

LUC - ultimate edition

Damian Gnieciak  
Szymon Ludwiniak  
Adam Szatkowski  
Jakub Kadziewicz  
Arek Pytka  
Kleks (?)  
Jan Napieralski

10 marca 2024

# Spis treści

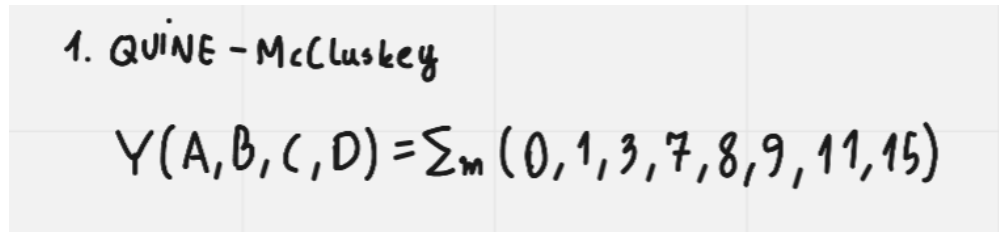
<b>1</b>	<b>Quine-McCluskey</b>	<b>4</b>
1.1	Treść zadania . . . . .	4
1.2	Rozwiązanie . . . . .	4
1.2.1	Małe kotki . . . . .	4
1.2.2	Pluszowe misie . . . . .	5
1.2.3	Składnik X . . . . .	5
1.2.4	Szalony naukowiec . . . . .	6
1.2.5	Magiczna tabelka . . . . .	6
1.3	Rozwiązanie siatką karnaugh . . . . .	7
1.4	Klik . . . . .	7
<b>2</b>	<b>Siatka karnaugh</b>	<b>8</b>
2.1	Siatki pomocnicze . . . . .	8
2.2	Treść zadania . . . . .	9
2.3	Rozwiązanie . . . . .	9
2.3.1	Siatka z wartościami liczbowymi . . . . .	9
2.3.2	Siatka bez połączonych jedynek . . . . .	9
2.3.3	Siatka z połączonymi jedynekami . . . . .	10
2.3.4	Wynik . . . . .	10
2.4	Klik . . . . .	10
<b>3</b>	<b>Kody</b>	<b>11</b>
3.0.1	Kod 8-4-2-1 (BCD) . . . . .	11
3.0.2	Kod +N . . . . .	11
3.0.3	Kod 2-4-2-1 . . . . .	11
3.0.4	Kod gray'a . . . . .	12
3.1	Detekcyjne i korekcyjne . . . . .	12
3.1.1	Kod 1 z 10 . . . . .	12
3.1.2	Kod 2 z 5 . . . . .	13
3.1.3	Kod z kontrolą parzystości . . . . .	14
3.1.4	Kod Hamming'a . . . . .	15
<b>4</b>	<b>Algebra boole'a</b>	<b>16</b>
4.1	Aksjomaty . . . . .	16
4.1.1	Prawo przemienności . . . . .	16
4.1.2	Prawo łączności . . . . .	16
4.1.3	Prawo rozdzielczości . . . . .	16
4.1.4	Prawo tożsamości . . . . .	16
4.1.5	Prawo komplementarności . . . . .	16
4.1.6	Prawo de Morgana . . . . .	16
4.1.7	Prawo sklejaniania . . . . .	16
4.1.8	Prawo pochłaniania . . . . .	16
4.2	Klik . . . . .	16
<b>5</b>	<b>Multipleksery</b>	<b>17</b>
5.1	Symbol multipleksa 4-bitowego . . . . .	17
5.2	Skrócona tabela prawdy . . . . .	17
5.3	Siatka karnaugh dla multipleksa 4-bitowego . . . . .	18
5.4	Równie wynikowe . . . . .	18
5.5	Schemat układu . . . . .	19

<b>6</b>	<b>Demultipleksery</b>	<b>20</b>
6.1	Symbol demultipleksera 4-bitowego . . . . .	20
6.2	Tabela prawdy . . . . .	20
6.3	Siatka karnaugh dla demultipleksera 4-bitowego . . . . .	21
6.4	Schemat układu . . . . .	21
<b>7</b>	<b>Liczniki asynchroniczne</b>	<b>22</b>
7.1	Zadanie . . . . .	22
7.2	komentarz . . . . .	22
7.3	Tablica prawdy . . . . .	22
7.4	Siatka Karnaugh . . . . .	23
7.5	Funkcja resetu . . . . .	23
7.6	Schemat układu . . . . .	24
<b>8</b>	<b>Liczniki synchroniczne</b>	<b>25</b>
8.1	Zadanie . . . . .	25
8.2	Tabela przejść . . . . .	25
8.3	Siatki Karnaugh . . . . .	25
8.4	Schemat układu . . . . .	26
8.5	Klik . . . . .	26
<b>9</b>	<b>Wyrażenie regularne</b>	<b>27</b>
9.1	Treść zadania . . . . .	27
9.2	Czarna magia i techniki zakazane . . . . .	27
9.2.1	Prolog . . . . .	27
9.2.2	Niebagatelny zwrot akcji . . . . .	28
9.2.3	Epilog . . . . .	29
9.3	Tabela stanów . . . . .	29
9.4	Zminimalizowana tabela stanów . . . . .	30
9.5	Finalny graf . . . . .	30
9.6	Wyrażenie regularne grafu $G^{++}$ . . . . .	30
<b>10</b>	<b>Grafy automatów</b>	<b>31</b>
10.1	Słówko o automatach . . . . .	31
10.2	Treść zadania . . . . .	31
10.3	Graf automatu moore'a . . . . .	31
10.4	Graf automatu mealy'ego . . . . .	32
<b>11</b>	<b>Informacje dodatkowe</b>	<b>33</b>
11.1	napięcia . . . . .	33
11.2	zbocza . . . . .	33
11.3	tabele dla przerzutników . . . . .	33
11.4	bramki na labach . . . . .	34
11.5	układy kombinacyjne a sekwencyjne . . . . .	35
11.6	minimum bramek, jakie potrzeba . . . . .	35
11.7	wielka piątka układów cyfrowych . . . . .	35
11.8	te pościgi, te wybuchy . . . . .	35

This page intentionally left blank

# 1 Quine-McCluskey

## 1.1 Treść zadania



1. QUINE - McCluskey

$$Y(A,B,C,D) = \sum_m (0,1,3,7,8,9,11,15)$$

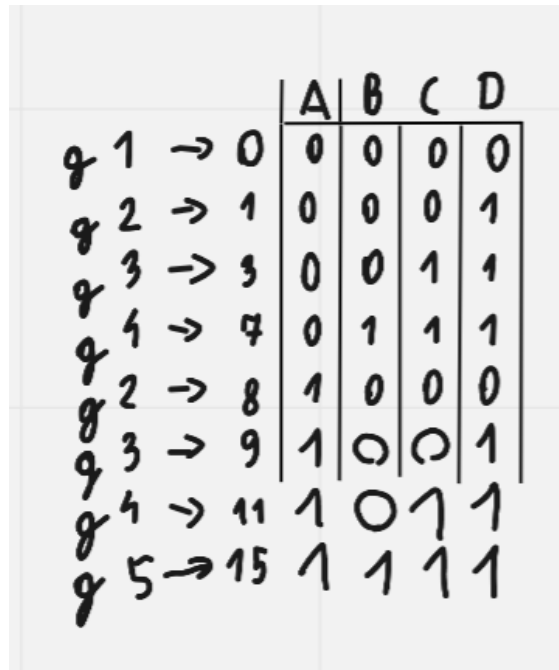
Rysunek 1: treść zadania

Funkcja Y dla zmiennych ABCD przyjmuje wartości 1 dla słów 0,1,3,7,8,9,11,15. A jest najstarszym bitem a D najmłodszym.

## 1.2 Rozwiązanie

### 1.2.1 Małe kotki

Każdemu słowu przypisujemy grupę która oznacza ilość wystąpień jedynek w słowie: **grupa pierwsza** (g 1) to słowo mające zero jedynek, a **grupa czwarta** (g 4) trzy jedynki.



