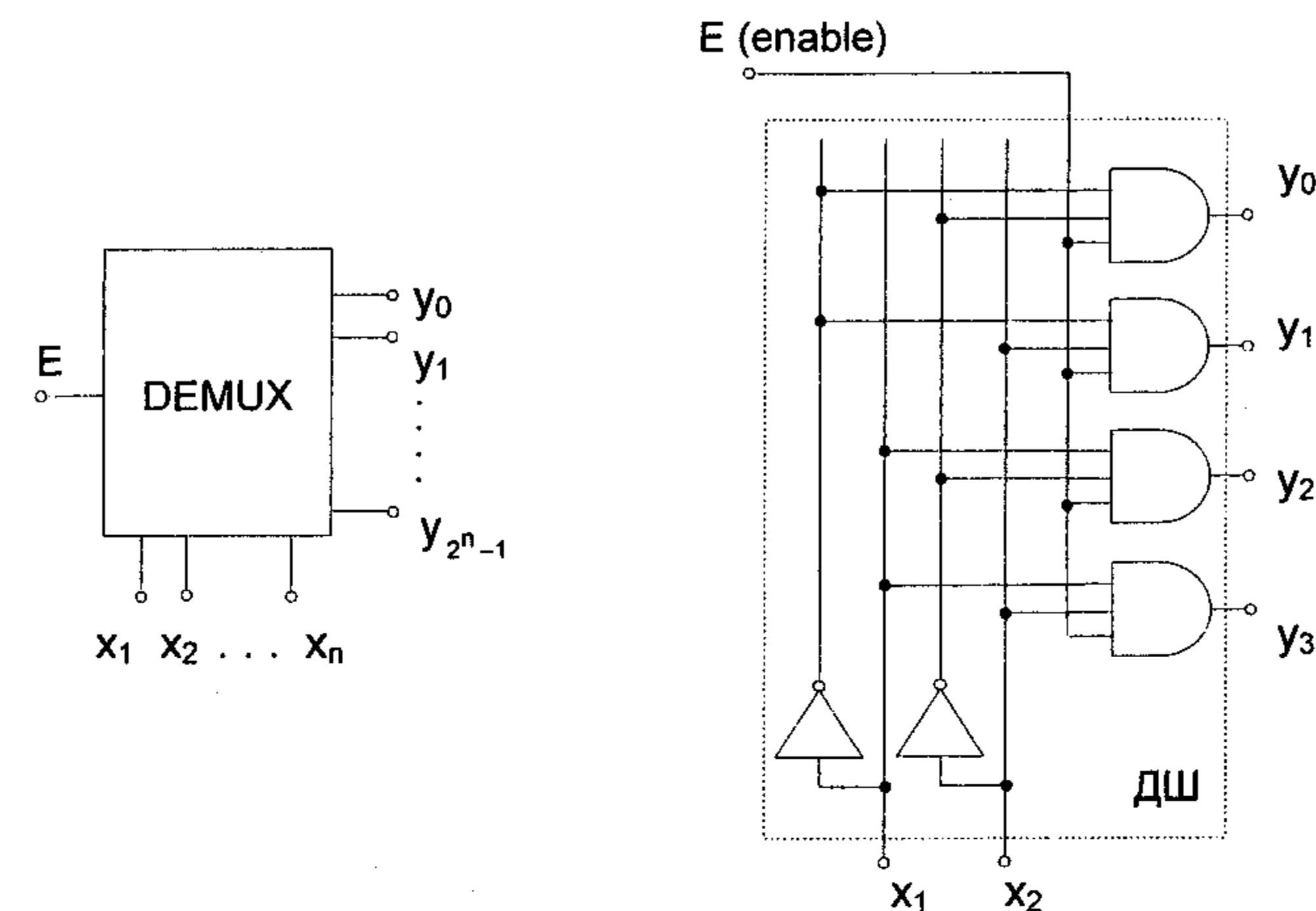


Фиг.4.4 Каскадна реализация на функцията $f = v(2, 4, 9, 11, 15)$ с помощта на дешифратори 2→4 и схема ИЛИ

Демултиплексорът (означаван често с DEMUX) има функцията да предава сигнала от единствения си вход s към един от своите 2^n ($y_0, y_1, \dots, y_{2^n-1}$) изхода в зависимост от подадената на неговите n на брой адресни (селекторни, управляващи) шини (x_1, x_2, \dots, x_n) двоична комбинация. С други думи, демултиплексорът представлява многопозиционен ключ, който комутира сигнала на входа към съответен изход в зависимост от зададения цифров код на селекторните му шини.

Обща блокова схема на цифров демултиплексор, както и примерна логическа структура на DEMUX 1→4 са дадени на фиг.4.5. Както може да се забележи, схемата на демултиплексор се получава директно от тази на пълен ДШ с разрешаващ вход. Поради тази причина самостоятелни схеми на демултиплексори в интегрално изпълнение не се произвеждат, а се ползват тези на ДШ с необходимата размерност.

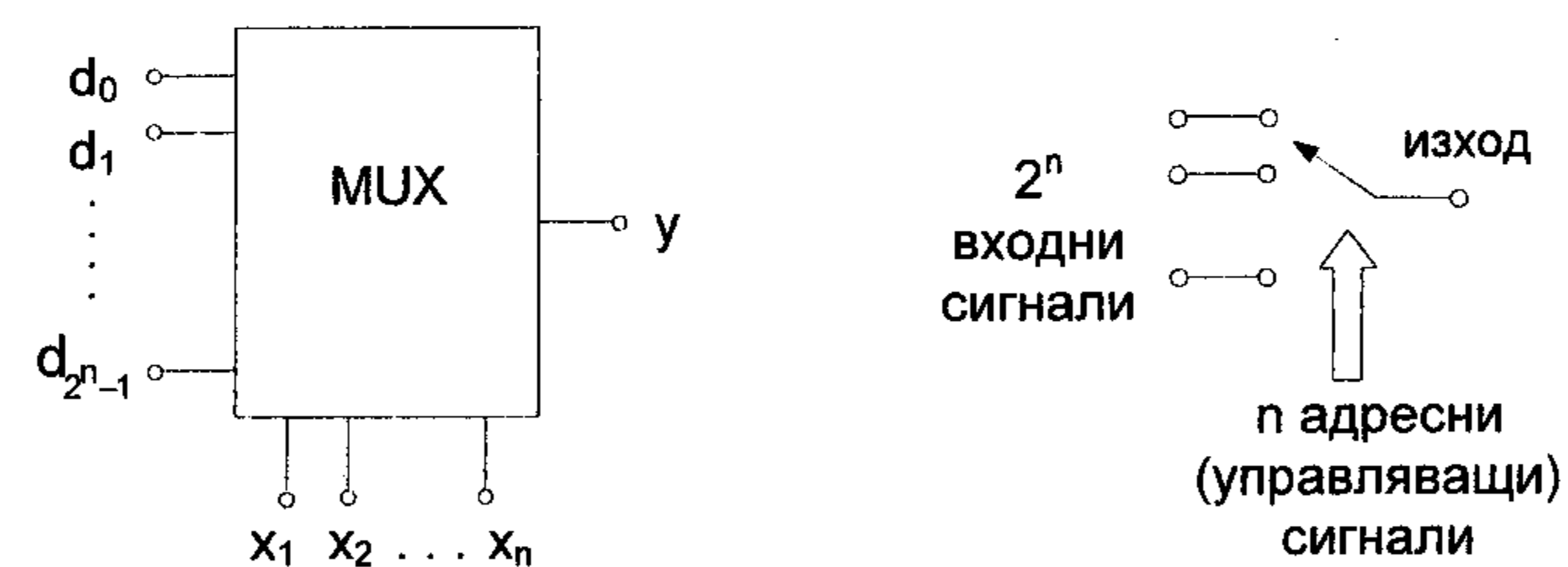
За дискутираната преди в текста ИС 74155 преобразуването ѝ в DEMUX 1→4 (респ. 1→8) става, като за информационен служи входът $\overline{1G}, \overline{2G}$, а адресни са входовете A, B (респ. A, B, C).



фиг.4.5 Демултиплексор – обща блокова и примерна логическа структура на DEMUX 1→4

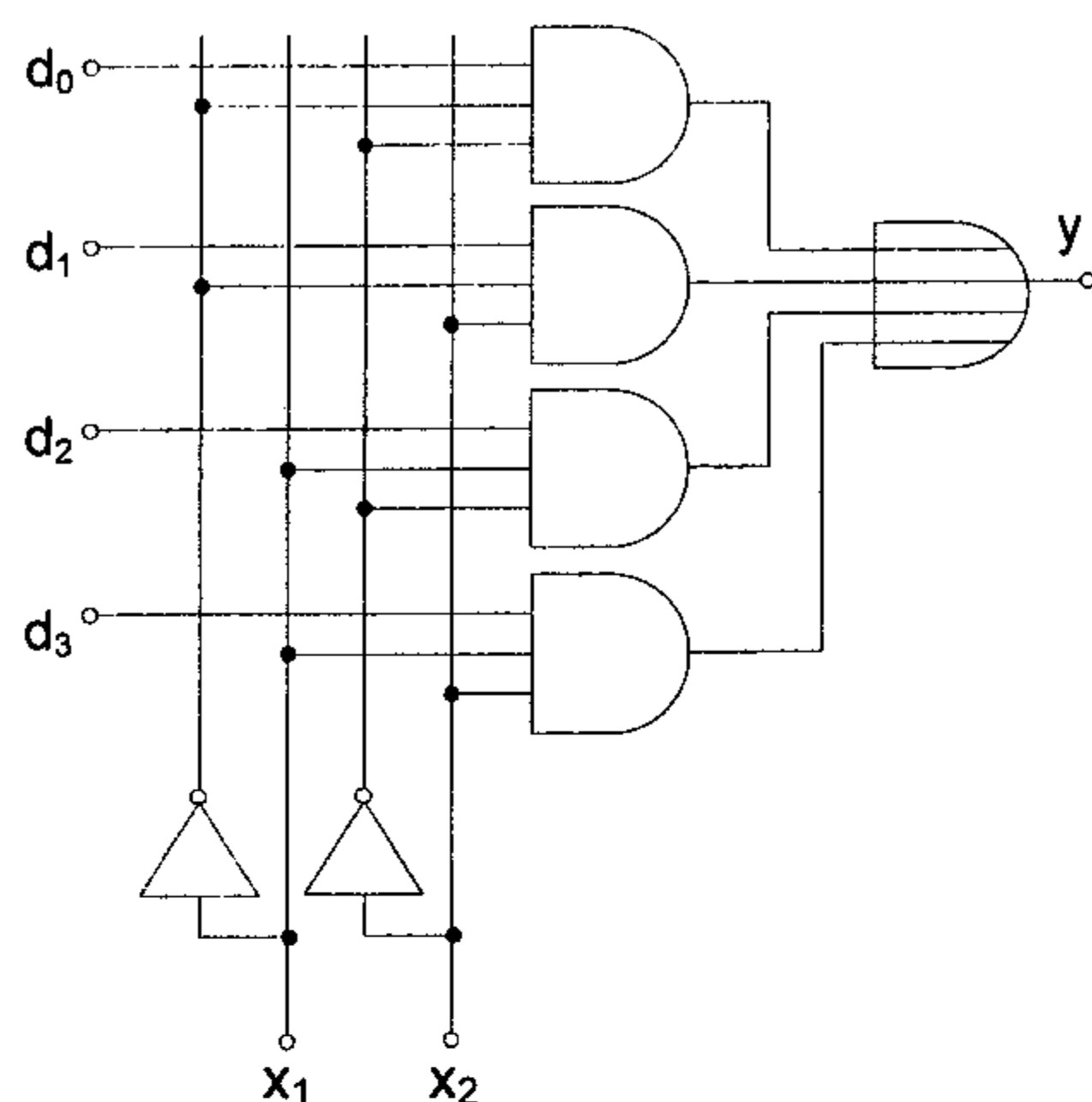
4.2 Мултиплексори

Чрез мултиплексор (MUX) се извършва предаване на информацията от един от 2^n ($d_0, d_1, \dots, d_{2^n-1}$) на брой входни (информационни) сигнали към единствения изход y , като изборът на входния канал се



фиг.4.6 Мултиплексор: обща блокова схема, функционален модел

осъществява посредством двоичен код, подаден на n на брой адресни (управляващи) сигнали (x_1, x_2, \dots, x_n). По аналогия с демултиплексора, схемата на MUX може да се разглежда отново като такава на многопозиционен ключ, чието положение се определя от подадената двоична комбинация на адресните шини – фиг.4.6.



фиг.4.7 Логическа схема на цифров мултиплексор 4→1

Схемата на мултиплексор може да се разглежда като получена от такава на ДШ, в която към схемите И е добавен по още един вход, а в изхода – схема ИЛИ. Фиг.4.7 представя логическата структура на цифров MUX с разрядност 4→1. В интегралните структури на мултиплексори, както и в повечето практически ИС със средна степен на интеграция, присъства допълнителен разрешаващ (стробиращ) вход, означаван обикновено с E, enable (в ИС 74153 - G). Прието е също, както в схемите на ДШ/DEMUX, селектиращите входове да се означават с A,B,C и т.н.

Функцията на изхода на мултиплексор се дава с израза:

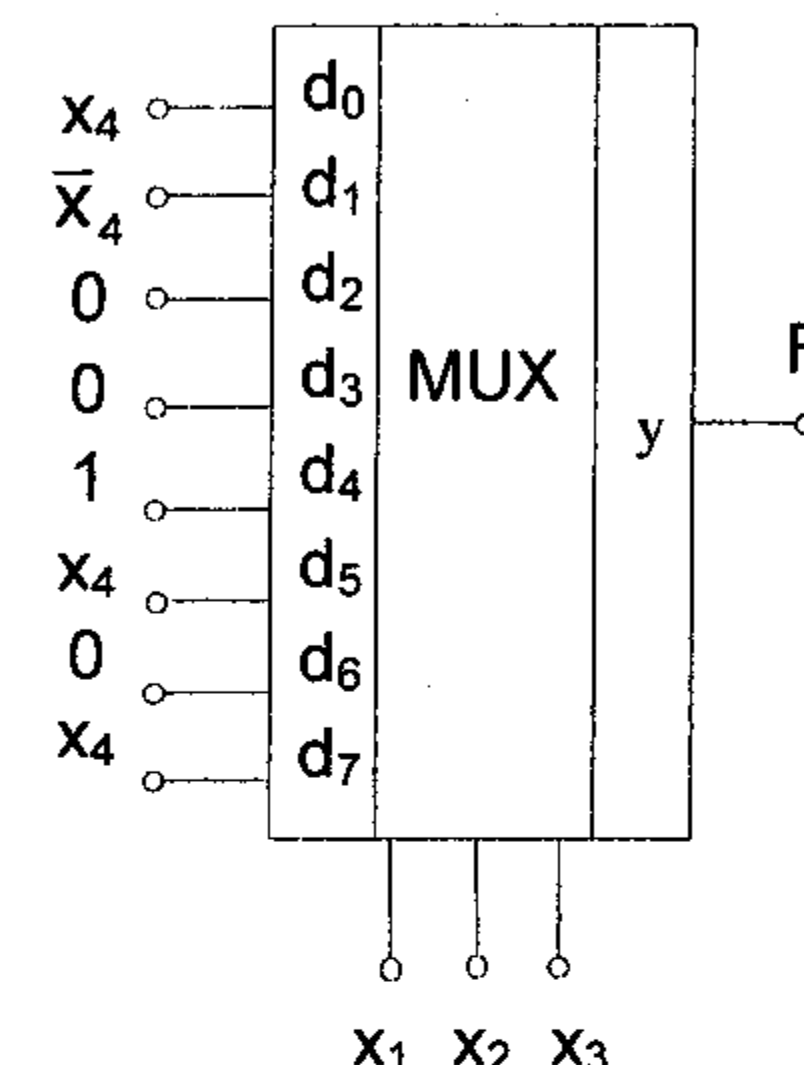
$$y = d_0 \bar{x}_1 \bar{x}_2 \dots \bar{x}_n \vee d_1 \bar{x}_1 \bar{x}_2 \dots x_n \vee \dots \vee d_{2^n-1} x_1 x_2 \dots x_n =$$

$$= d_0 m_0 \vee d_1 m_1 \vee \dots \vee d_{2^n-1} m_{2^n-1}$$

Посредством мултиплексор с n на брой управляващи входове може да се реализира произволна функция на n+1 аргумента. Това е така, тъй като адресните входове генерират общо 2^n набора на n аргумента (m_i), а съответният информационен вход d може да заема две стойности 0 и 1, т.е. могат да бъдат покрити всички набори на n+1 входни променливи. Така в зависимост от конкретната изисквана за реализиране функция, на входовете d_i се подава 0,1, x_{n+1} или \bar{x}_{n+1} . Фиг.4.8 илюстрира пример за реализация на функцията на четири аргумента чрез MUX с размерност 8→1. Представена е и съответната таблица на истинност, от която се извършва подборът на подаваните на информационните входове стойности по отношение на аргумента x_4 .

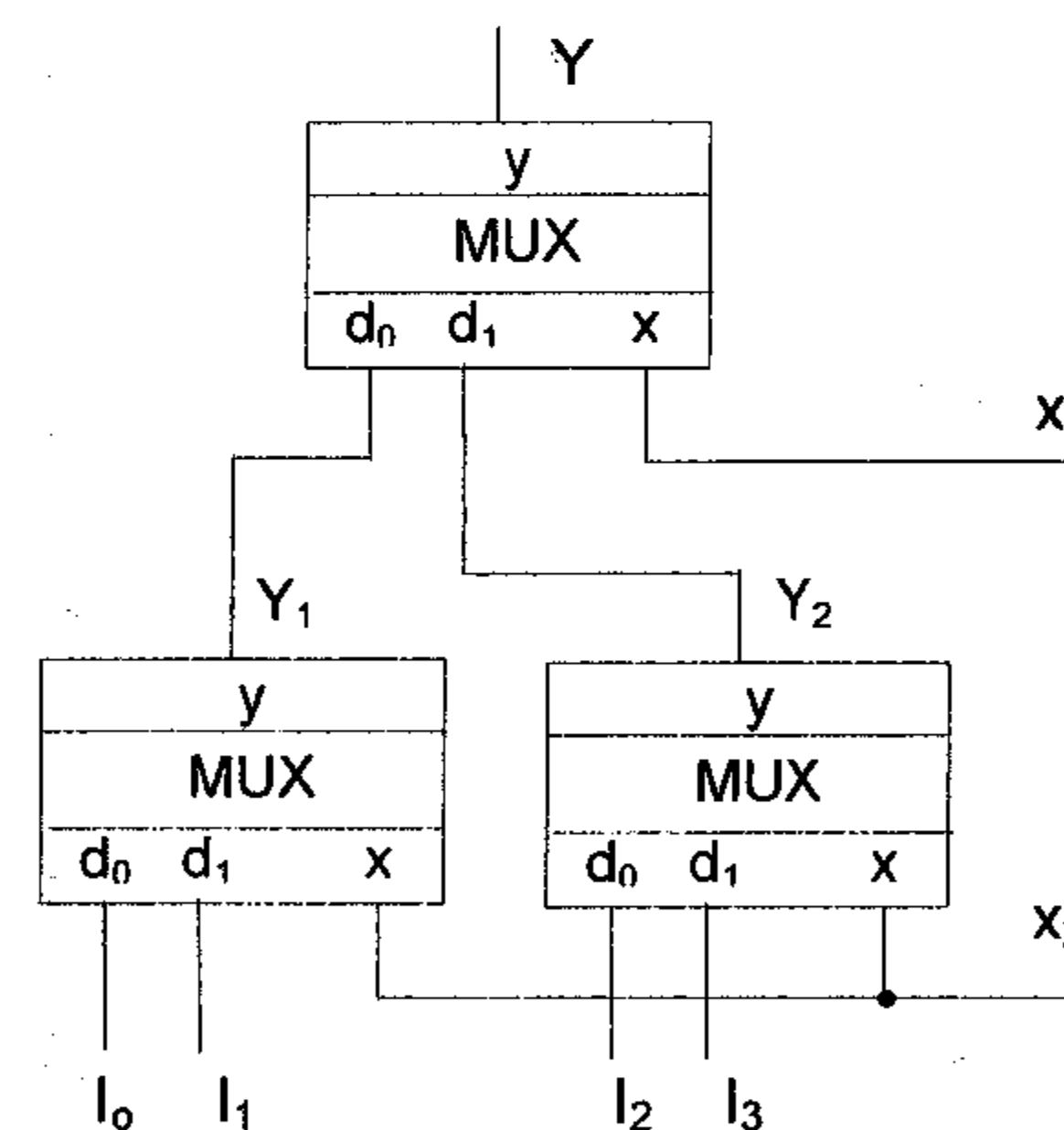
Както и при разгледаните по-рано в настоящата глава схеми на дешифратори, и тук мултиплексиране на входни сигнали с висока разрядност може да се постигне чрез каскадно (последователно) свързване на няколко мултиплексора с по-ниска разрядност. В този случай младшите адресни сигнали адресират изграждащите мултиплексори от първото стъпало, а старшите – тези от второто. На фиг.4.9 е показана реализация на мултиплексор с размерност 4→1 чрез мултиплексори 2→1 (с 1 адресиращ вход). Двата мултиплексора от входното стъпало се адресират от младшия адресен бит x_2 , а изходният

MUX – от x_1 . За стойността на изходната функция в тази каскадна реализация може да се запише изразът, даден вдясно от фиг.4.9.



x_1	x_2	x_3	x_4	F	d_i
0	0	0	0	0	$d_0 = x_4$
0	0	0	1	1	
0	0	1	0	1	$d_1 = \bar{x}_4$
0	0	1	1	0	
0	1	0	0	0	$d_2 = 0$
0	1	0	1	0	
0	1	1	0	0	$d_3 = 0$
0	1	1	1	0	
1	0	0	0	1	$d_4 = 1$
1	0	0	1	1	
1	0	1	0	0	$d_5 = x_4$
1	0	1	1	1	
1	1	0	0	0	$d_6 = 0$
1	1	0	1	0	
1	1	1	0	0	$d_7 = x_4$
1	1	1	1	1	

фиг.4.8 Реализация на логическата функция на четири променливи $F = \vee(1,2,8,9,11,15)$ чрез мултиплексор 8→1: схема и таблица на истинност



фиг.4.9 Каскадна реализация на мултиплексор с размерност 4→1 чрез схеми MUX 2→1

$$Y_1 = I_0 \bar{x}_2 \vee I_1 x_2; Y_2 = I_2 \bar{x}_2 \vee I_3 x_2$$

$$Y = Y_1 \bar{x}_1 \vee Y_2 x_1 =$$

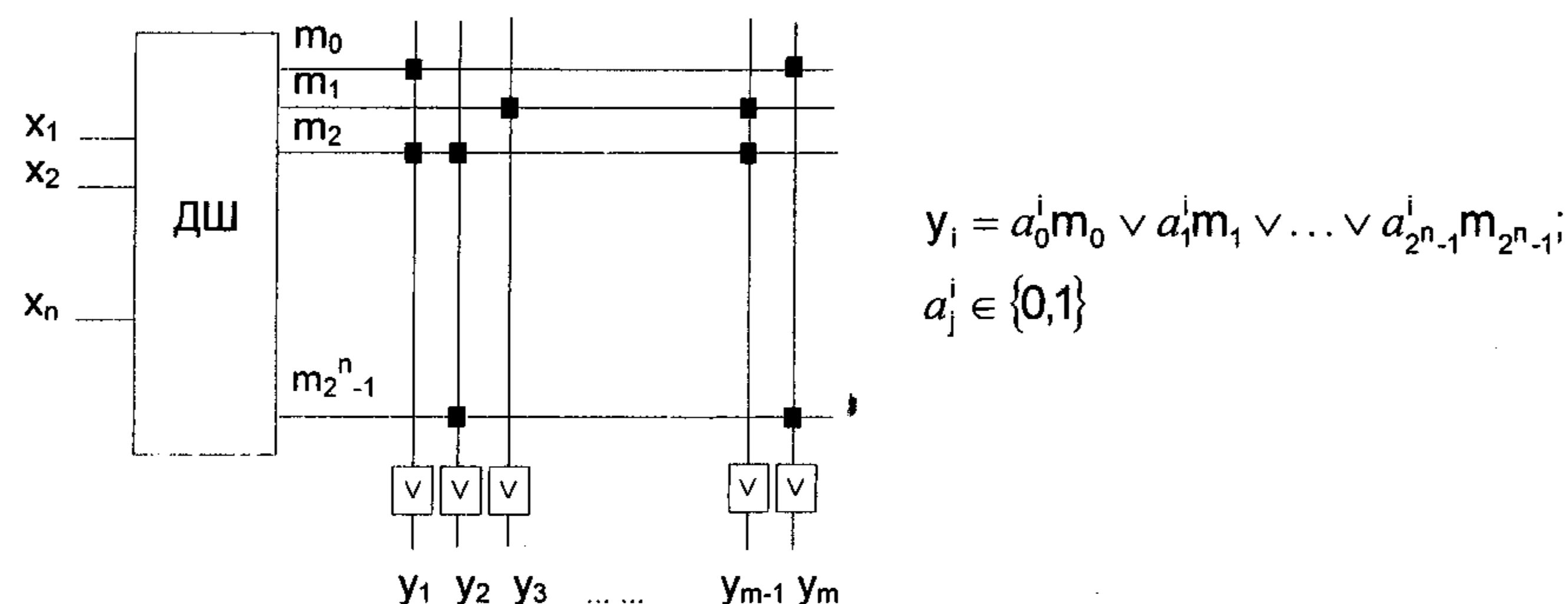
$$= (I_0 \bar{x}_2 \vee I_1 x_2) \cdot \bar{x}_1 \vee (I_2 \bar{x}_2 \vee I_3 x_2) \cdot x_1 =$$

$$= I_0 \bar{x}_1 \bar{x}_2 \vee I_1 \bar{x}_1 x_2 \vee I_2 x_1 \bar{x}_2 \vee I_3 x_1 x_2$$

На всеки от информационните входове $I_0 \div I_3$ може да постъпва произволна цифрова променлива. Ако на тези входове обаче се подава една от възможните стойности на една двоична променлива – напр. x_3 (вж. примера от фиг.4.8), може да се реализира логическа функция с разрядност с 1 по-висока от тази на адресните входове.

