



Układy logiczne

Technika cyfrowa i mikroprocesorowa
dr inż. Dominik Łuczak

1



Algebra Boole'a

2



Poznań University of Technology
Institute of Control, Robotics and Information Engineering

Operatory

	x	y
$\wedge$	0	1
	0	0
	1	1

	x	y
$\vee$	0	1
	0	0
	1	1

	x	y
$\rightarrow$	0	1
	0	1
	1	0

	x	y
$\oplus$	0	1
	0	0
	1	1

Figure 1. Truth tables



$x \wedge y$



$x \vee y$



$x \rightarrow y$



$x \oplus y$

Figure 4. Venn diagrams

3

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Operatory

Tablica prawdy dla negacji^[3]:

p	$\neg p$
0	1
1	0

	Schröder Peirce	Peano Russell	Hilbert	Łukasiewicz
Negacja	p'	$\sim p$	$\bar{p}$	Np

2019-10-12 dominił.Łukasiewicz@pwr.edu.pl 4

4

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Operatory

Tablica prawdy dla alternatywy

p	q	$p \vee q$
0	0	0
0	1	1
1	0	1
1	1	1

	Schröder Peirce	Peano, Russell Hilbert	Łukasiewicz
Alternatywa	$p + q$	$p \vee q$	Apq

2019-10-12 dominił.Łukasiewicz@pwr.edu.pl 5

5

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Operatory - Binegacja

A	B	A NOR B	$\overline{A \vee B}$
0	0	1	
0	1	0	
1	0	0	
1	1	0	

2019-10-12 dominił.Łukasiewicz@pwr.edu.pl 6

6

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Operator

Tablica prawdy dla alternatywy rozłącznej

p	q	$p \vee q$
0	0	0
0	1	1
1	0	1
1	1	0

EOR, EXOR, $\vee$, $\dot{\vee}$, $\vee$, $\oplus$

2019-10-12 domini.k.luczak@pwr.lodz.pl 7

7

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

negacji alternatywy wykluczającej

XNOR operation is $S = A \odot B$.

Input	Output
A B	A XNOR B
0 0	1
0 1	0
1 0	0
1 1	1

2019-10-12 domini.k.luczak@pwr.lodz.pl 8

8

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Operator

Tablica prawdy dla koniunkcji^[2]:

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

	Schröder Peirce	Peano Russell	Hilbert	Łukasiewicz
Koniunkcja	$p \cdot q$	$p \cdot q$	$p \& q$	Kpq

2019-10-12 domini.k.luczak@pwr.lodz.pl 9

9

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Operatory

Tablica prawdy dla dysjunkcji

p	q	$p \vee q$
0	0	0
0	1	1
1	0	1
1	1	1

2019-10-12 dominiak.kuczyński@pwr.edu.pl 10

10

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Operatory – wybrane zestawienie

NEGACJA $\neg$
 KONIUNKCJA $\wedge$
 ALTERNATYWA $\vee$
 ALTERNATYWA WYKLUCZAJĄCA $\oplus$

2019-10-12 dominiak.kuczyński@pwr.edu.pl 11

11

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Tożsamości

$A \wedge 0 = 0$	$A \vee 0 = A$
$A \wedge 1 = A$	$A \vee 1 = 1$
$A \wedge A = A$	$A \vee A = A$
$A \wedge \bar{A} = 0$	$A \vee \bar{A} = 1$

Przemienności

$$A \wedge B = B \wedge A$$

$$A \vee B = B \vee A$$

Boolean Expression	Description	Equivalent Switching Circuit	Boolean Algebra Law or Rule
$A + 1 = 1$	A in parallel with closed = "CLOSED"		Annulment
$A + 0 = A$	A in parallel with open = "A"		Identity
$A \cdot 1 = A$	A in series with closed = "A"		Identity
$A \cdot 0 = 0$	A in series with open = "OPEN"		Annulment
$A + A = A$	A in parallel with A = "A"		Idempotent
$A \cdot A = A$	A in series with A = "A"		Idempotent
$\text{NOT } \bar{A} = A$	NOT NOT A (Double negation) = "A"		Double Negation
$A + \bar{A} = 1$	A in parallel with NOT A = "CLOSED"		Complement
$A \cdot \bar{A} = 0$	A in series with NOT A = "OPEN"		Complement
$A + B = B + A$	A in parallel with B = B in parallel with A		Commutative
$A \cdot B = B \cdot A$	A in series with B = B in series with A		Commutative

