PRACE Instytutu Maszyn Matematycznych PAN

Praca B7 (20)

PEWNA METODA SYNTEZY SIECI LOGICZNYCH

Zygmunt SAWICKI

P R A C E

Instytut Maszyn Matematycznych

P o l s k i e j A k a d e m i i N a u k

T. III Praca B 7/20/

PEWNA METODA SYNTEZY SIECI LOGICZNYCH

Zygmunt SAWICKI

Warszawa 1964

Adres redakcji: Warszawa, Koszykowa 79, tel. 28-37-29.

P.427/64

Instytut Maszyn Matematycznych PAN
Praca B 7/20/

517.11:621.374:681.14-523.8

PEWNA METODA SYNTEZY SIECI LOGICZNYCH

Zygmunt SAWICKI

Pracę złożono 19.04.1963

Opisano syntezę sieci realizujących dwuwartościowe funkcje logiczne $W(x)$. Metoda oparta jest na pomocniczej funkcji f_k^i stanowiącej podstawę konstrukcji funkcji $W(x)$, jak również sieci realizującej tę funkcję. Przedstawiono także metodę minimalizacji postaci funkcji $W(x)$.

S P I S T R E Ś C I

W s t ę p.

Problem syntezy sieci realizujących funkcje logiczne jest ciągle aktualny i jest przedmiotem szeregu prac, zarówno w aspekcie teoretycznym jak i praktycznym.

Szybki rozwój maszyn cyfrowych wymaga z jednej strony coraz to doskonalszych metod syntezy, z drugiej strony pozwala na wzbogacenie tych metod przez wykorzystanie maszyn cyfrowych do projektowania sieci logicznych budowanych maszyn.

Do trudniejszych problemów, aktualnie rozwiązywanych, należy zaliczyć:

1. znalezienie jednoznacznej metody syntezy sieci realizujących dowolne dwuwartościowe funkcje logiczne nieograniczonej ilości zmiennych,
2. znalezienie metody jak w pkt.1, lecz prowadzącej do minimalizacji ilości elementów sieci,
3. opracowanie metod programowania zagadnień syntezy sieci logicznych na maszyny cyfrowe, ich konstrukcji w oparciu o standartowe układy funktorów, zespołów i zestawów.

Obecnie znany jest szereg prac, pozwalających na wykorzystanie ich wyników w praktyce jako metod projektowania, wymienię chociażby prace [2], [3], [5], [8], [9], [10], [11]. Jednakże zakres ich stosowalności jest praktycznie ograniczony, ze względu na ilość rozpatrywanych zmiennych funkcji lub na klasę realizowanych funkcji albo też ze względu na efektywność w użyciu metody, złożoność procedury postępowania i bardzo często ze względu na niedostateczne przygotowanie matematyczne inżynierów-projektantów, którzy mieliby stosować te metody.

Ze znanych autorowi metod syntezy, na szczególne podkreślenie zasługują metody przedstawione w pracach W.V.Quine'a, E.J.Mc Cluskey'a i S. Waligórskiego.

W pracy [2] E.J.Mc Cluskey przedstawia metodę opartą na kryterium minimalnej ilości wyrazów funkcji zadanej przy minimalnej ilo-

ści czynników tych wyrazów. Metoda ta jest dalszym rozwinięciem prac W.V. Quine'a.

S. Waligórski szereg prac poświęca syntezie sieci logicznych [5], [6], [8], [9]. Np. w pracy [5] formułuje, na bazie tablicy Quine'a, efektywną metodę syntezy sieci i minimalizacji postaci zadanej funkcji. Ponadto autor pracy [5] znalazł algorytm i zaprogramował ten problem na maszynę cyfrową.

Ostatnio ukazało się kilka prac poświęconych problemowi zastosowania maszyn cyfrowych do przestrzennego projektowania nowych maszyn cyfrowych, a w szczególności tabel połączeń poszczególnych zespołów przy uwzględnieniu szeregu warunków technicznych. Do tej grupy należą prace [10] i [11].

Autor niniejszej publikacji zadał sobie trud opracowania i przedstawienia pewnej metody zmechanizowanego sposobu postępowania w zakresie syntezy sieci realizujących dwuwartościowe funkcje logiczne o dowolnej ilości zmiennych i określonych w całym lub w części zbioru wszystkich możliwych kombinacji wartości tych zmiennych. Koncepcja tej metody zaczerpnięta jest z pracy [4].

1. Założenia i definicje.

Niech E będzie zbiorem złożonym z dwu elementów oznaczonych odpowiednio przez 0 i 1. Na tym zbiorze możemy określić funkcje dowolnej ilości zmiennych, przyjmujące wartości należące również do E.

Przez E^N oznaczymy zbiór wszystkich ciągów N-elementowych

$$(x_1, x_2, x_3, \ldots, x_N), \qquad /1/$$

których wyrazy należą do zbioru E.

Ciąg /1/ będziemy również oznaczać przez x t.j.:

$$x = (x_1, x_2, \ldots, x_N) \qquad /2/$$

Zbiór E^N zawiera 2^N różnych elementów. Funkcje N zmiennych określoną w zbiorze E^N oznaczymy przez $W = f(x)$. Funkcja $W(x)$ przyjmuje wartości należące do zbioru E. Oczywiście funkcja $W(x)$ nie musi być określona dla wszystkich $x \in E^N$.

Oznaczmy przez A^M układ M funkcji N zmiennych

$$\begin{aligned} W^{(1)} &= f^{(1)}(x_1, x_2, \ldots, x_N) \\ W^{(2)} &= f^{(2)}(x_1, x_2, \ldots, x_N) \\ &\cdots\cdots\cdots\cdots\cdots\cdots \\ W^{(M)} &= f^{(M)}(x_1, x_2, \ldots, x_N); \end{aligned} \qquad /3/$$

przyjmiemy skrócony zapis tego układu

$$W^{(m)} = W^{(m)}(x_1, x_2, \ldots, x_N) = W^{(m)}(x), \qquad /4/$$

gdzie: $m = 1, 2, \ldots, M$, przy czym $M \leq 2^N$

x_i - $(i = 1, 2, \ldots, N)$ zmienne dwuwartościowe.

Załóżmy, że układ funkcji $W^{(m)}(x)$ jest określony w zbiorze E^N przez tablicę zerojedynkową. Wiersze tablicy zawierają elementy zbioru E^N oraz odpowiednie wartości funkcji $W^{(m)}(x)$. W zależności od ilości jedynek w danym wierszu oraz w zależności od wartości funkcji $W^{(m_o)}(x)$ dla ustalonego $m = m_o$, zbiór E^N podzielimy na dwie rodziny podzbiorów:

$$\left\{B_j^1\right\} \quad \text{i} \quad \left\{B_j^0\right\} \qquad /5/$$

gdzie: $j = 1, 2, \ldots, N$.

Zbiorowi B_j^k $(k = 0, 1)$ odpowiadają wiersze tablicy zawierające dokładnie j jedynek, dla których wartość funkcji $W^{(m_o)}(x) = k$. Dla każdego elementu zbioru $b \in B_j^k$ można na zbiorze E^N określić funkcję będącą iloczynem logicznym j argumentowym, zbudowanym tylko z tych zmiennych x_n $(n = 1, 2, \ldots, N)$, których wartość równa jest 1. Funkcję tę oznaczymy przez $I_{j,b}^k$, tzn.

$$I^k_{j,b} = \prod_{e=1}^{j} x_{i_e}, \qquad /6/$$

gdzie: $b \in B^k_j$, przy czym i_e jest ciągiem określonym w sposób następujący:

a/ $1 \leq i_1 < \ldots < i_j \leq N$

b/ $W^{(m_o)}(x) = W^{(m_o)}(\ldots \underset{i_1}{0}, \underset{i_1}{1}, 0 \ldots \underset{i_2}{1}, \ldots \underset{i_j}{1}, \ldots) = k$

W punkcie /b/ pokazano sposób przyporządkowania wskaźnika i_e zmiennej x_{i_e}, dla $W^{(m_o)}(x) = k$.

Teraz zdefiniujemy funkcję $f^k_j(x)$ jako:

$$f^k_j(x) = \sum_{b \in B^k_j} I^k_{j,b} = \sum_{b \in B^k_j} \prod_{e=1}^{j} x_{i_e}, \qquad /7/$$

gdzie: $\sum$ - jest symbolem sumy logicznej,

$\prod$ - jest symbolem iloczynu logicznego.

Funkcje te stanowią podstawę syntezy sieci realizujących układ funkcji $W^{(m)}$, $(m = 1, 2, \ldots, M)$, którego postać zapiszemy w sposób następujący:

$$\begin{aligned} W^{(1)}(x) &= W^{(1)}(f^1_j, f^0_j) \\ W^{(2)}(x) &= W^{(2)}(f^1_j, f^0_j) \\ &\cdots\cdots\cdots\cdots\cdots \\ W^{(M)}(x) &= W^{(M)}(f^1_j, f^0_j) \end{aligned} \qquad /8/$$

Układ ten obejmuje dowolne funkcje logiczne określone w zbiorze E^N

i nie nakłada się żadnych ograniczeń, zarówno na ilość zmiennych x_n, jak również na wielkość układu funkcji $W^{(m)}(x)$. Jedynym ograniczeniem jest założenie, że zarówno $W^{(m)}(x)$, jak i $x_1, x_2, .., x_N$ nie zależą od czasu.

