Układy Cyfrowe i Systemy Wbudowane
Projekt

Temat: Organy

Wykonanie:
Zuzanna Bołtuć 236755
Łukasz Szumilas 236068

Prowadzący:
dr inż. Jarosław Sugier

Termin:
Poniedziałek, TN, godz. 11:00

Data oddania:
03 czerwca 2019

Spis treści

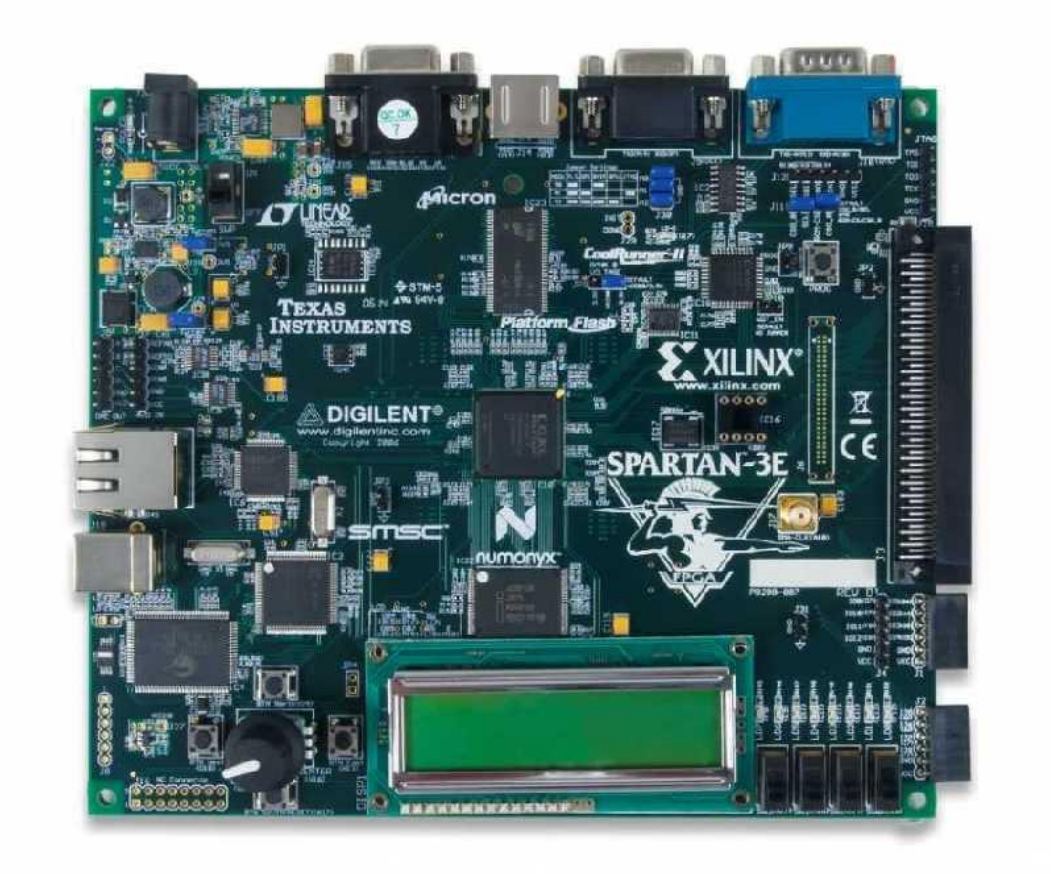
1. Wprowadzenie

1.1 Cel i zakres

Celem projektu było stworzenie organów wydających dźwięki za pomocą głośnika podpiętego do układu Spartan 3E, odgrywanie pojedynczych dźwięków za pomocą klawiszy klawiatury i ich wizualizacja na monitorze posiadającym kartę graficzną VGA (Video Graphics Array). By móc określić, które dźwięki są aktualnie wykonywane, obraz podświetlał odpowiednie klawisze. Zakres dźwięków obejmował jedną oktawę, składającą się na 12 dźwięków w tejże oktawie plus dźwięk bazowy o ton wyżej.

1.2 Sprzęt

Projekt był realizowany na gotowym, niepotrzebującym montażu ani modyfikacji układzie programowalnym FPGA Spartan-3E.

