LUC - ultimate edition

Damian Gnieciak & Szymon Ludwiniak

31 stycznia 2022

This page intentionally left blank

Spis treści

1 Quine-McCluskey

1.1 Treść zadania

1. QUINE - McCluskey

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

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

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_3	0	0	1	1
	m_9	1	0	0	1
4.	m_7	0	1	1	1
	m_{11}	1	0	1	1
5.	m_{15}	1	1	1	1

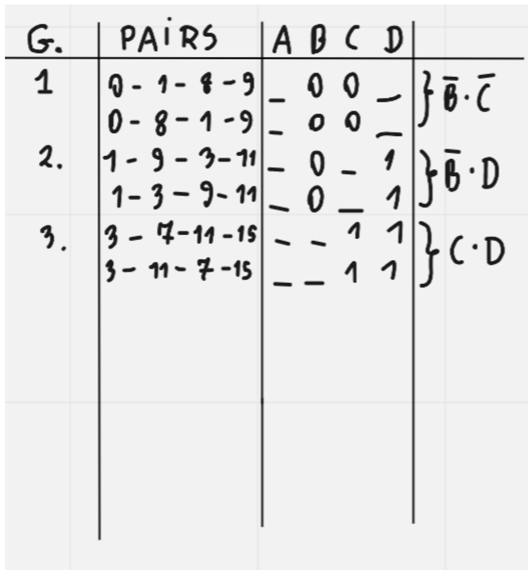
1.2.3 Składnik X

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	_
	$m_0 - m_8$	_	0	0	0
2.	$m_1 - m_3$	0	0	_	1
	$m_1 - m_9$	_	0	0	1
	$m_8 - m_9$	1	0	0	_
3.	$m_3 - m_7$	0	_	1	1
	$m_3 - m_{11}$	_	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

1.2.4 Szalony naukowiec

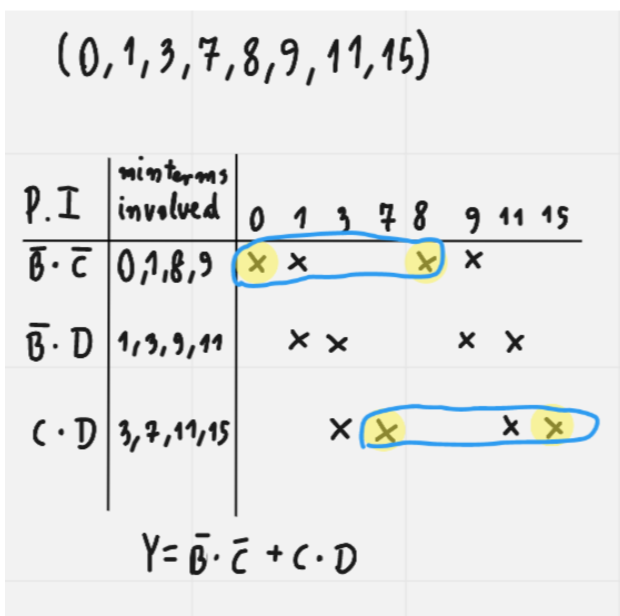
Nasze pary łączymy następnie w czwórki, stosując dokładnie tą samą metodę, mechanizm ten powtarzamy, póki możliwe jest uszczuplenie tabelki.
po maksymalnej optymalizacji naszej tabeli, łącząc klamrami wyrażenia z tych samych grup wpisujemy wyrażenia boolowskie analogicznie do siatek Karnaugha.



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

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 karnaugha)

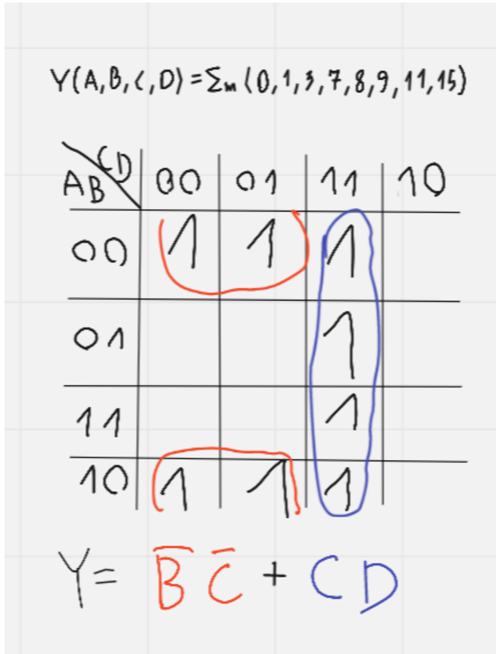


(0, 1, 3, 7, 8, 9, 11, 15)

P.I	minterms involved	0	1	3	7	8	9	11	15
$\bar{B} \cdot \bar{C}$	0,1,8,9	×	×			×	×		
$\bar{B} \cdot D$	1,3,9,11		×	×			×	×	
$C \cdot D$	3,7,11,15			×	×			×	×

$Y = \bar{B} \cdot \bar{C} + C \cdot D$

1.3 Rozwiązanie siatką karnaugha



1.4 Klik

http://quinemccluskey.com/

2 Siatka karnaugh

2.1 Treść zadania

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

2.2 Rozwiązanie

2.2.1 Siatka z wartościami liczbowymi

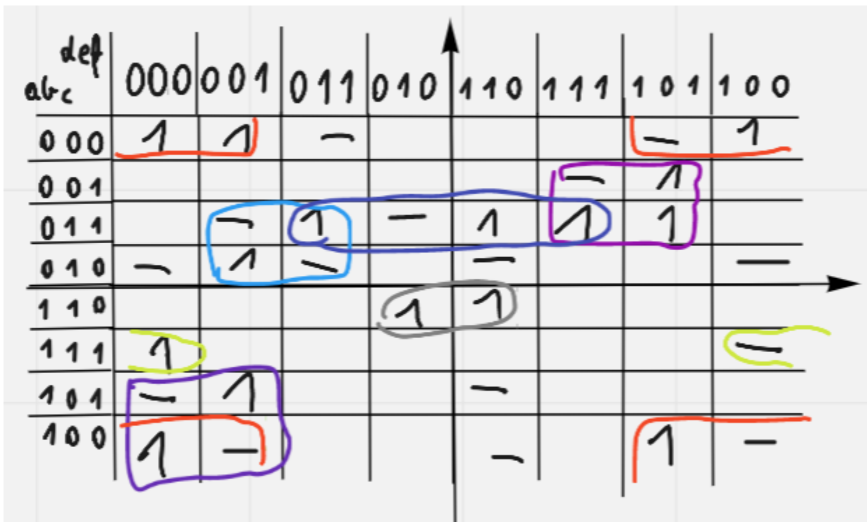
abc \ def	000	001	011	010	110	111	101	100
000	0	1	3	2	6	7	5	4
001	8	9	11	10	14	15	13	12
011	24	25	27	26	30	31	29	28
010	16	17	19	18	22	23	21	20
110	48	49	51	50	54	55	53	52
111	56	57	59	58	62	63	61	60
101	40	41	43	42	46	47	45	48
100	32	33	35	34	38	39	37	36

