Podstawy techniki cyfrowej zima 2020

Wykład

dr inż. Rafał Walkowiak

6.10.2020

Literatura

1. Układy cyfrowe, Barry Wilkinson, WKiŁ 2001
2. Podstawy projektowania układów logicznych i komputerów, M.M.Mano, Ch.R.Kime, WNT 2007
3. Komputerowe projektowanie układów cyfrowych, T.Łuba, B.Zbierzchowski, WKiŁ, 2000
4. Podstawy projektowania układów cyfrowych, Cezary Zieliński, PWN 2012
5. Język VHDL: projektowanie programowalnych układów logicznych, Kevin Shakill, WNT 2004
6. Układy cyfrowe, Zbiór zadań z rozwiązaniami, J.Tyszer, G.Mrugalski, Wydawnictwo PP

7. Układy Scalone TTL w systemach cyfrowych, J. Pienkos, J. Turczyński, Wkił, 1994

Zakres przedmiotu

- Wstęp: arytmetyka binarna, algebra Boole'a , kody binarne, BCD, podstawowe funkcje logiczne, sposoby przedstawiania funkcji logicznych - postacie kanoniczne, minimalizacja funkcji logicznych, łączna minimalizacja funkcji logicznych, hazard.
- Technologie CMOS,TTL i ich wpływ na właściwości użytkowe układów, bramki logiczne.
- Układy kombinacyjne: multipleksery i demultipleksery; komparatory, łączenie komparatorów; kodery, dekodery, translatory kodów; sumatory: sumatory binarne, dziesiętne.
- Podstawowe elementy sekwencyjne: zatrzask RS, zatrzask D, przerzutniki: D, JK, T; parametry czasowe, rejestry szeregowe, równoległe, przesuwne, rejestry liczące.
- Liczniki: synchroniczne i asynchroniczne, binarne, dziesiętne; łączenie liczników, synteza liczników, skracanie liczników, taktowanie systemów cyfrowych, częstotliwości maksymalne liczników;
- Automaty synchroniczne: Moora, Mealego, graf i tablica przejść automatu, minimalizacja stanów, kodowanie stanów, funkcje przejść i wyjść i implementacja automatu na przerzutnikach.
- Język opisu sprzętu VHDL : jednostki projektowe, obiekty, typy, typy rozstrzygalne, instrukcje współbieżne i sekwencyjne, komponenty, strukturalny i behawioralny opis układów, przykładowe realizacje układów kombinacyjnych, sekwencyjnych, automatów.
- Układy programowalne: ROM, PLD, PLA, PAL, FPGA.
- Synteza wyższego poziomu: implementacja układów cyfrowych dla realizacji algorytmów przetwarzania danych; , opisy układu: sieć działań algorytmu, diagram synchronicznego układu sekwencyjnego, diagram synchronicznego układu sekwencyjnego ze zintegrowaną ścieżką danych; projekt: schemat strukturalny, opis układu cyfrowego w języku opisu sprzętu.
- Układy mikroprogramowalne w sterowaniu układami cyfrowymi.
- Pamięci: statyczne i dynamiczne, RAM, CAM, łączenie pamięci, parametry, cykle zapisu i odczytu.
- Współpraca układów cyfrowych z otoczeniem; wprowadzanie i wyprowadzanie danych, wyświetlanie statyczne i dynamiczne.
- Sposoby organizacji systemów cyfrowych: iteracja w czasie i przestrzeni.
- Automaty asynchroniczne, minimalizacja liczby stanów i kodowanie stanów, przykłady implementacji.

Podstawowe informacje organizacyjne

- Zaliczenie: 2 sprawdziany z materiału ćwiczeń i egzaminu w trakcie semestru oraz egzamin
- Ocena - średnia ważona (2 składniki – ćwiczenia, 3 składniki – egzamin).
- Sprawdziany z określonego materiału ćwiczeń i wykładów w trakcie semestru.
- Egzamin w sesji.
- Warunki konieczne do zwolnienia z egzaminu : bardzo dobre zaliczenie ćwiczeń i min 80 % obecności na ćwiczeniach i wykładach ze sprawdzaną obecnością.

