ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1975

Seria: AUTOMATYKA z. 30 Nr kol. 447

Marian Budka

Instytut Automatyki Przemysłowej
i Pomiarów

MOŻLIWOŚCI REALIZACJI FUNKCJI LOGICZNYCH PRZY POMOCY WYBRANYCH UKŁADÓW SCALONYCH ŚREDNIEJ I DUŻEJ SKALI INTEGRACJI

Streszczenie. W artykule opisano niektóre problemy realizacji automatów cyfrowych z wybranych obwodów scalonych, średniej i dużej skali integracji. Przedstawiono wnioski dotyczące projektowania układów logicznych, realizowanych w oparciu o multipleksery pamięci ROM i uniwersalne zestawy logiczne.

1. Wstęp

Różnorodność cyfrowych układów scalonych małej, średniej i dużej skali integracji ułatwia realizację układów logicznych zapewniając równocześnie ich dużą pewność działania, małe gabaryty i stosunkowo niski koszt.

Dysponując tak szerokim asortymentem, zarówno pojedynczych elementów jak i typowych układów, projektant automatów cyfrowych ma w dużym stopniu ułatwione zadanie.

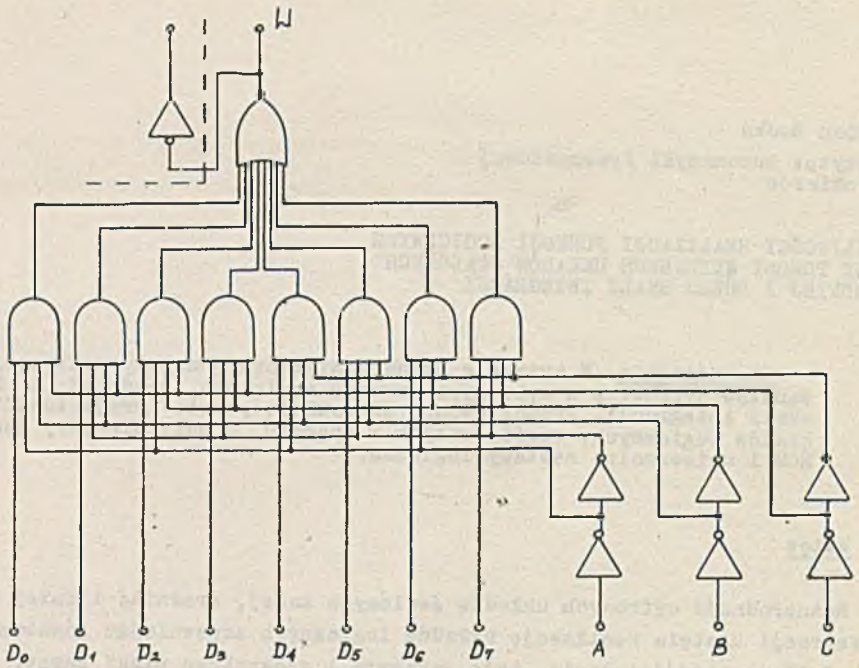
Ze wzrostem skali integracji cyfrowych obwodów scalonych funkcję automatu, realizowanego tradycyjnie z szeregu pojedynczych elementów(bramek i przerzutników), spełnia jeden wyspecjalizowany układ a synteza automatu sprowadza się do określenia tablic zależności i odpowiedniego zaprogramowania tegoż wyspecjalizowanego układu.

Do realizacji funkcji logicznych szczególnie przydatne są multipleksery, należące do układów scalonych średniej (MSI) skali oraz pamięci ROM i programowane zestawy logiczne (PLA) należące do układów scalonych dużej skali (LSI).

2. Projektowanie układów logicznych z zastosowaniem multiplekserów

Opanowanie na skalę masową produkcji obwodów scalonych średniej skali integracji (MSI) umożliwia stosowanie podzespołów, takich jak liczniki, rejestry, dekodery czy multipleksery do realizacji układów logicznych.

Rozpatrzmy przykładowo multiplekser 8-bitowy bez wejścia strobującego (74 152), którego schemat przedstawia rys. 1.



