

POLITECHNIKA WROCŁAWSKA

UKŁADY CYFROWE I SYSTEMY WBUDOWANE

Dokumentacja projektu
Organy z możliwością zapisywania i odtwarzania
melodii.

Autorzy:

Łukasz BIESZCZAD

Krzysztof BUCZAK

Prowadzący:

dr inż. Jarosław SUGIER

18 maja 2018

Spis treści

1	Wprowadzenie	2
1.1	Cel i zakres	2
1.2	Zagadnienia teoretyczne	2
1.3	Sprzęt	2
2	Projekt	3
2.1	Schemat i hierarchia projektu	3
2.2	Moduły	4
2.2.1	Modulator2	4
2.2.2	Synth2	8
2.2.3	Switch	9
2.2.4	RAM	12
2.2.5	MsgGenerator	13
3	Implementacja	14
3.1	Zasoby	14
3.2	"User manual" urządzenia	14
4	Podsumowanie	14
5	Literatura	14

1 Wprowadzenie

1.1 Cel i zakres

Celem projektu było stworzenie jednooktawowego "instrumentu" klawiszowego, obsługiwanego za pomocą klawiatury PS/2. Wciskanie poszczególnych klawiszy miało powodować odtwarzanie dźwięków przez podpięty do pinów głośniczek. Dodatkowo częścią zadania było także zaimplementowanie funkcjonalności nagrywania melodii (zapis do pamięci ROM) i odtwarzania nagranych materiału, a także wykorzystanie wyświetlacza LCD do pokazania stanu nagrywania i diody LED do przekazania informacji o trwającym właśnie nagrywaniu.

