Maszyny stanów w języku VHDL.

1 Treść zadania

Zadaniem do wykonania jest detektor sekwencji 011_100. Na zajęciach zostały zadane 2 obligatoryjne zadania i jedno dodatkowe:

- automat Moore,
- automat Mealy,
- dowolny automat jako projekt hierarchiczny z *HexTo7Seg*.

Automat ma posiadać zegar CLK_LF o częstotliwości $f_{CLK} = 33\frac{1}{3}MHz$, przyciski RESET (R7), Clock Enable (K0) i wejście X (K1) oraz wyjście Y (LED 0). Należy przeprowadzić symulację behawioralną wczytując odpowiednią 21-elementową sekwencję (STD_LOGIC_VECTOR(0 to 20)), a przed zboczem narastającym zegara x musi być już ustabilizowany (zmiana wejścia X ma nastapić 10 nanosekund przed cyklem zegara).

2 Grafy automatów

Automaty to układy cyfrowe realizujące zadane sekwencje (detekcję ciągu, odczytanie słów itp.). Każdy automat posiada swój stan początkowy i na podstawie ciągu wprowadzonych danych w odpowiednich cyklach zegarowych. Automat ma dać odpowiedni sygnał wejściowy w zależności od wprowadzonych danych. Rozróżniane na zajęciach są dwa rodzaje automatów: Moore i Mealy. Różnica polega na tym, że w tym pierwszym wyjście zależy tylko od stanu, w jakim automat się znajduje, a w drugim - od stanu i wejścia, jakie zostało podane. W automatach Moore'a zazwyczaj ostatni stan to jest ten, w którym dana sekwencja jest akceptowana, więc przyjmuje on wyjście 1 (wykryta poprawna sekwencja), pozostałe stany mają 0 (sekwencja niewykryta/niepełna). Automat Mealy z racji swoich warunków najczęściej ma 1 stan mniej. Z racji podanej wcześniej cechy, logiczna jedynka pojawi się na wyjściu od razu po wprowadzeniu danych, przed zboczem narastającym zegara.