Rys. 1

Funkcja logiczna określająca stan wyjścia takiego układu jest następująca:

$$\bar{W} = \bar{A}\bar{B}\bar{C}\,D_0 + A\bar{B}\bar{C}\,D_1 + \bar{A}B\bar{C}\,D_2 + AB\bar{C}\,D_3 + \bar{A}\bar{B}C\,D_4 +$$

$$+ A\bar{B}C\,D_5 + \bar{A}BC\,D_6 + ABC\,D_7 \qquad (2.1)$$

Analogicznie dla multipleksera 16-bitowego, z wejściem strobującym S, funkcja logiczna określająca stan wyjścia jest podana zależnością (2.2).

$$\bar{W} = \bar{S}\,(\bar{A}\bar{B}\bar{C}\bar{D}E_0 + A\bar{B}\bar{C}\bar{D}E_1 + \bar{A}B\bar{C}\bar{D}E_2 + AB\bar{C}\bar{D}E_3 + \bar{A}\bar{B}C\bar{D}E_4 +$$

$$+ A\bar{B}C\bar{D}E_5 + \bar{A}BC\bar{D}E_6 + ABC\bar{D}E_7 + \bar{A}\bar{B}\bar{C}DE_8 + A\bar{B}\bar{C}DE_9 +$$

$$+ \bar{A}B\bar{C}DE_{10} + AB\bar{C}DE_{11} + \bar{A}\bar{B}CDE_{12} + A\bar{B}CDE_{13} + \bar{A}BCDE_{14} + ABCDE_{15}) \qquad (2.2)$$

Zarówno wyrażenie (2.1) jak i (2.2) jest zanegowaną normalną postacią jedynki, tzn. zawiera wszystkie składniki jedynki, kombinacje sygnałów wej-

ściowych zanegowanych i niezanegowanych odpowiednio (2.2) dla czterech wejść A, B, C i D oraz (2.1) dla trzech wejść A, B i C.

Projektowanie układów logicznych na bazie multiplekserów sprowadza się do wyrażenia funkcji logicznej w normalnej postaci sumy i wyboru implikantów przynależnych tej funkcji. Techniczna realizacja układu polega na wyeliminowaniu implikantów zbędnych poprzez bramkowanie sygnałami $E_i = 0$ (2.2) lub $D_i = 0$ (2.1) i pozostawieniu implikantów wchodzących do normalnej postaci sumy funkcji poprzez podanie na odpowiednie wejścia E_k(2.2) lub D_k (2.1) sygnałów logicznych "1".

Sposób bramkowania sygnałami E_i lub D_i można określić przy pomocy siatek Karnaugha. Dla 3 i 4 zmiennych takie siatki są przedstawione na rys. 2.

a)

C \ BA	00	01	11	10
0	0 D_0	1 D_1	3 D_3	2 D_2
1	4 D_4	5 D_5	7 D_7	6 D_6

b)

DC \ BA	00	01	11	10
00	E_0	E_1	E_3	E_2
01	E_4	E_5	E_7	E_6
11	E_{12}	E_{13}	E_{15}	E_{14}
10	E_8	E_9	E_{11}	E_{10}

Rys. 2

Przykład

Należy zrealizować przy pomocy multipleksera funkcję logiczną:

$$F = \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C$$

C \ BA	00	01	11	10
0	1	0	0	1
1	0	1	0	0

Rys. 3

Funkcja F posiada 3 argumenty, zatem zrealizujemy ją na bazie 8-bitowego multipleksera SN54/74151 lub 74152. Siatka zależności dla funkcji F jest przedstawiona na rys. 3.

Stąd sposób bramkowania układu określa zależność:

$$D_0 = D_2 = D_5 = 1$$

$$D_1 = D_3 = D_4 = D_6 = D_7 = 0$$

Inny sposób określania bramkowania polega na zastosowaniu zapisu numerycznego normalnej postaci sumy np.:

$$F = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}C = \sum(0, 2, 5)_{CBA}$$

$$D_0 = D_2 = D_5 = 1$$

i stąd

$$D_1 = D_3 = D_4 = D_6 = D_7 = 0$$

W przypadku, gdy liczba argumentów (wejść) jest większa od $\log_2 N$ gdzie N oznacza liczbę bitów multipleksera, należy rozszerzyć wyrażenie logiczne tak, aby każdy składnik był mnożony przez sygnał, jego negację lub dowolne kombinacje sygnałów i ich negacji odpowiadających argumentom dodatkowym. W dalszym etapie należy określić siatki Karnaugha dla wszystkich kombinacji stanów wejść dodatkowych po czym uwzględniając, że implikanty otrzymane z poszczególnych siatek Karnaugha muszą być mnożone przez odpowiadające im kombinacje sygnałów dodatkowych oraz że sygnał wyjściowy jest sumą logiczną składowych odpowiadających wszystkim kombinacjom sygnałów dodatkowych, określić sposób bramkowania.

