

Technika Cyfrowa

Ćwiczenie 3

Kacper Bieniasz
Aleksander Buszek
Maciej Nowakowski
Albert Pęciak

14.05.2024

1 Temat ćwiczenia

Ćwiczenie polegało na zaproponowaniu, zbudowaniu i przetestowaniu układu sterującego windą w przykładowym trzykondygnacyjnym budynku.

Winda powinna posiadać:

- wskaźnik ruchu windy
- wskaźnik kierunku ruchu windy
- trzy czujniki otwarcia drzwi, po jednym na każdej kondygnacji
- trzy przyciski przywołania windy, po jednym na każdej kondygnacji
- trzy przyciski wyboru piętra w kabinie windy.

Winda powinna posiadać również stale aktualizowany wskaźnik aktualnego piętra.

Rzeczy niedopowiedziane w treści zadania, należało ustalić, doprecyzować i opisać samodzielnie.

2 Projektowanie i implementacja układu

Układ sterujący windą na wejściu otrzymuje dziewięć stanów logicznych. Bity przyw_n przekazują wartość 1 jeśli ktoś z piętra n wezwał windę. Bity winda_n wskazują piętro wybrane przyciskami w środku windy. Bity klamka_n sygnalizują czy na piętrze n drzwi są otwarte.

Układ zwraca dwubitową liczbę nr_p_bit_n informującą o obecnym piętrze oraz trzy bity sygnalizujące kierunek i zwrot jazdy. Sygnał gora - winda jedzie w góre, sygnał dol - winda jedzie w dół, sygnał stoi - winda stoi nieruchomo.

Schemat automatu

Pierwszym krokiem było zaprojektowanie automatu sterującego windą. W tym celu musimy wyznaczyć i opisać wszystkie stany w jakich winda może się znajdować:

- **1s** - winda stoi na pierwszym piętrze z zamkniętymi drzwiami. Kod stanu: **0000**
- **1o** - winda stoi na pierwszym piętrze z otwartymi drzwiami. Kod stanu: **0001**
- **1g** - winda z pierwszego piętra ruszyła do góry. Kod stanu: **0010**
- **2g** - winda z drugiego piętra ruszyła w dół. Kod stanu: **0011**



- **2s** - winda stoi na drugim piętrze z zamkniętymi drzwiami. Kod stanu: **0100**
- **2o** - winda stoi na drugim piętrze z otwartymi drzwiami. Kod stanu: **0101**
- **2g** - winda z drugiego piętra ruszyła do góry. Kod stanu: **0110**
- **3s** - winda stoi na trzecim piętrze z zamkniętymi drzwiami. Kod stanu: **0111**
- **3o** - winda stoi na trzecim piętrze z otwartymi drzwiami. Kod stanu: **1000**
- **3d** - winda z trzeciego piętra ruszyła w dół. Kod stanu: **1001**

Znając stany potrzebujemy sygnałów wysyłanych przez użytkownika zmieniających stany. Sygnałem wejściowym będą 4 bity I3 , I2, I1 oraz I0. Trzy pierwsze bity oznaczają piętro na które winda została wezwana (Bit I3 - piętro trzecie, I2 - piętro drugie itd.). Bit I0 wskazuje czy, którykolwiek z drzwi są otwarte i nie pozwalają windzie ruszyć z miejsca.

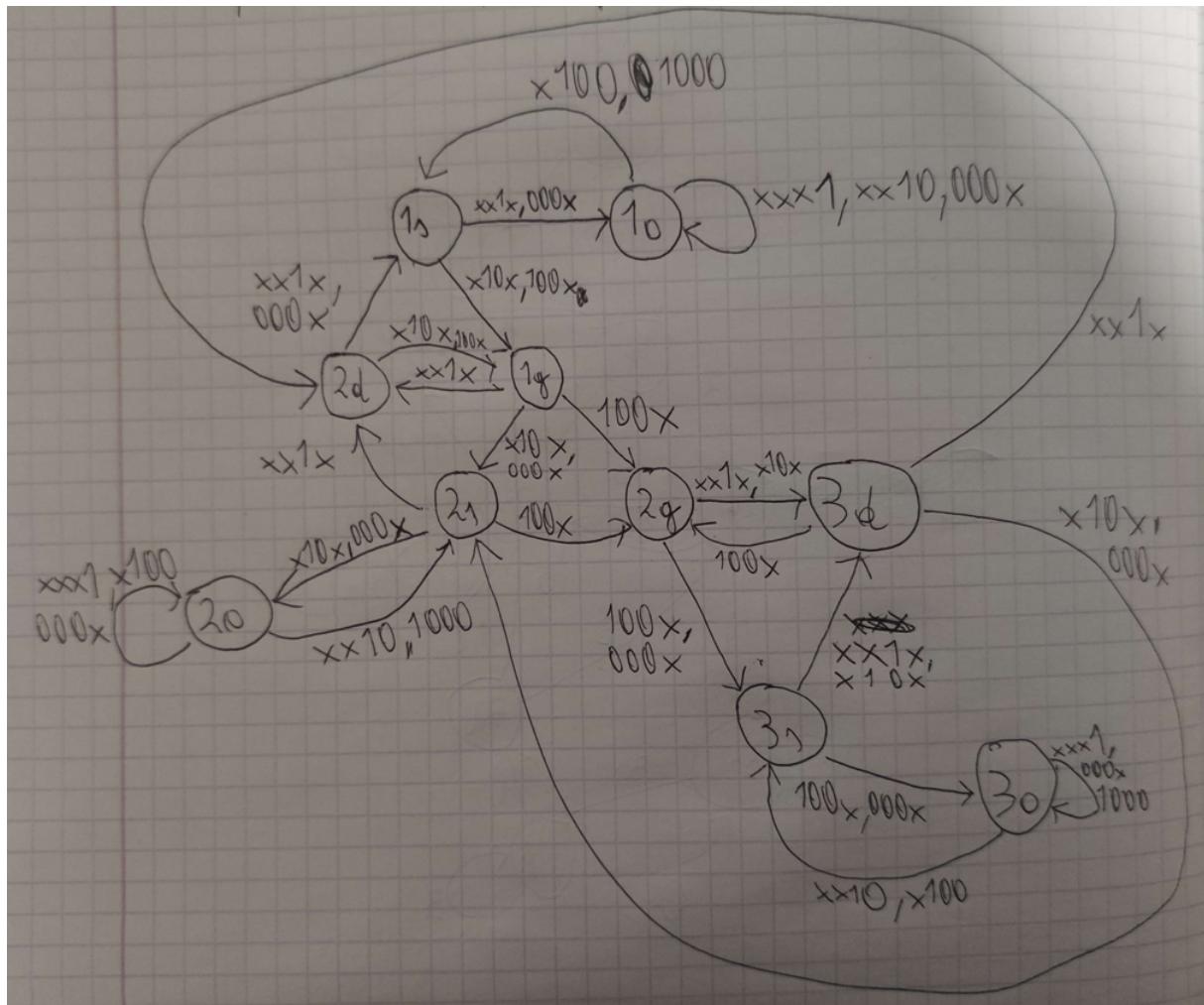
Dla każdego stanu należy określić reakcję na otrzymanie dowolnego inputu. To przyporządkowanie przedstawia poniższa tabela.

W poniższej tabeli znak x oznacza dowolną wartość bitu. Input 000x określa zarówno 0001 jak i 0000.

stan	input	nowy stan								
1s	xx1x	1o	x10x	1g	100x	1g	000x	1o	-	-
1o	xxx1	1o	xx10	1o	x100	1s	1000	1s	000x	1o
1g	xx1x	2d	x10x	2s	100x	2g	000x	2s	-	-
2d	xx1x	1s	x10x	1g	100x	1g	000x	1s	-	-
2s	xx1x	2d	x10x	2o	100x	2g	000x	2o	-	-
2o	xxx1	2o	xx10	2s	x100	2o	1000	2s	000x	2o
2g	xx1x	3d	x10x	3d	100x	3s	000x	3s	-	-
3s	xx1x	3d	x10x	3d	100x	3o	000	3o	-	-
3o	xxx1	3o	xx10	3s	x100	3s	1000	3o	000x	3o
3d	xx1x	2d	x10x	2s	100x	2g	000x	2s	-	-

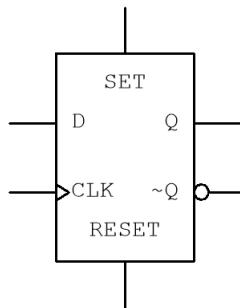
Rysunek 1: Tabela informująca jak input wpływa na stan windy

Schemat automatu bazujący na tabeli powyżej



Rysunek 2: Schemat automatu

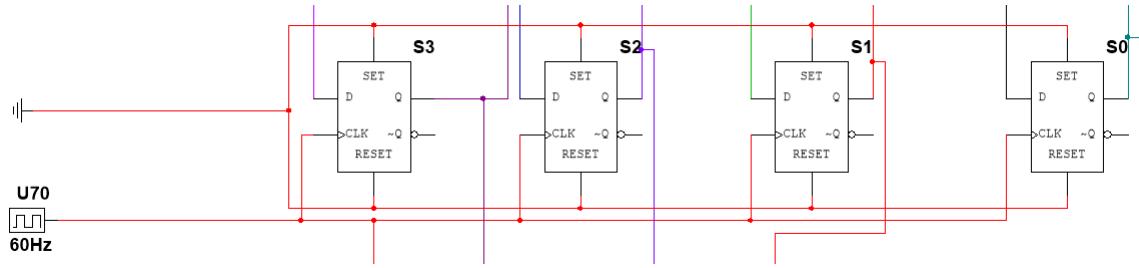
Do przechowywania aktualnego stanu windy wykorzystywany jest zestaw czterech przerzutników typu D.



D	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

Tabela 1: Tabela prawdy dla przerzutnika typu D

Rysunek 3: Przerzutnik typu T



Rysunek 4: Przerzutniki S_n przechowujące n -ty bit stanu windy

2.1 Zmieniacz

Układ zmieniacz implementuje aktualizację stanu na podstawie inputu przedstawioną w tabeli 1.