4.3. Постоянни памети. Програмируеми логически матрици (ПЛМ). Реализация на логически функции с ПЛМ

Постоянните запомнящи устройства (ПЗУ) поради своята универсална и регулярна структура са често използвани комбинационни структури. Към тях спадат класическите схеми на постоянни памети (Read-Only Memory, ROM или ROS, Read-Only Storage) и програмируемите памети (Programmable ROM, PROM). Тези структури принадлежат към класа на комбинационните структури, тъй като макар и да изпълняват функцията "памет", това става с фиксиране стойността на запомнената двоична информация преди използване на самата схема. Предназначението на този тип схеми е да съхраняват

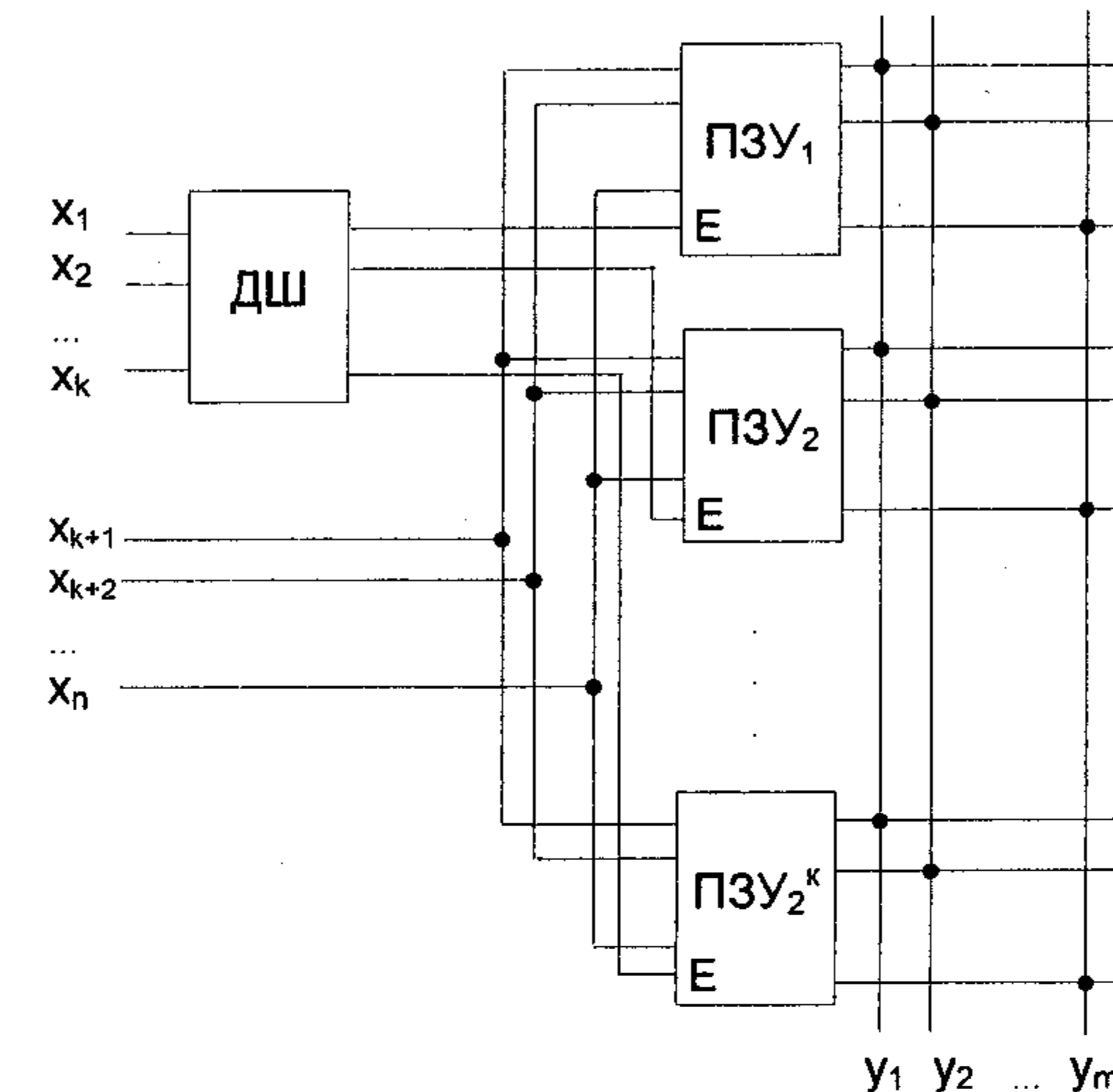


Фиг.4.10 Постоянно запомнящо устройство (ПЗУ): структура, изходни функции

предварително заложена и относително постоянна във времето информация, като предоставят възможност само за четене от тях. Такива схеми са широко използвани в персоналните компютри за съхранение на ядрото на операционната система, в специализираните микроконтролери (едночипови микрокомпютри, ЕМК) и др. Освен за съхраняване на постоянна информация паметите от тип ROM служат и за изграждане на блокове за реализация на сложни логически операции с използване на т.нар. програмируеми логически матрици, ПЛМ (PLA, Programmable Logic Arrays).

ПЗУ се състоят от двустъпална структура, даваща възможност за реализация на произволна функция на n входни променливи. Входното стъпало на устройството представлява пълен дешифратор, изходите на който реализират всички минчленове на входните променливи (m_i , $i=0, \dots, 2^n-1$) – фиг.4.10.

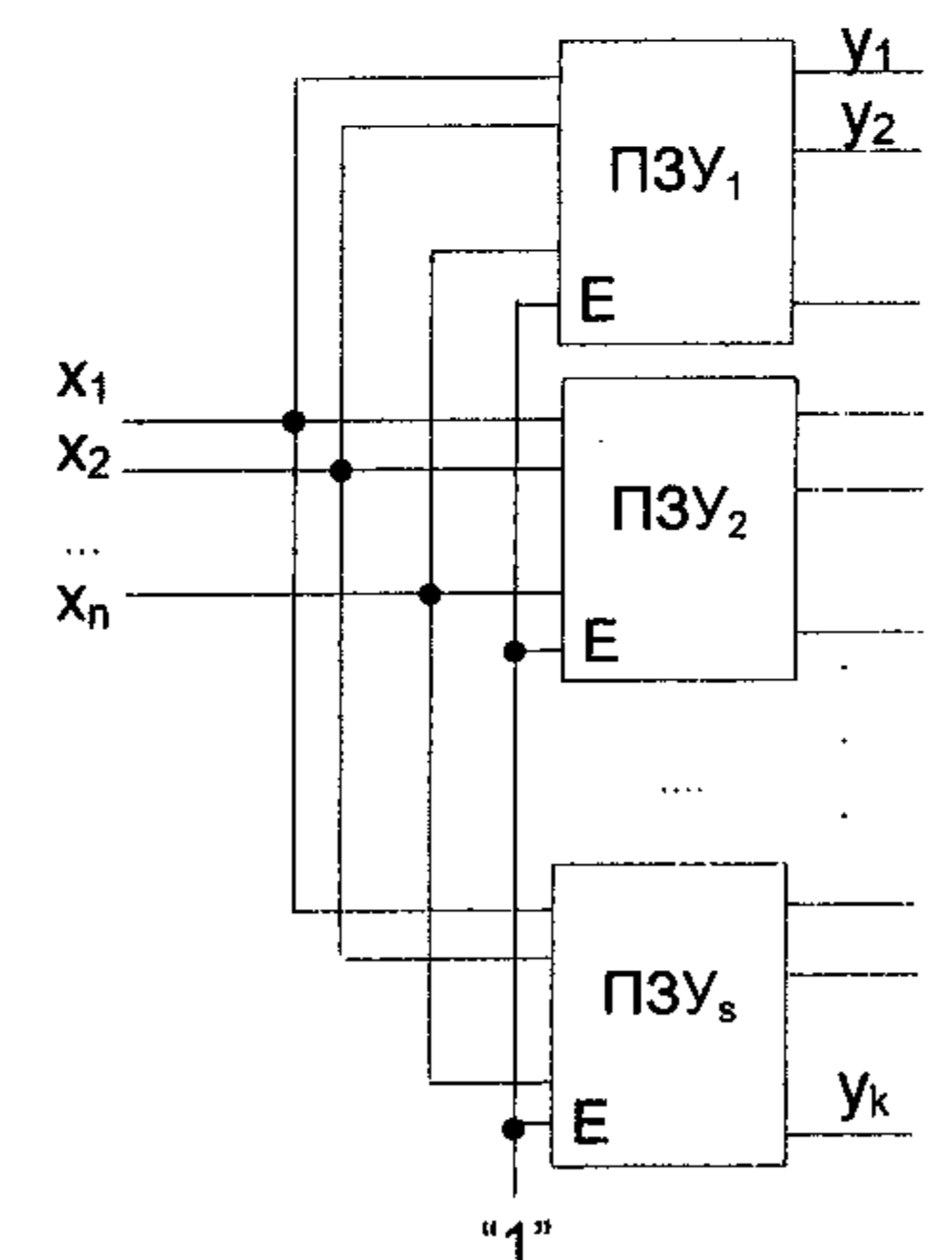
Двоичната комбинация на входа определя т.нар. адрес, дефиниращ активната изходна "адресна" шина, предоставяща достъп до елементите



Фиг.4.11 Разширяване възможностите на ПЗУ по отношение броя на входните променливи

от второто стъпало в структурата – конкретната "клетка" от паметта. Клетките памет образуват матрица от ИЛИ връзки (gates, гейтове). Изходите на ПЗУ са вертикалните шини y_j ($j=1, \dots, m$) в структурата, включващи и изходни усилвателни (драйверни) схеми. Конкретната функция във всеки изход зависи от записаната в матрицата на паметта информация – 0 или 1 в съответните клетки. Наличието на връзка, т.е. участието на i -тия минчлен в j -тата дизюнкция, се отбелязва с точка в матрицата на връзките. Показаната на фиг.4.10 структура може да реализира до m на брой функции на n входни променливи.

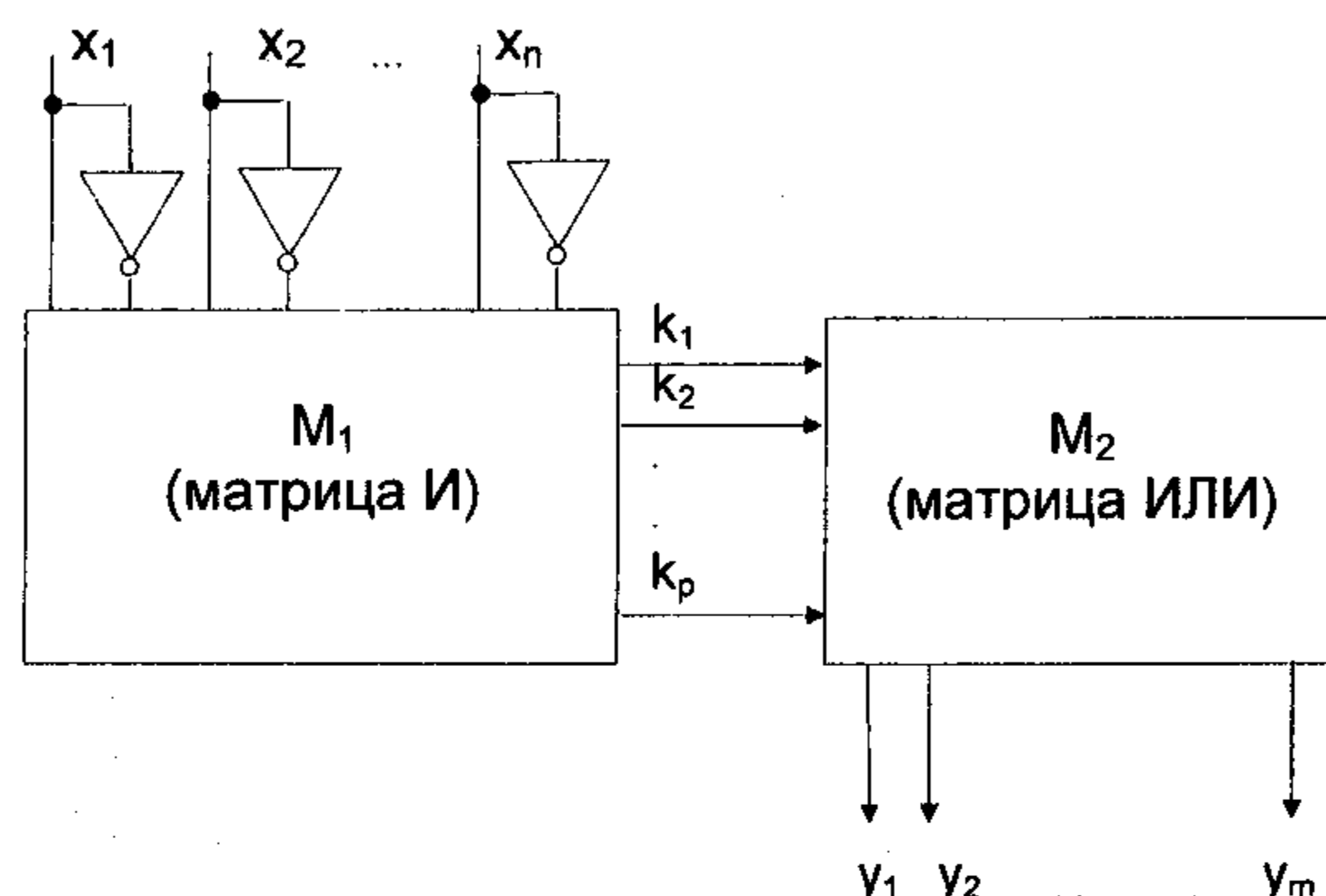
При положение, че броят на входовете на ПЗУ е недостатъчен (т.е. на входните променливи на функциите), могат да се използват няколко схеми, така както е показано на фиг.4.11. Използват се разрешаващите входове на включените



Фиг.4.12 Разширяване възможностите на ПЗУ по отношение броя на изходите

ПЗУ, управлявани от изходите на отделен дешифратор.

Ако броят на изходните шини (т.е. на реализираните функции) е недостатъчен, е необходимо да се добавят още схеми, свързани едновременно към входните сигнали – фиг.4.12.



Фиг.4.13 Обща структура на ПЛМ

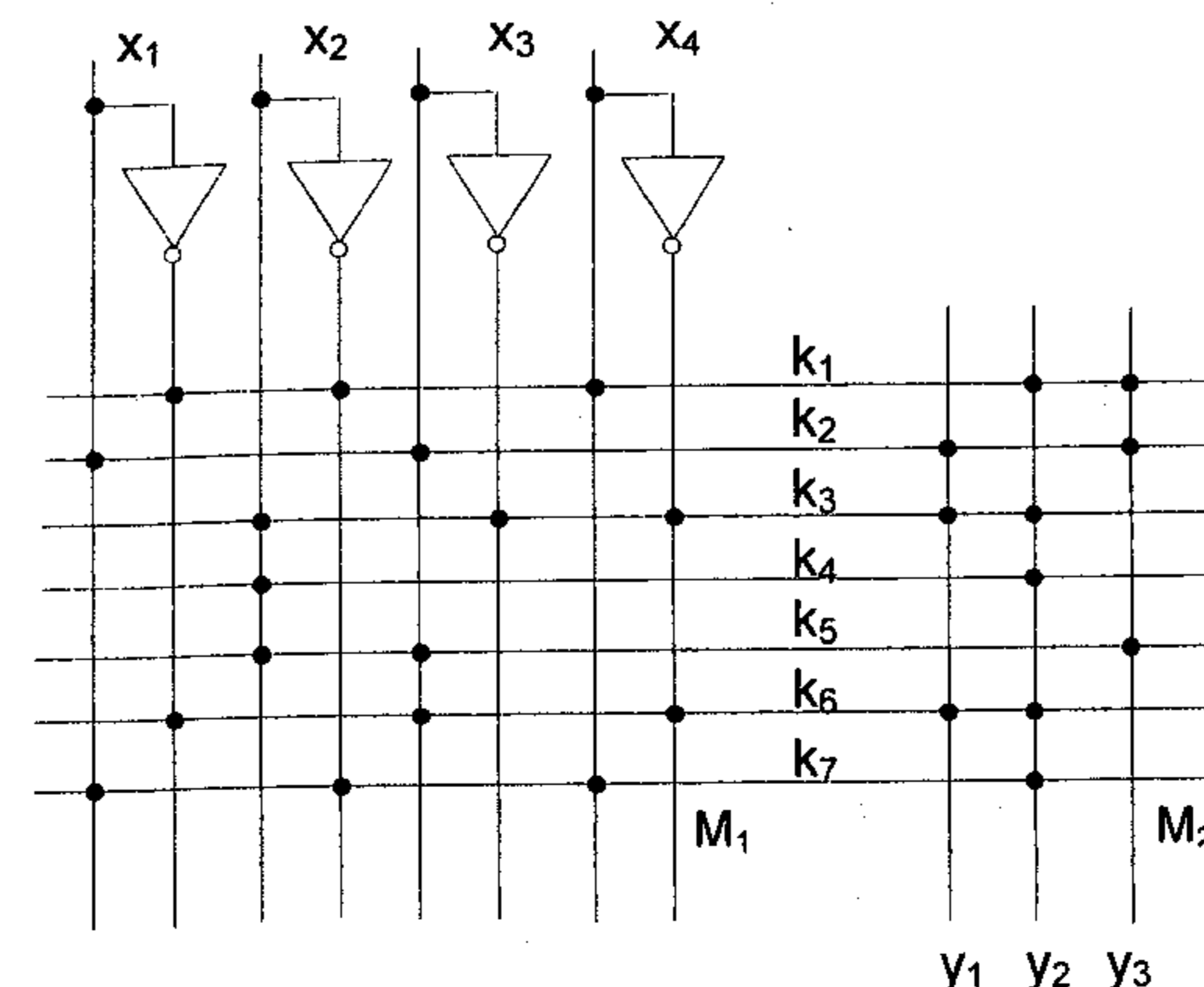
Програмируемите логически матрици (ПЛМ) са регулярни структури, състоящи се от две отделно обособени подматрици, условно обозначени тук като M_1 и M_2 – фиг.4.13. Матрицата M_1 е И-матрица, състояща се от $2n$ вертикални шини и p на брой хоризонтални шини. Всяка двойка вертикални шини съответства на правата и инверсна стойност на една от входните променливи x_i, \bar{x}_i , докато всяка хоризонтална шина – на конjunkция k_j от произволен набор входни променливи (в прав или инверсен вид).

Матрицата M_2 представлява ИЛИ-матрица, и има p хоризонтални шини (толкова, колкото и в матрицата M_1) и m на брой вертикални шини. На всеки изход от матрицата M_2 (и на ПЛМ) съответства дизюнкция y_k от определен брой конjunkции k_1 до k_p . Свързването на конкретните хоризонтална и вертикална шини във всяка от матриците се отбелязва с точка. По-долу е представен пример за реализация на логическите функции y_1 до y_3 с помощта на 4-входова ПЛМ.

$$y_1 = k_2 \vee k_3 \vee k_6 = x_1 x_3 \vee x_2 \bar{x}_3 \bar{x}_4 \vee \bar{x}_1 x_3 \bar{x}_4;$$

$$y_2 = k_1 \vee k_3 \vee k_4 \vee k_6 \vee k_7 = \bar{x}_1 \bar{x}_2 x_4 \vee x_2 \bar{x}_3 \bar{x}_4 \vee x_2 \vee \bar{x}_1 x_3 \bar{x}_4 \vee x_1 \bar{x}_2 x_4;$$

$$y_3 = k_1 \vee k_2 \vee k_5 = \bar{x}_1 \bar{x}_2 x_4 \vee x_1 x_3 \vee x_2 x_3.$$



Фиг.4.14 Реализация на примерни логически функции с ПЛМ: структура на връзките

ПЛМ се произвеждат като отделни схеми с голяма степен на интеграция. Задаването на конкретните връзки между шините (т.нар. персонализация или програмиране) се осъществява в процеса на производство на схемата или от потребителя чрез отделен процес (например облъчване с ултравиолетова светлина за изтриване на памети от типа EPROM и последващ процес на запис).

Освен реализацията на логически функции в ДНФ

чрез ПЛМ може да се имплементират и т.нар. “скобкови” форми на булеви функции с различна степен на вложеност. В тези случаи се използват част от изходните шини за реализация на подфункции, които участват като вложени в крайните функции, получавани в изходите на други шини на матрицата M_2 . В зависимост от конкретната задача могат да се използват няколко ПЛМ.

На фиг.4.15 е показан пример, при който крайните функции y_1 и y_2 реализират функциите “сума по модул две” и “логическа равнозначност”. Те от своя страна могат да се представят като релации между функциите ϕ_1 и ϕ_2 , реализиращи функцията сума по модул две от двойките променливи, съответно x_1, x_2 и x_3, x_4 , използвани като вложени:

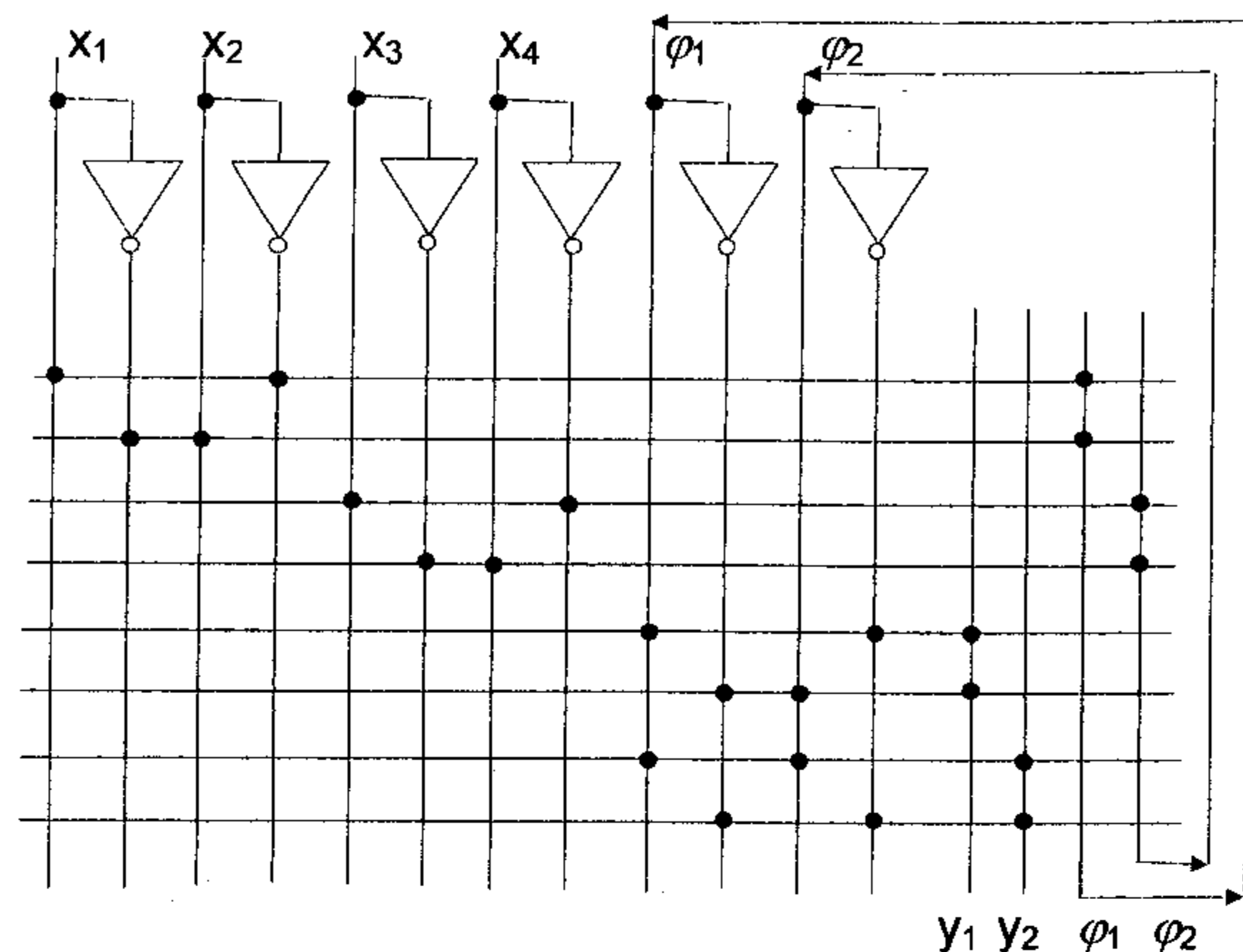
$$y_1 = (x_1 \oplus x_2) \oplus (x_3 \oplus x_4) = (x_1 \oplus x_2)(x_3 \oplus x_4) \vee \overline{(x_1 \oplus x_2)(x_3 \oplus x_4)} = \phi_1 \bar{\phi}_2 \vee \bar{\phi}_1 \phi_2;$$

$$y_2 = (x_1 \oplus x_2) \equiv (x_3 \oplus x_4) = (x_1 \oplus x_2)(x_3 \oplus x_4) \vee \overline{(x_1 \oplus x_2)(x_3 \oplus x_4)} = \phi_1 \phi_2 \vee \bar{\phi}_1 \bar{\phi}_2;$$

$$\phi_1 = x_1 \oplus x_2 = x_1 \bar{x}_2 \vee \bar{x}_1 x_2;$$

$$\phi_2 = x_3 \oplus x_4 = x_3 \bar{x}_4 \vee \bar{x}_3 x_4.$$

Ако разполагаме с ПЛМ с по-малък от необходимия брой входове, то можем да използваме няколко схеми. На фиг.4.16 е представен пример за реализация на функцията логическа равнозначност от две вложени функции само с помощта на двувходови ПЛМ. Изходната функция $Y = \phi_1 \phi_2 \vee \bar{\phi}_1 \bar{\phi}_2$ се получава, като входни променливи в съответната ПЛМ са функциите ϕ_1 и ϕ_2 .



Фиг.4.15 Реализация на ЛФ с ПЛМ с използване на вложени функции

Използването на вложеност при реализацията на ЛФ с ПЛМ може да се търси в по-общ план, като се разгледа въпросът в аспект на т.нар. функционална декомпозиция (ФД).