2019-10-12 dominiak.kuczyński@pwr.edu.pl 12

12



Poznan University of Technology
Institute of Control, Robotics and Information Engineering

łączności

$$A \wedge (B \wedge C) = (A \wedge B) \wedge C$$

$$A \vee (B \vee C) = (A \vee B) \vee C$$

2019-10-12 dominiak.luczak@pwr.edu.pl 13

13



Poznan University of Technology
Institute of Control, Robotics and Information Engineering

rozdzielczości

$$A \wedge (B \vee C) = A \wedge B \vee A \wedge C$$

$$A \vee B \wedge C = (A \vee B) \wedge (A \vee C)$$

2019-10-12 dominiak.luczak@pwr.edu.pl 14

14



Poznan University of Technology
Institute of Control, Robotics and Information Engineering

$$A \wedge \bar{A} = 0$$

$$A \vee \bar{A} = 1$$

2019-10-12 dominiak.luczak@pwr.edu.pl 15

15



Poznan University of Technology
Institute of Control, Robotics and Information Engineering

$$\bar{\bar{A}} = A$$

$$\overline{\bar{A} \wedge \bar{B}} = A \wedge B$$

$$\overline{\bar{A} \vee \bar{B}} = A \vee B$$

2019-10-12 domini.k.luczak@pwr.edu.pl 16

16



Poznan University of Technology
Institute of Control, Robotics and Information Engineering

De Morgana

$$\overline{\bar{A} \wedge \bar{B}} = \bar{A} \vee \bar{B}$$

$$\overline{\bar{A} \vee \bar{B}} = \bar{A} \wedge \bar{B}$$

$$\overline{\bar{A} \wedge \bar{B}} \neq \bar{A} \wedge \bar{B} \text{ r\u00f3\u017anie}$$

$$\overline{\bar{A} \vee \bar{B}} \neq \bar{A} \vee \bar{B} \text{ r\u00f3\u017anie}$$

2019-10-12 domini.k.luczak@pwr.edu.pl 17

17



Poznan University of Technology
Institute of Control, Robotics and Information Engineering

De Morgana

$$\overline{A \wedge B \wedge C \wedge \dots \wedge D} = \bar{A} \vee \bar{B} \vee \bar{C} \vee \dots \vee \bar{D}$$

$$\overline{A \vee B \vee C \vee \dots \vee D} = \bar{A} \wedge \bar{B} \wedge \bar{C} \wedge \dots \wedge \bar{D}$$

2019-10-12 domini.k.luczak@pwr.edu.pl 18

18



Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Redukcja przez absorbcję

$$\begin{aligned} A \vee (A \wedge B) &= \\ A \wedge (1 \vee (1 \wedge B)) &= \\ &= A \end{aligned}$$

$$A \wedge (A \vee B) = A \wedge A \vee A \wedge B = A \vee A \wedge B = A$$

2019-10-12 domini.k.luczak@pwr.edu.pl 19

19



Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Redukcja

$$(A \wedge B) \vee (A \wedge \bar{B}) = A (B \vee \bar{B}) = A$$

$$\begin{aligned} (A \vee B) \wedge (A \vee \bar{B}) &= \\ (A \wedge A \vee A \wedge \bar{B}) \vee (B \wedge A \vee B \wedge \bar{B}) &= \\ &= A \end{aligned}$$