Do układu zmieniacz przekazywane jest osiem bitów informujące o stanie windy (bitы Q3, Q2, Q1, Q0 oznaczające kod stanu) oraz input użytkownika (bitы I3, I2, I1, I0). Układ zwraca kod następnego stanu windy (bitы D3, D2, D1, D0).

Tabela prawdy

Q3	Q2	Q1	Q0	I3	I2	I1	I0	D3	D2	D1	D0
stan				input				nowy stan			
0	0	0	0	x	x	1	x	0	0	0	1
0	0	0	0	x	1	0	x	0	0	1	0
0	0	0	0	1	0	0	x	0	0	1	0
0	0	0	0	0	0	0	x	0	0	0	1
0	0	0	1	x	x	x	1	0	0	0	1
0	0	0	1	x	x	1	0	0	0	0	1
0	0	0	1	x	1	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	x	0	0	0	1
0	0	1	0	x	x	1	x	0	0	1	1
0	0	1	0	x	1	0	x	0	1	0	0
0	0	1	0	1	0	0	x	0	1	1	0
0	0	1	0	0	0	0	x	0	1	0	0
0	0	1	1	x	x	1	x	0	0	0	0
0	0	1	1	x	1	0	x	0	0	1	0
0	0	1	1	1	0	0	x	0	0	1	0
0	0	1	1	0	0	0	x	0	0	0	0
0	1	0	0	x	x	1	x	0	0	1	1
0	1	0	0	x	1	0	x	0	1	0	1
0	1	0	0	1	0	0	x	0	1	1	0
0	1	0	0	0	0	0	x	0	1	0	1
0	1	0	1	x	x	x	1	0	1	0	1
0	1	0	1	x	x	1	0	0	1	0	0
0	1	0	1	x	1	0	0	0	1	0	1
0	1	0	1	1	0	0	0	0	1	0	0
0	1	0	1	0	0	0	x	0	1	0	1
0	1	1	0	x	x	1	x	1	0	0	1
0	1	1	0	x	1	0	x	1	0	0	1
0	1	1	0	1	0	0	x	0	1	1	1
0	1	1	0	0	0	0	x	0	1	1	1
0	1	1	1	x	x	1	x	1	0	0	1
0	1	1	1	x	1	0	x	1	0	0	1
0	1	1	1	1	0	0	x	1	0	0	0
0	1	1	1	0	0	0	x	1	0	0	0
1	0	0	0	x	x	x	1	1	0	0	0
1	0	0	0	x	x	1	0	0	1	1	1
1	0	0	0	x	1	0	0	0	1	1	1
1	0	0	0	1	0	0	0	1	0	0	0
1	0	0	0	0	0	0	x	1	0	0	0
1	0	0	1	x	x	1	x	0	0	1	1
1	0	0	1	x	1	0	x	0	1	0	0
1	0	0	1	1	0	0	x	0	1	1	0
1	0	0	1	0	0	0	x	0	1	0	0

Tabela 3: Tabela prawdy dla układu **Zmieniacz**

Stany oznaczone x tak jak na schemacie 2 oznaczają dowolny stan. Dodatkowo liczba stanów możliwych do zakodowania na 4 przerzutnikach wynosi 16, a my użyliśmy tylko 10. W związku z tym dla wszystkich pozostałych stanów nieokreślonych niezależnie od input ustaliśmy stan, do którego przechodzi nasz automat

jako 0000.

Zważywszy na liczbę bitów wejścia użytą w tabeli prawdy, do wyznaczania wzorów użyto zewnętrznego narzędzia do rozwiązywania map Karnaugh. <https://www.charlie-coleman.com/experiments/kmap/>

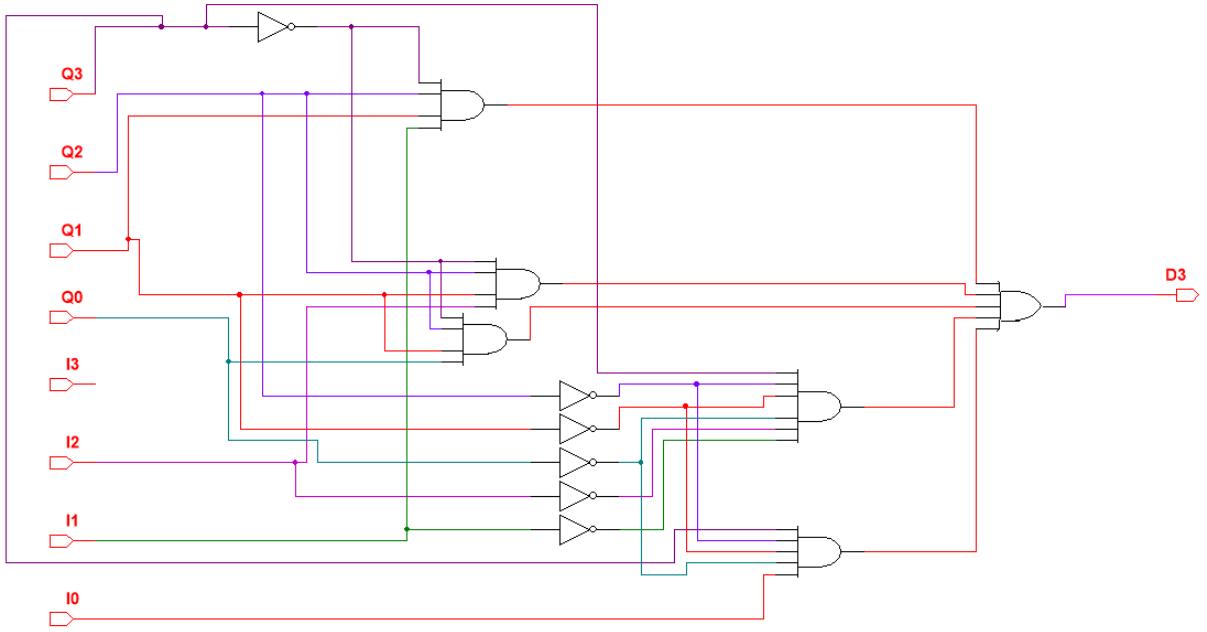
Bit D3

Wyznaczony wzór:

$$D3 = \overline{Q3} Q2 Q1 I1 + \overline{Q3} Q2 Q1 I2 + \overline{Q3} Q2 Q1 Q0 + Q3 \overline{Q2} \overline{Q1} \overline{Q0} I2 \overline{I1} + Q3 \overline{Q2} \overline{Q1} \overline{Q0} I0$$

<i>f</i>	<i>I3,I2,I1,I0</i>															
<i>Q3,Q2,Q1,Q0</i>	0000	0001	0011	0010	0110	0111	0101	0100	1100	1101	1111	1110	1010	1011	1001	1000
0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0011	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0010	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0110	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0111	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0101	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1101	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1111	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1110	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1010	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1011	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1000	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	1

Rysunek 5: Tabela Karnaugh dla bitu D3



Rysunek 6: Podukład wyznaczający bit D3

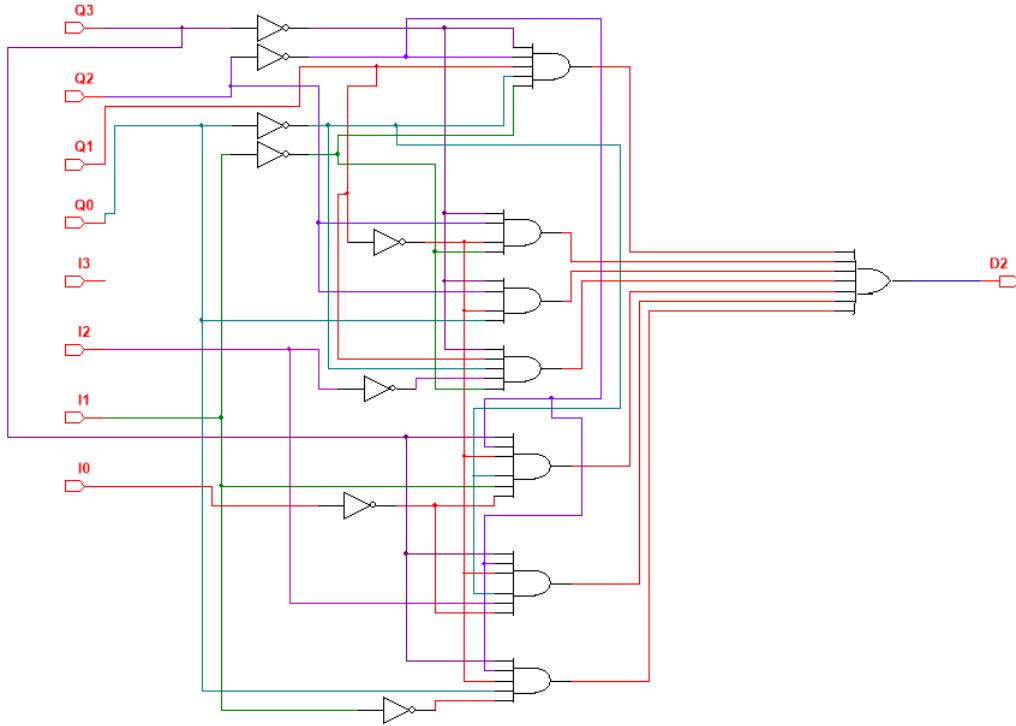
Bit D2

Wyznaczony wzór:

$$D2 = \overline{Q3} \overline{Q2} Q1 \overline{Q0} \overline{I1} + \overline{Q3} Q2 \overline{Q1} \overline{I1} + \overline{Q3} Q2 \overline{Q1} Q0 + \overline{Q3} Q1 \overline{Q0} \overline{I2} \overline{I1} + Q3 \overline{Q2} \overline{Q1} \overline{Q0} I1 \overline{I0} + \\ Q3 \overline{Q2} \overline{Q1} \overline{Q0} I2 \overline{I0} + Q3 \overline{Q2} \overline{Q1} Q0 \overline{I1}$$

f	I_3, I_2, I_1, I_0
Q_3, Q_2, Q_1, Q_0	0000 0001 0011 0010 0110 0111 0101 0100 1100 1101 1111 1110 1010 1011 1001 1000
0000	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0001	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0011	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0010	1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1
0110	1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1
0111	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0101	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0100	1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1
1100	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1101	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1111	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1110	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1010	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1011	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1001	1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1
1000	0 0 0 1 1 0 0 1 1 0 0 1 1 0 0 0

