Spis treści

1 Wygeneruj funkcję Boolowską zgodnie z algorytmem: 2

2 Wykonaj minimalizację funkcji Boolowskiej: 2

3 Zrealizuj funkcję dla systemu mikroprocesorowego: 4

4 Zrealizuj funkcję sprzętowo: 6

1 Wygeneruj funkcję Boolowską zgodnie z algorytmem:

1.0.1 Prezentacja implementacji rozwiązania:

	00	01	11	10	x3x4
00	A	F	G	M	
01	B	C	H	O	
11	C	D	J	K	
10	P	E	L	R	
x1x2					

Rysunek 1: Dla Łukasz Kordon

1.0.2 Wnioski:

Mamy tabelkę z literami, w której zaznaczamy litery występujące w naszym imieniu i nazwisku.

2 Wykonaj minimalizację funkcji Boolowskiej:

2.1 Zapisz funkcję Boolowską za pomocą wyrażeń algebraicznych:

2.1.1 Prezentacja implementacji rozwiązania:

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

Rysunek 2: Zastąpienie liter cyframi

$$A = \bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot \bar{x}_4$$
$$B = x_1 \cdot x_2 \cdot \bar{x}_3 \cdot x_4$$
$$C = \bar{x}_1 \cdot x_2 \cdot x_3 \cdot \bar{x}_4$$
$$D = x_1 \cdot \bar{x}_2 \cdot x_3 \cdot \bar{x}_4$$
$$E = x_1 \cdot \bar{x}_2 \cdot x_3 \cdot \bar{x}_4$$
$$Y = A + B + C + D$$

Rysunek 3: Realizacja funkcji logicznej względem jedynek.

2.1.2 Wnioski:

Funkcja Boolowska za pomocą wyrażeń algebraicznych.

2.2 Zapisz minimalną postać dysjunkcyjną:

2.2.1 Prezentacja implementacji rozwiązania:

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

Rysunek 4: Tabela dla minimalnej postaci dysjunkcyjnej

$$A = \bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot \bar{x}_4$$
$$B = x_1 \cdot x_2 \cdot \bar{x}_3 \cdot x_4$$
$$C = x_2 \cdot x_3 \cdot \bar{x}_4$$
$$D = x_1 \cdot x_3 \cdot \bar{x}_4$$
$$F = A + B + C + D$$

Rysunek 5: Minimalna postać dysjunkcyjna

$$Y = \overline{\bar{A} + \bar{B} + \bar{C} + \bar{D}}$$
$$Y = \overline{\bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot \bar{x}_4} \cdot \overline{x_1 \cdot x_2 \cdot \bar{x}_3 \cdot x_4} \cdot \overline{x_2 \cdot x_3 \cdot \bar{x}_4} \cdot \overline{x_1 \cdot x_3 \cdot \bar{x}_4}$$

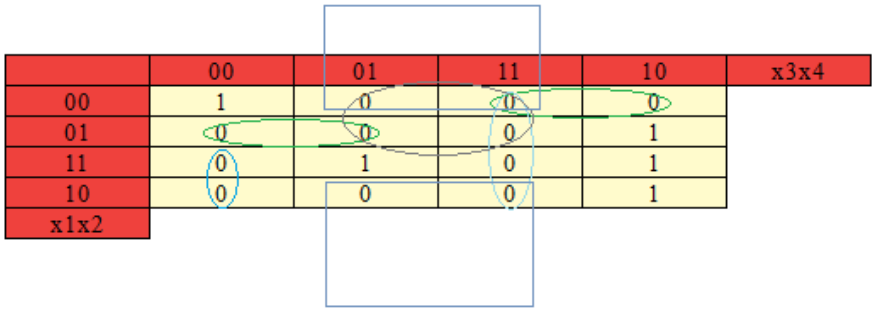
Rysunek 6: Wyrażenie przekształcone prawem de Morgana

2.2.2 Wnioski:

Rysunek 4 przedstawia tabelę z zaznaczonymi 1 na bazie których wyznaczona zostanie minimalna postać dysjunkcyjna z tysunku 5. Rysunek 6 przedstawia wyrażenie przekształcone prawem de Morgana.