2019-10-12 domini.k.luczak@pwr.edu.pl 20

20



Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Redukcja

$$A \vee (\bar{A} \wedge B) = A \vee B$$

$$A \wedge (\bar{A} \vee B) = A \wedge B$$

2019-10-12 domini.k.luczak@pwr.edu.pl 21

21

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Częściowe zestawienie

AND operations		OR operations		INV operations	
Truth table	Laws	Truth table	Laws	Truth table	Laws
$0 \cdot 0 = 0$	$A \cdot 0 = 0$	$0 + 0 = 0$	$A + 0 = A$	$0' = 1$	$A'' = A$
$1 \cdot 0 = 0$	$A \cdot 1 = A$	$1 + 0 = 1$	$A + 1 = 1$	$1' = 0$	
$0 \cdot 1 = 0$	$A \cdot A = A$	$0 + 1 = 1$	$A + A = A$		
$1 \cdot 1 = 1$	$A \cdot A' = 0$	$1 + 1 = 1$	$A + A' = 1$		

Associative Laws		Commutative Laws		Distributive Laws	
$(A \cdot B) \cdot C = A \cdot (B \cdot C) = A \cdot B \cdot C$		$A \cdot B \cdot C = B \cdot A \cdot C = \dots$		$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$	
$(A + B) + C = A + (B + C) = A + B + C$		$A + B + C = B + C + A = \dots$		$A + (B \cdot C) = (A + B) \cdot (A + C)$	

A	B	C	A·B	B·C	A·C	A·(B+C)	(A·B)+(A·C)	A+(B·C)	(A+B)·(A+C)
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	1	0	1	0	0	0	1	1
1	0	0	0	0	0	0	0	1	1
1	0	1	0	0	1	1	1	1	1
1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1	1	1

2019-10-12 dominiak.kuczyński@pwr.edu.pl 22

22

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Zadanie

$d = \overline{(a \wedge \overline{b} \vee c)}$

2019-10-12 dominiak.kuczyński@pwr.edu.pl 23

23



Implementacja sprzętowa

2019-10-12 dominiak.kuczyński@pwr.edu.pl 24

24

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Bramki logiczne

Logic function	Logic symbol	Truth table	Boolean expression															
Buffer		<table border="1"> <tr><th>A</th><th>Y</th></tr> <tr><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td></tr> </table>	A	Y	0	0	1	1	$Y = A$									
A	Y																	
0	0																	
1	1																	
Inverter (NOT gate)		<table border="1"> <tr><th>A</th><th>Y</th></tr> <tr><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td></tr> </table>	A	Y	0	1	1	0	$Y = \bar{A}$									
A	Y																	
0	1																	
1	0																	
2-input AND gate		<table border="1"> <tr><th>A</th><th>B</th><th>Y</th></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	A	B	Y	0	0	0	0	1	0	1	0	0	1	1	1	$Y = A \cdot B$
A	B	Y																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
2-input NAND gate		<table border="1"> <tr><th>A</th><th>B</th><th>Y</th></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	A	B	Y	0	0	1	0	1	1	1	0	1	1	1	0	$Y = \overline{A \cdot B}$
A	B	Y																
0	0	1																
0	1	1																
1	0	1																
1	1	0																

2019-10-12 dominiak.lucjusz@pwr.lodz.pl 25

25

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Bramki logiczne

2-input OR gate		<table border="1"> <tr><th>A</th><th>B</th><th>Y</th></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	A	B	Y	0	0	0	0	1	1	1	0	1	1	1	1	$Y = A + B$
A	B	Y																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
2-input NOR gate		<table border="1"> <tr><th>A</th><th>B</th><th>Y</th></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	A	B	Y	0	0	1	0	1	0	1	0	0	1	1	0	$Y = \overline{A + B}$
A	B	Y																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
2-input EX-OR gate		<table border="1"> <tr><th>A</th><th>B</th><th>Y</th></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	A	B	Y	0	0	0	0	1	1	1	0	1	1	1	0	$Y = A \oplus B$
A	B	Y																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
2-input EX-NOR gate		<table border="1"> <tr><th>A</th><th>B</th><th>Y</th></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	A	B	Y	0	0	1	0	1	0	1	0	0	1	1	1	$Y = \overline{A \oplus B}$
A	B	Y																
0	0	1																
0	1	0																
1	0	0																
1	1	1																