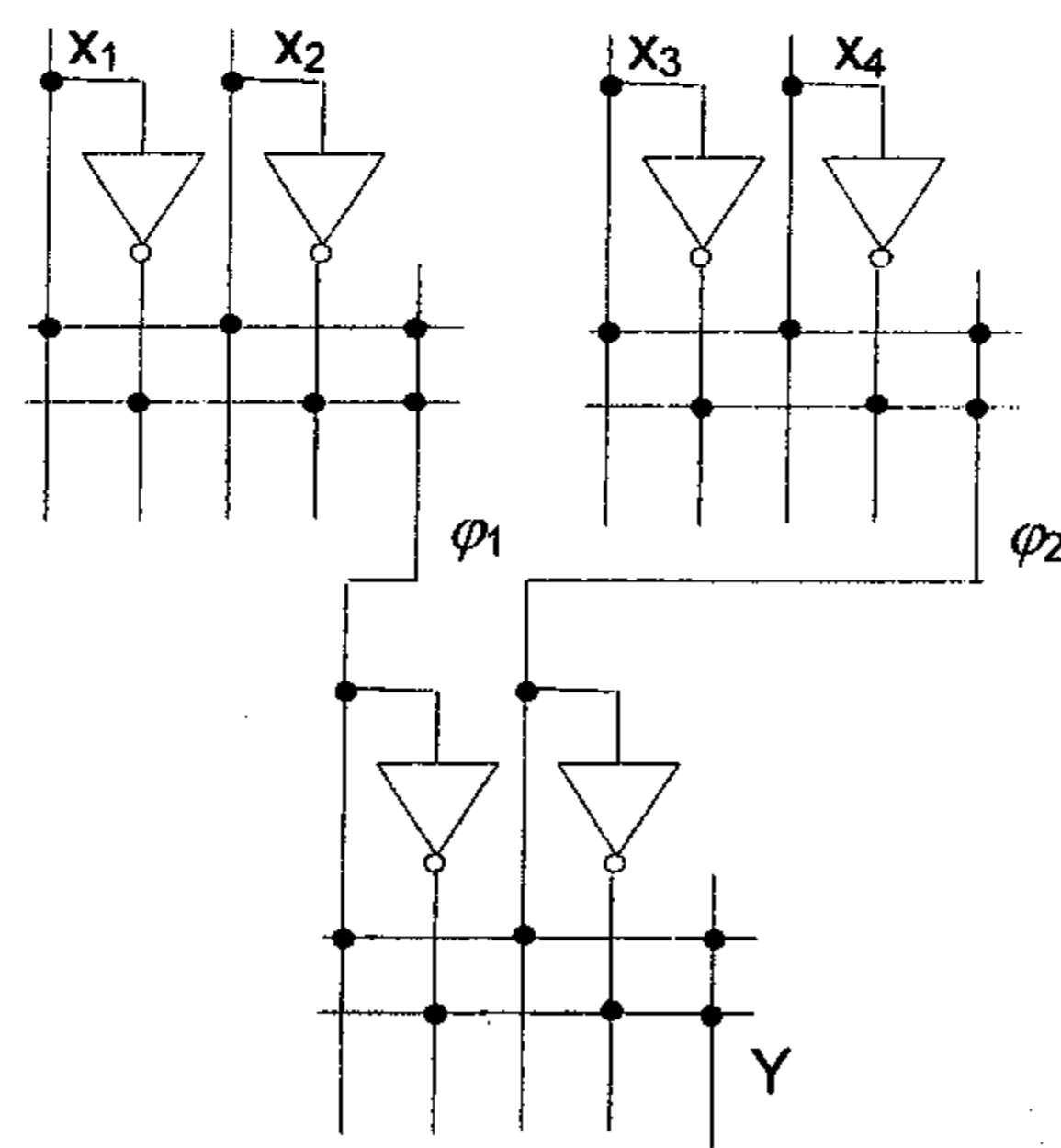
Това е начин на представяне на една логическа функция в следната форма:

$$f(x_1, x_2, \dots, x_n) = \varphi_m[\varphi_{m-1}(X^{m-1}), \varphi_{m-2}(X^{m-2}), \dots, \varphi_1(X^1), X^0],$$

където $X^i, i=0,1,\dots,m-1$ са подмножества от аргументите на функцията. Множеството X^0 се нарича *свободно* и не е включено като аргумент на функциите φ_k . Останалите аргументи X^B се наричат *свързани*: $X^B = X^1 \cup \dots \cup X^{m-1}$. В зависимост от съотношението на множествата на свободните и свързани аргументи различаваме два типа ФД:

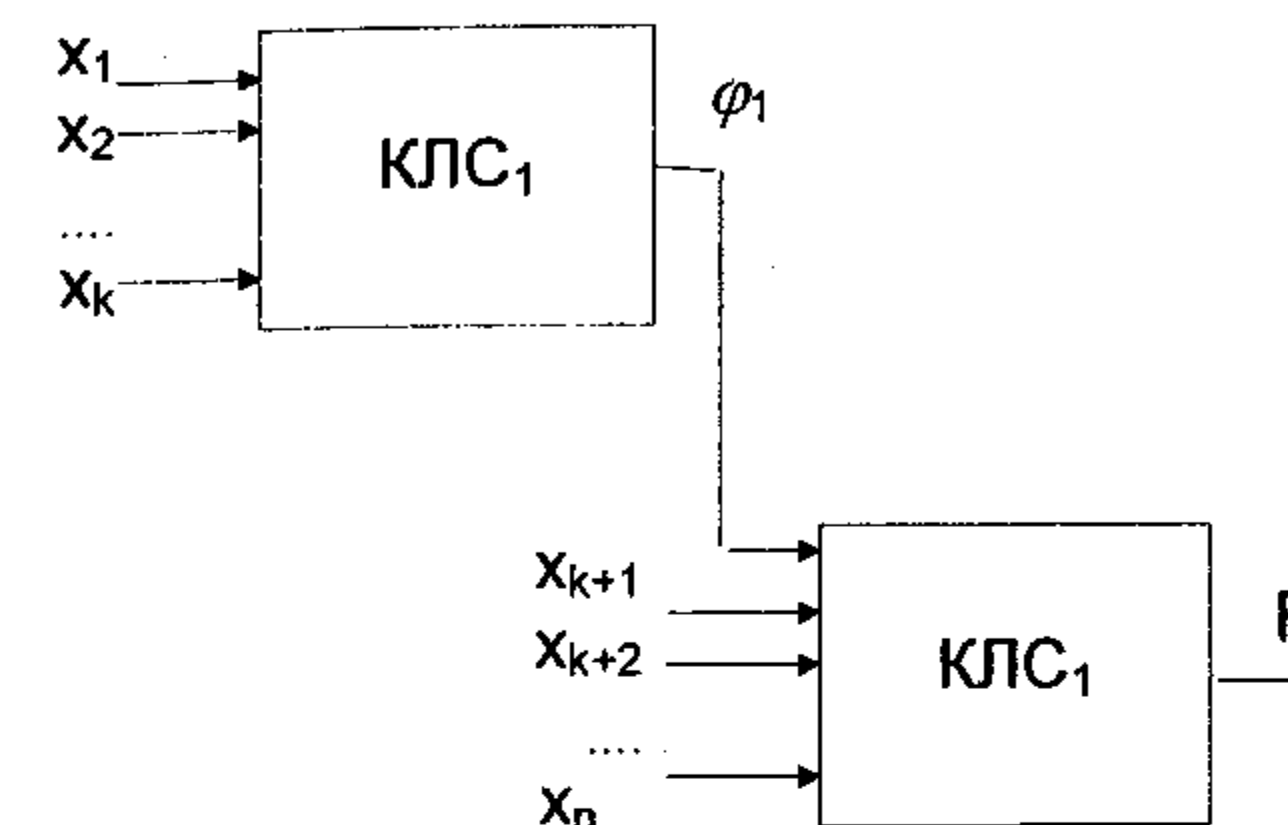
- неразделителна, при $X^0 \cap X^B \neq \emptyset$ – т.е. ако двете множества се пресичат;
- разделителна, при $X^0 \cap X^B = \emptyset$ – т.е. ако двете множества са непресичащи се.

Особен интерес представлява възможността функцията да се представи във вида:



Фиг.4.16 Получаване на функцията логическа равнозначност чрез двувходови ПЛМ

$f(x_1, \dots, x_n) = \varphi_2[\varphi_1(X^1), X^0]$ Такова представяне съответства на еднократна ФД, която позволява удобна схемна реализация с произволна комбинационна логическа схема (КЛС), в частност – ПЛМ. Фиг.4.17 представя обобщената блокова структура на такава реализация. Накратко: ако общото множество от аргументи на една функция можем



Фиг.4.17 Обща блокова структура на реализация на логическа функция чрез КЛС при еднократна ФД

да разделим така, че част от аргументите да не са определящи за вложената функция φ_1 , то тя ще се определя само от останалите и ще позволява реализация с отделна КЛС. За структурата от фиг.4.17 аргументите x_{k+1} до x_n са свободни и не определят φ_1 . Изходната функция F съответства на φ_2 от записа по-горе.

Трябва да се отбележи, че не съществува формален критерий за разделяне на аргументите на една функция на свободни и свързани. В

практическите задачи от такъв тип изборът на свободни аргументи се извършва емпирично.

Нека разгледаме пример за реализация на функцията f на пет аргумента, зададена чрез карта на Вейч с помощта на 3-входови ПЛМ. Тук за свободни се избират аргументите x_2 и x_3 . Картата се разделя на отделни подкарти в зависимост от комбинацията (наборите) от свободни аргументи. Забелязва се, че покритията на двете десни карти съответстват на

		x_1				x_2				f
		x_2		x_3		x_2		x_3		
x_3	x_4			1					1	
	x_5				1				1	
	x_4		1	1	1	1		1	1	
	x_5	1		1	1			1	1	

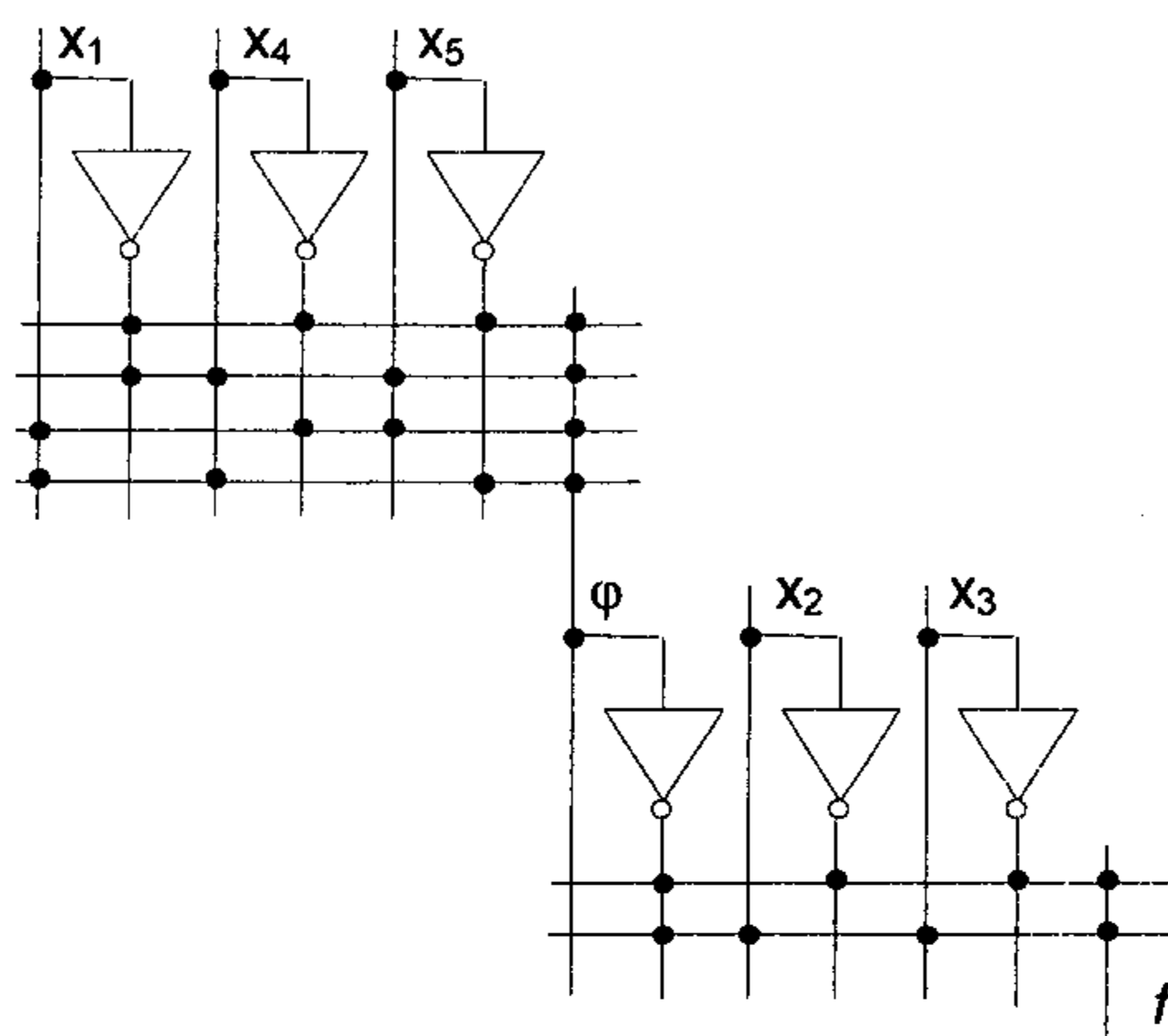
		$x_2 x_3$				$\bar{x}_2 \bar{x}_3$				φ
		x_1		x_4		x_1		x_4		
x_4	x_5					1	1	1	1	
	x_5					1	1	1	1	
x_4	x_5					1		1		
	x_5						1		1	

взаимно инверсни функции – т.е. вложената функция φ се избира една от тях. За функцията φ и крайната функция f имаме:

$$\varphi = \bar{x}_1 \bar{x}_4 \bar{x}_5 \vee \bar{x}_1 x_4 x_5 \vee x_1 \bar{x}_4 x_5 \vee x_1 x_4 \bar{x}_5;$$

$$f = 0.x_2 x_3 \vee 1.\bar{x}_2 \bar{x}_3 \vee \varphi.\bar{x}_2 x_3 \vee \bar{\varphi}.x_2 \bar{x}_3 = \bar{x}_2 \varphi \vee \bar{x}_3 \bar{\varphi}.$$

След заместването на вложената функция аргументи на функцията f вече са φ , x_2 и x_3 . Схемната реализация с две 3-входови ПЛМ ще има вида, показан на фиг.4.18.



Фиг.4.18 Реализация на функцията f чрез ФД и две тривходови ПЛМ

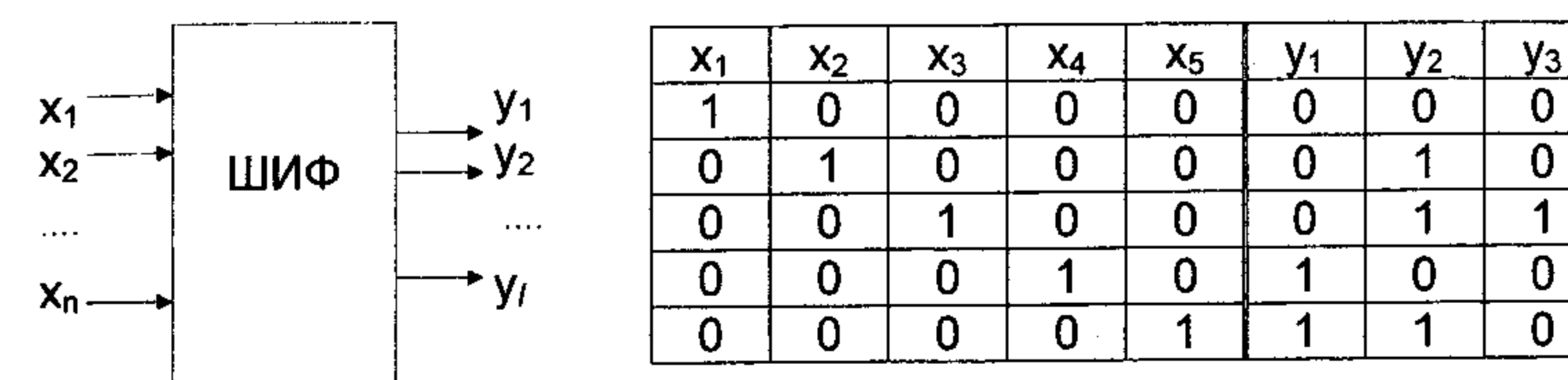
4.4. Шифратори

Шифраторите (*encoders*), са комбинационни схеми, извеждащи в изходите си двоичен код, съответстващ само на един активен вход от схемата. Тяхното действие може да се разглежда като обратно на това на дешифраторите.

Необходимо е броят на изходните двоични комбинации да „покрива“ всички входове на шифратора, т.е. ако имаме n на брой входове, изходите y_i следва да са на брой $l=2^k > n$, където k е цяло число.

Класическо приложение на шифраторите е при клавиатурите за персонални компютри и различни други цифрови устройства, при които всеки символ се кодира с определена двоична комбинация след натискане на съответен клавиш. Така например, ако общият брой на символите (букви, цифри и специални символи) в някои от стандартните типове клавиатури е 88 или 101, то шифраторът следва да има поне 7 изхода ($2^7=128$).

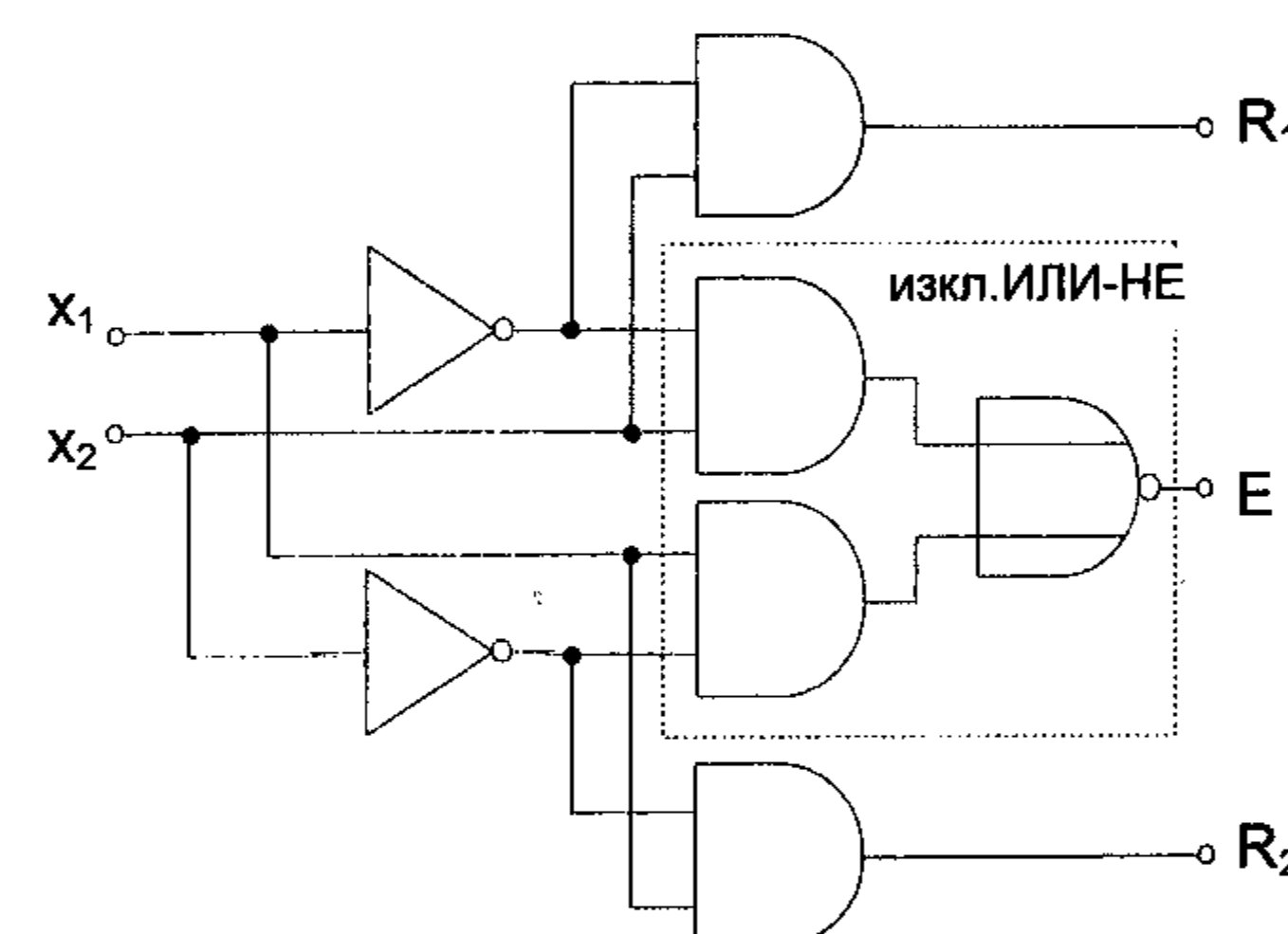
На фиг.4.19 е показан общият вид на шифратор и примерна таблица на истинност в случай, че шифраторът има 5 входа и 3 изхода.



Фиг.4.19 Шифратори - обща структура и примерна таблица на истинност

Реализацията на функцията за равенство (изход Е, equality) се получава чрез логическа схема изключващо ИЛИ-НЕ, а изходните функции R_1 ($x_1 < x_2$) и R_2 ($x_1 > x_2$) – с логически елементи от типа И.

Схема на n -битов цифров компаратор може да се изгради, като се използват едноразредни компаратори за всеки отделен разряд и допълнителна логика.



x_1	x_2	резултат	изход
0	0	$x_1 = x_2$	E
0	1	$x_1 < x_2$	R_1
1	0	$x_1 > x_2$	R_2
1	1	$x_1 = x_2$	E

$$E = \bar{x}_1 x_2 \vee x_1 \bar{x}_2 = x_1 \oplus x_2$$

$$R_1 = \bar{x}_1 x_2$$

$$R_2 = x_1 \bar{x}_2$$

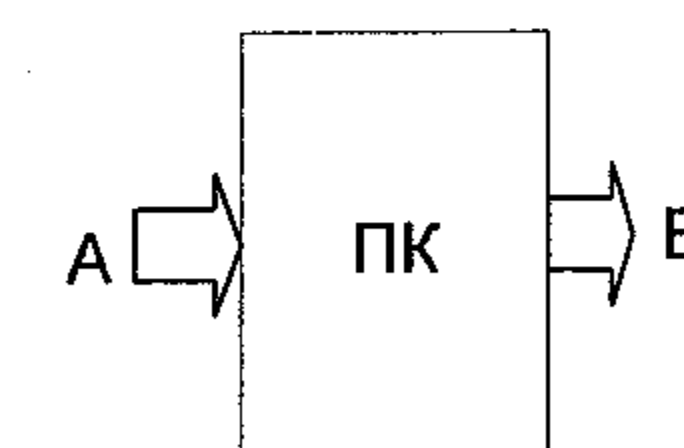
фиг.4.20 Едноразряден цифров компаратор: принципна схема, функции на изходите

4.5. Цифрови компаратори

Цифровият компаратор е комбинационна схема, служеща за определяне на съотношението между две бинарни числа x_1 и x_2 . В изходите на схемата се определя дали $x_1 > x_2$, $x_1 < x_2$, или $x_1 = x_2$. Логическата структура на едноразряден цифров компаратор, както и съответната таблица на истинност на изходните функции, е показана на фиг.4.20.

4.6. Кодови преобразуватели

Кодовите преобразуватели (преобразуватели на код, ПК) са схеми с n на брой входове и m на брой изходи, които преобразуват подадената на своите входове двоична информация от един код (напр.А) в друг код (напр.Б) в изходите си – фиг.4.21. Схемите на дешифратори, които бяха разгледани по-рано в тази глава, могат да се разглеждат като частен случай по отношение на



Фиг.4.21 Кодов преобразувател: обща структура

входовете, при което кодът А е двоичен (бинарен), и като частен случай по отношение на изходите, при което броят им m е равен на 2^n . Популярни в практиката схеми са тези на преобразуватели на двоичен в различни изходни кодове, напр. код на Грей, термометричен, кодове с излишък, седемсегментен и др.

По-долу е представен пример за синтез на преобразувател от код 8-4-2-1 в код с излишък 3. Входният код 8-4-2-1 съответства на десетичните цифри от 0 до 9, а изходният код – на двоичните набори на 4 променливи без първите и последните три набора (т.е. с излишък 3 по отношение на изходния бинарен код). На фиг.4.22 е показано определянето на функциите на изходите на схемата чрез карти на Вейч.

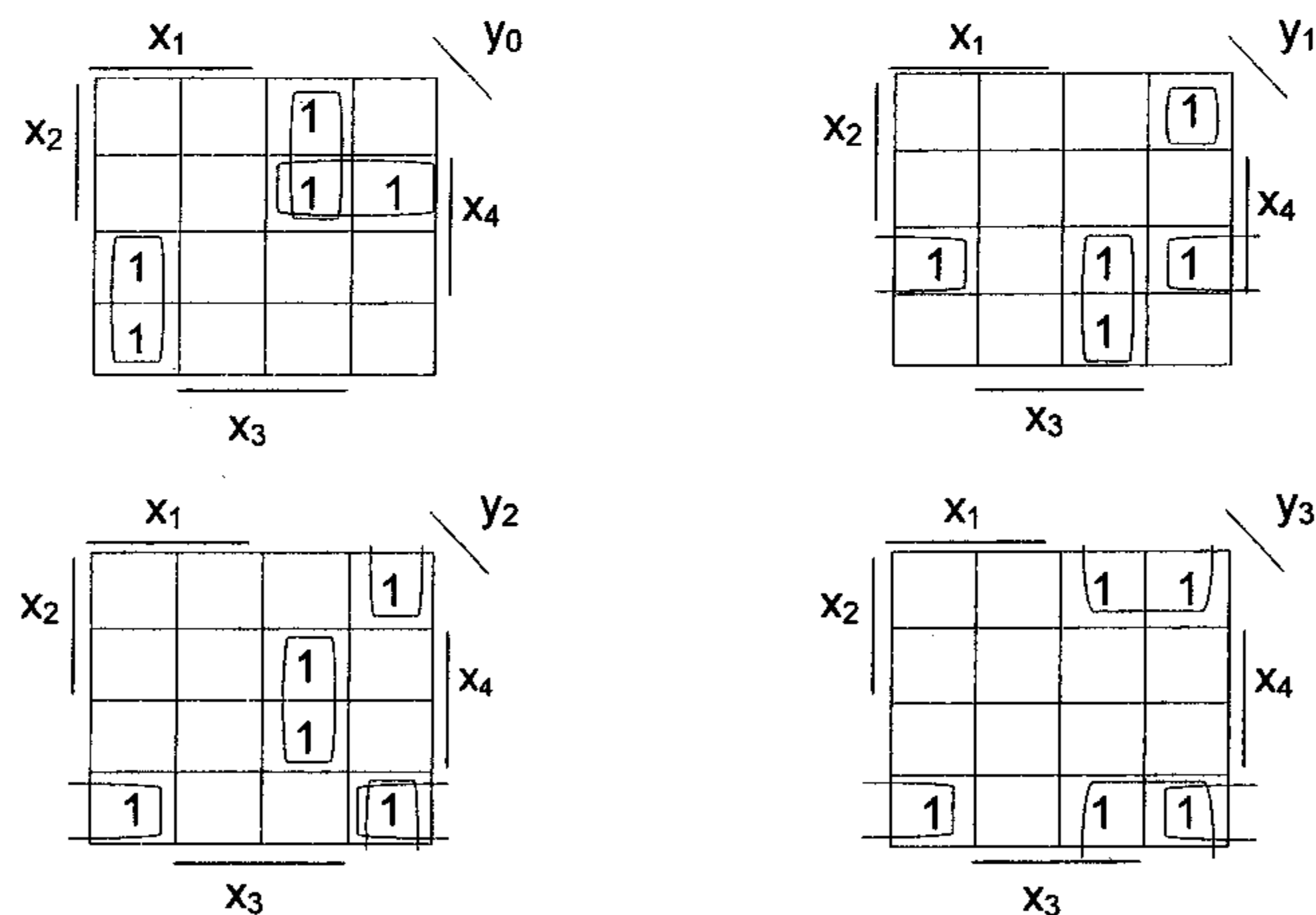
#	x_1	x_2	x_3	x_4	y_0	y_1	y_2	y_3
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

$$y_0 = x_1 \bar{x}_2 \bar{x}_3 \vee \bar{x}_1 x_2 x_3 \vee \bar{x}_1 x_2 x_4;$$

$$y_1 = \bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 \vee \bar{x}_1 \bar{x}_2 x_3 \vee \bar{x}_2 \bar{x}_3 x_4;$$

$$y_2 = \bar{x}_1 x_3 x_4 \vee \bar{x}_2 \bar{x}_3 \bar{x}_4 \vee \bar{x}_1 \bar{x}_3 \bar{x}_4;$$

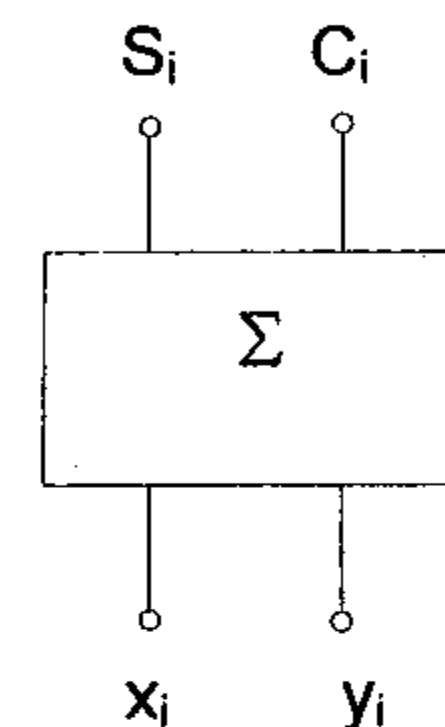
$$y_3 = \bar{x}_1 \bar{x}_4 \vee \bar{x}_2 \bar{x}_3 \bar{x}_4$$



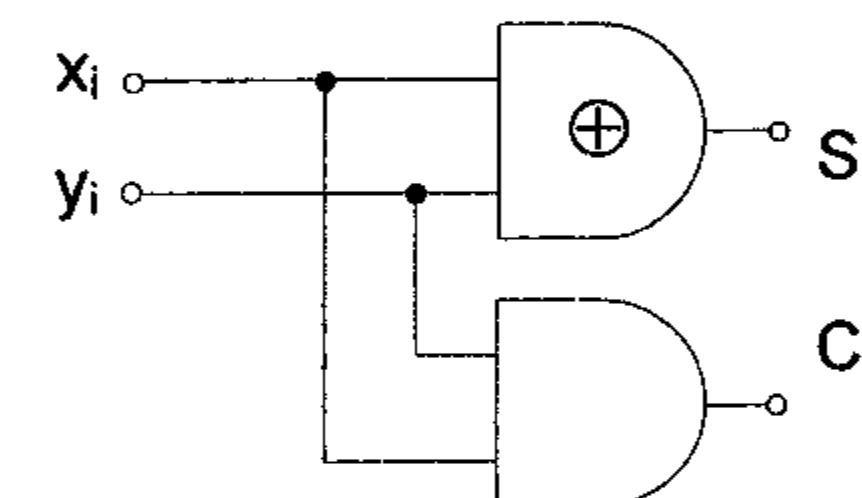
Фиг.4.22 Определяне аналитичните изрази на функциите на изходите на преобразувател на код 8-4-2-1 в код с излишък 3

4.7. Суматори

При сумирането на десетични числа, както е добре известно, се сумират последователно цифрите на единиците, десетиците и т.н., като при това след сумирането на отделните цифри към тях се добавя и числото на излишък над десет от сумирането на предходните по тегло цифри, т.нар. *пренос*. Аналогично при двоичното сумиране се събират цифрите (битове) с едно и също тегло (напр. 2^k), като към него се добавя



x_i	y_i	S_i	C_i
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

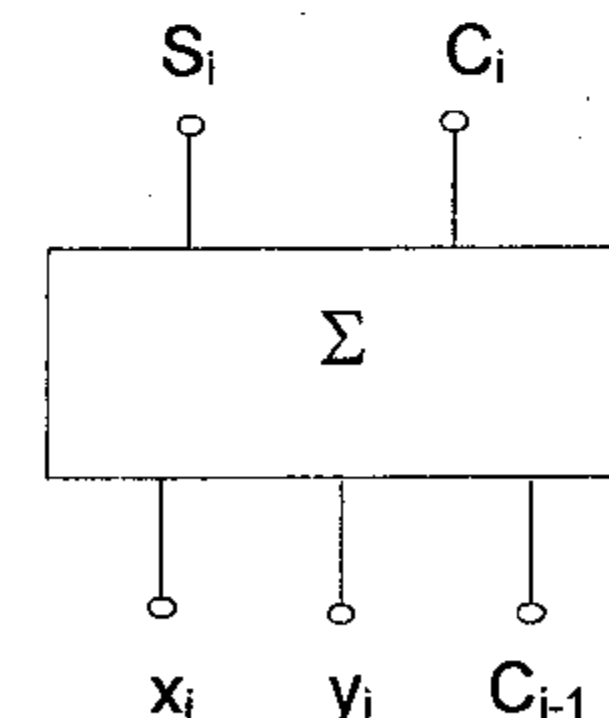


$$S_i = \bar{x}_i y_i \vee x_i \bar{y}_i = x_i \oplus y_i$$

$$C_i = x_i y_i$$

фиг.4.23 Едноразряден полусуматор: блоков вид, логическа схема, уравнения на изходите на полусумата и частичния пренос

и битът на преноса (carry bit) от сумата на битовете с тегло 2^{k-1} , ако той съществува (т.е., ако има стойност единица). Тази операция може да се представи в две стъпки – първо, сумиране на битовете с едно и също двоично тегло (едноименни битове) и второ, към получения резултат се добавя битът на пренос от сумирането на битовете с единица по-малко тегло.

