Laboratorium podstaw techniki cyfrowej			
Zadanie nr 4	Temat: Reklama Tekstowa	Data oddania opracowania	20.12.2024
PTCLABID:		16	
Autor:			

1. Opis zadania

Celem zadania jest wykorzystanie podstawowych konstrukcji języka VHDL do opisu prostego układu cyfrowego z wykorzystaniem prostych elementów wejścia wyjścia płyty DE2. Wejściem układu będą przełączniki (symbole SW(0)-SW(17)) a wyjściem wyświetlacze siedmiosegmentowe oznaczone HEX0-HEX7. Ćwiczenie zakłada zastosowanie strukturalnej metody projektowania i równań Boolowskich określających sposób przetwarzania sygnałów w jednostkach układu.

2. Realizacja ćwiczenia:

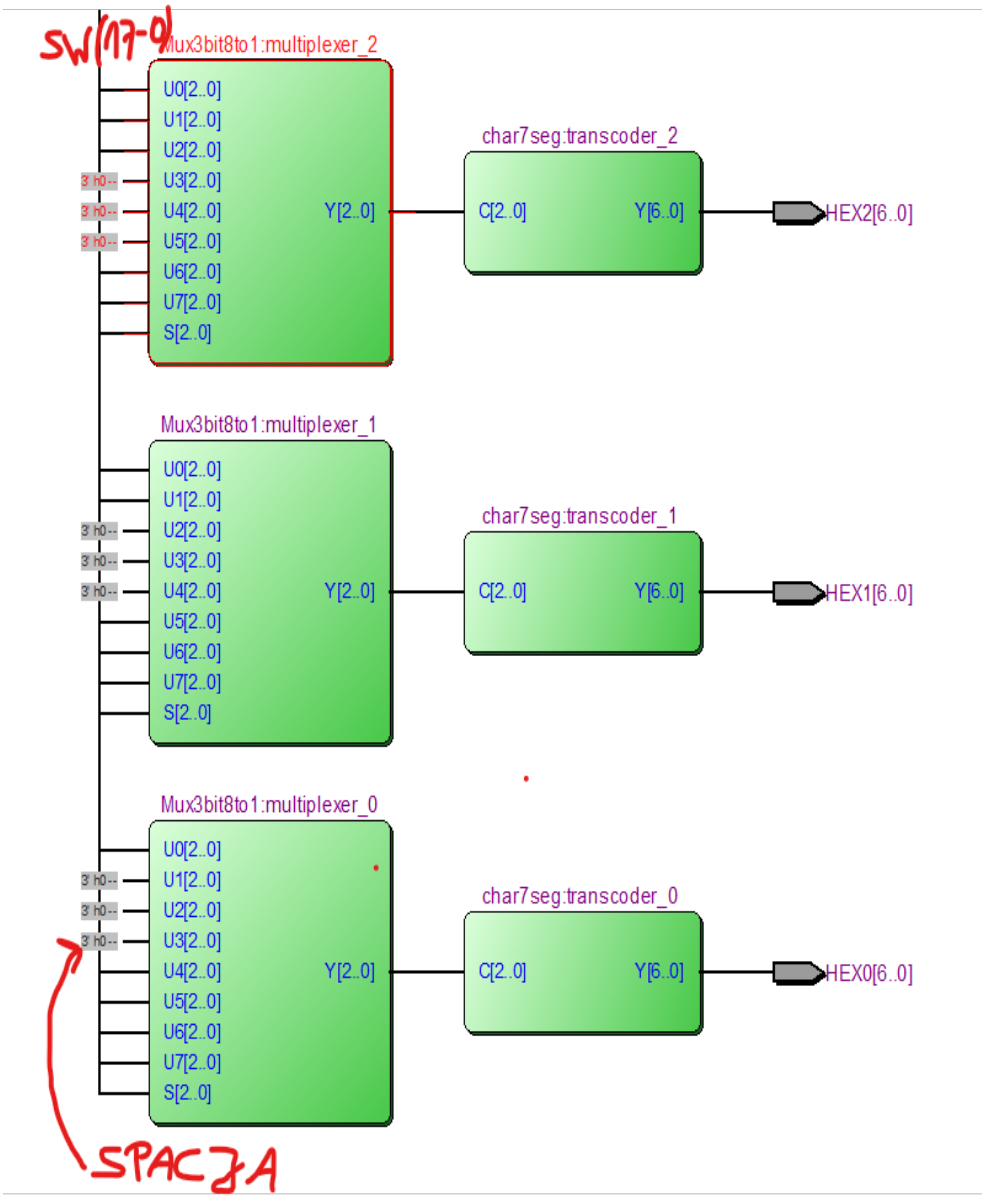
- Zaprojektowanie układu, który będzie wyświetlał na 8 wyświetlaczach siedmiosegmentowych od HEX0 do HEX7 słowo składającego się z liter E, F, G, H, I oraz spacji.
- Układ wyświetla na wyświetlaczach 8 znaków: 5 znaków liter i 3 znaki spacji
- Do zakodowania 5 znaków i spacji wykorzystano słowo kodowe 3 bitowe o 8 różnych wartościach (spacja zakodowana jest jako 000, a pozostałe znaki kodowane są od 7 w dół gdzie 1,2 to wartości dowolne)
- Słowo 8 znakowe przesuwa się cyklicznie w lewo na wyświetlaczach siedmiosegmentowych pod wpływem zmian na wejściach SW15-SW17 określających aktualne miejsce wyświetlania informacji wg. kodu Greya
- Wybrane słowo do wyświetlenia to w kolejności:
  SPACJA, SPACJA, E, F, I, H, G, SPACJA. Wektor wejść SW jest podzielony na wektory 3 bitowe, które zawierają informacje na temat kodowania jednego znaku (litery)

