3.2 Układy kombinacyjne

1. Funkcje boolowskie

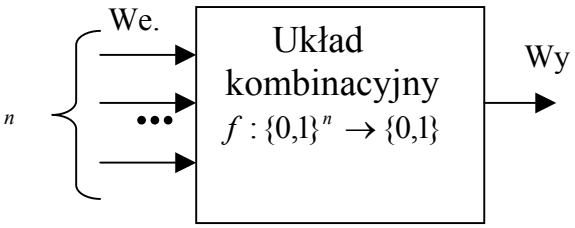
Definicja: *Funkcja boolowska (n-zmiennych)* (ang. Boolean function) to każda funkcja postaci $f:\{0,1\}^n \to \{0,1\}$, a układ (na ogół układ elektroniczny) realizujący funkcję boolowską nazywamy *układem kombinacyjnym*.

Funkcję boolowską nazywamy również *funkcją logiczną* lub *funkcją przełączającą*. Funkcja boolowska jest też definiowana nieco ogólniej jako funkcja postaci $f:D \to \{0,1\}^m$, gdzie $D \subset \{0,1\}^n$ jest niepustym podzbiorem zbioru wszystkich słów binarnych binarnych długości n.

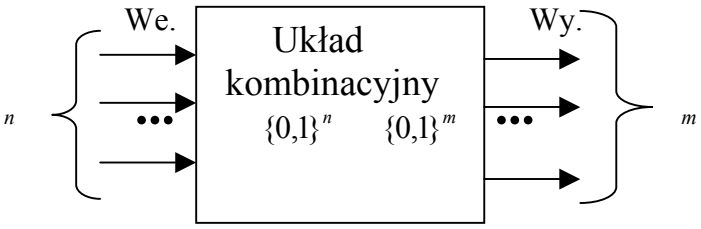
Jeśli mówimy „funkcja boolowska n argumentowa" nie precyzując D to przyjmujemy, że $D = \{0,1\}^n$. Jeśli $D = \{0,1\}^n$ to funkcję boolowską nazywamy *zupełną*. Jeśli D jest właściwym podzbiorem zbioru $\{0,1\}^n$, to funkcję boolowską nazywamy *niezupełną*. Jeśli $m > 1$, to funkcję boolowską nazywamy wektorową.

Na ogół mówiąc funkcja boolowska będziemy mieć na myśli funkcję $f:\{0,1\}^n \to \{0,1\}$.

Zauważmy, że nawet dla małych n liczba wszystkich funkcji boolowskich jest duża. Jest to bowiem liczba wariancji z powtórzeniami zbioru 2^n elementowego ze zbioru 2-elementowego, czyli 2^{2^n}. Dla n=2 mamy 16 funkcji ale dla n=5 już 2^{32} (to więcej niż 100 milionów).



Rys.1. Układ kombinacyjny o *n* wejściach i jednym wyjściu (układ jednowyjściowy) realizujący funkcję boolowską $f:\{0,1\}^n \to \{0,1\}$



Rys. 2. Układ kombinacyjny o *n* wejściach i *m* wyjściach (układ wielowyjściowy) realizujący funkcję boolowską $f:\{0,1\}^n \to \{0,1\}^m$

Uwaga: Funkcje boolowskie i ogólniej nieco układy logiczne można realizować (czyli konstruować) w różny sposób. Najczęściej (w praktyce prawie wyłącznie) układy logiczne realizowane są jako układy elektroniczne ale znane i stosowane są również realizacje

mechaniczne, elektromechaniczne i pneumatyczne. Na przykład w układach automatyki przeznaczonych do pracy atmosferach wybuchowych stosuje się pneumatyczne układy logiczne.