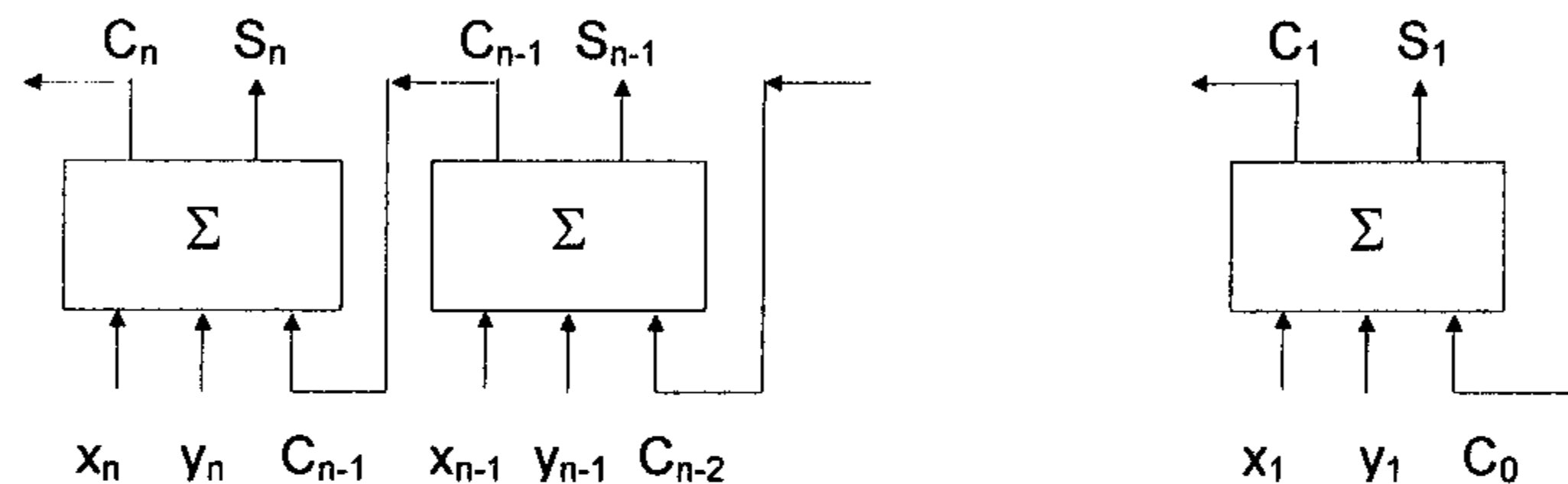


фиг.4.24 Едноразряден пълен суматор

Цифровото устройство с два входа, с което се извършва двоично сумиране на едноименни битове се нарича *полусуматор*. Пълната операция сумиране изисква т.нар. *пълен суматор*. На фиг.4.23 са дадени блоковият вид, таблицата на истинност на изходите (сума и пренос) и логическата схема на едноразряден двоичен полусуматор. Входове на схемата са двата бита с едно и също тегло x_i и y_i , а

изходи – функцията на частичната сума S_i и частичният пренос C_i . Функцията на изхода S_i се изразява чрез сума по модул две (изключващо ИЛИ). Фиг.4.24 представя блоковата структура на пълен едноразряден суматор. За разлика от схемата на полусуматор тук входна променлива е също така и битът на пренос от предходния разряд (с тегло $i-1$).

Многоразрядно сумиране се извършва със схема, чийто обобщен вид е представен на фиг.4.25 (за случай на две n -битови числа). Свързването на отделните побитови пълни суматори става в съответствие с представените по-горе правила за извършване на



фиг.4.25 Многоразряден суматор – принцип на изграждане

операцията сумиране на многобитови числа. Ако липсва предхождаща структура, чийто пренос да се отчита, входната променлива C_0 е излишна. При положение, че следваща структура за отчитане на преноса от най-старшия разряд не съществува в общото устройство, то наличието на пренос в C_n се отчита като препълване.

Задание

1. Реализирайте зададената логическа функция, като използвате пълен n -входов дешифратор и логическа схема:
 - разполагате с ДШ $n \rightarrow 2^n$ с прави изходи и n -входова схема ИЛИ;
 - разполагате с ДШ $n \rightarrow 2^n$ с инверсни изходи и n -входова схема И-НЕ.
 Снемете таблицата на истинност на реализираната схема.
2. Реализирайте логическа функция на n променливи чрез мултиплексор с размерност $(n-1) \rightarrow 1$. Снемете таблицата на истинност на схемата. Използвайте мултиплексори с разрядност $2 \rightarrow 1$, $3 \rightarrow 1$ или $4 \rightarrow 1$.
3. Постройте схема на мултиплексор чрез каскадна реализация от схеми с по-ниска разрядност:

- схема на мултиплексор $4 \rightarrow 1$, като разполагате със схеми с разрядност $2 \rightarrow 1$;
- схема на мултиплексор $8 \rightarrow 1$, като имате на разположение схеми с разрядност $4 \rightarrow 1$ и $2 \rightarrow 1$.

Снемете таблиците на изградените схеми.

4. Реализирайте чрез 4-входова ПЛМ посочените логически функции, зададени в ДНФ.
5. Постройте схема с три дву-входови ПЛМ, реализираща логическата функция сума по модул две от две вложени функции, осъществяващи същата логическа функция на две двойки променливи (вж.фиг.4.16).
6. Определете функциите на изходите на преобразувател от двоичен входен код във:
 - изходен с излишък 6;
 - код на Грей;
 - термометричен код;
 - седемсегментен код (управляващ панел за изобразяване на цифрите от 0 до 9).

Контролни въпроси

1. Каква комбинационна функция изпълнява дешифраторът? Каква е разликата между пълен и непълен ДШ?
2. За какъв тип дешифратор се отнасят дадените на фиг.4.1 таблица на истинност и уравнения на изходите (с активно ниско или с активно високо изходно ниво)?
3. Попълнете таблицата на истинност на пълен ДШ $3 \rightarrow 8$ с инверсни изходи. Запишете уравненията на изходите му.
4. Какво е изискването по отношение на дешифратора и логическата схема, с които разполагате, така че да може да се реализира произволна логическа функция, представена в СДНФ?
5. Защо схемите на ПЗУ са комбинационни? Каква е общата структура на този клас схеми?
6. Какво е предназначението на схемите на ПЛМ? От какви подматрици се състоят?
7. Каква функция изпълнява шифраторът? Какво е основното му приложение?

8. За какво служат преобразувателите на код? Като частен случай на коя схема могат да се разглеждат те по отношение на входовете и изходите си?
9. Какво означават понятията полусума и частичен пренос?
10. Как се реализира многоразряден суматор?

Т Е М А 5

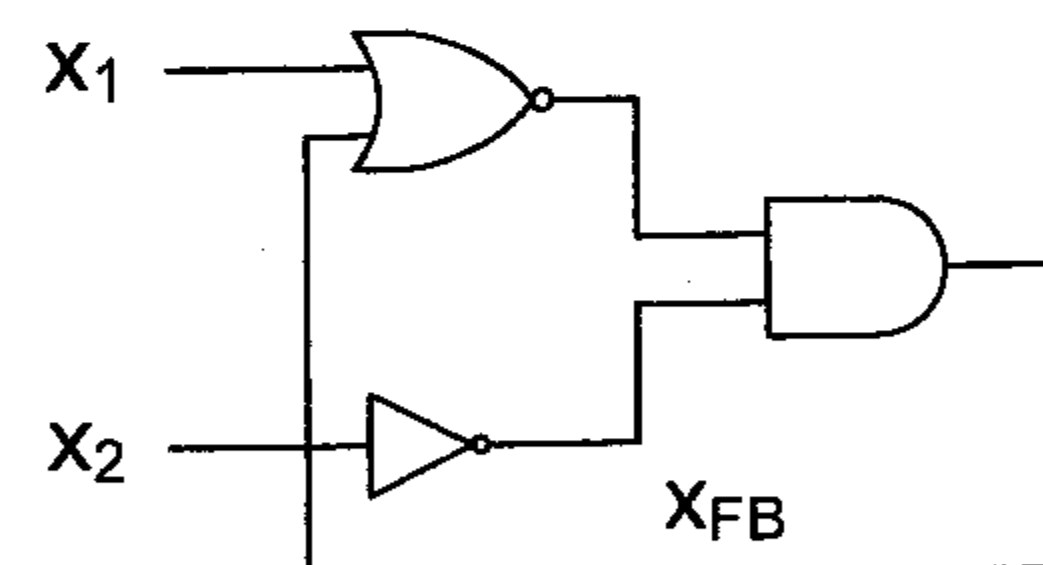
ПОСЛЕДОВАТЕЛНОСТНИ СХЕМИ. СТРУКТУРНИ МОДЕЛИ. ЕЛЕМЕНТИ ПАМЕТ. СИНТЕЗ И АНАЛИЗ НА КРАЙНИ АВТОМАТИ С ПАМЕТ

5.1. Последователностни схеми: структура, особености, автоматни модели

При разглежданите дотук схеми стойностите на изходните функции в даден момент t се определят само от входните комбинации, подадени в този момент, и не зависят от това какви са били входните въздействия в предшестващите моменти $t-1, t-2, \dots$

Интерес представляват и схемите, в които се получават *последователности* от изходни реакции, съответстващи на определени *последователности* от входни въздействия. При тях изходната реакция зависи не само от входното въздействие върху схемата в дадения момент, но и от предходните входни набори, приложени към схемата, т.е. изходните реакции зависят както от непосредствените в разглеждания момент входни въздействия, така и от нейната предистория. Схемите, които отговарят на това описание, се наричат последователностни схеми. Зависимостта на изходния сигнал от състоянието на схемата в предходния момент, се постига с въвеждане на обратна връзка в структурата на логическата схема.

На фиг. 5.1. е показана примерна логическа схема с обратна връзка и реакцията в изхода ѝ при различна стойност на сигнала x_{FB} , връщан по веригата на обратната връзка. Очевидно, стойността на изходния сигнал Y при стойности 0 на входните променливи x_1 и x_2 е различна в зависимост от тази на сигнала x_{FB} , т.е. поведението на схеми от подобен вид не може да бъде описвано с обикновени таблици на истинност.



x_1	x_2	x_{FB}	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Фиг. 5.1 Примерна логическа схема с обратна връзка и таблицата ѝ на истинност в зависимост от стойността на сигнала във веригата на ОБ

Действието на комбинационните схеми се описва с множеството на входните променливи $X=\{x_1, x_2, \dots, x_n\}$ и множеството изходните променливи $Z=\{z_1, z_2, \dots, z_m\}$. Тези две множества не са достатъчни за описание действието на последователностните схеми. Добавя се трето множество – множеството на вътрешните променливи $A=\{a_1, a_2, \dots, a_n\}$. Роля на вътрешни променливи играят обратните връзки в схемата. Всяка комбинация от стойности на вътрешните променливи задава определено *вътрешно състояние* на схемата. Всяка комбинация от стойностите на входните променливи се нарича *входна дума* или *входно въздействие*, а всяка комбинация от стойностите на изходните променливи се нарича *изходна дума* или *изходна реакция*.

Поведението на последователностните схеми се описва с две функции: функция на преходите φ и функция на изходите ψ . Функцията на преходите задава новото състояние, в което преминава схемата (автоматът), ако в момента t тя е била в състояние A и е получила като входно въздействие входната дума X .

$$A^{t+1} = \varphi(X^t, A^t)$$

Функцията на изходите задава зависимостта на изходната дума от входната дума и вътрешното състояние.

$$Z^t = \psi(X^t, A^t)$$

Математически модел на последователностната схема е абстрактният автомат, който се задава с множество от 6 елемента: $S=\{X, A, A_n, Z, \varphi, \psi\}$, където X е множеството на входните думи; A е множеството на вътрешните състояния на автомата; $A_n \subseteq A$ е началното състояние; Z е множеството на изходните думи; φ е функция на преходите; ψ е функция на изходите. Ще се ограничим в разглеждането на *крайните* автомати с памет, т. е. автомати с краен брой елементи в множествата на входните, изходните и вътрешните променливи.

Поведението на всяка последователностна схема се описва с един от двата автоматни модела – автоматен модел на Мили и автоматен модел на Мур.

Действието на схема, която работи по автоматния модел на Мили се описва по следния начин: Нека автоматът се намира в състояние $A_i \subseteq A$. Когато на неговия вход постъпи входната дума $X_m \subseteq X$, той изработва изходна реакция $Z_k \subseteq Z$ и в следващия момент преминава в състояние $A_j \subseteq A$. Новото състояние A_j и изходната реакция Z_k еднозначно се определят от входната дума и текущото състояние на автомата. Възможно е автоматът да осъществи преход между същите две състояния A_i и A_j , но под действие на друга входна дума и с издаване на друга изходна реакция.

$$\begin{aligned} A^{t+1} &= \varphi(X^t, A^t) \\ Z^t &= \psi(X^t, A^t) \end{aligned}$$

При автомата на Мили изходната реакция се свързва с прехода между състоянията, докато при автомата на Мур изходната реакция се свързва със самото състояние.

Действието схема, която работи по автоматния модел на Мур се описва по следния начин: Нека автоматът се намира в състояние $A_i \subseteq A$. Когато на неговия вход постъпи входната дума $X_m \subseteq X$, той преминава в състояние $A_j \subseteq A$ и тогава издава изходна дума $Z_k \subseteq Z$. Независимо по какъв път автоматът е достигнал състояние A_j , изходната реакция, която ще издаде, винаги ще бъде Z_k . Това означава, че при автомат на Мур функцията на изходите зависи само от вътрешното състояние.

$$\begin{aligned} A^{t+1} &= \varphi(X^t, A^t) \\ Z^t &= \psi(A^t) \end{aligned}$$

За всеки автомат на Мили S съществува еквивалентен автомат на Мур S' , в смисъл че на всяка произволно зададена входна последователност и съответно начално състояние двата автомата реагират с еднакви изходни последователности. Съществуват алгоритми за преход между двата типа автомати, но те няма да бъдат разглеждани.

5.2. Задаване на последователностни схеми

5.2.1 Таблично задаване

Табличното задаване включва две таблици – таблици на преходите и на изходите, задаващи съответно функцията на преходите и функцията на изходите на автомата. Всеки ред на тези таблици съответства на едно от входните въздействия на автомата, всяка колона – на едно от вътрешните му състояния. В клетката, образувана от пресичането на стълба A_i и реда X_j в таблицата на преходите (ТП), се записва състоянието, в което преминава автоматът от състояние A_i под действие на входен сигнал X_j – табл. 5.1, а в таблицата на изходите (ТИ) – съответстващият на този преход изходен сигнал – табл. 5.2.

Преходите и изходите на автомата могат еднозначно да бъдат определени и от общата таблица на преходите и изходите (ТПИ), съвместяваща информацията на двете описани таблици (във всяка нейна клетка се записват данните, които се съдържат в съответните клетки на таблицата на преходите и на таблицата на изходите) – табл. 5.3

При таблично задаване на автомата на Мур таблицата се редуцира в един ред. Затова обикновено се строи т.нар. белязана таблица на преходите. Тя съдържа един допълнителен ред, в който се записват изходните сигнали, съответстващи на вътрешните състояния на автомата – табл. 5.4.

Таблиците на преходите и изходите определят напълно последователността схема, тъй като заедно с функциите на преходите и изходите в тях се съдържат и множествата външни (входни и изходни) и вътрешни състояния на автомата, както и началното му състояние. Ако не е указано друго, за начално се приема състоянието A_0 .

$X \backslash A$	A_0	...	A_i	...	A_n
X_0					
\vdots					
X_j			A_k		
\vdots					
X_m					

Табл. 5.1

$X \backslash A$	A_0	...	A_i	...	A_n
X_0					
\vdots					
X_j			Z_k		
\vdots					
X_m					

Табл. 5.2

$X \backslash A$	A_0	...	A_i	...	A_n
X_0					
\vdots					
X_j			A_k / Z_k		
\vdots					
X_m					

Табл. 5.3

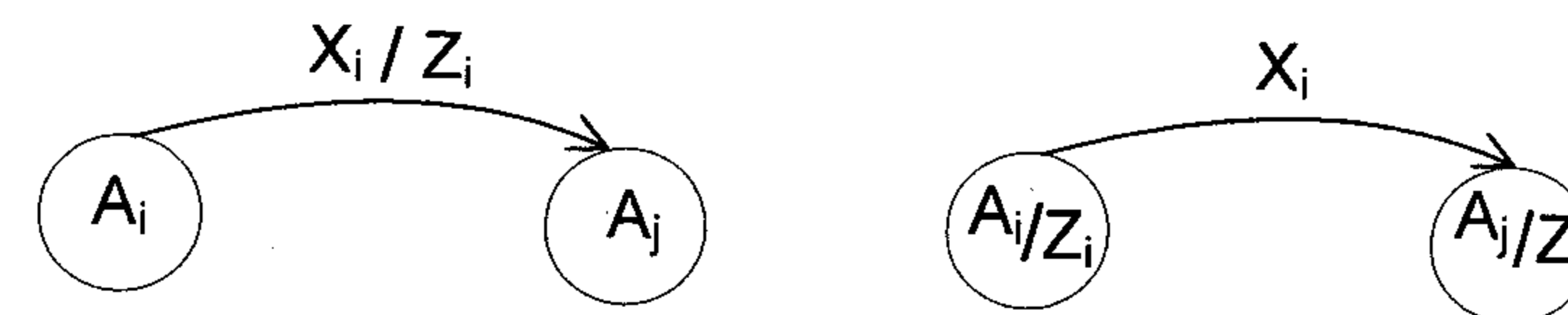
$X \backslash A$	A_0	...	A_i	...	A_n
X_0					
\vdots					
X_j			A_k		
\vdots					
X_m					
Z	Z_0	...	Z_i	...	Z_n

Табл. 5.4

5.2.2 Графично задаване

Последователностните схеми могат нагледно да се представят с помощта на автоматни графи на преходите. Графът на преходите се състои от възли и съединяващи ги ориентирани дъги. Всеки възел еднозначно съответства на определено вътрешно състояние на автомата. В кръгчето, съответстващо на възела, се записва символът на съответното вътрешно състояние. Ориентираната дъга сочи прехода на автомата от едно вътрешно състояние в друго. До дъгата се записват входните въздействия, предизвикващи този преход. При автомат на Мили до входните въздействия се записват и изходните реакции, които автоматът издава при съответния преход. При автомат на Мур тези

реакции се записват във възлите на графа. На фиг. 5.2 са показани примерни части от графи на автомат на Мили и Мур.



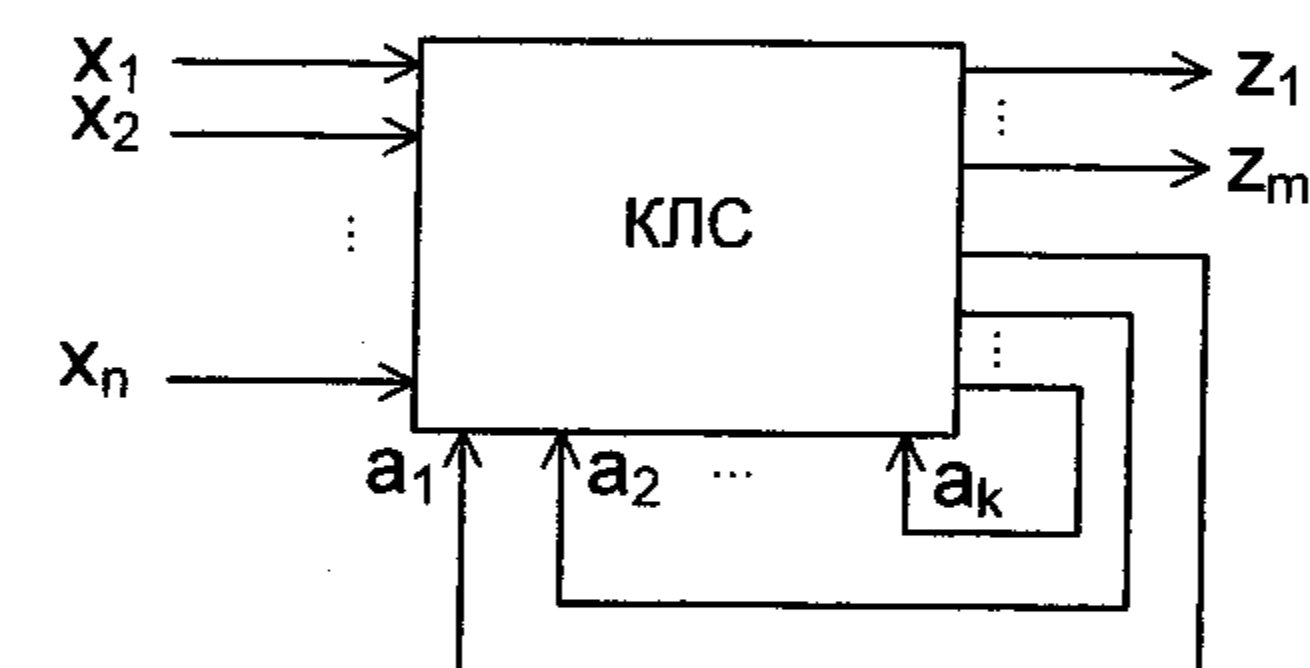
Фиг. 5.2 Части от графи на автомати на Мили и Мур

Разгледаните начини за описание на последователностна схема са еквивалентни един на друг, като всеки от тях е удобен за различни конкретни случаи.

5.3 Структурни модели на ПС

5.3.1. Структурен модел без елементи памет

Структурният модел без елементи памет има вида, показан на фиг. 5.3.

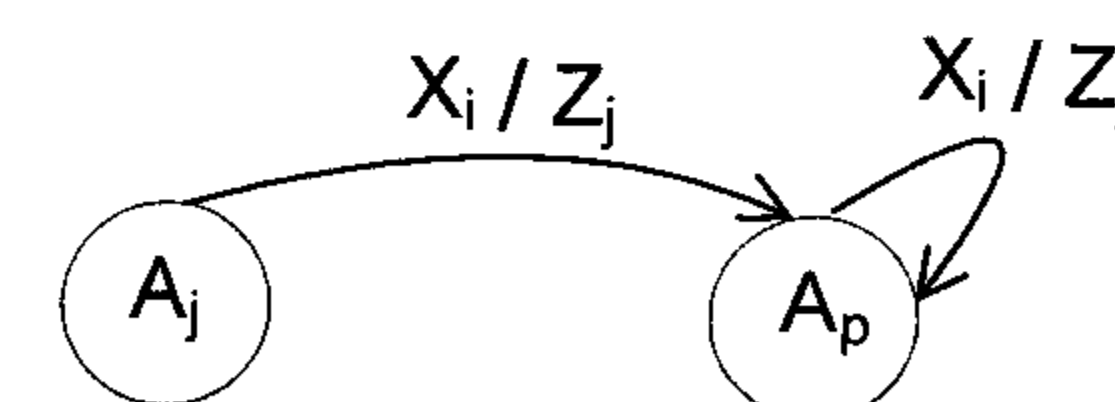


Фиг. 5.3 Структурен модел без елементи памет

Възможни са три начина на поведение на схема от този тип:

1. Еднократен преход.

При смяна на една входна дума с друга се формира нова изходна реакция, нови стойности на двоичните величини в обратните връзки и схемата преминава в ново състояние. Тя остава в това състояние при поддържане на входната дума. Графът на фиг. 5.4 илюстрира този начин на поведение.

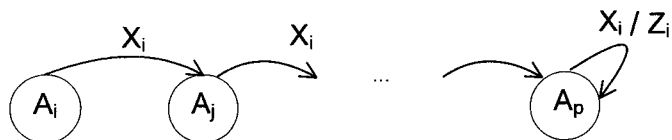


Фиг. 5.4 Еднократен преход

2. Многократен преход.