2.3 Zapisz minimalną postać koniunkcyjną:

2.3.1 Prezentacja implementacji rozwiązania:



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

Rysunek 7: Tabela dla minimalnej postaci koniunkcyjnej

$$A = \bar{x}_1 + x_4$$
$$B = \bar{x}_1 + \bar{x}_2 + x_3$$
$$C = \bar{x}_1 + x_2 + \bar{x}_3$$
$$D = x_3 + x_4$$
$$E = x_1 + \bar{x}_3 + \bar{x}_4$$
$$F = \bar{x}_2 + x_4$$
$$Y = A \cdot B \cdot C \cdot D \cdot F$$

Rysunek 8: Minimalna postać koniunkcyjna

2.3.2 Wnioski:

W przypadku minimalnej postaci koniunkcyjnej wyzaczenie z rysunku 8 uzyskuje się na bazie zer z rysunku 7.

3 Zrealizuj funkcję dla systemu mikroprocesorowego:

3.1 Zapisz funkcję w języku wysokiego poziomu:

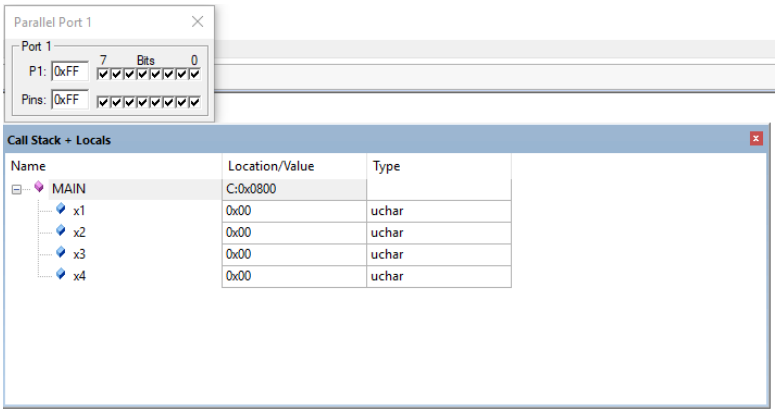
3.1.1 Prezentacja implementacji rozwiązania:

```
1000 #include "aduc831.h"   //Definitions of ADuC831 registers name
     #include "stdint.h"    //Standard integers
1002 #include "stdfloat.h" //Standard float
     #include "IO.h"        //Input/output definitions
1004
     #define ustaw(bajt,nr_bitu) (bajt|=(1<<nr_bitu))
1006 #define skasuj(bajt,nr_bitu) (bajt&=~(1<<nr_bitu))
     #define sprawdz(bajt,nr_bitu) ((bajt&(1<<nr_bitu))&&1)
1008
     void main(void)
1010 {
       unsigned char x1,x2,x3,x4;
1012   while(1)
```

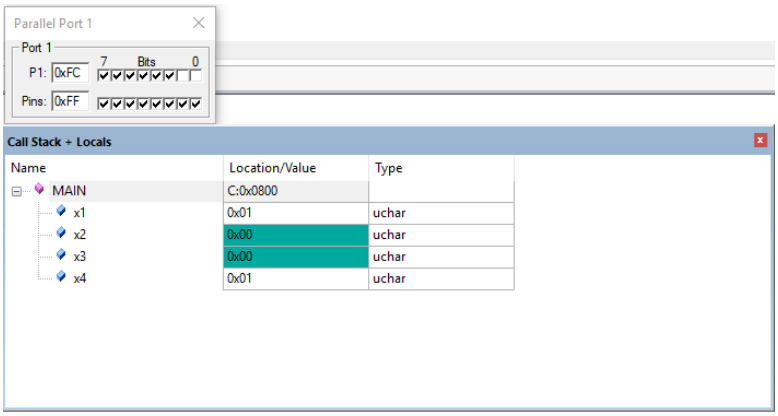
```
  {
1014    x1=sprawdz(P0,0);
    x2=sprawdz(P0,1);
1016    x3=sprawdz(P0,2);
    x4=sprawdz(P0,3);
1018
      if((!x1&!x2&!x3&!x4) | (x1&x2&!x3&x4) | (x2&x3&!x4) | (x2&x3&!x4))
1020    ustaw(P1,0);
    else
1022    skasuj(P1,0);

1024    if((!x1&x4) & (!x1&!x2&x3) & (!x1&x2&!x3) & (x3&x4) & (x1&!x3&!x4) & (!x2&
    x4))
      ustaw(P1,1);
1026  else
      skasuj(P1,1);
1028  }
}
```