2. Działania w algebrze Boole'a, typowe bramki podstawowe, symbole bramek

W 2 elementowej algebrze Boole'a definiujemy kilka dodatkowych użytecznych w praktyce działań. Zaczniemy od sumy modulo 2 (suma modulo 2 jest szczególnym przypadkiem sumy modulo m wprowadzanej w zbiorze $Z_m = \{0,1,...,m-1\}$). Sumę modulo 2 oznaczamy symbolem $\oplus$ i definiujemy za pomocą tabelki.

a	b	y=a⊕b
0	0	0
0	1	1
1	0	1
1	1	0

Rys. 3. Tabelka definiująca sumę modulo 2

Podstawowe własności sumy modulo 2 są następujące

$0 \oplus 0 = 0, \quad 1 \oplus 1 = 0, \quad 1 \oplus 0 = 1, \quad 0 \oplus 1 = 1, \quad x \oplus x = 0$

$x \oplus 1 = \overline{x}, \quad x \oplus 0 = x, \quad x_1 \oplus x_2 = \overline{x_1} \oplus \overline{x_2}, \quad x_1 \oplus x_2 = x_1\overline{x_2} + \overline{x_1}x_2$

$(x_1 \oplus x_2) \oplus x_3 = x_1 \oplus (x_2 \oplus x_3)$ (łączność sumy modulo 2)

$x_1 \oplus x_2 = x_2 \oplus x_1$ (przemienność sumy modulo 2)

$x_1 \oplus x_2 \oplus x_2 = x_1$

Przypomnijmy (por. rozdz.1), że zbiór $Z_m = \{0,1,2,...,m-1\}$ wraz z działaniami dodawania modulo m i mnożenia modulo m jest pierścieniem przemiennym z jednością a jeśli $m = p$, (gdzie p jest liczbą pierwszą) to Z_p jest ciałem. Zatem w szczególności $Z_2 = \{0,1\}$ z dodawania działaniami modulo 2 i mnożenia modulo 2 jest ciałem.

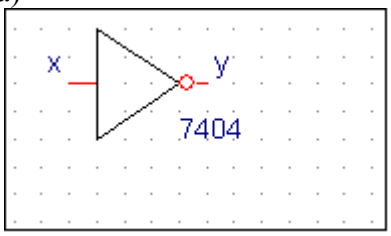
Łatwo sprawdzić, że 2 argumentowa suma modulo 2 zdefiniowana tabelką z Rys. 2.3 i suma modulo 2 zdefiniowana jako szczególny przypadek sumy modulo m (dla m=2) to to samo działanie. Z kolei iloczyn modulo 2 zdefiniowany jako szczególny przypadek sumy modulo m (dla m=2) to mnożenie logiczne. Zatem trójka uporządkowana $(Z_2, \oplus, \cdot)$ jest ciałem. Zatem wprowadzenie w algebrze Boole'a sumy modulo 2 jako działania 2 argumentowego pozwala spojrzeć na zbiór $\{0,1\}$ jak na ciało.

Z algebry liniowej wiadomo (jest to prosty do sprawdzenia fakt), że jeśli K jest ciałem, to iloczyn kartezjański $\underbrace{K \times K \times ... \times K}_{n} = K^n$ jest przestrzenią liniową nad ciałem K. Zatem

również $\underbrace{Z_2 \times Z_2 \times ... \times Z_2}_{n} = Z_2^n$ jest przestrzenią liniową nad $Z_2 = \{0,1\}$. Można więc słowa binarne o długości n (czyli elementy przestrzeni Z_2^n) nazywać wektorami.

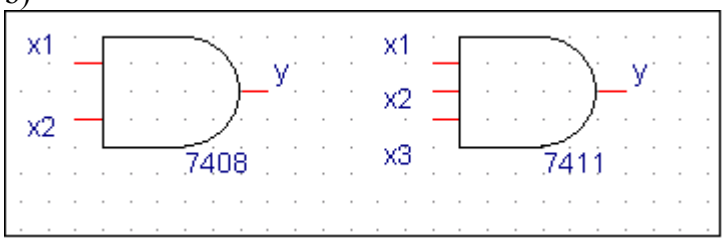
Podobnie jeśli mamy ustalone dowolne ciało K to można rozważać pierścień wielomianów $K[x]$ nad ciałem K. Tak też można zrobić w przypadku Z_2 wprowadzając wielomiany nad ciałem $Z_2 = \{0,1\}$.