Под действие на дадена входна дума, преди да достигне до устойчиво състояние, автоматът преминава през едно или няколко

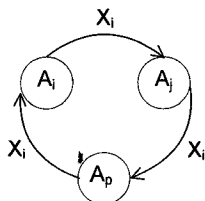
междинни (неустойчиви) състояния. Времето за превключване на схемата зависи от кратността на прехода. Графът на фиг. 5.5 илюстрира този начин на поведение.



Фиг. 5.5 Многократен преход

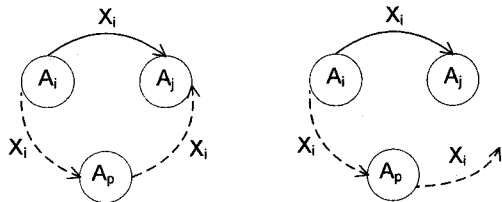
3. Зацикляне.

Схемата няма устойчиво състояние – тя цикли между няколко състояния. Състоянието, в което ще се установи, зависи от момента, в който се смени входното въздействие. Графът на фиг. 5.6 илюстрира зацикляне на автомата.



Фиг. 5.6 Многократен преход в цикъл

Схеми, реализирани по този модел, са чувствителни към състезания. В резултат на различната бързина, с която се разпространяват сигналите в различните клонове на комбинационната схема, е възможно автоматът да премине през няколко състояния, преди да се установи в състоянието, което е определено от функцията на преходите. Този вид преходи се наричат динамични. Ако вследствие на състезанията автоматът се установи в погрешно състояние, а не в предвиденото, преходът е критичен динамичен – фиг. 5.7.



Фиг. 5.7 Динамичен и критичен динамичен преход

За да реагират последователностните схеми на входните въздействия, необходимо е сигналите да имат някаква минимална продължителност, която зависи от параметрите на самите схеми. Максималната продължителност на сигналите може да бъде или да не

бъде ограничавана. Схеми, които не налагат ограничения върху максималната продължителност на сигналите, се наричат потенциални, а самите сигнали – потенциални сигнали. Съществуват схеми, които функционират правилно само ако входните сигнали не надхвърлят определена максимална продължителност. Тези схеми се наричат импулсни, а сигналите – импулсни сигнали или накратко импулси.

Многократните преходи (в цикъл или не), както и динамичните и критичните динамични преходи при този автоматен модел, могат да бъдат избегнати, ако входните въздействия имат импулсен характер и тяхната продължителност е ограничена до извършване на единствен преход.

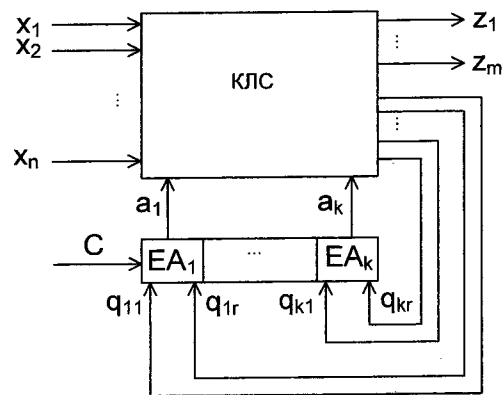
Схемите, които извършват само еднократни преходи, се наричат синхронни, а тези, които извършват както еднократни, така и многократни преходи, се наричат асинхронни.

5.3.2. Структурен модел с единичен блок памет.

В обратните връзки на схемата се поставят елементи памет, наречени още тригери или елементарни автомати. Тяхната функция е да съхраняват вътрешното състояние на схемата продължително време. Те се управляват чрез входовете им, където се подават т.нар. възбудителни функции на тригерите. Възбудителните функции определят новото състояние на елементите памет, респективно на схемата, а моментът, в който настъпва смяната на състоянията, се определя от т.нар. синхронизиращ сигнал. Синхронизиращият сигнал е импулсен сигнал, чието предназначение е да осигури еднократен преход на елементите памет. Той има две състояния: активно и пасивно, като

активното състояние е състоянието с ограничена продължителност.

Преходът на последователностната схема от едно състояние в друго се извършва по време на активното състояние. Освен че осигурява еднократен преход, синхронизиращият сигнал има и още едно предназначение – да освободи схемата от влиянието на състезанията, които по принцип са възможни в нейната комбинационна част. Неговото активно ниво се подава в момент, в който всички състезания са преминали,



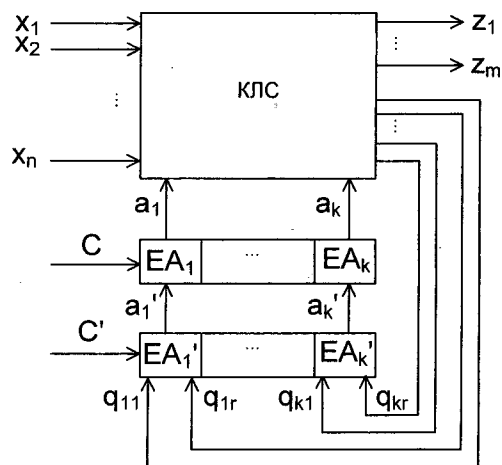
Фиг. 5.8. Структурен модел с единичен блок памет

възбудителните функции на тригерите са установени, а елементите памет съответно са предпазени от преходи в нежелани състояния.

Този начин на действие на структурния модел с единичен блок памет освобождава схемата от динамични и критични динамични преходи. Схеми от този тип извършват само еднократни преходи, т.е. те са синхронни. На фиг. 5.8. е показан структурният модел с единичен блок памет. „C” е синхросигналът; „г” е броят входове на елементарния автомат.

5.3.3. Структурен модел с двоен блок памет

Структурният модел с единичен блок памет осигурява необходимата функционалност на синхронните последователностни схеми, но строгото изискване за ограничена продължителност на синхронизацията импулсен сигнал е трудно реализуемо. Търсенето в посока замяна на импулсния сигнал с потенциален довежда до идеята за структурния модел с двоен блок памет. Блокът памет и синхросигналът са удвоени, за да се избегнат многократните преходи, които биха се случили при простата замяна на импулсния синхросигнал с потенциален. На фиг. 5.9



Фиг. 5.9 Структурен модел с двоен блок памет

е показан структурният модел с двоен блок памет. Спомагателният блок памет е този, който е означен с прим. Двата синхросигнала са потенциални, а активните им нива са разместени във времето. В момента, в който C' е в активно ниво, спомагателният блок памет променя съдържанието си в съответствие със стойностите на възбудителните функции на тригерите. В този момент спомагателният блок памет преминава в новото състояние, а

основният е в старото състояние. В момента, в който C' премине в неактивно ниво, C преминава в активно и съдържанието на спомагателния блок се копира в основния блок. За да бъде сигурно незастъпването на активните нива на C и C' във времето, те се правят взаимноинверсни. Тази структура е известна като структурата master-slave (управляващ-управляван).

5.4. Елементарни автомати

Елементарните автомати, наречени още елементи памет или тригери, са автомати на Мур с две вътрешни състояния (0 и 1), които притежават пълна система преходи. Пълна система преходи означава, че за всяка двойка вътрешни състояния A_i и A_j може да се намери поне една входна дума X_k , предизвикваща прехода $A_i \rightarrow A_j$.

Видове елементарни автомати

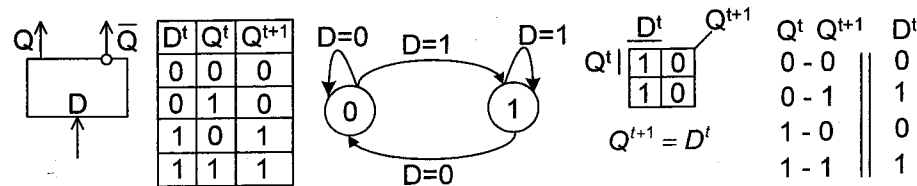
В зависимост от алгоритъма на функциониране тригерите се класифицират на следните основни видове: D, T, R-S, J-K.

Поведението на тригерите се описва по следните начини:

- словесно.
- таблично. Таблицата на преходите съдържа колонки за всички входни сигнали, за старото (Q^t) и за новото (Q^{t+1}) вътрешно състояние на тригера.
- графично. Графът на преходите има два върха, означени със състоянията 0 и 1 и четири дъги, съответстващи на четирите възможни прехода $0 \rightarrow 0$, $0 \rightarrow 1$, $1 \rightarrow 0$, $1 \rightarrow 1$.
- аналитично. Преходите на тригера се описват с логическа функция от вида $Q^{t+1} = F(Q^t, X^t)$. Тя най-лесно се получава от карта на Вейч, която се попълва съгласно таблицата на преходите.
- матрично. Матрицата на входовете има 4 реда, означени с четирите прехода на тригера и отделна колонка за всеки вход. В тези колонки се нанасят сигналите, които трябва да се подадат на дадения вход, за да се изпълни съответният преход. Ако за осъществяването на даден преход някой от входовете може да приема както стойност 0, така и 1, на съответното място в матрицата на входовете се записва неопределено значение X.

D тригер

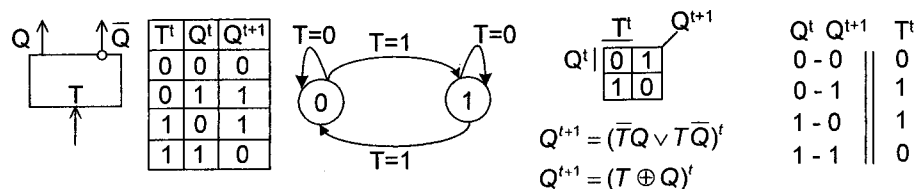
Неговото състояние повтаря сигнала на входа D. Тази дефиниция предполага задължителното използване на тактов сигнал, който обаче не е показан във формалните логически описания на фиг. 5.10.



Фиг. 5.10 D тригер

Т тригер

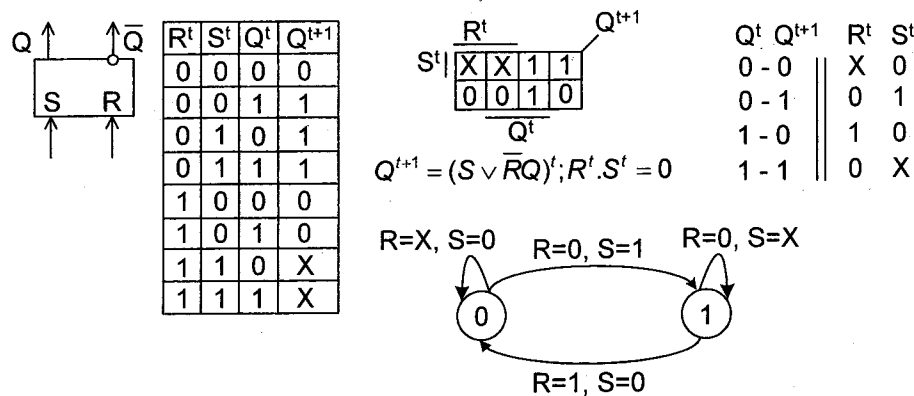
При входен сигнал $T=0$ тригерът запазва състоянието си, а при $T=1$ сменя състоянието си в противоположното.
Фиг. 5.11 показва всички останали начини на представяне на Т тригера.



Фиг. 5.11 Т тригер

R-S тригер

Входът R е вход за установяване в състояние 0 (нулев вход), входът S - за установяване в състояние 1 (единичен вход). Този тригер има два дублирани прехода - от 0 в 0 при $S=0$ и $R=X$ и от 1 в 1 при $R=0$ и $S=X$. Входната комбинация $R=S=1$ е забранена, т.е. $R.S=0$.
Фиг. 5.12 показва всички останали начини на представяне на R-S тригера.

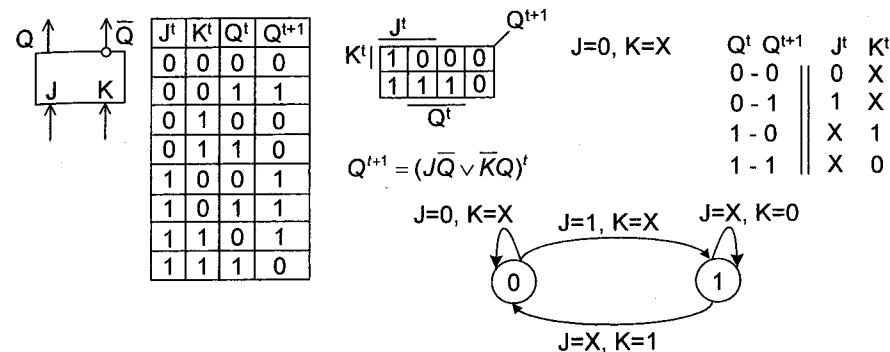


Фиг. 5.12 R-S тригер

J-K тригер

J входът е единичен, а K входът е нулев. Този тригер се отличава от R-S тригера по това, че няма забранена комбинация от входни сигнали. При $J=K=1$ той сменя състоянието си в противоположното на това, в което е бил, т.е. работи като Т тригер.

Фиг. 5.13 показва всички останали начини на представяне на J-K тригера.



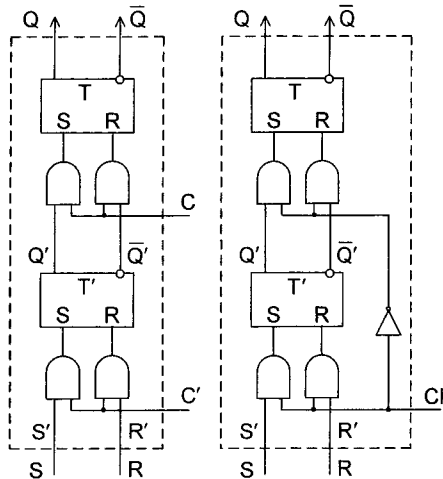
Фиг. 5.13 J-K тригер

В зависимост от начина на действието си тригерите се делят на асинхронни и синхронни. При асинхронните входната информация въздейства на тригера в момента, в който постъпва. При синхронните информацията се записва в тригера само по времето на тактовия импулс (C), получаван от отделен тактов генератор. Синхронните тригери от своя страна се делят на тактово управлявани (превключват по време на тактовия импулс), превключвани от фронта на тактовия импулс (превключват по време на положителния или отрицателния фронт) и тригери с двойна структура, които работят на принципа master-slave. Тактово управляваните тригери имат един съществен недостатък: входната информация се въвежда в тригера през цялото време, през което на тактовия вход C има ниво 1. С други думи, синхронният тригер се превръща в асинхронен и следва непосредствено измененията на сигналите на логическите си входове. Изход от това положение е да се наложи ограничението при $C=1$ сигналите на логическите входове на тригера да не се изменят. Този недостатък е отстранен при другите два вида синхронни тригери.

Принципната функционална схема на тригер тип R-S с двойна структура е показана на фиг. 5.14.

Тя се състои от два тригера, които се управляват от отделни тактови поредици C и C' . Първият (спомогателният, означеният с прим) тригер реагира по време на активното ниво 1 на тактовия сигнал C' . Основният тригер приема състоянието на спомогателния веднага след прехода от 0 в 1 на тактовия сигнал C . За нормалната работа на тригера с двойна структура като синхронен е необходимо активните нива $C'=1$ и $C=1$ на двете тактови поредици да не се застъпват, т.е. да е изпълнено условието $C'.C = 0$.

Тригерът с двойна структура може да се управлява и само от един тактов сигнал C , ако неговата инверсия се използва вътрешно като



Фиг. 5.13 Тригери с двойна структура

втори такт – $C' = CI$ и $C = \bar{C}I$. Схемата е показана на фиг. 5.13. На базата на тази схема се изгражда синхронен тригер тип J-K и синхронен тригер тип T, които работят с потенциални входни сигнали. Това е наложително, за да се избегнат възможните динамични преходи при поддържане на ниво 1 на J и K входовете и на T входа на съответните тригери. Тригерите от тип D се реализират само като тактувани.

5.5. Синтез на автомати с памет

Синтезът на последователностни схеми започва от задаване на автомата на някакъв език и завършва с получаването на структурната му схема. Първият

етап на синтеза е абстрактният синтез. Негова задача е формализиране на описанието и привеждането му в табличен или графов вид. След това се извършва анализ, целящ намаляването на броя на вътрешните състояния на автомата, като се предполага, че схемата ще стане по-проста. На тези етапи на синтеза няма да се спираме. Етапът, който ще разгледаме, е структурният синтез на последователностната схема. Изходен материал за провеждане на структурния синтез на схема с памет са нейните автоматни таблици или графът на преходите.

5.5.1. Структурен синтез на автомати със синхронизирани ЕП

Ще разгледаме алгоритъма за структурен синтез на ПС със синхронизирани ЕП с помощта на следния пример: Да се построи структурна схема на автомата, зададен с обща таблица на преходите и изходите (табл. 5.5). Да се използват T елементи памет и логически елементи И, ИЛИ, НЕ.

X \ A	A ₀	A ₁	A ₂	A ₃
X ₀	A ₀ Z ₀	A ₁ Z ₀	A ₂ Z ₀	A ₃ Z ₀
X ₁	A ₁ Z ₀	A ₂ Z ₀	A ₃ Z ₀	A ₀ Z ₁

Табл. 5.5

Стъпка 1: Кодирание на входните въздействия, вътрешните състояния и изходните реакции на автомата.

При N на брой входни въздействия входните променливи са n, където $\lfloor \log_2 N \rfloor \leq n \leq N$. В случая $N=2$, а $\log_2 2=1$, т.е. $n=1$. По аналогичен начин се определя броят на вътрешните променливи (тригерите) в съответствие с броя на вътрешните състояния, както и броят на изходните променливи в съответствие с броя на изходните реакции. За броя на вътрешните променливи получаваме $\log_2 4=2$, а за изходните променливи $\log_2 2=1$. Входната променлива именуваме x, вътрешните – Q₁ и Q₂, а изходната променлива – z. Таблиците от фиг. 5.14 показват избраното кодиране.

Стъпка 2: Построява се кодираната таблица на преходите и изходите – КТПИ.

X \ x	x
X ₀	0
X ₁	1

A \ Q ₁ Q ₂	Q ₁ Q ₂
A ₀	0 0
A ₁	0 1
A ₂	1 0
A ₃	1 1

Z \ z	z
Z ₀	0
Z ₁	1

Фиг. 5.15. Кодирание на външните и вътрешните състояния

На базата на ТПИ на автомата се построява разгъната кодирана ТПИ (РКТПИ) по следния начин: В колони се записват всички възможни комбинации от стойности на входните и вътрешните променливи в момента t. До всяка комбинация се записват кодираното ново състояние на автомата и кодираната изходна реакция, взети от ТПИ на автомата. За разглеждания пример РКТПИ представлява първите шест колони от таблицата, показана на фиг. 5.16.

Стъпка 3: Избор на ЕА. Разширение на РКТПИ.

X ^t	A ^t		A ^{t+1}		Z ^t	Y ^t	
x	Q ₁	Q ₂	Q ₁	Q ₂	z	T ₁	T ₂
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	1	1	0	0	1	1
1	1	0	1	1	0	0	1
1	1	1	0	0	1	1	1

Фиг. 5.16 РКТПИ на зададения автомат

Етапът на избор на елементарни автомати отпада, тъй като типът им е зададен в условието на задачата.

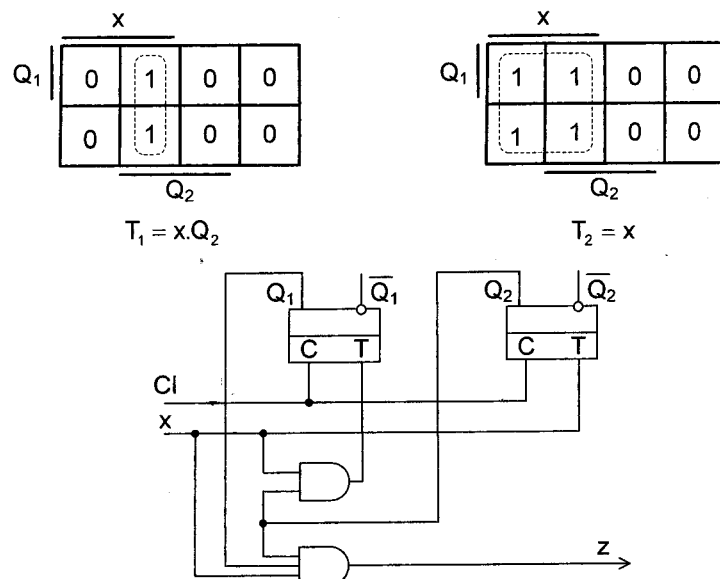
РКТПИ се допълва с възбудителните функции на тригерите. За тяхното съставяне се използва, от една страна матрицата на входовете на съответния тип ЕА, а от друга, желаният преход на тригера, който присъства в РКТПИ. Колоните в сиво, които допълват РКТПИ, представляват възбудителните функции на T тригерите, съставени по описания начин.

Стъпка 4: Определяне на възбудителните и изходните функции

Намират се най-подходящите за реализиране аналитични форми на възбудителните и изходните функции. Ако е зададен елементарен базис И, ИЛИ, НЕ, най-подходящи за реализиране са минималните форми на функциите. Построяват се карти на Вейч, нанасят се функциите и се определя минималната им форма. Минимални форми са желателни и при реализация с ПЛМ. Ако като елементарен базис са зададени дешифратори, мултиплексори или ПП, минимизация не е необходима.

Стъпка 5: Построява се структурната схема на автомата.

Картите на Вейч, минималните форми на функциите и структурната схема на конкретния автомат са показани на фиг. 5.17. Карта на Вейч за изходната функция z не е необходима. Очевидно, $z = x.Q_1.Q_2$.



Фиг. 5.17. Възбудителни функции и структурна схема на автомата

5.5.2. Особенности на синтеза на частични автоматы с памет

Частични се наричат автоматите, които се характеризират с непълнота в автоматните таблици. Такава ситуация възниква, когато при логаритмуването за определяне броя на входните и (или) вътрешните променливи резултатът не е цяло число и се налага закръгляне към по-голямо цяло число. В тези случаи не се използват всички възможни комбинации на двоичните величини за кодиране на входните въздействия и (или) вътрешни състояния. По тази причина възбудителните и изходните функции се получават непълно

определени. Друга причина за неопределеност на възбудителните и изходните функции може да бъде предварително поставено условие някои входни въздействия да не се подават при някои определени вътрешни състояния. Трета причина за непълно определени възбудителни функции са дублираните преходи в матриците на входовете на някои типове елементи памет. Неопределеността на функциите на комбинационния блок в автомата се използва при синтеза за получаване на оптимални за реализация аналитични форми на възбудителните и изходните функции.

Да се проведе структурен синтез на автомата, зададен с ТПИ от табл. 5.6. Да се използват J-K тригери и И, ИЛИ, НЕ логически елементи.

$\backslash A$	A_0	A_1	A_2
X			
X_1	A_1 Z_3	A_2 Z_2	A_1 Z_2
X_2	—	A_0 Z_3	A_2 Z_1
X_3	A_0 Z_2	—	A_2 Z_3

Определяме броя входни променливи и ги именуваме: $\lceil \log_2 3 \rceil = 2$ (x_1, x_2). Аналогично за вътрешните и изходните променливи: $\lceil \log_2 3 \rceil = 2$ (Q_1, Q_2), $\log_2 4 = 2$ (z_1, z_2).

Таблиците на кодиране са показани на фиг. 5.18, а КТПИ – на фиг. 5.19.

Табл. 5.6

$\backslash X$	X_1	X_2	X_3
x			
x_1	0	0	1
x_2	0	1	0

$\backslash A$	A_0	A_1	A_2
Q			
Q_1	0	0	1
Q_2	0	1	1

$\backslash Z$	Z_1	Z_2	Z_3	Z_4
z				
Q_1	1	0	1	0
Q_2	0	0	1	1

Фиг. 5.18 Кодиране на външните и вътрешните състояния

X^t	A^t	A^{t+1}	Z^t	Y^t
$x_1 \ x_2$	$Q_1 \ Q_2$	$Q_1 \ Q_2$	$z_1 \ z_2$	$J_1 \ K_1 \ J_2 \ K_2$
0 0	0 0	0 1	1 1	0 x 1 x
0 0	0 1	1 1	0 0	1 x x 0
0 0	1 0	x x	x x	x x x x
0 0	1 1	0 1	0 0	x 1 x 0
0 1	0 0	- -	x x	x x x x
0 1	0 1	0 0	1 1	0 x x 1
0 1	1 0	x x	x x	x x x x
0 1	1 1	1 1	1 0	x 0 x 0
1 0	0 0	0 0	0 0	0 x 0 x
1 0	0 1	- -	x x	x x x x
1 0	1 0	x x	x x	x x x x
1 0	1 1	1 1	1 1	x 0 x 0
1 1	0 0	x x	x x	x x x x
1 1	0 1	x x	x x	x x x x
1 1	1 0	x x	x x	x x x x
1 1	1 1	x x	x x	x x x x

Фиг. 5.19 РКТПИ на зададения частичен автомат

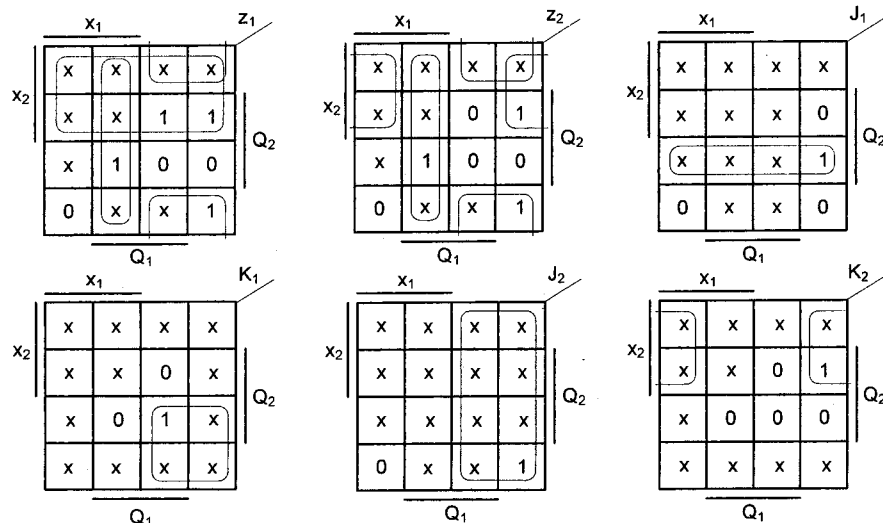
РКТПИ е допълнена с 4 колони, съответстващи на четирите възбудителни функции (J_1, K_1, J_2, K_2). Клетките в тези колони са попълнени въз основа на матрицата на входовете на J-K елемента памет – фиг. 5.12.

