Politechnika Wrocławska

Układy cyfrowe i systemy wbudowane

Dokumentacja projektu
Organy z możliwością zapisywania i odtarzania melodii.

Autorzy:
Łukasz Bieszczad
Krzysztof Buczak

Prowadzący:
dr inż. Jarosław Sugier

18 maja 2018

Spis treści

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 nagranego materiału, a także wykorzystanie wyświetlacza LCD do pokazania stanu nagrywania i diody LED do przekazania informacji o trwającym właśnie nagrywaniu.

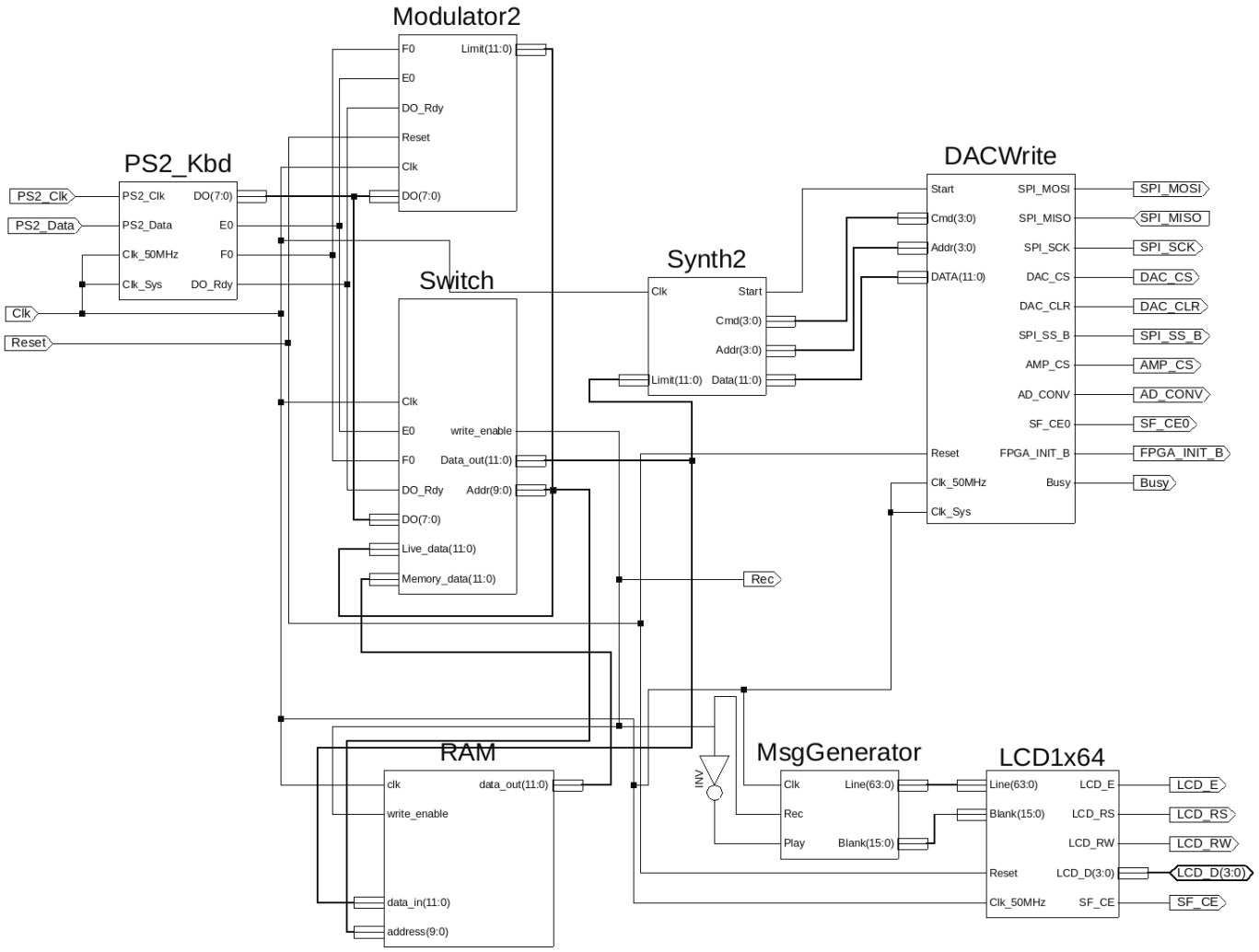
1.2 Zagadnienia teoretyczne

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:

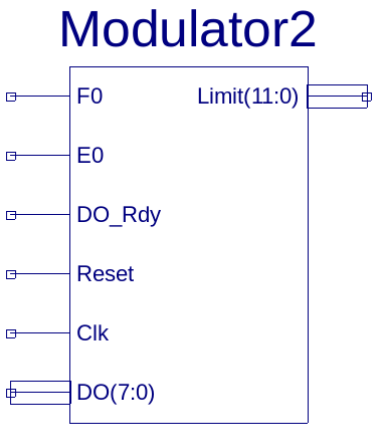
- schemat
  - DACWrite
  - Modulator2
  - PS2_Kdb
  - Synth2
  - Switch
  - RAM
  - LCD1x64
  - MsgGenerator
  - ADC_DAC.ucf
  - GenIO.ucf
  - LCD.ucf

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

```
architecture Behavioral of Modulator2 is
   type state_type is (Silence, C, Cis, D, Dis, E, F, Fis, G, Gis, A, Ais, H, C2);
   signal state, next_state: state_type;
begin

   process1 : process( Clk )
   begin
         if rising_edge( Clk ) then
            if Reset = '1' then
               state <= Silence;
            else
```

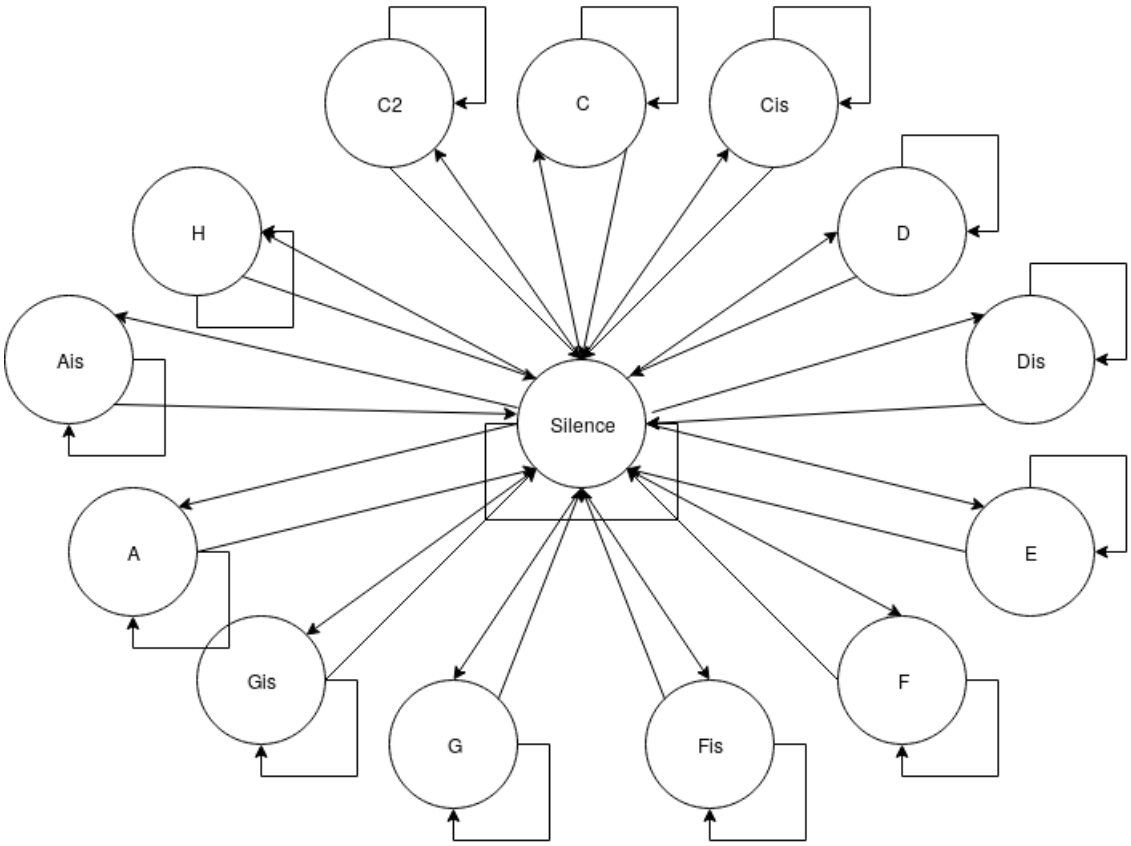
```
                state <= next_state;
            end if;
        end if;
    end process process1;

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

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

            case state is
                when Silence =>
                    if DO = X"1C" then
                        next_state <= C;
                    elsif DO = X"1D" then
                        next_state <= Cis;
                    elsif DO = X"1B" then
                        next_state <= D;
                    elsif DO = X"24" then
                        next_state <= Dis;
                    elsif DO = X"23" then
                        next_state <= E;
                    elsif DO = X"2B" then
                        next_state <= F;
                    elsif DO = X"2C" then
                        next_state <= Fis;
                    elsif DO = X"34" then
                        next_state <= G;
                    elsif DO = X"35" then
                        next_state <= Gis;
                    elsif DO = X"33" then
                        next_state <= A;
                    elsif DO = X"3C" then
                        next_state <= Ais;
                    elsif DO = X"3B" then
                        next_state <= H;
                    elsif DO = X"42" then
                        next_state <= C2;
                    end if;

                when C =>
                    next_state <= state;

                when Cis =>
                    next_state <= state;

                when D =>
                    next_state <= state;

                when Dis =>
                    next_state <= state;

                when E =>
                    next_state <= state;

                when F =>
                    next_state <= state;

                when Fis =>
                    next_state <= state;

                when G =>
                    next_state <= state;

                when Gis =>
```

```
                next_state <= state;

            when A =>
                next_state <= state;

            when Ais =>
                next_state <= state;

            when H =>
                next_state <= state;

            when C2 =>
                next_state <= state;
        end case;
    elsif F0 = '1' then
        next_state <= Silence;
    end if;
end process process2;

with state select
    Limit <= X"5D4" when C,
             X"581" when Cis,
             X"532" when D,
             X"4E7" when Dis,
             X"4A1" when E,
             X"45E" when F,
             X"41F" when Fis,
             X"3E4" when G,
             X"3AC" when Gis,
             X"377" when A,
             X"345" when Ais,
             X"316" when H,
             X"2EA" when C2,
             X"000" when others;

end Behavioral;
```