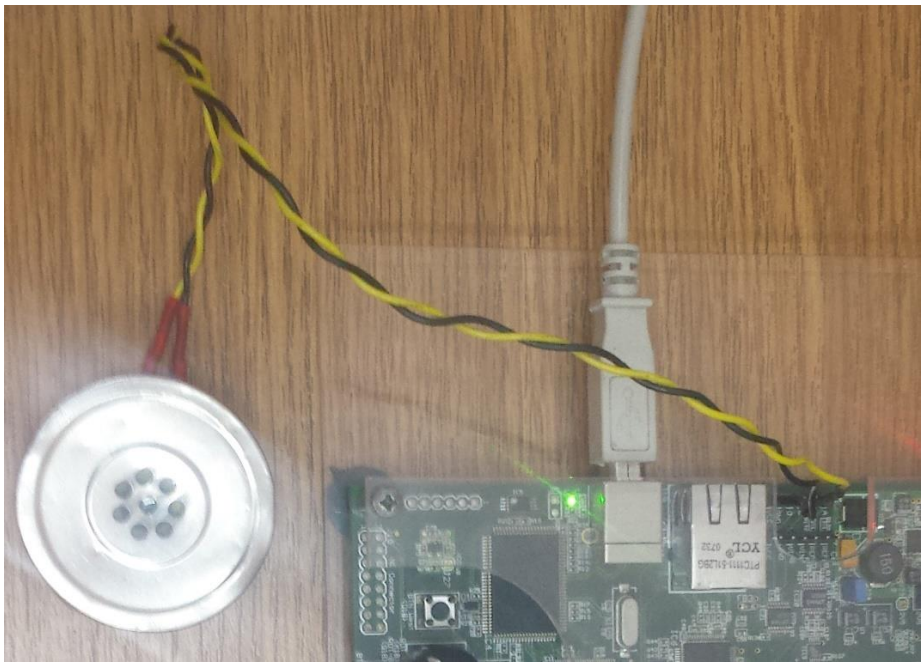


Rysunek 1 – Układ Spartan-3E

Obraz generowany był dzięki monitorowi VGA Samsung SyncMaster203b.

Rysunek 2 – monitor VGA

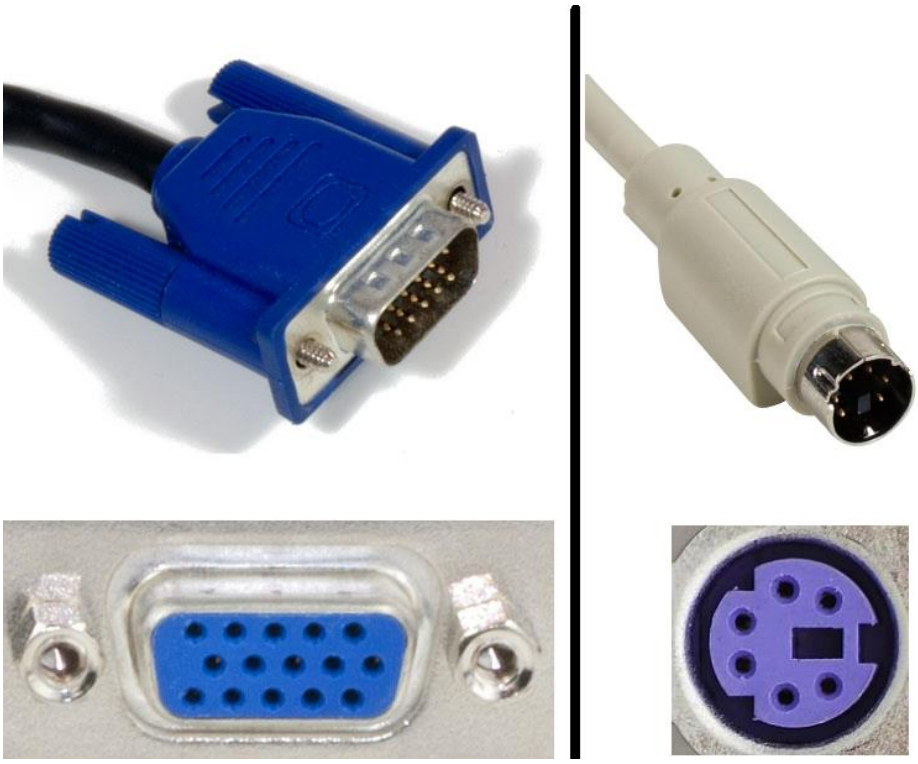
Dźwięki wydobywały się z prostego głośniczka z dwoma przewodami podpiętymi do układu Spartan. Jeden przewód odpowiadał za zasilanie. Drugi pobierał z układu dane, które przetwarzał na fale akustyczne na membranie dzięki przetwornikowi, tworząc odpowiednie nuty.