Systemy cyfrowe

- System cyfrowy – to układ powiązanych ze sobą elementów projektowany w celu realizacji takich zadań jak:
  - przetwarzanie informacji (w tym obliczenia)
  - sterowanie urządzeniami i innymi systemami i obiektami (np. silniki, zawory, piece itp.)
- Przetwarzane informacje zapisane są za pomocą wartości z określonego ograniczonego zbioru (np. liczb w różnych systemach liczbowych).

Systemy liczbowe

Pozycyjne systemy liczbowe – np. dziesiętny, dwójkowy (czyli binarny), ósemkowy, szesnastkowy – systemy o podstawie zliczania 10, 2, 8 lub 16 zawierają zbiór cyfr o liczności równej tzw. podstawie systemu.

W pozycyjnym systemie liczbowym znaczenie cyfry zależy od miejsca w zapisie liczby.

W systemie niepozycyjnym stałe jest znaczenie cyfry np. system rzymski – XLV

W zapisie pozycyjnym występują wagi pozycji od najmniej znaczącej do najbardziej znaczącej.

W stosowanych najczęściej w układach cyfrowych systemach liczbowych, podstawa może mieć wartość **2, 8, 10, 16.**
Mówimy wtedy odpowiednio o zapisie (systemie liczbowym):
dwójkowym (binarnym) (ang. **binary**), cyfry 0,1
ósemkowym (oktalnym) (ang. **octal**), cyfry 0,1,2,3,4,5,6,7
dziesiętnym (decymalnym) (ang. **decimal**),
szesnastkowym (heksadecymalnym) (ang. **hexadecdimal**), cyfry 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F.

Systemy liczbowe

NKB – naturalny kod binarny (typ kodu binarnego)

- System pozycyjny o podstawie systemu równej 2
- Liczby określone są bez znaku – tylko wartości dodatnie np. 1010 to dziesiętnie 10 poszczególne bity to $b_1 = b_3 = 1$ $b_2=b_0=0$, cyfra (bit) b_0 to cyfra (bit) najmniej znacząca, cyfra b_3 to cyfra najbardziej znacząca.
- Wartość dziesiętna liczby binarnej (N- długość słowa kodowego, b_i – cyfra pozycji i) = ∑ (i=0, N-1) $2^i \cdot b_i$
- Wartość cyfry b_i zależy zatem od jej pozycji w liczbie
- wartość $b_i = 2^i$ (i numerowanie od zera)
- 2^N różnych wartości kodu (NKB to kod pełny – wszystkie zapisy zerojedynkowe są wykorzystane)

Konwersja liczb całkowitych między systemem binarnym i dziesiętnym.

Systemy liczbowe uzupełnieniowe

System liczbowy „Uzupełnienie do K” (uzupełnienie do podstawy zliczania K)

Uzupełnienie do K liczby **N** o **n** cyfrach zapisanej w systemie o podstawie K definiujemy jako liczbę równą:

$$K^n – N$$

Zatem liczba w kodzie **Uzupełnienie do K** jest reprezentowana przez wartość różnicy K^n i N – wartość ta jest wyrażona w systemie o podstawie K (czyli za pomocą K cyfr)

Liczba w systemie uzupełnienie do K służy do zapisu liczb dodatnich.

Np. ciąg 345 to może być liczba dziesiętna trzycyfrowa K=10 n=3

liczba z systemu dziesiętnego 345 zapisana w systemie uzupełnienie do 10 może mieć postać 655,

liczba 345 w systemie dziesiętnym: 345=000345 (zera nieznaczące)

Liczba dziesiętna 345 w systemie uzupełnienie do 10: 655=99655 (dziewiątki nieznaczące)

Liczba w NKB 101 (dziesiętnie 5) w systemie **uzupełnienie do 2** (U2) ma postać 011 lub przykładowo 111011 (jedynki są nieznaczące); wartość 3 (to 011) uzupełnia 5 do 8, a wartość 59 (to 111011) uzupełnia 5 do 64.