2019-10-12 dominiak.lucjusz@pwr.lodz.pl 26

26

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Symbole

	AND	OR	NOT	XOR	NAND	NOR	XNOR
IEC 60617-12							
US ANSI 91-1984							
DIN 40700							

2019-10-12 dominiak.lucjusz@pwr.lodz.pl 27

27

Poznan University of Technology
Institute of Control, Robotics and Information Engineering

2019-10-12 domini.k.luczak@pwr.tl 28

28

Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Implementacja

$d = (a \wedge b \vee c)$

2019-10-12 domini.k.luczak@pwr.tl 29

29

Poznan University of Technology
Institute of Control, Robotics and Information Engineering

<https://logic.ly/>

2019-10-12 domini.k.luczak@pwr.tl 30

30

Poznan University of Technology
Institute of Control, Robotics and Information Engineering

<https://circuitverse.org/simulator>

31

Poznan University of Technology
Institute of Control, Robotics and Information Engineering

<https://logic.ly/>

$d = (a \wedge b \vee c)$

32

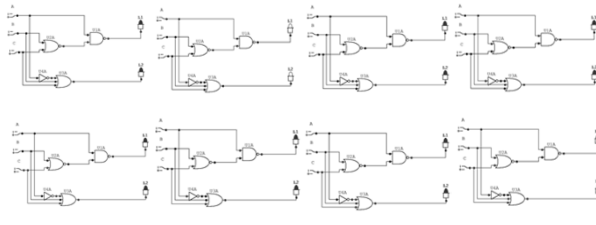
Poznan University of Technology
Institute of Control, Robotics and Information Engineering

$d = (a \wedge b \vee c)$

33

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

$d = (a \wedge b \vee c) = (\bar{a} \vee b \vee c)$



2019-10-12 dominiak.krzysztof@pwr.edu.pl 34

34

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Zadanie domowe

$d = (a \wedge b \vee a \wedge \bar{b} \vee c)$

2019-10-12 dominiak.krzysztof@pwr.edu.pl 35

35



Implementacja programowa

2019-10-12 dominiak.krzysztof@pwr.edu.pl 36

36

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

C/C++

Operator	Meaning	Example	Result
&&	Logical and	(5<2)&&(5>3)	False
	Logical or	(5<2) (5>3)	True
!	Logical not	!(5<2)	True

2019-10-12 domini.k.luczak@pwr.lodz.pl 37

37

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

C/C++

Operator	precedence
!	High
&&	Medium
	Low

A B && C	means	A (B && C)
A && B C && D	means	(A && B) (C && D)
A && B && C D	means	((A && B) && C) D
!A && B C	means	((!A) && B) C

2019-10-12 domini.k.luczak@pwr.lodz.pl 38

38

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Język C – operatory logiczne

Symbol	Nazwa	Przykład	Zapis matematyczny
!	Logiczna negacja	!a	$\neg a$ lub $\bar{a}$
	Logiczna alternatywa (OR)	a b	a b lub a $\vee$ b
&&	Logiczna koniunkcja (AND)	a && b	a & b lub a $\wedge$ b

2019-10-12 domini.k.luczak@pwr.lodz.pl 39

39

 Poznań University of Technology Institute of Control, Robotics and Information Engineering	
Język C – operatory logiczne	
operator	rodzaj porównania
==	czy równe
>	większy
>=	większy bądź równy
<	mniej
<=	mniej bądź równy
!=	czy różny (nierówny)

40

 Poznań University of Technology Institute of Control, Robotics and Information Engineering					
Język C – operatory logiczne					
Operator	Przykład	Wynik	Przykład II	Wynik II	
==	'a'==1	falsz	'a'==97	prawda	
>	2>3	falsz	2>1	prawda	
>=	2>=3	falsz	2>=2	prawda	
<	2<1	falsz	1<'a'	prawda	
<=	3.5<=2.5	falsz	2.5<=3.5	prawda	
!=	'a'!='a'	falsz	'a'!='b'	prawda	