Rysunek 3 – głośnik podpięty do układu

Granie na organach odbywało się za pomocą standardowej klawiatury komunikującej się z układem poprzez port szeregowy PS/2

Rysunek 4 – klawiatura PS/2



Rysunek 5 – wejście/wyjście: VGA | PS/2

1.3 Zagadnienia teoretyczne

1.3.1 Generowanie dźwięków

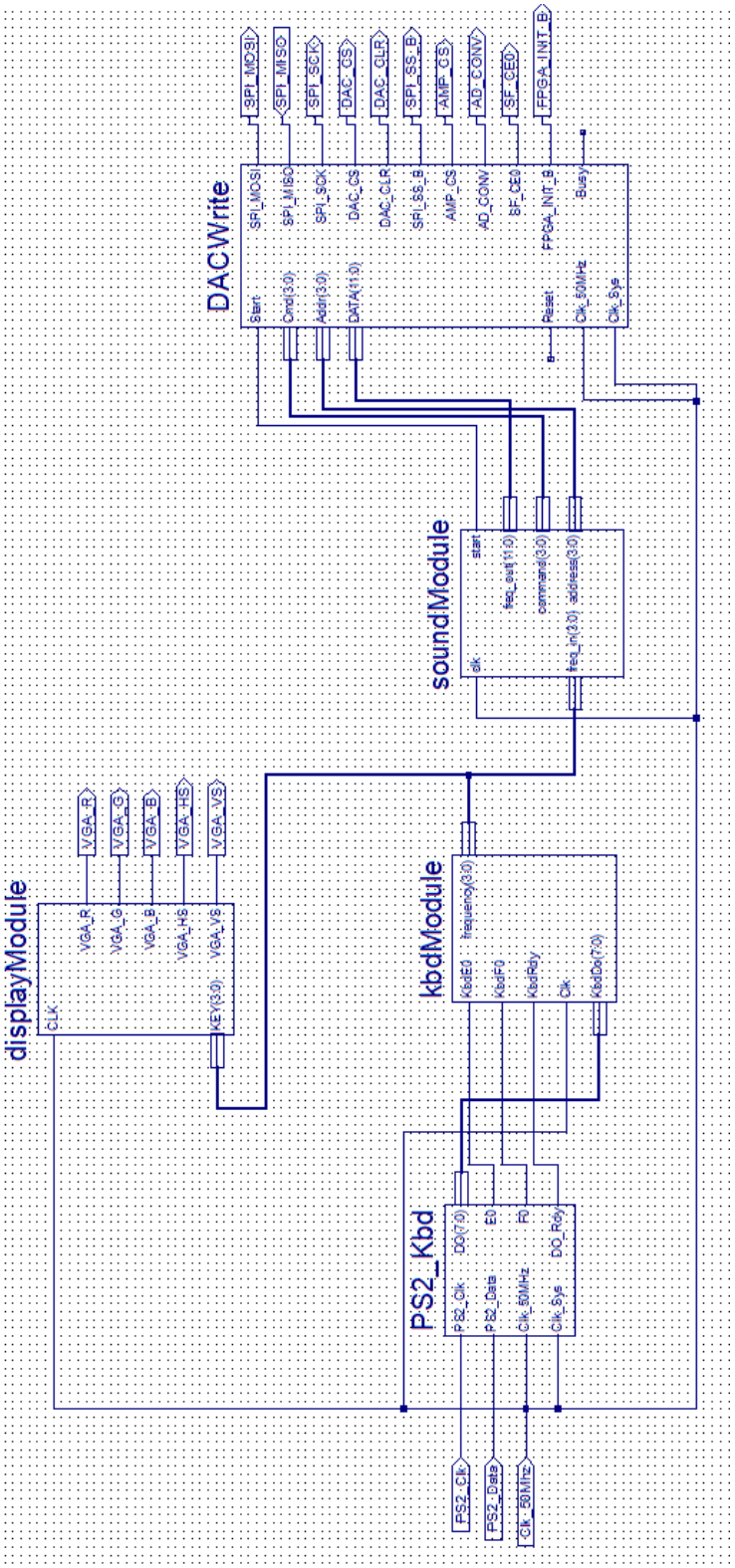
Do kreowania pojedynczych dźwięków przy jednocześnie jak najprostszej implementacji kodu najbardziej nadawało się generowanie sygnału piłokształtnego. Powstawał on dzięki inkrementacji liczb od 0 do 31 przy zboczu narastającym zegara, po czym licznik zaczynał działać od nowa. Przy tak generowanym sygnale trzeba było uwzględnić taktowanie zegara, które wynosiło 50 MHz. By wydobyć dźwięk o odpowiednio wysokim tonie, taką liczbę trzeba było podzielić przez zakres generowanego sygnału (czyli 32) i częstotliwość wzorcową dźwięku, który chcemy uzyskać. Przykładowo więc dla dźwięku z oktawy trzykreślnej c^3 o częstotliwości wzorcowej 1046,5 było to działanie: *50 000 000 : 32 : 1046,5 ≈ 1493,07*, a więc 5-bitowy sygnał piłokształtny (bo liczący od 0 do 31 – 5 bitów) musiał być wygenerowany o częstotliwości 1494 Hz (zaokrąglając w górę). Z racji tego licznik zewnętrzny tworzył sygnał 1494 razy w trakcie jednego taktu, po czym go resetował. Przed wysłaniem informacji do głośnika, sygnał jest jeszcze przepuszczony przez przetwornik cyfrowo-analogowy.