		A	B	C	D
g 1	→ 0	0	0	0	0
g 2	→ 1	0	0	0	1
g 3	→ 3	0	0	1	1
g 4	→ 7	0	1	1	1
g 2	→ 8	1	0	0	0
g 3	→ 9	1	0	0	1
g 4	→ 11	1	0	1	1
g 5	→ 15	1	1	1	1

Rysunek 2: małe kotki

### 1.2.2 Pluszowe misie

Przepisujemy tabelkę tak jak przedstawione to na rysunku - poszczególne grupy są oddzielone od siebie żeby ułatwić następne etapy.

G.	MINTERM	A	B	C	D
1.	$m_0$	0	0	0	0
2.	$m_1$	0	0	0	1
	$m_8$	1	0	0	0
3.	$m_9$	0	0	1	1
	$m_7$	1	0	0	1
4.	$m_2$	0	1	1	1
	$m_{11}$	1	0	1	1
5.	$m_{15}$	1	1	1	1

Rysunek 3: pluszowe misie

### 1.2.3 Składnik X

Słowa które znajdują się w różnych grupach i różnią się od siebie jednym bitem łączymy w pary przepisując je do kolejnej tabeli, wyrażenia przepisujemy bez zmian a na bicie który się różnił wpisujemy symbol podłogi

G.	PAIRS	A	B	C	D
1.	$m_0 - m_1$	0	0	0	—
	$m_8 - m_9$	—	0	0	0
2.	$m_1 - m_9$	0	0	—	1
	$m_8 - m_7$	—	0	0	1
3.	$m_9 - m_7$	1	0	0	—
	$m_3 - m_{11}$	0	—	1	1
	$m_2 - m_{11}$	—	0	1	1
	$m_9 - m_{11}$	1	0	—	1
4.	$m_7 - m_{15}$	—	1	1	1
	$m_{11} - m_{15}$	1	—	1	1

Rysunek 4: składnik X

### 1.2.4 Szalony naukowiec

Nasze pary łączymy następnie w czwórki, stosując dokładnie tą samą metodę, mechanizm ten powtarzamy, póki możliwe jest uszczuplenie tabelki.

po maksymalnej optymalizacji naszej tabeli, łącząc klamrami wyrażenia z tych samych grup wpisujemy wyrażenia boolowskie analogicznie do siatek Karnaugh'a.

G.	PAIRS	A	B	C	D	
1	0-1-8-9 0-8-1-9	0	0	-	-	$\bar{B} \cdot \bar{C}$
2.	1-9-3-11 1-3-9-11	0	-	1	-	$\bar{B} \cdot D$
3.	3-7-11-15 3-11-7-15	-	-	1	1	$C \cdot D$

Rysunek 5: szalony naukowiec

### 1.2.5 Magiczna tabelka

Tworzymy tabelkę tak jak poniżej. W kolumnach szukamy pojedynczych X. Wiersze w których nie występują zaznaczone krzyżyki odrzucamy i zapisujemy wyrażenie (analogicznie do siatek karnaugh'a)

(0, 1, 3, 7, 8, 9, 11, 15)							
P.I	minterms involved	0	1	3	7	8	9 11 15
$\bar{B} \cdot \bar{C}$	0, 1, 8, 9	x	x			x	x
$\bar{B} \cdot D$	1, 3, 9, 11		x	x			x x
$C \cdot D$	3, 7, 11, 15			x	x		x x
$Y = \bar{B} \cdot \bar{C} + C \cdot D$							

Rysunek 6: magiczna tabelka

### 1.3 Rozwiązanie siatką karnaugh

$Y(A,B,C,D) = \sum m(0,1,3,7,8,9,11,15)$

$\begin{matrix} C/D \\ A/B \end{matrix}$	00	01	11	10
00	1	1	1	
01			1	
11			1	
10	1	1	1	

$Y = \overline{B}\overline{C} + CD$

Rysunek 7: rozwiązanie siatką karnaugh

### 1.4 Klik

<http://quinemccluskey.com/>



## 2.1 Siatki pomocnicze

a \ b	0	1
0	0	1
1	2	3

a \ b	00	01	11	10
0	0	1	3	2
1	4	5	7	6

a \ b	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

a \ b	000	001	011	010	110	111	101	100
00	0	1	3	2	6	7	5	4
01	8	9	11	10	14	15	13	12
11	24	25	27	26	30	31	29	28
10	16	17	19	18	22	23	21	20

a \ b	000	001	011	010	110	111	101	100
000	0	1	3	2	6	7	5	4
001	8	9	11	10	14	15	13	12
011	24	25	27	26	30	31	29	28
010	16	17	19	18	22	23	21	20
110	48	49	51	50	54	55	53	52
111	56	57	59	58	62	63	61	60
101	40	41	43	42	46	47	45	44
100	32	33	35	34	38	39	37	36

Rysunek 8: duże siatki

## 2.2 Treść zadania

Zminimalizować met. siatek Karnaugh podaną funkcję boolowską:  
 $f(a,b,c,d,e,f) = \sum(0, 1, 4, 13, 17, 27, 29, 30, 31, 32, 37, 41, 50, 54, 56) +$   
 $\sum_d(3, 5, 15, 16, 19, 20, 22, 25, 26, 33, 36, 38, 40, 46, 60)$

Rysunek 9: treść zadania

$f(a, b, c, d, e, f)$  - oznacza że układ który tworzymy ma 6 wejść. a - najstarszy bit(msb), f - najmłodszy bit(lsb)

$\sum_m$  - dla tych wartości funkcja przyjmuje wartość „1”

$\sum_\phi$  - dla tych wartości funkcja przyjmuje wartość „-”

$\Pi$  - dla tych wartości funkcja przyjmuje wartość „0”

## 2.3 Rozwiązanie

### 2.3.1 Siatka z wartościami liczbowymi

def abc	000	001	011	010	110	111	101	100
000	0	1	3	2	6	7	5	4
001	8	9	11	10	14	15	13	12
011	24	25	27	26	30	31	29	28
010	16	17	19	18	22	23	21	20
110	48	49	51	50	54	55	53	52
111	56	57	59	58	62	63	61	60
101	40	41	43	42	46	47	45	44
100	32	33	35	34	38	39	37	36

Rysunek 10: siatka pomocnicza

### 2.3.2 Siatka bez połączonych jedynek

def abc	000	001	011	010	110	111	101	100
000	1	1	-				-	1
001						-	1	
011		-	1	-	1	1	1	
010	-	1	-		-			-
110				1	1			
111	1							-
101	-	1			-			
100	1	-			-		1	-

Rysunek 11: siatka bez zaznaczonych obszarów

### 2.3.3 Siatka z połączonymi jedynkami

Obszary, które możemy ująć wspólnie **muszą być symetryczne względem osi symetrii poziomej i pionowej** - znaczy to tyle, że jeśli "złoży się" siatkę wzdłuż osi symetrii, obszar powinien się pokrywać z jego drugą częścią po przeciwległej stronie (chyba, że cały obszar znajduje się w jednej ćwiartce). Warto nadmienić, że w siatkach o większych wymiarach (3x2 i więcej) każda ćwiartka też ma swoje osie symetrii działające analogicznie do całej siatki. W praktyce dotyczy się to tylko siatek o min. 3 zmiennych. W obszarach **można zawierać jedynki (lub analogicznie zera) i myślniki (stany niedozwolone)**. **Każdy obszar musi mieć  $2^n$  elementów i mieć kształt prostokąta**. Gdy jedynki są na krawędziach siatki, można je połączyć ze sobą (podobnie z narożnikami) przy zachowaniu symetrii, tj. po jednej i drugiej stronie siatki musi znajdować się tyle samo elementów.

def	000	001	011	010	110	111	101	100
000	1	1	-				-	1
001							1	
011		1	1	-	1	1	1	
010	-	1	1					-
110				1	1			
111	1							1
101	1	1			-			
100	1	-			-		1	-

Rysunek 12: siatka z zaznaczonymi obszarami

### 2.3.4 Wynik

$$Y = (\bar{b} \cdot \bar{c} \cdot \bar{e}) + (\bar{a} \cdot b \cdot c \cdot e) + (a \cdot c \cdot d \cdot f) + (\bar{a} \cdot b \cdot \bar{d} \cdot f) + (a \cdot \bar{b} \cdot \bar{d} \cdot \bar{e}) + (a \cdot \bar{b} \cdot c \cdot \bar{e} \cdot \bar{f}) + (a \cdot \bar{b} \cdot \bar{c} \cdot e \cdot \bar{f})$$

