Układy cyfrowe i systemy wbudowane 2 - projekt

Prowadzący: mgr. Antioni Sterna

VHDL - Organki

Michał Polak 235046
Paweł Tomków 236035

30 maja 2019
LaTeX

Spis treści

1 Wstęp

1.1 Cel i zakres projektu

Celem projektu było stworzenie programu w języku VHDL, który symulowałby działanie tak zwanych popularnych "Organek". Program ten umożliwiał odtwarzanie dźwięku po wciśnięciu klawisza na klawiaturze PS2. Każdy z 10 klawszów ma zamisane wewnątrz kodu jaki dźwięk będzie z niego odtwarzany za pomocą modułu DACWrite.

1.2 Etapy projektu

- Wygenerowanie sygnału prostokątnego
- Wygenerowanie dźwięku "piły"
- Stworzenie modułu generującego oba te sygnały
- Dodanie obsługi klawiatury PS2
- Połączenie obsługi klawiatury i generowania sygnałów tak aby cały projekt działał.

1.3 Zasoby sprzętowe

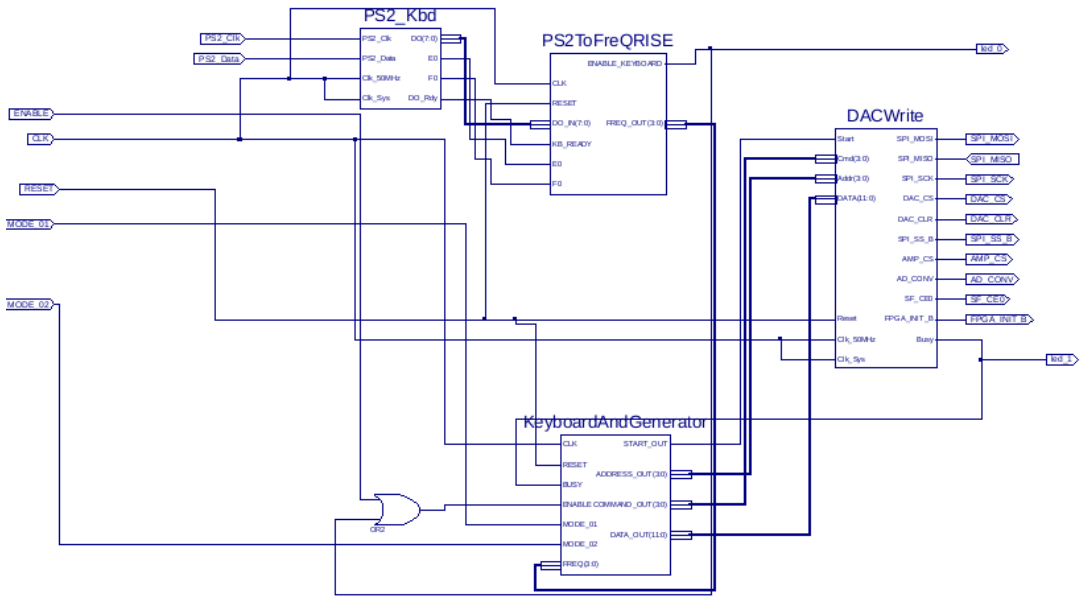
W naszym projekcie korzystaliśmy z platformy sprzętowej: „Spartan3E Starter Kit" oraz klawiatury komputerowej podłączonej do zestawu na porcie PS2. Pełna specyfiacja modułów oraz dokładny opis wykorzystanych zasobów sprzętowych znajduje się w dokumentacji.

1.4 Wykorzystane interfejsy i protokoły

Do interakcji z naszym układem wykorzystaliśmy klawiaturę komputerową podpietą na porcie PS2. Oraz głośnik który otrzymaliśmy od prowadzącego.