Listing 1: Procesy modułu Modulator2.

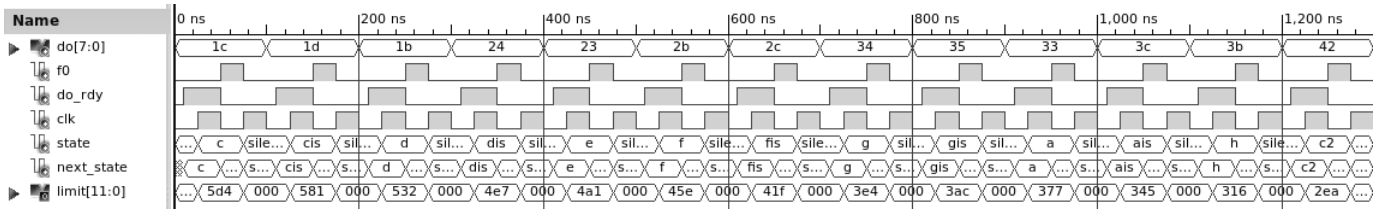
Architektura jednostki na samym początku posiada deklarację wszystkich stanów, które odpowiadają dźwiękom w oktawie lub ciszy. Pierwszy proces jest typowym przykładem procesu odpowiedzialnego za przełączanie stanu w maszynie stanów. Kolejny proces odpowiada za wybranie odpowiedniego stanu w zależności 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

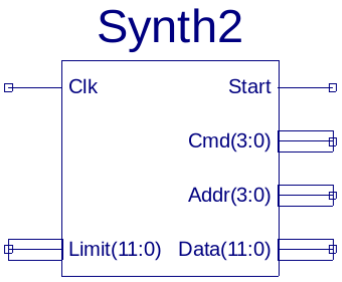


Rysunek 4: Wyniki symulacji modułu Modulator2.

Na powyższej symulacji widać jak w momencie impulsu sygnału do_rdy z najbliższym taktem zegara zmieniany jest stan maszyny zgodnie z wciśniętym klawiszem oraz ustawiany jest odpowieni 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