2.2.2 Siatka bez połączonych jedynek

abc \ def	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	–

2.2.3 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 tyczy 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.



2.2.4 Wynik

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

2.3 Klik

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

3 Kody

3.1 Dwójkowo-dziesiętne (BCD)

3.1.1 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 1: BCD

3.1.2 Kod +3

Kod „+3” (D = B + 3) – kod niewagowy, samouzupełniający (do 9 przez negację wszystkich bitów)

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 2: +3

3.1.3 Kod 2-4-2-1

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

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

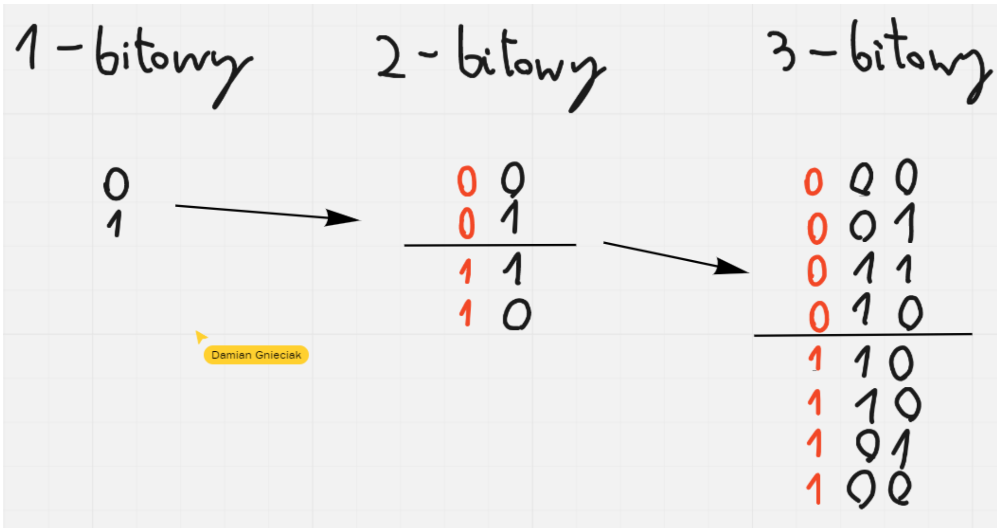
Rysunek 3: 2421

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).

3.2 Refleksyjne

3.2.1 Kod gray'a

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. Przykładem kodu refleksyjnego jest kod Graya.



Rysunek 4: Gray

3.3 Detekcyjne i korekcyjne

3.3.1 Kod 1 z 10

	z kontrolą parzystości	„1 z 10”	„2 z 5”
0	00000	000000000**1**	00011
1	00011	00000000**1**0	00101
2	00101	0000000**1**00	01001
3	00110	000000**1**000	10001
4	01001	00000**1**0000	00110
5	01010	0000**1**00000	01010
6	01100	000**1**000000	01010
7	01111	00**1**0000000	01100
8	10001	0**1**00000000	10100
9	10010	**1**000000000	11000

Rysunek 5: 1 z 10

3.3.2 Kod 2 z 5

	z kontrolą parzystości	„1 z 10”	„2 z 5”
0	0 0 0 0 0	0 0 0 0 0 0 0 0 0 **1**	0 0 0 1 1
1	0 0 0 1 1	0 0 0 0 0 0 0 0 **1** 0	0 0 1 0 1
2	0 0 1 0 1	0 0 0 0 0 0 0 **1** 0 0	0 1 0 0 1
3	0 0 1 1 0	0 0 0 0 0 0 **1** 0 0 0	1 0 0 0 1
4	0 1 0 0 1	0 0 0 0 0 **1** 0 0 0 0	0 0 1 1 0
5	0 1 0 1 0	0 0 0 0 **1** 0 0 0 0 0	0 1 0 1 0
6	0 1 1 0 0	0 0 0 **1** 0 0 0 0 0 0	0 1 0 1 0
7	0 1 1 1 1	0 0 **1** 0 0 0 0 0 0 0	0 1 1 0 0
8	1 0 0 0 1	0 **1** 0 0 0 0 0 0 0 0	1 0 1 0 0
9	1 0 0 1 0	**1** 0 0 0 0 0 0 0 0 0	1 1 0 0 0

Rysunek 6: 2 z 5

3.3.3 Kod z kontrolą parzystości

	z kontrolą parzystości	„1 z 10”	„2 z 5”
0	0 0 0 0 0	0 0 0 0 0 0 0 0 0 **1**	0 0 0 1 1
1	0 0 0 1 1	0 0 0 0 0 0 0 0 **1** 0	0 0 1 0 1
2	0 0 1 0 1	0 0 0 0 0 0 0 **1** 0 0	0 1 0 0 1
3	0 0 1 1 0	0 0 0 0 0 0 **1** 0 0 0	1 0 0 0 1
4	0 1 0 0 1	0 0 0 0 0 **1** 0 0 0 0	0 0 1 1 0
5	0 1 0 1 0	0 0 0 0 **1** 0 0 0 0 0	0 1 0 1 0
6	0 1 1 0 0	0 0 0 **1** 0 0 0 0 0 0	0 1 0 1 0
7	0 1 1 1 1	0 0 **1** 0 0 0 0 0 0 0	0 1 1 0 0
8	1 0 0 0 1	0 **1** 0 0 0 0 0 0 0 0	1 0 1 0 0
9	1 0 0 1 0	**1** 0 0 0 0 0 0 0 0 0	1 1 0 0 0