3.1.2 Prezentacja wyników testów:



Rysunek 9: Stan 1



Rysunek 10: Stan 2

3.1.3 Wnioski:

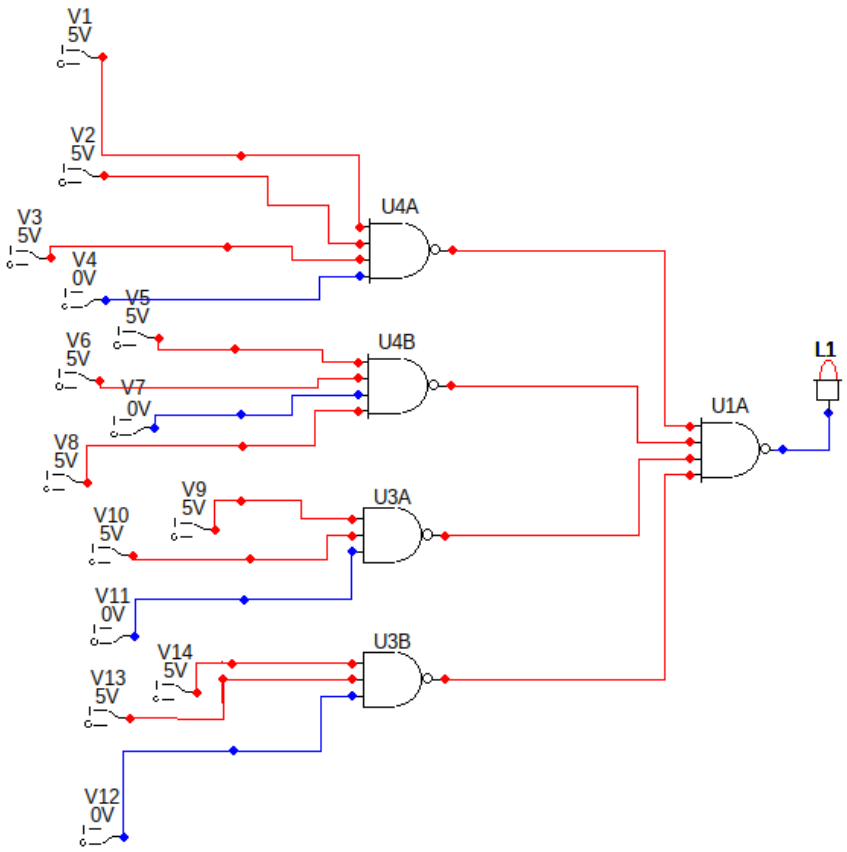
Realizacja funkcji z poprzednich punktów w postaci algorytmu zaimplementowanego w mikrokontrolerze co przedstawia listing w części prezentacji implementacji rozwiązania. Zastosowano makra do poprawiające czytelność kodu oraz przyszpieszające pisanie algorytmu. Czterem zmiennym przypisano typ char bez znaku. W pętli nieskończonej zastosowano warunki logiczne, które ustawiają dany bit lub kasują w zależności od ich zaistnienia. Rysunek 9 przedstawia stan wyjściowy natomiast rysunek 10 przedstawia spełnienie warunków z pętli oraz skasowanie logicznej jedynki na pinie 0 oraz 1 portu pierwszego.