„Reprezentacja uzupełnieniowa”

- Binarna liczba **dodatnia** jest zapisywana w NKB na wystarczającej liczbie pozycji i uzupełniana zerami na pozycjach bardziej znaczących: $(3)_{10} = (011)_2 = (0011)_2$ – najstarsze zero jest bitem znaku

- Binarna liczba **ujemna** jest zapisywana:
  - w kodzie uzupełnienie do 2 i
  - poprzedzona 1 (dodatkowo jeśli potrzeba) na pozycji znaku i
  - uzupełniona jedynkami na pozycjach bardziej znaczących: $(-3)_{10} = (101)_{UZ} = (1101)_{UZ}$
  - Gdy najstarszy bit kodu U2 =1 to staje się bitem znaku i nie potrzeba umieszczać 1 przed kodem U2 dla $-8_D = 1000_{UZ}$
- **Notacja uzupełnieniowa liczb binarnych pozwala na dodawanie liczb dodatnich i ujemnych realizowane przez sumator zaprojektowany dla liczb wyrażonych w NKB.**
- **Por. Układy cyfrowe Wilkinson 1.3.2**

Reprezentacje liczb binarnych ze znakiem

- Reprezentacja liczby binarnej **znak moduł** (ZM) – najstarszy bit określa znak liczby, pozostałe bity w NKB, bit znaku 0 (liczba dodatnia) lub 1(liczba ujemna)
- **Reprezentacja uzupełnieniowa** liczby binarnej (RU) korzysta z U2
  - bit znaku (0) i moduł liczby dodatniej w NKB,
  - bit znaku (1) i moduł liczby ujemnej w kodzie U2 (najbardziej znaczące jedynki można usunąć z zapisu)
  - Przykład -8, moduł 8 u2(8)=1000 -8= RU(11000)=RU(1000) 8=RU(01000)

N1	ZM(N1)	RU(N1)		N2	ZM(N2)=RU(N2)
-8	11000	1000		7	0111
-7	1111	1001		6	0110
-6	1110	1010		5	0101
-5	1101	1011		4	0100
-4	1100	1100		3	0011
-3	1011	1101		2	0010
-2	1010	1110		1	0001
-1	1001	1111		0	0000

Dodawanie liczb ujemnych wykorzystanie notacji UZ

1101	-3
1110	+(-2)
(1)1011	= -5

1101	-3
0010	+2
1111	= -1

1101	-3
0101	+5
(1)0010	= 2

Przeniesienie jest ignorowane, wynik poprawny gdy dwa przeniesienia: na najstarszy bit i z najstarszego bitu są jednakowe. Reprezentacja uzupełnieniowa pozwala na jednakowe traktowanie liczb dodatnich i ujemnych w operacji dodawania – tj. dodanie dwóch jedynek daje wynik 0 na bieżącej pozycji i zwiększa wartość pozycji następnej o 1.

Odejmowanie liczb – dodawanie liczby przeciwnej

```
   0011 (3d)
+  1011 (-5d)
-------------
   1110 (-2d)

   0101 (5d)
+  1101 (-3d)
---------------
   0010  (2d)
```

Binarna liczba ujemna – bit znaku i liczba binarna w uzupełnieniu do 2

00010110 = 22 (d)

Wyznaczenie liczby w kodzie U2 –

Metoda 1:

Etap 1: 11101001 negacja bitów

Etpa 2: 11101010 dodanie jedynki = -22 (d)

Metoda 2:

Negacja bitów bardziej znaczących - starszych niż najmniej znaczący bit równy 1.

UWAGA:

100 -> 100 dodatkowy bit niepotrzebny dla -4

101-> 011 konieczny dodatkowy bit dla -5 ->1011

Odejmowanie binarne

- D – dodatnia
- U – ujemna
- D – U= D+D=D (sprawdzenie przepełnienia)
- D1 – D2 = D gdy (D1>D2) lub U gdy (D1<D2)
- U-D= U + U = U (sprawdzenie przepełnienia)

Testy poprawności wyniku dodawania:

- Wynik niepoprawny (przepełnienie, nadmiar) - gdy podczas dodawania wartości przeniesienia na najwyższą pozycję i z najwyższej pozycji są różne.
- Wynik niepoprawny gdy nieparzysta liczba jedynek 4 bitów: najstarszych bitów argumentów i wyniku oraz przeniesienia z najstarszego bitu