Rysunek 7: Tabela Karnaugh dla bitu D2



Rysunek 8: Podukład wyznaczający bit D2

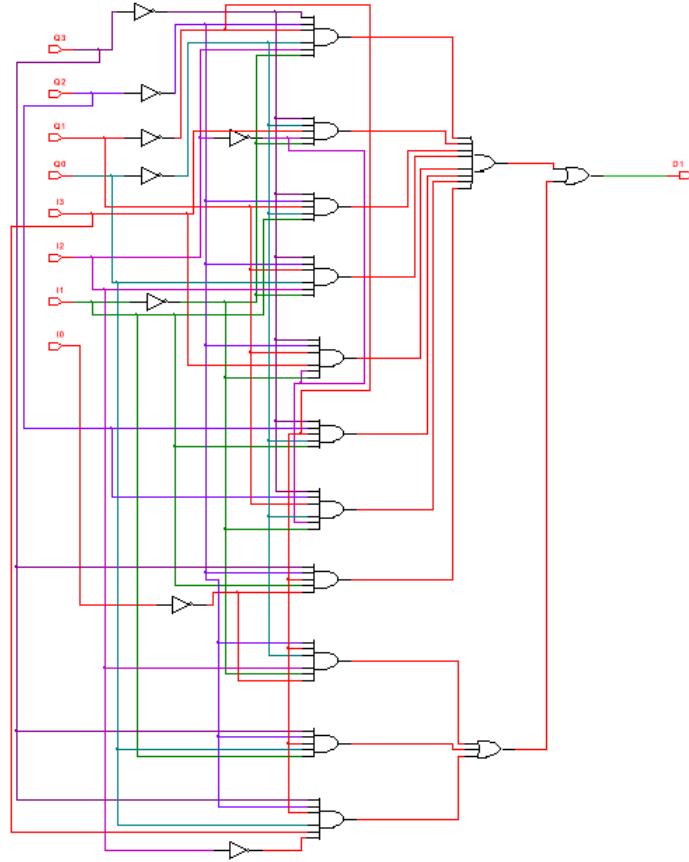
Bit D1

Wyznaczony wzór:

$$D1 = \overline{Q3} \overline{Q2} \overline{Q1} \overline{Q0} I2 \overline{I1} + \overline{Q3} \overline{Q0} I3 \overline{I2} \overline{I1} + \overline{Q3} \overline{Q2} Q1 \overline{Q0} I1 + \overline{Q3} \overline{Q2} Q1 Q0 I2 \overline{I1} + \overline{Q3} \overline{Q2} Q1 I3 \overline{I2} \overline{I1} \\ + \overline{Q3} Q2 \overline{Q1} \overline{Q0} I1 + \overline{Q3} Q2 Q1 \overline{Q0} \overline{I2} \overline{I1} + Q3 \overline{Q2} \overline{Q1} I1 \overline{I0} + \overline{Q2} \overline{Q1} \overline{Q0} I2 \overline{I1} \overline{I0} + Q3 \overline{Q2} \overline{Q1} Q0 I1 + Q3 \overline{Q2} \overline{Q1} Q0 I3 \overline{I2}$$

f	$I3, I2, I1, I0$
$Q3, Q2, Q1, Q0$	0000 0001 0011 0010 0110 0111 0101 0100 1100 1101 1111 1110 1010 1011 1001 1000
0000	0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1
0001	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0011	0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1
0010	0 0 1 1 1 1 - 0 0 0 1 1 1 1 1 1
0110	1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
0111	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0101	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0100	0 0 1 1 1 1 0 0 0 1 1 1 1 1 1 1
1100	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1101	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1111	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1110	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1010	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1011	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1001	0 0 1 1 1 1 0 0 0 1 1 1 1 1 1 1
1000	0 0 0 0 1 1 0 0 1 1 0 0 1 1 0 0

Rysunek 9: Tabela Karnaugh dla bitu D1



Rysunek 10: Podukład wyznaczający bit D1

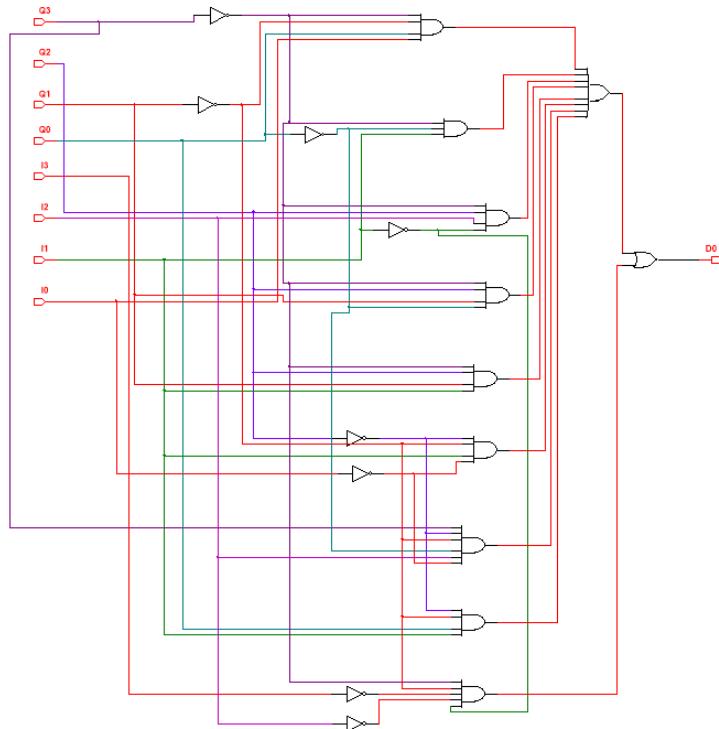
Bit D0

Wyznaczony wzór:

$$\begin{aligned}
 D0 = & \overline{Q3} \overline{Q1} Q0 I0 + \overline{Q3} \overline{Q0} I1 + \overline{Q3} Q2 I2 \overline{I1} + \overline{Q3} Q2 Q1 \overline{Q0} + \overline{Q3} Q2 Q1 I1 + \overline{Q2} \overline{Q1} I1 \overline{I0} + Q3 \overline{Q2} \overline{Q1} \overline{Q0} I2 \overline{I0} \\
 & + \overline{Q2} \overline{Q1} Q0 I1 + \overline{Q3} \overline{Q1} \overline{I3} \overline{I2} \overline{I1}
 \end{aligned}$$

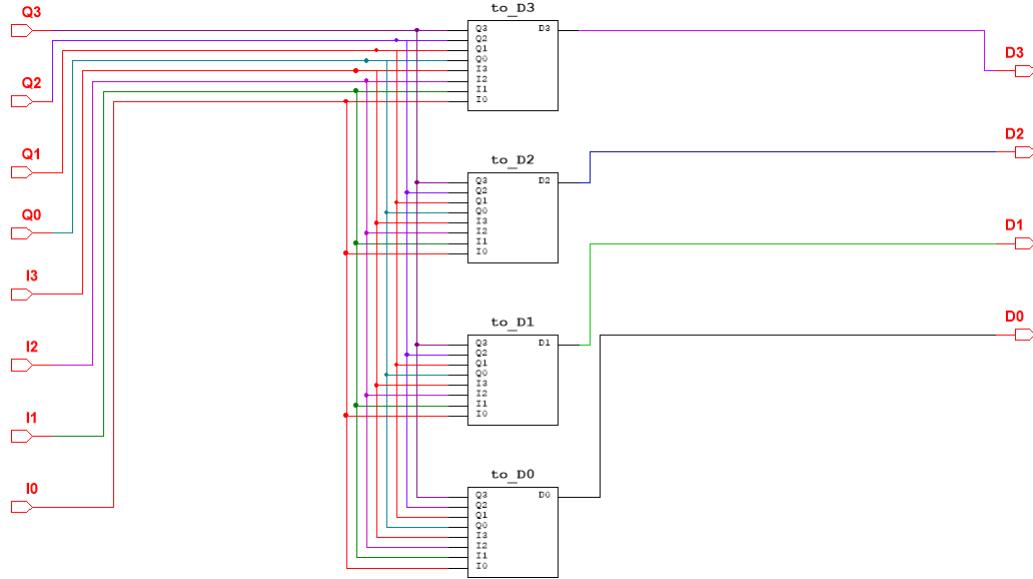
f	$13,12,11,10$
Q_3, Q_2, Q_1, Q_0	$0000 \quad 0001 \quad 0011 \quad 0010 \quad 0110 \quad 0111 \quad 0101 \quad 0100 \quad 1100 \quad 1101 \quad 1111 \quad 1110 \quad 1010 \quad 1011 \quad 1001 \quad 1000$
0000	1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0
0001	1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0
0011	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0010	0 0 1 1 1 1 0 0 0 0 1 1 1 1 1 0
0110	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0111	0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0
0101	1 1 1 0 0 1 1 1 1 1 0 0 1 1 0 0
0100	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
1100	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1101	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1111	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1110	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1010	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1011	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1001	0 0 1 1 1 1 0 0 0 0 1 1 1 1 1 0
1000	0 0 0 1 1 0 0 1 1 0 0 1 1 0 0 0