2.1 Moore

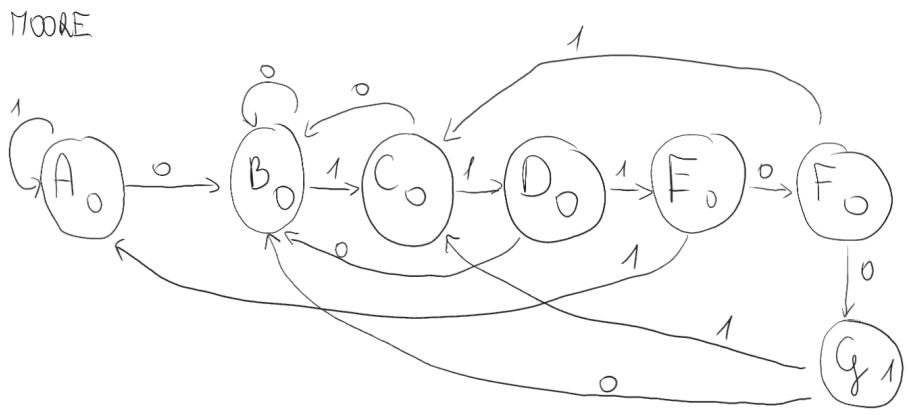


Diagram 1: Graf w postaci automatu Moore'a

2.2 Mealy

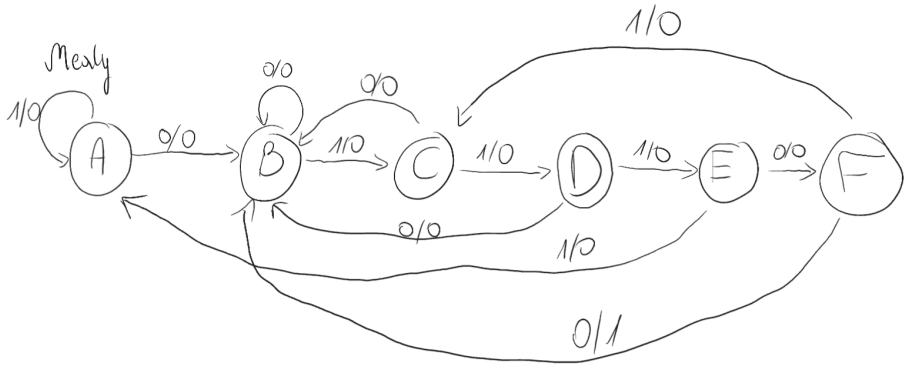


Diagram 2: Graf w postaci automatu Mealy'ego

3 Pliki VHDL

Oba automaty podzielone zostały formalnie na 2 procesy:
Pierwszy z nich odpowiada za zmianę stanu podczas zbocza narastającego zegara oraz resetowanie układu, a drugi jest instrukcją *case...when*, w której zakodowany jest każdy ze stanów oraz odpowiednie przejścia. Poza procesem występuje też wyprowadzenie wyjścia, w zależności od stanu (Moore) lub stanu i wejścia (Mealy).

3.1 Moore

Kod pliku tworzącego detektor sekwencji zrealizowany jako automat Moore:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity moore is
5      Port ( X : in  STD_LOGIC;
6             Y : out  STD_LOGIC;
7             CE : in  STD_LOGIC;
8             CLK : in  STD_LOGIC;
9             RST : in  STD_LOGIC);
10 end moore;
11 architecture Behavioral of moore is
12 type state_type is (A, B, C, D, E, F, G);
13 signal state, next_state : state_type;
14
15 begin
16 process1 : process(CLK)
17   begin
18     if rising_edge(CLK) then
19       if RST = '1' then
20         state <= A;
21       else
22         state <= next_state;
23       end if;
24     end if;
25   end process process1;
26
27 process2 : process(state, X, CE)
28   begin
29     next_state <= state;
30     if CE = '1' then
```

```
31        case state is
32          when A =>
33            if X = '0' then
34              next_state <= B;
35            end if;
36          when B =>
37            if X = '1' then
38              next_state <= C;
39            end if;
40          when C =>
41            if X = '0' then
42              next_state <= B;
43            else
44              next_state <= D;
45            end if;
46          when D =>
47            if X = '0' then
48              next_state <= B;
49            else
50              next_state <= E;
51            end if;
52          when E =>
53            if X = '0' then
54              next_state <= F;
55            else
56              next_state <= A;
57            end if;
58          when F =>
59            if X = '0' then
60              next_state <= G;
61            else
62              next_state <= C;
63            end if;
64          when G =>
65            if X = '0' then
66              next_state <= B;
67            else
68              next_state <= C;
69            end if;
70        end case;
71      end if;
72    end process process2;
73  Y <= '1' when state = G
74        else '0';
75  end Behavioral;
```

3.2 Mealy

Kod pliku tworzącego detektor sekwencji zrealizowany jako automat Moore:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity mealy is
5      Port ( X : in  STD_LOGIC;
6             Y : out  STD_LOGIC;
7             CE : in  STD_LOGIC;
8             CLK : in  STD_LOGIC;
9             RST : in  STD_LOGIC);
10 end mealy;
11 architecture Behavioral of mealy is
12 type state_type is (A, B, C, D, E, F);
13 signal state, next_state : state_type;
14
15 begin
16 process1 : process(CLK)
17   begin
18     if rising_edge(CLK) then
19       if RST = '1' then
20         state <= A;
21       else
22         state <= next_state;
23       end if;
24     end if;
25   end process process1;
26
27 process2 : process(state, X, CE)
28   begin
29     next_state <= state;
30     if CE = '1' then
31       case state is
32         when A =>
33           if X = '0' then
34             next_state <= B;
35           end if;
36         when B =>
37           if X = '1' then
38             next_state <= C;
39           end if;
40         when C =>
41           if X = '0' then
42             next_state <= B;
43           else
44             next_state <= D;
45           end if;
46         when D =>
47           if X = '0' then
48             next_state <= B;
49           else
50             next_state <= E;
51           end if;
52         when E =>
53           if X = '0' then
54             next_state <= F;
55           else
56             next_state <= A;
57           end if;
58         when F =>
59           if X = '0' then
60             next_state <= B;
```

```
61            else
62              next_state <= C;
63            end if;
64        end case;
65      end if;
66    end process process2;
67 Y <= '1' when state = F and X = '0'
68        else '0';
69 end Behavioral;
```