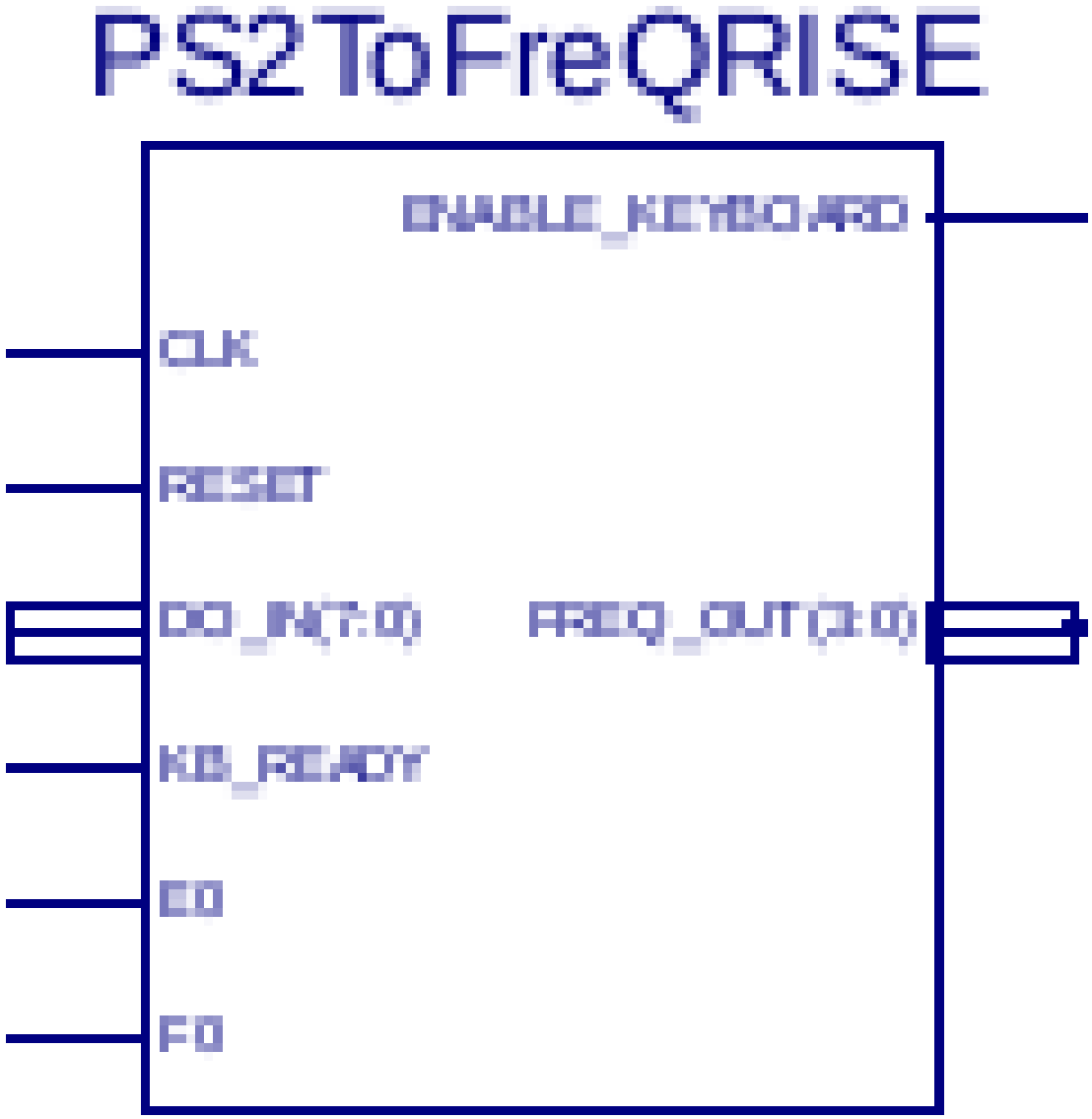
2 Projekt

2.1 Schemat



Obraz 1 - Schemat układu.