Rysunek 7: z kontrolą parzystości

3.3.4 Kod Hamming'a

Hamminga

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

Rysunek 8: Hamming

4 Algebra boole'a

4.1 aksjomaty

Ludwiniak smierdzi hehehe

4.1.1 Prawo przemienności

$A + B = B + A$

$A \cdot B = A \cdot B$

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

$A + 0 = 0$

$A \cdot 0 = 0$

$A + 1 = 1$

$A \cdot 1 = A$

$A + A = A$

$A \cdot A = A$

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

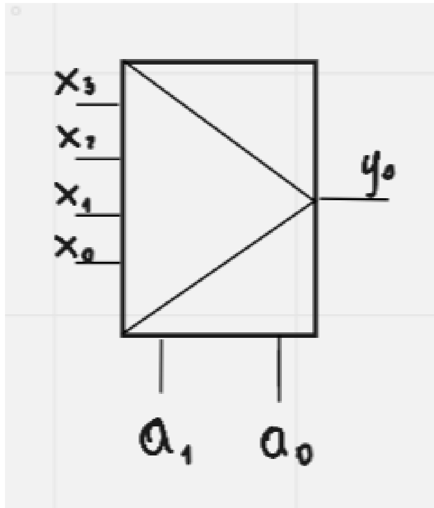
$A \cdot \overline{B} + B = A + B$

4.2 Klik

`http://www.zsk.ict.pwr.wroc.pl/zsk/repository/dydaktyka/ptcm/wyk/tc1_9_wyk_3.pdf`

5 Multipleksery

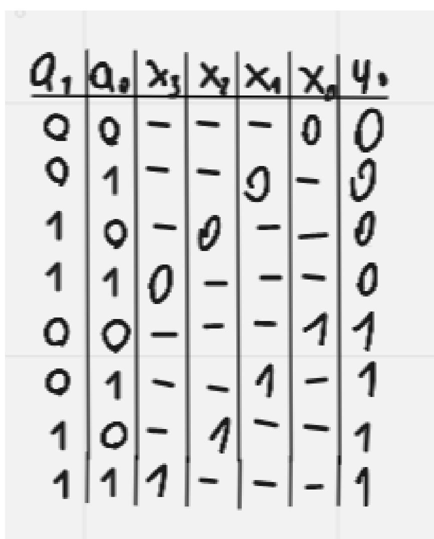
5.1 Symbol multipleksera 4-bitowego



Rysunek 9: 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

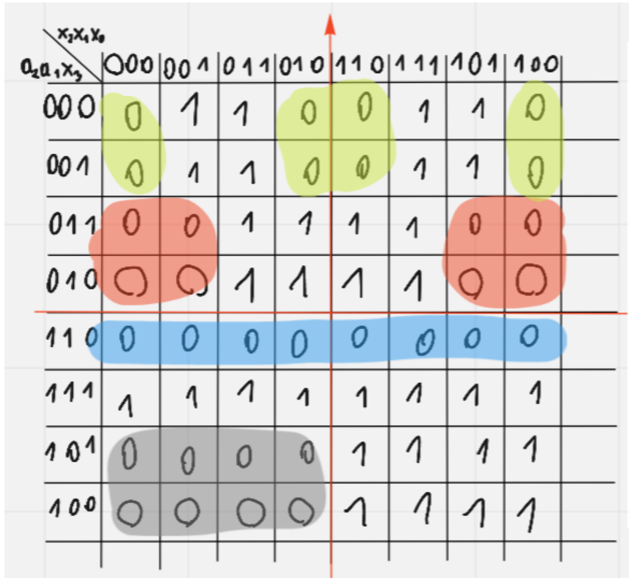


a_1	a_0	x_3	x_2	x_1	x_0	y_0
0	0	-	-	-	0	0
0	1	-	-	0	-	0
1	0	-	0	-	-	0
1	1	0	-	-	-	0
0	0	-	-	-	1	1
0	1	-	-	1	-	1
1	0	-	1	-	-	1
1	1	1	-	-	-	1

Rysunek 10: 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 multipleksera 4-bitowego



Rysunek 11: siatka karnaugha dla multipleksera 4-bitowego

Rysunek 12: siatka karnaugh

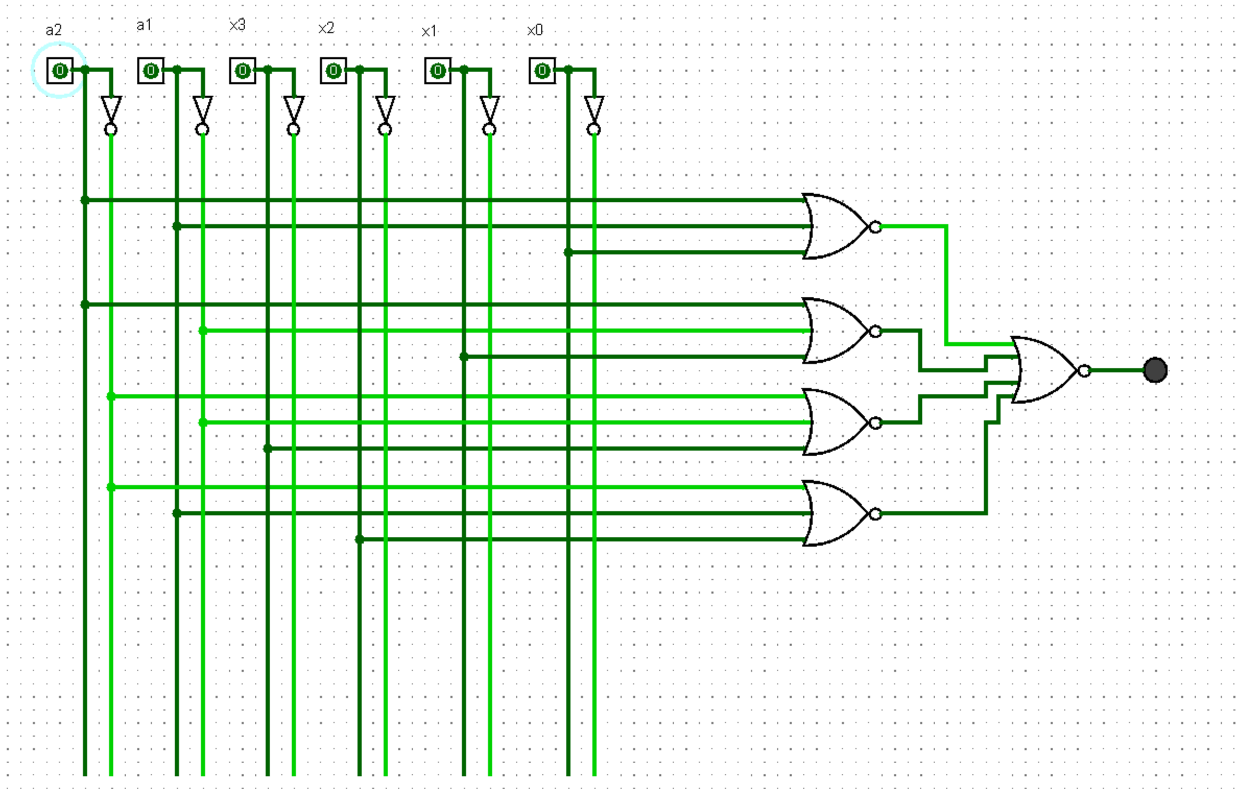
Multiplekser był wykonywany w pełni na bramkach NOR więc w ramach ułatwienia, w siatce zaznaczane były implicenty (zera).