Sposób ten ilustruje przykład [7].

Przykład

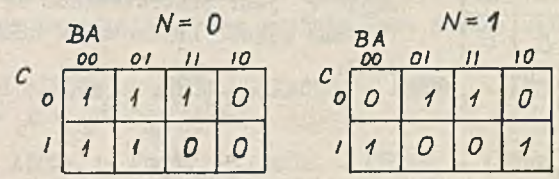
Zrealizować w oparciu o 8-bitowy multiplekser funkcję logiczną: $P = A\bar{C} + \bar{A}CN + \bar{B}\bar{N}$. Potraktujmy argumenty A, B i C jako podstawowe, zaś N jako argument dodatkowy. W związku z tym należy rozszerzyć w myśl powyższych zasad implikant $A\bar{C}$, który nie jest mnożony przez N lub $\bar{N}$.

$$A\bar{C}\,(N + \bar{N}) = A\bar{C}N + A\bar{C}\bar{N}$$

Stąd

$$F = A\bar{C}N + A\bar{C}\bar{N} + \bar{A}CN + \bar{B}\bar{N}$$

Utwórzmy dwie siatki Karnaugha, oddzielnie dla $N = 0$ i $N = 1$.



Rys. 4

Sposób bramkowania określić można przy pomocy następującej tabeli:

Wejścia bramkujące	N = 0	N = 1	Wyjście
D_0	1 $\bar{N}=\bar{N}$	0 N=0	$\bar{N}+0=\bar{N}$
D_1	1 $\bar{N}=\bar{N}$	1 N=N	$\bar{N}+N=1$
D_2	0 $\bar{N}=0$	0 N=0	0+0=0
D_3	1 $\bar{N}=\bar{N}$	1 N=N	$\bar{N}+N=1$
D_4	1 $\bar{N}=\bar{N}$	1 N=N	$\bar{N}+N=1$
D_5	1 $\bar{N}=\bar{N}$	0 N=0	$\bar{N}+0=\bar{N}$
D_6	0 $\bar{N}=0$	1 N=N	0+N=N
D_7	0 $\bar{N}=0$	0 N=0	0+0=0

W tabeli wykorzystano zależność, że: $F = \bar{N} F_0 + N F_1$, gdzie F_0 i F_1 są to wartości funkcji logicznej F odpowiednio dla N = 0 i N = 1.

Stąd bramkowanie jest następujące:

$$D_0 = D_5 = \bar{N}$$

$$D_1 = D_3 = D_4 = 1$$

$$D_2 = D_7 = 0$$

$$D_6 = N$$

Odnośnie zastosowań multiplekserów do realizacji funkcji logicznych można sformułować następujące wnioski:

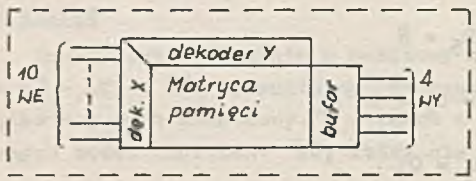
- na bazie multiplekserów można zrealizować dowolny układ kombinacyjny; w przypadku dużej liczby argumentów do bramkowania mogą być potrzebne sygnały realizowane przez dodatkowe, proste układy logiczne,
- pewność działania takich układów, w szczególności bez konieczności budowy zespołów współpracujących z dodatkowych elementów, jest duża i praktycznie jest równa niezawodności samego multipleksera,
- sposób projektowania i techniczna realizacja układu są bardzo proste; odpadają w zasadzie wszystkie problemy wiążące się z uruchomieniem systemu logicznego,
- system logiczny zrealizowany przy pomocy multiplekserów może współpracować z blokami pamięci w przypadku synchronicznej pracy układu,
- problem minimalizacji funkcji logicznych realizowanych na multiplekserach w zasadzie nie istnieje.