2. Uogólniona i syntetyczna postać funkcji $W^{(m)}(x)$.

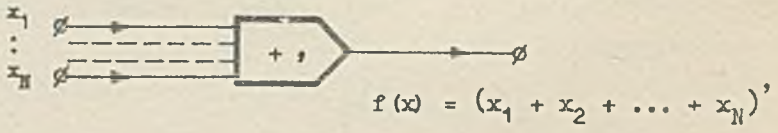
Konstrukcja funkcji $W^{(m)}(x)$ jest oparta o pomocnicze funkcje f_j^1 i f_j^0 zgodnie z twierdzeniem przedstawionym w pracy [4] oraz wzorami /7/ i /8/.

Postać funkcji $W^{(m)}(x)$ wg [4] była uwarunkowana specyficznymi właściwościami sieci współpracującej z elementem pamięciowym stanowiącym wyjście tej sieci. Taka postać jest nieodpowiednia dla sieci budowanych w oparciu o jeden i tylko jeden typ funktora zwanego negatorem, definicję którego podajemy poniżej. Przedstawiona metoda syntezy dotyczy właśnie sieci, w skład których wchodzi wyłącznie negator.

Negator realizuje następująca postać funkcji:

$$f(x) = (x_1 + x_2 + \ldots + x_N)', \qquad /9/$$

którego symbol graficzny przedstawia rysunek

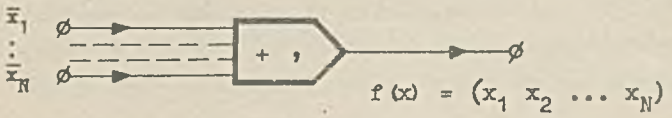


Rys. 1

W omawianej sieci funktor iloczynu logicznego realizowany będzie przy pomocy tego negatora wykorzystując znane prawo De'Morgana, a mianowicie:

$$f(x) = x_1 \cdot x_2 \cdot \ldots \cdot x_N = (\bar{x}_1 + \bar{x}_2 + \ldots + \bar{x}_N)'$$

natomiast w symbolice graficznej zapiszemy:



Rys. 2

Funkcja $W^{(m_o)}$ przedstawiona w [4] została tutaj przekształcona i sprowadzona do postaci odpowiadającej realizacji sieci omawianego typu. Sieć zbudowana z negatorów posiada jednolitą strukturę węzłów i rozgałęzień. W węzłach następuje sumowanie logiczne wartości wejściowych i negacja wartości sumy. Rozgałęzienia łączą poszczególne węzły. Postać funkcji $W^{(m_o)}$ powinna odpowiadać przedstawionej strukturze sieci. Powyższe założenie zostało spełnione, co bezpośrednio wynika z wyrażeń oznaczonych numerami /11/, /12/, /13/ i /14/.

Zgodnie z twierdzeniem przedstawionym w [4] mogą zachodzić cztery przypadki mające wpływ na postać funkcji $W^{(m_o)}$. Przypadki te określone są dwoma parametrami, mianowicie

1. wartością funkcji $W^{(m_o)}(x)$ dla $x = 0$
2. wartością N; N - parzyste, lub N - nieparzyste /N-ilość zmiennych funkcji $W^{(m_o)}$/.

Powyższe przypadki zestawimy w następującej tablicy:

Oznaczenie typu funkcji $W^{(m_o)}(x)$, $(r = 1,2,3,4)$.

Kolejne przypadki	Oznaczenie $W^{(m_o)}(x)$	$W^{(m_o)}(x)$ dla $x=0$	Wartość N
1	$W_1^{(m_o)}(x)$	1	parzyste
2	$W_2^{(m_o)}(x)$	1	nieparzyste
3	$W_3^{(m_o)}(x)$	0	nieparzyste
4	$W_4^{(m_o)}(x)$	0	parzyste

W wyniku odpowiednich przekształceń wyrażeń typu $W^{(m_o)}(x)$ przedstawionych w [4] otrzymujemy cztery wzory ogólnych postaci funkcji typu $W_r^{(m_o)}(x)$, $(r = 1,2,3,4)$, pozwalających na łatwe i bezpośrednie przejście od zadanej formy funkcji do jej postaci opisującej poszukiwaną sieć i tym samym do konstrukcji tej sieci.

$$W_1^{(m_o)}(x) = (\!(\ldots\ (\!(\!(f_1^0 + f_2^0)' + f_2^1 + f_3^1)' + f_3^0 + f_4^0)' + \ldots + f_{N-2}^1 + f_{N-1}^1)' + f_{N-1}^0 + f_N^0)' + f_N^1 \qquad /11/$$

$$W_2^{(m_o)}(x) = (\!(\ldots\ (\!(\!(f_1^0 + f_2^0)' + f_2^1 + f_3^1)' + f_3^0 + f_4^0)' + \ldots + f_{N-1}^1 + f_N^1)' + f_N^0)' \qquad /12/$$

$$W_3^{(m_o)}(x) = (\!(\ldots\ (\!(\!(\!(f_1^1 + f_2^1)' + f_2^0 + f_3^0)' + f_3^1 + f_4^1)' + f_4^0 + f_5^0)' + \ldots + f_{N-2}^1 + f_{N-1}^1)' + f_{N-1}^0 + f_N^0)' + f_N^1 \qquad /13/$$

$$W_4^{(m_o)}(x) = (\!(\ldots\ (\!(\!(\!(f_1^1 + f_2^1)' + f_2^0 + f_3^0)' + f_3^1 + f_4^1)' + f_4^0 + f_5^0)' + \ldots + f_{N-1}^1 + f_N^1)' + f_N^0)' \qquad /14/$$

Przedstawione wyżej wyrażenia posiadają jednolitą i prostą strukturę, pozwalającą na zmechanizowanie sposobu postępowania w ich zastosowaniu do syntezy sieci logicznych. Wzory te obejmują wszystkie możliwe przypadki dwuwartościowych funkcji logicznych, określonych na całym lub części zbioru E^N.

3. Konstrukcja sieci realizujących funkcje $W_r^{(m_o)}(x)$.

Konstrukcja sieci realizujących zadany układ funkcji $W_r^{(m_o)}(x)$ składa się z negatorów opisanych w rozdziale 2.

Zgodnie z definicją funkcji f_j^k poszczególne jej składniki, tj.

$I^k_{j,b}$, zależą od zmiennych nienegowanych $x_1, x_2, \ldots, x_N$. Operowanie takimi zmiennymi ma miejsce na całym etapie formalnych przekształceń, począwszy od założeń, skończywszy na ostatecznej postaci funkcji zadanej opisującej poszukiwaną sieć. Funktory $I^k_{j,b}$ są składnikami funkcji f^k_j i są realizowane przez negatory na zasadzie

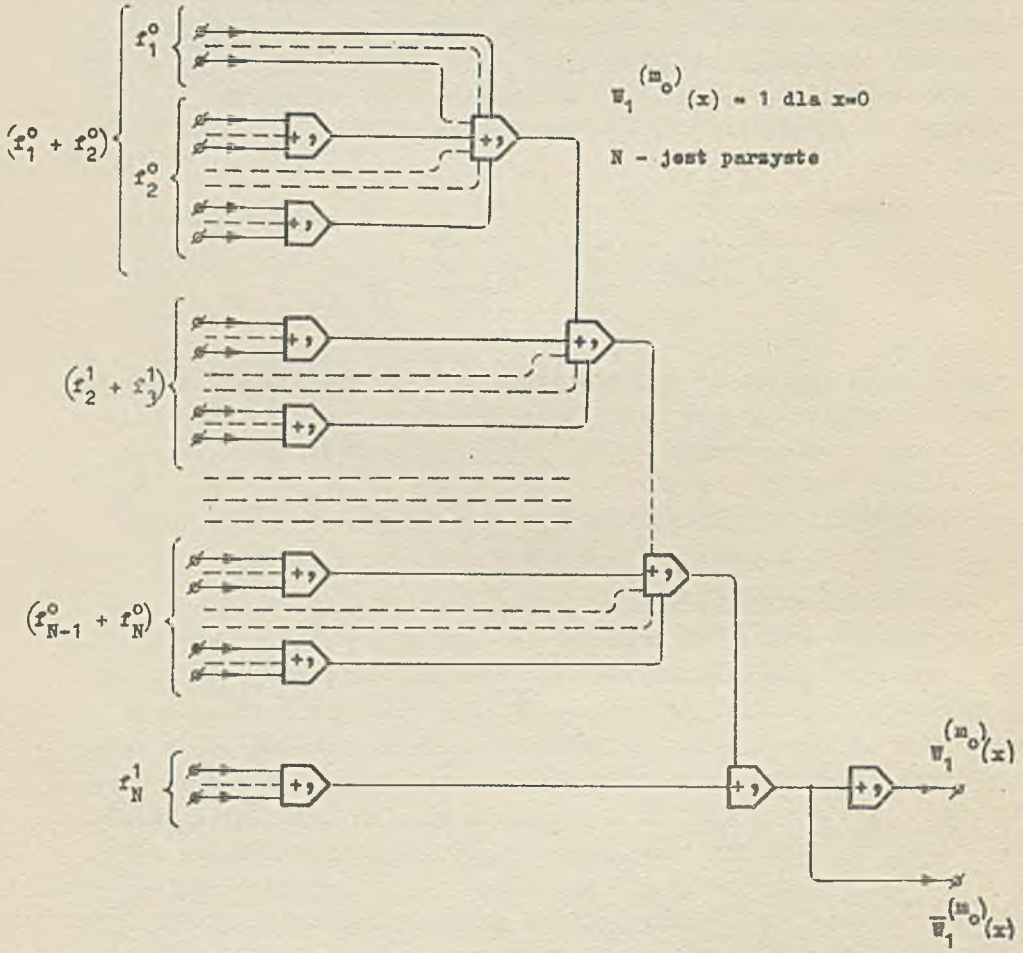
$$I^k_{j,b} = \left(\sum_{e=1}^{j} \bar{x}_{i_e}\right)' = \prod_{e=1}^{j} x_{i_e}$$