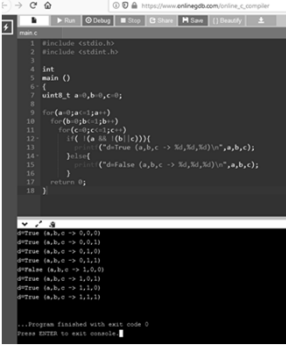
41

 Poznań University of Technology Institute of Control, Robotics and Information Engineering									
Opierający logikę w językach komputerowych									
język komputerowy	językoargumentowe			dowargumentowe					
	regalia	komunikacja	alternatywa	alternatywa	rozdzielanie	implikacja	rozmowność	zmienna	
AGM ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
AGM ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
CPM ¹⁷¹ , C ₂ ¹⁷¹	1	AA	11	1A	X	X	X	X	
Copm ¹⁷¹	AND	AND	OR	X	X	X	X	X	
CCOL ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Fur ¹⁷¹	X	AND	OR	X	X	X	X	X	
Fur ¹⁷¹ ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Fur ¹⁷¹ ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
GA ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Jam ¹⁷¹	1	AA	11	1A	X	X	X	X	
Mod ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Par ¹⁷¹	1	AA	11	1A	X	X	X	X	
Par ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Par ¹⁷¹	1	AA	11	1A	X	X	X	X	
Par ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Par ¹⁷¹	1	AA	11	1A	X	X	X	X	
Par ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Par ¹⁷¹	1	AA	11	1A	X	X	X	X	
Par ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Par ¹⁷¹	1	AA	11	1A	X	X	X	X	
Par ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Par ¹⁷¹	1	AA	11	1A	X	X	X	X	
Par ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Par ¹⁷¹	1	AA	11	1A	X	X	X	X	
Par ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Par ¹⁷¹	1	AA	11	1A	X	X	X	X	
Par ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Par ¹⁷¹	1	AA	11	1A	X	X	X	X	
Par ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Par ¹⁷¹	1	AA	11	1A	X	X	X	X	
Par ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Par ¹⁷¹	1	AA	11	1A	X	X	X	X	
Par ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Par ¹⁷¹	1	AA	11	1A	X	X	X	X	
Par ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Par ¹⁷¹	1	AA	11	1A	X	X	X	X	
Par ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Par ¹⁷¹	1	AA	11	1A	X	X	X	X	
Par ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Par ¹⁷¹	1	AA	11	1A	X	X	X	X	
Par ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Par ¹⁷¹	1	AA	11	1A	X	X	X	X	
Par ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Par ¹⁷¹	1	AA	11	1A	X	X	X	X	
Par ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Par ¹⁷¹	1	AA	11	1A	X	X	X	X	
Par ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Par ¹⁷¹	1	AA	11	1A	X	X	X	X	
Par ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Par ¹⁷¹	1	AA	11	1A	X	X	X	X	
Par ¹⁷¹	NOT	AND	OR	X	X	X	X	X	
Par ¹⁷¹	1	AA	11	1A	X	X	X	X	

42

Poznan University of Technology
Institute of Control, Robotics and Information Engineering

C/C++ $d=(a \wedge b \vee \bar{c})$
//Wariant 1
if(!(a && !(b||c)))
{
}

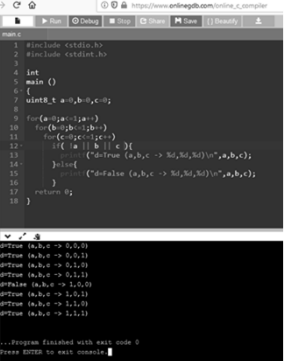


2019-10-12 43

43

Poznan University of Technology
Institute of Control, Robotics and Information Engineering

C/C++ $d=(a \wedge b \vee \bar{c})=$
 $=(\bar{a} \vee b \vee \bar{c})$
//Wariant 1
if(!(a && !(b||c)))
{
}
//Wariant 2 – po
uproszczeniu
if(!a || b || c){
}



2019-10-12 44

44

Poznan University of Technology
Institute of Control, Robotics and Information Engineering