Чрез карти на Вейч (фиг. 5.20) определяме минималните форми на функциите на изходите и на възбуждане:

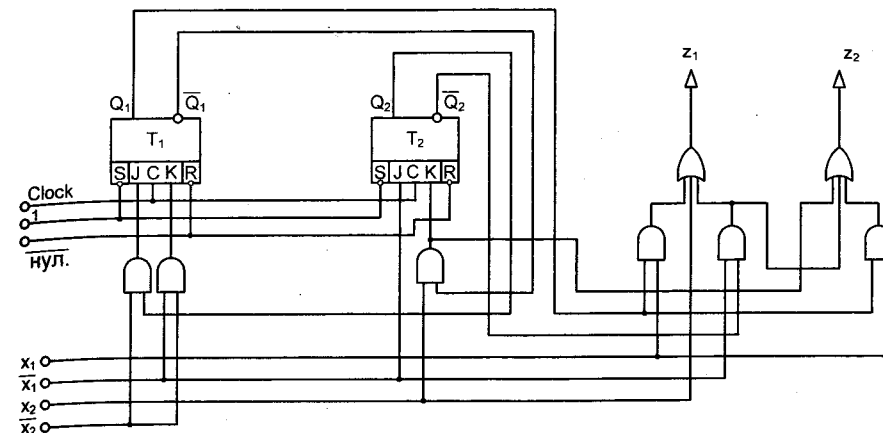
$$\begin{aligned} z_1 &= x_2 \vee x_1 \cdot Q_1 \vee \bar{x}_1 \cdot \bar{Q}_2 & z_2 &= x_2 \cdot \bar{Q}_1 \vee \bar{x}_1 \cdot \bar{Q}_2 \vee x_1 \cdot Q_1 \\ J_1 &= \bar{x}_2 \cdot Q_2 & K_1 &= \bar{x}_1 \cdot \bar{x}_2 \\ J_2 &= \bar{x}_1 & K_2 &= x_2 \cdot \bar{Q}_1 \end{aligned}$$

Получената в резултат на синтеза структурна схема не съдържа никакви непълноти. При включване на захранването или пик в напрежението е възможно тя да попадне в неработно състояние. (Неработни са състоянията, които са получени при доопределяне функциите на автомата, но които не фигурират в зададените автоматни таблици). Изход от това положение е да се предвиди вход, с помощта на който схемата да може да бъде приведена в нулево състояние (всички тригери нулирани) в произволен момент. За целта се използват асинхронните S и R входове на интегрално изпълнените ЕП. Активно ниво за тези входове е 0, което личи от означеното кръгче. На входа S се подава константа 1, а на входа R се предвижда подаване на константа 0, когато е необходимо.

Построяваме структурната схема на автомата – фиг. 5.21.



Фиг. 5.20 Карти на Вейч на функциите на изходите и функциите на възбуждане



Фиг. 5.21 Структурна схема

Автоматът, който разглеждаме, освен че има неработно състояние, има и забранени преходи – по някаква причина не се предвиждат преходи от A_0 под действие на X_2 и от A_1 под действие на X_3 . Трябва да направим проверка какво би се случило, ако все пак се създадат условия за осъществяване на тези преходи. От картите на Вейч установяваме стойностите на възбудителните функции за набори 0100 и 1001, а оттам и новото състояние на тригерите. Изводът е, че автоматът ще премине в работно състояние – 01 (A_1) и 11 (A_2).

Това е достатъчно, за да приемем, че автоматът ще бъде работоспособен.

5.6. Анализ на автомати със синхронизирани елементи памет

При синхронните автомати с памет динамичен анализ на комбинационната част не се прави, тъй като синхросигналът по дефиниция се прилага едва след като са преминали всички състезания в схемата. Автоматите от този тип не се изследват и за критични динамични преходи, тъй като идеята за синхронизацията ги изключва.

Статичният анализ е процес, обратен на синтеза. Изходен материал е структурната схема на автомата, а резултат от анализа е таблица на преходите и изходите или граф, описващ поведението на схемата.

Нека направим анализ на схемата, която току-що синтезирахме.

Стъпка 1: Определят се възбудителните функции на тригерите и изходните функции в аналитичен вид: $J_1 = \bar{x}_2 \cdot Q_2$, $K_1 = \bar{x}_1 \cdot \bar{x}_2$, $J_2 = \bar{x}_1$, $K_2 = x_2 \cdot \bar{Q}_1$, $z_1 = x_2 \vee x_1 \cdot Q_1 \vee \bar{x}_1 \cdot \bar{Q}_2$, $z_2 = x_2 \cdot \bar{Q}_1 \vee \bar{x}_1 \cdot \bar{Q}_2 \vee x_1 \cdot Q_1$.

Стъпка 2: Построява се КТПИ, но в ред, обратен на този при синтеза. След разписването на всички набори на двоичните променливи x_1, x_2, Q_1, Q_2 се попълват таблиците на истинност на функциите $J_1, K_1, J_2, K_2, z_1, z_2$.

z₂. След като са известни стойностите на функциите на възбуждане и старото състояние на тригера, се определя новото състояние. (При синтеза новото състояние е известно от ТПИ и въз основа на желаните преходи на тригерите, определяме функциите на възбуждане). Фиг. 5.22 съдържа разгънатата кодирана таблица на преходите и изходите.

X ^t		A ^t		A ^{t+1}		Z ^t		Y ^t			
x ₁	x ₂	Q ₁	Q ₂	Q ₁	Q ₂	z ₁	z ₂	J ₁	K ₁	J ₂	K ₂
0	0	0	0	0	1	1	1	0	1	1	0
0	0	0	1	1	1	0	0	1	1	1	0
0	0	1	0	0	1	1	1	0	1	1	0
0	0	1	1	0	1	0	0	1	1	1	0
0	1	0	0	0	1	1	1	0	0	1	1
0	1	0	1	0	1	1	1	0	0	1	0
0	1	1	1	1	1	1	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	1	0	0	1	0	0	0
1	0	1	0	1	0	1	1	0	0	0	0
1	0	1	1	1	1	1	1	1	0	0	0
1	1	0	0	0	0	1	1	0	0	0	1
1	1	0	1	0	0	1	1	0	0	0	1
1	1	1	0	1	0	1	1	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0

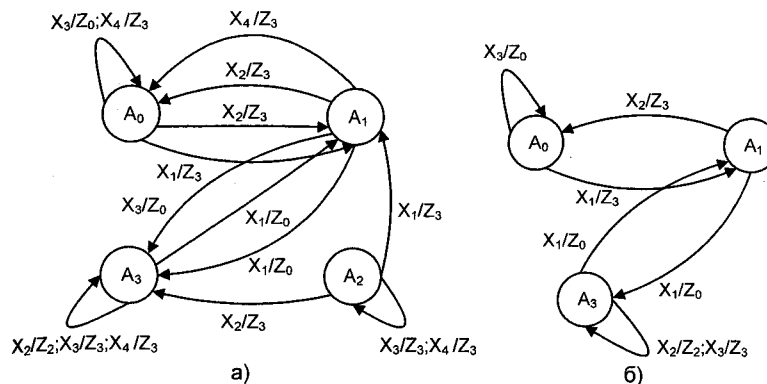
Фиг. 5.22 РКТПИ

Стъпка 3: Въвеждат се обобщени означения и се строят абстрактните автоматни таблици или граф.

При анализа на всеки автомат с памет, РКТПИ е винаги напълно определена. Предположения дали автоматът е частичен или пълен се правят след анализ на получения автоматен граф. Валидни са съображения от следния общ характер: Ако в графа има върхове, към които или от които няма преходи към други върхове, то тези върхове вероятно съответстват на неработно състояние.

Ако съществува входно въздействие, което привежда автомата в неработно състояние, но няма входна дума, под действие на която последователностната схема да се върне в някое от работните състояния, очевидно това входно въздействие е забранено.

За примера, който разглеждаме, прилагането на стъпка 3 довежда до резултата, показан на фиг. 5.23 а.



Фиг. 5.23 Графи, описващи действието на автомата

При съставянето на графа са приети следните обобщени означения: Състояние 00 е наречено A₀, 01 е наречено A₁, 10 и 11 – съответно A₂ и A₃.

За комбинациите от стойности на входните променливи x₁ и x₂ са приети следните означения: на комбинация 00 съответства X₁, на 01 → X₂, на 10 → X₃, на 11 → X₄.

За стойностите на изходните променливи z₁ и z₂ е прието: комбинация 00 да бъде наречена Z₀, 01, 10 и 11 съответно Z₁, Z₂, Z₃.

Внимателният оглед на графа показва, че автоматът по никакъв начин не може да попадне в състояние A₂. Това означава, че състояние A₂ е неработно. Под действие на входната дума X₄ се извършват преходи, изцяло дублирани от преходи под действие на други входни думи. Това обстоятелство би могло да означава, че X₄ е входна дума, която е излишно да бъде подавана на входовете на схемата. Изцяло дублирани са и преходите от A₀ в A₁ под действие на X₁ и X₂. Същото се отнася и за преходите от A₁ в A₃ под действие на X₁ и X₃. Тези разсъждения водят до построяване на редуцирания граф, показан на фиг. 5.23 б.

Поведението на последователностната схема, описана от графа на фиг. 5.23 б, изцяло съвпада с поведението на схемата, зададена с ТПИ от табл. 5.6. Ако възприемем кодирането от фиг. 5.17, ще получим пълно съвпадение между автоматите, зададени с автоматната таблица и графа.

Задание

1. Да се синтезира ПС (автомат на Мур), зададена чрез ТП, кодиране на входните въздействия, вътрешните състояния, изходните реакции и типа на елементите памет.

2. Да се проследи действието на последователностната схема при зададени входна последователност и начално състояние.
3. Да се синтезира частична ПС (автомат на Мили) при зададени ТПИ, кодиране на входните въздействия, вътрешните състояния, изходните реакции и типа на елементите памет. Комбинационната част от схемата да се реализира с дешифратори.
4. Да се проследи действието на последователностната схема при зададени входна последователност и начално състояние.
5. Да се проследи и обясни действието на ПС при забранен входен набор и/или забранено вътрешно състояние.
6. Да се направи анализ на работата на даден автомат.
7. Да се изследва дали някои от състоянията могат да бъдат неработни, а някои от входните въздействия – фиктивни.

Контролни въпроси

1. Коя е принципната разлика в структурата на КПС и ПС?
2. Формулирайте основната разлика между схема, работеща по автоматен модел на Мили, и схема, работеща по автоматен модел на Мур.
3. Ако два автомата са функционално еквивалентни, но единият е реализиран като автомат на Мили, а другият – като автомат на Мур, кой от двата е с по-голям брой вътрешни състояния и защо?
4. Какви са недостатъците на структурния модел на ПС без елементи памет? По какъв начин тези недостатъци са преодоляни в следващите структурни модели?
5. По какво се различават в действието си синхронните и асинхронните автомати?
6. Тригерите от кой тип задължително трябва да бъдат реализирани като синхронни автомати?
7. Кой са основните стъпки, през които преминава синтезът на една ПС?
8. Кодирането на вътрешните състояния влияе ли върху функциите на възбуждане на тригерите? А на изходните функции? Върху кои функции влияе начинът на кодиране на входните въздействия?
9. Кога една ПС се нарича частична? По какъв начин се отразява частичността на автомата върху възбудителните и изходните функции?
10. Може ли да се определят новото състояние и изходната реакция, ако ПС е поставена в забранено вътрешно състояние?
11. Може ли да се определят новото състояние и изходната реакция, ако на ПС се подаде забранено входно въздействие?
12. Кой са евристичните правила, според които може да се направи предположение, че анализираната схема реализира частичен автомат?

ТЕМА 6

БРОЯЧИ – ФУНКЦИИ, ВИДОВЕ, СИНТЕЗ НА ПЪЛНИ И ЧАСТИЧНИ БРОЯЧИ. БРОЯЧИ В ИНТЕГРАЛНО ИЗПЪЛНЕНИЕ. РЕГИСТРИ – ФУНКЦИИ, ВИДОВЕ, СИНТЕЗ. РЕГИСТРИ В ИНТЕГРАЛНО ИЗПЪЛНЕНИЕ

6.1. Броячи

6.1.1 Начин на функциониране

Броячът е последователностна схема, чийто алгоритъм на функциониране извършва преброяване на постъпващите входни въздействия. Всеки брояч се характеризира с коефициент или модул на броене. Коефициентът на броене е равен на броя на работните вътрешни състояния на автомата, а броят тригери, от които се състои броячът, представлява неговата разрядност. При постъпване на определена входна дума автоматът преминава в състояние, чийто номер е с единица по-голям от номера на предходното. При достигане състоянието с най-голям номер следващият преход е към началното състояние. Този начин на функциониране се отнася за брояч, работещ в режим на сумиране. Нарича се сумиращ брояч или брояч в права посока. Изваждащият брояч, или броячът в обратна посока, сменя номера на състоянията, през които преминава в намаляващ ред. От състоянието с най-малък номер следва преход в състоянието с най-голям номер.

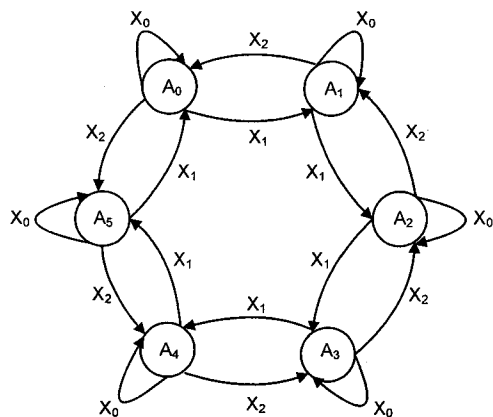
6.1.2 Видове броячи

Първата класификация е вече споменатата - според посоката на броене. Брояч, който брой в права или обратна посока под действие на различни входни думи, се нарича реверсивен брояч. На фиг. 6.1 е показан граф на реверсивен брояч до 6. Под действие на входна дума X_0 броячът запазва състоянието си, под действие на X_1 брой в права посока, а под действие на X_2 – в обратна посока.

Друга класификация е според това каква част от вътрешните състояния на автомата са работни. Ако всички състояния са работни, броячът е пълен; в противен случай е частичен.

Трета класификация е според начина на разпространение на преноса. Различаваме брояч с паралелен пренос, с текущ пренос и с последователен пренос.

Четвърта класификация е според начина на функциониране: синхронни и асинхронни. При синхронните броячи състоянието на тригерите се сменя едновременно, а при асинхронните промяната на състоянието на младшия тригер води до промяна в състоянието на по-старшите.



Фиг. 6.1 Граф на реверсивен брояч до 6

6.1.3 Синтез на броячи

Броячът е автомат на Мур, при който изходните реакции се кодират по същия начин, както и вътрешните състояния. Изходи на тази последователна схема са правите изходи на тригерите, от които е съставена. Синтезът на синхронните броячи протича по класическия алгоритъм за синтез на синхронни автомати с памет. Предпочитаният тип тригери е Т, тъй като възбудителните им функции се получават най-прости. Възможно е J-K тригер да се преобразува в Т тригер, като на двата входа J и K едновременно се подава входното въздействие X. Друга възможност е на входовете J и K да се подаде константа 1, а импулсите за броене да се подават на тактовите входове. Частичните броячи се синтезират по правилата за синтез на частични автомати с памет.

В табл. 6.1 е показана таблицата на преходите на пълен 3-разряден сумиращ брояч. Коефициентът му на броене е 8, тъй като $2^3=8$. Под действие на входна дума X_1 броячът запазва състоянието си, а под действие на X_2 брои в права посока. Начално е състоянието A_0 .

A \ X	A ₀	A ₁	A ₂	A ₃	A ₄	A ₅	A ₆	A ₇
X ₁	A ₀	A ₁	A ₂	A ₃	A ₄	A ₅	A ₆	A ₇
X ₂	A ₁	A ₂	A ₃	A ₄	A ₅	A ₆	A ₇	A ₀

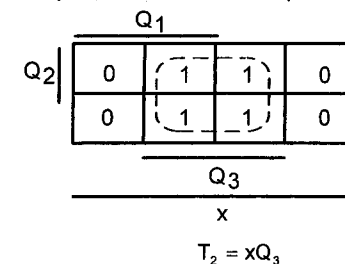
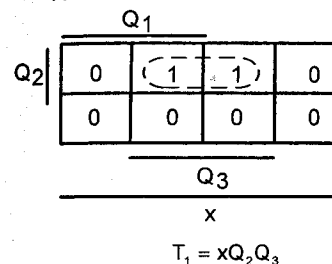
Табл. 6.1

Входните въздействия кодираме с една входна променлива x . Приемаме, че $x = 0$ съответства на входна дума X_1 , а $x = 1$ - на входна дума X_2 . В табл. 6.2 е дадена кодираната таблица на преходите. Работим с Т тригери. При $x = 0$ броячът не променя състоянието си и функциите на възбуждане на Т тригерите ще бъдат равни на нула. Строим таблица с 8 реда, която включва само входното въздействие $x = 1$.

X^i	A^i			A^{i+1}			Y^i		
x	Q ₁	Q ₂	Q ₃	Q ₁	Q ₂	Q ₃	T ₁	T ₂	T ₃
1	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	0	0	1	1
1	0	1	0	0	1	1	0	0	1
1	0	1	1	1	0	0	1	1	1
1	1	0	0	1	0	1	0	0	1
1	1	0	1	1	1	0	0	1	1
1	1	1	0	1	1	1	0	0	1
1	1	1	1	0	0	0	1	1	1

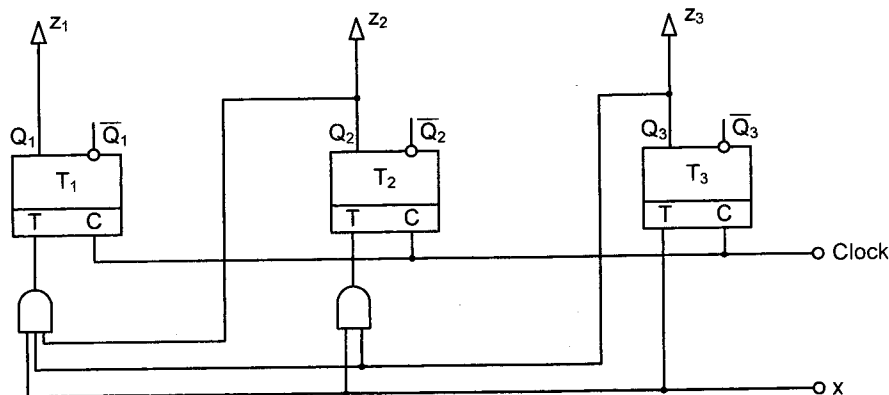
Табл. 6.2

Определяме възбудителните функции на тригерите. От таблица 6.2 се вижда, че $T_3 = x$. С помощта на карти на Вейч определяме T_1 и T_2 – фиг. 6.2.



Фиг. 6.2 Възбудителни функции на тригерите

Полученият резултат за T_1 , T_2 и T_3 е логичен, като се има предвид, че най-младшият тригер Q_3 сменя състоянието си всеки път, а по-старшите Q_2 и Q_1 сменят състоянието си при наличие на пренос. На фиг. 6.3 е показана схемата на брояча.



Фиг. 6.3 Структурна схема на брояч до 8 с паралелен пренос

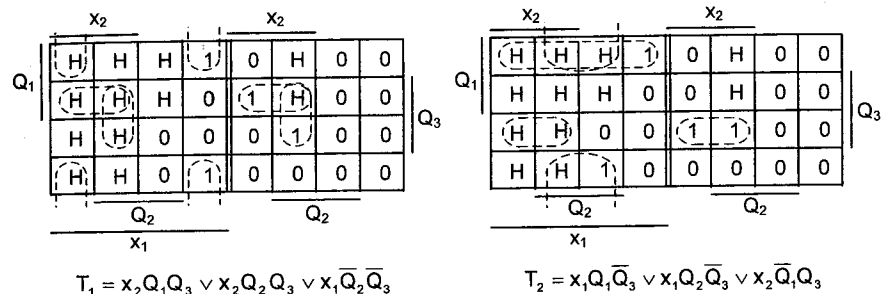
Като пример за частичен брояч ще разгледаме реверсивния брояч, чийто граф е даден на фиг. 6.1. Входните въздействия кодираме както следва: $X_0 - 00$, $X_1 - 01$, $X_2 - 10$. Под действие на X_0 автоматът запазва състоянието си, под действие на X_1 брои в права посока, а под действие на X_2 - в обратна посока. Вътрешните състояния от A_0 до A_5 са кодирани с двоичните цифри от 0 до 5. В табл. 6.3 е дадена разгънатата кодирана таблица на преходите на автомата. Тя е с 16 вместо с 32 реда, тъй като Т тригерите не променят състоянието си под действие на входна дума 00 и възбудителните им функции за тази част от таблицата са нула. Освен това, входна дума 11 не се използва, т.е. наборите, за които x_1 и x_2 имат стойност 1, са фиктивни.

На фиг. 6.4 са показани карти на Вейч, в които са нанесени възбудителните функции на T_1 и T_2 (очевидно е, че $T_3 = x_1 \vee x_2$), а на фиг. 6.5 е дадена схемата на автомата.

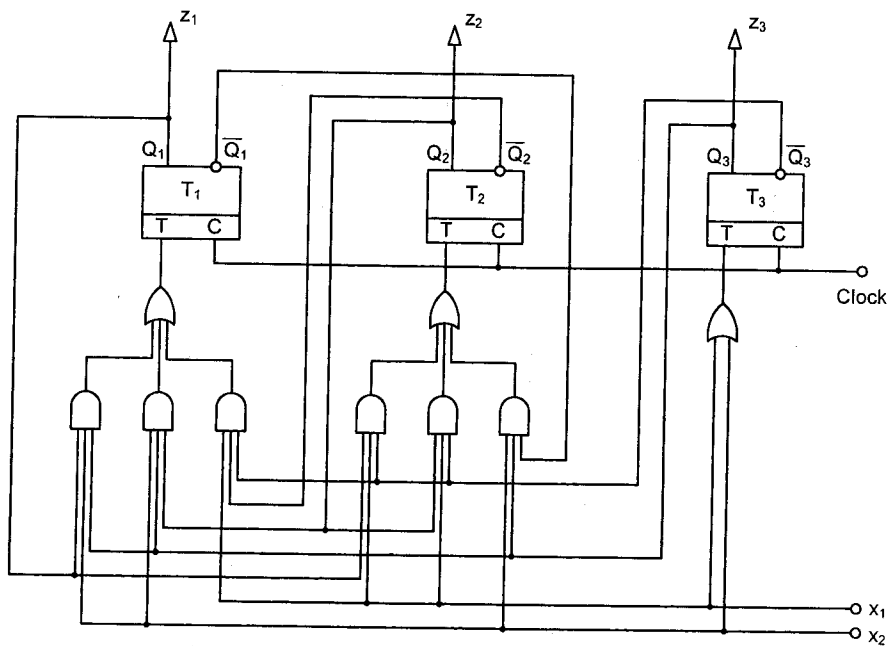
Нека се върнем на функциите на възбуждане на тригерите на пълния триразряден брояч: $T_3 = x$, $T_2 = xQ_3$, $T_1 = xQ_2Q_3$. Друг начин на тяхното представяне е $T_3 = x$, $T_2 = T_3Q_3$, $T_1 = T_2Q_2$. Ако функциите се реализират по първия начин (фиг. 6.3), броячът е с паралелен пренос, а ако се реализират по втория начин (фиг. 6.6), броячът е с текущ пренос. Бързодействието на брояча с текущ пренос е по-малко, тъй като за формирането на възбудителната функция на даден тригер е необходимо вече да е формирана възбудителната функция от предходния разряд. За сметка на това се използват само двувходови елементи И.

X^t		A^t			A^{t+1}			Y^t		
x_1	x_2	Q_1	Q_2	Q_3	Q_1	Q_2	Q_3	T_1	T_2	T_3
0	1	0	0	0	0	0	1	0	0	1
0	1	0	0	1	0	1	0	0	1	1
0	1	0	1	0	0	1	1	0	0	1
0	1	0	1	1	1	0	0	1	1	1
0	1	1	0	0	1	0	1	0	0	1
0	1	1	0	1	0	0	0	1	0	1
0	1	1	1	0	X	X	X	H	H	H
0	1	1	1	1	X	X	X	H	H	H
1	0	0	0	0	1	0	1	1	0	1
1	0	0	0	1	0	0	0	0	0	1
1	0	0	1	0	0	0	1	0	1	1
1	0	0	1	1	0	1	0	0	0	1
1	0	1	0	0	0	1	1	1	1	1
1	0	1	0	1	1	0	0	0	0	1
1	0	1	1	0	X	X	X	H	H	H
1	0	1	1	1	X	X	X	H	H	H

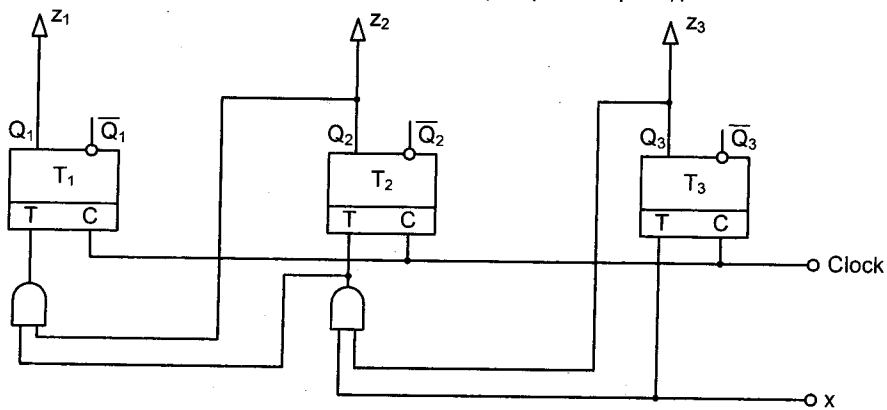
Табл. 6.3



Фиг. 6.4 Възбудителни функции на тригерите



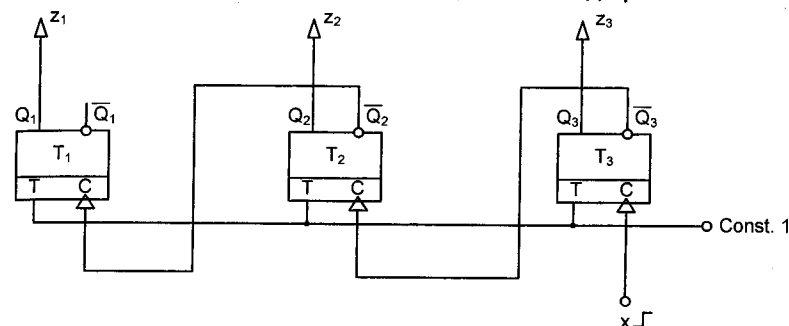
Фиг. 6.5 Структурна схема на реверсивен брояч до 6



Фиг. 6.6 Структурна схема на брояч до 8 с текущ пренос

Третият вид брояч според различния начин на разпространение на преноса е броячът с последователен пренос. Този вид брояч е асинхронен.

На фиг. 6.7 е показана схемата на триразряден асинхронен брояч. Използваните Т тригери превключват при преден фронт (0→1) на тактовия импулс. При този вид броячи елементарните автомати не сменят състоянието си едновременно. След промяната в състоянието на най-младшия тригер следва евентуална промяна в състоянията на по-старшите. Максималното времезакъснение се получава при преход от състояние 111 в състояние 000. Преди да достигне състояние 000, броячът преминава през 2 междинни състояния: 110 и 100. Като се вземе предвид, че времезакъснението на елементите памет е многократно по-голямо от това на логическите елементи, става ясно, че този вид брояч е най-бавен.

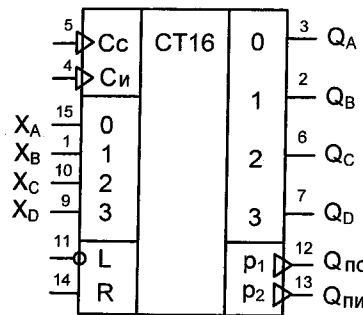


Фиг. 6.7 Структурна схема на триразряден асинхронен брояч - брояч с последователен пренос

6.1.4 Броячи в интегрално изпълнение

От броячите в интегрално изпълнение ще разгледаме схемите 74LS193 и 74LS93.

74LS193 е синхронен реверсивен брояч до 16. На фиг. 6.9 са показани изводите на схемата.