Rysunek 13: ostateczna funkcja

## 2.4 Klik

<https://www.charlie-coleman.com/experiments/kmap/>

## 3 Kody

- **dziesiętne-binarne**: pozwalają zapisać liczbę dziesiętną w systemie binarnym
- **refleksyjne**: wartości poszczególnych bitów zależą od innych wartości (jedno wynika z drugiego)
- **wagowe**: Wartość danego bitu zależy od jego pozycji. TODO: dokończ
- **detekcyjne**: Pozwalają wykryć błąd kodowania danych - **refleksyjne**: Pozwalają wykryć i skorygować błędy danych

### 3.0.1 Kod 8-4-2-1 (BCD)

Kod 8421, znany jako BCD - kod wagowy (istnieje bezpośredni związek pomiędzy wagą a pozycją cyfry). Wagi jak w kodzie binarnym, stąd łatwość wykonywania operacji arytmetycznych, tymi samymi metodami, co dla liczb dwójkowych. Np. liczba 127 w kodzie 8421 będzie wyglądała tak: 0001 0010 0111

Decimal Digit	BCD			
	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Rysunek 14: BCD

Źródła dodatkowe:

<https://www.youtube.com/watch?v=8sEUY-PBfaM>

<https://miniwebtool.com/bcd-to-decimal-converter/>

### 3.0.2 Kod +N

Kody z rodziny "+N" ( $D = B + N$ ) czyli np. "+3" to kody niewagowe, samouzupełniające się. Aby zapisać liczbę dziesiętną w tej postaci, trzeba przekonwertować ją na naturalny kod binarny i dodać N zapisane binarnie

### 3.0.3 Kod 2-4-2-1

Kod 2421 – kod wagowy, samouzupełniający, przydatny w układach zliczających.

Wszystkie mają małą odporność na zakłócenia – np. zmiany na pozycjach mogą nie występować jednocześnie – zmiana 0111 na 1111, zamiast na 1000 (w sterowaniu to problem).

Kod Excess 3 wygląda tak:

0 0011

1 0100

2 0101

3 0110

4 0111

5 1000

6 1001

7 1010

8 1011

9 1100

uzyskuje się go dodając do liczby wartość 3.

Rysunek 15: +3

Wartość dziesiętna	Kod Aikena
0	0000
1	0001
2	0010
3	0011
4	0100
5	1011
6	1100
7	1101
8	1110
9	1111

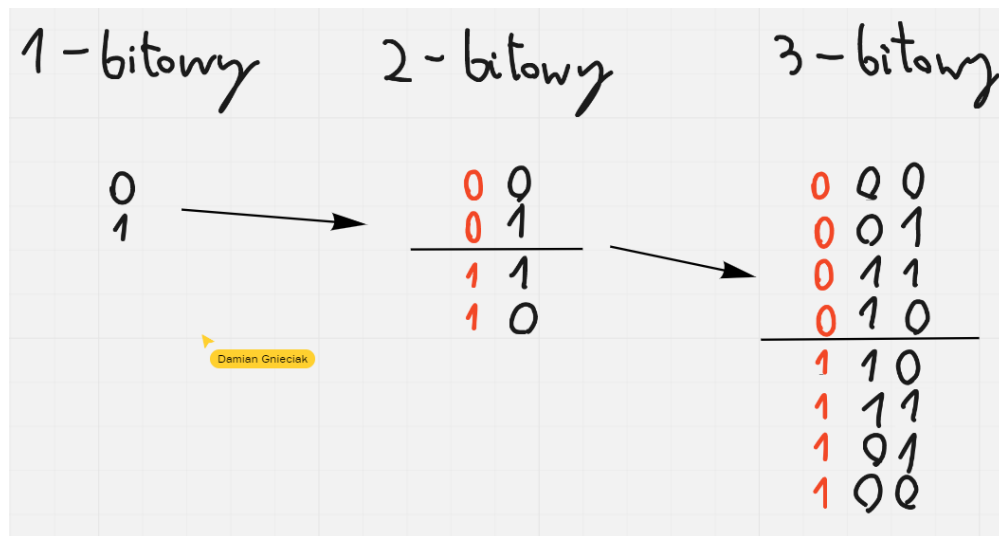
Rysunek 16: 2421

### 3.0.4 Kod gray'a

Kod refleksyjny. Możliwość powstawania błędów niejednoczesnej zmiany na pozycjach kodu jest wyeliminowana w kodach, w których nie więcej niż jeden bit zmienia swoją wartość przy przejściu między kolejnymi zakodowanymi wartościami.

## 3.1 Detekcyjne i korekcyjne

### 3.1.1 Kod 1 z 10



Rysunek 17: Gray

	z kontrolą parzystości	„1 z 10”	„2 z 5”
0	00000	0000000000 <b>1</b>	00011
1	00011	000000000 <b>10</b>	00101
2	00101	0000000 <b>100</b>	01001
3	00110	000000 <b>1000</b>	10001
4	01001	00000 <b>10000</b>	00110
5	01010	0000 <b>100000</b>	01010
6	01100	000 <b>1000000</b>	01010
7	01111	00 <b>10000000</b>	01100
8	10001	0 <b>100000000</b>	10100
9	10010	<b>1000000000</b>	11000

Rysunek 18: 1 z 10

### 3.1.2 Kod 2 z 5

	z kontrolą parzystości	„1 z 10”	„2 z 5”
0	00000	0000000000 <b>1</b>	00011
1	00011	000000000 <b>10</b>	00101
2	00101	0000000 <b>100</b>	01001
3	00110	000000 <b>1000</b>	10001
4	01001	00000 <b>10000</b>	00110
5	01010	0000 <b>100000</b>	01010
6	01100	000 <b>1000000</b>	01010
7	01111	00 <b>10000000</b>	01100
8	10001	0 <b>100000000</b>	10100
9	10010	<b>1000000000</b>	11000

Rysunek 19: 2 z 5

### 3.1.3 Kod z kontrolą parzystości

	z kontrolą parzystości	„1 z 10”	„2 z 5”
0	00000	000000000 <b>1</b>	00011
1	00011	00000000 <b>10</b>	00101
2	00101	0000000 <b>100</b>	01001
3	00110	000000 <b>1000</b>	10001
4	01001	00000 <b>10000</b>	00110
5	01010	0000 <b>100000</b>	01010
6	01100	000 <b>1000000</b>	01010
7	01111	00 <b>10000000</b>	01100
8	10001	0 <b>100000000</b>	10100
9	10010	<b>1000000000</b>	11000

Rysunek 20: z kontrolą parzystości

### 3.1.4 Kod Hamming'a

Hamminga	
0	0 0 0 0 0 0 0 0
1	0 0 0 0 1 1 1 1
2	0 0 1 1 0 0 1 1
3	0 0 1 1 1 1 1 0
4	0 1 0 1 0 1 0 1
5	0 1 0 1 1 0 1 1
6	0 1 1 0 0 1 1 1
7	0 1 1 0 1 0 0 0
8	1 0 0 1 0 1 1 1
9	1 0 0 1 1 0 0 0