4 Zrealizuj funkcję sprzętowo:

4.1 W układzie małej skali integracji SSI:

4.1.1 Zapisz funkcję tylko za pomocą bramek NAND:

Prezentacja implementacji rozwiązania:

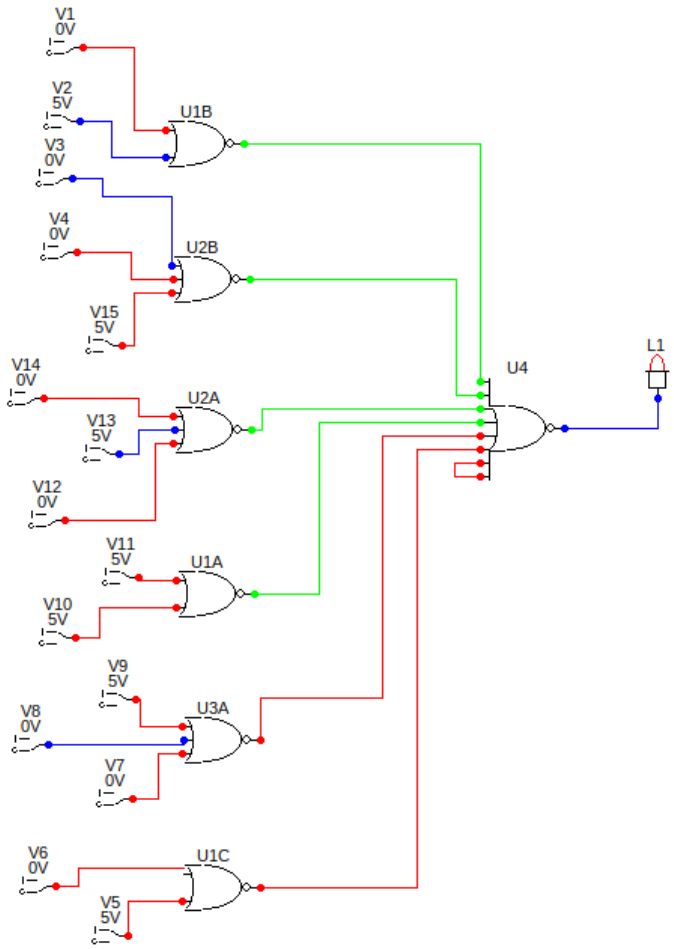


Rysunek 11: NAND

Wnioski: Realizacja w programie Circuit Maker za pomocą bramek NAND została przedstawiona na rysunku 11. Jesto to implementacja funkcji dysjunkcyjnej. Po odpowiednim ustawieniu zmiennych wejściowych uzyskujemy stan logiczny dodatni lub ujemny na wyjściu.

4.1.2 Zapisz funkcję tylko za pomocą bramek NOR:

Prezentacja implementacji rozwiązania



Rysunek 12: NOR

Wnioski: Realizacja w programie Circuit Maker za pomocą bramek NOR została przedstawiona na rysunku 12. Jesto to implementacja funkcji koniunkcyjnej. Po odpowiednim ustawieniu zmiennych wejściowych uzyskujemy stan logiczny dodatni lub ujemny na wyjściu.

4.1.3 Zapisz funkcję w tylko za pomocą dowolnych bramek:

4.2 W układzie średniej skali integracji MSI:

4.2.1 MUX(2)-NAND:

Prezentacja implementacji rozwiązania:

		0			1	
	A	00	x3x4	B	01	x3x4
0	00	1	0	00	0	0
1	01	0	1	01	0	1
3	11	0	3	11	1	3
4	10	0	4	10	0	4
	x1x2			x1x2		

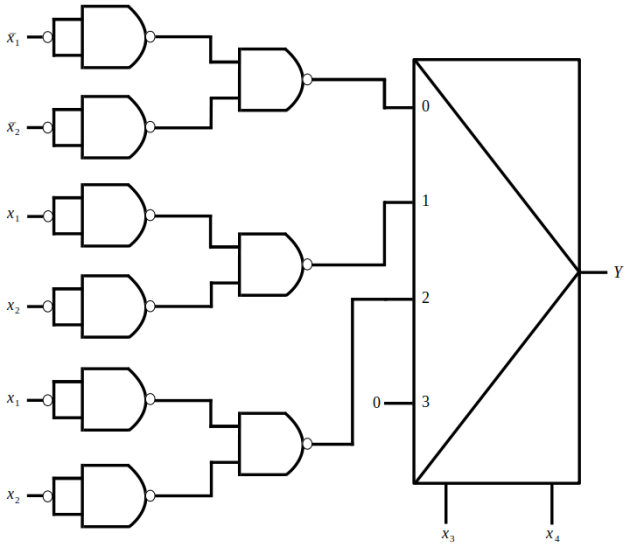
Rysunek 13: Minimalizacja funkcji resztkowych przy strukturze MUX(2)-NAND