a)



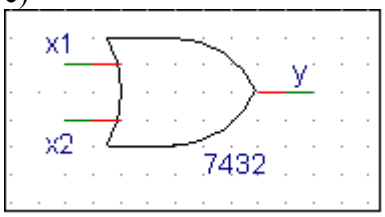
Bramka NOT realizuje funkcję $y = \overline{x}$

b)



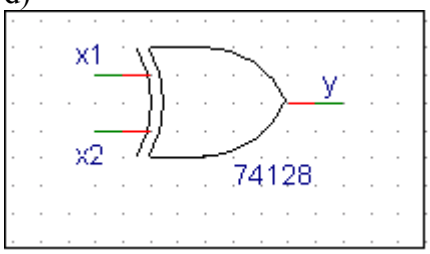
Bramka AND czyli bramka iloczynu (dwuwejściowego) realizuje funkcję $y = x_1 x_1$. Stosowane są również bramki wielowejściowe.

c)



Dwuwejściowa bramka OR czyli bramka sumy (dwuwejściowej) realizuje funkcję $y = x_1 + x_1$. Stosowane są również bramki wielowejściowe.

d)



Dwuwejściowa bramka EXOR czyli bramka sumy modulo dwa (dwuwejściowa) realizuje funkcję $y = x_1 \oplus x_1$. Stosowane są również bramki wielowejściowe EXOR. Bardzo pożyteczna jest w praktyce tożsamość $x_1 \oplus x_2 = x_1 \overline{x_2} + \overline{x_1} x_2$

Rys. 4. Symbole typowych bramek a) bramka NOT, b) bramka AND, c) bramka OR d) bramka EXOR

Uwaga: Teoretycznie rzecz biorąc możemy używać bramek OR, AND, NOR, NAND i EXOR o dowolnej liczbie wejść. Jednak w praktyce liczba wejść pojedynczej bramki rzadko większa jest od 4 ponieważ zbyt duża liczba wejść pogarsza parametry elektryczne układu, a przede wszystkim powiększa czas propagacji bramki.

W praktyce nie można też łączyć wyjścia bramki z dowolną ilością wejść współpracujących bramek. Parametr mówiący tym ile wejść można dołączyć do jednego wyjścia nosi nazwę *obciążalności bramki* lub *wzmocnienia logicznego*.

3. Postacie kanoniczne funkcji boolowskiej *n*-zmiennych

Wyrażenie boolowskie (lub *forma boolowska*) to formalnie poprawnie zbudowany opis funkcji boolowskiej f zawierający wprowadzone w algebrze Boole'a działania i nawiasy opisujące kolejność wykonywanych działań na wprowadzonych zmiennych (argumentach funkcji f). Najczęściej w tworzonych zapisach bierzemy dodatkowo pod uwagę następujący priorytet wykonywanych działań: negacja ma największy priorytet, potem mnożenie a potem dodawanie.

Iloczyn pełny n zmiennych (inaczej *minterm*) to iloczyn zawierający n różnych zmiennych zanegowanych lub nie. *Minterm* dla n zmiennych (lub dla n argumentów) jest funkcją przyjmująca wartość 1 tylko dla jednego układu argumentów $x_1, x_2, ..., x_n$.
Iloczyn niepełny n zmiennych to iloczyn zawierający $m < n$ różnych zmiennych zanegowanych lub nie.

Przykład: $f(x_1, x_2, x_3) = \overline{x_1} \cdot x_2 \cdot x_3$ jest iloczynem pełnym 3 zmiennych

Suma pełna n zmiennych (inaczej *maksterm*) to suma zawierająca n różnych zmiennych zanegowanych lub nie. *Maksterm* dla n zmiennych (lub dla n argumentów) jest funkcją przyjmująca wartość 0 tylko dla jednego układu argumentów $x_1, x_2, ..., x_n$.
Suma niepełna n zmiennych to iloczyn zawierający $m < n$ różnych zmiennych zanegowanych lub nie.