3. Realizacja funkcji logicznych przy pomocy układów scalonych dużej skali integracji

3.1. Zastosowanie pamięci ROM do realizacji układów logicznych

Standartowe rozwiązanie pamięci ROM (Read-Only Memory) zapewniały realizację określonych operacji logicznych czy arytmetycznych w zależności od ich struktur uzyskiwanych w procesie technologicznym. W tej sytuacji koszty układów produkowanych masowo były niskie. Stan taki prowadził jednak do ograniczenia zastosowania pamięci ROM prawie wyłącznie do typowych bloków przeliczających. Realizacja programowanej pamięci ROM poszerzyła znacznie zakres zastosowań tych układów.

W procesie wytwarzania takich pamięci ROM uzyskuje się podstawową strukturę na bazie tranzystorów, zaś proces programowania pamięci polega na odpowiednim ukształtowaniu tej struktury poprzez oddziaływanie zewnętrznych czynników po zakończeniu procesu technologicznego. Schemat funkcjonalny pamięci ROM 1024 x 4 przedstawia rys. 5.



Rys. 5

Obok licznych zastosowań pamięci ROM mogą służyć do realizacji układów logicznych, zarówno kombinacyjnych jak i sekwencyjnych.

Projektowanie układów kombinacyjnych na bazie pamięci ROM polega na określeniu tablicy zależności automatu a techniczna realizacja automatu sprowadza się do zaprogramowania pamięci.

Programowanie polega na zmianie struktury pamięci przy pomocy zewnętrznego źródła napięcia o regulowanej wartości.

Sposób postępowania wyjaśnia przykład.

Przykład

Zrealizować na bazie pamięci ROM automat o czterech wejściach A, B, C, D i 4 wyjściach F_1, F_2, F_3 i F_4.

$$F_1 = B\bar{C}\bar{D} + A\bar{C}\bar{D} + AB\bar{C}D + A\bar{B}CD + \bar{A}BC\bar{D}$$

$$F_2 = \bar{A}\bar{B}CD + \bar{A}C\bar{D} + ABC\bar{D} + A\bar{C}D + A\bar{B}\bar{C}\bar{D}$$

$$F_3 = AB + AC + BC + \bar{A}\bar{C}D$$

$$F_4 = \bar{A}B\bar{C}D + \bar{A}BC\bar{D} + \bar{A}C\bar{D} + \bar{B}C + ABCD$$

W przypadku tradycyjnego sposobu realizacji automatu, o wyjściach F_1, F_2, F_3 i F_4 do budowy układu, potrzeba bramek o łącznej liczbie wejść powyżej 60.

Rozwiązanie: Na podstawie wyrażeń F_1, F_2, F_3 i F_4 tworzymy, posługując się siatkami Karnaugha lub obliczając wartości tych funkcji dla wszystkich kombinacji sygnałów wejściowych, tablicę zależności.

	ADRES				WYJŚCIA			
	A	B	C	D	F_1	F_2	F_3	F_4
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	1	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	0	1
4	0	1	0	0	1	0	0	0
5	0	1	0	1	0	0	1	1
6	0	1	1	0	1	1	1	1
7	0	1	1	1	0	0	1	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	0	1	0	0
10	1	0	1	0	0	0	1	1
11	1	0	1	1	1	0	1	1
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	1	1	0
14	1	1	1	0	0	1	1	0
15	1	1	1	1	0	0	1	0

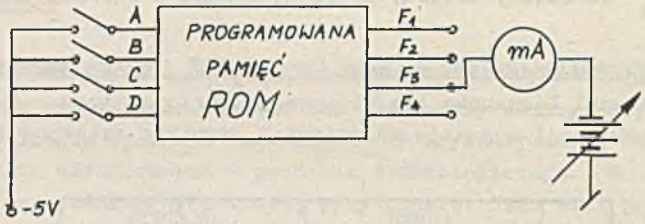
Siatka podstawowa programowanej pamięci ROM zapewnia $F_1 = F_2 = F_3 = F_4 = 0$ dla wszystkich kombinacji sygnałów wejściowych. Zatem programowanie obejmie jedynie te pozycje, w których $F_i = 1$.