		3			2	
x3x4	C	11	x3x4	D	10	x3x4
0	00	0	0	00	0	
1	01	0	1	01	1	
3	11	0	3	11	1	
4	10	0	4	10	1	
	x1x2			x1x2		

Rysunek 14: Minimalizacja funkcji resztkowych przy strukturze MUX(2)-NAND cd.

$$A = \bar{x}_1 + \bar{x}_2 \quad C = 0$$
$$B = x_1 + x_2 \; D = x_1 + x_2$$

Rysunek 15: Minimalizacja funkcji resztkowych przy strukturze MUX(2)-NAND - wynik

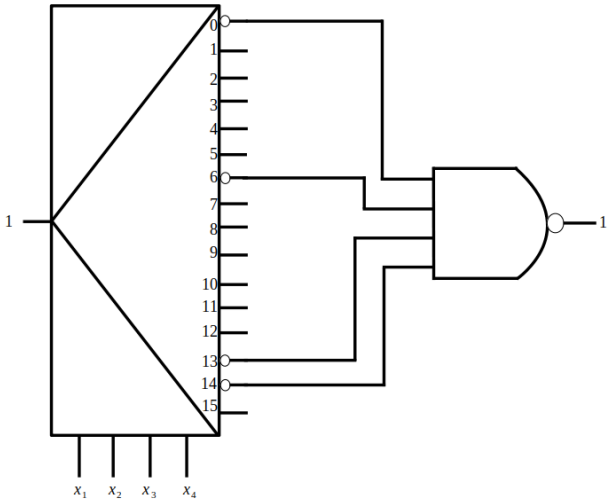


Rysunek 16: MUX(2)-NAND

Wnioski: Dzięki przeprowadzeniu minimalizacji funkcji resztkowych przedstawionych na rysunkach 13 oraz 14 uzyskano rownania z rysunku 15 co umożliwia wykorzystanie mniejszej liczby bramek logicznych.

4.2.2 NAND-DMUX:

Prezentacja implementacji rozwiązania:

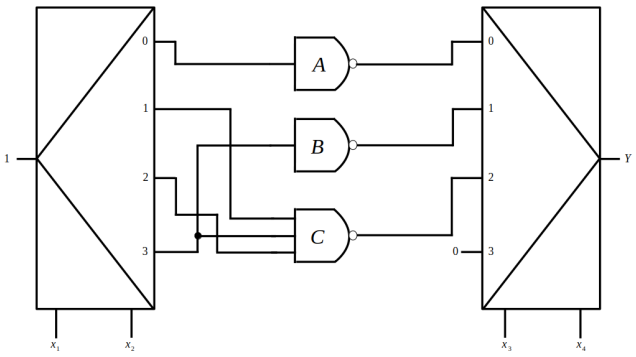


Rysunek 17: NAND-DMUX

Wnioski: Na podstawie danych z rysunku 13 i 14. Laczymy te wyjscia demulitpleksera z bramka NAND, które pojawiają się jako jedynki w tabeli.

4.2.3 MUX-NAND-DMUX:

Prezentacja implementacji rozwiązania:



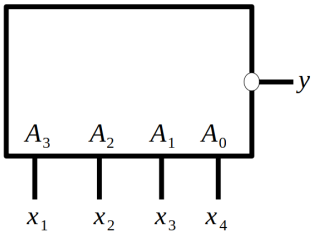
Rysunek 18: MUX-NAND-DMUX

Wnioski:

4.3 Dużej skali integracji LSI wykorzystując układ pamięci RAM:

4.3.1 Prezentacja implementacji rozwiązania:

Prezentacja implementacji rozwiązania:



Rysunek 19: Dużej skali integracji

Nr.	A3	A2	A1	A0	Q
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	0

Rysunek 20: Dużej skali integracji LSI wykorzystując układ pamięci RAM

4.3.2 Wnioski:

liczba wejść adresowych pamięci odpowiada liczbie zmiennych logicznych (jeżeli 4 zmienne x1x2x3x4 to należy uzupełnić 16 wartości funkcji logicznych dla każdego adresu) Q to wartość funkcji logicznej.