Układ zawiera następujące komponenty:

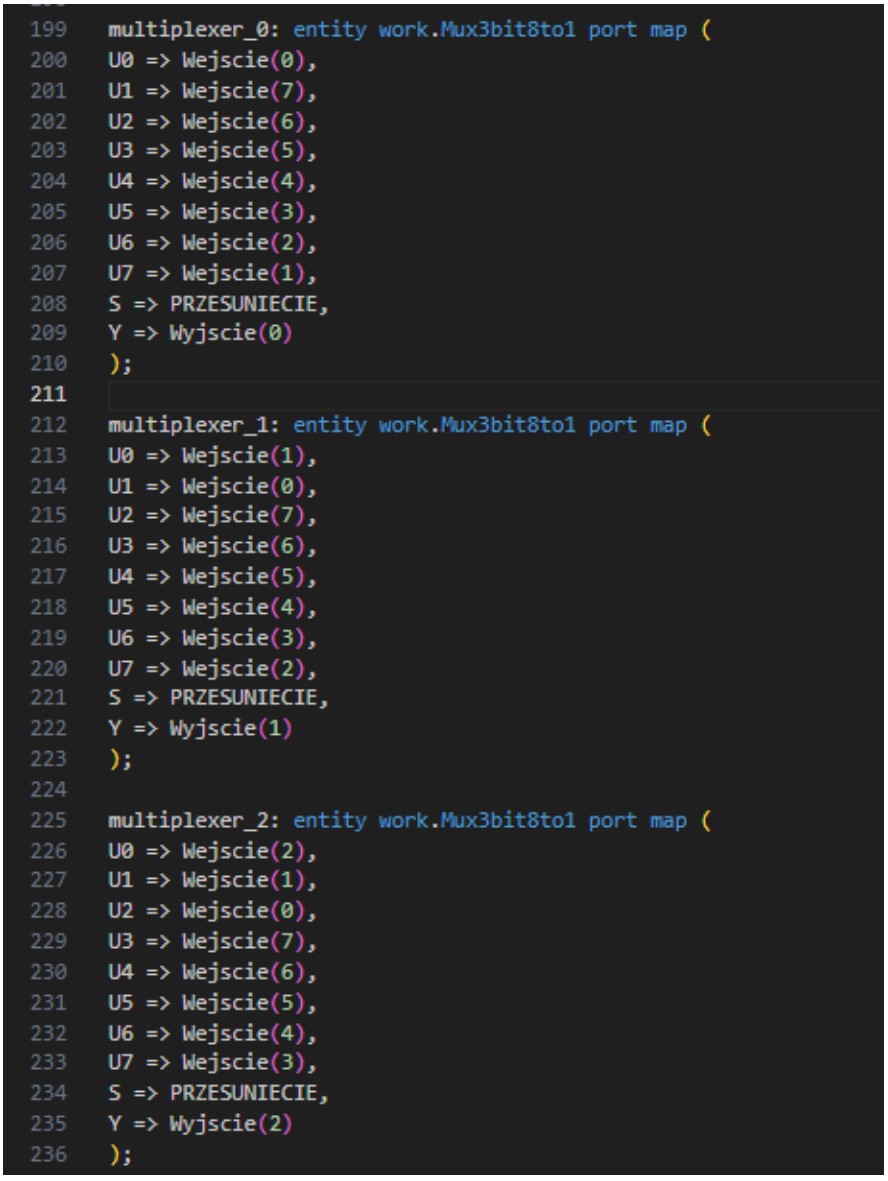
- Transkoder - zamienia kod (3 bitowy) na kod 7 bitowy wyświetlacza 7 segmentowego
- Multiplekser 1 z 8 – wybiera jedno z 8 wejść dostarczających kody znaków i spacji, jego wyjście podłączone jest do wejścia transkodera. Każde wejście multipleksera jest wektorem 3 bitowym, który koduje wyświetlane znaki. Każdy wyświetlacz prezentuje inną informację, więc informacje na tych samych wejściach różnych multiplekserów są różne. Sygnał podłączony do wejścia S decyduje o sygnale wyjściowym multipleksera. S reprezentuje sygnał 3 bitowy z wejść (SW15-SW17) i jest reprezentowany w kodzie Graya. Pozwala to na przesunięcie