W naszym przypadku programowanie rozpoczynamy od drugiego wiersza tablicy.

Wybieramy adres 0 0 0 1, tzn. na wejścia A, B i C podajemy napięcie - 5 V (zero logiczne) zaś wejście D pozostaje otwarte (jedynka logiczna) por. rys. 6.

Na wyjście F_3 podłączamy źródło napięcia o regulowanej wartości (rys. 6) w szereg z miliamperomierzem.

Stopniowo zwiększamy wartość ujemnego napięcia, aż prąd I wzrośnie do wartości około 50 mA, wówczas odpowiednie złącze w podstawowej siatce pamięci zostanie przepalone i wartość prądu spadnie do około 15 mA. W ten sposób zaprogramowano wartość $F_3 = 1$ dla $A = B = C = 0$ i $D = 1$.



Rys. 6

Powtórzenie wyżej opisanej procedury dla każdego wiersza tablicy zależności stanowi całość programowania pamięci.

Do projektowania układów sekwencyjnych na bazie programowanych pamięci ROM szczególnie nadaje się metoda Huffmana, która może prowadzić do uzyskania tablic zależności dogodnych do programowania pamięci ROM.

Automaty sekwencyjne, zarówno asynchroniczne jak i synchroniczne, realizowane na bazie pamięci ROM są układami z logicznymi sprzężeniami zwrotnymi. W tej sytuacji pewna ilość wejść pamięci ROM połączona jest torami sprzężeń z wyjściami tejże pamięci.

Synteza asynchronicznych automatów sekwencyjnych realizowanych na bazie pamięci ROM polega zwykle na utworzeniu tablicy programu, zredukowanej tablicy programu itd. [6], z tym, że ostatni etap sprowadza się do określenia, na podstawie siatki przejść i siatki wyjść tablicy programowania pamięci. W tablicy programowania, wejścia adresowe są podzielone na właściwe wejścia automatu i wejścia, na które wprowadzany jest stan wewnętrzny układu.

Sposób przeprowadzenia syntezy asynchronicznego automatu sekwencyjnego ilustruje następujący przykład.

Przykład

Zrealizować na bazie pamięci ROM automat sekwencyjny o podanej na rys. 7 tablicy programu.

Na podstawie powyższych siatek zależności można utworzyć tablicę programowania pamięci ROM.

Schemat układu logicznego na bazie pamięci ROM przedstawiono na rys.8.

Sam proces programowania pamięci ROM, w przypadku realizacji automatu sekwencyjnego, jest taki sam jak dla układów kombinacyjnych.

Projektowanie synchronicznych automatów sekwencyjnych realizowanych na bazie pamięci ROM przebiega tak samo, jak dla układów asynchronicznych, tzn. kończy je określenie tablicy programowania pamięci.

Dodatkowo w przypadku zastosowania, omawianych powyżej statycznych pamięci ROM, w torach sprzężeń zwrotnych należy wprowadzić synchroniczne elementy pamięci dla przekazywania sygnałów logicznych sprzężeń zwrotnych w odpowiednich momentach narzuconych przez generator taktu.

CB

00	01	11	10	Z
(1)	4	–	6	
(2)	4	–	7	
(3)	4	–	7	
1	(4)	5	–	
–	4	(5)	6	
2	–	5	(6)	
3	–	5	(7)	

CB

00	01	11	10
(1)	(4)	(5)	6
(3)	4	5	(7)
(2)	4	–	7
2	–	5	(6)