1.3.2 Generowanie obrazu

Sterownik VGA odpowiadający za obraz musi generować liczniki odchylenia poziomego i pionowego odpowiedzialne za synchronizację wyświetlania. Konkretne kształty można definiować poprzez ograniczenie tych odchyleń, wypełniając wnętrze kolorami. Istotne są także informacje na temat palety barw składających się z trzech podstawowych kolorów (czerwony, zielony, niebieski), wygaszając piksele będące poza wyświetlaczem. Do odpowiedniego wyświetlania potrzebujemy wewnętrzny zegar o sygnale 25 MHz, oprzeć go można o zegar wbudowany z taktowaniem 50 MHz.

2. Projekt

2.1 Schemat szczytowy



Rysunek 6 – pełen schemat

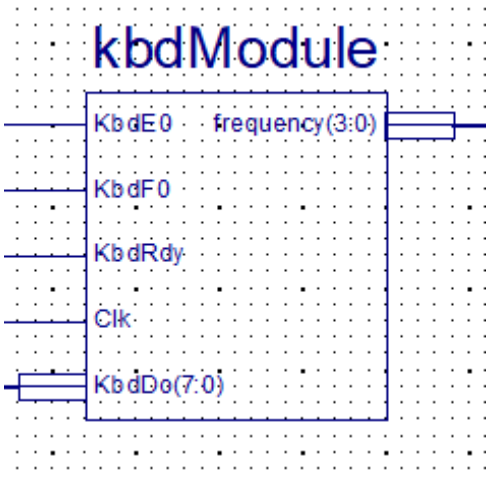
Schemat szczytowy składa się z 5 modułów, w tym dwóch gotowych, ściągniętych ze strony uczelni i odpowiednio podpiętych do reszty schematu. Do każdego z nich podpięty jest zegar wbudowany w układ z taktowaniem 50 MHz.

Na samym początku znajduje się gotowy moduł PS2_Kbd, odpowiadający za sczytywanie kodów klawiatury, które z kolei odpowiadają poszczególnym dźwiękom. Odpowiednie kody na wyjściu przechodzą do już samodzielnie zaimplementowanego modułu kbdModule, który przypisuje odpowiednim kodom liczby całkowite, aby je w prosty sposób rozróżnić. Dalej dane przechodzą naraz do displayModule i soundModule. Pierwszy odpowiada za podświetlanie klawisza w organach rozrysowanych na monitorze. Drugi za generowanie sygnału piłokształtnego o określonej częstotliwości. Z soundModule konkretna częstotliwość wychodzi na gotowy moduł DACWrite, który odpowiada za przetworzenie cyfrowego sygnału na wejściu na analogowy na wyjściu, który później odpowiada za wydawanie dźwięku.

2.2 Opisy poszczególnych modułów

2.2.1 kbdModule

konwertuje kod klawisza klawiatury na liczbę całkowitą



Rysunek 7 – moduł kbdModule

Wejścia:

- KbdE0 - sygnalizuje, czy kod z klawiatury był poprzedzony bajtami X”E0”, czyli kodem rozszerzonym
- KbdF0 - sygnalizuje, czy kod był poprzedzony kodem rozszerzonym oznaczającym zwolnienie klawisza

- KbdRdy - sygnalizuje, czy zakończono wysyłanie kodu
- Clk – zegar
- KbdDo(7:0) – ośmiobitowa informacja zawierająca kod klawisza

Wyjścia:

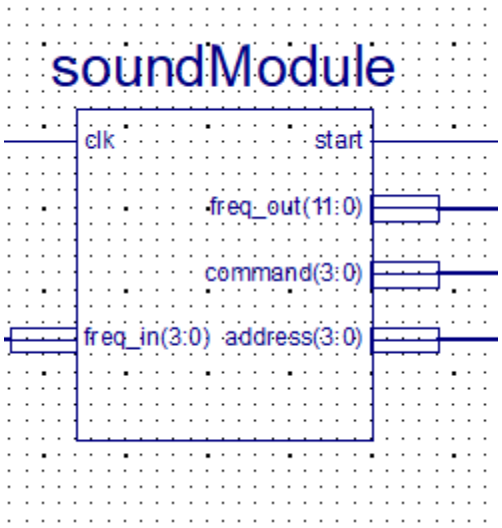
- frequency(3:0) – liczba całkowita przypisywana do kodu klawisza

```
entity kbdModule is
    Port ( KbdE0 : in  STD_LOGIC;
           KbdF0 : in  STD_LOGIC;
           KbdRdy : in  STD_LOGIC;
           KbdDo : in  STD_LOGIC_VECTOR (7 downto 0);
           Clk : in  STD_LOGIC;
           frequency : out  STD_LOGIC_VECTOR (3 downto 0));
end kbdModule;

   --przypisanie odpowiednich liczb znakom z klawiatury
      if rising_edge( Clk ) and KbdRdy = '1' then

      case (KbdF0 & KbdE0 & KbdDo) is
         when ( "00" & X"1C" ) => frequency <="0001"; -- dźwięk c
         when ( "00" & X"1D" ) => frequency <="0010"; -- c#
         when ( "00" & X"1B" ) => frequency <="0011"; -- d
         when ( "00" & X"24" ) => frequency <="0100"; -- d#
         when ( "00" & X"23" ) => frequency <="0101"; -- e
         when ( "00" & X"2B" ) => frequency <="0110"; -- f
         when ( "00" & X"2C" ) => frequency <="0111"; -- f#
         when ( "00" & X"34" ) => frequency <="1000"; -- g
         when ( "00" & X"35" ) => frequency <="1001"; -- g#
         when ( "00" & X"33" ) => frequency <="1010"; -- a
         when ( "00" & X"3C" ) => frequency <="1011"; -- a#
         when ( "00" & X"3B" ) => frequency <="1100"; -- b
         when ( "00" & X"42" ) => frequency <="1101"; -- c
         when others => frequency <="0000";
      end case;

      end if;
```

Fragment kodu kbdModule

2.2.2 soundModule

przypisaną liczbę całkowitą z kbdModule przelicza na odpowiednią częstotliwość sygnału, po czym ten sygnał generuje za pomocą 5-bitowego licznika



Rysunek 8 – moduł soundModule

Wejścia:

- clk – zegar
- freq_in(3:0) – liczba całkowita pobrana z kbdModule odpowiadająca kodom z klawiatury

Wyjścia:

- start – wysyła wiadomość do DACWrite, czy dane na jego wejściu można zatrzasnąć
- freq_out(11:0) – sygnał piłokształtny o odpowiedniej częstotliwości, z racji tylko 5 bitów informacji reszta była wypełniona zerami
- command(3:0) – zatrzaskiwana data, wpisana na twardo liczba 3 (11 binarnie)
- address(3:0) – zatrzaskiwana data, wpisana na twardo liczba 15 (1111 binarnie)

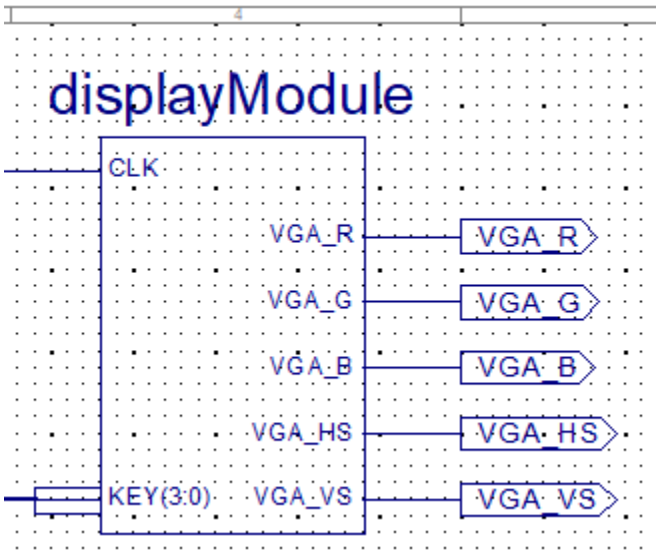
```
--sygnały wewnątrz modułu
signal freq_temp : integer;
signal data_sample : UNSIGNED (4 downto 0) :=  X"0"&'0';
signal counter : UNSIGNED (11 downto 0) := X"000";
signal clk_cnt :STD_LOGIC;
begin

freq_proc: process(freq_in, freq_temp)
begin
   --wybór odpowiedniej częstotliwości
   case freq_in is
      when "0001" => freq_temp <=1494;
      when "0010" => freq_temp <=1409;
      when "0011" => freq_temp <=1330;
      when "0100" => freq_temp <=1255;
      when "0101" => freq_temp <=1184;
      when "0110" => freq_temp <=1118;
      when "0111" => freq_temp <=1055;
      when "1000" => freq_temp <=996;
      when "1001" => freq_temp <=940;
      when "1010" => freq_temp <=888;
      when "1011" => freq_temp <=838;
      when "1100" => freq_temp <=791;
      when "1101" => freq_temp <=747;
      when others => freq_temp <=0;
   end case;

end process;
--licznik liczący do 1494 lub mniej, zależnie od podanego freq_temp
--tyle razy wygenerujemy sygnał piłokształtny podczas jednego taktu zegara
counter_proc: process(clk, data_sample)
   variable iterator : natural range 0 to 1494;
   begin
      iterator := 0;
      if(rising_edge(clk)) then
        counter <= counter + 1;
        start <= '0';
        if STD_LOGIC_VECTOR(counter) = freq_temp then
         counter <= X"000";
         data_sample <= data_sample + 1;
         start <= '1';
        end if;
      end if;
   freq_out <= STD_LOGIC_VECTOR (data_sample)&"0000000";
   address <= "1111";
   command <= "0011";

end process;
```

```
--generowanie sygnału piłokształtnego
clk_it: process( clk_cnt )
  variable iterator2 : integer := 0;
   begin
     iterator2 := 0;
      if(rising_edge(clk_cnt)) then
         if(iterator2 = 31) then
               iterator2 := 0;
            else
               iterator2 := iterator2 + 1;
         end if;
      end if;
   end process;
```