5.4 Równie wynikowe

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

Rysunek 13: funkcja z siatki karnaugh

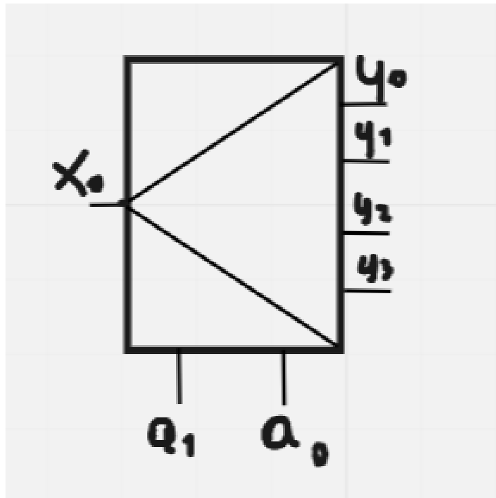
5.5 Schemat układu



Rysunek 14: schemat układu

6 Demultipleksery

6.1 Symbol demultipleksera 4-bitowego



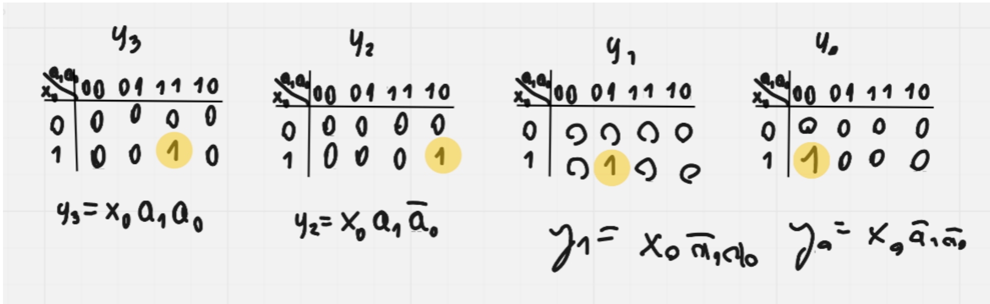
Rysunek 15: Symbol demultipleksera

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

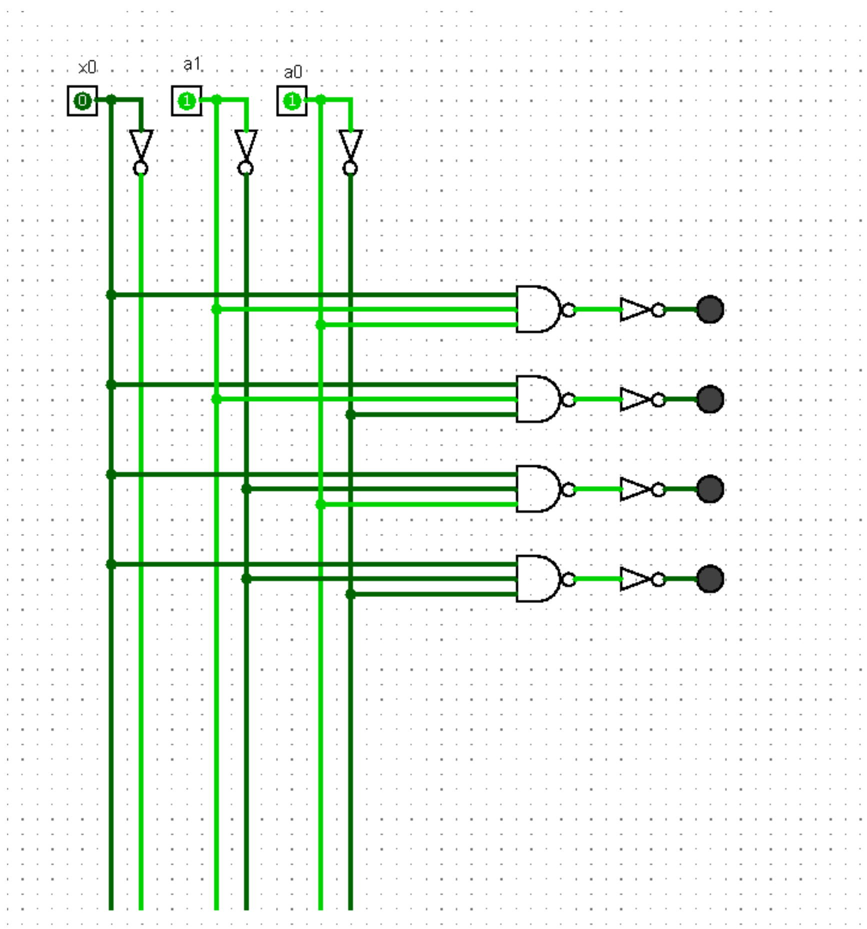
Rysunek 16: tabela prawdy dla demultipleksera 4-bitowego

