ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1975

Seria: Automatyka z. 34 Nr kol. 458

Ferdynand Wagner
Marian Budka

Instytut Automatyki Przemysłowej i Pomiarów

SYNTEZA UKŁADÓW LOGICZNYCH REALIZOWANYCH NA BAZIE DEMULTIPLEKSERÓW

Streszczenie. Artykuł prezentuje metody projektowania układów logicznych realizowanych przy pomocy demultiplekserów. Przedstawiono sposoby syntezy i realizacji automatów cyfrowych w systemach demultiplekser (demultipleksery) - układ bramek oraz multiplekser - demultiplekser. Pokazane metody ilustrowane są praktycznymi przykładami.

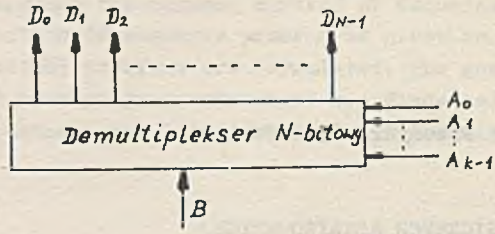
1. WSTĘP

Obok multiplekserów elementami scalonymi średniej skali integracji MSI stwarzającymi duże możliwości w zakresie realizacji funkcji logicznych, są demultipleksery (dekodery). Możliwości te wynikają z tego, że na poszczególnych wyjściach demultipleksera generowane są pełne iloczyny (składniki jedynki postaci kanonicznej) sygnałów wejściowych adresowych.

2. SYNTEZA UKŁADÓW LOGICZNYCH REALIZOWANYCH NA BAZIE DEMULTIPLEKSERÓW

Rozpatrzmy demultiplekser (dekoder) N-bitowy (N-wyjściowy) o liczbie wejść adresowych k, gdzie $k = \log_2 N$ (rys. 1). Stan wyjść $D_0, D_1 \ldots D_{N-1}$ takiego demultipleksera opisany jest następującymi funkcjami:

$$\begin{aligned}
\bar{D}_0 &= \bar{A}_0\bar{A}_1\ldots\bar{A}_{k-1} \cdot \bar{B} \\
\bar{D}_1 &= A_0\bar{A}_1\ldots\bar{A}_{k-1} \cdot \bar{B} \\
\bar{D}_2 &= \bar{A}_0 A_1\ldots\bar{A}_{k-1} \cdot \bar{B} \qquad (2.1) \\
&\ldots\ldots\ldots\ldots\ldots\ldots \\
&\ldots\ldots\ldots\ldots\ldots\ldots \\
\bar{D}_{N-1} &= A_0 A_1\ldots A_{k-1} \cdot \bar{B}
\end{aligned}$$



Rys. 1. Oznaczenie demultipleksera N-bitowego

Zatem na poszczególnych wyjściach generowane są w postaci zanegowanej określone składniki jedynki zmiennych adresowych $A_0, A_1 \ldots A_{k-1}$, mnożone przez zmienną wejściową informacyjną B.

Dokonując operacji sumowania logicznego wybranych składników można zrealizować dowolną funkcję logiczną o liczbie argumentów nie większej niż liczba wejść adresowych. Operacja ta jest ułatwiona z uwagi na to, że na wejściach demultiplekserów generowane są składniki jedynki zmiennych adresowych w postaci zanegowanej, czyli sumę można realizować przy pomocy bramki NAND.

Sposób przeprowadzenia syntezy takiego układu prezentuje przykład 1.

Przykład 1. Zrealizować przy pomocy demultipleksera układ logiczny, realizujący funkcję $F = \bar{A}_0\bar{A}_1\bar{A}_2\bar{A}_3 + \bar{A}_0A_1\bar{A}_2 + A_0\bar{A}_1\bar{A}_2A_3 + A_0A_1A_2$.

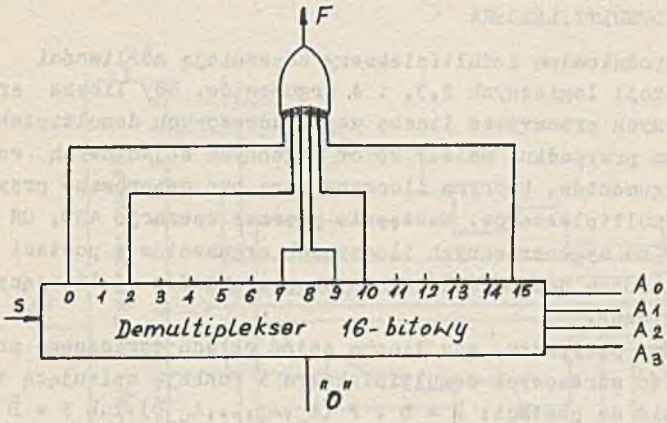
Rozwiązanie problemu polega na sprowadzeniu zadanej funkcji do normalnej postaci sumy oraz przedstawienie jej przy pomocy numerycznego zapisu dziesiętnego [3], który wprost określa sposób realizacji układu. Zgodnie z powyższym otrzymamy:

$$F = \bar{A}_0\bar{A}_1\bar{A}_2\bar{A}_3 + \bar{A}_0A_1\bar{A}_2\bar{A}_3 + \bar{A}_0A_1\bar{A}_2A_3 + A_0\bar{A}_1\bar{A}_2A_3 + A_0A_1A_2\bar{A}_3 + A_0A_1A_2A_3 +=$$

$$= \sum (0,2,7,9,10,15)_{A_0A_1A_2A_3}$$

Schemat układu przedstawiony jest na rys. 2. Do realizacji układu użyto demultiplekser 16-bitowy SN 74154.

Przy zastosowaniu demultiplekserów szczególnie ekonomiczna jest realizacja układu logicznego wielowyjściowego (przykład 2) o dużej liczbie implikantów.



Rys. 2. Realizacja układu jednowyjściowego (przykład 1)

Przykład 2. Zrealizować przy pomocy demultipleksera automat cyfrowy opisany funkcjami logicznymi:

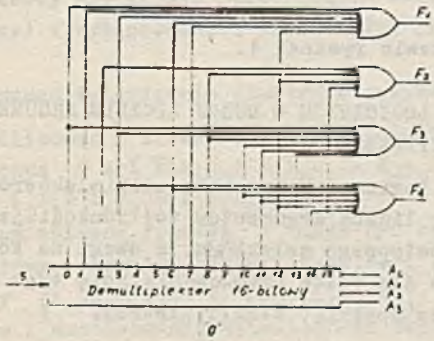
$$F_1 = \sum (0,1,3,6,7,12)_{A_0A_1A_2A_3}$$

$$F_2 = \sum (2,8,12,15)_{A_0A_1A_2A_3}$$

$$F_3 = \sum (0,3,8,10,11,13)_{A_0A_1A_2A_3}$$

$$F_4 = \sum (3,6,10,11,12,14)_{A_0A_1A_2A_3}$$

Realizacja układu na bazie demultipleksera 16-bitowego przedstawiona jest na rys. 3.



Rys. 3. Realizacja układu wielowyjściowego (przykład 2)

3. REALIZACJA UKŁADÓW LOGICZNYCH O LICZBIE WEJŚĆ WIĘKSZEJ NIŻ LICZBA WEJŚĆ ADRESOWYCH DEMULTIPLEKSERA

Aktualnie produkowane demultipleksery zapewniają możliwości prostej realizacji funkcji logicznych 2,3, i 4 argumentów. Gdy liczba argumentów funkcji logicznych przewyższa liczbę wejść adresowych demultipleksera w najogólniejszym przypadku, należy zbiór zmiennych wejściowych rozdzielić w podzbiory argumentów, których iloczyny mogą być generowane przy pomocy oddzielnych demultiplekserów. Następnie poprzez operacje AND, OR i NOT przeprowadzone na wygenerowanych iloczynach argumentów w postaci prostej i zanegowanej, można doprowadzić do uzyskania układów realizujących żądane funkcje logiczne.

W szczególnym przypadku, gdy liczba wejść układu logicznego przewyższa o 1 liczbę wejść adresowych demultipleksera i funkcję opisującą ten układ można sprowadzić do postaci: $F = B \cdot F(A_0, A_1, \ldots, A_{k-1})$ lub $F = \bar{B} \cdot F(A_0, A_1, \ldots, A_{k-1})$, możliwe jest wykorzystanie wejścia informacyjnego demultipleksera jako dodatkowego.

W innym przypadku zbiór argumentów można rozdzielić na podzbiory, tak że iloczyny zmiennych generowane przez demultiplekser poprzez proste operacje logiczne ze zmiennymi dodatkowymi tworzą żądaną funkcję logiczną. Wówczas realizacja układu może być przeprowadzona w systemie demultiplekser (demultipleksery) - prosty układ bramek.

Gdyby demultipleksery miały wyjścia z "otwartym kolektorem", zadanie realizacji dowolnych funkcji logicznych byłoby ułatwione, gdyż można by dokonywać "sumowania na drucie", tzn. bez końcowych operacji OR, które byłyby zastąpione elementami negacji. Praktycznie jest to możliwe jedynie w przypadku dekoderów "1 z 10", co jednak ogranicza liczbę argumentów do 3, gdyż taki dekoder generuje jedynie 10 z 16 możliwych pełnych iloczynów wejść adresowych w postaci prostej i zanegowanej.