Z powyższego wynika, że z chwilą przejścia do konstrukcji sieci w oparciu o negatory, przy realizacji iloczynów logicznych operujemy zmiennymi zanegowanymi w ten sposób, że na wejścia negatorów wprowadzamy z m i e n n e n e g o w a n e $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$. Przy realizacji sum logicznych na wejścia negatorów wprowadzamy z m i e n n e n i e n e g o w a n e $x_1, x_2, \ldots, x_n$.

Wzory ogólnych postaci funkcji $W_r^{(m_o)}(x)$ dla $r = 1,2,3,4$ mają identyczną strukturę, co można stwierdzić porównując je między sobą. Dzięki tej właściwości, sieci realizujące te funkcje mają też jednakową konstrukcję. Różnią się jedynie sposobem wprowadzania zmiennych na ich wejścia. Na wejścia negatorów wchodzących w skład sieci realizujących funktory iloczynu logicznego, wprowadzamy wyłącznie z m i e n n e n e g o w a n e. Na wejścia negatorów realizujących funktory sumy logicznej wprowadzamy wyłącznie z m i e n n e n i e n e g o w a n e. Z tych właśnie względów istnieje jedna i tylko jedna ogólna postać konstrukcji sieci realizującej funkcje $W_r^{(m_o)}(x)$. Sieć tego typu oznaczymy przez SWR. Wielkość sieci SWR zależy od stopnia komplikacji zadanej funkcji $W_{r_o}^{(m_o)}(x)$ lecz jej konstrukcja zachowuje typowy charakter.

Poniżej przedstawiamy ogólną postać sieci SWR odpowiadającą poszczególnym wzorem $W_r^{(m_o)}(x)$ dla $r = 1, 2, 3, 4$.

Sieć, która realizuje funkcję $W_1^{(m_o)}(x)$ nazywać będziemy siecią SW1.

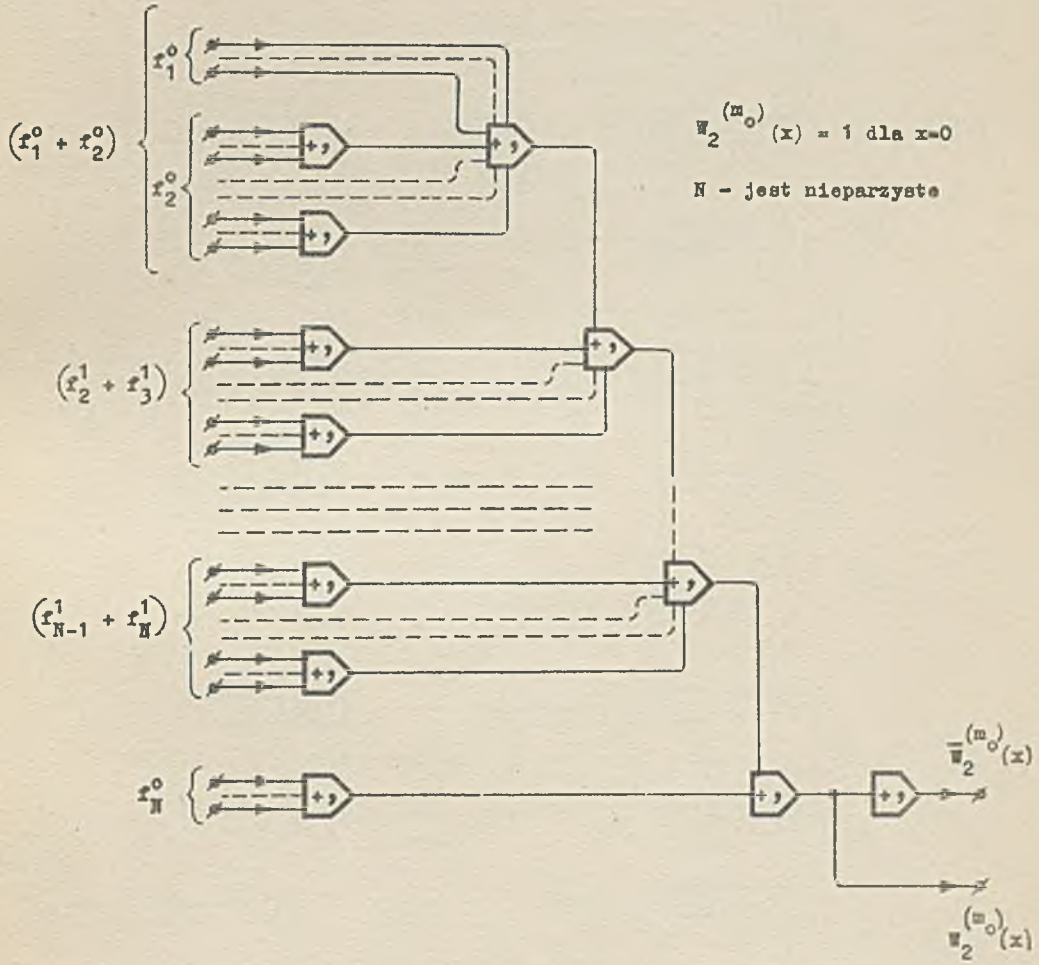


Rys. 3. Sisć typu SW1.

Uwagi:

1. Argumenty funkcji f_j^0 nie są zanegowane na wejściu sieci.
2. Argumenty funkcji f_j^k $(j = 2, 3, \ldots, N)$ są zanegowane na wszystkich wejściach sieci.

Sieć, która realizuje funkcję $W_2^{(m_o)}(x)$ nazywać będziemy siecią SW2.

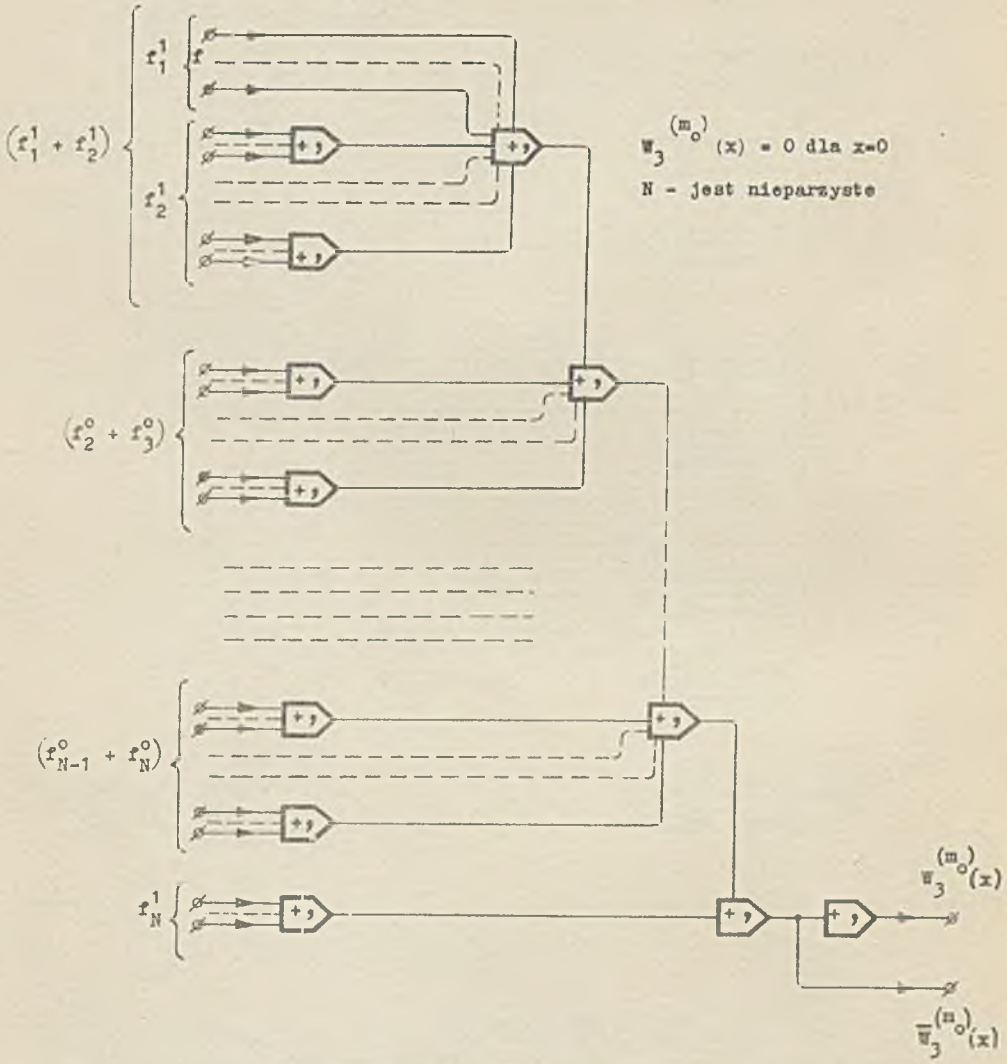


Rys. 4. Sieć typu SW2.