6.3 Siatka karnaugh dla demultipleksera 4-bitowego



Rysunek 17: siatka karnaugh

6.4 Schemat układu



Rysunek 18: schemat układu

7 Liczniki asynchroniczne

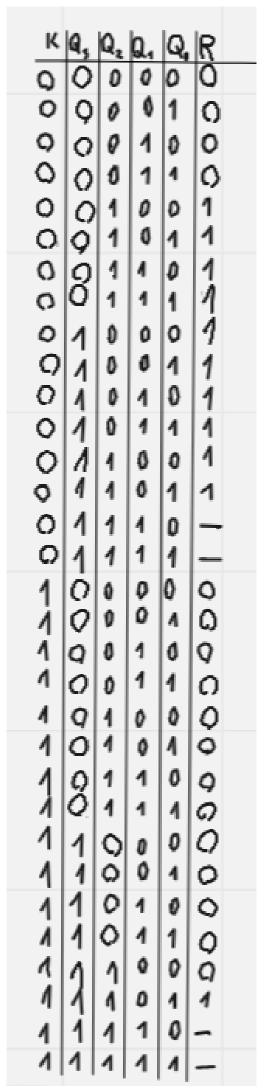
7.1 Zadanie

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

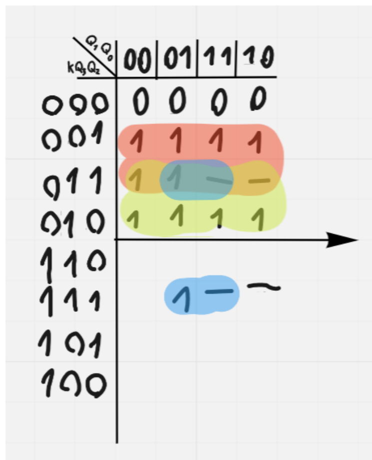


K	Q_3	Q_2	Q_1	Q_0	R
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	0
0	0	1	0	0	1
0	0	1	0	1	1
0	0	1	1	0	1
0	0	1	1	1	1
0	1	0	0	0	1
0	1	0	0	1	1
0	1	0	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
0	1	1	0	1	1
0	1	1	1	0	—
0	1	1	1	1	—
1	0	0	0	0	0
1	0	0	0	1	0
1	0	0	1	0	0
1	0	0	1	1	0
1	0	1	0	0	0
1	0	1	0	1	0
1	0	1	1	0	0
1	0	1	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	—

Rysunek 19: 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.

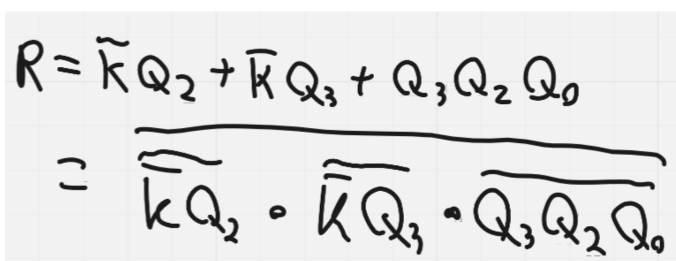
7.4 Siatka Karnaugh



Rysunek 20: Siatka Karnaugh

Z tabeli robimy prostą siateczkę.