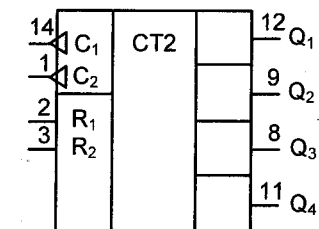


Фиг. 6.8 Изводи на схема 74LS193

Тригерите превключват от положителния фронт на импулсите, подавани на входовете Cc или Ci. Cc и Ci са тактови входове за работа съответно в режим на събиране и изваждане. В режим на събиране трябва Si=1, а в режим на изваждане Cc=1. Qpc и Qpi са сигнали за пренос в режим на събиране и изваждане. При последователно включване на броячи Qpc и Qpi се включват съответно към Cc и Ci на следващия брояч. XA, XB, XC, XD са информационни входове за установяване

тригерите на брояча в определено състояние. L е установяващ вход. При $L = 0$ тригерите се установяват в състояние, определено от сигналите X_A , X_B , X_C , X_D . R е нулиращ вход. Тригерите се нулират при $R = 1$, а броячът брои при $R = 0$.

74LS93 е асинхронен брояч с коефициент на броене 2, 8 или 16. На фиг. 6.9 са показани изводите на схемата. Схемата се състои от 4 тригера, които превключват по задния фронт на входните импулси. Изходът на първия тригер не е свързан вътрешно с входа на следващия и това позволява той да се използва като брояч на 2, а останалите 3 тригера съставляват брояч на 8. Вход на първия брояч е C_1 , а на втория - C_2 . Ако е необходим брояч до 16, трябва външно да се свърже изходът на първия тригер (Q_1) с входа на втория тригер (C_2). R_1 и R_2 са нулиращи входове, обединени с операция И. Тригерите се нулират при $R_1=1$ и $R_2=1$. Броячът работи при $R_1 = 0$ или $R_2 = 0$.



Фиг. 6.9 Изводи на схема 74LS93

6.2. Регистри

6.2.1 Начин на функциониране и видове

Обикновеният регистър е подредена съвкупност от n елемента памет, предназначена за помнене на двоични n -разрядни числа, всяко от които съвпада с едно от всичките 2^n възможни вътрешни състояния на автомата. Броят на тригерите определя т.нар. дължина или разрядност на регистъра. Под въздействие на входна дума i автоматът преминава в състояние i независимо от състоянието му в предходния момент. В този смисъл обикновеният регистър е регистър с паралелен запис на информацията.

Преместващ е регистърът, който има възможност да преобразува запаметената дума чрез изменение на положението на разрядите на думата спрямо началното им положение. Между тригерите на преместващия регистър са изградени връзки, наречени вериги на преместване. Те осигуряват възможността за предаване на символите на запомнената дума от едни елементи памет към други. На базата на изместващ регистър се реализира последователен запис на

информацията. Освен основните операции (запис и/или преместване) даден регистър може да има допълнителни функции: поразрядни логически операции, нулиране и др.

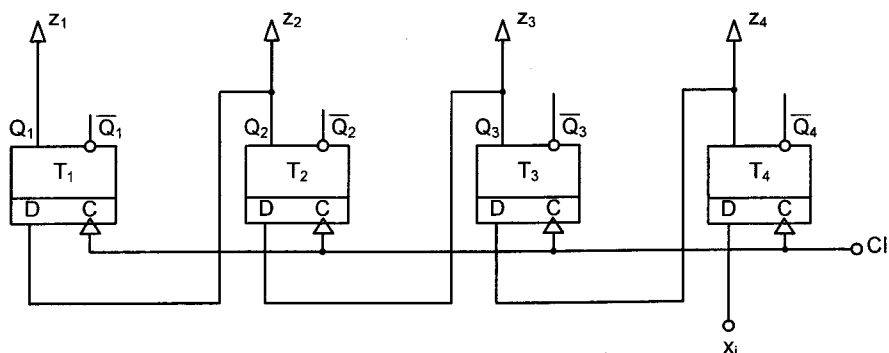
6.2.2 Синтез на регистри

При регистри с паралелен запис на информацията информационната дума, която се подава на входовете на тригерите, се записва в регистъра независимо от старото състояние, т.е. $Q_i^{t+1} = X_i^t$. Очевидно е, че за този начин на функциониране на автомата е най-подходящо да се използват тригери от тип D, за които, както е известно, $Q_i^{t+1} = D^t$. Информационната дума, подадена на входовете на D тригерите, ще се запише в тях в следващия момент от абстрактното автоматно време $D_i^t = X_i^t = Q_i^{t+1}$. За да определим функциите на възбуждане на тригерите, съставлящи обикновения регистър, бихме могли да използваме и класическия алгоритъм за синтез на КАП. Резултатът ще бъде същият: $D_i = X_i$. Ако решим да използваме R-S тригери, за да запишем 1 независимо от старото състояние, трябва на S входа да подадем 1, а на R входа - 0. За да запишем 0, трябва на S входа да подадем 0, а на R входа - 1. Иначе казано, $S_i = x_i, R_i = \bar{x}_i$. Ако разполагаме с J-K тригери, аналогично $J_i = x_i, K_i = \bar{x}_i$. Най-неподходящи за реализацията на регистър с паралелен запис на информацията са тригерите от тип T. При тях получените функции на възбуждане са твърде сложни: $T_i = x_i \oplus Q_i$.

Обикновено, но не задължително, регистърът е автомат на Мур, чиито изходи се вземат от правите изходи на тригерите.

Когато информационната дума се формира последователно във времето, за нейния запис и съхранение се използва регистър за последователен запис на информацията, т.е. преместващ регистър. Преместващите регистри имат един вход откъм най-младшия или най-старшия разряд и вериги на преместване наляво или надясно. Ако регистърът измества наляво, тригерът в i -тия разряд приема състоянието на съседния отляво младши тригер Q_{i-1} . Ако измества надясно, тригерът в i -тия разряд приема състоянието на съседния отляво старши тригер Q_{i+1} .

На фиг. 6.10 е показана структурната схема на 4-разряден преместващ наляво регистър на база D тригери. Очевидно, функциите на възбуждане на елементите памет имат вида $D_1 = Q_2, D_2 = Q_3, D_3 = Q_4, D_4 = x_1$. X_i е стойността на i -тия разряд на информационната дума, постъпващ към входа на най-младшия разряд на регистъра в момента t_i , $i=1,2,3,4$.



фиг. 6.10 Структурна схема на 4-разряден преместващ наляво регистър

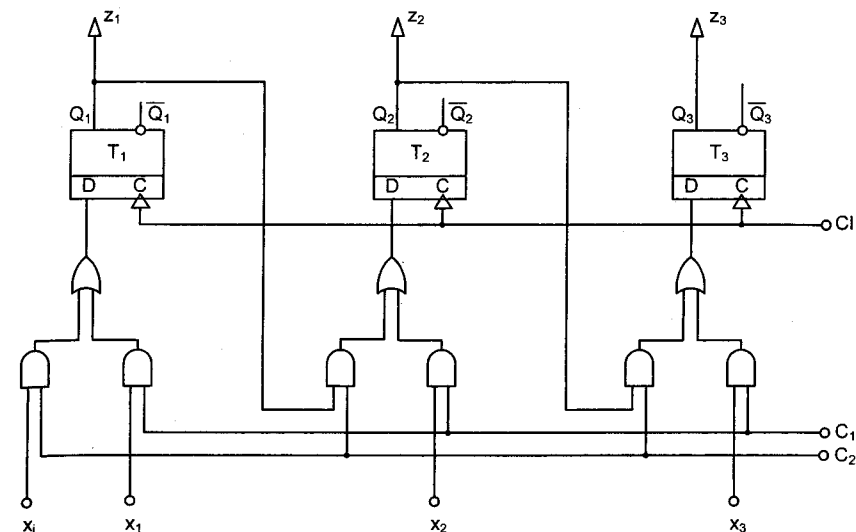
За реализацията на преместващ регистър трябва да се използват синхронни по фронт тригери или такива с двойна структура.

Ако е необходимо регистърът да изпълнява повече от една операции (напр. паралелен запис и преместване), те се извършват под действие на зададени управляващи сигнали, които не бива да са активни едновременно. Възбудителните функции на тригерите са дизюнкция от отделните възбудителни функции за всяка една от операциите. На фиг. 6.11 е показана схемата на 3-разряден регистър с паралелен и последователен запис на информацията. Последователният запис е откъм най-старшият разряд и преместването е надясно. Паралелният запис се извършва под действие на управляващия сигнал C_1 , а последователният – под действие на C_2 . C_1 и C_2 не бива едновременно да имат стойност 1, т.е. $C_1 \cdot C_2 = 0$. Тригерите са от тип D и възбудителните им функции имат следния вид:

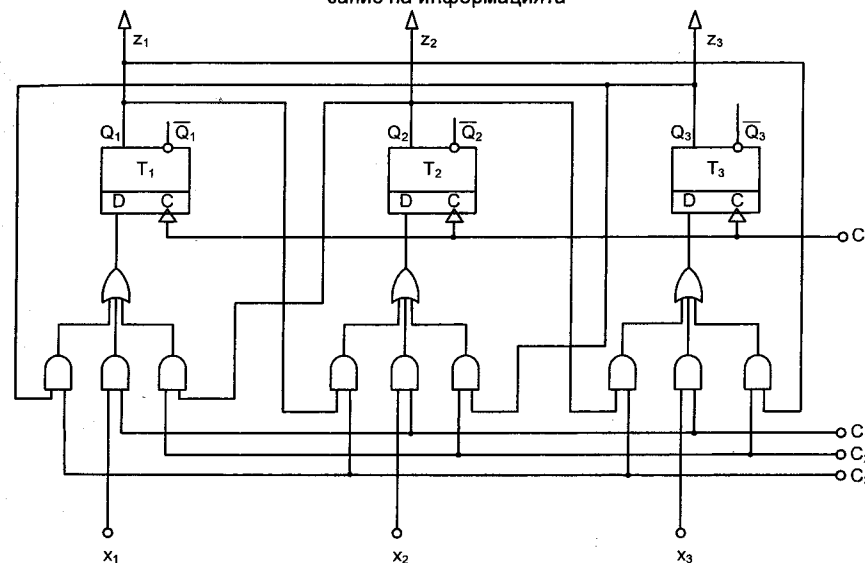
$$D_1 = C_1 \cdot x_1 \vee C_2 \cdot x_1 \quad D_2 = C_1 \cdot x_2 \vee C_2 \cdot Q_1 \quad D_3 = C_1 \cdot x_3 \vee C_2 \cdot Q_2$$

Регистър, който има възможност за преместване на информацията в двете посоки, се нарича реверсивен. Преместващ регистър, чийто най-старши (младши) разряд не се губи при преместването, а се подава на входа на най-младшия (старшия) тригер, се нарича цикличен. На фиг. 6.12 е показана схемата на 3-разряден цикличен реверсивен регистър с възможност за паралелен запис на информацията на база D тригери. C_1 е управляващ сигнал за паралелен запис, C_2 – за циклично изместване наляво, а C_3 – за циклично изместване надясно.

$$D_1 = C_1 \cdot x_1 \vee C_2 \cdot Q_2 \vee C_3 \cdot Q_3 \quad D_2 = C_1 \cdot x_2 \vee C_2 \cdot Q_3 \vee C_3 \cdot Q_1 \\ D_3 = C_1 \cdot x_3 \vee C_2 \cdot Q_1 \vee C_3 \cdot Q_2$$



Фиг. 6.11 Структурна схема на 3-разряден регистър с паралелен и последователен запис на информацията



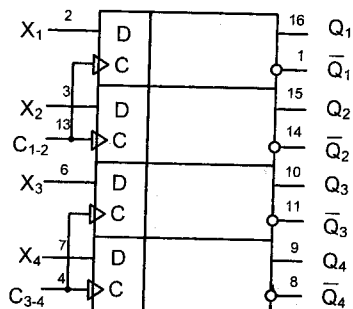
фиг. 6.12 Структурна схема на 3-разряден цикличен реверсивен регистър с възможност за паралелен запис на информацията

6.2.3 Регистри в интегрално изпълнение

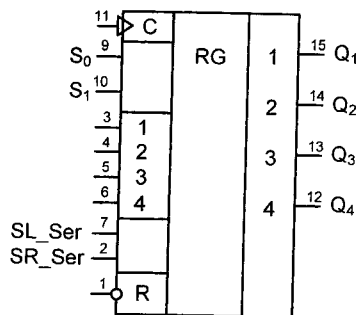
От регистрите в интегрално изпълнение ще разгледаме схемите 74LS75 и 74LS194. На фиг. 6.13 а) и 6.13 б) са показани изводите на двете схеми.

74LS75 представлява два двуразрядни паралелни регистра-памят (latch регистър). Изходът повтаря сигнала от входа D винаги, когато на тактовия вход C нивото е 1. Изходът остава в това състояние, в което е бил входът D в момента на превключване на сигнала C от 1 в 0.

74LS194 е синхронен реверсивен 4-разряден преместващ регистър. С е тактов вход. Тригерите превключват от положителния фронт на тактовия импулс. S_0 и S_1 са входове за управление на режима на работа. На фиг. 6.14 са показани режимите на работа при различните стойности на S_0 и S_1 . $X_1 + X_4$ са паралелни входове, а $Q_1 + Q_4$ са паралелни изходи. SL_Ser и SR_Ser са последователни входове при преместване отляво надясно и отдясно наляво. R е нулиращ вход с активно ниво 0. Регистърът работи при R = 1.



Фиг.6.13 а) Изводи на схема 74LS75



Фиг.6.13 б) Изводи на схема 74LS194

S_1	S_0	Режим
0	0	Забранен вход C
0	1	Преместване отляво надясно
1	0	Преместване отдясно наляво
1	1	Въвеждане от входовете $X_1 + X_4$

Фиг. 6.14 Режими на работа на схема 74LS194

Задание

1. Да се синтезира 3-разряден изваждащ брояч на база Т тригери както с паралелен, така и с текущ пренос.
2. Да се синтезира 3-разряден асинхронен изваждащ брояч на база Т тригери.
3. Да се синтезира реверсивен брояч до 11 на база J-K тригери.
4. Да се синтезира брояч, който брои в следната последователност: 001, 011, 101, 100, 001... При реализацията на логическата схема да се използват И-НЕ елементи и J-K тригери.
5. Да се построи делител на честота с коефициент на деление 7, като се използва структурната схема на брояча до 8. Елементите памет разполагат с вход за нулиране.
6. Да се синтезира 4-разряден паралелен регистър с общ за регистъра нулиращ вход. Да се използват тригери по избор.
7. Да се синтезира 3-разряден регистър за паралелен запис на информацията и с възможност за извеждане на правия или обратния код на записаното число. Да се работи с J-K тригери.
8. Да се синтезира 3-разряден регистър за последователен запис на информацията при изместване наляво на база D тригери. Регистърът да има възможност за поразрядно логическо сумиране на записаната в него информация и входна дума, постъпваща в паралелен код.
9. Да се синтезира триразряден регистър за последователен запис на информацията при изместване наляво на база D тригери. Регистърът да има възможност за поразрядно логическо умножение на записаната в него информация и входна дума, постъпваща в паралелен код.

Контролни въпроси

1. Какво означава коефициент (модул) на броене?
2. Какво означава реверсивен брояч?
3. Какво означава пълн и частичен брояч? Пълно или непълно определени са функциите на възбуждане на частичните броячи? Защо?
4. По какво се различават синхронните и асинхронните броячи?
5. Кой тип тригери са най-подходящи за реализация на броячи?
6. Кой вид броячи е най-бърз и кой е най-бавен? Защо?

7. За какво служи регистърът?
8. Кои са основните видове регистри?
9. Кой тип тригери са най-подходящи за реализация на регистри?
10. Какво означава реверсивен регистър? Какво означава цикличен регистър?
11. Могат ли да се използват тригери с единична структура при реализация на преместващ регистър? Защо?

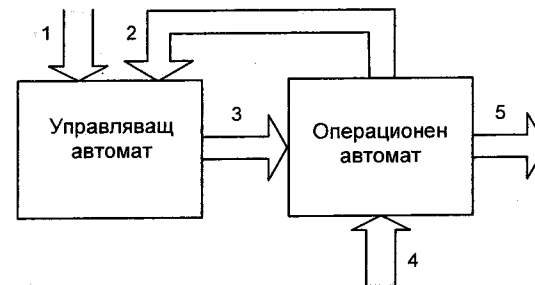
ТЕМА 7

СИНТЕЗ НА МИКРОПРОГРАМНИ АВТОМАТИ

7.1 Управляващи и операционни автомати

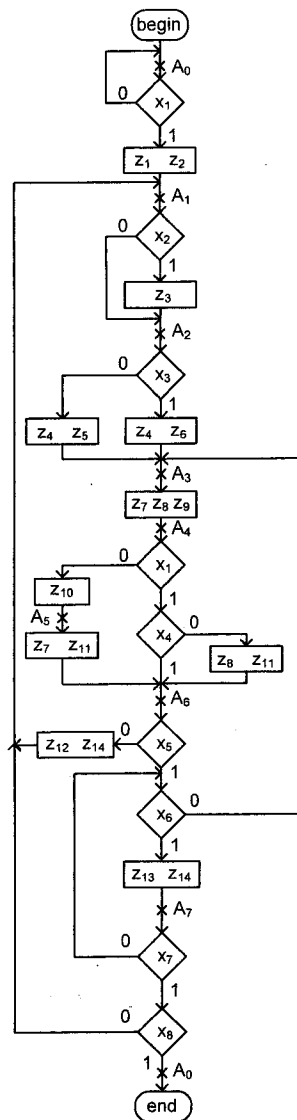
Основен принцип, който улеснява синтеза на цифровите устройства, е разделянето на устройството на две части: управляваща и управляема. Управляващата част се нарича управляващ автомат или управляващо устройство, а управляемата – операционна част или операционен автомат (фиг. 7.1)

Операционната част се състои от възли за елементарна обработка на информацията: регистри, броячи, суматори и пр. Запис на информация в регистър, нулиране на брояч или регистър, увеличаване или намаляване с единица съдържанието на брояч са примери на елементарни операции, които се извършват в един такт на абстрактното автоматно време. Такива операции се наричат микрооперации. Възможно е няколко микрооперации да се изпълняват в един и същи такт. Тогава те съставляват микрокоманда. За да бъде изпълнена, всяка аритметична и друга обработваща операция се разлага на последователност от микрооперации, които се изпълняват от операционната част. Управляващият автомат има за задача да зададе реда на тяхното изпълнение и по този начин да осигури изпълнението на дадената операция.



Фиг.7.1 Управляващ и операционен автомат и връзките между тях

- 1 – Шини, по които постъпва кодът на изпълняваната операция.
- 2 – Шини за обратна връзка между операционния и управляващия автомат.
- 3 – Шини за управляващи въздействия към операционния автомат.
- 4 – Шини за вход.
- 5 – Шини за изход.



Фиг. 7.2 Блок схема на алгоритъм на микропрограма

Редът на изпълнение на микрооперациите се определя от алгоритъма за изпълнение на съответната операция. Този ред, заедно с логическите условия, които го задават, се нарича микропрограма. Автоматът, който функционира в съответствие с микропрограмата, се нарича микропрограмен автомат, т.е. в контекста на разглежданата тематика микропрограмен и управляващ автомат са синоними. Задачата за синтез на микропрограмен автомат се свежда до задача за синтез на устройство, което функционира по алгоритъма на микропрограмата и има за входни въздействията, постъпващи по шините 1 и 2. Шина 2 показва състоянието на възлите в операционния автомат - например нулиране на брояч или препълване на регистър. В зависимост от тези състояния обработката може да продължи по различни клонове на алгоритъма. Изходни реакции на синтезираното устройство ще бъдат управляващите сигнали, генерирани по шина 3. Всеки управляващ сигнал съответства на една микрооперация, изпълнявана от операционния автомат.

7.2 Задаване и синтез на микропрограмни автомати

Микропрограмата се описва във формата на блок схема, като на всяка микрокоманда съответства операторен блок, а на всяко логическо условие – условен блок.

На фиг. 7.2 е показана примерна блокова схема на алгоритъм. С x_i са отбелязани логическите условия, а със z_i – микрооперациите. Преходът от блок схема на микропрограмата към начин на представяне удобен за синтез на автомат с памет продължава с определяне вътрешните състояния на автомата. Ако реализираме схемата по автоматен модел на Мур, на всеки операторен блок трябва да съпоставим изходна реакция. Броят на вътрешните състояния при

автоматен модел на Мили по принцип винаги е по-малък или равен на състоянията от функционално еквивалентен нему автомат на Мур. По тази причина ще определим вътрешните състояния като състояния от автомат на Мили. Автоматът ще бъде зададен с таблица на преходите и изходите по следната процедура:

№	Ас	Ан	Условие	Микро-операция
1	A ₀	A ₀ (000)	\bar{x}_1	-
2		A ₁ (001)	x_1	z_1, z_2
3	A ₁	A ₂ (010)	\bar{x}_2	-
4		A ₂ (010)	x_2	z_3
5	A ₂	A ₃ (011)	\bar{x}_3	z_4, z_5
6		A ₃ (011)	x_3	z_4, z_6
7	A ₃	A ₄ (100)	1	z_7, z_8, z_9
8	A ₄	A ₅ (101)	\bar{x}_1	z_{10}
9		A ₆ (110)	x_1, x_4	-
10		A ₆ (110)	x_1, \bar{x}_4	z_8, z_{11}
11	A ₅	A ₆ (110)	1	z_7, z_{11}
12	A ₆	A ₁ (001)	\bar{x}_5	z_{12}, z_{14}
13		A ₃ (011)	x_5, \bar{x}_6	-
14		A ₇ (111)	x_5, x_6	z_{13}, z_{14}
15	A ₇	A ₇ (111)	x_6, \bar{x}_7	z_{13}, z_{14}
16		A ₃ (011)	\bar{x}_6, \bar{x}_7	-
17		A ₁ (001)	x_7, \bar{x}_8	-
18		A ₀ (000)	x_7, x_8	-

Табл. 7.1

1. Изходът на блока „начало“ и входът на блока „край“ се отбелязват с едно и също състояние A₀.

2. Като е известно, при автомат на Мили изходните реакции се свързват с преходите между състоянията. Между две вътрешни състояния е възможно да има само една микрокоманда, т.е. само един операторен блок. Ето защо изходът на всеки операторен блок се отбелязва със съответното състояние A₁, A₂, ..., A_m.

3. Строи се таблица с 4 колони: колона за старо състояние, колона за ново състояние, колона за логическо условие, при което се извършва преходът (входно въздействие) и колона за микрооперациите, извършвани при дадения преход (изходна реакция).

4. Търси се път от едно състояние A_i до друго състояние A_j, като по този път да има най-много един операторен блок и произволен брой логически условия. За

всеки намерен път се записва един ред от таблицата.

Таблицата на преходите и изходите на микропрограмния автомат (МПА) от фиг. 7.2 е показана в табл. 7.1.

МПА се характеризират с голям брой входни и изходни променливи. Всеки преход обаче се определя от много малък брой от входните променливи – най-често една или две. Останалите са без значение за този преход. Поради това в колоната „логическо условие“ на таблицата се записва конюнкция само от тези входни променливи, които са от значение за съответния преход. Следователно МПА е една частична последователна схема, която е силно неопределена по вход. Тя е силно неопределена и по изходите, защото изходните шини са много, но в даден момент много малко от тях са активни едновременно, т.е. използва се малка част от възможните комбинации на изходните променливи. МПА е силно неопределен и по преходи – теоретично възможните преходи са много, а реалните – малко. Ето защо таблица на преходите и изходите от предложени тип е най-подходяща за описание на МПА. Тя съдържа описание само на определените преходи, поради което нейната големина е много по-малка в сравнение с таблица, построена по класическия алгоритъм за синтез на краен автомат с памет.

Синтезът на микропрограмния автомат продължава с кодиране на вътрешните състояния на автомата – табл. 7.2.

	Q ₁	Q ₂	Q ₃
A ₀	0	0	0
A ₁	0	0	1
A ₂	0	1	0
A ₃	0	1	1
A ₄	1	0	0
A ₅	1	0	1
A ₆	1	1	0
A ₇	1	1	1

Табл. 7.2 Кодиране на вътрешните състояния

Следващата стъпка е съставяне на произведения, съответстващи на всеки ред от ТПИ. Всяко произведение характеризира условието за извършване на прехода – автоматът да се намира в дадено Ас и да е налице определеното входно въздействие.

$$\begin{aligned}
 k_1 &= \bar{Q}_1 \bar{Q}_2 \bar{Q}_3 \bar{x}_1; & k_2 &= \bar{Q}_1 \bar{Q}_2 \bar{Q}_3 x_1; & k_3 &= \bar{Q}_1 \bar{Q}_2 \bar{Q}_3 \bar{x}_2; \\
 k_4 &= \bar{Q}_1 \bar{Q}_2 \bar{Q}_3 x_2; & k_5 &= \bar{Q}_1 \bar{Q}_2 \bar{Q}_3 \bar{x}_3; & k_6 &= \bar{Q}_1 \bar{Q}_2 \bar{Q}_3 x_3; \\
 k_7 &= \bar{Q}_1 \bar{Q}_2 \bar{Q}_3 \cdot 1; & k_8 &= \bar{Q}_1 \bar{Q}_2 \bar{Q}_3 \bar{x}_1; & k_9 &= \bar{Q}_1 \bar{Q}_2 \bar{Q}_3 x_1 x_4; \\
 k_{10} &= \bar{Q}_1 \bar{Q}_2 \bar{Q}_3 x_1 \bar{x}_4; & k_{11} &= \bar{Q}_1 \bar{Q}_2 \bar{Q}_3 \cdot 1; & k_{12} &= \bar{Q}_1 \bar{Q}_2 \bar{Q}_3 \bar{x}_5; \\
 k_{13} &= \bar{Q}_1 \bar{Q}_2 \bar{Q}_3 x_5 \bar{x}_6; & k_{14} &= \bar{Q}_1 \bar{Q}_2 \bar{Q}_3 x_5 x_6; & k_{15} &= \bar{Q}_1 \bar{Q}_2 \bar{Q}_3 x_6 \bar{x}_7; \\
 k_{16} &= \bar{Q}_1 \bar{Q}_2 \bar{Q}_3 \bar{x}_6 \bar{x}_7; & k_{17} &= \bar{Q}_1 \bar{Q}_2 \bar{Q}_3 x_7 \bar{x}_8; & k_{18} &= \bar{Q}_1 \bar{Q}_2 \bar{Q}_3 x_7 x_8
 \end{aligned}$$

Изходните реакции (микрооперациите) се издават в момента на прехода от Ас в А_н. Следователно всяка микрооперация е дизюнкция от конюнкциите, които обуславят съответния преход.

$$\begin{aligned}
 z_1 &= k_2; & z_2 &= k_2; & z_3 &= k_4; & z_4 &= k_5 \vee k_6; & z_5 &= k_5; & z_6 &= k_6; & z_7 &= k_7 \vee k_{11}; & z_8 &= k_7 \vee k_{10}; \\
 z_9 &= k_7; & z_{10} &= k_8; & z_{11} &= k_{10} \vee k_{11}; & z_{12} &= k_{12}; & z_{13} &= k_{14} \vee k_{15}; & z_{14} &= k_{12} \vee k_{14} \vee k_{15}
 \end{aligned}$$

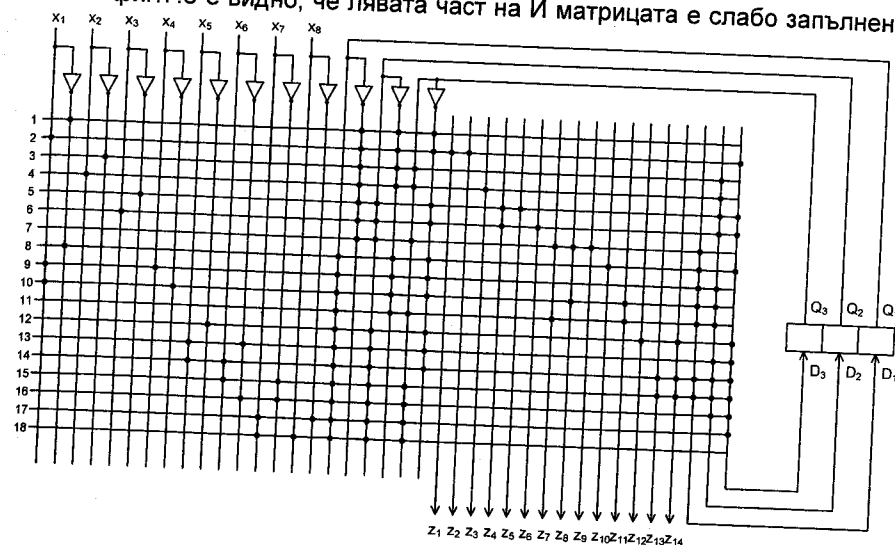
Ако приемем, че работим с D тригери, техните възбудителни функции ще бъдат дизюнкции от тези конюнкции, които трябва да установят тригера в състояние 1. За възбудителната функция на i-тия тригер ще вземем дизюнкция от конюнкциите, съответстващи на редовете, в които той има стойност 1 като съставна част от новото състояние.

$$\begin{aligned}
 D_1 &= k_7 \vee k_8 \vee k_9 \vee k_{10} \vee k_{11} \vee k_{14} \vee k_{15} \\
 D_2 &= k_3 \vee k_4 \vee k_5 \vee k_6 \vee k_9 \vee k_{10} \vee k_{11} \vee k_{13} \vee k_{14} \vee k_{15} \vee k_{16} \\
 D_3 &= k_2 \vee k_5 \vee k_6 \vee k_8 \vee k_{12} \vee k_{13} \vee k_{14} \vee k_{15} \vee k_{16} \vee k_{17}
 \end{aligned}$$

Представянето на условията за преход като конюнкция от състояние на автомата и логическо условие, а на изходните реакции и на възбудителните функции като дизюнкция от тези конюнкции, предполага реализация с ПЛМ. Минимизация на изходните и възбудителните функции обикновено не се търси, тъй като за да е възможно слепване, трябва да има повтарящи се логически условия, а това е малко вероятно. Матричната реализация за МПА от фиг. 7.2 е показана на фиг. 7.3

7.3 Замяна на входните променливи

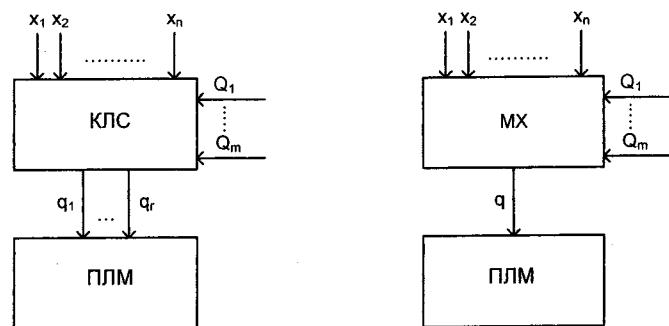
От фиг. 7.3 е видно, че лявата част на И матрицата е слабо запълнена.