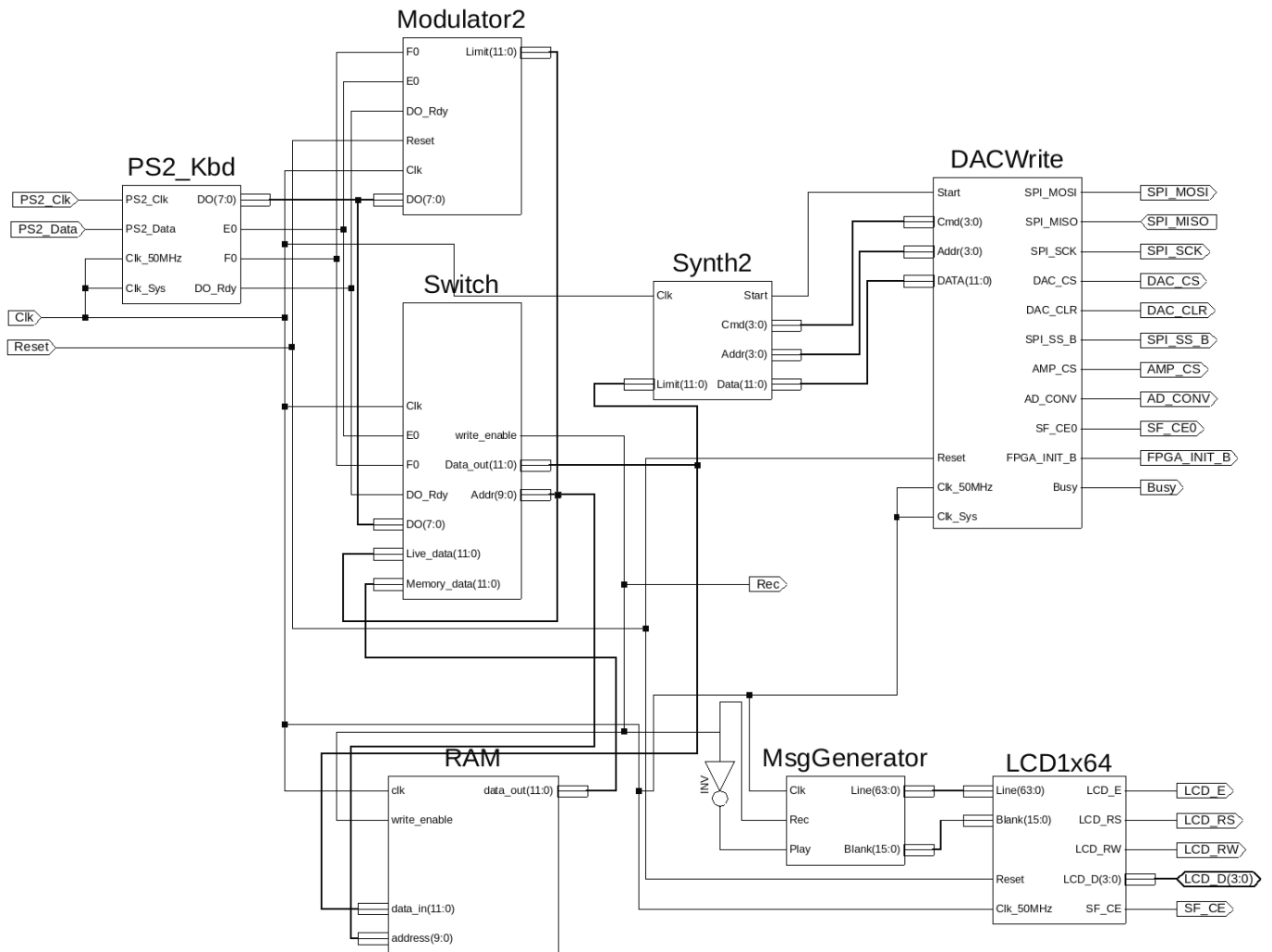
1.2 Zagadnienia teoretyczne

1.3 Sprzęt

Językiem projektu był język opisu sprzętu VHDL. Stanowisko laboratoryjne/projektowe zostało wyposażone w układ Spartan-3E oraz komputer z oprogramowaniem Xilinx ISE, pozwalającym kompilować kod VHDL pod dostarczony sprzęt, a także wykonywać symulacje testujące działanie systemu. Wykorzystaliśmy także port PS/2 (klawiatura symulująca keyboard), piny do podłączenia głośnika, ekran LCD wyświetlający stan nagrywania (pozostały czas) i układ pamięci ROM do przechowywania nagranych melodii.

2 Projekt

2.1 Schemat i hierarchia projektu



Rysunek 1: Schemat całego projektu.

Hierarchia modułów:

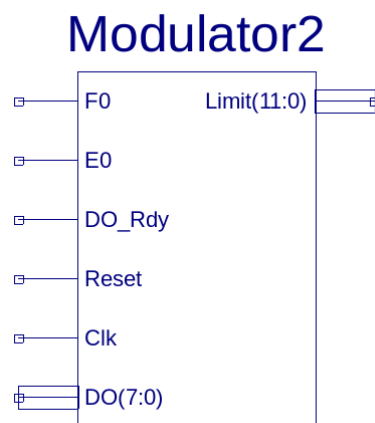
- schemat
 - DACWrite
 - Modulator2
 - PS2_Kdb
 - Synth2
 - Switch
 - RAM
 - LCD1x64
 - MsgGenerator
 - ADC_DAC.ucf
 - GenIO.ucf
 - LCD.ucf

2.2 Moduły

W tym podrozdziale znajdują się opisy i symulacje modułów stworzonych na zajęciach w ramach projektu.

2.2.1 Modulator2

Ten moduł odpowiedzialny jest za wysyłanie wartości ograniczającej licznik w module Synth2 na podstawie sygnału pochodzącego z klawiatury. Jest oparty na maszynie stanów, która składa się 14 stanów odpowiadających różnym dźwiękom w oktawie lub ciszy. Wartości, które zwraca moduł na wyjściu zostały obliczone w taki sposób, aby przy częstotliwości 50 MHz licznik w następnym module generował falę piłokształtną o odpowiedniej dla danego dźwięku i stanu częstotliwości.



Rysunek 2: Symbol modułu Modulator2.

Wejścia modułu:

- F0 - symbolizuje zwolnienie klawisza klawiatury
- E0 - symbolizuje czy dane to tzw. kod rozszerzony
- DO_Rdy - symbolizuje zakończenie odbierania kodu
- Clk - symbolizuje zegar
- DO[7:0] - symbolizuje otrzymane dane z klawiatury

Wyjścia modułu:

- limit[11:0] - symbolizuje wartość graniczną dla licznika

Fragmenty kodu VHDL

```
1 architecture Behavioral of Modulator2 is
2     type state_type is (Silence , C, Cis , D, Dis , E, F, Fis , G, Gis , A, Ais , H, C2);
3     signal state, next_state: state_type;
4 begin
5
6     process1 : process( Clk )
7     begin
8         if rising_edge( Clk ) then
9             if Reset = '1' then
10                 state <= Silence;
11             else
```

```

13         state <= next_state;
14     end if;
15 end process process1;

17 process2: process (state, DO, F0, DO_Rdy)
18 begin
19     next_state <= state;

21     if DO_Rdy = '1' and F0 = '0' then

23         case state is
24             when Silence =>
25                 if DO = X"1C" then
26                     next_state <= C;
27                 elsif DO = X"1D" then
28                     next_state <= Cis;
29                 elsif DO = X"1B" then
30                     next_state <= D;
31                 elsif DO = X"24" then
32                     next_state <= Dis;
33                 elsif DO = X"23" then
34                     next_state <= E;
35                 elsif DO = X"2B" then
36                     next_state <= F;
37                 elsif DO = X"2C" then
38                     next_state <= Fis;
39                 elsif DO = X"34" then
40                     next_state <= G;
41                 elsif DO = X"35" then
42                     next_state <= Gis;
43                 elsif DO = X"33" then
44                     next_state <= A;
45                 elsif DO = X"3C" then
46                     next_state <= Ais;
47                 elsif DO = X"3B" then
48                     next_state <= H;
49                 elsif DO = X"42" then
50                     next_state <= C2;
51                 end if;

53             when C =>
54                 next_state <= state;

55             when Cis =>
56                 next_state <= state;

57             when D =>
58                 next_state <= state;

59             when Dis =>
60                 next_state <= state;

61             when E =>
62                 next_state <= state;

63             when F =>
64                 next_state <= state;

65             when Fis =>
66                 next_state <= state;

67             when G =>
68                 next_state <= state;

69             when Gis =>
70                 next_state <= state;
71         end case;
72     end if;
73 end process process2;
74
75
77

```

```

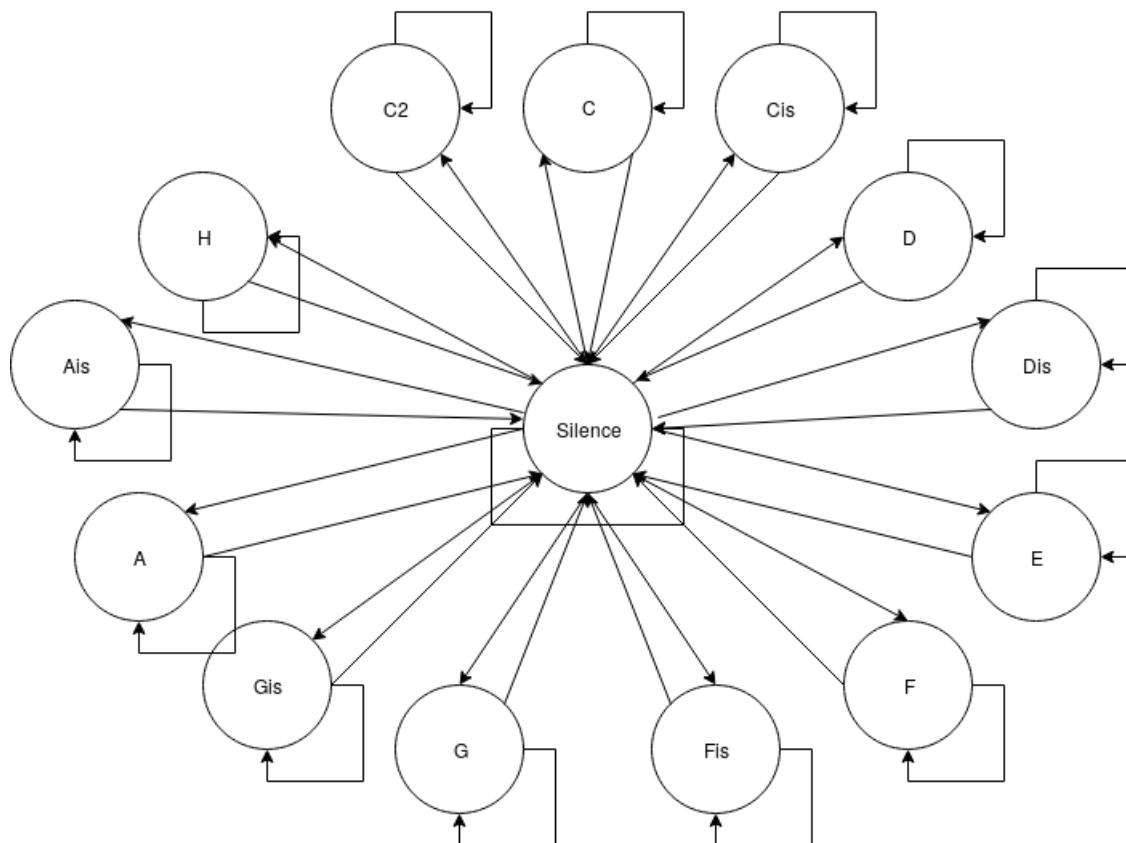
79         next_state <= state;
81     when A =>
82         next_state <= state;
83     when Ais =>
84         next_state <= state;
85     when H =>
86         next_state <= state;
87     when C2 =>
88         next_state <= state;
89     end case;
90     elsif F0 = '1' then
91         next_state <= Silence;
92     end if;
93 end process process2;
94
95 with state select
96     Limit <= X"5D4" when C,
97             X"581" when Cis ,
98             X"532" when D,
99             X"4E7" when Dis ,
100            X"4A1" when E,
101            X"45E" when F,
102            X"41F" when Fis ,
103            X"3E4" when G,
104            X"3AC" when Gis ,
105            X"377" when A,
106            X"345" when Ais ,
107            X"316" when H,
108            X"2EA" when C2,
109            X"000" when others;
110
111 end Behavioral;