C/C++ Zadanie domowe
 $d=(a \wedge b \vee a \wedge b \vee c)$

2019-10-12 45

45



Gray code -> Binary

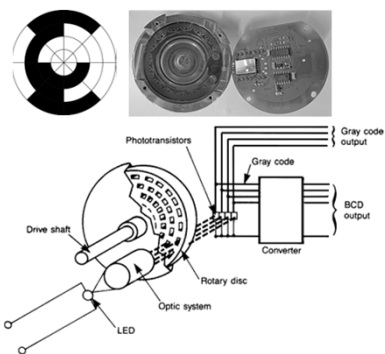
46



Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Gray

Decimal	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000



2019-10-12

dominik.kucak@pwr.edu.pl

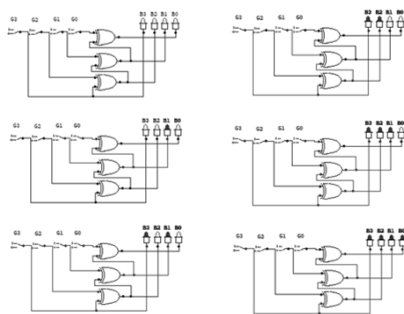
47



Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Gray

Decimal	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000



2019-10-12

dominik.kucak@pwr.edu.pl

48

48



Gray code -> Binary
C/C++

49



Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Język C – operatory bitowe

Symbol	Nazwa
~	Bitowa negacja
	Bitowa alternatywa (OR)
&	Bitowa koniunkcja (AND)
^	Bitowa różnica symetryczna (XOR)
>>	Bitowe przesunięcie w prawo
<<	Bitowe przesunięcie w lewo

2019-10-12

dominik.kuczakowski@pwr.edu.pl

50

50



Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Język C – operatory bitowe

Bitowa negacja

a	~a	
0	1	$25_{10} = 0x19_{16} = 0001\ 1001_2$
1	0	$\sim(25)_{10} = \sim(0001\ 1001)_2 = 1110\ 0110_2$

2019-10-12

dominik.kuczakowski@pwr.edu.pl

51

51

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Język C – operatory bitowe

Bitowa alternatywa

a	b	a b
0	0	0
0	1	1
1	0	1
1	1	1

$$25_{10}=0x19_{16}=0001\ 1001_2$$

$$6_{10}=0x06_{16}=0000\ 0110_2$$

$$(0001\ 1001)_2 | (0000\ 0110)_2 = 0001\ 1111_2$$

$$\begin{array}{r} 00011001_2 \\ | \\ 00000110_2 \\ \hline 00011111_2 \end{array}$$

2019-10-12 domini.k.luczak@pwr.edu.pl 52

52

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Język C – operatory bitowe

Bitowa koniunkcja

a	b	a&b
0	0	0
0	1	0
1	0	0
1	1	1

$$19_{10}=0x13_{16}=0001\ 0011_2$$

$$85_{10}=0x55_{16}=0101\ 0101_2$$

$$(0001\ 0011)_2 \& (0101\ 0101)_2 = 0001\ 0001_2$$

$$\begin{array}{r} 00010011_2 \\ \& 01010101_2 \\ \hline 00010001_2 \end{array}$$

2019-10-12 domini.k.luczak@pwr.edu.pl 53

53

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Język C – operatory bitowe

Bitowa różnica symetryczna/alternatywa wykluczająca

a	b	a^b
0	0	0
0	1	1
1	0	1
1	1	0

$$45_{10}=0x2D_{16}=0010\ 1101_2$$

$$25_{10}=0x19_{16}=0001\ 1001_2$$

$$(0010\ 1101)_2 \wedge (0001\ 1001)_2 = 0011\ 0100_2$$

$$\begin{array}{r} 00101101_2 \\ \wedge 00011001_2 \\ \hline 00110100_2 \end{array}$$