Fragmenty kodu soundModule

2.2.3 displayModule

wyświetla na monitorze organy i w zależności od danych na wyjściu kbdModule podświetla odpowiedni klawisz



Rysunek 12 – symulacja modułu soundModule

Wejścia:

- CLK - zegar

- KEY(3:0) – liczba całkowita reprezentująca klawisz pobrany z kbdModule

Wyjścia:

- VGA_R – zmienna odpowiadająca za kolor czerwony na wyjściu
- VGA_G – kolor zielony
- VGA_B – kolor niebieski
- VGA_HS – impuls synchronizacji poziomej (horizontal)
- VGA_VS – impuls synchronizacji pionowej (vertical)

```
--generowanie obrazu na podstawie wewnętrznego zegara 25 MHz
process (clk25mhz)
begin
  if clk25mhz'event and clk25mhz = '1' then
    if (hc >= "0010010000" ) -- 144
    and (hc < "1100010000" ) -- 784       zakresy licznika szerokości
    and (vc >= "0000000000" ) -- 0                      i wysokości
    and (vc < "1010000000" ) -- 640
    then
      -- C   takie zakresy będą miały liczniki vc i hc, tworząc obraz klawisza C
      if vc >= 0 and vc <= 640 and hc >= 184 and hc <= 234 then
         if KEY = "0001" then
            VGA_R <= '0';
            VGA_G <= '1';       --jeśli klawisz KEY równoważny przyciskowi klawiatury
            VGA_B <= '0';       --będzie odpowiadał dźwiękowi C,
         else                   --obraz podświetli się na zielono
            VGA_R <= '1';
            VGA_G <= '1';       --w innym przypadku na biało
            VGA_B <= '1';
         end if;
      --klawisze podzielone są na dwa prostokąty
      elsif vc >= 300 and vc <= 640 and hc >= 234 and hc <= 252 then
         if KEY = "0001" then
            VGA_R <= '0';
            VGA_G <= '1';
            VGA_B <= '0';
         else
            VGA_R <= '1';
            VGA_G <= '1';
            VGA_B <= '1';
         end if;
      -- C#    kolejne klawisze organów działają analogicznie
      elsif vc >= 0 and vc <= 300 and hc >= 234 and hc <= 274 then
         if KEY = "0010" then
            VGA_R <= '0';
            VGA_G <= '1';
            VGA_B <= '0';
         else
            VGA_R <= '0';
            VGA_G <= '0';
            VGA_B <= '0';
```

Fragmenty kodu displayModule

2.3 Symulacje wybranych modułów

Poniżej przedstawione zostały zrzuty ekranu dla kilku przykładowych symulacji obrazujących działanie poszczególnych modułów.

2.3.1 Symulacja licznika 32-bit

Symulacja na *Rysunku 10* obrazuje działanie licznika. Sygnał stepcounter odlicza od 0 do zadanej częstotliwości, a następnie ponownie się zeruje. Sygnał counter jest licznikiem 32-bitowym – odlicza od 0 do 31, a następnie znowu się zeruje.