Rysunek 11: Tabela Karnaugh dla bitu D0



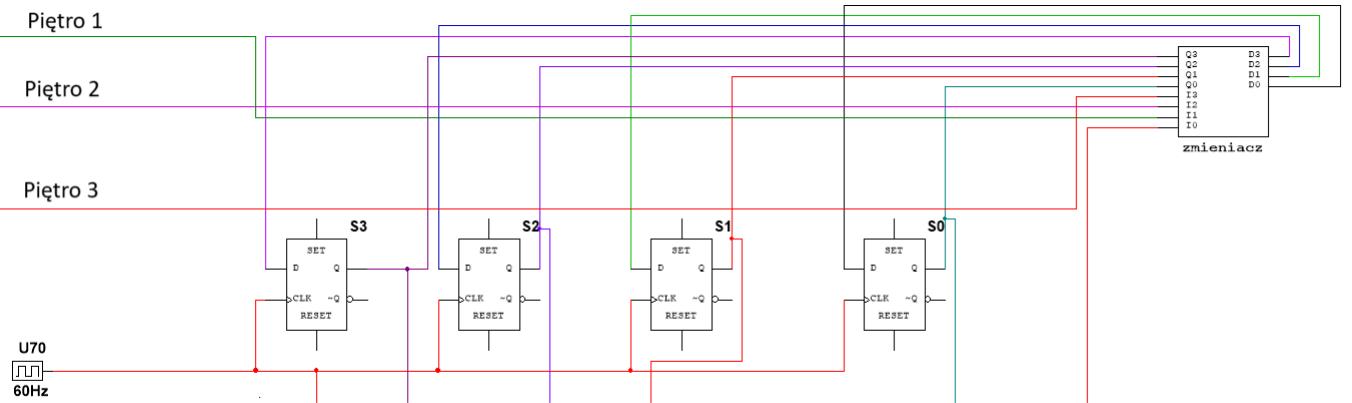
Rysunek 12: Podkład wyznaczający bit D0

Cały układ



Rysunek 13: Implementacja całego podukładu zmieniacz

Układ Zmieniacz połączony z przerzutnikami



Rysunek 14: Podpięcie układu zmieniacz pod przerzutniki przetrzymujące stan

Bit sygnalizujący piętro, do którego wzywana jest winda pobierane są z inputu układu **sterowanie_windy**. Bit Piętro n to połączone bramką OR inputy `pryw_n` oraz `winda_n`. Przerzutniki przechowują stan w postaci czterobitowego kodu. Potrzebny jest nam układ, który bezpośrednio wskaże nam piętro na którym winda się znajduje, poinformuje o możliwości otworzenia drzwi oraz powiadomi czy i w jakim kierunku winda się porusza.

2.2 Outputer

Przed zaprojektowaniem podukładu trzeba określić liczbę bitów wyjściowych oraz co będą one oznaczały.

Układ będzie zwracał 5 bitów:

- Bit N1 oraz N0 - informacja o aktualnym piętrze na którym znajduje się winda w postaci dwubitowej liczby. 01 to piętro pierwsze, 10 piętro drugie itd..

- Bit O - informacja o tym czy drzwi można otworzyć (1 - można, 0 - nie można).
- Bity R1 i R0 - kierunek ruchu windy. 10 - winda idzie w górę, 01 - winda idzie w dół, 00 - winda się nie porusza.

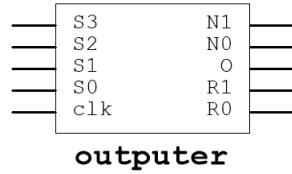


Tabela prawdy

Automat składa się z 10 stanów. Do wyznaczenia pięciu bitów wyjściowych należy wziąć pod uwagę liczby, które nie reprezentują żadnego stanu.

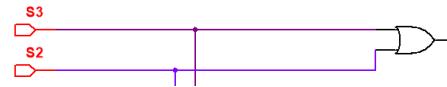
S3	S2	S1	S0	N1	N0	O	R1	R0
stan				output				
0	0	0	0	0	1	0	0	0
0	0	0	1	0	1	1	0	0
0	0	1	0	0	1	0	0	1
0	0	1	1	0	1	0	1	0
0	1	0	0	1	0	0	0	0
0	1	0	1	1	0	1	0	0
0	1	1	0	1	0	0	0	1
0	1	1	1	1	1	0	0	0
1	0	0	0	1	1	1	0	0
1	0	0	1	1	1	0	1	0

Tabela 4: Tabela prawdy dla układu **Outputer**

Stanom 1010, 1011, 1100, 1101, 1110 i 1111 nie zostały przypisane żadne stany więc nie mają wpływu na oczekiwany output. Możemy zatem dobrać wartości bitów outputu wedle uznania.

Bit N1

		$S_1 S_0$			
		00	01	11	10
00		0	0	0	0
S ₃ S ₂	01	1	1	1	1
	11	-	-	-	-
	10	1	1	-	-



Rysunek 15: Tworzenie sygnału dla przerzutnika T4

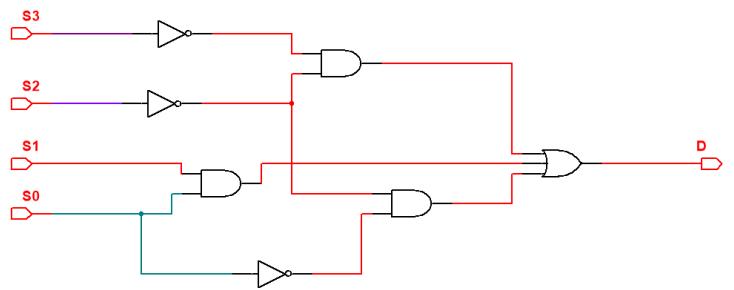
Z czerwonego otrzymujemy: S_2

Z zielonego otrzymujemy: S_3

Wynik: $S_2 + S_3$

Bit N0

		$S_1 S_0$			
		00	01	11	10
00		1	1	1	1
S ₃ S ₂	01	0	0	1	0
	11	-	-	-	-
	10	1	1	-	-



Rysunek 16: Tworzenie sygnału dla przerzutnika T3

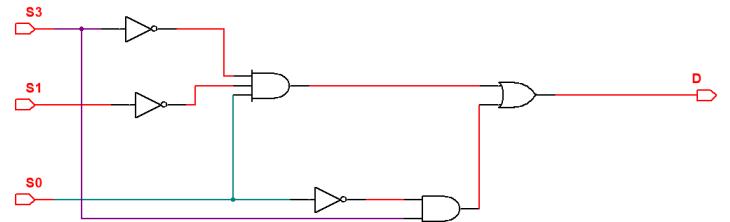
Z czerwonego otrzymujemy: $S_1 S_0$

Z zielonego otrzymujemy: $\overline{S_2}$

Wynik: $\overline{S_2} + S_1 S_0$

Bit O

		S1S0			
		00	01	11	10
S3S2	00	0	1	0	0
	01	0	1	0	0
	11	-	-	-	-
	10	1	0	-	-



Rysunek 17: Tworzenie sygnału dla przerzutnika T2

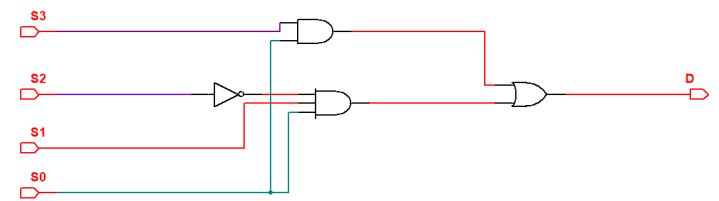
Z czerwonego otrzymujemy: $\overline{S3} \overline{S1} S0$

Z zielonego otrzymujemy: $S3 \overline{S0}$

Wynik: $S2 S1 S0 + S3 \overline{S0}$

Bit R1

		S1S0			
		00	01	11	10
S3S2	00	0	0	1	0
	01	0	0	0	0
	11	-	-	-	-
	10	0	1	-	-



Rysunek 18: Tworzenie sygnału dla przerzutnika T1

Z zielonego otrzymujemy: $\overline{S2} S1 S0$

Z czerwonego otrzymujemy: $S3 S0$

Wynik po uproszczeniu: $S2 S1 S0 + S3 S0$

Bit R0

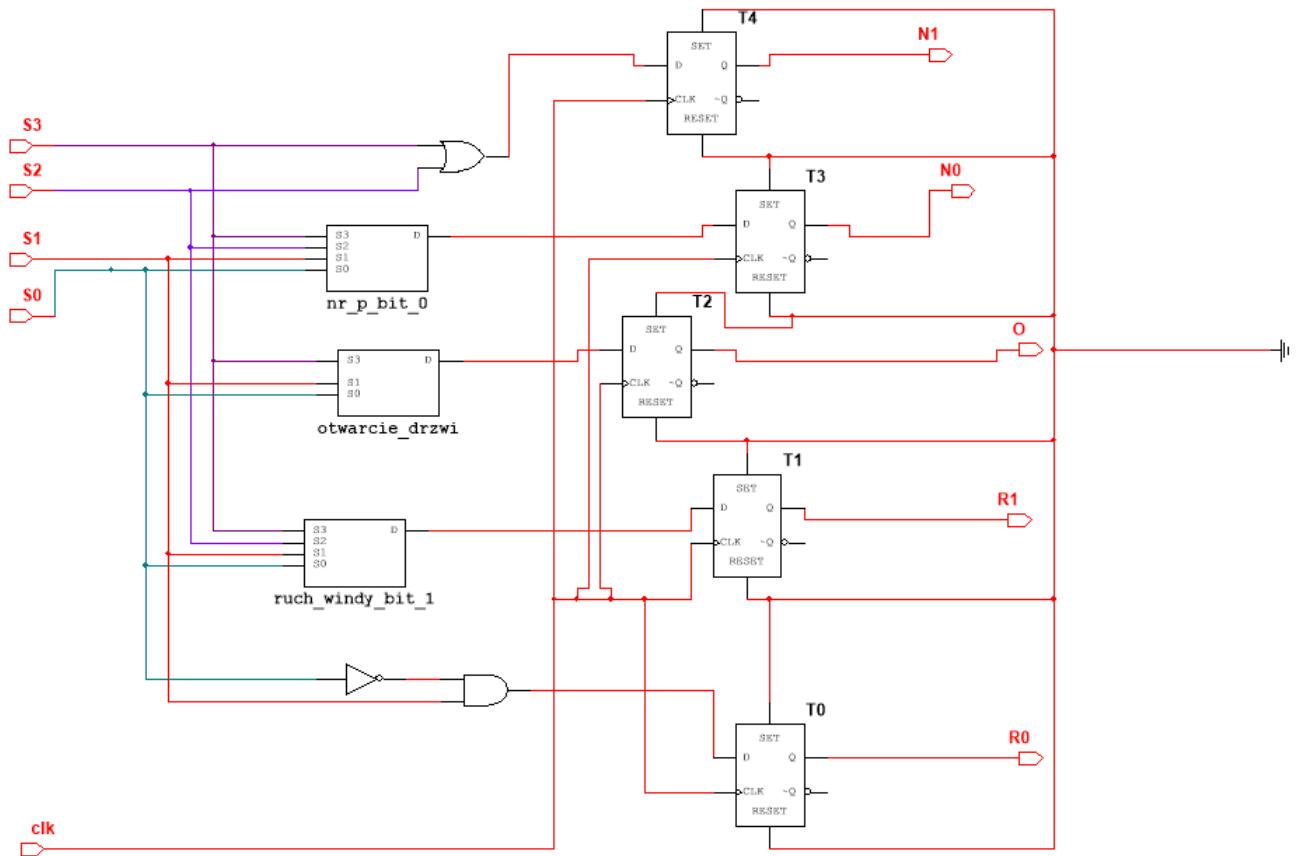
				<i>S1S0</i>
				00 01 11 10
<i>S3S2</i>	00	0 0 0	1	
	01	0 0 0	1	
	11	- - -	-	
	10	0 0 -	-	



Rysunek 19: Tworzenie sygnału dla przerzutnika T0

Wynik: $S1 \bar{S0}$

Układ Outputer wykorzystuje przerzutniki do ciągłego nadawania sygnału tak by po zmianie stanu windy (i ticku zegara) zmienił się output.