Dodawanie liczb a przepełnienie

0011 (3d)
+ 0011 (3d)

0110 (6d) wynik dodatni – poprawnie

1101 (-3)
+ 1101 (-3)

1010 (-6) wynik ujemny – poprawnie

0101 (5d)
+ 0101 (5d)

1010 -(6d) Wynik ujemny - niepoprawny

1011 (-5)
+ 1011 (-5)

0110 (6) wynik dodatni - niepoprawny

Kody dwójkowe niewagowe

pozycja binarna nie posiada wagi

Kod cyfra	Z nadmiarem 3	Graya	Wattsa	Johnsona	Wskaźników 7 segmentowych
0	0011	0000	0000	00000	0111111 7
1	0100	0001	0001	00001	0000110 2 6
2	0101	0011	0011	00011	1011011 1
3	0110	0010	0010	00111	1001111 3 5
4	0111	0110	0110	01111	1100110 4
5	1000	0111	1110	11111	1101101
6	1001	0101	1010	11110	1111100
7	1010	0100	1011	11100	0000111
8	1011	1100	1001	11000	1111111
9	1100	1101	1000	10000	1100111

Kody dwójkowo-dziesiętne

- Do reprezentacji cyfr dziesiętnych
- 10 cyfr dziesiętnych (0,1,2,3,4,5,6,7,8,9) zakodowanych za pomocą ciągu 4 bitów (ciąg ten dostarcza 16 kombinacji na 4 bitach) – 6 kombinacji jest niewykorzystanych.

Warianty kodów dwójkowo-dziesiętnych:

- Kody wagowe – pozycja binarna posiada przypisaną wagę
- Kody niewagowe – pozycja binarna nie posiada wagi

Kody dwójkowo-dziesiętne wagowe

kod	Naturalny NKB		Aikena		
Wagi cyfra	8421	2*421	2421	7421	84-2-1
0	0000	0000	0000	0000	0000
1	0001	0001	0001	0001	0111
2	0010	0010	0010	0010	0110
3	0011	0011	0011	0011	0101
4	0100	0100	0100	0100	0100
5	0101	0101	1011	0101	1011
6	0110	0110	1100	0110	1010
7	0111	0111	1101	1000	1001
8	1000	1110	1110	1001	1000
9	1001	1111	1111	1010	1111

Kody detekcyjne

kod	1 z 10	2 z 5	2 z 7	Bin z Bitem parzystości
Wagi-> cyfra	9876543210	niewagowy	5043210	8421 BP
0	0000000001	00011	0100001	0000 0
1	0000000010	00101	0100010	0001 1
2	0000000100	01001	0100100	0010 1
3	0000001000	10001	0101000	0011 0
4	0000010000	00110	0110000	0100 1
5	0000100000	01010	1000001	0101 0
6	0001000000	10010	1000010	0110 0
7	0010000000	01100	1000100	0111 1
8	0100000000	10100	1001000	1000 1
9	1000000000	11000	1010000	1001 0

Kody z kontrolą parzystości i ze stałą liczbą jedynek pozwalają na wykrycie pewnych błędów przy przesyłaniu słów kodowych.

Cyfry dziesiętne kodowane w NKB – kod BCD 8421

- Dziesiętny charakter informacji lecz kodowanie cyfr w NKB
- 2345 (10)= 0010 0011 0100 0101(BCD) 4 pozycje (4 bitowe) cyfr dziesiętnych
- Dodawanie liczb w kodzie BCD realizowane tak jak dodawanie liczb binarnych, lecz:
  - wystąpienie podczas dodawania liczb przeniesienia na pozycję kolejnej cyfry dziesiętnej (kolejne 4 bity) wymaga skorygowania (czyli dodania wartości 6) na tej pozycji, z której przeniesienie wystąpiło
  - wystąpienie wyniku na 4 bitach (jednaj pozycji cyfry dziesiętnej) spoza zakresu (10-15) wymaga skorygowania wyniku czyli dodania wartości 6 na tej pozycji cyfry dziesiętnej, która nie jest poprawna; może wystąpić przeniesienie, które należy uwzględnić na kolejnej pozycji (ale bez korekcji pozycji bieżącej), możliwa również propagacja przeniesienia np. dla liczb 3456 +6545.