wyświetlanych znaków przy zmianie tylko 1 bitu w wektorze S (na wejściach SW15-SW17).

Schemat Układu



Rys1. Schemat układu dla słowa składającego się z 5 liter i 3 spacji dla 3 kolejnych modułów wyświetlaczy(HEX0, HEX1, HEX2)
3 h0 oznacza znak spacji, tam gdzie nie występuję układ otrzyma sygnał z danego wektora U*. Jak przypisane są dane wejścia w podanym multiplekserze zostało przedstawione na rysunku poniżej.



```
multiplexer_0: entity work.Mux3bit8to1 port map (
U0 => Wejscie(0),
U1 => Wejscie(7),
U2 => Wejscie(6),
U3 => Wejscie(5),
U4 => Wejscie(4),
U5 => Wejscie(3),
U6 => Wejscie(2),
U7 => Wejscie(1),
S => PRZESUNIECIE,
Y => Wyjscie(0)
);

multiplexer_1: entity work.Mux3bit8to1 port map (
U0 => Wejscie(1),
U1 => Wejscie(0),
U2 => Wejscie(7),
U3 => Wejscie(6),
U4 => Wejscie(5),
U5 => Wejscie(4),
U6 => Wejscie(3),
U7 => Wejscie(2),
S => PRZESUNIECIE,
Y => Wyjscie(1)
);

multiplexer_2: entity work.Mux3bit8to1 port map (
U0 => Wejscie(2),
U1 => Wejscie(1),
U2 => Wejscie(0),
U3 => Wejscie(7),
U4 => Wejscie(6),
U5 => Wejscie(5),
U6 => Wejscie(4),
U7 => Wejscie(3),
S => PRZESUNIECIE,
Y => Wyjscie(2)
);
```

Rys2. Konkretyzacja 3 kolejnych multiplekserów

Wejscie – tablica 8 elementowa, każdy element jest 3 bitowy

Wejscie(0-4) – oznacza 5 wektorów wejściowych (3 bitowych)

Wejscie(5-7) – oznacza spacje

PRZESUNIECIE – 3 bitowy sygnał sterujący multiplekserem, decyduje z którego wejścia ma pobrać sygnał multiplekser.

Aby otrzymać przesunięcie w lewo musimy dla każdego następnego multipleksera przypisać na kolejne wejście. Dla multipleksera 0 dla U0 przypisujemy Wejscie(0), więc dla multipleksera 1 dla U0 musimy przypisać kolejne wejście czyli U1.

Takie podłączenie sygnałów wejściowych multiplekserów sprawi, że na wyświetlaczach nie będzie wyświetlana ta sama litera 2 razy (jeżeli wektory Wejście(0-4) są różne) a po zmianie sygnału wektora PRZESUNIECIE wyświetlane słowo będzie przesuwać się w lewo.