Uwagi:

1. Argumenty funkcji f_j^o nie są zanegowane na wejściu sieci.
2. Argumenty funkcji f_j^k ($j = 2, 3, \ldots, N$) są zanegowane na wszystkich wejściach sieci.

Sieć, która realizuje funkcję $W_3^{(m_o)}(x)$ nazywać będziemy siecią SW3.

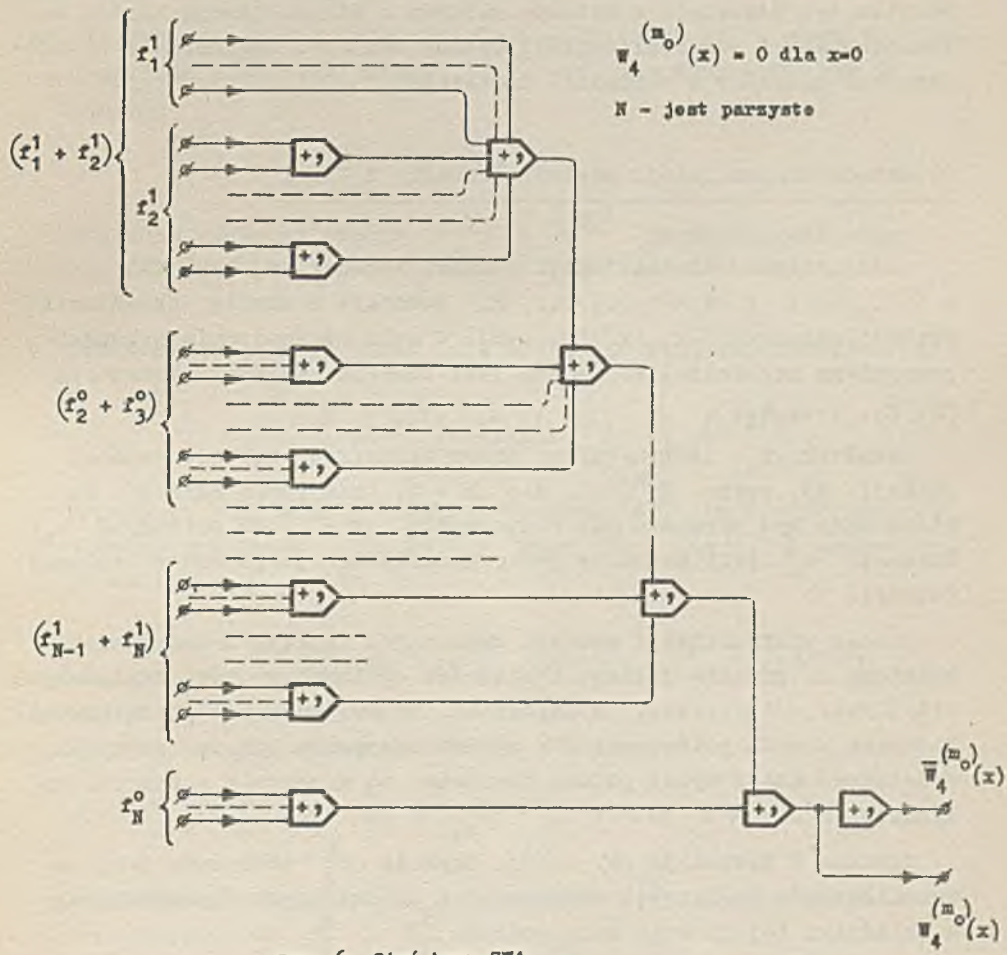


Rys. 5. Sieć typu SW3.

Uwagi:

1. Argumenty funkcji f_1^1 nie są zanegowane na wejściu sieci.
2. Argumenty funkcji f_j^1 $(j = 2, 3, \ldots, N)$ są zanegowane na wszystkich wejściach sieci.

Sieć, która realisuje funkcję $W_4^{(m_o)}(x)$ nazywać będziemy siecią SW4.



Rys. 6. Sieć typu SW4.

Uwagi:

1. Argumenty funkcji f_1^1 nie są zanegowane na wejściu sieci.
2. Argumenty funkcji f_j^1 $(j = 2, 3, \ldots, N)$ są zanegowane na wszystkich wejściach sieci.

Istnieje problem znalezienia możliwie minimalnej ilości wyrazów funkcji zadanej, zbudowanej zgodnie z opisanymi wyżej wzorami. Problem ten pozostaje w ścisłym związku z minimalizacją ilości elementów sieci SWR realizującej zadaną funkcję. Zagadnienie to rozpatrywać będziemy w rozdziale następnym.

4. Metoda minimalizacji postaci funkcji $W_r^{(m)}(x)$.

Zagadnienie minimalizacji postaci funkcji $W_r^{(m)}(x)$ dla $r = 1, 2, 3, 4$ i $m = 1, 2, \ldots, M$, powstaje z chwilą określenia funkcji zmiennych $x_1, x_2, \ldots, x_N$. W celu uproszczenia rozważań przyjmiemy założenie, że zadana jest funkcja $W_{r_o}^{(m_o)}(x)$ zmiennych $x_1, x_2, \ldots, x_N$.

Wskaźnik r_o jest ustalony przez warunki początkowe zadanej funkcji, tj. przez $W_{r_o}^{(m_o)}(x)$ dla $x = 0$, oraz przez wartość N, która może być parzysta lub nieparzysta, N - ilość zmiennych x_n. Wskaźnik m_o jest ustalony przez nadanie kolejnego numeru zadanej funkcji.

Proces minimalizacji postaci omawianych funkcji przeprowadzać będziemy na gruncie jednego z czterech ogólnych wzorów, oznaczonych /11/, /12/, /13/, /14/, w zależności od ww. warunków początkowych, jednakże sposób postępowania w każdym przypadku będzie identyczny. Podstawowe wzory wyżej podane zbudowane są w oparciu o pomocniczą funkcję f_j^k.

Zgodnie z definicją /6/ i /7/, funkcja f_j^k zbudowana jest z sumoiloczynów logicznych zmiennych x_n, zmiennych nienegowanych; a składniki tej funkcji mają postać:

$$I_{j,b}^k\left(x^{(j)}\right) = \prod_{e=1}^{j} x_{1_e},$$

gdzie: $j = 1, 2, \ldots, N$; $k = 1, 0$; symbol $x^{(j)}$ wyraża zmienną określoną przez ciąg j-elementowy zbudowany z dowolnych j elementów ciągu x oznaczonego przez /1/ i /2/.

Rozpatrzmy kilka właściwości $I^k_{j,b}(x^{(j)})$, które będą wykorzystane przy minimalizacji postaci zadanej funkcji

a/ Dla każdego $i \leqslant j$ wszystkie elementy ciągu $x^{(i)}$ mogą być jednocześnie elementami ciągu $x^{(j)}$, co oznaczymy symbolem C, tzn.

$$x^{(i)} C \; x^{(j)} \qquad /15/$$

Ponieważ elementy ciągów $x^{(i)}$ i $x^{(j)}$ są podciągami ciągu $x = (x_1, x_2, \ldots, x_N)$, to dla $i \leqslant j$ może zachodzić /15/.

b/ Dla każdego $i \leqslant j$, oraz $x^{(i)} C \; x^{(j)}$ zachodzi relacja /16/

$$I^k_{i,b}(x^{(i)}) = I^k_{i,b}(x^{(i)}) + I^k_{j,b}(x^{(j)}) \qquad /16/$$

Wyrażenie /16/ wynika z faktu, że wyraz $I^k_{j,b}(x^{(j)})$ składa się z dwóch czynników: $I^k_{i,b}(x^{(i)})$ oraz $I^k_{j-i,b}(x^{(j-i)})$, wobec tego mamy:

$$I^k_{i,b}(x^{(i)}) = I^k_{i,b}(x^{(i)})\left(1 + I^k_{j-i,b}(x^{(j-i)})\right)$$

c/ Dla każdego j zachodzi relacja /17/

$$\text{gdy } I^k_{j,b}(x^{(j)}) = 1, \text{ to } I^l_{j,b}(x^{(j)}) = 0 \text{ dla } k \neq l \qquad /17/$$

Powyższa zależność wynika z faktu, że $x^{(j)}$ jednocześnie nie może należeć do B^k_j i B^l_j.