Dodawanie w kodzie BCD

```
  89            1000 1001
 +18    +       0001 1000
-----   ----------------------
 107            1010 0001 przeniesienie -> korekcja
                     0110
        ----------------------
                1010 0111 wartość spoza przedziału ->korekcja
                0110
        ----------------------
              1 0000 0111      przeniesienie bez konieczności korekcji
```

Por. Układy cyfrowe Wilkinson 1.3.3

Kody alfanumeryczne

- Kody służące do kodowania **znaków** w systemach cyfrowych, w urządzeniach współpracujących z komputerem, np. drukarki, ekrany alfanumeryczne.
- Przykładami kodów alfanumerycznych są kody: ASCII ISO-7, ISO 8859, Unicode, Windows-1250.
- Kod ASCII – ISO-7 7 bitowy – pełny zbiór zawiera 128 znaków, pierwsze 33 znaki służą do sterowania systemem drukowania lub wyświetlania, pozostałe znaki to: duże i małe litery, cyfry, znaki przestankowe i inne.

Kod ISO-7

Kod ISO-7 Tablica 2.11

b_4 b_3 b_2 b_1 \ b_7 b_6 b_5	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
0 0 0 0	NUL	DLE	SPACE[2]	0[2]	@[1]	P	\[1]	p
0 0 0 1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	STX	DC2	„	2	B	R	b	r
0 0 1 1	ETX	DC3	≠	3	C	S	c	s
0 1 0 0	EOT	STOP	S[1]	4	D	T	d	t
0 1 0 1	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	ACK	SYN	&	6	F	V	f	v
0 1 1 1	BEL	ETB	'	7	G	W	g	w
1 0 0 0	BS	CAN	(	8	H	X	h	x
1 0 0 1	HT	EM	)	9	I	Y	i	y
1 0 1 0	LF	SUB	*	:	J	Z	j	z
1 0 1 1	VT	ESC	+	;	K	[[1]	k	{[1]
1 1 0 0	FF	FS	”	<	L	\[1]	l	|[1]
1 1 0 1	CR	GS	—	=	M	]	m	}
1 1 1 0	SO	RS	·	>	N	^[1]	n	—[1]
1 1 1 1	SI	US	/	?	O	—	o	DEL

1) Wymienne, 2) Wyjątki dla kodu karty dziurkowanej: literze b odpowiada brak dziurek, cyfrze 0 odpowiada otwór w kolumnie 0

Objaśnienia:

NUL — bez informacji
SOH — początek nagłówka
STX — początek tekstu
ETX — koniec tekstu
EOT — koniec transmisji
EWQ — zapytanie
ACK — odpowiedź pozytywna
BEL — dzwonek
BS — ruch powrotny, cofanie
HT — tabulacja pozioma
LF — zmiana wiersza
VT — tabulacja pionowa
FF — zmiana formularza
CR — powrót wózka
SO — poza kodem
SI — w kodzie
DLE — zmiana znaczenia ciągu znaków
DC1 — sterowanie urządzeniem 1
DC2 — sterowanie urządzeniem 2
DC3 — sterowanie urządzeniem 3
STOP — stop
NAK — odpowiedź negatywna
SYN — synchronizacja
ETB — koniec transmisji bloku danych
CAN — nieważny, anulowanie
EM — koniec zapisu, koniec nośnika informacji
SUB — zastąpienie, podstawienie
ESC — przełączenie, zmiana zestawu znaków
FS — oddzielenie głównych grup
GS — oddzielenie grupy informacji
RS — oddzielenie podgrup (pozycji)
US — oddzielenie części grup
DEL — kasowanie (znak ignorowany)

Algebra Boole'a *

Narzędzie matematyki (algebra logiki) służąca do opisu i projektowania systemów cyfrowych.