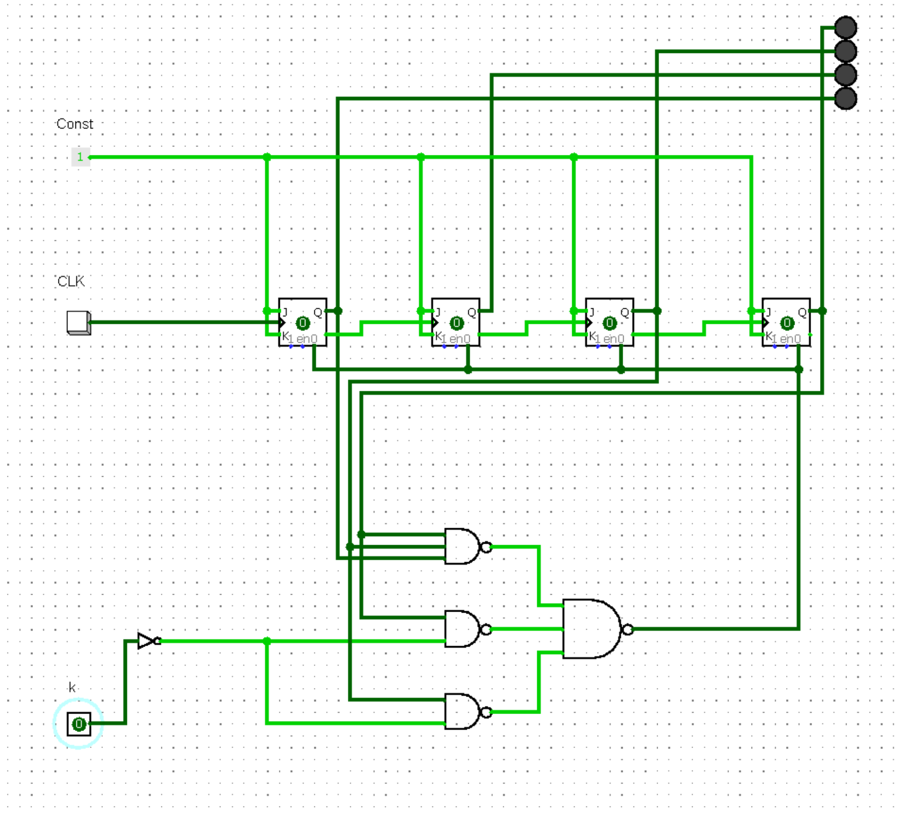
7.5 Funkcja resetu



Rysunek 21: Funkcja resetu

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

7.6 Schemat układu



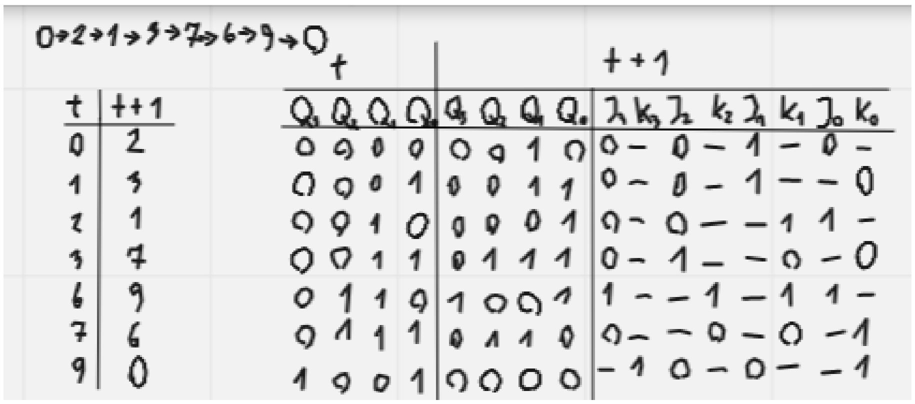
Rysunek 22: Schemat układu

8 Liczniki synchroniczne

8.1 Zadanie

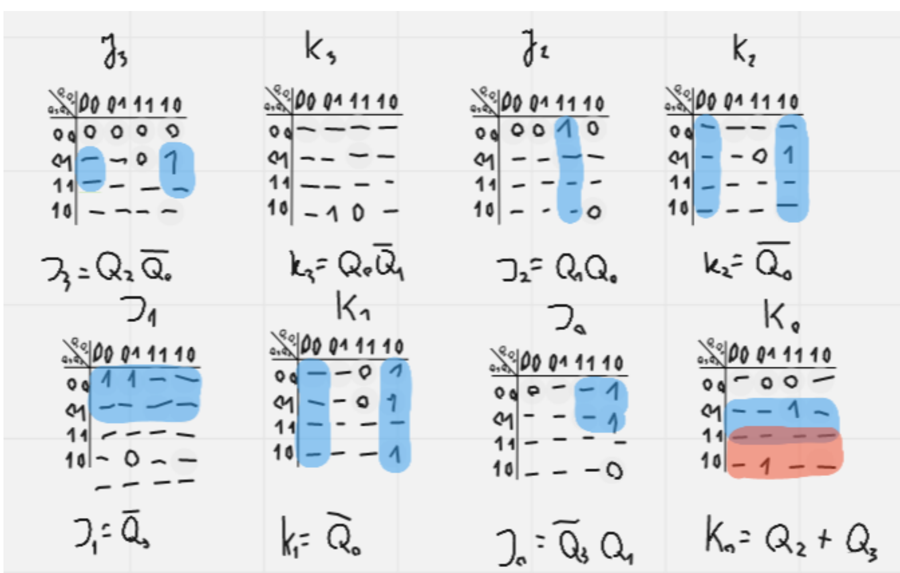
Zaprojektować licznik synchroniczny realizujący liczenie w postaci $0 \to 2 \to 1 \to 3 \to 7 \to 6 \to 9$. Układ zostanie zrealizowany na przerzutnikach JK - Jebać Kleksa))))
W tym zadaniu najlepiej sprawdza się instrukcja doktora Antoniego Sterny, polecam przeczytać bo jest gitówa.

8.2 Tabela przejść



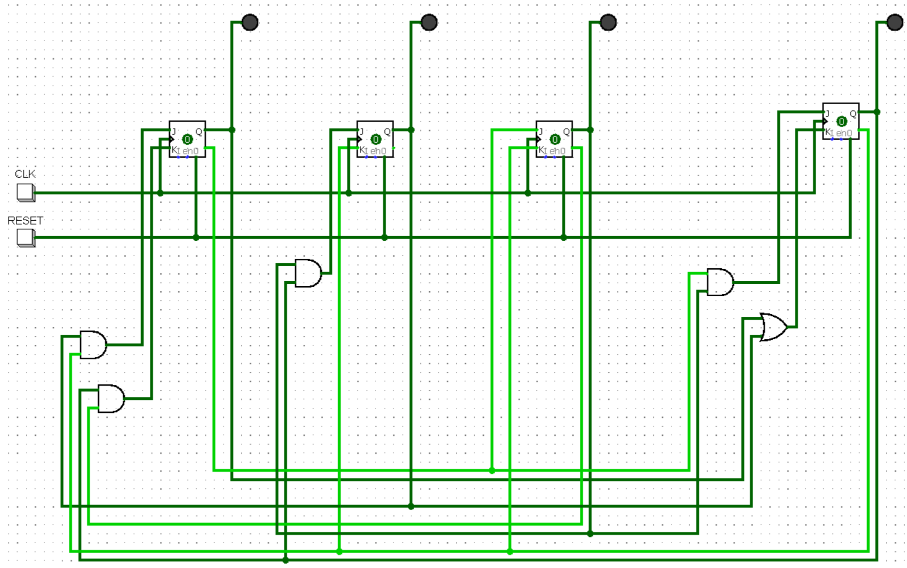
Rysunek 23: Tabela przejść

8.3 Siatki Karnaugh



Rysunek 24: Siatki Karnaugh

8.4 Schemat układu



Rysunek 25: Schemat układu

8.5 Klik

`http://staff.iiar.pwr.wroc.pl/antoni.sterna/luc/LUC_synteza_licznikow.pdf`

9 Wyrażenie regularne

9.1 Treść zadania

$$
\begin{array}{ll|l}
s_1 = (z_1 + z_2z_1 + z_2z_2z_1) * z_2z_2 & & y_1 \\
s_2 = (z_1 + z_2z_1 + z_2z_2z_1) * z_2z_2(z_2) * \bar{z}_1 & & y_2 \\
s_3 = \overline{(s_1 + s_2)} & & y_0 = \varepsilon
\end{array}
$$