```

Listing 1: Procesy modułu Modulator2.

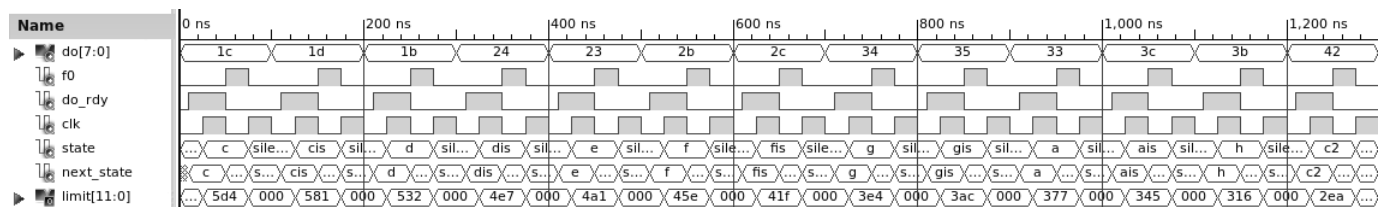
Architektura jednostki na samym początku posiada deklarację wszystkich stanów, które odpowiadają dźwiękom w oktawie lub ciszy. Pierwszy proces jest typowym przykładem procesu odpowiedzialnego za przełączanie stanu w maszynie stanów. Kolejny proces odpowiada za wybranie odpowiedniego stanu w zależności od wciśniętego klawisza. Ostatni element modułu to ustawianie odpowiedniego limitu w zależności od aktualnego stanu.

Graf maszyny stanu



Rysunek 3: Graf maszyny stanów modułu Modulator2.

Symulacja

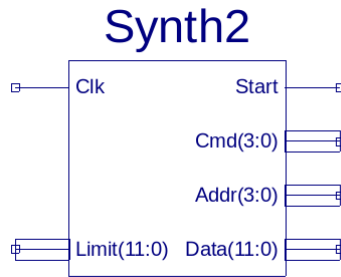


Rysunek 4: Wyniki symulacji modułu Modulator2.

Na powyższej symulacji widać jak w momencie impulsu sygnału do_rdy z najbliższym taktom zegara zmieniany jest stan maszyny zgodnie z wciśniętym klawiszem oraz ustawiany jest odpowiedni limit. W momencie impulsu f0 (puszczenie klawisza) stan przełączany jest na silence, a limit na wartość zerową.

2.2.2 Synth2

Moduł ten odpowiada za generowanie fali piłokształtnej o odpowiedniej częstotliwości. Częstotliwość ta zależy od wartości limitu, którą moduł otrzymuje na wejściu.



Rysunek 5: Symbol modułu Modulator2.