Zmienne boolowskie – mogą przyjąć jedna z dwóch wartości – 0 lub 1 – są to zmienne binarne (jednobitowe)

Podstawowe funkcje algebry Boola –

- Iloczyn logiczny I (AND) – „· " ,,∩'' ,,∧'' (alternatywne oznaczenia)
- Suma logiczna LUB (OR) – ,,+'' ,,∪'' ,,∨'' (alternatywne oznaczenia)
- Negacja NIE (NOT) – „linia nad zmienną" ,, ' " (alternatywne oznaczenia)

Funkcja boolowska (logiczna, przełączająca) – jest działaniem na zmiennych boolowskich i przyjmuje wartości ze zbioru {0,1}.

Algebra Boole'a jest zgodna z następującymi postulatami:

* Literatura Wilkinson 2.3 str. 35-53

Postulaty Huntingtona (1)

Notacja: $Z = \{0,1\}$ – zbiór wartości

a, b – dowolne zmienne binarne

A1 Domknięcie działań: $a + b \subset Z$ $A \cdot B \subset Z$

A2 Elementy stałe: Istnieją takie 0 i 1 : $a+0=a$ i $a \cdot 1=a$

A3 Przemienność: $a+b=b+a$ $a \cdot b= b \cdot a$

A4 Rozdzielność: $a \cdot (b+c)=a \cdot b+a \cdot c$

$a+(b \cdot c)=(a+b) \cdot (a+c)$ rozdzielność dodawania względem mnożenia

A5 Istnienie negacji: dla a istnieje a' : $a+a'=1$ $a \cdot a'=0$

Postulaty Huntingtona (2)

Zasada dualności:

Wyrażenie dualne powstanie poprzez zamianę operatorów binarnych i stałych: $+ \rightarrow \cdot$, $\cdot \rightarrow +$, $0 \rightarrow 1$, $1 \rightarrow 0$

Wartościowania (prawda, fałsz) wyrażenia prostego i dualnego jest jednakowe.

np wyrażenie proste: $a \cdot (b+c) = a \cdot b + a \cdot c$

Wyrażenie dualne: $a + (b \cdot c) = (a+b) \cdot (a+c)$

Przekształcanie funkcji logicznych

- Dla minimalizacji postaci wyrażeń (funkcji) boolowskich służą tożsamości i twierdzenia algebry boole'a.
- Minimalizacja pozwala na uzyskanie prostszej, tańszej implementacji funkcji – tańsza implementacja ma mniej składników oraz/lub składniki prostsze.

Twierdzenia algebry Boole'a

- Idempotentność (łac. taki sam) –

  $a+a=a\ ,\quad a \cdot a=a$

- Jednoznaczność negacji –

  dla kazdego a istnieje tylko jeden element $\overline{a}$

- Dominacja - dla każdego a $\quad a \cdot 0 = 0 \quad a+1=1$
- Podwójna negacja –

  dla kazdego a zachodzi $a = \overline{\overline{a}}$

- Pochłanianie - $a+(a \cdot b)= a \quad a \cdot (a+b)= a$

Twierdzenia algebry Boole'a

- Uproszczenie

$$a+(\overline{a}\cdot b)=a+b \qquad \mathrm{a}\cdot(\overline{\mathrm{a}}+\mathrm{b})=\mathrm{a}\cdot\mathrm{b}$$

a(1+b)+a'b=a+b(a+a')=a+b

- Minimalizacja - $a\cdot b+a\cdot\overline{b}=a \qquad (a+b)\cdot(a+\overline{b})=a$
- Łączność – (a+b)+c=a+(b+c) (a·b) ·c=a ·(b ·c)
- Konsensus (zgoda) –

Wystarczy jedna 1 dla b i c,

wystarczy jedno 0 dla b i c

$$a\cdot b+\overline{a}\cdot c+b\cdot c=a\cdot b+\overline{a}\cdot c$$

$$(a+b)\cdot(\overline{a}+c)\cdot(b+c)=(a+b)\cdot(\overline{a}+c)$$

Prawo de Morgana

$$\overline{a + b + c + \ldots} = \overline{a} \cdot \overline{b} \cdot \overline{c} \cdot \overline{}$$

$$\overline{a \cdot b \cdot c \cdot \ldots} = \overline{a} + \overline{b} + \overline{c} + \overline{\ldots}.$$