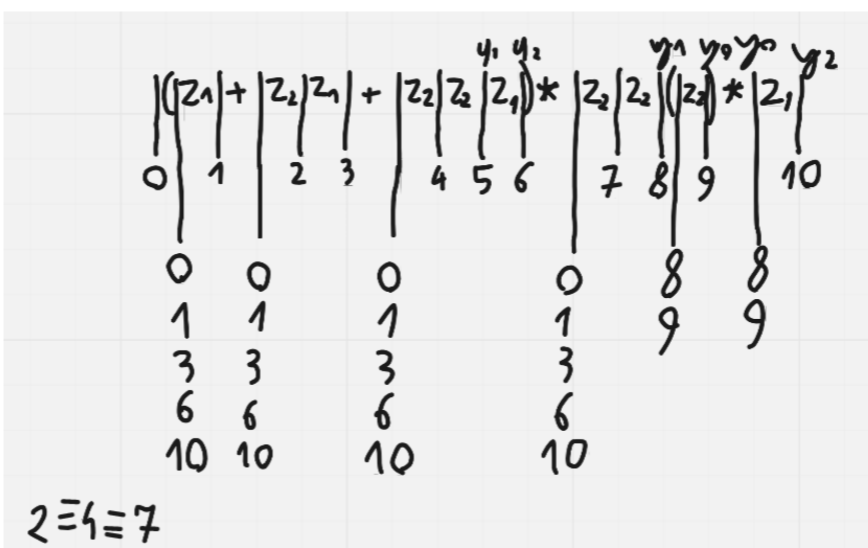
Rysunek 26: treść zadania

Wyrażenia regularne to jeden wielki pierdolnik. Nasze wyrażenie podane w zadaniu podaje na wyjściu trzy możliwe wartości, y_0, y_1, y_2. y_2 pojawia się wtedy gdy nie jest aktywne ani y1 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.

9.2 Czarna magia i techniki zakazane

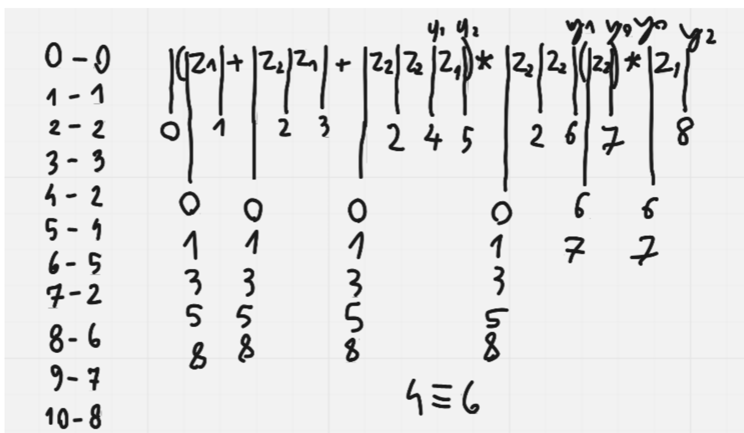
9.2.1 Prolog



Rysunek 27: treść zadania

Tutaj zaczyna się dym. Stawiamy krótkie pionowe kreski po każdej zetce oraz na samym początku wyrażenia (tam oczywiście zapisujemy 0). Drugim krokiem jest narysowanie długich kresek na początku każdego słowa czyli przed dowolną grupką naszych z_n. Teraz trafiamy na jeden z bardziej problematycznych momentów, musimy ustalić do którego słowa można wejść z którego miejsca. Patrząc po rysunku, do słowa pierwszego w wyrażeniu czyli po prostu z_1 można wejść zera, ale można też do niego trafić gdy już przejdziemy przez wyraz z_1 czyli również z jedynki. Patrzymy dalej: jeśli przejdziemy przez słowo z_2z_1 możemy znowu przejść do słowa pierwszego (w naszym przypadku dalej z_1) czyli dopisujemy trójkę (bo nią kończy się słowo z_2z_1). To samo dla ostatniego słowa w iteracji. Ważnym jest że iterację można również ominąć, nikt ci nie każe wchodzić do pętli więc do pierwszego słowa za iteracją też można przejść z zera. Jadąc do samego końca zauważamy, że z dziesiątki znowu można przejść na sam początek (zapętlamy działanie całego układu). Taka sytuacja z tego co zauważyliśmy ma miejsce tylko gdy wyrażenie zaczyna się od iteracji (tutaj jak ktoś wie coś więcej to prosimy o ekspertyzę). Patrząc na drugą iterację widzimy że do niej można przejść tylko z ósemki i dziewiątki. W końcu z samego początku żeby się tam dostać musimy przejść przez z_2z_2. Sporo tego a to dopiero początek xD. W lewym dolnym rogu mamy zapisane że 2 jest równoważna z czwórką i siódemką. Patrząc na wyrażenie widzimy że przejście do dwójki, czwórki i siódemki odbywa się poprzez przejście z 0,1,3,6,10 za pomocą z_2 czyli są one tożsame. Koniec kroku pierwszego. Xue hua piao piao.

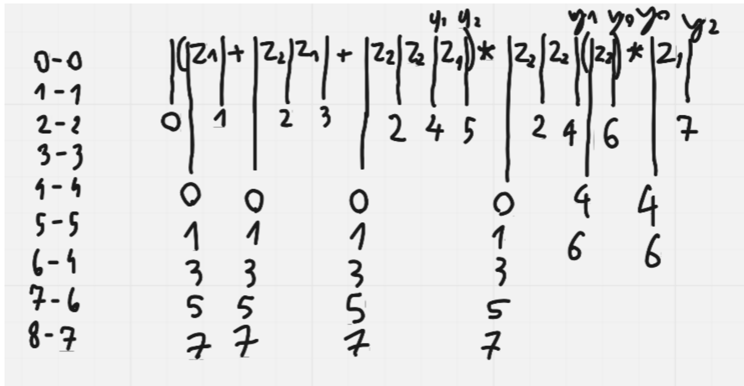
9.2.2 Niebagatelny zwrot akcji



Rysunek 28: treść zadania

Teraz będzie w miarę lekko, jako że $2 \equiv 4 \equiv 7$ to musimy trochę namieszać, rozpisujemy taką tabelkę jak po prawej stronie i widzimy, że: zero pozostało zerem, dwójka dwójką ale czwórka stała się dwójką. Skoro czwórka jest dwójką to tracimy jedną cyfrę, więc żeby zachować ciągłość przesuwamy wszystkie następne o jedną pozycję, czyli piątka staje się czwórką, szóstka piątką itd. Gdy dojdziemy do siódemki zapisujemy że jest ona równa dwójce, czyli oprócz tego że pozbyliśmy się czwórki to teraz nie ma jeszcze siódemki, czyli ósemka staje się szóstką.
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.