Przykład: $f(x_1, x_2, x_3) = \overline{x_1} + x_2 + x_3$ jest iloczynem pełnym 3 zmiennych.■

Postać normalna sumacyjna (inaczej *postać normalna dysjunkcyjna*) funkcji boolowskiej n zmiennych f to opis funkcji w postaci sumy iloczynów pełnych lub niepełnych.

Postać normalna iloczynowa (inaczej *postać normalna koniunkcyjna*) funkcji boolowskiej n zmiennych f to opis funkcji w postaci iloczynu sum pełnych lub niepełnych.

Implikant. Funkcja boolowska $g: \{0,1\}^n \to \{0,1\}$ implikuje inną funkcję boolowską $f: \{0,1\}^n \to \{0,1\}$ jeśli dla każdego układu $x = (x_1, x_2, ..., x_n) \in \{0,1\}^n$ takiego, że $g(x) = 1$ mamy $f(x) = 1$. Funkcję g nazywamy *implikantem* funkcji f.

Przykład: Niech $f(x_1, x_2) = \overline{x_1} + x_1 \cdot x_2$. funkcja $g(x_1, x_2) = \overline{x_1}$ jest implikantem funkcji $f(x_1, x_2) = \overline{x_1} + x_1 \cdot x_2$

Przykład 2. Jeśli funkcja boolowska f przedstawiona jest w postaci normalnej sumacyjnej, to każdy iloczyn lub suma iloczynów występujących w tym przedstawieniu jest implikantem tej funkcji.■

Implikant prosty funkcji boolowskiej n zmiennych zapisanej w postaci normalnej sumacyjnej to iloczyn m różnych zmiennych (ewentualnie zanegowanych), $m \le n$ będący implikantem funkcji f posiadający tę własność, że po usunięciu jakiejkolwiek zmiennej z tego iloczynu tak powstała funkcja przestaje być implikantem funkcji f.

Dla każdej funkcji boolowskiej spełniona jest oczywista (proste sprawdzenie) ale pożyteczna równość:

$$f(x_1, x_2, ..., x_n) = \overline{x_1} f(0, x_2, ..., x_n) + x_1 f(1, x_2, ..., x_n)$$

Jest to tzw. *wzór na rozłożenie sumacyjne funkcji boolowskiej*.

Podobnie uzyskujemy wzór

$$f(x_1, x_2, ..., x_n) = (\overline{x_1} + f(1, x_2, ..., x_n))(x_1 + f(0, x_2, ..., x_n))$$

Jest to tzw. *wzór na rozłożenie iloczynowe funkcji boolowskiej*.

Stosując wielokrotnie (n krotnie) wzór na rozłożenie sumacyjne funkcji boolowskiej otrzymujemy następujące twierdzenie o postaci kanonicznej sumacyjnej (lub dysjunkcyjnej)

Twierdzenie (o postaci kanonicznej sumacyjnej):
Każdą funkcję boolowską $f : \{0,1\}^n \to \{0,1\}$ można przedstawić jednoznacznie w postaci sumy implikantów prostych

$$f(x_1, x_2, ..., x_n) = \sum_{k=0}^{2^n - 1} \alpha_k \cdot I_k(x_1, x_2, ..., x_n) \qquad (*)$$

gdzie $\alpha_k = f(k_{n-1}, k_{n-2}, ..., k_0)$ a $k_{n-1}k_{n-2}...k_0$ jest n bitowym zapisem NKB liczby $k \in <0, 2^n - 1>$. $I_k(x_1, x_2, ..., x_n)$ jest (n-argumentowym) mintermem o numerze k, tzn. funkcją postaci $I_k(x_1, x_2, ..., x_n) = \overline{x_1} \cdot x_2 \cdot ... \cdot x_n$, gdzie negacje lub ich brak odpowiada liczbie k w taki sposób: umieszczamy ciąg bitów $k_{n-1}k_{n-2}...k_0$ nad zmiennymi $x_1, x_2, ..., x_n$ i tam gdzie nad zmienną stoi 0, wpisujemy negację, a tam gdzie stoi 1, nie wpisujemy negacji.