Funkcje logiczne dwóch zmiennychi ich wartości

zmienne binarne a b

Wartości argumentów	**ab** 00	**ab** 01	**ab** 10	**ab** 11	**Równanie** funkcji	**Nazwa** funkcji	**Skrót** Nazwy
	0	0	0	0	0	Stała Zero	
	0	0	0	1	$a \cdot b$	Iloczyn logiczny	AND
	0	0	1	0	$a \cdot b'$	Zakaz przez b	
	0	0	1	1	a	Identyczna z a	
	0	1	0	0	$a' \cdot b$	Zakaz przez a	
	0	1	0	1	b	Identyczna z b	
	0	1	1	0	$(a' \cdot b)+(a \cdot b')$	Suma modulo	XOR
Wartości	0	1	1	1	$a+b$	Suma logiczna	OR
funkcji	1	0	0	0	$(a+b)'$	Negacja sumy	NOR
	1	0	0	1	$(a \cdot b)+(a' \cdot b')$	Równoważność	EQU
	1	0	1	0	b'	Negacja b	
	1	0	1	1	$a+b'$	Implikacja $b \Rightarrow a$	
	1	1	0	0	a'	Implikacja a	
	1	1	0	1	$a'+b$	Implikacja $a \Rightarrow b$	
	1	1	1	0	$(a \cdot b)'$	negacja iloczynu	NAND
	1	1	1	1	1	Stała 1	

Popularne funkcje logiczne

- Szczególnie popularne AND, OR, NAND, NOR, XOR,NOT
- XOR – wartość funkcji równa 1 dla różnych argumentów
- Zależności dla XOR (suma wyłaczna) i XNOR (równoważność):
  - $a \oplus b = a'b + b'a = (a+b)(a'+b')$
  - $(a \oplus b)' = a' \oplus b = b' \oplus a = ab + a'b' = (a'+b)(a+b')$
  - $a \oplus 1 = a'$ $\quad a \oplus 0 = a$
- Różne interpretacje logiczne wielowejściowych bramek XOR/XNOR. Najczęściej bramka wykrywa nieparzystą liczbę jedynek (XOR) lub parzystą liczbę jedynek XNOR.

System funkcjonalnie pełny - SFP

- Zbiór funkcji pozwalający na przedstawienie, wyrażenie każdej innej funkcji logicznej.
- 3 przykłady S.F.P:
  - {NAND},
  - {OR,NOT},
  - {AND,NOT},
  - {NOR}

Sposoby przedstawiania funkcji logicznych

- Tablica prawdy np.

Nr kombinacji	$x_0 x_1 x_{2***} x_{n-1}$	f
0 1 2 3 4 5 * * * 2^n -1	0 0 0 *** 0 0 0 0 *** 1 1 1 1 *** 1	Wartości funkcji

nr	Komb. Argum.	Wartość funkcji
0	00	1
1	01	1
2	10	1
3	11	0

- Nr kombinacji wejść, wartości kombinacji wejść, odpowiadające wejściu wartości na wyjściu
- Zawiera wszystkie kombinacje zero-jedynkowe zmiennych niezależnych i odpowiadające im wartości funkcji

Sposoby przedstawiania funkcji logicznych

- Tablice Karnaugha
- Kombinacji wejść odpowiada pole tablicy, w polu umieszczmy właściwą dla kombinacji wartość.
- Sąsiednie (w poziomie i pionie – także cyklicznie) pola tablicy Karnaugha odpowiadają kombinacji argumentów różniącej się jedną wartością.
- Na rysunku zapisano kombinacje wejść – nie wartości funkcji funkcji

		a	
b		0	1
	0	00	01
	1	10	11

		ba			
c		00	01	11	10
	0	000	001	011	010
	1	100	101	111	110

Tablica dla funkcji 2 i 3 zmiennych wejściowych

Reprezentacja funkcji logicznych za pomocą tablic Karnaugha

b \ a	0	1
0	0	1
1	0	1

c \ ba	00	01	11	10
0	0	1	1	0
1	0	1	1	0

dc \ ba	00	01	11	10
00	1	1	1	1
01	0	∅	0	0
11	0	∅	0	0
10	1	1	0	1

∅ - oznaczenie wartości dowolnej na wyjściu