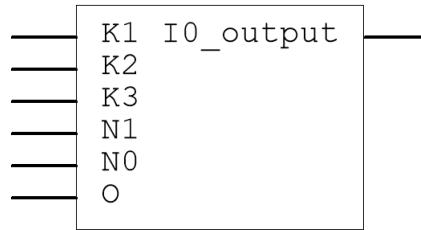


Rysunek 20: Cały układ outputer

Kontroler drzwi

Układ kontroler_drzwi na podstawie informacji czy można otworzyć drzwi, na którym piętrze stoi winda i która klamka została pociągnięta przekazuje do układu zmieniacz informację o tym, czy drzwi zostały otwarte.

kontroler_drzwi



Rysunek 21: Czarna skrzynka kontrolera drzwi

Tabele prawdy

K3	N1	N0	INF3
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Wynik : $K3 \ N1 \ N0$

Tabela 5: Odczytywanie informacji z 3 piętra

K2	N1	N0	INF2
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

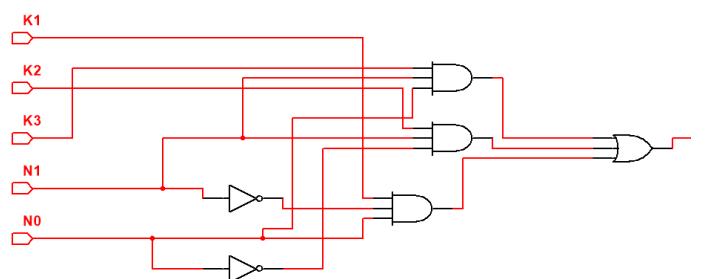
Wynik : $K2 \ N1 \ \overline{N0}$

Tabela 6: Odczytywanie informacji z 2 piętra

K1	N1	N0	INF1
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

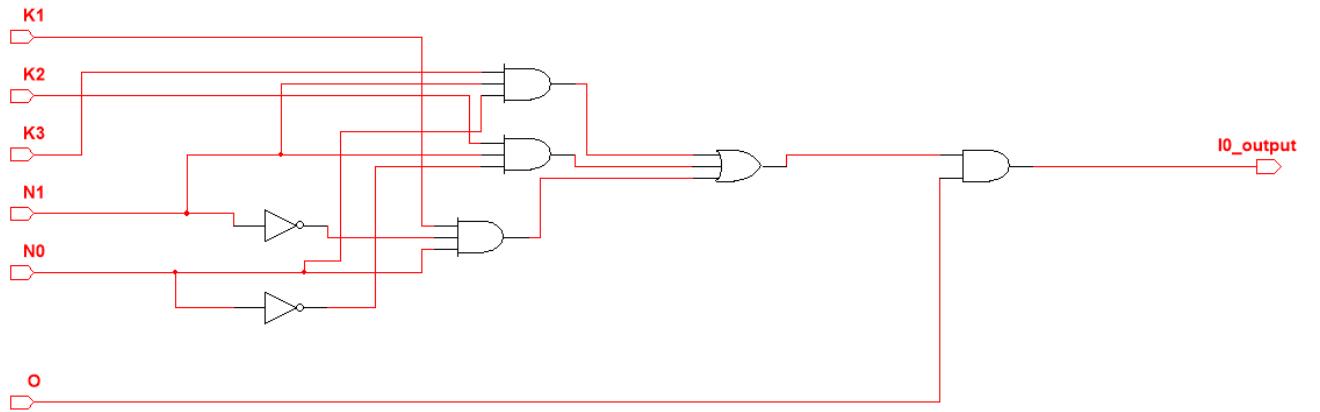
Wynik : $K1 \ \overline{N1} \ N0$

Tabela 7: Odczytywanie informacji z 1 piętra



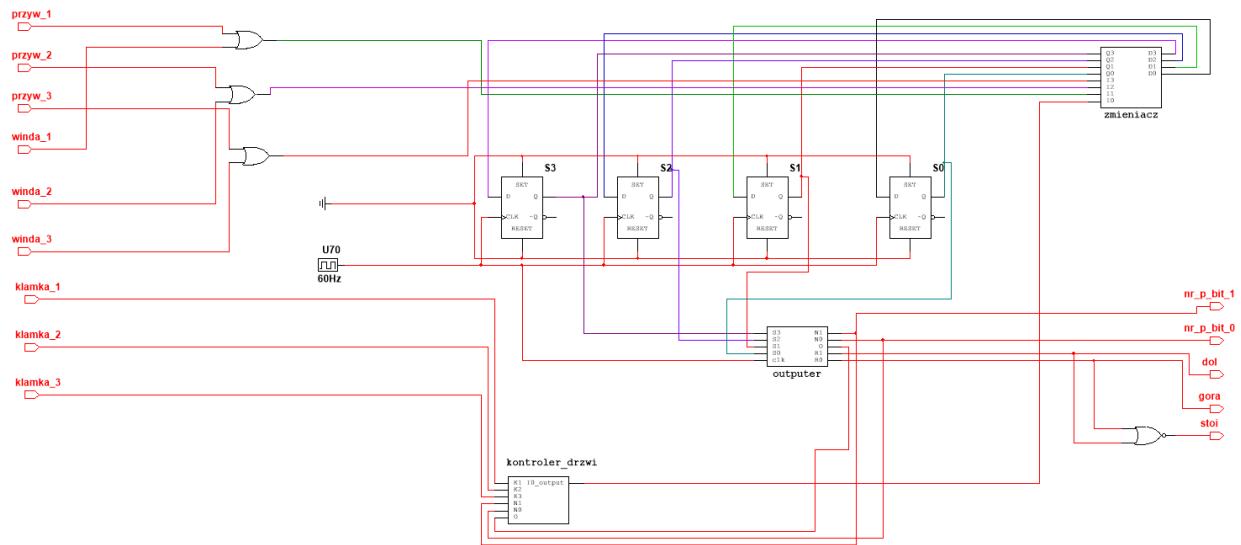
Rysunek 22: Informacje o stanie drzwi z trzech pięter uogólnione bramką OR

Posiadając informację o stanie drzwi z każdego piętra oraz stan windy możemy określić rzeczywistą wartość bitu wejściowego dla układu **zmieniacz**.