Na podstawie zależności /16/ wnioskujemy, że skuteczność redukcji wyrażenia typu $I^k_{i,b}(x^{(i)}) + I^k_{j,b}(x^{(j)})$, $x^{(i)} C \; x^{(j)}$ jest tym efektywniejsza, im większa jest różnica $j-i$.

W procesie redukcji zbędnych wyrazów lub czynników, występujących w członach zadanej funkcji $W^{(m_o)}_{r_o}(x)$, wyróżnimy trzy etapy, które kolejno omówimy.

4.1. Pierwszy etap minimalizacji $W_{r_o}^{(m_o)}(x)$.

Wyjściową postacią zadanej funkcji jest jeden z czterech wzorów oznaczonych /11/, /12/, /13/ i /14/, które są zbudowane z wyrażeń typu $\left(f_j^k + f_{j+1}^k\right)$ $j = 1,2,\ldots, N$; $k = 1$; lub $k=0$. Podstawiając za f_j^k i f_{j+1}^k odpowiednie sumoiloczyny $I_{j,b}^k$ i $I_{j+1,b}^k$ zgodnie z /7/ otrzymujemy ciąg wyrażeń typu:

$$\left(\sum_{b \in B_j^k} I_{j,b}^k + \sum_{b \in B_{j+1}^k} I_{j+1,b}^k\right)$$

Z łatwością zauważymy, że odnośnie tych wyrażeń zastosowanie ma wzór redukcyjny oznaczony przez /16/. W wyniku zastosowania takiej operacji redukcji zbędnych wyrazów otrzymujemy w praktyce zasadnicze uproszczenie postaci zadanej funkcji $W_{r_o}^{(m_o)}(x)$. Dalsze uproszczenia są możliwe i skuteczne, szczególnie w tym wypadku, gdy dana funkcja $W_{r_o}^{(m_o)}(x)$ jest określona tylko na części zbioru E^N.

4.2. Drugi etap minimalizacji $W_{r_o}^{(m_o)}(x)$.

Wyjściowa postać zadanej funkcji $W_r^{(m_o)}(x)$ opisana jest przez wyrażenia typu:

$$\left(\left(f_j^l + f_{j+1}^l\right)' + f_{j+1}^k + f_{j+2}^k\right)'$$

gdzie: $j=1,2,\ldots, N-2$; $k \neq l$; $k = 1,0$; $l = 0,1$;

Upraszczanie postaci tego wyrażenia polega na wprowadzeniu do niego dodatkowych wyrazów $I_{j,b}^k$ i $I_{j+1,b}^k$ lecz takich, które będą powodowały redukcję zbędnych wyrazów. Oczywiście te dodatkowe wyrazy muszą spełniać pewne warunki, które zabezpieczą niezmienność pierwotnych zależności zadanej funkcji. Dodatkowe wyrazy mogą być określone szczególnie na tych elementach zbioru E^N, na których nie jest określona zadana funkcja $W_{r_o}^{(m_o)}(x)$.

Opisaną wyżej operację redukcji możemy zapisać w następujący sposób:

$$\left(\left(f_j^1 + f_{j+1}^1\right)' + f_{j+1}^k + f_{j+2}^k\right)' =$$

$$= \left(\left(f_j^1 + f_{j+1}^1\right)' + f_{j+1}^k + f_{j+2}^k + I_{j,b}^k + \ldots + I_{j+1,b}^k\right)' \qquad /18/$$

Wyrazy $I_{j,b}^k$ i $I_{j+1,b}^k$ w wyrażeniu /18/ muszą spełniać warunki:

1/ Warunek 1 - oznaczony przez /16/

2/ Warunek 2 - dla każdego $I_{j,b}^k(x)^{(j)}$ nie może być $x^{(j)} C x^{(j+1)}$, podczas gdy: $x^{(j)} \in B_j^k$, a $x^{(j+1)} \in B_{j+1}^1$ dla $k \neq 1$.

Przedstawione warunki odnoszą się do p i e r w o t n y c h p o s t a c i funkcji f_j^k i f_{j+1}^k, a nie do postaci otrzymanych po pierwszym etapie redukcji.

4.3. Trzeci etap minimalizacji $W_{r_o}^{(m_o)}(x)$.

Metoda upraszczania postaci funkcji zadanej oparta jest na tym etapie również na podstawowych wzorach oznaczonych przez /11/,/12/, /13/ i /14/. Rozpatrzymy tylko jeden z tych wzorów, np. /13/, bowiem inne wykorzystuje się w ten sam sposób. W celu ułatwienia przedstawienia tej metody, w wyrażeniu /13/ ponumerujemy wszystkie kolejne nawiasy od 1, 2, 3, ..., do N-1.

$$W_3^{(m_o)}(x) = \left(\ldots \left(\left(f_1^1 + f_2^1\right)_1' + f_2^o + f_3^o\right)_2' + f_3^1 + f_4^1\right)_3' +$$

$$+ \ldots + f_{N-1}^o + f_N^o\Big)_{N-1}' + f_N^1$$

Metoda upraszczania przedstawionej postaci funkcji $W_3^{(m_o)}(x)$ polega na redukcji zbędnych wyrazów w kolejnych wyrażeniach objętych nawiasami, a które należy rozpatrzyć począwszy od pierwszego i skończywszy na wyrazie f_N^1, to znaczy, że:

1. Wyrażenie zawarte między nawiasami 1 i 2 względem wyrażenia objętego nawiasem 1 tj.

$$\left(\left(f_1^1 + f_2^1\right)_1' + f_2^0 + f_3^0\right)_2'$$

2. Wyrażenie zawarte między nawiasami 2 i 3 względem wyrażenia objętego nawiasem 2 tj.

$$\left(\left(\left(f_1^1 + f_2^1\right)_1' + f_2^0 + f_3^0\right)_2' + f_3^1 + f_4^1\right)_3'$$

i t.d.

.....

3. Wyraz N-ty tj. f_N^1 względem wyrażenia objętego nawiasem N-1. Wszystkie wymienione wyrażenia charakteryzują się identycznymi właściwościami, a sposób redukcji zbędnych wyrazów dla każdego z nich jest również taki sam.

Rozpatrzmy metodę upraszczania wyrażenia pierwszego, zaś wnioski wyciągniemy odnośnie wszystkich wyżej wymienionych postaci:
Do wyrażenia $\left(\left(f_1^1 + f_2^1\right)_1' + f_2^0 + f_3^0\right)_2'$ podstawimy odpowiednie sumoiloczyny zbudowane z $I_{j,b}^k$ dla $j=1,2,3$; oraz $k=1,0$. W wyniku tej operacji otrzymujemy:

$$\left(\left(\sum_{b\in B_1^1} I_{1,b}^1\left(x^{(1)}\right) + \sum_{b\in B_2^1} I_{2,b}^1\left(x^{(2)}\right)\right)_1' + \sum_{b\in B_2^0} I_{2,b}^0\left(x^{(2)}\right) + \sum_{b\in B_3^0} I_{3,b}^0\left(x^{(3)}\right)\right)_2'$$

Wyrażenie objęte nawiasem 1 /bez negacji/ oznaczymy przez W_1; wyrażenie zawarte między nawiasami 1 i 2 /bez negacji/ oznaczymy przez $W_{1,2}$. Wyrażenia W_1 i $W_{1,2}$ przyjmują wartości 1, lub 0 dla każdego z możliwych ciągów $x^{(1)}$, $x^{(2)}$, $x^{(3)}$, co zestawimy w następującej tablicy:

Lp.	W_1	$W_{1,2}$
1	0	0
2	1	0
3	0	1
4	1	1

Właśnie przypadek /3/ w powyższej tablicy wykorzystujemy do redukcji zbędnych wyrazów w wyrażeniu $W_{1,2}$.

Twierdzenie 1.

Każdy wyraz $I^0_{2,b}(x^{(2)})$, $x^{(2)} \in B^0_2$, oraz $I^0_{3,b}(x)^{(3)}$, $x^{(3)} \in B^0_3$ będący składnikiem p i e r w o t n e j p o s t a c i $W_{1,2}$, dla którego $W_1 = 0$ i jednocześnie $W_{1,2} = 1$ może być wykreślony z wyrażenia $W_{1,2}$ jako zbędny. Wyrażenie W_1 na tym etapie redukcji może być brane w postaci uproszczonej, uzyskanej po pierwszym i drugim etapie uproszczeń.

Słuszność tego twierdzenia wynika bezpośrednio z postaci dyskutowanego wyrażenia

$$((W_1)'_1 + W_{12})'_2$$

oraz podanej wyżej tabelki /przypadek 3/.