Równość (*) to tzw. *kanoniczna postać sumacyjna* (*dysjunkcyjna*) funkcji boolowskiej $f(x_1, x_2, ..., x_n)$.

Podobnie jak wprowadzamy kanoniczną postać sumacyjną można wprowadzić kanoniczną postać iloczynowi (koniunkcyjną) i wykazać twierdzenie o postaci kanonicznej iloczynowej.

4. Układy funkcjonalnie pełne

Układy funkcjonalnie pełne (lub jak czasem mówimy *systemy funkcjonalnie pełne*) to takie zbiory działań, (zestawy bramek) z których możemy zbudować metodą superpozycji (łączenia bramek) dowolną funkcję boolowską.

Przykłady:
{negacja, suma logiczna, iloczyn logiczny} czyli {NOT, OR, AND}
{negacja, suma logiczna} czyli {NOT, OR}
{negacja, iloczyn logiczny} czyli {NOT, AND}
{NAND 2 wejściowy}
{NOR 2 wejściowy}

Dowody, że powyższe zbiory są układami funkcjonalnie pełnymi sprowadzają się do przedstawienia za pomocą poszczególnych działań z danego zbioru działań:

sumy, iloczynu i negacji

Korzystamy w tym celu z prawa podwójnego przeczenia i praw de Morgana. Z kolei funkcjonalna pełność zbioru działań {negacja, suma logiczna, iloczyn logiczny}wynika z twierdzenia o postaci kanonicznej sumacyjnej funkcji boolowskiej. ■

Można wykazać, że żadne inne działanie dwuargumentowe poza NAND i NOR nie wystarcza do zdefiniowania wszystkich działań jedno i dwuargumentowych. Inaczej mówiąc {NAND 2-wejściowy}i {NOR 2-wejściowy}to jedyne układy funkcjonalnie pełne złożone z jednego działania 2 argumentowego.

5. Zadawanie funkcji boolowskiej

Tabelka prawdy. Najprostszym pojęciowo opisem funkcji boolowskiej jest tabelka prawdy czyli bezpośrednie podanie wartości funkcji dla wszystkich możliwych układów argumentów. Dla większej liczby argumentów jest to jednak metoda kłopotliwa. Wyobraźmy sobie bowiem tabelkę dla funkcji boolowskiej 16 czy 32 argumentowej.

Przykład: Rozważmy funkcję boolowską $f:\{0,1\}^3 \to \{0,1\}$ zadaną wyrażeniem boolowskim

$$f(x_1,x_2,x_3)=\overline{x_1}\cdot\overline{x_2}\cdot\overline{x_3}+\overline{x_1}\cdot x_2\cdot\overline{x_3}+x_1\cdot\overline{x_2}\cdot\overline{x_3}+x_1\cdot\overline{x_2}\cdot x_3+x_1\cdot x_2\cdot x_3$$

Tabelka prawdy jest dla tej funkcji następująca

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

Rys. 5. Tabelka prawdy opisująca pewną funkcję boolowską

Wyrażenie boolowskie. Wygodnie jest funkcję boolowską zadawać wyrażeniem boolowskim czyli forma boolowską.

Tablica Karnaugha. Kolejną stosowaną w praktyce metodą jest *tablica Karnaugh.* Jest to w gruncie rzeczy odmiana tabelki prawdy.