CB

y_1y_2	00	01	11	10
00	00	00	00	10
01	01	00	00	01
11	11	00	–	01
10	11	–	00	10

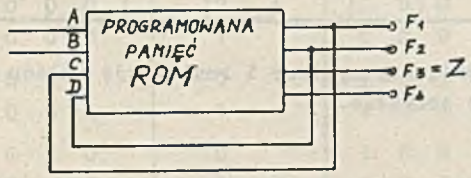
Y_1Y_2

CB

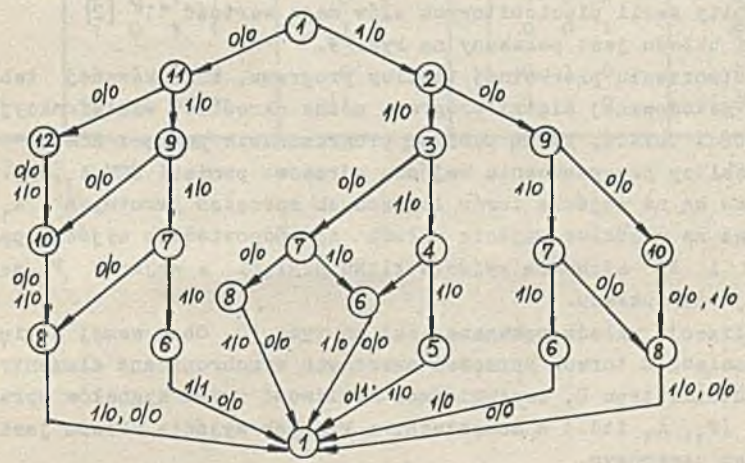
y_1y_2	00	01	11	10
00	0	0	0	0
01	1	ϕ	ϕ	1
11	0	0	ϕ	ϕ
10	0	ϕ	0	0

Z

Rys. 7



Rys. 8



Rys. 9

	ADRES				WYJŚCIA			
	A	B	C	D	F_1	F_2	F_3	F_4
	C	B	y_1	y_2	Y_1	Y_3	Z	ϕ
0	0	0	0	0	0	0	0	-
1	0	0	0	1	0	1	1	-
2	0	0	1	0	1	1	0	-
3	0	0	1	1	1	1	0	-
4	0	1	0	0	0	0	0	-
5	0	1	0	1	0	0	0	-
6	0	1	1	0	0	0	0	-
7	0	1	1	1	0	0	0	-
8	1	0	0	0	1	0	0	-
9	1	0	0	1	0	1	1	-
10	1	0	1	0	1	0	0	-
11	1	0	1	1	0	1	0	-
12	1	1	0	0	0	0	0	-
13	1	1	0	1	0	0	0	-
14	1	1	1	0	0	0	0	-
15	1	1	1	1	0	0	0	-

Sposób przeprowadzenia syntezy i realizację układu synchronicznego przedstawia następujący przykład.

Przykład

Zrealizować automat synchroniczny generujący sygnał "1", gdy cztery kolejne bity serii pięciobitowych słów mają wartość "1" [2].

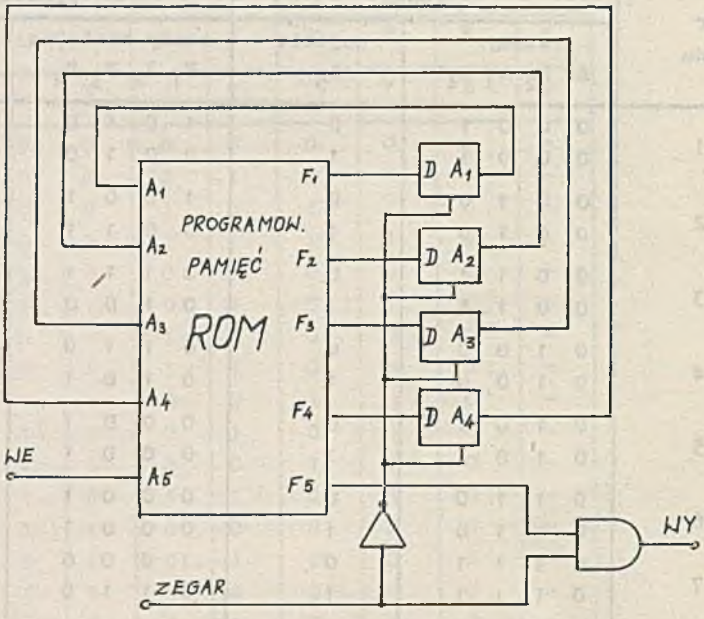
Graf układu jest pokazany na rys. 9.

Po utworzeniu pierwotnej tablicy programu, zredukowanej tablicy programu, zakodowanej siatki programu można określić wielofunkcyjną tablicę zależności układu, zwaną tablicą programowania pamięci ROM.