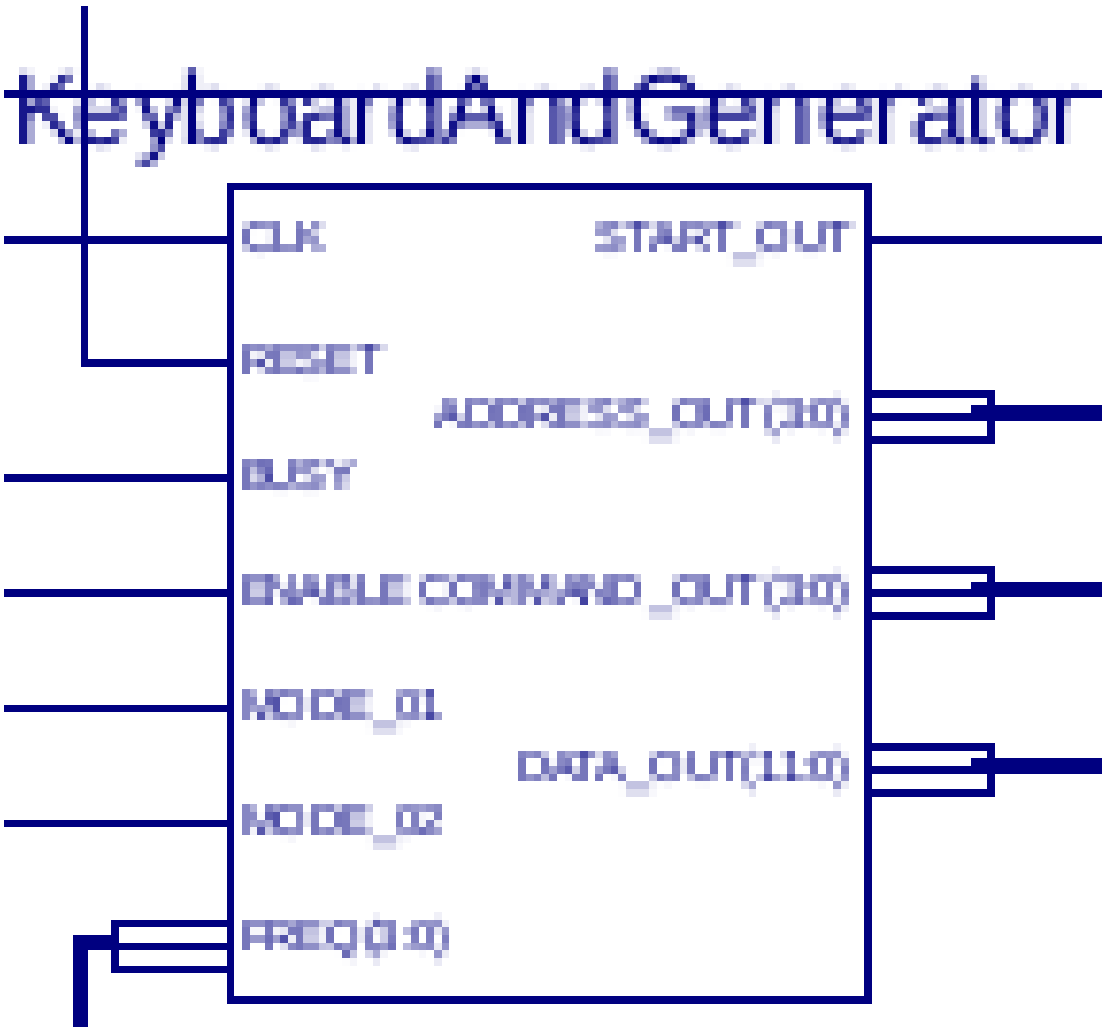
2.2 Moduły



Obraz 2 - Moduł konwertujący sygnał z PS2.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity PS2ConvToFreQ is

    PORT(
         DO_IN      : in std_logic_vector (7 downto 0);
         EO         : in std_logic;
         FO         : in std_logic;
         DO_Rdy_IN  : in std_logic;
         FREQ_OUT   : out std_logic_vector (3 downto 0);
         ENABLE_KEYBOARD : out std_logic
        );

end PS2ConvToFreQ;
```

```
architecture Behavioral of PS2ConvToFreQ is

begin

   with DO_IN & E0 & F0 select

      FREQ_OUT <= "0000" when X"15" & "00", -- Q // C
                  "0001" when X"1E" & "00", -- 2 // C#
                  "0010" when X"1D" & "00", -- W // D
                  "0011" when X"26" & "00", -- 3 // D#
                  "0100" when X"24" & "00", -- E // E
                  "0101" when X"2D" & "00", -- R // F
                  "0110" when X"2E" & "00", -- 5 // F#
                  "0111" when X"2C" & "00", -- T // G
                  "1000" when X"36" & "00", -- 6 // G#
                  "1001" when X"35" & "00", -- Y // A
                  "1010" when X"3D" & "00", -- 7 // A#
                  "1011" when X"3C" & "00", -- U // B
                  "1111" when others;

   with DO_IN & E0 & F0 select

      ENABLE_KEYBOARD <=  '1' when X"15" & "00", -- Q // C
                  '1' when X"1E" & "00", -- 2 // C#
                  '1' when X"1D" & "00", -- W // D
                  '1' when X"26" & "00", -- 3 // D#
                  '1' when X"24" & "00", -- E // E
                  '1' when X"2D" & "00", -- R // F
                  '1' when X"2E" & "00", -- 5 // F#
                  '1' when X"2C" & "00", -- T // G
                  '1' when X"36" & "00", -- 6 // G#
                  '1' when X"35" & "00", -- Y // A
                  '1' when X"3D" & "00", -- 7 // A#
                  '1' when X"3C" & "00",
                  '0' when others;