4 Testbench

Procedura testowa polega na stworzeniu symulacji behawioralnej, która przyjmuje następujący ciąg wejściowy: $***abcdef***abcde\overline{f}***$, czyli trzy losowe wejścia, następnie ciąg prawidłowy, potem znowu trzy losowe symbole, poprawny ciąg z zanegowanym ostatnim sygnałem oraz trzy losowe wejścia. Należało pamiętać, aby $f_{CLK} = 33\frac{1}{3}$ MHz oraz żeby wywołanie zmiany sygnału zegara nastąpiło po ustabilizowaniu wartości sygnału wejścia x. Wartość opóźnienia została ustalona na 10 ns. Dodatkowo w oknie symulacji na ekran został wyprowadzony wewnętrzny sygnał *state* (w oknie Wave).

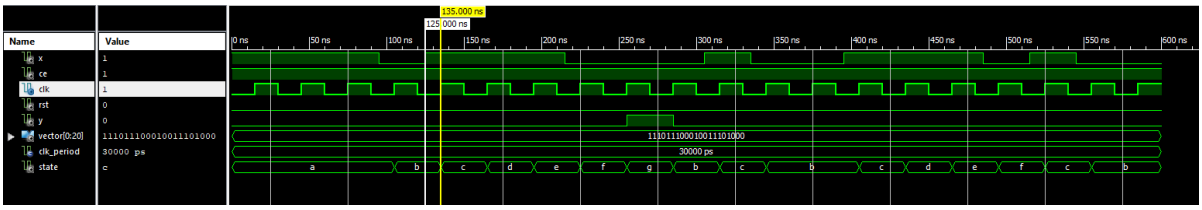
4.1 Moore

Listing testbencha dla automatu Moore:

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3  ENTITY moore_tbh IS
4  END moore_tbh;
5  ARCHITECTURE behavior OF moore_tbh IS
6      COMPONENT moore
7      PORT(
8           X : IN  std_logic;
9           Y : OUT  std_logic;
10          CE : IN  std_logic;
11          CLK : IN  std_logic;
12          RST : IN  std_logic
13         );
14     END COMPONENT;
15
16    signal X : std_logic ;
17    signal CE : std_logic := '1';
18    signal CLK : std_logic := '0';
19    signal RST : std_logic := '0';
20
21     signal Y : std_logic;
22     signal vector : STD_LOGIC_VECTOR(0 to 20) := "11101110000100111101000";
23     constant CLK_period : time := 30 ns;
24
25 BEGIN
26    uut: moore PORT MAP (
27           X => X,
28           Y => Y,
29           CE => CE,
30           CLK => CLK,
31           RST => RST
32         );
33
34    CLK_process :process
35    begin
36     CLK <= '0';
```