W tablicy programowania wejścia adresowe pamięci ROM $A_1, A_2, \ldots, A_5$ podzielone są na wejścia torów logicznych sprzężeń zwrotnych A_1, A_2, A_3 i A_4 oraz na właściwe wejście układu A_5. Odpowiednio wyjścia pamięci F_1, F_2, F_3 i F_4 stanowią wyjścia bloku pamięci, a wyjście F_5 jest właściwym wyjściem układu.

Realizacja układu pokazana jest na rys. 10. Obok samej pamięci ROM, układ posiada w torach sprzężeń zwrotnych synchroniczne elementy pamięci - przerzutniki typu D, zapewniające możliwość zmian sygnałów sprzężeń zwrotnych (F_1, A_1 itd.) w momentach, w których wyjście układu jest blokowane sygnałem zegarowym.

Nr stanu	ADRES					WYJŚCIA				
	stany				wejście	stany następne				WY
	A_1	A_2	A_3	A_4	A_5	F_1	F_2	F_3	F_4	F_5
1	0	0	0	1	0	1	0	1	0	0
	0	0	0	1	1	0	0	1	0	0
2	0	0	1	0	0	1	0	0	1	0
	0	0	1	0	1	0	0	1	1	0
3	0	0	1	1	0	0	1	1	1	0
	0	0	1	1	1	0	1	0	0	0
4	0	1	0	0	0	0	1	1	0	0
	0	1	0	0	1	0	1	0	1	0
5	0	1	0	1	0	0	0	0	1	1
	0	1	0	1	1	0	0	0	1	0
6	0	1	1	0	0	0	0	0	1	0
	0	1	1	0	1	0	0	0	1	1
7	0	1	1	1	0	1	0	0	0	0
	0	1	1	1	1	0	1	1	0	0
8	1	0	0	0	0	0	0	0	1	0
	1	0	0	0	1	0	0	0	1	0
9	1	0	0	1	0	1	0	1	0	0
	1	0	0	1	1	0	1	1	1	0
10	1	0	1	0	0	1	0	0	0	0
	1	0	1	0	1	1	0	0	0	0
11	1	0	1	1	0	1	1	0	0	0
	1	0	1	1	1	1	0	0	1	0
12	1	1	0	0	0	1	0	1	0	0
	1	1	0	0	1	1	0	1	0	0



Rys. 10

3.2. Zastosowanie programowanych zestawów logicznych (PLA) do realizacji funkcji logicznych

Uniwersalne zestawy logiczne (Programmable Logic Arrays) są efektem zastosowania ekonomicznych metod projektowania struktur, dużej skali integracji (LSI), opartych na bazie MOS.

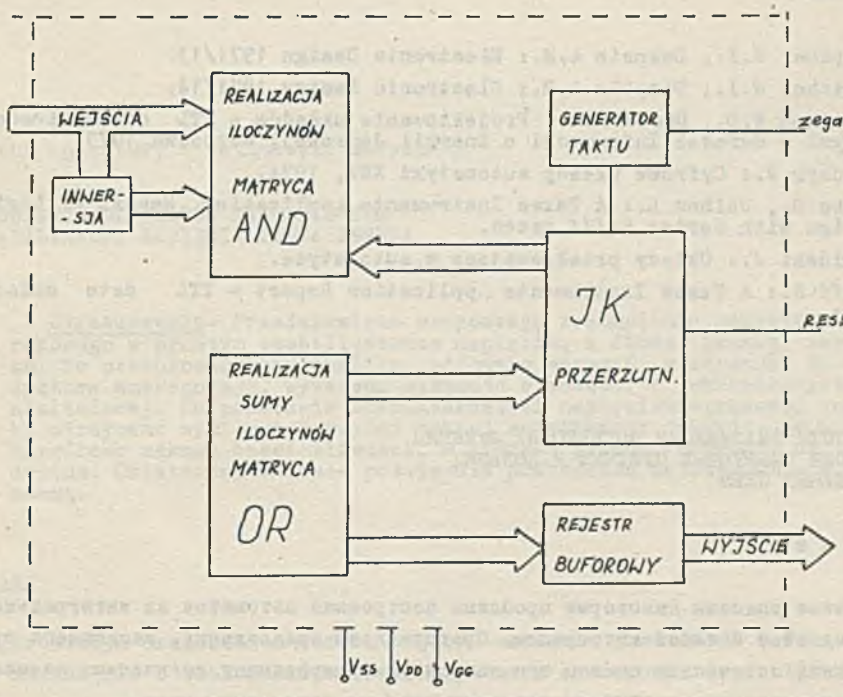
Programowanie odbywa się w procesie wytwarzania zespołów w zasadzie za pomocą jednej fotomaski zaprojektowanej przy użyciu komputera.