Rysunek 21: Hamming



## 4 Algebra boole'a

### 4.1 Aksjomaty

#### 4.1.1 Prawo przemienności

$$A + B = B + A$$

$$A \cdot B = B \cdot A$$

#### 4.1.2 Prawo łączności

$$(A + B) + C = A + (B + C)$$

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

#### 4.1.3 Prawo rozdzielczości

$$A + (B \cdot C) = (A + B) \cdot (A + C)$$

$$A \cdot (B + C) = A \cdot B + A \cdot C$$

#### 4.1.4 Prawo tożsamości

$$A + 0 = A$$

$$A \cdot 0 = 0$$

$$A + 1 = 1$$

$$A \cdot 1 = A$$

$$A + A = A$$

$$A \cdot A = A$$

#### 4.1.5 Prawo komplementarności

$$A + \overline{A} = 1$$

$$A \cdot \overline{A} = 0$$

#### 4.1.6 Prawo de Morgana

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

#### 4.1.7 Prawo sklejania

$$A \cdot \overline{B} + A \cdot B = A$$

$$(A + \overline{B}) \cdot (A + B) = A$$

#### 4.1.8 Prawo pochłaniania

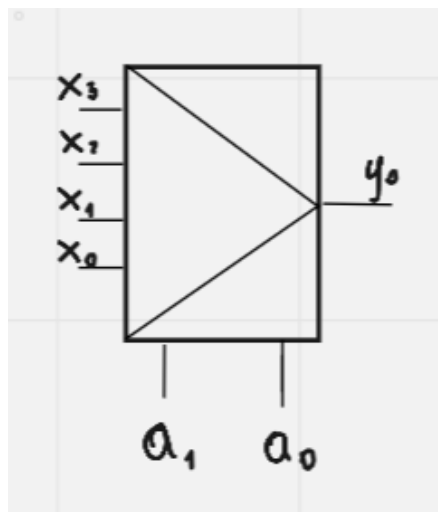
$$A \cdot \overline{B} + B = A + B$$

### 4.2 Klik

[http://www.zsk.ict.pwr.wroc.pl/zsk/repository/dydaktyka/ptcm/wyk/tc1\\_9\\_wyk\\_3.pdf](http://www.zsk.ict.pwr.wroc.pl/zsk/repository/dydaktyka/ptcm/wyk/tc1_9_wyk_3.pdf)

## 5 Multipleksery

### 5.1 Symbol multipleksera 4-bitowego



Rysunek 22: Symbol multipleksera

Żeby odróżnić multiplekser od demultipleksera należy pamiętać, że MULTI-plekser ma MULTUM wejść. *xd* Podstawa trójkąta na ikonie jest ustawiona zawsze w stronę wejść. Zadaniem multipleksera jest wybór za pomocą  $n$  wejść adresowych jednego z  $2^n$  wejść danych. Multiplekser  $n$ -bitowy oznacza, że ma  $n$  wejść danych.

### 5.2 Skrócona tabela prawdy

$a_1$	$a_0$	$x_3$	$x_2$	$x_1$	$x_0$	$y_0$
0	0	-	-	-	0	0
0	1	-	-	0	-	0
1	0	-	0	-	-	0
1	1	0	-	-	-	0
0	0	-	-	-	1	1
0	1	-	-	1	-	1
1	0	-	1	-	-	1
1	1	1	-	-	-	1

Rysunek 23: tabela prawdy dla multipleksera 4-bitowego

Nie ma sensu zapisywać całej tabeli prawdy, dlatego rozpisuje się jej uproszczoną wersję tak jak poniżej. Zgodnie z działaniem multipleksera nie ma sensu rozpatrywać wszystkich kombinacji, w końcu sygnały na wszystkich wejściach danych, poza wybranym przez wejścia adresowe, są pomijane.

### 5.3 Siatka karnaugh dla multiplexera 4-bitowego

$x_3x_4x_5$ $a_2a_1x_2$	000	001	011	010	110	111	101	100
000	0	1	1	0	0	1	1	0
001	0	1	1	0	0	1	1	0
011	0	0	1	1	1	1	0	0
010	0	0	1	1	1	1	0	0
110	0	0	0	0	0	0	0	0
111	1	1	1	1	1	1	1	1
101	0	0	0	0	1	1	1	1
100	0	0	0	0	1	1	1	1

Rysunek 24: siatka karnaugh dla multiplexera 4-bitowego

Rysunek 25: siatka karnaugh

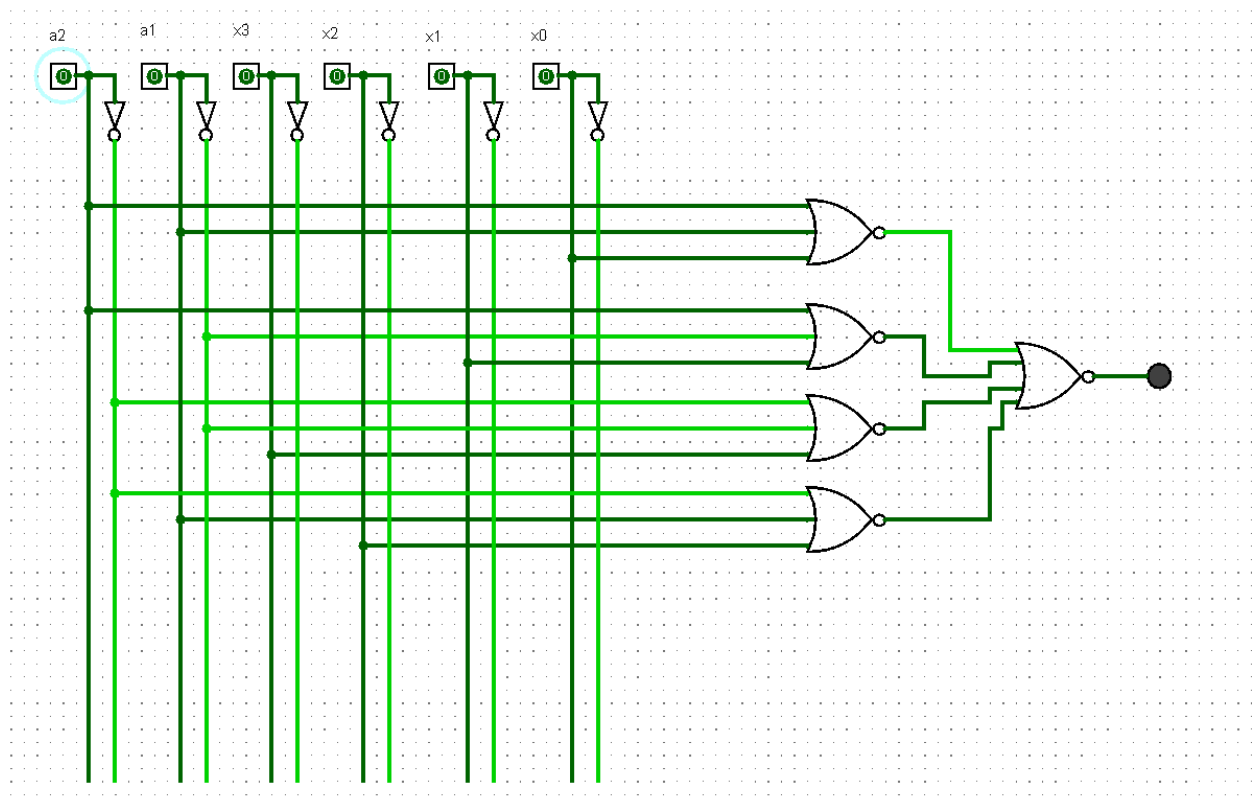
Multiplexer był wykonywany w pełni na bramkach NOR więc w ramach ułatwienia, w siatce zaznaczane były implicenty (zera).

### 5.4 Równie wynikowe

$$\begin{aligned}
 Y &= (a_2 + a_1 + x_0) \cdot (a_2 + \overline{a_1} + x_1) \cdot (\overline{a_2} + \overline{a_1} + x_3) \cdot (\overline{a_2} + a_1 + x_2) = \\
 &= \overline{(a_2 + a_1 + x_0)} + \overline{(a_2 + \overline{a_1} + x_1)} + \overline{(\overline{a_2} + \overline{a_1} + x_3)} + \overline{(\overline{a_2} + a_1 + x_2)}
 \end{aligned}$$