Фиг. 7.3 Матрична реализация на МПА

Това е логично и валидно за всички МПА, тъй като условията, които участват за реализирането на даден преход, обикновено не са много. Лявата част на ИЛИ матрицата също е слабо запълнена, защото микрооперациите са дизюнкция от неговия брой конюнкции. Следователно ПЛМ е с голям брой входове, изходи и редове, но не се използва рационално. Така възниква идеята да се използва комбинационна логическа схема, която да превключва към входовете на ПЛМ само тези входове (логически условия), които са необходими за конкретния преход. Тази схема наподобява мултиплексор – ролята на адресни входове играят

изходите на тригерите, а информационни входи са логическите условия. По този начин вътрешните състояния ще управляват превключването на входовете към изходите на КЛС. За да има схемата само един изход, т.е. да бъде точно мултиплексор, е необходимо всеки преход да зависи от едно единствено условие. Ако в блок схемата на МПА има поредица от логически условия, те трябва да бъдат разкъсани с въвеждането на допълнителни вътрешни състояния след всеки условен блок. На фиг. 7.4 е показана схематично идеята за замяна на входните променливи.



Фиг. 7.4 Замяна на входните променливи на МПА

7.4 Декомпозиция на матриците на микропрограмен автомат

При голям брой входи и изходи на МПА е възможно да се окаже, че те са повече от входовете и изходите на ПЛМ, с които разполагаме. Едновременно с това видяхме, че левите части на И и ИЛИ матриците са слабо запълнени. Изход от проблема се търси в разделянето на входните променливи и микрооперациите на непресичащи се подмножества, което да позволи различните преходи и микрооперации да се реализират от различни подматрици. Такова разделяне се нарича декомпозиция на матриците на МПА. Декомпозиция може да се получи, ако се реализира следната процедура:

1. Съставят се подмножества от входните променливи, които се срещат в един ред от таблицата на преходите и изходите. Ако множеството на редовете на таблицата е $R = \{r_1, r_2, \dots, r_n\}$, множествата от входните променливи в един ред са $X(r_1), X(r_2), \dots, X(r_n)$, като $X(r_1) \cup X(r_2) \cup \dots \cup X(r_n) = X$.

2. Съставят се подмножества от микрооперации, които се срещат в един и същи ред в таблицата на преходите и изходите – $Z(r_1), Z(r_2), \dots, Z(r_n)$, като $Z(r_1) \cup Z(r_2) \cup \dots \cup Z(r_n) = Z$.

3. Съставя се граф с толкова върха, колкото реда има таблицата на преходите и изходите. Между върховете i и j има ненасочена дъга, ако редовете i и j имат поне една обща входна променлива или поне една обща микрооперация.

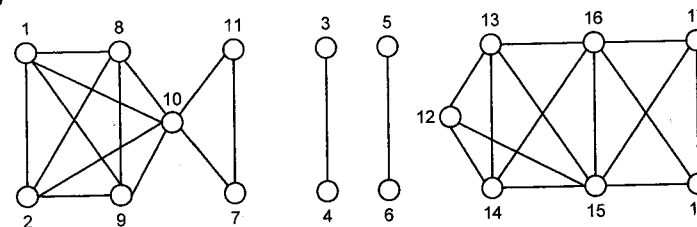
4. В така построения граф се търсят несвързани помежду си подграфи. Те определят разделянето на подмножества на редовете на таблицата на преходите и изходите и съответно на входните променливи и микрооперациите.

5. В съответствие с резултатите от стъпка 4 се разделят И и ИЛИ матриците.

Нека да приложим процедурата за табл. 7.1. След стъпка 1 и стъпка 2 се получават следните подмножества от входни и изходни променливи:

$X(r_1) = \{x_1\}; X(r_2) = \{x_1\}; X(r_3) = \{x_2\}; X(r_4) = \{x_2\}; X(r_5) = \{x_3\}; X(r_6) = \{x_3\}; X(r_7) = \Phi; X(r_8) = \{x_1\}; X(r_9) = \{x_1, x_4\}; X(r_{10}) = \{x_1, x_4\}; X(r_{11}) = \Phi; X(r_{12}) = \{x_5\}; X(r_{13}) = \{x_5, x_6\}; X(r_{14}) = \{x_5, x_6\}; X(r_{15}) = \{x_6, x_7\}; X(r_{16}) = \{x_6, x_7\}; X(r_{17}) = \{x_7, x_8\}; X(r_{18}) = \{x_7, x_8\}$

$Z(r_1) = \Phi; Z(r_2) = \{z_1, z_2\}; Z(r_3) = \Phi; Z(r_4) = \{z_3\}; Z(r_5) = \{z_4, z_5\}; Z(r_6) = \{z_4, z_6\}; Z(r_7) = \{z_7, z_8, z_9\}; Z(r_8) = \{z_{10}\}; Z(r_9) = \Phi; Z(r_{10}) = \{z_8, z_{11}\}; Z(r_{11}) = \{z_7, z_{11}\}; Z(r_{12}) = \{z_{12}, z_{14}\}; Z(r_{13}) = \Phi; Z(r_{14}) = \{z_{13}, z_{14}\}; Z(r_{15}) = \{z_{13}, z_{14}\}; Z(r_{16}) = \Phi; Z(r_{17}) = \Phi; Z(r_{18}) = \Phi$



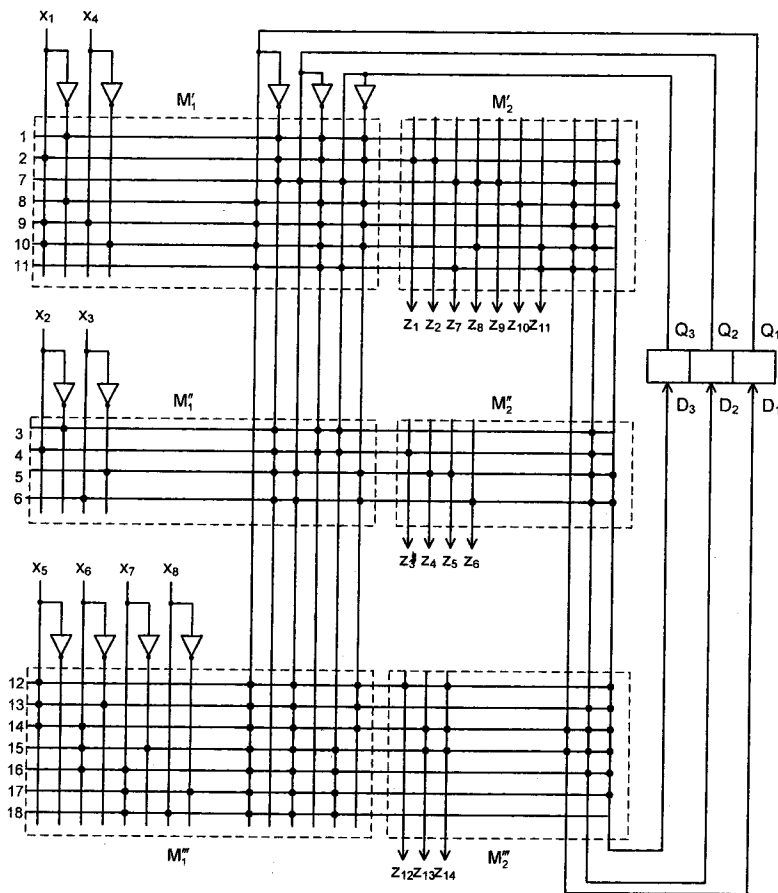
Фиг. 7.5

Въз основа на така определените подмножества се построява граф в съответствие със стъпка 3 от процедурата (фиг. 7.5.).

От построения граф може да се приеме следното разделяне на подмножества на множеството от редовете, входните променливи и микрооперациите:

$\{r_1, r_2, r_7, r_8, r_9, r_{10}, r_{11}\}; \{r_3, r_4, r_5, r_6\}; \{r_{12}, r_{13}, r_{14}, r_{15}, r_{16}, r_{17}, r_{18}\}$
 $\{x_1, x_4\}; \{x_2, x_3\}; \{x_5, x_6, x_7, x_8\}$
 $\{z_1, z_2, z_7, z_8, z_9, z_{10}, z_{11}\}; \{z_3, z_4, z_5, z_6\}; \{z_{12}, z_{13}, z_{14}\}$

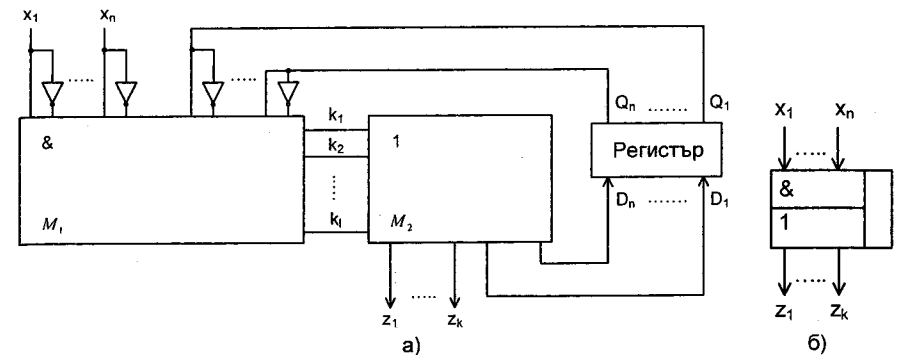
Като резултат от това разделяне на редовете на таблицата на преходите и изходите, на входните променливи и микрооперациите, матриците M_1 (И) и M_2 (ИЛИ) се декомпозират на три подматрици (фиг. 7.6.).



Фиг. 7.6 Декомпозиция на матриците на МПА

7.5 Декомпозиция на микропрограмни автомати

МПА е удобно да бъдат реализирани чрез ПЛМ с памет. ПЛМ с памет съдържа освен двете матрици – И-матрица и ИЛИ-матрица, и регистър в обратната връзка между матриците M_1 и M_2 . На фиг. 7.7 а) и б) са показани вътрешната структура и условното означение на такава ПЛМ.



Фиг. 7.7 Вътрешна структура и условно означение на ПЛМ с памет

Ако броят на микрооперациите на МПА е по-голям от броя на изходите z на ПЛМ с памет, необходимо е да се използват няколко ПЛМ с памет паралелно свързани входове. Съдържанието на матрицата M_1 на тези ПЛМ е идентично. Идентично е и съдържанието на дясната част на M_2 , т.е. във всички ПЛМ ще се извършват еднакви преходи. Различни ще бъдат само микрооперациите, т.е. лявата част на матриците M_2 .

Ако броят на входовете или броят на редовете в таблицата на преходите и изходите на МПА е по-голям от броя на входовете и броя на конюнкциите на ПЛМ с памет, се налага декомпозиция на МПА на по-прости МПА, които могат да се реализират на една ПЛМ с памет. При декомпозиция от последователен тип във всеки момент от функционирането на микропрограмния автомат само един от съставните МПА ще бъде активен и ще издава съответните микрооперации. Останалите ще бъдат в пасивно състояние. За да се управляват преходите от активно в пасивно състояние се въвеждат допълнителни управляващи сигнали.

Декомпозицията на МПА на съставни МПА се реализира, след като е избрано разделяне на множеството от вътрешни състояния A на подмножества, чийто брой определя и броя на съставните МПА. Процедурата за това разделяне е следната:

1. Образуват се подмножества от микрооперации за всяко състояние на МПА – $Z(A_i)$, като $Z(A_0) \cup Z(A_1) \cup \dots \cup Z(A_m) = Z$. „ m “ е броят вътрешни състояния.
2. Образуват се подмножества от входни променливи за всяко състояние на МПА – $X(A_i)$, като $X(A_0) \cup X(A_1) \cup \dots \cup X(A_m) = X$.
3. Строи се граф с върхове, означени със състоянията на МПА. Между върховете A_i и A_j има дъга, ако $X(A_i) \cap X(A_j) \neq \Phi$ или ако $Z(A_i) \cap Z(A_j) \neq \Phi$, т.е.

състоянията A_i и A_j имат поне една обща входна променлива или поне една обща микрооперация.

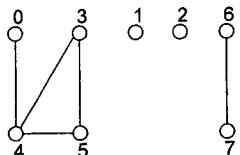
4. В така построения граф се търсят несвързани помежду си подграфи. Те определят разделянето на множеството A на подмножества от вътрешни състояния. На всяко подмножество съответства един от съставните МПА. За всеки съставен МПА се строи отделна таблица на преходите и изходите.

Изложеният алгоритъм, приложен към МПА, зададен с табл.7.1, дава следните резултати:

$X(A_0)=\{x_1\}$; $X(A_1)=\{x_2\}$; $X(A_2)=\{x_3\}$; $X(A_3)=\Phi$; $X(A_4)=\{x_1, x_4\}$; $X(A_5)=\Phi$;
 $X(A_6)=\{x_5, x_6\}$; $X(A_7)=\{x_6, x_7\}$

$Z(A_0)=\{z_1, z_2\}$; $Z(A_1)=\{z_3\}$; $Z(A_2)=\{z_4, z_5, z_6\}$; $Z(A_3)=\{z_7, z_8, z_9\}$; $Z(A_4)=\{z_8, z_{10}, z_{11}\}$;
 $Z(A_5)=\{z_7, z_{11}\}$; $Z(A_6)=\{z_{12}, z_{13}, z_{14}\}$; $Z(A_7)=\{z_{13}, z_{14}\}$

Построеният граф е показан на фиг. 7.8.



Фиг. 7.8 Граф, описващ резултата от декомпозицията на автомата

Несвързаните помежду си подграфи задават разделянето на множеството вътрешни състояния на 3 подмножества: $\{A_0, A_3, A_4, A_5\}$; $\{A_1, A_2\}$; $\{A_6, A_7\}$. Към множеството от състояния на всеки автомат добавяме и пасивно състояние b_i , където i е номер на автомата:

$\{A_0, A_3, A_4, A_5, B_1\}$; $\{A_1, A_2, B_2\}$; $\{A_6, A_7, B_3\}$.

Пасивните състояния ще се използват в случаите, когато автоматът трябва да премине от състояние, което принадлежи на един от съставните МПА, към състояние, което принадлежи на друг съставен МПА. За да се осъществи такъв преход, е необходимо първият автомат да премине в пасивно състояние, като издаде съответна изходна реакция, която в ролята си на входно въздействие да накара втория автомат да премине от пасивно в необходимото състояние. Преходът от състояние A_p , което принадлежи на автомат i , към състояние A_q , принадлежащо на автомат с номер j , се осъществява с два прехода: $A_p - B_i$ и $B_j - A_q$. Първият преход ще бъде съпроводен с издаване на допълнителна микрооперация z_{dq} , която от своя страна ще бъде необходимото входно въздействие за извършване на втория преход. Тази идея е залегнала в построяването на таблиците на преходите и изходите на съставните автомати (фиг.7.9).

След построяването на автоматните таблици, се построяват матриците M_1 и M_2 на съставните МПА, като при кодирането на вътрешните състояния, кодът 00...0 се използва за състоянието B_i . На фиг. 7.10 е показано кодирането на вътрешните състояния и матричната реализация на един от съставните МПА.

Накрая се построява микропрограмният автомат, като изходите z_{di} се свързват с едноименните входове в другите съставни МПА. Изпълнението на тази процедура води до построяването на МПА, състоящ се от съставни МПА, работещи последователно. Началното състояние на всеки съставен МПА е B_i , с изключение на този, съдържащ състоянието A_0 . За него началното състояние е A_0 . В произволен момент от време всички съставни МПА се намират в състояние B_i , а само един (активният в момента) – в някое от работните си състояния.

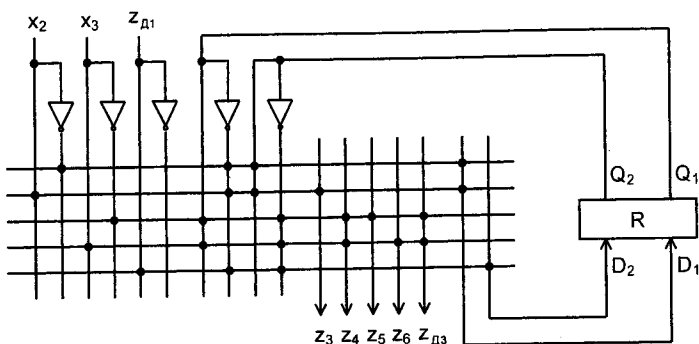
№	Ас	Ан	Условие	Микро-операция
1	A_0	A_0	\bar{x}_1	-
2	A_0	B_1	x_1	$z_1 z_2 z_{d1}$
7	A_3	A_4	1	$z_7 z_8 z_9$
8	A_4	A_5	\bar{x}_1	z_{10}
9	A_4	B_1	$x_1 x_4$	z_{d6}
10	A_4	B_1	$x_1 \bar{x}_4$	$z_8 z_{11} z_{d6}$
11	A_5	B_1	1	$z_7 z_{11} z_{d6}$
	B_1	A_3	z_{d3}	-
	B_1	A_0	z_{d0}	-

№	Ас	Ан	Условие	Микро-операция
3	A_1	A_2	\bar{x}_2	-
4	A_1	A_2	x_2	z_3
5	A_2	B_2	\bar{x}_3	$z_4 z_5 z_{d3}$
6	A_2	B_2	x_3	$z_4 z_6 z_{d3}$
	B_2	A_1	z_{d1}	-

№	Ас	Ан	Условие	Микро-операция
12	A_6	B_3	\bar{x}_5	$z_{12} z_{14} z_{d1}$
13	A_6	B_3	$x_5 \bar{x}_6$	z_{d3}
14	A_6	A_7	$x_5 x_6$	$z_{13} z_{14}$
15	A_7	A_7	$x_6 \bar{x}_7$	$z_{13} z_{14}$
16	A_7	B_3	$\bar{x}_6 \bar{x}_7$	z_{d3}
17	A_7	B_3	$x_7 \bar{x}_8$	z_{d1}
18	A_7	B_3	$x_7 x_8$	z_{d0}
	B_3	A_6	z_{d6}	-

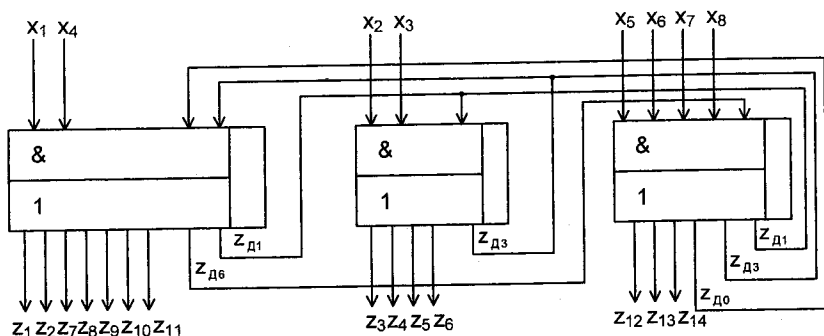
Фиг. 7.9 Таблицы на преходите и изходите на компонентните автомати

	Q_1	Q_2
A_1	0	1
A_2	1	0
B_2	0	0



Фиг. 7.10 Матрична реализация на втория от съставните МПА

Окончателната схема на декомпозиция МПА е показана на фиг. 7.11



Фиг. 7.11 МПА, декомпозиран на три съставни автомата и реализиран с ПЛМ с памет

Задание

1. Да се синтезира МПА по зададена блок схема на алгоритъма.
2. Да се реализира схема на автомата, като получените възбудителни и изходни функции се реализират чрез ПЛМ с необходимата размерност.
3. Да се направи декомпозиция на матриците M_1 и M_2 на получения автомат, така че да се използват ПЛМ с предварително зададена размерност.

4. Да се синтезира МПА, като се използват ПЛМ с памет със зададена размерност.

Контролни въпроси

1. Какво е операционен и какво управляващ автомат?
2. Какво е микрооперация, микрокоманда и микропрограма?
3. По какво микропрограмните автомати се различават от досега разгледаните ПС?
4. Каква е процедурата за преминаване от блокова схема на алгоритъма към таблица на преходите и изходите на МПА?
5. По какво таблицата на преходите и изходите на микропрограмните автомати се различава от таблицата на преходите и изходите на ПС?
6. Какви недостатъци има матричната реализация на микропрограмните автомати и на какво се дължат те?
7. Какви особености на МПА определят възможността за декомпозиция на матриците на ПЛМ в матричната реализация на МПА?
8. От какви съображения се определя броят на ПЛМ, с които ще се реализира МПА при използване на декомпозиция на матриците?
9. Какво представлява ПЛМ с памет?
10. Какъв подход се използва, ако броят на микрооперациите на МПА е по-голям от броя на редовете или от броя на изходите на ПЛМ с памет?
11. Какъв подход се използва, ако броят на преходите или броят на променливите на МПА е по-голям от броя на редовете или от броя на променливите на ПЛМ с памет?
12. Как се определя броят на съставните МПА, на които се декомпозира зададеният МПА?
13. Как се определят множествата от вътрешните състояния на съставните МПА?
14. Как се определят функциите на преходите и изходите на съставните МПА?
15. В какви вътрешни състояния се намират в даден момент от време съставните МПА?

Приложение
Карты на Вейч за 2,3,4,5,6,7 променливи

	x_1	
x_2	3	1
	2	0

	x_1			
x_2	6	7	3	2
	4	5	1	0

	x_1			
x_2	12	14	6	4
	13	15	7	5
	9	11	3	1
	8	10	2	0

	x_1				x_2			
x_3	28	30	22	20	12	14	6	4
	29	31	23	21	13	15	7	5
	25	27	19	17	9	11	3	1
	24	26	18	16	8	10	2	0

	x_1				x_3			
x_4	60	62	54	52	28	30	22	20
	61	63	55	53	29	31	23	21
	57	59	51	49	25	27	19	17
	56	58	50	48	24	26	18	16
x_2	44	46	38	36	12	14	6	4
	45	47	39	37	13	15	7	5
	41	43	35	33	9	11	3	1
	40	42	34	32	8	10	2	0

	x_1								x_3							
x_5	108	110	102	100	124	126	118	116	60	62	54	52	44	46	38	36
	109	111	103	101	125	127	119	117	61	63	55	53	45	47	39	37
	105	107	99	97	121	123	115	113	57	59	51	49	41	43	35	33
	104	106	98	96	120	122	114	112	56	58	50	48	40	42	34	32
x_2	76	78	70	68	92	94	86	84	28	30	22	20	12	14	6	4
	77	79	71	69	93	95	87	85	29	31	23	21	13	15	7	5
	73	75	67	65	89	91	83	81	25	27	19	17	9	11	3	1
	72	74	66	64	88	90	82	80	24	26	18	16	8	10	2	0

ЛИТЕРАТУРА

1. Фридман, А., П. Менон, Теория и проектирование переключательных схем, Мир, Москва, 1978.
2. Даковски, Л., Логически основи на ЦЕИМ, Техника, София, 1978.
3. Тодорова, С., Й. Русева, Д. Григорова, Синтез и анализ на логически схеми, Издателство на Русенски университет „А. Кънчев“, 1998.
4. Даковски, Л., Н. Николов, Ръководство по логика и програмируеми автомати, Техника, 1990.
4. Балканджиев, Л., Е. Пандов, Д. Манова, Анализ и синтез на логически схеми – ръководство за лабораторни упражнения, Издателство на ТУ – София, 2001.
5. Иванов, С., Ю. Петкова, Анализ и синтез на логически схеми, Издателство на ТУ – Варна.
6. Кисъов, В. Т., Теория на крайните автомати, Техника, 1976.
7. Floyd, T. L., Digital Fundamentals, Pearson Higher Education, 2009.

Уеб сайтове:

1. <http://scitec.uwichill.edu.bb/cmp/online/P10F/boolean.htm>
2. <http://www.asic-world.com/digital/tutorial.html>
3. http://www.play-hookey.com/digital/boolean_algebra.html
4. <http://www.ee.surrey.ac.uk/Projects/Labview/index.html>
5. http://www.cs.usask.ca/content/resources/tutorials/csconcepts/2000_8/fra medIndex.html
6. <http://www.ee.surrey.ac.uk/Projects/Labview/common/glossary.html#STD>

АНАЛИЗ И СИНТЕЗ НА ЛОГИЧЕСКИ СХЕМИ

Автори:

- © гл. ас. Диана Григорова
- © доц. д-р Валентин Моллов

Рецензент:

- © доц. д-р Николай Николов

Стилов редактор:

- © Стояна Саева

Даден за печат: м. ноември 2009 г.

Излязъл от печат: м. ноември 2009 г.

Печатни коли 7.25

Поръчка № 32 с

Тираж 100 броя

Цена 7.00 лв.

Формат 60/84/16

ISBN : 978-954-438-781-5

Издателство на Техническия университет - София