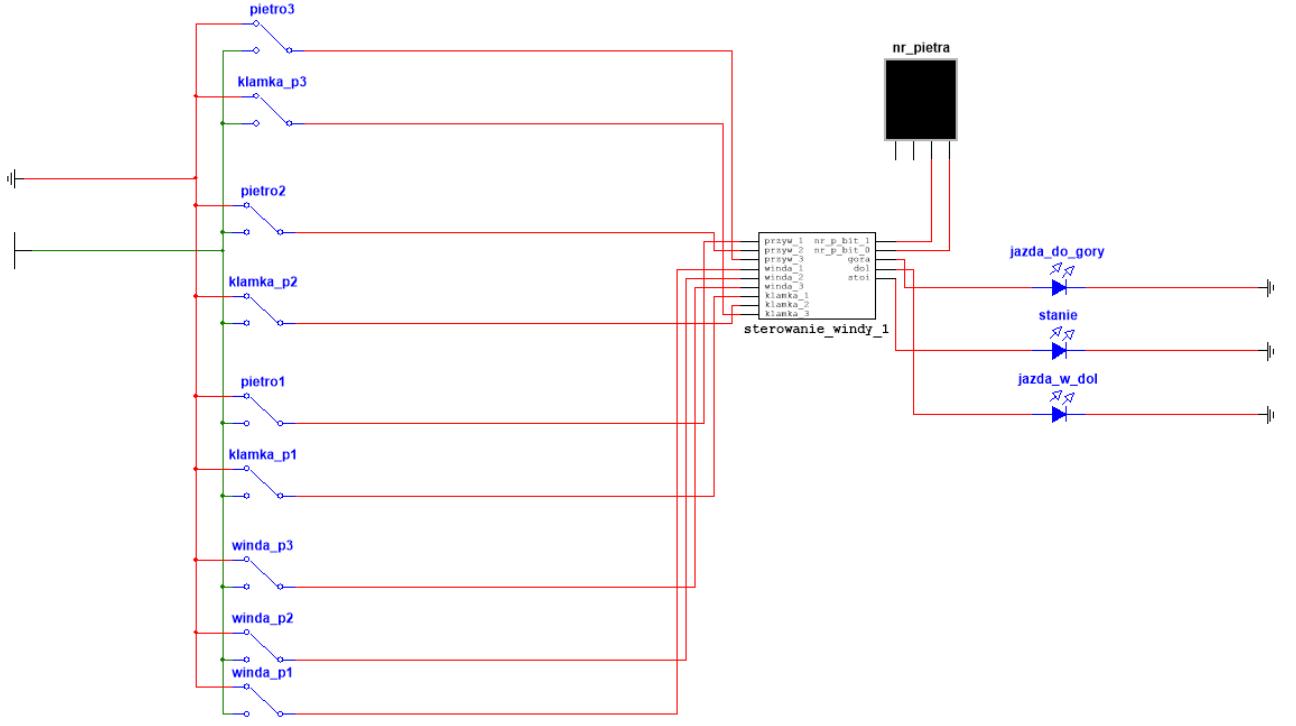


Rysunek 23: Cały układ kontroler_drzwi

Układ sterowania windy w całości



Rysunek 24: Układ sterowania windą



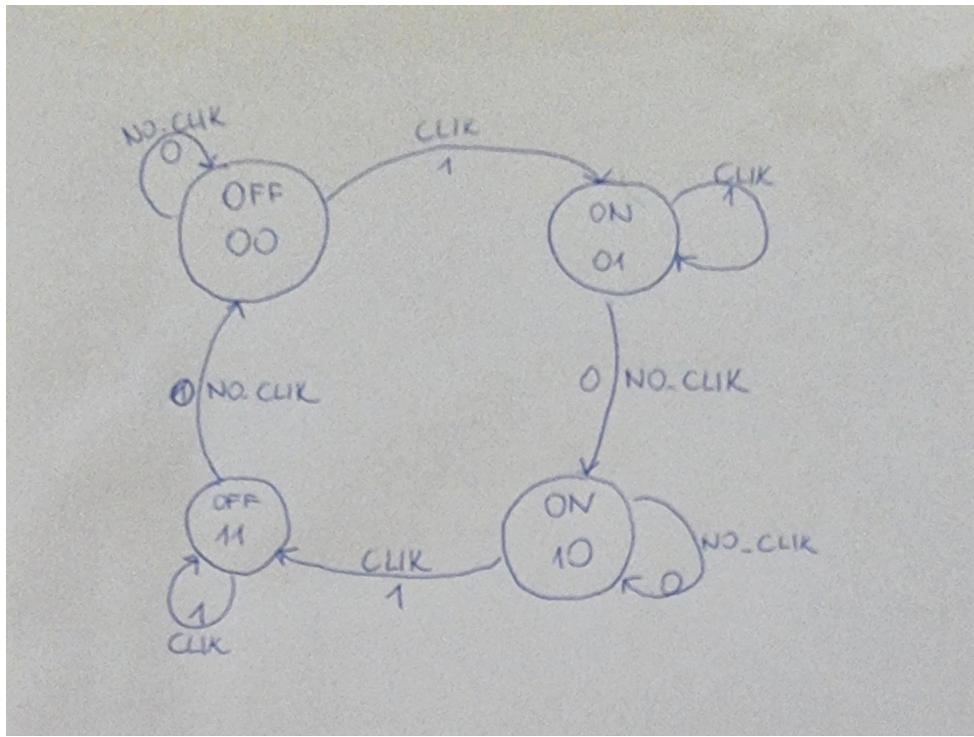
Rysunek 25: Implementacja windy w multisim

3 Testowanie układu

Z racji, że nie rozbiliśmy układu na mniejsze podzespoły, powstały automat okazał się bardzo ciężki do automatycznego przetestowania. Zbudowaliśmy prosty automat, który jesteśmy w stanie przetestować i w analogiczny sposób testowalibyśmy nasz główny automat obsługujący pracę windy.

3.1 Automat testowany

W tym celu zbudowaliśmy prosty automat omówiony na wykładzie, który obsługuje działanie lampki. Jeśli wciskamy przycisk, to lampka powinna się zaświecić lub zgasić, jednak w przedstawionym przez nas automacie zapobiegamy sytuacji, gdy wciskamy dłużej przycisk i lampka nieustannie zmienia swój stan ze świecącego na nieświecący i odwrotnie. Schemat automatu prezentuje się następująco:



Rysunek 26: Schemat działania automatu

Zbudowany automat Moore'a posiada 4 stany:

- **OFF 00** - lampka nie świeci - jest to stan wyłączenia lampki. Jeśli nie wciskamy przycisku, to lampka pozostaje w obecnym stanie. Gdy naciskamy przycisk przechodzimy do stanu **ON 01**, a lampka się zaświeca.
- **ON 01** - lampka świeci - jest to stan, który zapewnia, że po wciśnięciu przycisku lampka nie będzie się nieustannie włączala i wyłączala, a zostanie włączona. Gdy po włączeniu lampki przestaniemy wciskać przycisk przechodzimy do stanu **ON 10**.
- **ON 10** - lampka świeci - jest to stan, gdy po włączeniu lampki puściliśmy przycisk. Dopóki ponownie nie wciśniemy przycisku pozostajemy w tym stanie, natomiast po jego wciśnięciu przechodzimy do stanu **OFF 11**.
- **OFF 11** - lampka nie świeci - jest to drugi stan, który zapobiega mruganiom lampki. Dopóki przycisk jest wciśnięty pozostajemy w obecnym stanie, natomiast po jego puszczeniu, przechodzimy do stanu **OFF 00**.

Zastosowanie 4 stanów zamiast 2 gwarantuje, że nasza lampka nie będzie mrugała, a w sposób deterministyczny wybiera czy ma świecić, czy nie.

3.2 Implementacja układu testowanego

Zacznijmy od stworzenia tabeli funkcji wyjścia. Gdy **LIGHT** przyjmuje wartość 1 oznacza, że lampka się świeci. Wartości dla poniżej tabeli wywnioskowałem z rysunku układu powyżej.

		Q_1
	0	1
Q_0	0	1
1	1	0

Tabela 8: Tabela prawdy dla funkcji wyjścia

Z czerwonego otrzymujemy: $Q_1 \overline{Q_0}$

Z zielonego otrzymujemy: $Q_0 \overline{Q_1}$

Ostateczny wynik: $Q_1 \overline{Q_0} + Q_0 \overline{Q_1} = Q_1 \vee Q_0$

Można zauważyć, że wynik z tablicy Karnough dla Q_0, Q_1 to alternatywa rozłączna dla tych bitów więc przekształcamy wynik do postaci XOR.

Następnie konstruujemy tabelę, dla funkcji przejścia. Mówi ona, jak zmieni się stan wewnętrzny w zależności od impulsu X czyli clik, jak pokazano na schemacie automatu.

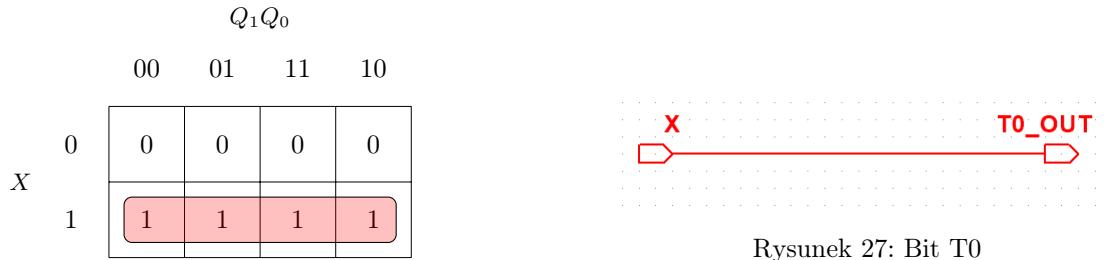
$Q_1 Q_0$	$X = 0$	$X = 1$
00	00	01
01	10	01
11	00	11
10	00	11

Tabela 9: Tabela prawdy dla funkcji przejścia

Mając rozpisaną funkcję przejścia konstruujemy tablice Karnough dla bitów. Bit T_0 to bit odpowiadający wartościami po prawej stronie w każdej z 2 kolumn zależnych od X natomiast bit T_1 to po prawej stronie.

3.3 Bit T_0

Bit T0

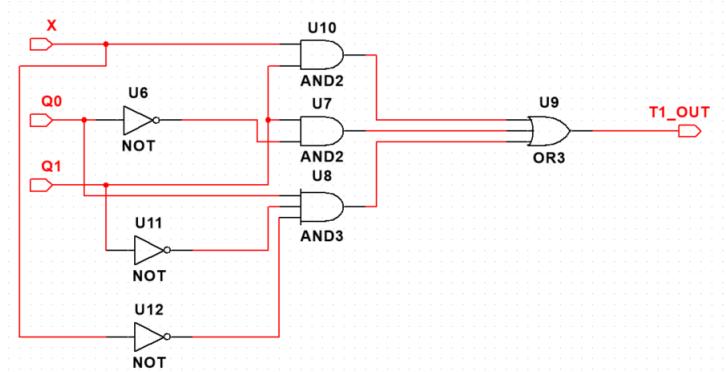


Rysunek 27: Bit T0

Ostateczny wynik: X

3.4 Bit T_1

		$Q_1 Q_0$	
		00	01
X	0	0	1
	1	0	1



Z czerwonego otrzymujemy: $Q_1 X$

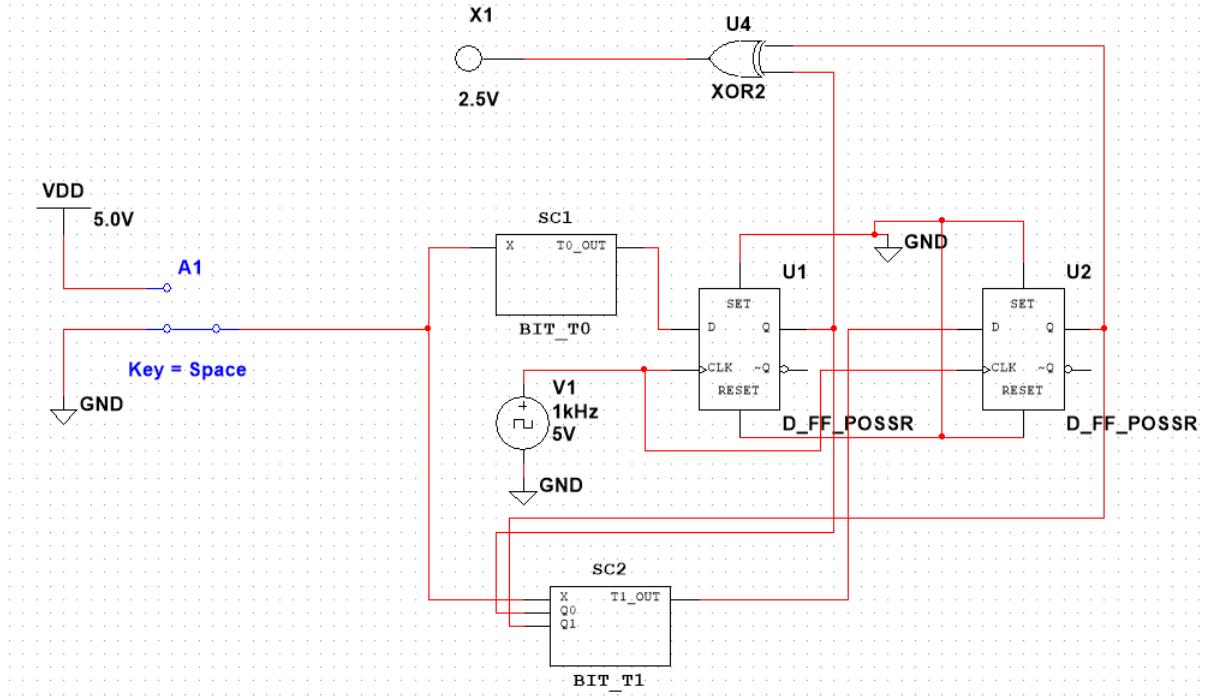
Z zielonego otrzymujemy: $Q_1 \bar{Q}_0$