Twierdzenie 1 należy odnosić w sposób analogiczny do wszystkich dalszych wyrażeń określonych we wstępie tego punktu.

5. Praktyczne zastosowanie przedstawionej metody syntezy.

Przedstawiona metoda syntezy sieci realizujących zadany układ funkcji $W_r^{(m)}(x)$ pozwala na efektywne rozwiązanie postawionego za-

dania, przy stosunkowo prostym i zmechanizowanym postępowaniu. Wyróżniamy tu trzy zasadnicze części:

1. Formułowanie problemu,
2. Poszukiwanie najprostszej postaci funkcji zadanej,
3. Konstruowanie sieci realizujących zadane funkcje logiczne.

Każdą z tych części pokrótce omówimy.

5.1. Formułowanie problemu.

Prawidłowe postawienie problemu polega na określeniu zależności funkcji $W_r^{(m)}(x)$ od wszystkich elementów zbioru E^N, mianowicie należy podać:

a/ wszystkie elementy x, dla których funkcja $W_r^{(m)}(x)$ jest równa 1,

b/ wszystkie elementy x, dla których funkcja $W_r^{(m)}(x)$ jest równa 0,

c/ jeżeli wskaźnik $m > 1$, należy określić zależności wszystkich funkcji, zgodnie z /a/ i /b/.

W przypadku, gdy występują elementy zbioru E^N, na których $W_r^{(m)}(x)$ nie jest określona, to nie jest konieczne wskazywanie tych elementów. Przedstawiona metoda automatycznie uwzględnia te przypadki z punktu widzenia uproszczenia zadanej funkcji. Stawianie problemu może być w formie tabel zerojedynkowych, jak również w formie symbolicznej, na przykład jako sumoiloczyny zmiennych negowanych i nienegowanych $x_1, x_2, \ldots, x_N$.

5.2. Poszukiwanie najprostszej postaci funkcji zadanych.

Po sformułowaniu problemu realizujemy kolejno następujący ciąg kroków:

a/ określamy typ funkcji odpowiadający jednemu z czterech wzorów

/11, /12/, /13/, /14/; o czym decyduje $W_r^{(m)}(x) = 0$ dla $x = 0$ i wartość N, tj. parzystość lub nieparzystość N.

b/ Tworzymy wyrażenia typu:

1. $(f_1^k + f_2^k)$ $k = 1$, albo $k = 0$
2. $(f_2^k + f_3^k)$ $k = 0$, albo $k = 1$
3. $(f_3^k + f_4^k)$ $k = 1$, albo $k = 0$

i t.d.

c/ Przeprowadzamy redukcję zbędnych wyrazów w poszczególnych wyrażeniach wymienionych w punkcie b, najpierw zgodnie z wzorem /16/ następnie zgodnie z wzorem /18/.

d/ Przeprowadzamy redukcję zbędnych wyrazów w ramach otrzymanej funkcji zgodnie z twierdzeniem 1.

e/ Wypisujemy ostateczną postać funkcji $W_{r_o}^{(m_o)}(x)$.

Uwaga.

W przypadku, gdy jest zadany układ M funkcji tych samych N zmiennych, wtedy dla każdej funkcji z osobna realizujemy wszystkie kroki określone przez /a/, /b/, /c/, /d/ i /e/.

5.3. Konstrukcja sieci realizujących zadane funkcje logiczne.

Ogólna postać sieci przedstawiona jest w rozdziale 3. Rozróżniamy cztery typy sieci o nazwach SW1, SW2, SW3 i SW4. Wybór typu sieci zależy od typu realizowanej funkcji, o czym mówiliśmy w rozdziale 2 i rozdziale 5 /pkt. 5.2.a./. Każdy typ sieci jest konstruowany na identycznych zasadach. Pierwsza warstwa negatorów realizuje iloczyny logiczne na zasadzie negacji sumy argumentów negowanych. Dlatego na wejściu tych negatorów wprowadza się wyłącznie argumenty zanegowane. Druga warstwa tworzy łańcuch szeregowo połączonych negatorów realizujących negację sumy iloczynów pochodzących z warstwy pierwszej. Konstrukcja ta wiernie odpowiada postaci realizowanej funkcji.

W przypadku, gdy mamy układ funkcji tych samych zmiennych, wtedy konstruujemy jedną sieć dla całego układu zadanych funkcji. Polega to na tym, że jeżeli poszczególne funkcje mają te same funktory iloczynu logicznego, a nawet całe człony, to w tej wspólnej sieci realizujemy tylko jeden funktor lub człon dla tych funkcji. Dzięki temu osiąga się dalsze zasadnicze uproszczenie sieci, co ma duże znaczenie praktyczne.

6. Podsumowanie.

Praca stanowi zamkniętą całość; przedstawiona metoda syntezy pozwala konstruktorowi w sposób mechaniczny rozwiązywać sieci realizujące zadane funkcje logiczne.

Metoda minimalizacji postaci zadanej funkcji prowadzi do uproszczenia konstrukcji sieci. Efektywność upraszczania można porównywać z innymi znanymi metodami np. przedstawioną w pracy [2].

Rozwiązano kilkanaście przypadkowo dobranych układów funkcji metodą wyżej opisaną i metodą [2]. Okazało się, że w większości przypadków otrzymane sieci przy pomocy omawianej metody były mniejsze od sieci otrzymanych metodą [2]; były również przypadki odwrotne. Należy podkreślić, że przedstawiona metoda nie nakłada ograniczeń na ilość zmiennych i na wielkość układu. Poważną wadą tej metody jest to, że otrzymywana sieć zawiera więcej szeregowo połączonych negatorów, niż sieć otrzymywana przy pomocy innej metody, np. [2].

Wydaje się, że przedstawioną metodę syntezy można będzie łatwo w pełni sformalizować poprzez podanie algorytmu pozwalającego zaprogramować ją na maszynę cyfrową.

Ze względu na zunifikowaną postać konstrukcji otrzymywanych sieci, przedstawioną metodę będzie można rozszerzyć na metodę projektowania przestrzennego z uwzględnieniem danych technicznych podzespołów i zespołów oraz zastosować maszynę cyfrową do projektowania konstrukcji. Oczywiście najpierw trzeba opracować odpowiednie algorytmy i programy.

Wymienione wyżej nowe problemy w odniesieniu do tej pracy pozostawiam jako otwarte.

Pragnąłbym tą drogą podziękować dr S. Majerskiemu, mgr inż. S. Waligórskiemu i mgr J. Wierzbowskiemu za cenne wskazówki odnośnie formy i treści tej pracy.

DODATEK

1. Przykłady syntezy sieci realizujących funkcje $W_r^{(m)}(x)$

Rozpatrzmy kilka przykładów zastosowania przedstawionej metody do syntezy sieci realizujących funkcje logiczne. Zadane funkcje niech będą sformułowane za pomocą tablic zerojedynkowych.

Tablica 3

Lp.	Zmienne				Funkcje $W_r^{(m)}(x)$			
	a	b	c	d	$W^{(1)}$	$W^{(2)}$	$W^{(3)}$	$W^{(4)}$
0	0	0	0	0	0	1	1	0
1	1	0	0	0	1	0	0	0
2	0	1	0	0	–	1	1	0
3	1	1	0	0	0	–	1	1
4	0	0	1	0	1	0	–	–
5	1	0	1	0	0	0	0	1
6	0	1	1	0	1	–	1	1
7	1	1	1	0	–	1	0	0
8	0	0	0	1	0	1	–	–
9	1	0	0	1	1	0	0	0
10	0	1	0	1	–	–	1	0
11	1	1	0	1	0	0	0	1
12	0	0	1	1	1	1	–	–
13	1	0	1	1	0	1	1	1
14	0	1	1	1	1	–	1	1
15	1	1	1	1	0	1	0	0

Przeprowadzimy kolejno syntezę dwóch sieci realizujących funkcje $W^{(1)}$ i $W^{(2)}$. Założymy, że każda z tych funkcji będzie reali-

zowana przez odrębną sieć. Kolejność postępowania przyjmiemy zgodnie z zaleceniami przedstawionymi w punktach 5.1., 5.2. i 5.3. Kolejne kroki oznaczać będziemy przez K_k, $k = 1,2,3,...$

1.1. Synteza sieci realizującej funkcje $W_r^{(1)}(x)$.

K1. Na podstawie tabeli 3 określającej $W_r^{(1)}(x)$ stwierdzamy, że problem jest postawiony prawidłowo.