Tablica Karnaugha to często stosowany sposób opisu, zadawania funkcji boolowskiej z reguły niewielkiej liczby zmiennych 2,3,4,5. Tablice Karnaugh można również wykorzystywać do minimalizacji funkcji boolowskich (niewielkiej liczby zmiennych). Na przykład dla funkcji boolowskiej 4 zmiennych $f(x_1,x_2,x_3,x_4)=\overline{x_1}\cdot x_2\cdot x_3\cdot x_4$ konstruujemy tablicę Karnaugh następująco:

Wypisujemy wartości argumentów x_1,x_2 w pierwszej kolumnie, tak by stanowiły 2 bitowy kod Gray'a. Podobnie wypisujemy wartości argumentów x_3,x_4 w pierwszym wierszu, tak by stanowiły 2 bitowy kod Gray'a

x3x4 / x1x2	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	0	0	0	0
10	0	0	0	0

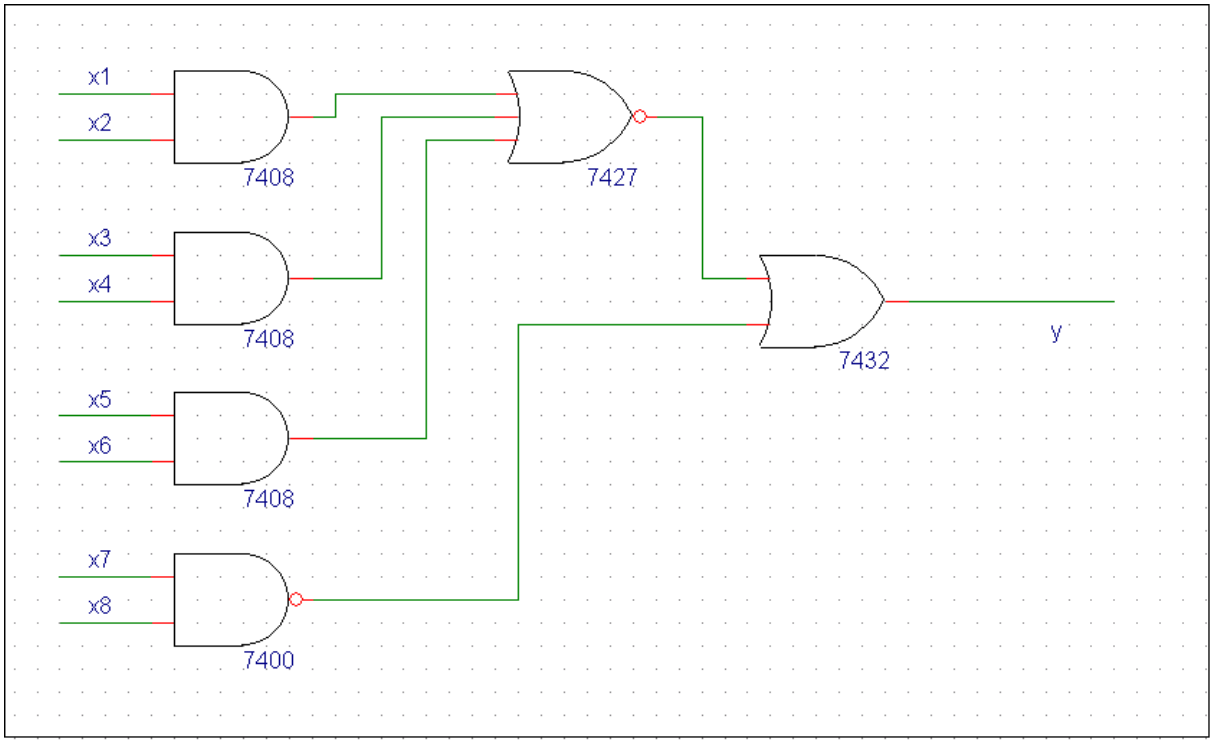
Rys. 6. Tablica Karnaugh dla funkcji $f(x_1,x_2,x_3,x_4)=\overline{x_1}\cdot x_2\cdot x_3\cdot x_4$

6. Realizacja funkcji boolowskich

Układy kombinacyjne można budować w różny sposób np.:

- bezpośrednio łącząc bramki. Z bramek NAND, NOR i OR możemy zbudować układ pokazany na Rys. 7.
- wykorzystując bloki funkcjonalne takie jak np. multipleksery, kodery, dekodery, układy PLA i układy PLD
- wykorzystując pamięci stałe ROM, PROM, EPROM