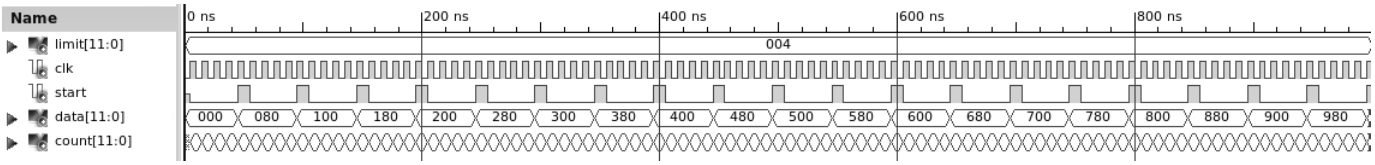
Fragmenty kodu VHDL

```
architecture Behavioral of Synth2 is
    signal count: UNSIGNED(11 downto 0) := X"000";
    signal waveCount: UNSIGNED(4 downto 0):= X"0"&'0';
    signal lmt: STD_LOGIC_VECTOR(11 downto 0) := X"000";
begin
    lmt <= Limit;
    process(Clk)
    begin
        if rising_edge(Clk) then
            count <= count + 1;
            start <= '0';
            if STD_LOGIC_VECTOR(count) = lmt then
                count <= X"000";
                waveCount <= waveCount + 1;
                start <= '1';
            end if;
        end if;
    end process;

    Data <= STD_LOGIC_VECTOR(waveCount)&"0000000";
    Cmd <= "0011";
    Addr <= "1111";
end Behavioral;
```

Listing 2: Proces modułu Synth2.

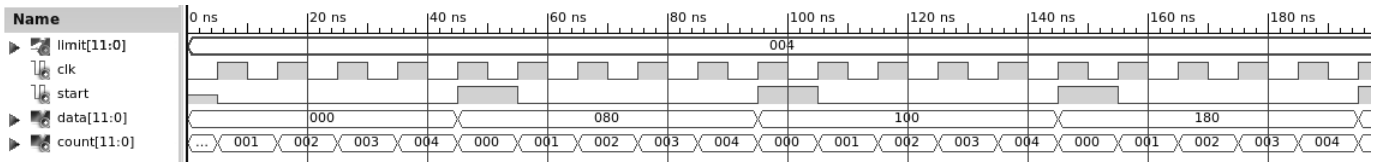
Moduł z każdym cyklem zegara zwiększa o jeden wartość wewnętrznego licznika, gdy licznik osiągnie limit zwiększany jest drugi licznik, który odpowiada generowanej fali. Wartość waveCount jest "doklejana" na początek wyjścia Data, aby osiągnąć większą amplitudę fali, czyli głośniejszy dźwięk. Komenda $Cmd <= "0011"$ oznacza natchmiastowe 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.

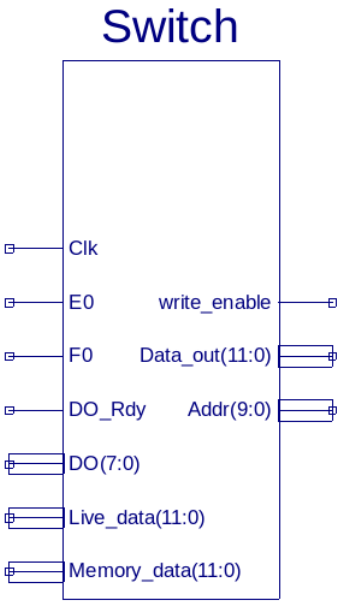
Symulacja została przeprowadzona dla $Limit <= X"004"$, co oznacza, że co 4 cykle zegara wartość *waveCount* powinna się zwiększać, co za tym idzie wyjście *Data* powinno również się zwiększać.



Rysunek 7: Wyniki symulacji modułu Synth2 w powiększeniu.

2.2.3 Switch

Moduł ten jest odpowiedzialny za zarządzanie trybem pracy urządzenia. Na podstawie danych otrzymywany 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.

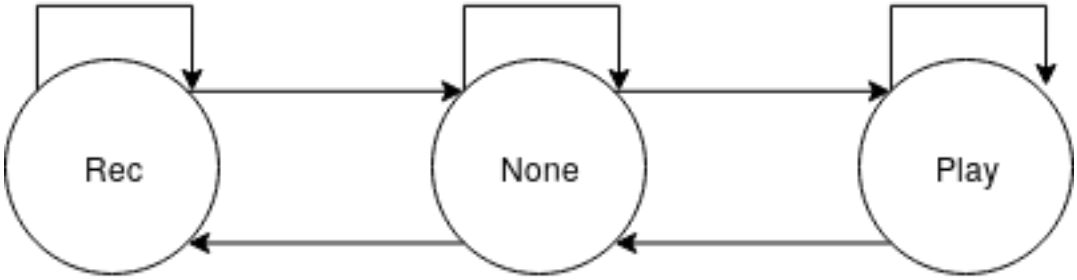
Wejścia modułu

- 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 pozowlenie 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.