Wejścia modułu:

- Clk - symbolizuje zegar
- Limit[11:0] - symbolizuje limit dla wewnętrznego licznika

Wyjścia modułu:

- Start - symbolizuje gotowość danych do przetworzenia przez DAC
- Cmd - symbolizuje polecenie do wykonania
- Addr - symbolizuje adres przetwornika DAC
- Data[11:0] - symbolizuje dane do przetworzenia przez DAC

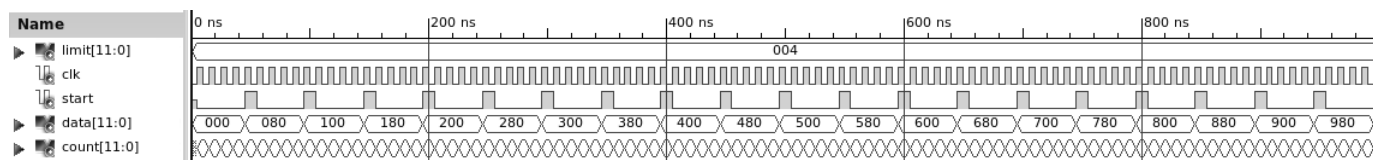
Fragmenty kodu VHDL

```
1 architecture Behavioral of Synth2 is
  signal count: UNSIGNED(11 downto 0) := X"000";
3  signal waveCount: UNSIGNED(4 downto 0) := X"0"&'0';
  signal lmt: STD_LOGIC_VECTOR(11 downto 0) := X"000";
5 begin
  lmt <= Limit;
7  process(Clk)
  begin
9    if rising_edge(Clk) then
      count <= count + 1;
11     start <= '0';
      if STD_LOGIC_VECTOR(count) = lmt then
13       count <= X"000";
       waveCount <= waveCount + 1;
15       start <= '1';
      end if;
17     end if;
  end process;
19
  Data <= STD_LOGIC_VECTOR(waveCount)&"000000";
21  Cmd <= "0011";
  Addr <= "1111";
23 end Behavioral;
```

Listing 2: Proces modułu Synth2.

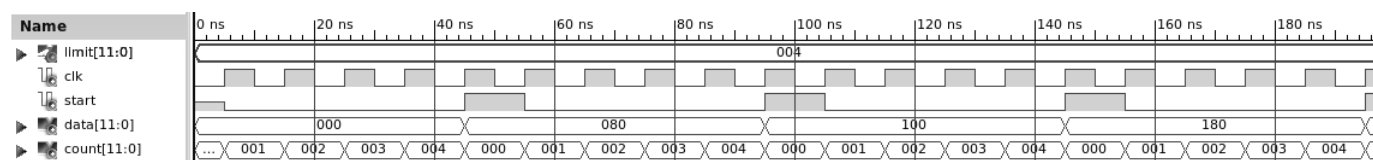
Moduł z każdym cyklem zegara zwiększa o jeden wartość wewnętrznego licznika, gdy licznik osiągnie limit zwiększany jest drugi licznik, który odpowiada generowanej fali. Wartość *waveCount* jest "doklejana" na początek wyjścia *Data*, aby osiągnąć większą amplitudę fali, czyli głośniejszy dźwięk. Komenda *Cmd* <= "0011" oznacza natychmiastowe zaktualizowanie wartości na wybranym przetworniku o wskazaną wartość w *Data*. Natomiast adres *Addr* <= "1111" oznacza wszystkie przetworniki na raz.

Symulacja



Rysunek 6: Wyniki symulacji modułu Synth2.

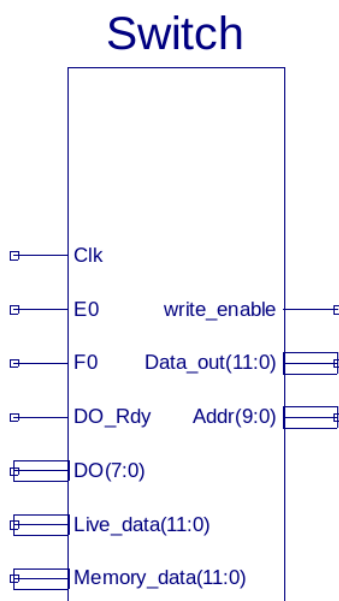
Symulacja została przeprowadzona dla *Limit* <= X"004", co oznacza, że co 4 cykle zegara wartość *waveCount* powinna się zwiększać, co za tym idzie wyjście *Data* powinno również się zwiększać.



Rysunek 7: Wyniki symulacji modułu Synth2 w powiększeniu.

2.2.3 Switch

Moduł ten jest odpowiedzialny za zarządzanie trybem pracy urządzenia. Na podstawie danych otrzymywanych z klawiatury decyduje, czy urządzenie powinno odtwarzać dźwięki według klawiszy zczytywanych z klawiatury, nagrywać dane do pamięci RAM, czy też odtwarzać melodię z pamięci. Jest oparty na prostej trójstanowej maszynie stanu: *None*, *Play*, *Rec*.