```
37      wait for CLK_period/2;
38      CLK <= '1';
39      wait for CLK_period/2;
40    end process;
41
42    stim_proc: process
43    begin
44      X <= '1';
45      wait for 5 ns;
46        for i in 0 to 20 loop
47        X <= vector(i);
48        wait for 30 ns;
49      end loop;
50        wait;
51    end process;
52 END;
```

Wynik symulacji można zaobserwować na zrzucie ekranu:



Symulacja behawioralna dla automatu Moore

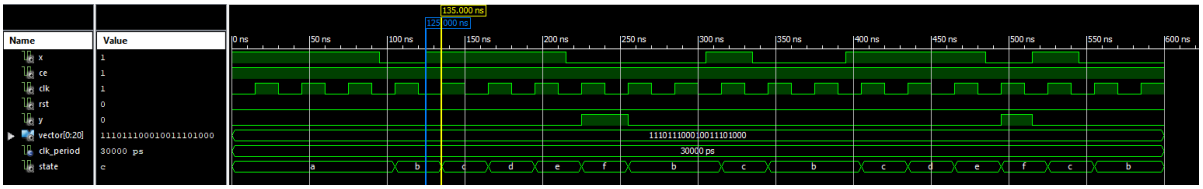
4.2 Mealy

Listing testbencha dla automatu Mealy:

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3  ENTITY mealy_tbh IS
4  END mealy_tbh;
5  ARCHITECTURE behavior OF mealy_tbh IS
6      COMPONENT mealy
7      PORT(
8           X : IN  std_logic;
9           Y : OUT  std_logic;
10          CE : IN  std_logic;
11          CLK : IN  std_logic;
12          RST : IN  std_logic
13         );
14     END COMPONENT;
15
16    signal X : std_logic ;
17    signal CE : std_logic := '1';
18    signal CLK : std_logic := '0';
19    signal RST : std_logic := '0';
20
21     signal Y : std_logic;
22   signal vector : STD_LOGIC_VECTOR(0 to 20) := "11101110001001101000";
23     constant CLK_period : time := 30 ns;
24
25 BEGIN
26    uut: mealy PORT MAP (
27           X => X,
28           Y => Y,
29           CE => CE,
30           CLK => CLK,
```

```
31            RST => RST
32          );
33
34     CLK_process :process
35     begin
36      CLK <= '0';
37      wait for CLK_period/2;
38      CLK <= '1';
39      wait for CLK_period/2;
40     end process;
41
42     stim_proc: process
43     begin
44      X <= '1';
45      wait for 5 ns;
46        for i in 0 to 20 loop
47        X <= vector(i);
48        wait for 30 ns;
49      end loop;
50        wait;
51     end process;
52 END;
```

Wynik symulacji można zaobserwować na zrzucie ekranu:



Symulacja behawioralna dla automatu Mealy

5 Wgranie na płytę ZL-9572

Wgranie powyższego programu nie sprawiło prawie żadnych problemów, diody reagowały na przyciski poprawnie, tj. razem z wybranym kierunkiem oraz przy wciśniętym przycisku *CE (Clock Enable)*. Należało pamiętać, aby w odpowiednich odstępach czasowych klikać lub odpuszczać przycisk, gdyż trzeba było "wstrzelić się" w takty zegara. Ponadto w pliku *.ucf* odkomentowane oraz zedytowane zostały następujące linie

```
1  # Clocks
2  NET "CLK" LOC = "P7" | BUFG = CLK;
3  NET "CLK" PERIOD = 30ns HIGH 50%;
4
5  # Keys
6  NET "CE" LOC = "P42";
7  NET "X" LOC = "P40";
8  NET "RST" LOC = "P39";  # GSR
9
10 # LEDS
11 NET "Y"  LOC = "P35";
```

6 Projekt hierarchiczny

Projekt hierarchiczny polegał na stworzeniu układu, który wykorzystuje logicznie poprzednie zadanie (detektor sekwencji - dowolnie Moore lub Mealy *CLR - Clear*, *CE - Clock Enable*, *DIR - Direction*), ale połączony jest z blokiem *HexTo7Seg*. Podczas zajęć udało nam się zaimportować schemat z poprzedniego zadania jako blok, który można użyć na potrzeby projektu hierarchicznego.

7 Wnioski

Pierwsze dwa zadania zostały w pełni wykonane bez problemu. W zadaniu trzecim, czyli projekcie hierarchicznym zabrakło czasu na przetestowanie i ewentualne poprawienie układu. Obserwacja zmiany stanów automatów w symulacjach behawioralnych ukazuje, że automaty działają w sposób planowy. Warto zauważyć, że w symulacji automatu Moore'a wyjście pokazuje sygnał 1 tylko raz, gdzie w symulacji automatu Mealy'ego wyjście Y wskazuje sygnał 1 dwa razy. Jest to spowodowane konstrukcją automatu Mealy i tego, że po wprowadzeniu przedostatniego znaku sekwencji automat wszedł do ostatniego stanu, a z racji iż miał już na wejściu logiczne '0', to na wyjściu, zgodnie z grafem, pojawił się sygnał '1'. Drugie wystąpienie jedynki logicznej jest krótsze niż pierwsze, ponieważ szybciej nastąpiła zmiana sygnału wejściowego, przez co warunek wykrycia sekwencji nie został spełniony (mimo iż automat nie zmienił jeszcze stanu, ponieważ nadal czekał na zbocze narastające zegara).