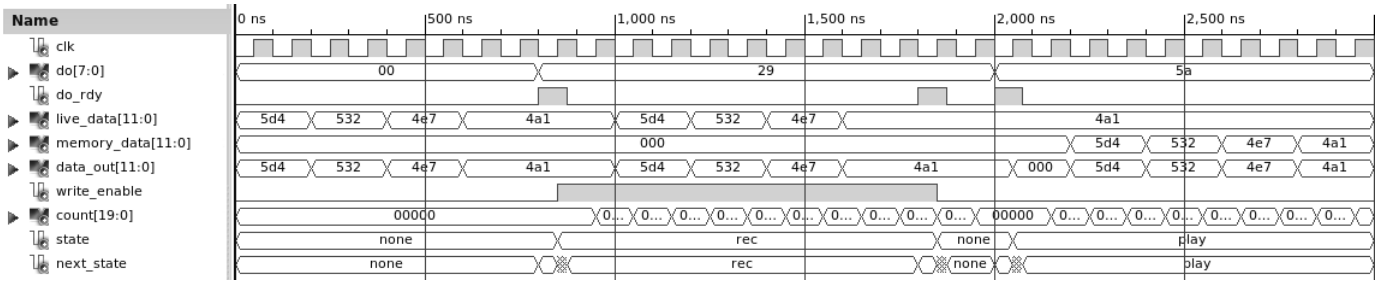
Fragmenty kodu VHDL

```
architecture Behavioral of Switch is
   type state_type is (None, Rec, Play);
   signal state, next_state: state_type;
   signal TmpAddr: UNSIGNED(9 downto 0) := X"00"&"00";
   signal count: UNSIGNED(19 downto 0) := X"00000";

begin
   process1 : process( Clk )
      begin
         if rising_edge( Clk ) then
            state <= next_state;
         end if;
   end process process1;

   process2: process (state, DO, F0, DO_Rdy)
   begin
      next_state <= state;
```

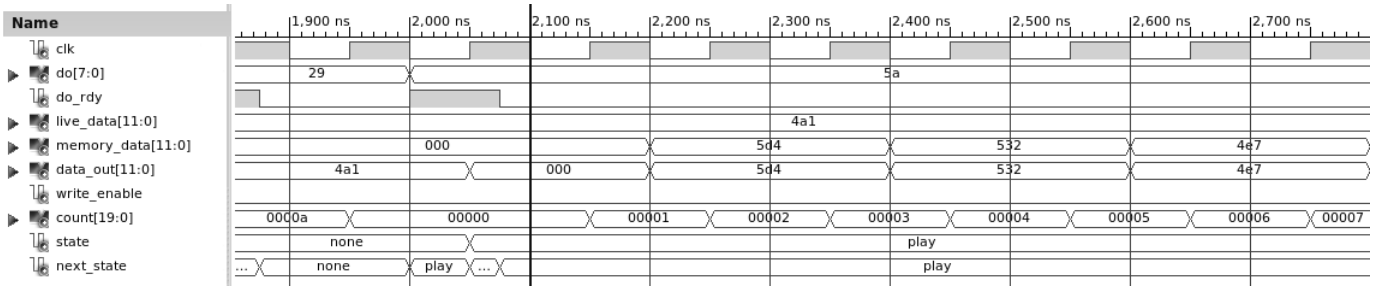
```
18
        if DO_Rdy = '1' and F0 = '0' then
20          case state is
                when None =>
22                  if DO = X"5A" then
                        next_state <= Play;
24                  elsif DO = X"29" then
                        next_state <= Rec;
26                  end if;

28              when Play =>
                    if DO = X"5A" then
30                      next_state <= None;
                    end if;
32
                when Rec =>
34                  if DO = X"29" then
                        next_state <= None;
36                  end if;
            end case;
38      end if;
    end process;
40
    with state select
42      write_enable <= '1' when Rec,
                        '0' when others;
44
    with state select
46      Data_out <= Live_data when Rec,
                    Memory_data when Play,
48                  Live_data when None;

50  Addr <= STD_LOGIC_VECTOR(TmpAddr);

52  process3: process (state, clk)
    begin
54      if rising_edge(clk) then
            if state = rec or state = play then
56              count <= count + 1;
                if count = x"F4240" then
58                  TmpAddr <= TmpAddr + 1;
                    count <=  X"00000";
60              end if;
            else
62              count <= X"00000";
                TmpAddr <= X"00"&"00";
64          end if;
        end if;
66  end process;
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 wejscia $memory_data$, 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

2.2.5 MsgGenerator

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