Rysunek 8: Symbol modułu Switch.

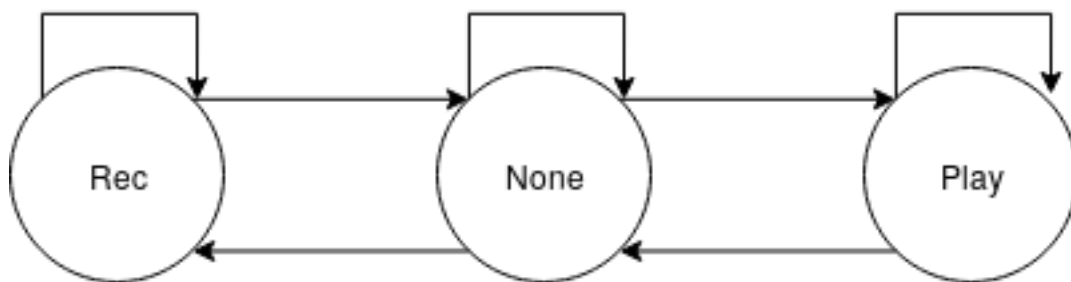
Wejścia modułu

- Clk - symbolizuje zegar
- F0 - symbolizuje zwolnienie klawisza klawiatury
- E0 - symbolizuje czy dane to tzw. kod rozszerzony
- DO_Rdy - symbolizuje zakończenie odbierania kodu
- DO[7:0] - symbolizuje otrzymane dane z klawiatury
- LiveData[11:0] - symbolizuje dane otrzymywane z modułu Synth2.
- MemoryData[11:0] - symbolizuje dane otrzymywane z modułu RAM.

Wyjścia modułu

- write_enable - symbolizuje pozwolenie na zapisywanie do pamięci RAM
- Data_out[11:0] - symbolizuje wychodzące dane wybrane spośród LiveData lub MemoryData
- Addr[11:0] - symbolizuje miejsce w pamięci RAM, z którego dane mają być wczytane lub do którego mają być zapisane.

Graf maszyny stanów



Rysunek 9: Graf maszyny stanów modułu Switch.

Rec oznacza nagrywanie, *Play* oznacza odtwarzania nagrania, a *None* oznacza odtwarzanie tylko dźwięków odczytanych z klawiatury.

Fragmenty kodu VHDL

```
architecture Behavioral of Switch is
2   type state_type is (None, Rec, Play);
   signal state, next_state: state_type;
4   signal TmpAddr: UNSIGNED(9 downto 0) := X"00"&"00";
   signal count: UNSIGNED(19 downto 0) := X"00000";
6
begin
8   process1 : process( Clk )
       begin
10      if rising_edge( Clk ) then
           state <= next_state;
12      end if;
       end process process1;
14
       process2: process (state, DO, F0, DO_Rdy)
16   begin
           next_state <= state;
```