K2. Również na podstawie tabeli 3 określamy typ funkcji. Dla $W_r^{(1)}(x)$ mamy:

a/ $W_r^{(1)}(0) = 0$

b/ N - parzyste

wobec tego obowiązuje wzór /14/, na podstawie którego otrzymujemy:

$$W_4^{(1)}(x) = \left(\left(\left(f_1^1 + f_2^1\right)' + f_2^0 + f_3^0\right)' + f_3^1 + f_4^1\right)' + f_4^0\Big)'$$

K3. Wypisujemy wyrażenia $\left(f_j^k + f_{j+1}^k\right)$

$$f_1^1 + f_2^1 = a + c + bc + ad + od$$

$$f_2^0 + f_3^0 = ab + ac + abd + aod$$

$$f_3^1 + f_4^1 = bcd$$

$$f_4^0 = abcd$$

K4. Redukcja wyrazów zgodnie z /16/.
Po przeprowadzeniu redukcji otrzymujemy:

$$f_1^1 + f_2^1 = a + c$$

$$f_2^0 + f_3^0 = ab + ac$$

$$f_3^1 + f_4^1 = bcd$$

$$f_4^0 = abcd$$

K5. Redukcja wyrazów zgodnie z /18/.
Po przeprowadzeniu redukcji otrzymujemy:

$f_1^1 + f_2^1 = a + c$ nie ulega zmianie

$f_2^0 + f_4^0 = ab + ac$ nie ulega zmianie

$f_3^1 + f_4^1 = bcd + \underline{bc} = bc$

$f_4^0 = abcd + \underline{abc} = abc$

Wyrazy podkreślone wprowadziliśmy zgodnie z /18/ i uzyskaliśmy wyeliminowanie po jednym argumencie.

K6. Redukcja wyrazów zgodnie z twierdzeniem 1.
Po przeprowadzeniu redukcji wg /16/ i /18/ otrzymaliśmy:

$$W_4^{(1)}(x) = ((((a + c)' + ab + ac)' + bc)' + abc)'$$

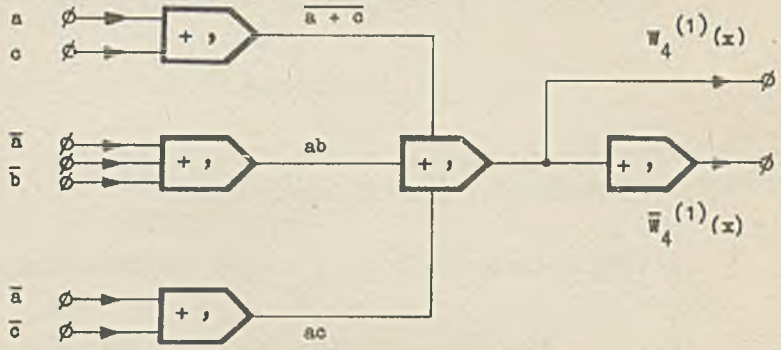
Przeprowadzając redukcję wg Tw.1 otrzymujemy:

$$W_4^{(1)}(x) = ((a + c)' + ab + ac + abc)' = ((a + c)' + ab + ac)'$$

zostaż wykreślony wyraz bc, ponieważ dla bcd = 1 wyraz

$$((a + c)' + ab + ac)' = 1.$$

K7. Konstrukcja sieci realizującej funkcję $W_4^{(1)}(x)$



Rys. 7

1.2. Synteza sieci realizującej funkcję $W_r^{(2)}(x)$.

K1. Na podstawie tablicy 3 określającej $W_r^{(2)}(x)$ stwierdzamy, że problem jest postawiony poprawnie.

K2. Określamy typ funkcji $W_r^{(2)}(x)$

a/ $W_r^{(2)}(0) = 1$

b/ N - parzyste

wobec tego obowiązuje wzór /11/, na podstawie którego otrzymujemy:

$$W_1^{(2)}(x) = \left(\left(\left(f_1^0 + f_2^0\right)' + f_2^1 + f_3^1\right)' + f_3^0 + f_4^0\right)' + f_4^1$$

K3. Wypisujemy wyrażenie $\left(f_j^k + f_{j+1}^k\right)$

$$f_1^0 + f_2^0 = a + c + ad + ac$$

$$f_2^1 + f_3^1 = cd + abc + acd$$

$$f_3^0 + f_4^0 = abd$$

$$f_4^1 = abcd$$

K4. Redukcja wyrazów zgodnie z /16/.
Po przeprowadzeniu redukcji otrzymujemy:

$$f_1^0 + f_2^0 = a + c$$

$$f_2^1 + f_3^1 = cd + abc$$

$$f_3^0 + f_4^0 = abd \qquad \text{bez zmian}$$

$$f_4^1 = abcd \qquad \text{bez zmian}$$

K5. Redukcja wyrazów zgodnie z /18/.
Po przeprowadzeniu redukcji otrzymujemy:

$$f_1^0 + f_2^0 = a + c \qquad \text{bez zmian}$$

$$f_2^1 + f_3^1 = cd + abc + bc = bc + cd$$

$$f_3^0 + f_4^0 = abd + \underline{bd} = bd$$

$$f_4^1 = abcd + \underline{abc} = abc$$

K6. Redukcja wyrazów zgodnie z twierdzeniem 1.
Po przeprowadzeniu redukcji wg /16/ i /18/ otrzymaliśmy:

$$W_1^{(2)}(x) = \left(\left(\left(a + c\right)' + bo + od\right)' + bd\right)' + abc$$

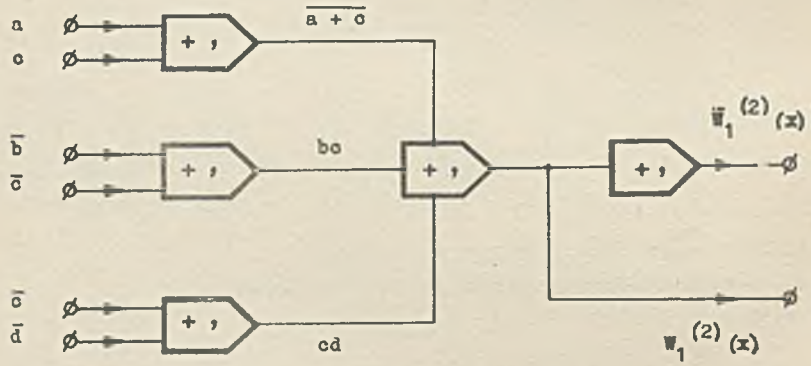
Przeprowadzając redukcję tego wyrażenia wg Tw. 1 otrzymujemy:

$$\left(\left(a + o\right)' + bo + od\right)' = 1 \quad \text{dla} \quad abd=1$$

wobec tego wyraz bd redukujemy.
Ostateczne otrzymujemy:

$$W_1^{(2)}(x) = \left(\left(a + o\right)' + bo + cd + abc\right) = \left(a+o\right)' + bo + od$$

K7. Konstrukcja sieci realizującej $W_1^{(2)}(x)$.



Rys. 8

1.3. Synteza sieci realizującej funkcje $W_r^{(3)}(x)$ i $W_r^{(4)}(x)$

Załóżymy, że zadane funkcje zależą od tych samych zmiennych, tzn. że sieci mogą mieć ogłony wspólne.

K1. Na podstawie tablicy 3 stwierdzamy, że problem jest postawiony poprawnie.

K2. Na podstawie tablicy 3 określamy typy zadanych funkcji $W_r^{(3)}(x)$ i $W_r^{(4)}(x)$

Dla $W_r^{(3)}(0) = 1$; N - parzyste

Dla $W_r^{(4)}(0) = 0$; N - parzyste

Wobec tego dla $W_r^{(3)}(x)$ obowiązuje wzór /11/,
natomiast dla $W_r^{(4)}(x)$ obowiązuje wzór /14/

$$W_1^{(3)}(x) = \left(\left(\left(f_1^0 + f_2^0\right)' + f_2^1 + f_3^1\right)' + f_3^0 + f_4^0\right)' + f_4^1$$

$$W_4^{(4)}(x) = \left(\left(\left(f_2^1 + f_2^1\right)' + f_2^0 + f_3^0\right)' + f_3^1 + f_4^1\right)' + f_4^1\Big)'$$

K3. Wypisujemy wyrażenia $\left(f_j^k + f_{j+1}^k\right)$

Dla $W_1^{(3)}(x)$ mamy:

$$f_1^0 + f_2^0 = a + ac + ad$$

$$f_2^1 + f_3^1 = ab + bc + bd + acd + bcd$$

$$f_3^0 + f_4^0 = abc + abd + abcd$$

$$f_4^1 = 0$$

Dla $W_4^{(4)}(x)$ mamy:

$$f_1^1 + f_2^1 = ab + ac + bc$$

$$f_2^0 + f_3^0 = abc + ad + bd$$

$$f_3^1 + f_4^1 = abd + acd + bcd$$

$$f_4^0 = abcd$$

K4. Redukcja wyrazów zgodnie z /16/.

Dla $W_1^{(3)}(x)$ otrzymujemy:

$$f_1^0 + f_2^0 = a$$

$$f_2^1 + f_3^1 = ab + bc + bd + acd$$

$$f_3^0 + f_4^0 = abc + abd$$

$$f_4^1 = 0$$

Dla $W_4^{(4)}(x)$ otrzymujemy:

$$f_1^1 + f_2^1 = ab + ac + bc \qquad \text{bez zmian}$$

$$f_2^0 + f_3^0 = abc + ad + bd \qquad \text{bez zmian}$$

$$f_3^1 + f_4^1 = abd + acd + bcd \qquad \text{bez zmian}$$

$$f_4^0 = abcd \qquad \text{bez zmian}$$

K5. Redukcja wyrazów zgodnie z /18/.