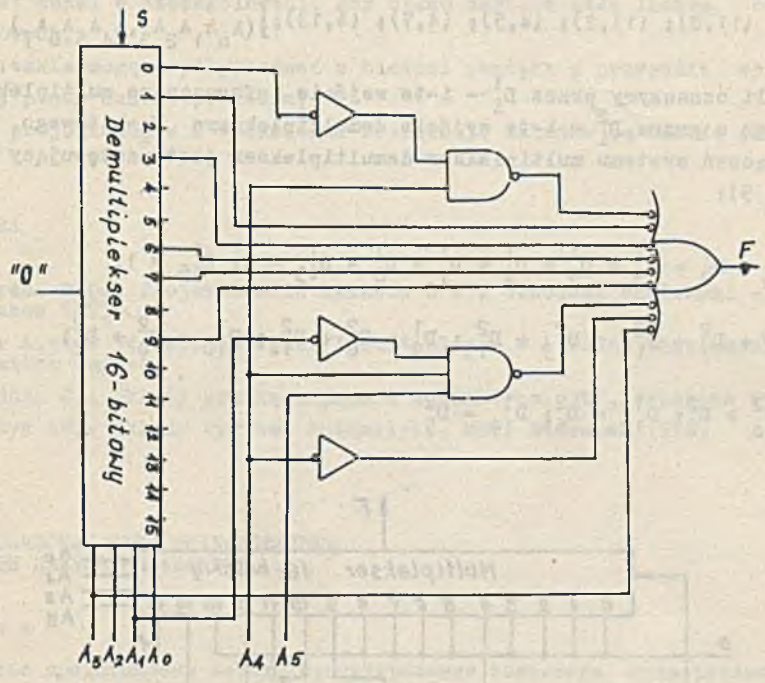
Przykład 3. Zrealizować przy pomocy demultipleksera układ logiczny opisany funkcją: $F = A_0\bar{A}_1\bar{A}_2 + \bar{A}_3 + A_4 + A_0A_1\bar{A}_2\bar{A}_3 + A_1A_2\bar{A}_3 + \bar{A}_1A_4A_5 + \bar{A}_0\bar{A}_1\bar{A}_2\bar{A}_3A_4$

Rodzielając argumenty na podzbiory $\{A_0, A_1, A_2, A_3\}$ oraz $\{A_4, A_5\}$, można zrealizować układ zadany powyższą funkcją w systemie demultiplekser - prosty układ bramek.

Schemat układu przedstawia rysunek 4.

4. REALIZACJA FUNKCJI LOGICZNYCH O DUŻEJ LICZBIE ARGUMENTÓW W SYSTEMIE MULTIPLEKSER - DEMULTIPLEKSER

Realizacja funkcji logicznych na bazie multiplekserów jest nieco utrudniona w przypadku, gdy liczba argumentów tej funkcji jest większa od liczby wejść adresowych dostępnego selektora, z uwagi na konieczność rozbudowy dodatkowych obwodów dla celów bramkowania. Rolę tego pomocniczego, wielowyjściowego układu może spełniać demultiplekser.



Rys. 4. Układ zrealizowany w systemie demultiplekser - prosty układ bramek (przykład 3)

Uwagi te można ująć w formie następującego wniosku:

Każdą funkcję logiczną o liczbie argumentów ($\log_2 N_1 + \log_2 N_2$) można zrealizować przy pomocy multipleksera N_1-bitowego (N_2) i demultipleksera N_2-bitowego (N_1) oraz bramek NAND, realizujących operacje sumy logicznej na zanegowanych iloczynach argumentów generowanych przez demultiplekser o liczbie nie większej niż liczba pomocniczych funkcji logicznych (bramkujących multiplekser) z wyłączeniem z nich stałej zero, jeden zmiennych dodatkowych.

Istotę powyższego twierdzenia ilustruje przykład 4.

Przykład 4. Zrealizować z elementów średniej skali integracji układ opisany funkcją logiczną $F = \bar{A}_0\bar{A}_1A_2\bar{A}_3A_4A_5\bar{A}_6A_7 + \bar{A}_0A_2A_3A_4\bar{A}_5A_6\bar{A}_7 + A_0\bar{A}_1A_2A_3\bar{A}_4A_5A_6 + A_0A_1A_2A_3 + A_1\bar{A}_2A_3\bar{A}_4A_6\bar{A}_7 + \bar{A}_0\bar{A}_1A_2\bar{A}_3A_4A_6$

Układ logiczny opisaną funkcją F może być zrealizowany w systemie multiplekser 16-bitowy - demultiplekser 16-bitowy. Po rozdzieleniu argumentów na podzbiory $\{A_0, A_1A_2A_3\}$ i $\{A_4, A_5, A_6, A_7\}$, rozszerzeniu odpowiednich implikantów i zastosowaniu dziesiętnego zapisu numerycznego oddzielnie dla każdego zbioru zmiennych otrzymamy:

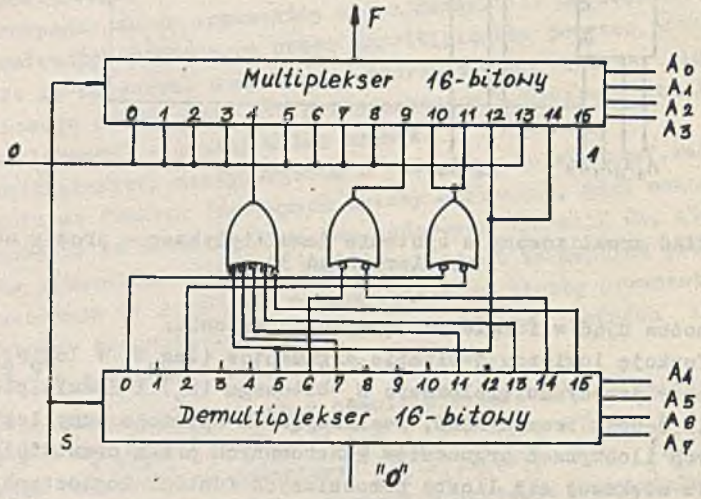
$$F = \sum\left[(4,11);(12,5);\ (14,5);\ (9,6);\ (9,14);\ (15,-);\ (10,0);\ (10,2);\ (11,0);\ (11,2);\ (4,5);\ (4,7);\ (4,13);\right]_{(A_0A_1A_2A_3,A_4A_5A_6A_7)}$$

Jeżeli oznaczymy przez D_i^1 - i-te wejście informacyjne multipleksera N_1-bitowego a przez D_i^2 - i-te wyjście demultipleksera N_2-bitowego, to sposób połączeń systemu multiplekser-demultiplekser jest następujący (jak np na rys. 5):

$$D_0^1 = D_1^1 + D_2^1 = D_3^1 = D_5^1 = D_6^1 = D_7^1 = D_8^1 = D_{13}^1 = 0;\ D_{15}^1 = 1;$$

$$D_4^1 = D_5^2 + D_7^2 + D_{11}^2 + D_{13}^2 + D_{15}^2;\ D_9^1 = D_6^2 + D_{14}^2;\ D_{10}^1 = D_0^2 + D_2^2;$$

$$D_{11}^1 = D_0^2 + D_2^2;\ D_{12}^1 = D_5^2;\ D_{14}^1 = D_5^2$$



Rys. 5. Realizacja układu z przykładu 4 w systemie multiplekser - demultiplekser

5. Wnioski

Odnośnie zastosowań demultiplekserów do realizacji układów logicznych można sformułować pewne wnioski i uwagi, a mianowicie:

- w oparciu o demultipleksery łatwo można zrealizować dowolny układ kombinacyjny, w szczególności wielowyjściowy; w przypadku dużej liczby argumentów wykorzystuje się kilka demultiplekserów,
- możliwa jest realizacja automatów o dużej liczbie wejść w systemie multiplekser - demultiplekser,

- niezawodność działania układów realizowanych przy pomocy demultiplekserów jest duża, w szczególności, gdy układ zawiera małą liczbę obwodów dodatkowych,
- układy takie mogą współpracować z blokami pamięci w przypadku synchronicznej pracy demultipleksera,
- sposób projektowania i techniczna realizacja układów są bardzo proste.

LITERATURA

[1] Anderson W.D.: Projektowanie układów z TTL obwodami scalonymi - OI EJ Warszawa 1973.
[2] Douce A.: SN 54/74154 Decoder/Demultiplexer - A TEXAS Instruments Application Report.
[3] Siwiński J.: Układy przełączające w automatyce. WNT, Warszawa 1968.
[4] Traczyk AW.: Układy Cyfrowe Automatyki. WNT, Warszawa 1974.

СИНТЕЗ ЛОШЧЕСКИХ СХЕМ РЕАЛИЗОВАННЫХ
ПРИ ПОМОЩИ ДЕМУЛЬТИПЛЕКСЕРОВ

Р е з ю м е

В статье представлены методы проектирования лошческих схем реализованных при помощи демультиплексеров. Представлено методы синтеза лошческих схем в системе демультиплексер (демультиплексеры) - лошчнские элементы и мультиплексер - демультиплексер методы илюстровано практическими примерами

SYNTHESIS OF LOGIC CIRCUITS USING DEMULTIPLEXERS

S u m m a r y

The paper presents methods of synthesis logic circuits using demultiplexers.

The methods of synthesis and realization switching circuits with systems demultiplexer (demultiplexers) - gates and data selectors - demultiplexer have been presented. The methods have been ilustrated with practical examples.