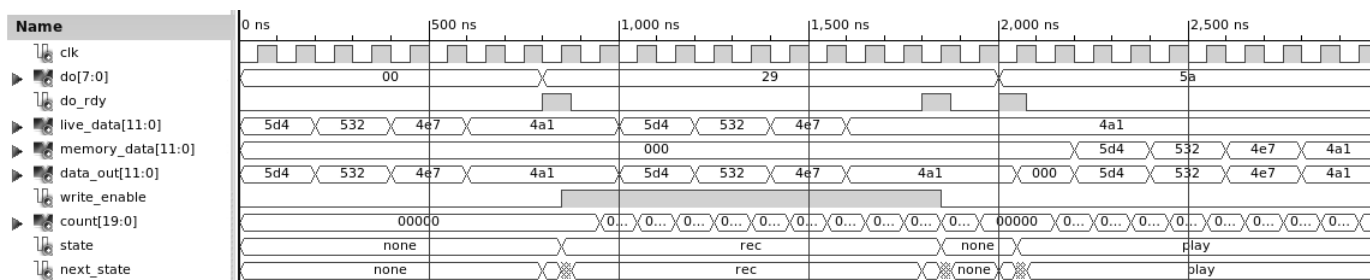
```

18      if DO_Rdy = '1' and F0 = '0' then
19          case state is
20              when None =>
21                  if DO = X"5A" then
22                      next_state <= Play;
23                  elsif DO = X"29" then
24                      next_state <= Rec;
25                  end if;
26
27              when Play =>
28                  if DO = X"5A" then
29                      next_state <= None;
30                  end if;
31
32              when Rec =>
33                  if DO = X"29" then
34                      next_state <= None;
35                  end if;
36          end case;
37      end if;
38  end process;
39
40  with state select
41      write_enable <= '1' when Rec,
42                    '0' when others;
43
44  with state select
45      Data_out <= Live_data when Rec,
46                Memory_data when Play,
47                Live_data when None;
48
49  Addr <= STD_LOGIC_VECTOR(TmpAddr);
50
51  process3: process (state, clk)
52  begin
53      if rising_edge(clk) then
54          if state = rec or state = play then
55              count <= count + 1;
56              if count = x"F4240" then
57                  TmpAddr <= TmpAddr + 1;
58                  count <= X"00000";
59              end if;
60          else
61              count <= X"00000";
62              TmpAddr <= X"00"&"00";
63          end if;
64      end if;
65  end process;
66  end Behavioral;

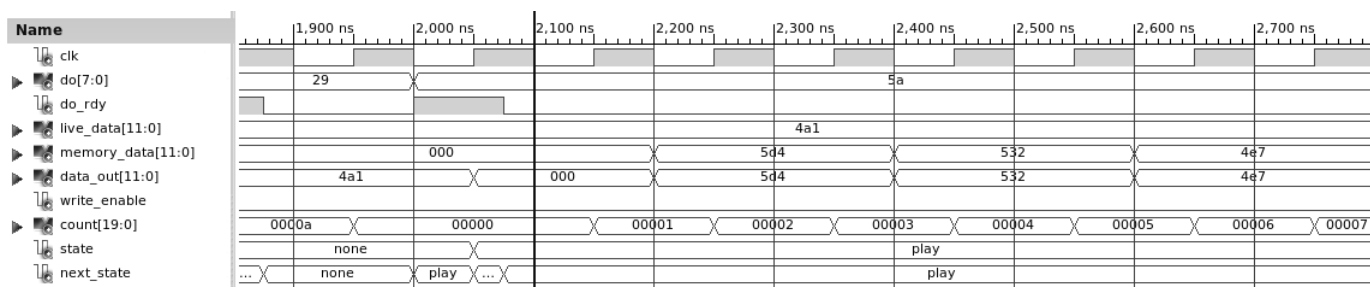
```

Pierwszy proces odpowiada za przechodzenie do następnego stanu. Drugi proces wybiera następny stan na podstawie wciśniętego klawisza. Moduł wybiera, który z *Live_data* lub *Memory_data* wysłać na wyjście w zależności od tego w jakim stanie się znajduje maszyna. Ostatni proces jest odpowiedzialny za przełączanie adresu co 0,02 sekundy w trakcie nagrywania lub odtwarzania nagrania oraz za zerowanie adresu w trybie *None*.

Symulacja



Rysunek 10: Symulacja modułu Switch cz.1.

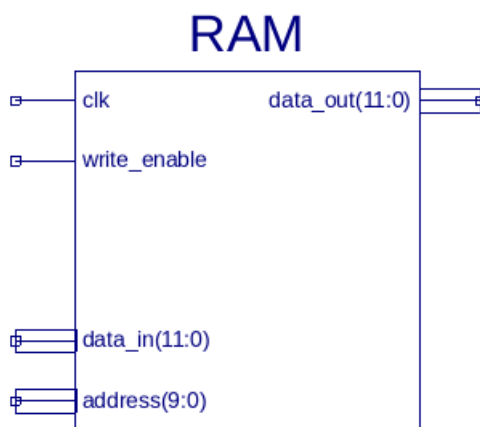


Rysunek 11: Symulacja modułu Switch cz.2.

Gdy stan maszyny jest w trybie *None* na wyjście są wysyłane dane z *LiveData* podobnie, w trybie *Rec* z tą różnicą, że w trybie nagrywania dodatkowo aktywny jest sygnał *write_enable*, który pozwala na zapisywanie danych do pamięci RAM. W stanie *Play* na wyjście modułu wysyłane są dane z wejścia *memory_data*, a *live_data* jest ignorowane. Dodatkowo w trakcie nagrywania i odtwarzania zwiększa się licznik. Gdy osiągnie on wartość $X''F4240''$ czyli 1000000, zmienia się adres zapisu lub odczytu z modułu RAM. Dzięki temu, przy częstotliwości 50 MHz, sygnał jest próbkowany 50 razy w ciągu sekundy, czyli co 0,02s.