end Behavioral;
```



Obraz 3 - Moduł generujący sygnał dla DACWrite.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_arith.all;
use IEEE.NUMERIC_STD.ALL;

entity KeyboardAndGenerator is

   Port (CLK : in STD_LOGIC;
         RESET       : in STD_LOGIC;
         BUSY        : in STD_LOGIC;
         ENABLE      : in STD_LOGIC;
         MODE_01     : in STD_LOGIC;
         MODE_02     : in STD_LOGIC;
         FREQ        : in STD_LOGIC_VECTOR (3 downto 0);
         START_OUT   : out STD_LOGIC;
         ADDRESS_OUT : out STD_LOGIC_VECTOR (3 downto 0);
         COMMAND_OUT : out STD_LOGIC_VECTOR (3 downto 0);
         DATA_OUT    : out STD_LOGIC_VECTOR (11 downto 0)
        );

end KeyboardAndGenerator;
```

```
architecture Behavioral of KeyboardAndGenerator is
   constant MAX_DELAY_RECTANGLE : positive := 600000;
   constant MAX_DELAY_SAW : positive := 100;
   constant MAX_DELAY_DIFFERENT_SOUNDS : positive := 1500;
   subtype t_dac_data is integer range 0 to 4095;
   subtype t_counter_rectangle is integer range 0 to MAX_DELAY_RECTANGLE;
   subtype t_counter_saw is integer range 0 to MAX_DELAY_SAW;
   subtype t_counter_different_sounds is integer range 0 to MAX_DELAY_DIFFERENT_SOUNDS;

   type t_state is (s_init, s_start, s_wait);

   constant DELAY_RECTANGLE : t_counter_rectangle := 25000;
   constant DELAY_SAW : t_counter_saw := 50;
   signal DELAY_KEYBOARD: t_counter_different_sounds := 50;

   constant ADDRESS : std_logic_vector(3 downto 0) := x"F"; -- All registers
   constant COMMAND : std_logic_vector(3 downto 0) := x"2"; -- Write to and Update (Pow
   signal   DATA    : t_dac_data := 0;
   signal   FREQ_signal: std_logic_vector (3 downto 0) := x"0";
   signal iBeepGen: UNSIGNED (4 downto 0) := "00000";

   signal counter_rectangle : t_counter_rectangle;
   signal counter_saw : t_counter_saw;
   signal counter_sound : t_counter_different_sounds;
   signal state : t_state;

begin

SetFreq: process( FREQ, DELAY_KEYBOARD)

begin
   case FREQ is
      when "0000" =>      DELAY_KEYBOARD <= 1494;
      when "0001" =>      DELAY_KEYBOARD <= 1410;
      when "0010" =>      DELAY_KEYBOARD <= 1330;
      when "0011" =>      DELAY_KEYBOARD <= 1256;
      when "0100" =>      DELAY_KEYBOARD <= 1186;
      when "0101" =>      DELAY_KEYBOARD <= 1119;
      when "0110" =>      DELAY_KEYBOARD <= 1056;
      when "0111" =>      DELAY_KEYBOARD <= 997;
      when "1000" =>      DELAY_KEYBOARD <= 940;
      when "1001" =>      DELAY_KEYBOARD <= 887;
      when "1010" =>      DELAY_KEYBOARD <= 838;
      when "1011" =>      DELAY_KEYBOARD <= 791;
      when others =>      DELAY_KEYBOARD <= 0;
   end case;

end process;

Generate_signal : process (CLK, RESET, FREQ, DELAY_KEYBOARD)
```

```
begin
 if RESET = '1' then
   counter_rectangle <= 0;
   counter_saw <= 0;
   DATA <= 0;
   START_OUT <= '0';
   state <= s_init;
 elsif rising_edge(CLK) then
   if MODE_01 = '0' and MODE_02 = '1' then
     case state is
       when s_init     => if BUSY = '0' then
                            START_OUT <= '1';
                            state <= s_start;
                          end if;

       when s_start    => START_OUT <= '0';
                          counter_rectangle <= DELAY_RECTANGLE;
                          state <= s_wait;

       when s_wait     => if counter_rectangle = 0 then
                            if DATA = 0 then
                              DATA <= 2500;   -- 4095
                            else
                              DATA <= 0;
                            end if;
                            state <= s_init;
                          end if;
     end case;

      if counter_rectangle > 0 then
         counter_rectangle <= counter_rectangle - 1;
      end if;

  DATA_OUT <= std_logic_vector(to_unsigned(DATA, 12));

   elsif MODE_01 = '1' and MODE_02 = '0' then
     case state is
       when s_init     => if BUSY = '0' then
                            START_OUT <= '1';
                            state <= s_start;
                          end if;

       when s_start    => START_OUT <= '0';
                          counter_saw <= DELAY_SAW;
                          state <= s_wait;

       when s_wait     => if counter_saw = 0 then
                            if DATA > 4000 then
                               DATA <= 0;
```

```
                        else
                            DATA <= DATA + 4;   -- 4095
                        end if;
                        state <= s_init;
                    end if;
    end case;

    if counter_saw > 0 then
     counter_saw <= counter_saw - 1;

    end if;

  DATA_OUT <= std_logic_vector(to_unsigned(DATA, 12));

   elsif MODE_01 = '1' and MODE_02 = '1' and ENABLE = '1' then
        case state is
       when s_init     => if BUSY = '0' or DELAY_KEYBOARD = 0 then
                          START_OUT <= '1';
                          state <= s_start;
                        end if;

       when s_start    => START_OUT <= '0';
                        counter_sound <= DELAY_KEYBOARD;
                        state <= s_wait;

       when s_wait     => if counter_sound = 0 then
                            iBeepGen <= iBeepGen + 1;
                          state <= s_init;
                        end if;
    end case;

    if counter_sound > 0 then
     counter_sound <= counter_sound - 1;
    end if;

  DATA_OUT <= std_logic_vector(iBeepGen & "0000000");

   end if;
 end if;

end process;

   ADDRESS_OUT <= ADDRESS;
   COMMAND_OUT <= COMMAND;

end Behavioral;
```