Wejścia	Wektor	Informacja
SW[2-0]	WEJSCIE(0)	ZNAK1
SW[5-3]	WEJSCIE(1)	ZNAK2
SW[8-6]	WEJSCIE(2)	ZNAK3
SW[11-9]	WEJSCIE(3)	ZNAK4
SW[14-12]	WEJSCIE(4)	ZNAK5
SW[17-15]	PRZESUNIECIE	PRZESUNIECIE
-	WEJSCIE(5-7)	SPACJA

Tabela 1. Tabela, opisująca, jakie dane z wejścia układu zawierają w sobie poszczególne wektory oraz jaką informację przenoszą dane wektory

Kod w NKB	Znak
111	E
110	F
101	G
100	H
011	I
000	SPACJA

Tabela 2: Znaki przypadające na odpowiednią 3-bitową informacje

Znaki są zakodowane jako kolejne litery wartościami od 7 w dół. Wartości 1 oraz 2 nie są wykorzystywane i nie kodują żadnego znaku.

MULTIPLEKSER			
	0	1	2
U0	WEJSCIE(0)	WEJSCIE(1)	WEJSCIE(2)
U1	WEJSCIE(7)	WEJSCIE(0)	WEJSCIE(1)
U2	WEJSCIE(6)	WEJSCIE(7)	WEJSCIE(0)
U3	WEJSCIE(5)	WEJSCIE(6)	WEJSCIE(7)
U4	WEJSCIE(4)	WEJSCIE(5)	WEJSCIE(6)
U5	WEJSCIE(3)	WEJSCIE(4)	WEJSCIE(5)
U6	WEJSCIE(2)	WEJSCIE(3)	WEJSCIE(4)
U7	WEJSCIE(1)	WEJSCIE(2)	WEJSCIE(3)
S	PRZESUNIECIE	PRZESUNIECIE	PRZESUNIECIE
Y	WYJSCIE(0)	WYJSCIE(1)	WYJSCIE(2)

Tabela 3: Tabela opisująca sygnały podłączone do multiplekserów

Wyjscie – tablica 8 elementowa, przechowująca wektor 3 bitowy dla każdego multipleksera (wyjście(0-7))

Po multiplekserze(x), wybrany sygnał z wyjście(x) jest przekazywany na wejście C transkodera(x), który zamienia sygnał 3 bitowy na odpowiedni sygnał 7 bitowy służący do wyświetlenia odpowiedniego znaku na wyświetlaczu.

3. Funkcje wyjść multipleksera

Sygnał wyjściowy Y multipleksera to 3-bitowy sygnał składający się z kolejnych bitów Y(0), Y(1), Y(2) (bity od najmłodszego do najstarszego). Sygnał ten kontrolowany jest przez sygnał wejściowy S (3-bitowy sygnał składający się z S(0), S(1), S(2)).

S(2) \ S(1), S(0)	00	01	11	10
0	U0	U1	U2	U3
1	U7	U6	U5	U4

Tabela 4: Tablica prawdy dla wyjścia multipleksera Y(S)

Powyższą tabelę można rozpisać na poszczególne bity wyjścia multipleksera:

S(2) \ S(1), S(0)	00	01	11	10
0	U0(0)	U1(0)	U2(0)	U3(0)
1	U7(0)	U6(0)	U5(0)	U4(0)

Tabela 5: Tablica prawdy dla wyjścia multipleksera Y(0) (S(0), S(1), S(2))

Funkcja Y(0) (S(0), S(1), S(2)) w postaci kanonicznej:

Y(0) (S(0),S(1),S(2)) =
not S(2) * not S(1) * not S(0) * U0(0) --000
+ not S(2) * not S(1) * S(0) * U1(0) --001
+ not S(2) * S(1) * S(0) * U2(0) --011
+ not S(2) * S(1) * not S(0) * U3(0) --010
+ S(2) * S(1) * not S(0) * U4(0) --110
+ S(2) * S(1) * S(0) * U5(0) --111
+ S(2) * not S(1) * S(0) * U6(0) --101
+ S(2) * not S(1) * not S(0) * U7(0) --100

S(2) \ S(1), S(0)	00	01	11	10
0	U0(1)	U1(1)	U2(1)	U3(1)
1	U7(1)	U6(1)	U5(1)	U4(1)

Tabela 6: Tablica prawdy dla wyjścia multipleksera Y(1) (S(0), S(1), S(2))

Funkcja Y(1) (S(0), S(1), S(2)) w postaci kanonicznej:

Y(1) (S(0),S(1),S(2)) =
not S(2) * not S(1) * not S(0) * U0(1) --000
+ not S(2) * not S(1) * S(0) * U1(1) --001
+ not S(2) * S(1) * S(0) * U2(1) --011
+ not S(2) * S(1) * not S(0) * U3(1) --010
+ S(2) * S(1) * not S(0) * U4(1) --110
+ S(2) * S(1) * S(0) * U5(1) --111
+ S(2) * not S(1) * S(0) * U6(1) --101
+ S(2) * not S(1) * not S(0) * U7(1) --100

S(2) \ S(1), S(0)	00	01	11	10
0	U0(2)	U1(2)	U2(2)	U3(2)
1	U7(2)	U6(2)	U5(2)	U4(2)

Tabela 7: Tablica prawdy dla wyjścia multipleksera Y(2) (S(0), S(1), S(2))

Funkcja Y(2) (S(0), S(1), S(2)) w postaci kanonicznej:

Y(2) (S(0),S(1),S(2)) =
not S(2) * not S(1) * not S(0) * U0(2) --000
+ not S(2) * not S(1) * S(0) * U1(2) --001
+ not S(2) * S(1) * S(0) * U2(2) --011
+ not S(2) * S(1) * not S(0) * U3(2) --010
+ S(2) * S(1) * not S(0) * U4(2) --110
+ S(2) * S(1) * S(0) * U5(2) --111
+ S(2) * not S(1) * S(0) * U6(2) --101
+ S(2) * not S(1) * not S(0) * U7(2) --100

4. Transkoder

KOD ZNAKU	ZNAK	Reprezentacja 7 bitów potrzebna dla wyświetlacza	Wartość dziesiętna reprezentacji
111	E	0000110	6
110	F	0001110	14
101	G	1000010	66
100	H	0001001	9
011	I	1111001	121
000	SPACJA	1111111	127

Tabela 8: Tabela przedstawia sposób kodowania każdego znaku oraz jego reprezentacje dla wyświetlacza siedmiosegmentowego, u którego do zapalenia
segmentu niezbędny jest sygnał niski

C(2) \ C(1), C(0)	00	01	11	10
0	1	-	1	-
1	1	0	0	0

Tabela 9: Tablica prawdy dla wyjścia transkodera dla Y(0) (C(2),C(1),C(0)) (górny segment na wyświetlaczu oznaczany jako a)

Funkcja Y(0) (C(2),C(1),C(0)) w postaci kanonicznej:
Y(0) (C(2),C(1),C(0)) =
= C(2) * C(1) * C(0) * 0 --E (111)
+ C(2) * C(1) * not C(0) * 0 --F (110)
+ C(2) * not C(1) * C(0) * 0 --G(101)
+ C(2) * not C(1) * not C(0) * 1 --H(100)
+ not C(2) * C(1) * C(0) * 1 --I (011)
+ not C(2) * C(1) * not C(0) * D
+ not C(2) * not C(1) * C(0) * D
+ not C(2) * not C(1) * not C(0) * 1 --SPACJA

C(2) \ C(1), C(0)	00	01	11	10
0	1	-	0	-
1	0	1	1	1

Tabela 10: Tablica prawdy dla wyjścia transkodera dla Y(1) (C(2),C(1),C(0)) (prawy górny segment na wyświetlaczu oznaczany jako b)

Funkcja Y(1) (C(2),C(1),C(0)) w postaci kanonicznej:
Y(1) (C(2),C(1),C(0)) =
= C(2) * C(1) * C(0) * 1 --E (111)
+ C(2) * C(1) * not C(0) * 1 --F (110)
+ C(2) * not C(1) * C(0) * 1 --G(101)
+ C(2) * not C(1) * not C(0) * 0 --H(100)
+ not C(2) * C(1) * C(0) * 0 --I (011)
+ not C(2) * C(1) * not C(0) * D
+ not C(2) * not C(1) * C(0) * D
+ not C(2) * not C(1) * not C(0) * 1 --SPACJA

C(2) \ C(1), C(0)	00	01	11	10
0	1	-	0	-
1	0	0	1	1

Tabela 11: Tablica prawdy dla wyjścia transkodera dla Y(2) (C(2),C(1),C(0)) (prawy dolny segment na wyświetlaczu oznaczany jako c)