Rysunek 26: funkcja z siatki karnaugh

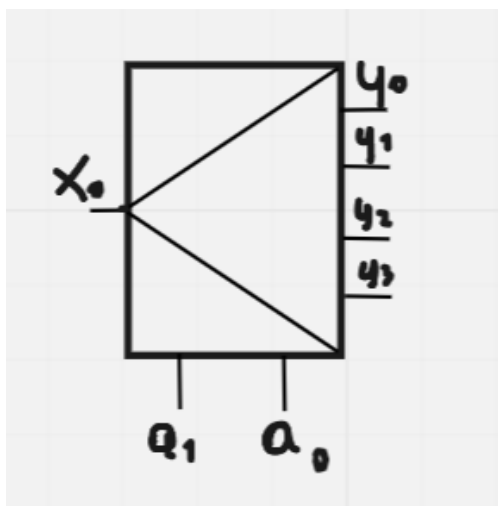
## 5.5 Schemat układu



Rysunek 27: schemat układu

## 6 Demultipleksery

### 6.1 Symbol demultipleksera 4-bitowego



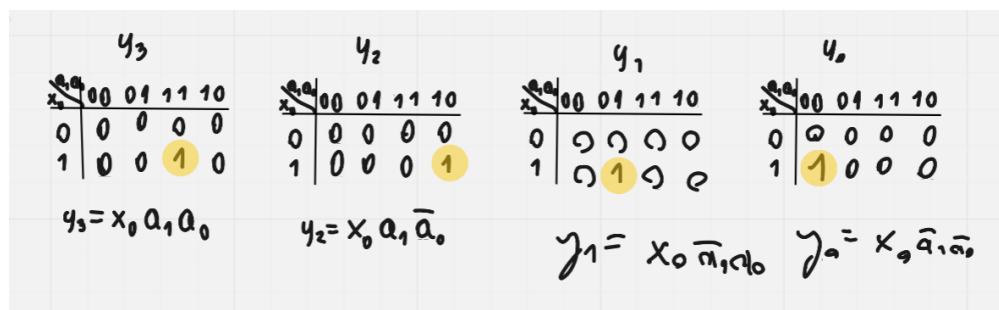
Rysunek 28: Symbol demultipleksera

### 6.2 Tabela prawdy

$x_0$	$a_1$	$a_0$	$y_3$	$y_2$	$y_1$	$y_0$
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

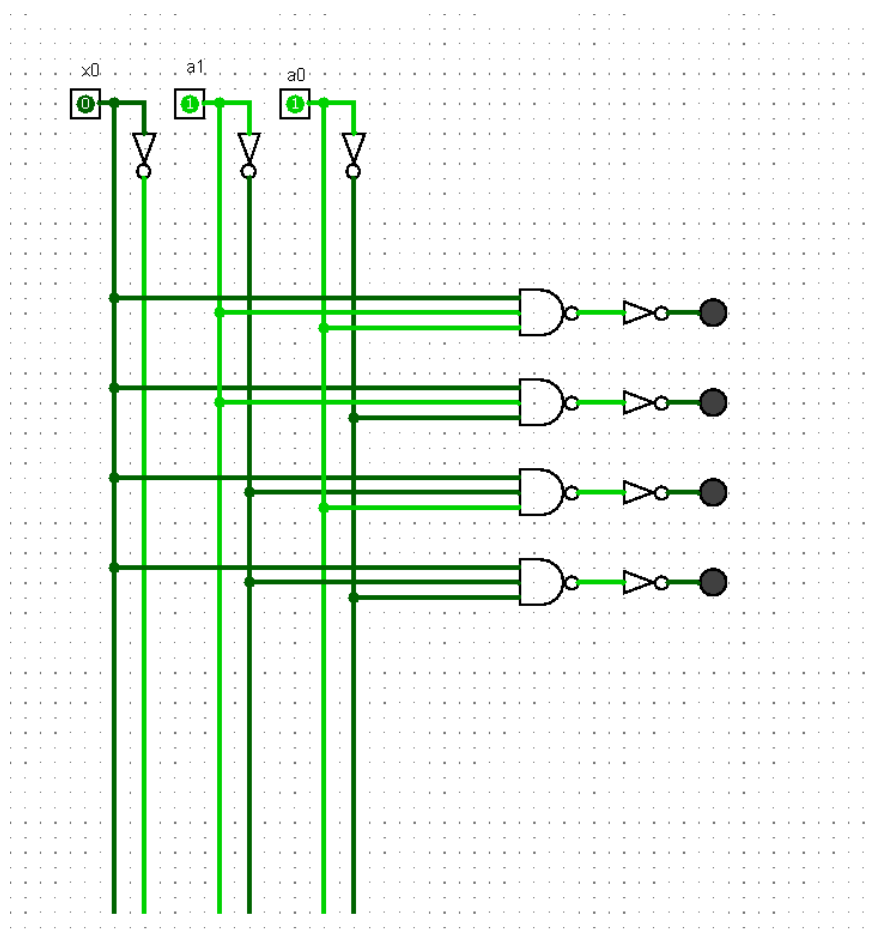
Rysunek 29: tabela prawdy dla demultipleksera 4-bitowego

### 6.3 Siatka karnaugh dla demultipleksera 4-bitowego



Rysunek 30: siatka karnaugh

### 6.4 Schemat układu



Rysunek 31: schemat układu

## 7 Liczniki asynchroniczne

### 7.1 Zadanie

Zrealizować licznik **asynchroniczny mod 4/13**

### 7.2 komentarz

Licznik asynchroniczny różni się od synchronicznego tym że tylko wejście zegarowe pierwszego przerzutnika jest podpięte do zegara. Wejście CLK każdego kolejnego flip flopa podpięte jest pod wyjście poprzedniego. Jeżeli licznik ma liczyć w przód to podpinamy zegar do zanegowanego wyjścia a jeśli w tył to do zwykłego Q.

### 7.3 Tablica prawdy

K	Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>	R
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	0
0	0	1	0	0	1
0	0	1	0	1	1
0	0	1	1	0	1
0	0	1	1	1	1
0	1	0	0	0	1
0	1	0	0	1	1
0	1	0	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
0	1	1	0	1	1
0	1	1	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	0	1	0
1	0	0	1	0	0
1	0	0	1	1	0
1	0	1	0	0	0
1	0	1	0	1	0
1	0	1	1	0	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	0	1	0
1	1	0	1	0	0
1	1	0	1	1	0
1	1	1	0	0	0
1	1	1	0	1	1
1	1	1	1	0	1
1	1	1	1	1	1

Rysunek 32: Tablica prawdy

Syntezę licznika zaczynamy od rozpisania tablicy prawdy, część miejsc w tablicy jest zaznaczona jako stany nieistotne, bo nasz licznik nigdy do nich nie dotrze. Taki zapis może ułatwić nam robotę podczas liczenia siatek. Skoro licznik liczy maksymalnie do 13 no to bez sensu jest zapisywanie zer dla 14, 15 i 16 skoro i tak układ się zresetuje na trzynastce. Dodatkowo warto zauważyć że gdy licznik jest ustawiony na mod 4 to wszystkie wartości powyżej 4 też oznaczają reset, to podejście pozwala uniknąć sytuację gdy jakiś gagatek postanowi przełączyć licznik z mod 13 na mod 4 w momencie gdy stan licznika pokazuje np. 8.

## 7.4 Siatka Karnaugh

$\begin{matrix} Q_3 Q_2 \\ k Q_1 Q_0 \end{matrix}$	00	01	11	10
000	0	0	0	0
001	1	1	1	1
011	1	1	—	—
010	1	1	1	1
110				
111		1	—	—
101				
100				

Rysunek 33: Siatka Karnaugh

Z tabeli robimy prostą siateczkę.