Z żółtego otrzymujemy: $\bar{Q}_0 \bar{Q}_1 X$

Ostateczny wynik: $Q_1 X + Q_1 \bar{Q}_0 + \bar{Q}_0 \bar{Q}_1 X$

Rysunek 28: Bit T_1

Układ obsługujący lampkę w całości:

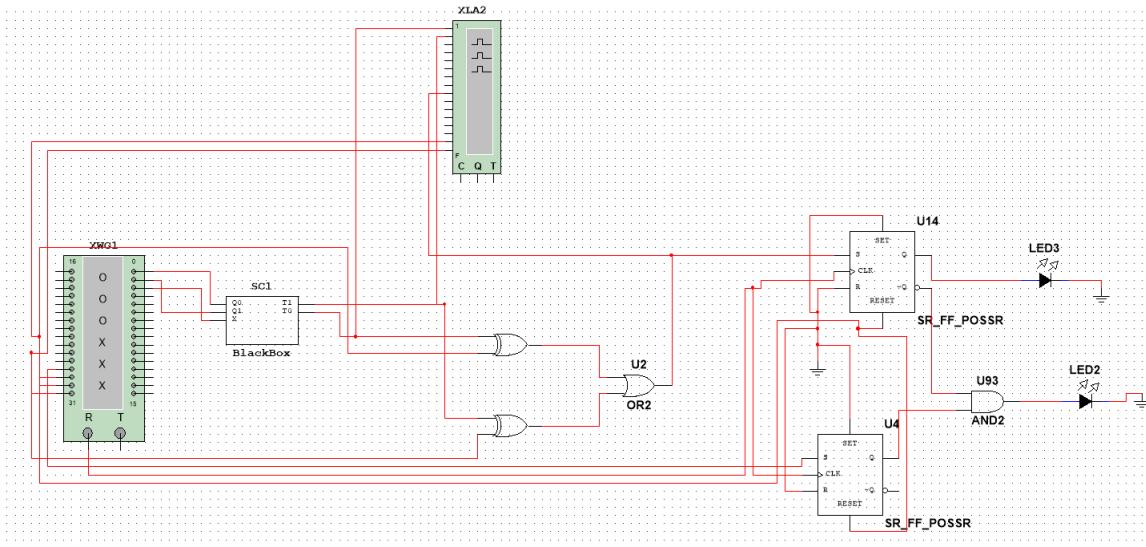


Rysunek 29: Schemat automatu w całości

Po omówieniu układu testowanego możemy przejść do właściwej części testowania.

3.5 Układ testujący

Testowanie będzie polegało na tym, że generator słów będzie generował każdy stan, w którym może być automat i dla każdego stanu rozważamy 2 przypadki- przycisk wciśnięty(1) oraz przycisk niewciśnięty(0). Generujemy również oczekiwany wynik, po przejęciu sygnału wciśnięcia bądź nie w danym stanie i porównujemy go ze stanem, zwróconym przez układ bramek logicznych dla bitów T_0 oraz T_1 .

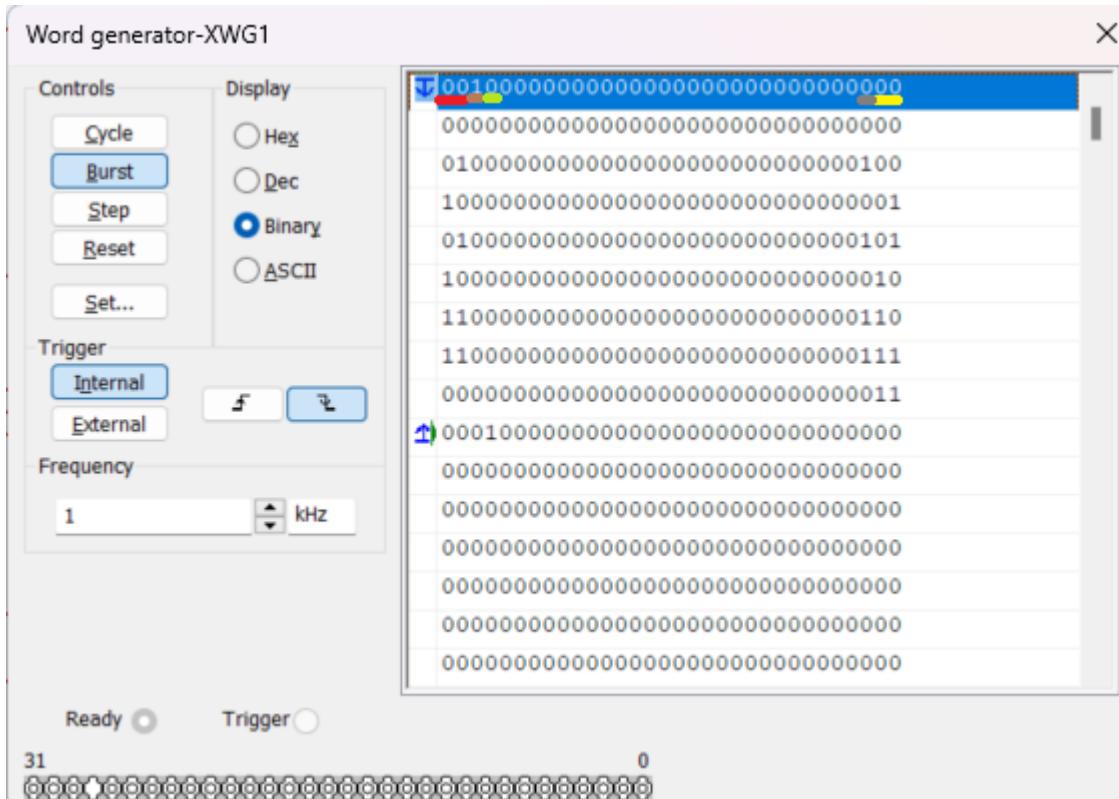


Rysunek 30: Schemat układu testującego w całości

W podukładzie nazwanym BlackBox znajdują się bramki logiczne dla bitów T_0 oraz T_1 opisane w podrozdziale powyżej. Opis poszczególnych komponentów znajduje się poniżej.

3.5.1 Generator słów

Generator słów wysyła 10 sygnałów - sygnał rozpoczęcia testowanie potrzebny do zresetowania przerzutników, 8 sygnałów zawierających informacje o stanie wciśnięcia przycisku oraz stanie automatu, oraz ostatni sygnał zakończenia symulacji.

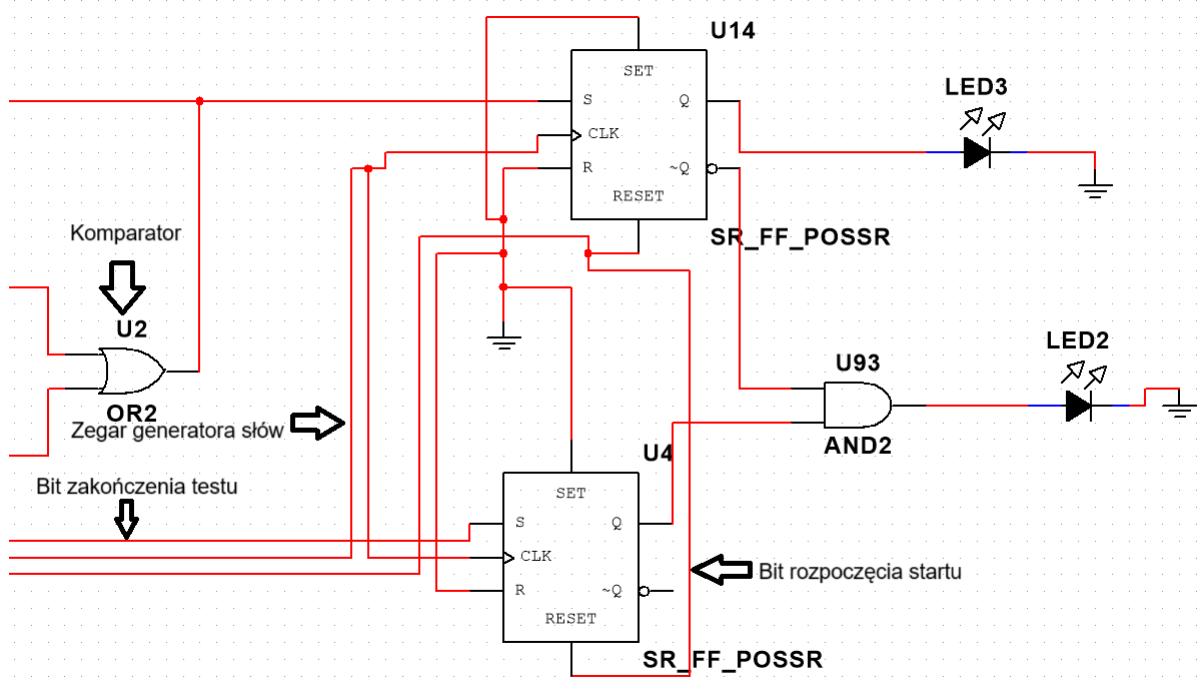


Rysunek 31: Ustawienia bitów w generatorze słów

- **Wejściowy stan automatu** - 2 bity od prawej to bity opisujące wewnętrzny stan automatu przed zmianą stanu.
- **Stan przycisku** - trzeci bit od prawej strony informujący czy przycisk jest wcisnięty(1) czy nie(2).
- **Bit zakończenia testowania** - czwarty bit od lewej strony informujący, kiedy kończymy testowanie.
- **Bit rozpoczęcia testowania** - trzeci bit od lewej strony informujący kiedy rozpoczynamy testowanie.
- **wyjściowy stan automatu** - 2 pierwsze bity od lewej strony informujące w jakim stanie oczekujemy na stan wewnętrzny automatu po przejściu przez bramki logiczne.