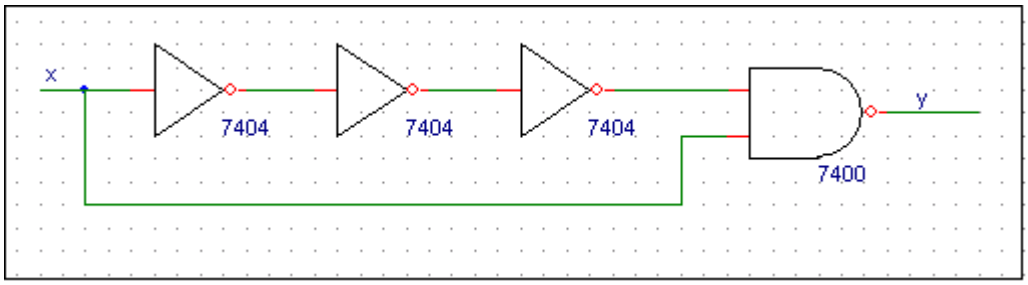
W naturalny sposób do realizacji funkcji boolowskiej można wykorzystać pamięć ROM. Nie jest to metoda oszczędna, ale umożliwia na ogół łatwą realizację i modyfikację funkcji boolowskiej np. w przypadku zastosowania pamięci typu flash.



Rys. 7. Przykład układu kombinacyjnego realizującego pewną funkcję boolowską $f : \{0,1\}^8 \rightarrow \{0,1\}$

7. Opóźnienia bramek, hazardy

Bramki rzeczywiste są układami elektronicznymi, realnymi układami fizycznymi wnoszącymi zawsze pewne opóźnienie. Typowe wartości czasu opóźnienia bramek to czasy rzędu 10ns (dla układów TTL średniej skali integracji) do 10 ps (bramki wewnątrz układu scalonego VLSI). Bramkę można więc traktować jako układ opóźniający. Prowadzić to może do powstania stanów nieustalonych układzie kombinacyjnym i do powstania fałszywych sygnałów logicznych. Zjawiska tego typu występujące w układach logicznych nazywamy hazardami. Przykład układu w którym występuje zjawisko hazardu pokazany jest na Rys. 8.



Rys. 8. Przykład układu kombinacyjnego w którym występuje zjawisko hazardu

8. Minimalizacja wyrażeń boolowskich

Minimalizacja wyrażeń boolowskich to,proces przekształcania wyrażeń boolowskich w celu otrzymania możliwie najprostszej postaci wyrażenia. Dokładniej minimalizacja wyrażenia boolowskiego to minimalizacja długości wyrażenia boolowskiego zapisującego daną funkcję boolowską. Odpowiada to minimalnej ilości dwuwejściowych bramek AND i OR realizującej daną funkcję boolowską.

Do znanych i praktycznie stosowanych metod minimalizacji dla małej liczby zmiennych należą:

- metoda tablic Karnaugh
- metoda Quine'a – McCluskey'a

Dla dużej liczby zmiennych stosuje się najczęściej program komputerowy do minimalizacji wyrażeń boolowskich o nazwie ESPRESSO. Ściśle rzecz biorąc jest kilka programów ESSPRESSO m.in. program ESPRESSO II i ESPRESSO-MV .

9. Uwagi końcowe

Powyższy rozdział traktuje jedynie o podstawach układów kombinacyjnych, więc można odnieść wrażenie, że cały aparat matematyczny z którego korzysta teoria układów kombinacyjnych to jedynie algebra Boole'a, a głównym problemem jest minimalizacja funkcji boolowskiej. Tak jednak nie jest. W bardziej zaawansowanej teorii układów kombinacyjnych korzysta się z bardzo rozbudowanego aparatu matematycznego, m.in. teorii grafów np. korzysta się z twierdzeń o kolorowaniu grafów.

Bardzo ważnymi problemami nie poruszonymi w tym rozdziale są problemy diagnostyki i testowalności układów kombinacyjnych.