Każde wyrażenie logiczne można sprowadzić do normalnej postaci sumy, tzn. sumy iloczynów argumentów [6]. Struktura PLA zapewnia realizację oddzielnie iloczynów argumentów i oddzielnie sumy poszczególnych członów, przy udziale zespołu przerzutników stanowiących komórki pamięci PLA.

Struktura uniwersalnego zestawu logicznego przedstawiona została na rys. 11.

Przykładowo: PLA TMS 2000 IC zapewnia uzyskanie 60 iloczynów argumentów przy 17 wejściach, 18 wyjściach i 8 przerzutnikach, zaś TMS 2200 IC 72 iloczynów przy 13 wejściach.

Projektowanie układów logicznych z uniwersalnych zestawów logicznych sprowadza się do znalezienia funkcji logicznej określającej stan wyjść i przekształcenia jej do normalnej postaci sumy.



Rys. 11

4. Wnioski i uwagi końcowe

1. Realizacja układów logicznych, przy pomocy cyfrowych układów scalonych średniej i dużej skali integracji, zapewnia ich dużą niezawodność oraz łatwość projektowania i realizacji.

2. Do realizacji automatów kombinacyjnych mogą być użyte multipleksery w szczególności, gdy liczba wejść automatu nie przewyższa liczby wejść adresowych multipleksera o więcej niż jeden. W przeciwnym przypadku do bramkowania multipleksera potrzebne są sygnały logiczne uzyskane przy pomocy prostych układów pomocniczych.

3. Programowane pamięci ROM są szczególnie dogodne do realizacji kombinacyjnych układów logicznych o dużej liczbie wejść i wyjść.

4. Automaty sekwencyjne realizować można przede wszystkim przy użyciu programowanych zestawów logicznych (PLA), gdyż zawierają one w swojej strukturze komórki pamięci (przerzutniki) zapewniające pracę bez logicznych sprzężeń zwrotnych, co zmniejsza ilość połączeń zewnętrznych.

5. Problem minimalizacji funkcji logicznych realizowanych na bazie multiplekserów, pamięci ROM czy programowanych zestawów logicznych, w zasadzie nie istnieje.

LITERATURA

[1] Fletcher W.I., Despain A.M.: Electronic Design 1971/13.
[2] Fletcher W.I., Despain A.M.: Electronic Design 1971/14.
[3] Anderson W.D., Donce A.G.: Projektowanie układów z TTL obwodami scalonymi - Ośrodek Informacji o Energii Jądrowej, Warszawa 1973.
[4] Traczyk W.: Cyfrowe układy automatyki WNT, 1974.
[5] Crowe R., Delhom L.: A Texas Instruments Application Report - Logic design with Series 54/74 gates.
[6] Siwiński J.: Układy przełączające w automatyce.
[7] Wolff S.: A Texas Instruments Application Report - TTL data selectors.

ВОЗМОЖНОСТИ РЕАЛИЗАЦИИ ЛОГИЧЕСКИХ ФУНКЦИЙ
ПРИ ПОМОЩИ НЕКОТОРЫХ СРЕДНИХ И БОЛЬШИХ
ИНТЕГРАЛЬНЫХ СХЕМ

Р е з ю м е

В статье описаны некоторые проблемы построения автоматов из интегральных схем средней и большой интеграции. Представлены предложения, касающиеся проектирования логических систем при помощи унифицированных логических элементов;селекторов, памяти РОМ и программированных логических схем.

POSSIBILITIES OF IMPLEMENTIG LOGIC FUNCTIONS WITH SELECTED INTEGRATED CIRCUITS

S u m m a r y

The implementing of logic functions with selected integrated circuits of MSI and LSI has been presented in this paper along with the description of switching circuits with data selectors, read-only memory and programmable logic arrays.