## 7.5 Funkcja resetu

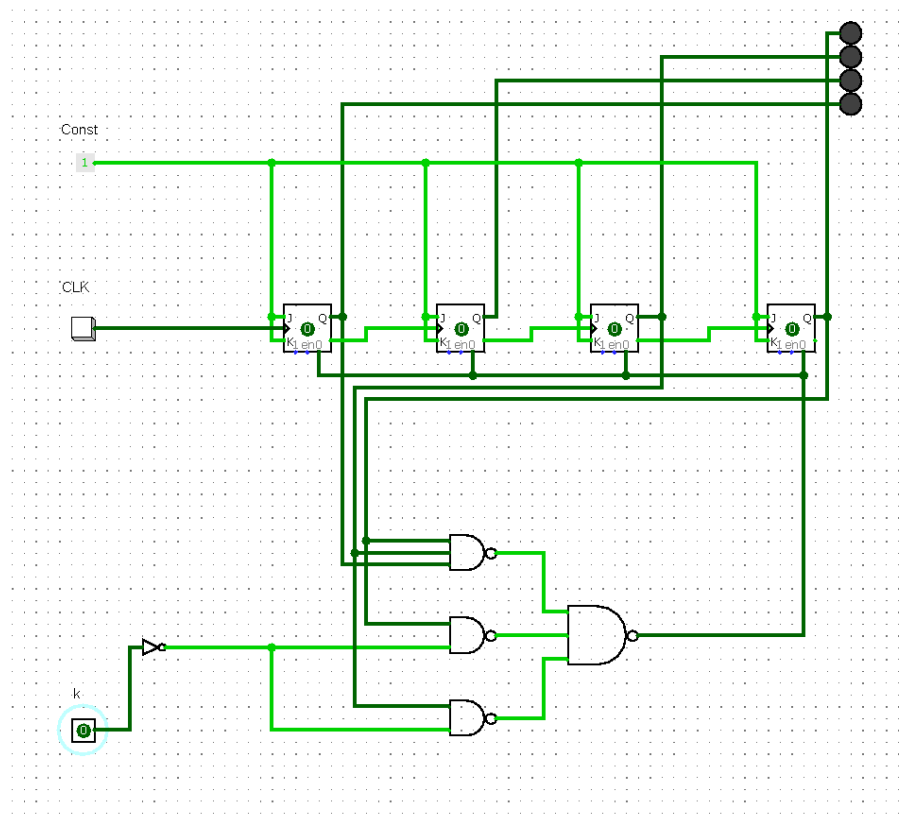
$$\begin{aligned}
 R &= \bar{K}Q_2 + \bar{K}Q_3 + Q_3Q_2Q_0 \\
 &= \overline{\bar{K}Q_2} \cdot \overline{\bar{K}Q_3} \cdot \overline{Q_3Q_2Q_0}
 \end{aligned}$$

Rysunek 34: Funkcja resetu

Funkcja, która wyszła z siatki została przekształcona by można było ją zbudować tylko na NANDach i negacjach.



## 7.6 Schemat układu



Rysunek 35: Schemat układu

## 8 Liczniki synchroniczne

### 8.1 Zadanie

Zaprojektować licznik synchroniczny realizujący liczenie w postaci  $0 \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow 7 \rightarrow 6 \rightarrow 9 \rightarrow 0$ . Układ zostanie zrealizowany na przerzutnikach JK - Jębać Kleksa ))))

W tym zadaniu najlepiej sprawdza się instrukcja doktora Antoniego Sterni, polecam przeczytać bo jest gitowa.

### 8.2 Tabela przejść

$0 \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow 7 \rightarrow 6 \rightarrow 9 \rightarrow 0$									
		$t$				$t+1$			
$t$	$t+1$	$Q_3$	$Q_2$	$Q_1$	$Q_0$	$J_3$	$K_3$	$J_2$	$K_2$
0	2	0	0	0	0	0	-	0	-
1	3	0	0	0	1	0	-	0	-
2	1	0	0	1	0	0	-	0	-
3	7	0	0	1	1	0	-	1	-
6	9	0	1	1	0	1	-	1	-
7	6	0	1	1	1	0	-	0	-
9	0	1	0	0	1	-	1	0	-

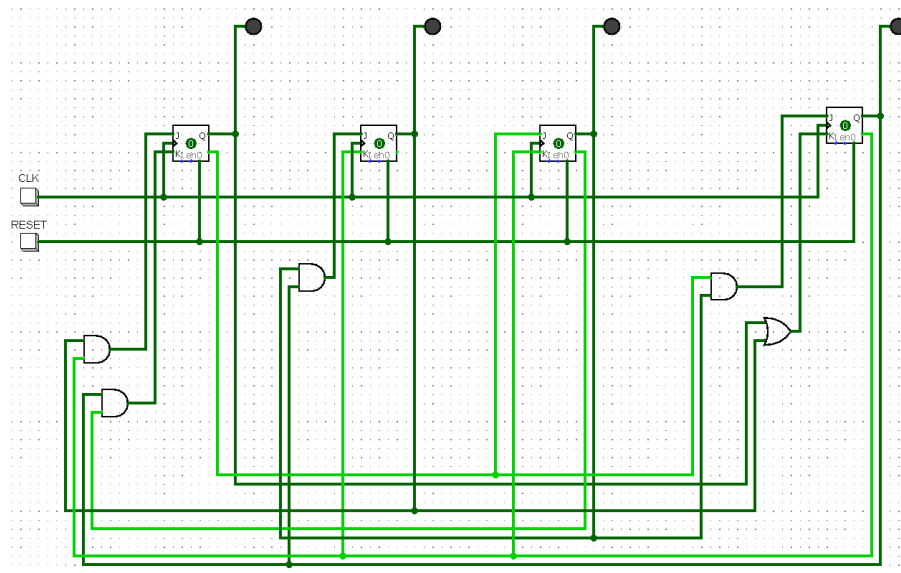
Rysunek 36: Tabela przejść

### 8.3 Siatki Karnaugh

$J_3$	$K_3$	$J_2$	$K_2$
 $J_3 = Q_2 \bar{Q}_0$	 $K_3 = Q_0 \bar{Q}_1$	 $J_2 = Q_1 Q_0$	 $K_2 = \bar{Q}_0$
$J_1$	$K_1$	$J_0$	$K_0$
 $J_1 = \bar{Q}_3$	 $K_1 = \bar{Q}_0$	 $J_0 = \bar{Q}_3 Q_1$	 $K_0 = Q_2 + Q_3$

Rysunek 37: Siatki Karnaugh

## 8.4 Schemat układu



Rysunek 38: Schemat układu

## 8.5 Klik

[http://staff.iiar.pwr.wroc.pl/antoni.sterna/luc/LUC\\_synteza\\_licznikow.pdf](http://staff.iiar.pwr.wroc.pl/antoni.sterna/luc/LUC_synteza_licznikow.pdf)

## 9 Wyrażenie regularne

### 9.1 Treść zadania

$$\begin{array}{l|l} s_1 = (z_1 + z_2 z_1 + z_2 z_2 z_1) * z_2 z_2 & y_1 \\ s_2 = (z_1 + z_2 z_1 + z_2 z_2 z_1) * z_2 z_2 (z_2) * \bar{z}_1 & y_2 \\ s_3 = \overline{(s_1 + s_2)} & y_0 = \varepsilon \end{array}$$

Rysunek 39: treść zadania

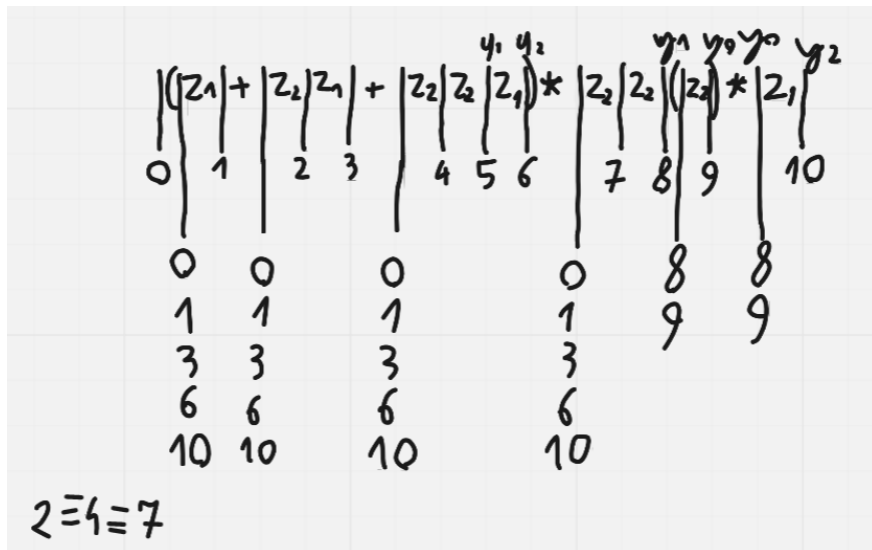
Wyrażenia regularne to jeden wielki pierdolnik. Nasze wyrażenie podane w zadaniu podaje na wyjściu trzy możliwe wartości,  $y_0$ ,  $y_1$ ,  $y_2$ .  $y_0$  pojawia się wtedy gdy nie jest aktywne ani  $y_1$  ani  $y_2$  xD. Opis całego wyrażenia sprowadza się do pracy nad  $s_2$ , bo w końcu  $s_2$  zawiera w sobie  $s_1$ . Celem zadania jest skonstruowanie takiego układu czy narysowanie takiego grafu, który będzie się zachowywał dla odpowiednich wejść ( $z$ ) tak jak jest to zapisane w wyrażeniu.

operatory używane w wyrażeniach regularnych:

- Operator  $+$  oznacza sumę logiczną, czyli że w wyrażeniu możemy wybrać którą ścieżką pójdziemy np:  $z_1 + z_2$ . oznacza że możemy wybrać albo  $z_1$  albo  $z_2$ .
- Operator  $*$  oznacza iterację czyli że dana część ujęta w nawias przed gwiazdką może być wykonywana w nieskończoność (zero lub więcej razy).