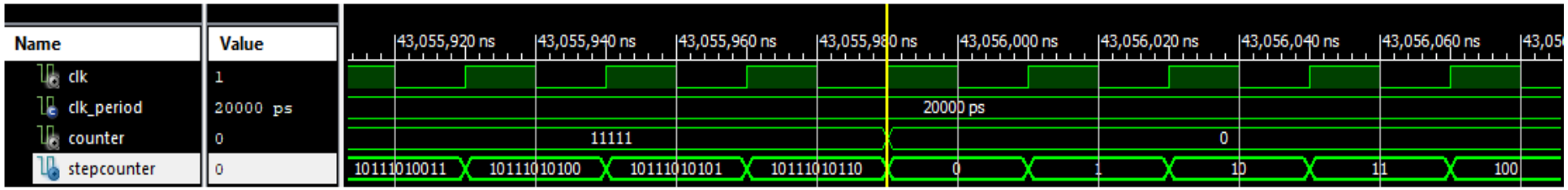
2.3.2 Symulacja kbdModule

W tej symulacji na wejście KbdDo zostały podane przykładowe kody odpowiadające klawiszom przypisanym do konkretnych dźwięków. Na symulacji na *rysunku 11* widać, że wyjście frequency przyjmuje odpowiednie wartości.

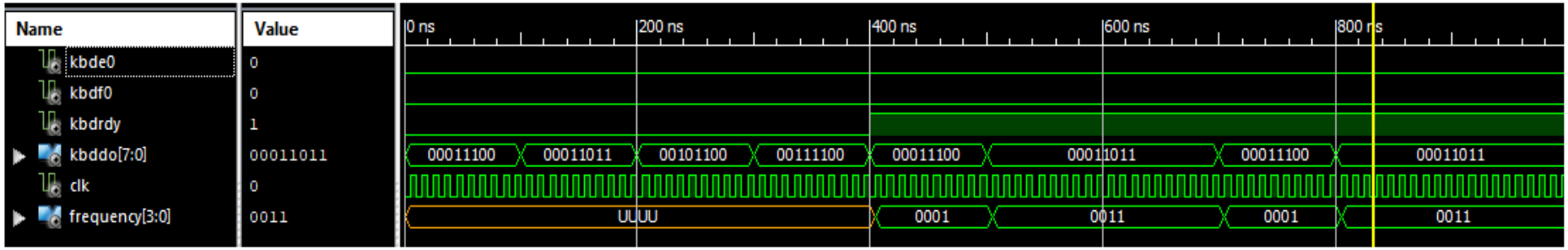
Aby wejście z klawiatury zostało odpowiednio odczytane, wejścia KbdF0 i KbdE0 muszą mieć przypisaną wartość 0, a KbdRdy wartość 1. Na symulacji widać, że po zmianie wartości KbdRdy na 1 moduł zaczyna sczytywać wejścia z klawiatury.

2.3.3 Symulacja soundModule

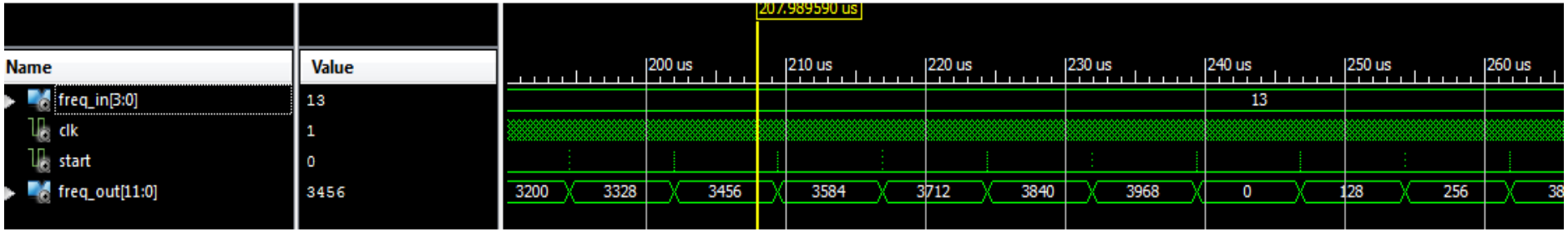
W tym teście na *Rysunku 12* na wejściu freq_in została przypisana liczba 13 odpowiadająca dźwiękowi C z następnej oktawy. Wyjście freq_out zostaje podane na wejście modułu *DACWrite*, jego wartości to wielokrotności liczby 128. Maksymalna wartość to 3968, ponieważ jest to wielokrotność liczby 31, a program działa w oparciu o licznik 32-bitowy.