2.3 Symulacje

screeny z symulacji

3 Implementacja

3.1 Rozmiar

3.2 Szybkość

3.3 Użytkowanie

4 Podsumowanie

4.1 Ocena krytyczna

Udało nam się zrealizować większość założonych przez nas funkcjonalności. Projekt nie został jednak ukończony w całości z powodu ograniczonego czasu, problemów związanych z wiedzą oraz doświadczeniem w pracy z VHDL-em oraz złego oszacowania zasobów. Jedna z funkcjonalności na którą zabrakło nam czasu było zaimplementowanie własnego modułu DACWrite. Największe problemy pojawiły się podaczs generowanie pierwszego sygnału, tak aby był słyszalny przez głośnik. Na początku prac z projektem posiadaliśmy wiedzę zdobytą na laboratorium przedmiotu Urządzenia Cyfrowe i Systemy Wbudowane 1 jednak nie okazała się ona wystarcza jąca do zrealizowania naszych założeń. Wiele problemów sprawiła także naprawa ostrzeżeń 737, które p o jawiały się przy praktycznie każdej zmianie ko du. Myślimy, że praca nad tym projektem poszerzyła bardzo naszą wiedzę związaną z VHDL-em oraz pracy z układami typu Spartan. Jesteśmy częściowo zadowoleni z rezultatu prac z powodu wielu problemów w trakcie trwania projektu, jednak żałujemy, iż nie udało nam się zrealizować projektu w całości.

4.2 Otwarość na rozszrzenia

Projekt pozosta je otwarty na możliwość rozwoju oraz dodawania nowych funkcjonalności. Przede wszystkim można zaimplementować wyświetlanie klawiszy pianina na ekranie. Podczas naciśnięcia klawisza na klawiaturze nastąpiła by animancja wciskanego klawisza pianina wyświetlanego na monitorze - jest to jedno z trudniejszych zagadnień.

5 Literatura

- Spartan-3E FPGA Starter Kit Board User Guide, http://www.zsk.ict.pwr.wroc.pl/zskftp/fpga.
- Prezentacja dotycząca języka VHDL dr.inz. J. Sugier http://www.z sk.ict.pwr.wroc.pl/zs k/repository/dydaktyka /uc/ucsw1_vhdl.pdf
- Dokumentacja VHDL https://www. ics.uci.edu/ jmoorkan/vhdlre f/Synario%20VHDL%20Man ual.pdf/