3.5.2 Układ wyświetlający wyniki testu

Komparator porównuje stan oczekiwany z rzeczywistym stanem automatu i wysyła sygnał do górnego przerzutnika. Jeśli przerzutnik otrzyma informację o błędny wyniku testu aktualizuje sygnał wysyłany z Q (i \bar{Q}) co zapala górną diodę. W przeciwnym wypadku, po poprawnym zmienieniu liczb dolny przeprzutnik otrzymuje informację o zakończonym teście i zapala diodę dolną.



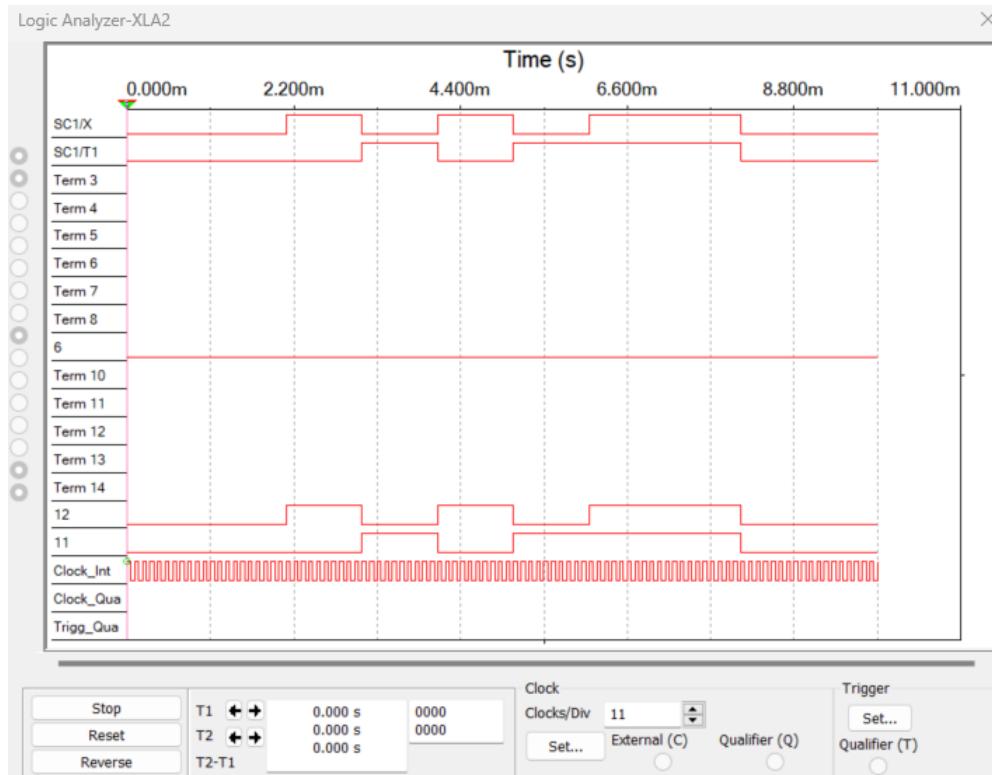
Rysunek 32: Układ odpowiedzialny za wyświetlanie wyników

- **górnego przerzutnika** w momencie otrzymania przeczącego sygnału od komparatora zapala LED3; jeśli nie ma błędu przesyła sygnał do bramki AND2
- **dolny przerzutnik** w momencie zakończenia testu wysyła do AND2 odpowiedni sygnał

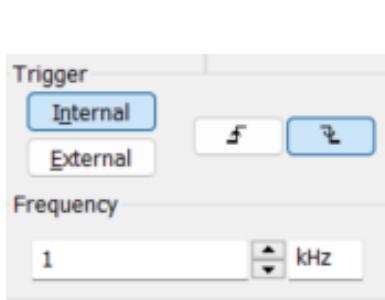
Gdy układ nie wykryje błędu oraz popłynie sygnał z generatora słów o zakończeniu testowania, LED2 zapali się na niebiesko sygnalizując pozytywne zakończenie testowania układu.

3.5.3 Analizator stanów logicznych

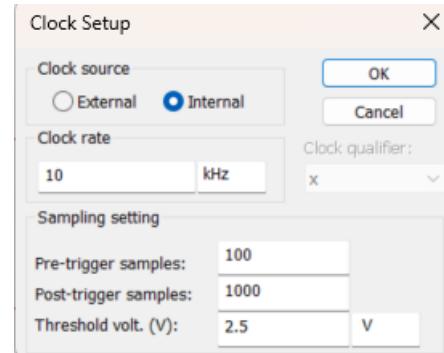
Do analizatora stanów logicznych wpięte jest 5 bitów: 2 górne to stan automatu po przejściu przez bramki logiczne, środkowy to wynik komparatora, gdy wykryje błąd to zauważymy wzorzystą spadek na wartości tego bitu, natomiast 2 dolne to oczekiwany stan automatu.



Rysunek 33: Wskazania analizatora stanów logicznych po pozytywnym teście



Rysunek 34: Ustawienia generatora słów



Rysunek 35: Ustawienia analizatora stanów logicznych

Przy testowaniu układu pamiętamy o twierdzeniu Nyquista. Częstotliwość generatora słów to 1kHz, częstotliwość próbkowania analizatora wynosi 10kHz, co jest zgodne z owym twierdzeniem.

3.6 Testowanie układu windy

Z racji, że nie rozobiliśmy projektu na mniejsze współpracujące ze sobą automaty testowanie całego układu byłoby bardzo ciękie w zaimplementowaniu. W tym celu stworzyliśmy prosty układ, który jest łatwo przetestować aby zaprezentować sposób, w który należałoby przetestować nasz układ.

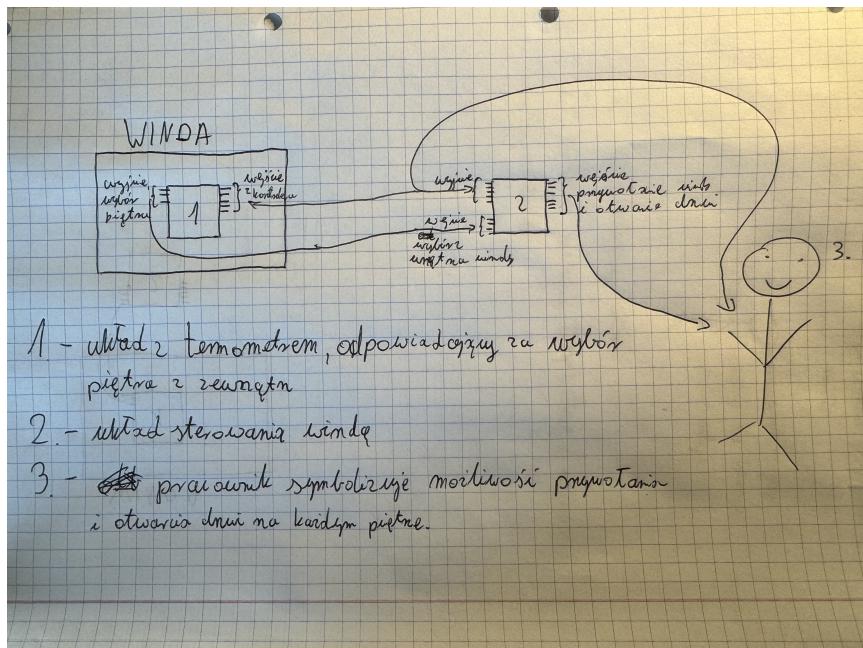
4 Wnioski i zastosowanie

Podzielenie automatu do obsługi windy na mniejsze automaty odpowiadające za różne elementy działania całego układu np. automat do obsługi pięter oraz drugi automat do obsługi zachowania drzwi ułatwiały skonstruowanie poprawnego układu testującego. Dodatkowo w znacznym stopniu zmniejszyły rozmiar tabeli prawdy, co umożliwiły użycie tabel Karnough robionych ręcznie, ponieważ przy tak dużym automacie jaki zaprezentowaliśmy w powyższym sprawozdaniu był zbyt duży do ręcznej optymalizacji.

Zaprojektowanie automatu Mealy'ego zamiast automatu Moore'a zmniejszyły liczbę potrzebnych do zaprojektowania stanów, ale prawdopodobnie skomplikowałoby jego zapis, a projekt stracił by na czytelności i intuicyjności.

Zastosowanie

Zaproponowany przez naszą grupę układ sterujący windą można zastosować w fabryce serów. Proces dojrzewania serów potrafi trwać od kilku dni do nawet 6 lat. Nieodpowiednie warunki mogą negatywnie wpływać na charakterystykę produktu do tego mogą się przyczynić do pęknięć i rozwoju pleśni. Podczas tego etapu bardzo ważne jest kontrolowanie temperatury w dojrzewalni, która musi być stała przez cały proces.



Rysunek 36: Szkic koncepcyjny zastosowania układu

Winda zostanie przystosowana do przechowywania serów oraz zostanie w niej zamontowany termometr. Zostaje w nim również zapisany przedział temperatur dla dojrzewającego obecnie rodzaju sera. W związku z tym, że temperatura musi być stała, a warunki atmosferyczne są zmienne, termometr na wejściu otrzymuje informacje na którym piętrze znajduje się widna oraz czy się porusza. Na podstawie otrzymanych wartości ze sterownika windą oraz relacji jaka zachodzi między obecną temperaturą, a zapisanym przedziałem decyduje o zmianie lub pozostaniu na obecnym piętrze (jego działanie odpowiada wyborze piętra z wnętrza windy). Różnicę temperatur na piętrach otrzymujemy dzięki umieszczeniu pierwszego piętra pod ziemią, drugiego na parterze, a trzeciego w wyższej części budynku. Wezwanie windy będzie zarezerwowane dla pracowników znajdujących się na konkretnych piętrach chcących skontrolować przebieg procesu dojrzewania.