2.2.4 RAM

Jest to pamięć, która pozwala na zapisywanie i czytanie danych.



Rysunek 12: Symbol modułu RAM.

Wejścia modułu

- clk - symbolizuje zegar
- write_enable - symbolizuje pozwolenie na zapisywanie do pamięci
- data_in[11:0] - symbolizuje dane do zapisania
- address[9:0] - symbolizuje adres z którego należy czytać, lub do którego należy zapisywać dane

Wyjścia modułu

- data_out[11:0] - symbolizuje dane wczytywane z pamięci.

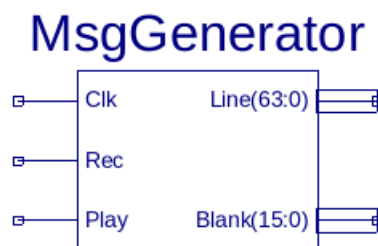
Fragmenty kodu VHDL

```
1 architecture Behavioral of RAM is
  type ram_type is array (0 to 1023) of std_logic_vector (11 downto 0);
3  signal memory : ram_type := (others => X"000");
begin
5  process (clk)
    begin
7      if (clk'event and clk = '1') then
          if (write_enable = '1') then
9              memory(conv_integer(address)) <= data_in;
              end if;
11             data_out <= memory(conv_integer(address));
            end if;
13  end process;
end Behavioral;
```

Moduł w zależności od wartości *write_enable* pozwala na zapisywanie lub wczytywanie z podanego na wejściu adresu.

2.2.5 MsgGenerator

Moduł ten odpowiada za wyświetlanie na ekranie wizualnej reprezentacji ile czasu pozostało do końca czasu nagrywania.



Rysunek 13: Symbol modułu MsgGenerator.

Wejścia modułu

- Clk - symbolizuje zegar
- Rec - symbolizuje tryb nagrywania
- Play - symbolizuje tryb odtwarzania

Wyjścia modułu

- Line[63:0] - symbolizuje dane wyjściowe na wyświetlaczu LCD
- Blank[15:0] - symbolizuje które znaki mają być wyświetlane

Fragmenty kodu VHDL

```
architecture Behavioral of MsgGenerator is
2   signal counter: UNSIGNED(27 downto 0) := X"0000000";
   signal show: UNSIGNED(3 downto 0) := X"0";
4 begin
   Line <= x"0000000000000000";
6   process (Clk)
   begin
8       if rising_edge(Clk) then
           if Rec = '1' then
10              counter <= counter + 1;
                  if counter = x"3d09000" then
12                      show <= show +1;
                          counter <= X"0000000";
14                      end if;
                  else
16                      counter <= X"0000000";
                          show <= X"0";
18                      end if;
                  end if;
20   end process;

22   with show select
       Blank <= X"0001" when X"0",
24              X"0003" when X"1",
                  X"0007" when X"2",
26              X"000F" when X"3",
                  X"001F" when X"4",
28              X"003F" when X"5",
                  X"007F" when X"6",
30              X"00FF" when X"7",
                  X"01FF" when X"8",
32              X"03FF" when X"9",
                  X"07FF" when X"A",
34              X"0FFF" when X"B",
                  X"1FFF" when X"C",
36              X"3FFF" when X"D",
                  X"7FFF" when X"E",
38              X"FFFF" when X"F",
                  X"0000" when others;
40 end Behavioral;
```

Listing 3: Procesy modułu MsgGenerator.

Moduł wyświetla tylko same zera, jednak z bieżącym czasem kolejne zera znikają, co symbolizuje kończący się czas nagrywania.

3 Implementacja

3.1 Zasoby

3.2 "User manual" urządzenia

4 Podsumowanie

Zadanie udało się zrealizować w całości. Instrument jest w pełni działający, a ze względu na jasny podział na moduły można bez trudu dopisywać do niego kolejne funkcjonalności. Także sama wartość

”merytoryczna” keyboarda nie pozostawia wiele do życzenia, ponieważ faktycznie pokrywa on całą oktawę, a wysokości dźwięków różnią się od siebie dokładnie tak jak w prawdziwym instrumencie, dzięki czemu mając nuty do utworu muzycznego możemy go zagrać.

5 Literatura