Funkcja Y(2) (C(2),C(1),C(0)) w postaci kanonicznej:
Y(2) (C(2),C(1),C(0)) =
= C(2) * C(1) * C(0) * 1 --E (111)
+ C(2) * C(1) * not C(0) * 1 --F (110)
+ C(2) * not C(1) * C(0) * 0 --G(101)
+ C(2) * not C(1) * not C(0) * 0 --H(100)
+ not C(2) * C(1) * C(0) * 0 --I (011)
+ not C(2) * C(1) * not C(0) * D
+ not C(2) * not C(1) * C(0) * D
+ not C(2) * not C(1) * not C(0) * 1 --SPACJA

C(2) \ C(1), C(0)	00	01	11	10
0	1	-	1	-
1	1	0	0	1

Tabela 12: Tablica prawdy dla wyjścia transkodera dla Y(3) (C(2),C(1),C(0)) (dolny segment na wyświetlaczu oznaczany jako d)

Funkcja Y(3) (C(2),C(1),C(0)) w postaci kanonicznej:
Y(3) (C(2),C(1),C(0)) =
= C(2) * C(1) * C(0) * 0 --E (111)
+ C(2) * C(1) * not C(0) * 1 --F (110)
+ C(2) * not C(1) * C(0) * 0 --G(101)
+ C(2) * not C(1) * not C(0) * 1 --H(100)
+ not C(2) * C(1) * C(0) * 1 --I (011)
+ not C(2) * C(1) * not C(0) * D
+ not C(2) * not C(1) * C(0) * D
+ not C(2) * not C(1) * not C(0) * 1 --SPACJA

C(2) \ C(1), C(0)	00	01	11	10
0	1	-	1	-
1	0	0	0	0

Tabela 13: Tablica prawdy dla wyjścia transkodera dla Y(4) (C(2),C(1),C(0)) (lewy dolny segment na wyświetlaczu oznaczany jako d)

Funkcja Y(4) (C(2),C(1),C(0)) w postaci kanonicznej:
Y(4) (C(2),C(1),C(0)) =
= C(2) * C(1) * C(0) * 0 --E (111)
+ C(2) * C(1) * not C(0) * 0 --F (110)
+ C(2) * not C(1) * C(0) * 0 --G(101)
+ C(2) * not C(1) * not C(0) * 0 --H(100)
+ not C(2) * C(1) * C(0) * 1 --I (011)
+ not C(2) * C(1) * not C(0) * D
+ not C(2) * not C(1) * C(0) * D
+ not C(2) * not C(1) * not C(0) * 1 --SPACJA

C(2) \ C(1), C(0)	00	01	11	10
0	1	-	1	-
1	0	0	0	0

Tabela 14: Tablica prawdy dla wyjścia transkodera dla Y(5) (C(2),C(1),C(0)) (lewy górny segment na wyświetlaczu oznaczany jako e)

Funkcja Y(5) (C(2),C(1),C(0)) w postaci kanonicznej:

Y(5) (C(2),C(1),C(0)) =
= C(2) * C(1) * C(0) * 0 --E (111)
+ C(2) * C(1) * not C(0) * 0 --F (110)
+ C(2) * not C(1) * C(0) * 0 --G(101)
+ C(2) * not C(1) * not C(0) * 0 --H(100)
+ not C(2) * C(1) * C(0) * 1 --I (011)
+ not C(2) * C(1) * not C(0) * D
+ not C(2) * not C(1) * C(0) * D
+ not C(2) * not C(1) * not C(0) * 1 --SPACJA

C(2) \ C(1), C(0)	00	01	11	10
0	1	-	1	-
1	0	1	0	0

Tabela 15: Tablica prawdy dla wyjścia transkodera dla Y(6) (C(2),C(1),C(0)) (środkowy segment na wyświetlaczu oznaczany jako f)

Funkcja Y(6) (C(2),C(1),C(0)) w postaci kanonicznej:

Y(6) (C(2),C(1),C(0)) =
= C(2) * C(1) * C(0) * 0 --E (111)
+ C(2) * C(1) * not C(0) * 0 --F (110)
+ C(2) * not C(1) * C(0) * 1 --G(101)
+ C(2) * not C(1) * not C(0) * 0 --H(100)
+ not C(2) * C(1) * C(0) * 1 --I (011)
+ not C(2) * C(1) * not C(0) * D
+ not C(2) * not C(1) * C(0) * D
+ not C(2) * not C(1) * not C(0) * 1 --SPACJA