### 9.2 Czarna magia i techniki zakazane

#### 9.2.1 Prolog



Rysunek 40: stany podstawowe i przedpodstawowe

### 9.2.2 Niebagatelny zwrot akcji

Rysunek 41: stany podstawowe i przedpodstawowe

Znowu patrzymy na nasze zmienione wyrażenie i widzimy że czwórka jest tożsama z szóstką, w końcu przejście do jednego i drugiego odbywa się poprzez dwójkę i  $z_2$ . No i proces powtarzamy do skutku aż już nic więcej nam się nie skróci.

### 9.2.3 Epilog

	$((z_1 + z_2 z_1) + z_2 z_2 z_1) * z_2 z_2 ((z_1) * z_1) z_2$											
0-0	0	1	2	3	2	4	5	2	4	6	7	
1-1												
2-2												
3-3												
4-4	0	0		0			0	4		4		
5-5	1	1		1			1	6		6		
6-6	3	3		3			3					
7-6	5	5		5			5					
8-7	7	7		7			7					

Rysunek 42: stany podstawowe i przedpodstawowe

Finalnie wyrażenie wygląda tak jak na obrazku. Jest trochę lepiej. Najtrudniejsze już za nami chociaż kolejny krok wymaga szczególnego skupienia bo łatwo o zasadzenie jakiegoś baboła.

### 9.3 Tabela stanów

y	0	0	0	0	1	2	0	2	
q	0	1	2	3	4	5	6	7	*
z <sub>1</sub>	1	1	3	1	5	1	7	1	*
z <sub>2</sub>	2	2	4	2	6	2	6	2	*

0 ≡ 1 ≡ 3                      5 ≡ 7

Rysunek 43: tabela stanów

Jak widać na załączonym obrazku, stany  $0 \equiv 1 \equiv 3$  oraz  $5 \equiv 7$ . Podczas minimalizowania stanów należy pamiętać o zgodności sygnałów wyjściowych.

## 9.4 Zminimalizowana tabela stanów

Po zminimalizowaniu tabeli, należy pamiętać o zmienieniu kolejności numerowania stanów.

0-0	
1-0	
2-1	
3-0	
4-2	
5-3	
6-4	
7-3	

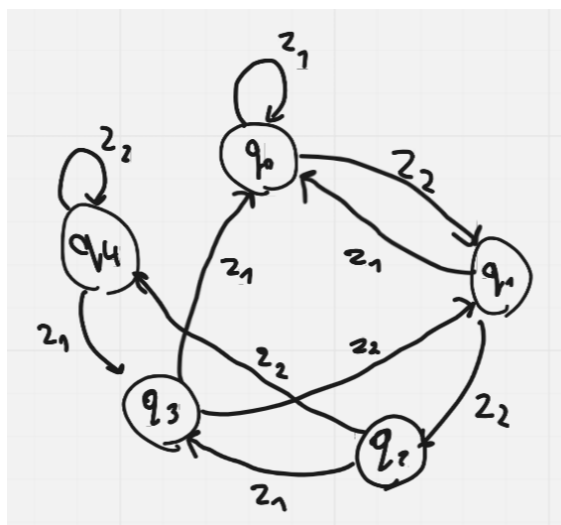
  

y	0	0	1	2	0
q	0	1	2	3	4
z <sub>1</sub>	0	0	3	0	3
z <sub>2</sub>	1	2	4	1	4

Rysunek 44: zminimalizowana tabela stanów

## 9.5 Finalny graf

Na podstawie zminimalizowanej tabeli można w łatwy sposób (a jest w ogóle tutaj coś trudnego?) sporządzić końcowy graf ;)



Rysunek 45: finalny graf

## 9.6 Wyrażenie regularne grafu $G^{++}$

Graf z poprzedniego podpunktu możemy również zapisać za pomocą oznaczeń. W tym celu rozpisujemy, w jakie miejsce przechodzi dany stan przy podaniu odpowiedniego wejścia:

$$G^{++} = {}^0(q_0^1(z_1q_0, z_2q_1^2(z_1q_0, z_2q_2^3(z_1q_3^4(z_1q_0, z_2q_1)^4, z_2q_4^4(z_1q_3, z_2q_4)^4)^3)^2)^1)^0$$

Ten sam zapis, gotowy do skopiowania do programu (np. do statemachines czy APW):

$$G^{++} = (q_0(z_1q_0, z_2q_1(z_1q_0, z_2q_2(z_1q_3(z_1q_0, z_2q_1), z_2q_4(z_1q_3, z_2q_4))))))$$

## 10 Grafy automatów

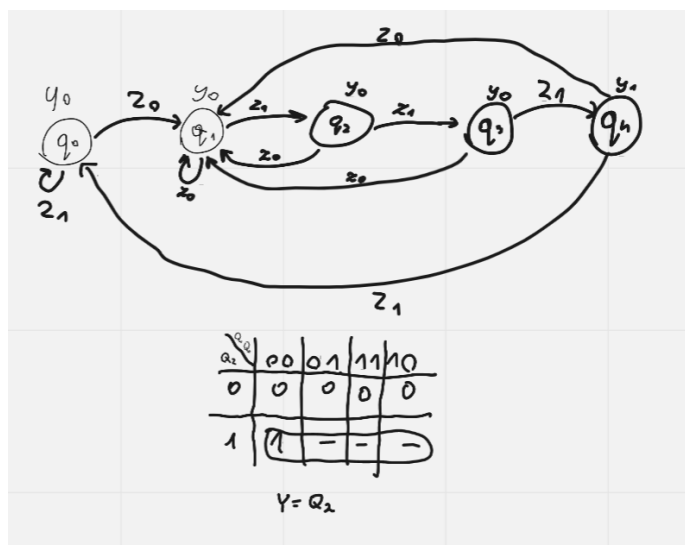
### 10.1 Słówko o automatach

Automaty Moore'a i Mealy tworzone są na przerzutnikach, więc posiadają pamięć. Każdy automat posiada swój stan początkowy i na podstawie ciągu wprowadzonych danych w odpowiednich cyklach zegarowych (zegar może być sterowany ręcznie). Automat ma dać odpowiedni sygnał wyjściowy w zależności od wprowadzonych danych. Automaty to układy cyfrowe realizujące zadane sekwencje (detekcję ciągu, odczytanie słów itp.). Rozróżniamy na naszych zajęciach dwa rodzaje automatów: Moore'a i Mealy. Różnica na dobrą sprawę polega tylko na tym, że w tych pierwszych na wyrysowanym grafie wyjścia znajdują się na stanach, a w drugim - na przejściach. Graf tworzymy prosto z treści zadania. W automatach Moore'a zazwyczaj ostatni stan to jest ten, w którym dana sekwencja jest akceptowana, więc nasze wyjście to  $y_1$ , pozostałe stany mają  $y_0$ . Automat Mealy ma wyjścia na przejściach, toteż najczęściej stanów będzie mniej o 1. Z racji podanej wcześniej cechy, logiczna jedynka pojawi się na wyjściu od razu po wprowadzeniu danych, przed podaniem sygnału na wejście zegarowe przerzutników. Unikać tego można dokładając tuż przed wyjściem (diodą, odbiornikiem) kolejny przerzutnik (typu D) z podpiętym wejściem CLK i RESET. Wtedy, mimo iż układ został zbudowany na zasadach automatu Mealy, działa jak automat Moore'a.