Dla $W_1^{(3)}(x)$ otrzymujemy:

$$f_1^0 + f_2^0 = a \qquad \text{bez zmian}$$

$$f_2^1 + f_3^1 = b + cd \qquad \text{dodaliśmy b+cd}$$

$$f_3^0 + f_4^0 = abc + abd \qquad \text{bez zmian}$$

$$f_4^1 = 0$$

Dla $W_4^{(4)}(x)$ otrzymujemy:

$$f_1^1 + f_2^1 = c + ab \qquad \text{dodaliśmy c}$$

$f_2^0 + f_3^0 = d + abc$ dodaliśmy d

$f_3^1 + f_4^1 = od + abd$ dodaliśmy od

$f_4^0 = abo$ dodaliśmy abo

Po dwóch etapach redukcji otrzymujemy wyrażenia:

$$W_1^{(3)}(x) = (((a)' + b + od)' + abc + abd)'$$

$$W_4^{(4)}(x) = ((((c + ab)' + d + abc)' + od + abd)' + abc)'$$

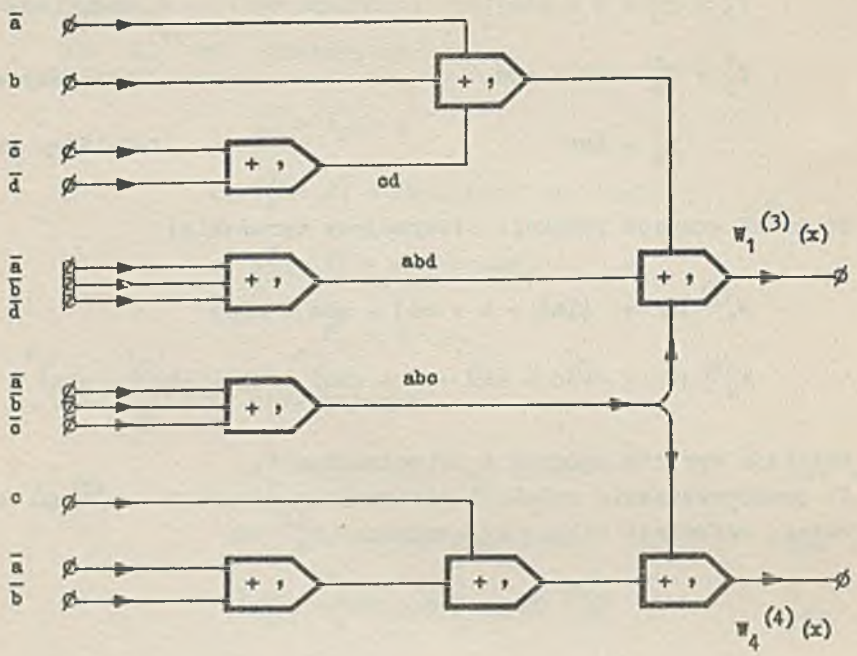
K6. Redukcja wyrazów zgodnie z twierdzeniem 1.
Po przeprowadzeniu redukcji otrzymujemy wyrażenie $W_1^{(3)}(x)$ bez zmian, natomiast silnie uproszczone $W_4^{(4)}(x)$

$$W_4^{(4)}(x) = ((c + ab)' + abc)'$$

Uproszczenie uzyskaliśmy zgodnie z twierdzeniem 1 przez skreślenie następujących wyrazów:

$$W_4^{(4)}(x) = ((((c + ab)' + \not{d} + abc)' + od + a\not{b}d)' + abc)'$$

K7. Konstrukcja sieci realizujących $W_1^{(3)}(x)$ i $W_4^{(4)}(x)$
Wyrażenia te posiadają jednakowy wyraz abc, dlatego też wystarczy tylko jeden negator realizujący ten funktor.



Rys. 9. Sieć realizująca $W_1^{(3)}(x)$ i $W_4^{(4)}(x)$.

1.4. Zakończenie.

Celem sprawdzenia przydatności praktycznej przedstawionej metody syntezy sieci logicznych, rozwiązano kilkanaście wziętych przypadkowo sieci realizujących funkcje i kilka specjalnie dobranych, jako niekorzystnych dla podanej metody. Te same sieci zostały rozwiązane inną metodą, stosowaną często w praktyce, opisaną w pracy [2]. Jako kryterium porównawcze wzięto ilość negatorów w otrzymanej sieci i łączną ilość wszystkich argumentów w całej sieci. Wyniki zestawiono w poniższej tablicy. Na uwagę zasługują liczby obrazujące sumaryczną ilość negatorów i argumentów w 15-u rozwiązanych sieciach przy zastosowaniu obydwu metod. Widzimy nieznaczne różnice, które praktycznie eliminują się. Można by wyciągnąć aczkolwiek os-

Tablica 4

Lp.	Numer funkcji W^m_x	Metodą opisaną w tej pracy		Metodą opisaną w pracy [2]		Uwagi
		ilość negator.	ilość argum.	ilość negator.	ilość argum.	
1	$W^{(1)}$	9	22	7	21	Przykład wzięty przypadkowo
2	$W^{(2)}$	6	12	6	18	" "
3	$W^{(3)}$	4	7	4	7	" "
4	$W^{(4)}$	7	17	7	22	" "
5	$W^{(5)}$	5	13	5	13	" "
6	$W^{(6)}$	6	13	5	11	" "
7	$W^{(7)}$	7	17	7	22	" "
8	$W^{(8)}$	10	23	7	21	" "
9	$W^{(9)}$	10	24	6	17	Przykłady wzięte niekorz.dla metody opisanej w tej pracy
10	$W^{(10)}$	16	46	10	41	
11	$W^{(11)}$	12	34	10	33	
12	$W_4^{(1)}$	5	10	4	8	Przykł.rozwiązane w niniej.pracy, wzięte przypadkowo
13	$W_4^{(2)}$	5	10	6	16	
14	$W_1^{(3)}$	5	14	6	15	"
15	$W_4^{(4)}$	3	6	6	13	"
		110	268	96	276	Sumaryczna ilość negat. i argum.

trożny wniosek, że wielkości sieci dla większych zestawów byłyby prawie jednakowe, przy zastosowaniu porównywanych metod. Ostateczny wniosek w tej sprawie będzie można wyciągnąć z doświadczeń przeprowadzonych w większym zakresie. Należy jednak podkreślić inne zalety tej metody, polegające na prostocie zastosowania, nieograniczoności ilości zmiennych i wielkości układu funkcji tych samych

zmiennych, oraz możliwości stosunkowo łatwego zaprogramowania tej metody na maszynę cyfrową. Wadą tej metody jest stosunkowo większa ilość szeregowo łączonych negatorów w otrzymywanych sieciach, co niewątpliwie powoduje zwiększenie czasu trwania procesów przejściowych zachodzących w otrzymanej sieci.

Literatura

1. BUTLER K.J.jr., WARFELD J.N.: Digital Computer Program for Reducing Logical Statements to a Minimal Form, Proc.Natl.Electronics Cont.1959:15. 456-466.

2. Mc CLUSKEY E.J.: Minimization of Boolean Functions, The Bell System Technical Journal, November 1956.

3. QUINE W.V.: On Cores and Prime Implicants of Truth Functions, Amer.Math. Monthly 1959:66, 755-760.

4. SAWICKI Z.: Analiza i synteza sieci zakończonej elementem pamięciowym, Prace ZAM PAN, Warszawa, 1961:1, A14.

5. WALIGÓRSKI S.: Calculation of Quine's Table for Truth Functions, Prace ZAM PAN, Warszawa, 1961:2, A15.

6. WALIGÓRSKI S.: On Normal Equivalents of Truth Functions, Prace ZAM PAN, Algorytmy, Warszawa, 1962:1,11.

7. WALIGÓRSKI S.: Projekt logiczny arytmometru i sterowania ABC, Biul. ZAM, Ser. D, Warszawa, 1958:1.

8. WALIGÓRSKI S.: Calculation of Prime Implicants of Truth Function, Algorytmy, Warszawa, 1963:2.

9. WALIGÓRSKI S.: On Superpositions of Zero-One Functions, Algorytmy, Warszawa, 1963:2.

10. ROSENTHAL C.W.: Computing Machine Aids to a Development Project, IRE Transactions on Electronic Computer, September 1961.

11. LEINER A.L., WEINBERGER A., COLEMAN C., LOBERMAN H.: Using Digital Computers in the Design and Maintenence of New Computer, IRE Transactions of Electronic Computer, December 1961.

A CERTAIN METHOD OF LOGICAL DESIGN.

Summary

The subject of the paper is a synthesis of a network that realizes zero-one functions $W(x)$. The method is based on an auxiliary function f_j^k which constitutes the basis for forming functions $W(x)$, as well as for constructing the network that realizes these functions. The paper also comprises a method of minimizing the form of functions $W(x)$.