5. Symulacja czasowa

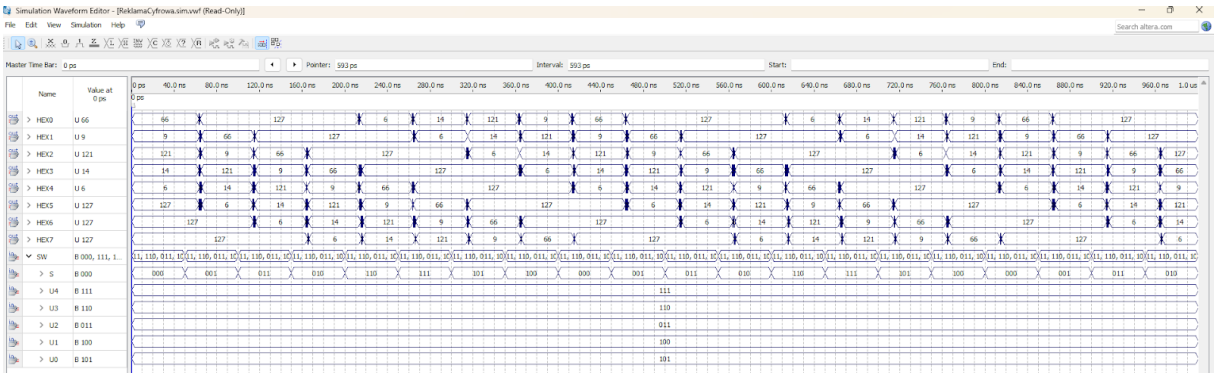
Wybrane słowo: SPACJA, SPACJA, E, F, I, H, G, SPACJA

Aby wyświetlić taką sekwencję na układzie musimy oryginalne słowo:

SPACJA SPACJA SPACJA E F I H G przesunąć o 1 pozycję w lewo, aby wyświetlić oryginalne słowo należy na wejściu nadać odpowiednie sygnały:

ZNAK	WEKTOR	SYGNAŁ NA WEJŚCIU	WARTOŚĆ 7 BITOWA W SYSTEMIE 10 NA WYŚWIETLACZU
G	U0	101	66
H	U1	100	9
I	U2	011	121
F	U3	110	14
E	U4	111	6
SPACJA	-	-	127
SPACJA	-	-	127
SPACJA	-	-	127
PRZESUNIECIE	S	000	0

Tabela 16. Sygnały na wejściu potrzebne do uzyskania oryginalnego słowa



Rys 3. Przebiegi symulacji czasowej dla S – zmieniającego się o 50ns.

Jak widać na powyższym rysunku układ działa poprawnie, dla S – 001 otrzymujemy oczekiwane słowo, wszystkie wartości przesuwają się przy zmieniającym się S w następnym kroku o „1 w dół” o oznacza że np. przechodzą z HEX0 do HEX1. HEX0 oznacza wyświetlacz najbardziej po prawo a HEX7 najbardziej po lewo, więc przesuwanie działa poprawnie.

Opis zawartości symulacji

- SW – sygnały z wejść SW[17-0]
- U0 – Wektor 3 bitowy składający się z sygnałów SW(2-0) reprezentuje literę „G”
- U1 - Wektor 3 bitowy składający się z sygnałów SW(5-3) reprezentuje literę „H”
- U2- Wektor 3 bitowy składający się z sygnałów SW(8-6) reprezentuje literę „I”
- U3 - Wektor 3 bitowy składający się z sygnałów SW(11-9) reprezentuje literę „F”
- U4 - Wektor 3 bitowy składający się z sygnałów SW(14-12) reprezentuje literę „E”
- S – Wektor 3 bitowy składający się z sygnałów wejść SW17-SW15 określający etap pracy wyświetlacza; działa w kodzie Graya.
- hex0-hex7 – siedmiosegmentowe wyświetlacze. (hex7 wyświetlacz najbardziej po lewej hex0 po prawej). W symulacji wyświetlają wartość potrzebną do zapalenia odpowiednich segmentów wyświetlacza jako wartość dziesiętną. Wartości przesuwają się po wyświetlaczach w lewo (w symulacji przechodzą kolejno z hex0 do hex7 w dół) z każdą zmianą S