### 10.2 Treść zadania

Automat będący detektorem sekwencji "0111"

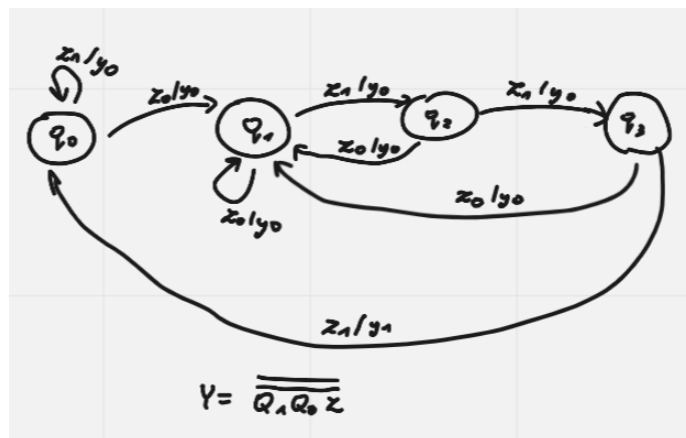
### 10.3 Graf automatu moore'a



Rysunek 46: treść zadania



## 10.4 Graf automatu mealy'ego



Rysunek 47: treść zadania

## 11 Informacje dodatkowe

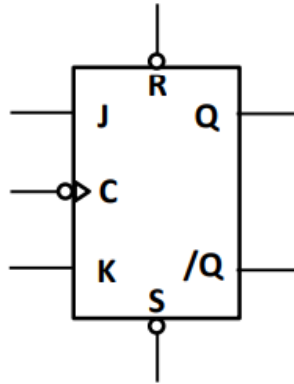
### 11.1 napięcia

Zgodnie z teorią z wykładu: w technologii TTL (Transistor-Transistor Logic)

logiczne zero to napięcie od 0V do 0,8V

logiczne jeden oznacza napięcia od 2,4V do 5V.

### 11.2 zbocza



Rysunek 48: przerzutnik jk

Kółko przy wejściach CLK, RESET, SET oznacza, że dochodzi do zmiany stanu/wyzwolenia na zboczu opadającym czyli podczas przejścia z logicznej jedynki do logicznego zera. Inaczej nazywa się to sterowanie jedynką lub zerem (bez kółek).

### 11.3 tabele dla przerzutników

D	Q(t)	Q(t+1)	Q(t) → Q(t+1)	D
0	0	0	0 → 0	0
0	1	0	0 → 1	1
1	0	1	1 → 0	0
1	1	1	1 → 1	1

Tabela 1: tabela prawdy dla przerzutnika typu D (delay/data)

T	Q(t)	Q(t+1)	Q(t) → Q(t+1)	T
0	0	0	0 → 0	0
0	1	1	0 → 1	1
1	0	1	1 → 0	1
1	1	0	1 → 1	0

Tabela 2: tabela prawdy dla przerzutnika typu T (toggle)

J	K	Q(t)	Q(t+1)	Q(t) → Q(t+1)	J	K
0	0	0/1	0/1	0 → 0	0	-
0	1	0/1	0	0 → 1	1	-
1	0	0/1	1	1 → 0	-	1
1	1	0/1	1/0	1 → 1	-	0

Tabela 3: tabela prawdy dla przerzutnika typu JK (Jedynkujące-Kasujące)

R	S	Q(t)	Q(t+1)	Q(t) → Q(t+1)	R	S
0	0	0/1	0/1	0 → 0	-	0
0	1	0/1	1	0 → 1	0	1
1	0	0/1	0	1 → 0	1	0
1	1	zabroniony	zabroniony	1 → 1	0	-

Tabela 4: tabela prawdy dla przerzutnika typu RS (Reset-Set)

### LEGENDA:

0/1 (w następnej komórce 0/1) - 0 przechodzi w 0 przy danym stanie wejść, 1 przechodzi w 1 przy danym stanie wejść,

0/1 (w następnej komórce 1/0) - 0 przechodzi w 1 przy danym stanie wejść, 1 przechodzi w 0 przy danym stanie wejść,

zabroniony - w przypadku przerzutników RS nie wolno na wejściach podać dwóch jedynek logicznych (zer, jeśli wejścia są zanegowane), przerzutnik się wysypuje.

### Wy tłumaczenie:

Przerzutnik typu D: Delay/Data - przerzuca na wyjście wartość wejścia niezależnie od poprzedniego stanu;

Przerzutnik typu T: Toggle - gdy aktywowany przełącza przerzutnik w stan przeciwny poprzedniemu;

Przerzutnik typu JK: JedynkującoKasujący - wejście J, gdy aktywowane ustawia wyjście przerzutnika w stan 1, wejście K gdy aktywowane ustawia wyjście przerzutnika w stan 0, gdy oba aktywne, zamienia wyjście na przeciwne poprzedniemu;

Przerzutnik typu RS: ResetSet - działa analogicznie do JK, z tym, że wprowadzenie jedynek na obu wejściach jest zabronione; Reset gdy aktywny ustawia na 0 wyjście, Set gdy aktywny ustawia na 1 wyjście.

### Słowa zakończenia

Przerzutniki D, T, JK są synchroniczne (potrzebują wejścia zegarowego), ale nie oznacza to, że nie można robić na nich układów asynchronicznych (wejście zegarowe pierwszego przerzutnika podpinamy do zegara, wejście kolejnego do zanegowanego wyjścia poprzedniego przerzutnika).

Przerzutnik RS jest asynchroniczny, nie posiada wejścia zegarowego (można je wprowadzić, w programach realizujących układy istnieją RS synchroniczne, wystarczy dołożyć 2 bramki NAND do Seta i Resetu z CLK).

## 11.4 bramki na labach

W pracowniach na UNILOGACH (nie próbujcie googlować, nie znajdziecie i tak) korzystamy głównie z bramek NAND (2,3,4 wejściowe), NOR (2,3,4 wejściowe), NOT, XOR oraz przerzutników typu D, JK oraz szyn rozszerzających, dlatego większość zadań było dostosowywane do tego, żeby móc wykorzystać tylko powyższe bramki. Na zdalnym egzaminie może nie mieć to większego znaczenia, lecz w poleceniu może znaleźć się fraza "układ na bramkach/przerzutnikach typu...", więc warto o tym pamiętać.

## 11.5 układy kombinacyjne a sekwencyjne

Układ kombinacyjny to taki, w którym stan wyjścia zależy tylko od stanu wejść, a sekwencyjny posiada pamięć - wyjście zależy od wejść i stanu poprzedniego.

Układ sekwencyjny opisuje 2 funkcje:

- delta ( $\delta$ ) -  $Q \times X \rightarrow Q'$
- lambda ( $\lambda$ ) -  $Q \times X = Y$

## 11.6 minimum bramek, jakie potrzeba

Do zbudowania każdego układu wystarczą 3 bramki: NOT, NOR, NAND, każda bardziej złożona bramka czy przerzutnik można zbudować z powyższych.

## 11.7 wielka piątka układów cyfrowych

Piątka dla każdego układu cyfrowego:

- Q - zbiór stanów układu,
- X - zbiór wejść układu,
- Y - zbiór wyjść układu,
- $\delta$  - funkcja przejść układu,
- $\lambda$  - funkcja wyjść układu.

Piątka dla automatów skończonych:

- Q - zbiór stanów automatu,
- $\Sigma$  - skończony alfabet wejściowy automatu,
- $\delta$  - funkcja przejść automatu,
- $q_0$  - stan początkowy układu (należący do Q),
- F - zbiór stanów końcowych układu (zawierające się w Q).

## 11.8 te pościgi, te wybuchy

Hazard występuje w układach, gdy na żadnej z dróg nie ma sprzężenia zwrotnego, każda prowadzi przez układ kombinacyjny

Hazard zasadniczy występuje, gdy na jednej z dróg znajduje się układ kombinacyjny, a druga prowadzi przez układ pamięci

Wyścig pojawia się, gdy każda z dróg prowadzi przez układ ze sprzężeniem zwrotnym, na każdej z dróg występuje układ pamięci (sygnał "ściga się", żeby zmienić dwa bity co może naruszyć stabilność układu)