2019-10-12 domini.k.luczak@pwr.edu.pl 54

54

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Język C – operatory bitowe

Bitowe przesunięcie w prawo (>>)

a >> b

$a=48_{10}=0x30_{16}=0011\ 0000_2$

$(a>>2) = (0011\ 0000_2 >> 2) = 0000\ 1100_2 = 12_{10}$

$\frac{48}{2^2} = \frac{48}{4} = 12$

2019-10-12 dominik.borczak@pwr.edu.pl 55

55

 Poznan University of Technology
Institute of Control, Robotics and Information Engineering

Język C – operatory bitowe

Bitowe przesunięcie w lewo (<<)

a << b

$a=48_{10}=0x30_{16}=0011\ 0000_2$

$(a<<2) = (0011\ 0000_2 << 2) = 1100\ 0000_2 = 192_{10}$

$48 \cdot 2^2 = 48 \cdot 4 = 192$

2019-10-12 dominik.borczak@pwr.edu.pl 56

56

https://www.online-gdb.com/online_c_compiler

Decimal	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define BYTE_TO_BINARY_PATTERN "%c%c%c"
5  #define BYTE_TO_BINARY(byte) \
6  (byte & 0xFF ? "1" : "0"), \
7  (byte & 0xFF ? "1" : "0"), \
8  (byte & 0xFF ? "1" : "0"), \
9  (byte & 0xFF ? "1" : "0")
10
11
12
13
14
15
16
17
18
19
20 * This function converts a reflected binary gray 8 bit code number to a binary number.
21 * Each gray code bit is exclusive-ored with all more significant bits.
22 */
23 uint8_t GrayToBinary (uint8_t num)
24 {
25     uint8_t mask = num >> 1;
26     while (mask > 0)
27     {
28         num = num ^ mask;
29         mask = mask >> 1;
30     }
31     return num;
32 }
33
34 int
35 main ()
36 {
37     printf ("Hello World!\n");
38
39     uint8_t gray = 0xFF;
40     uint8_t binary = GrayToBinary (gray);
41     printf ("%d bits of result\n", sizeof (binary));
42     printf ("%c", BYTE_TO_BINARY_PATTERN " to binary " BYTE_TO_BINARY_PATTERN
43            " ", BYTE_TO_BINARY (gray), BYTE_TO_BINARY (binary));
44
45     return 0;
46 }

```

1111 1111 to binary 1010

...Program finished with exit code 0

Press ENTER to exit console

2019-10-12 dominik.borczak@pwr.edu.pl 57

57

https://www.online-gdb.com/online_c_compiler

Decimal	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

```

1 int
2 main()
3 {
4     printf("Hello World!\n");
5
6     for(int i = 0; i < 16; i++){
7         printf("%d\t", i);
8     }
9
10    uint8_t binary = gray2binary(gray);
11    //Print 8 bits of result
12    printf("Binary: %s\n", binary);
13    printf("Gray: %s\n", gray);
14    return 0;
15 }

```

...Program finished with exit code 0
Press ENTER to exit console.

58

<https://repl.it/languages/c>

```

1 #include <stdio.h>
2 #include <stdint.h>
3
4 //Define the gray to binary pattern
5 #define GRAY_TO_BINARY_PATTERN "000000010001001001001100001100011100101101001011100011110000"
6
7 //Define the binary to gray pattern
8 #define BINARY_TO_GRAY_PATTERN "000000010001001001001100001100011100101101001011100011110000"
9
10 //Function to convert gray to binary
11 uint8_t gray2binary(uint8_t gray)
12 {
13     uint8_t mask = 0x80;
14     while(mask > 0){
15         mask = mask >> 1;
16         gray = gray ^ mask;
17     }
18     return gray;
19 }
20
21 int
22 main()
23 {
24     printf("Hello World!\n");
25
26     for(int i = 0; i < 16; i++){
27         printf("%d\t", i);
28     }
29
30     uint8_t binary = gray2binary(gray);
31     //Print 8 bits of result
32     printf("Binary: %s\n", binary);
33     printf("Gray: %s\n", gray);
34     return 0;
35 }

```

...Program finished with exit code 0
Press ENTER to exit console.

59