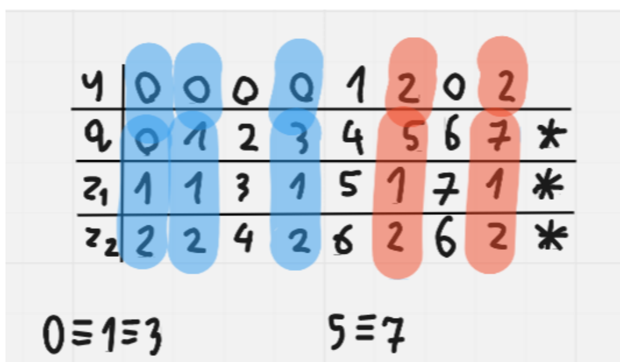
9.2.3 Epilog



Rysunek 29: treść zadania

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 babola.

9.3 Tabela stanów

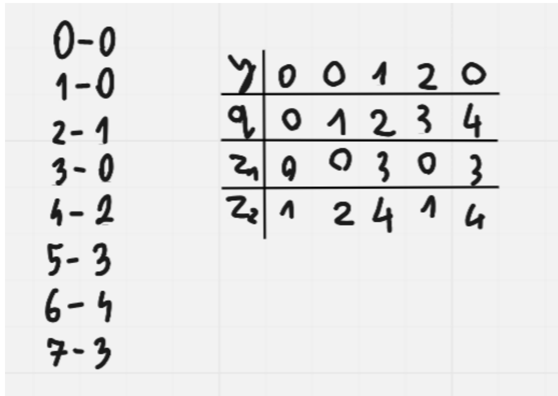


Rysunek 30: treść zadania

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 zminimalizowania tabeli, należy pamiętać o zmienieniu kolejności numerowania stanów.



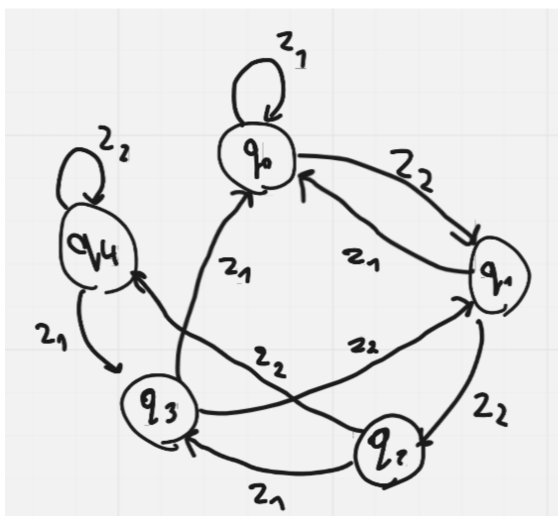
0-0
1-0
2-1
3-0
4-2
5-3
6-4
7-3

y	0	0	1	2	0
q	0	1	2	3	4
z_1	0	0	3	0	3
z_2	1	2	4	1	4

Rysunek 31: treść zadania

9.5 Finalny graf

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



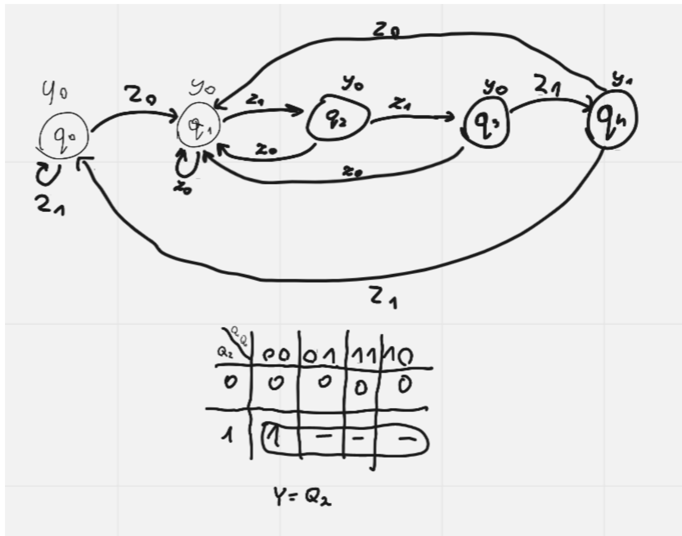
Rysunek 32: treść zadania

10 Grafy automatów

10.1 Treść zadania

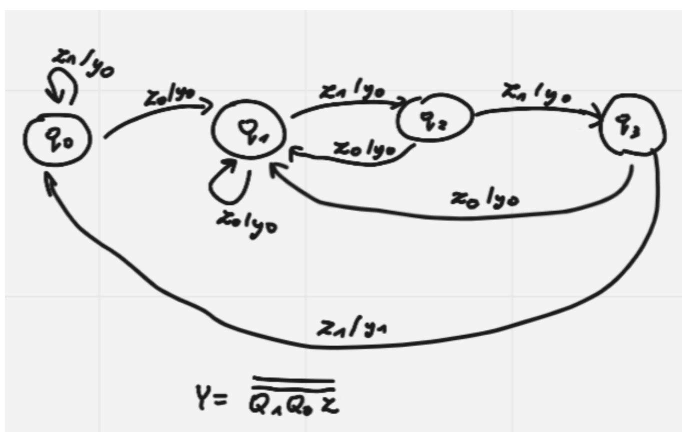
Automat będący detektorem sekwencji "0111"

10.2 Graf automatu moore'a



Rysunek 33: treść zadania

10.3 Graf automatu mealy'ego



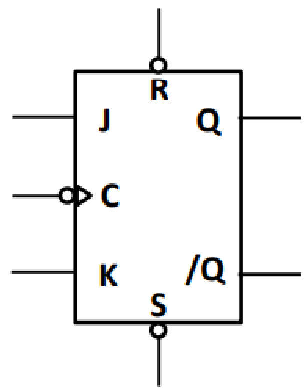
Rysunek 34: treść zadania

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 35: 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.