Rysunek 10 – symulacja licznika



Rysunek 11 – symulacja modułu kbdModule



Rysunek 12 – symulacja modułu soundModule

3. Implementacja

3.1 Rozmiar

Całkowity rozmiar, jaki zajmuje program to 247 MB.

3.1.1 Zajętość LUT

Zużycie logiki: 289 z 9312 (1%)

Ogólna liczba LUT: 325 z 9312 (3%)

Użyte jako logika: 289

Użyte jako route-thru: 36

3.1.2 Slice

Liczba przerzutników Slice: 128 z 9312 (1%)

Ogólna liczba zajętych Slice: 198 z 4656 (4%)

3.1.3 BRAM

Grupa nie zużywała blokowego RAM-u.

3.2 Prędkość

Minimalny okres: 8.975 ns.

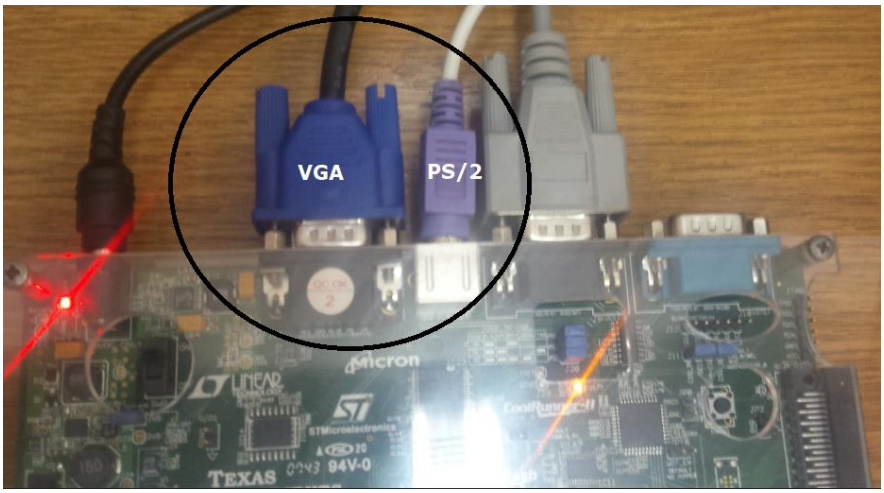
3.3 Podręcznik obsługi urządzenia

Wszystkie części wymagane do poprawnego użytkowania organów zostały przedstawione na *Rysunkach 1-4*. Są to:

- Spartan 3E
- Klawiatura z portem szeregowym PS/2
- Głośnik
- Monitor

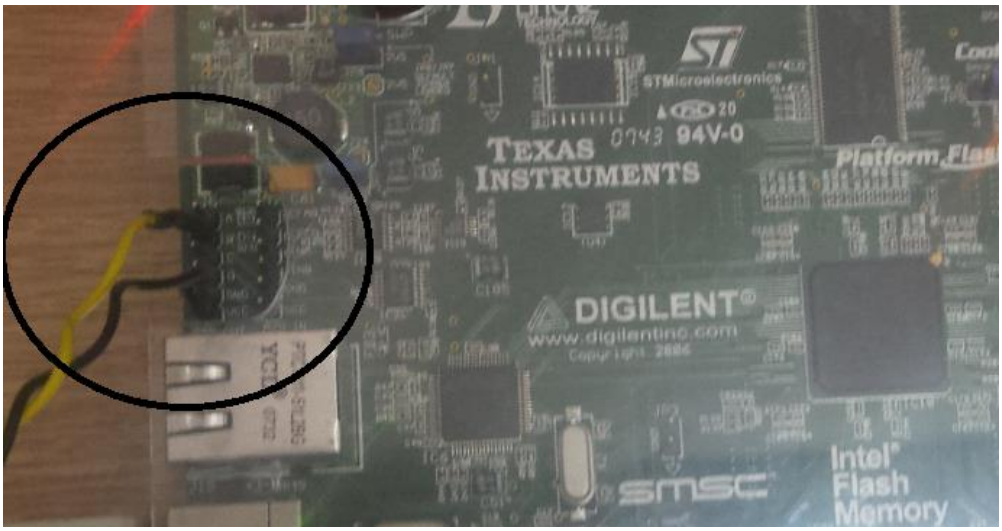
Po podłączeniu układu FPGA i monitora do zasilania należy podpiąć w odpowiednie wejścia urządzenia zewnętrzne. Porty dla monitora i klawiatury

to odpowiednio VGA i PS/2 pokazane wcześniej na *Rysunku 5*. Miejsca do podłączenia pokazano poniżej:



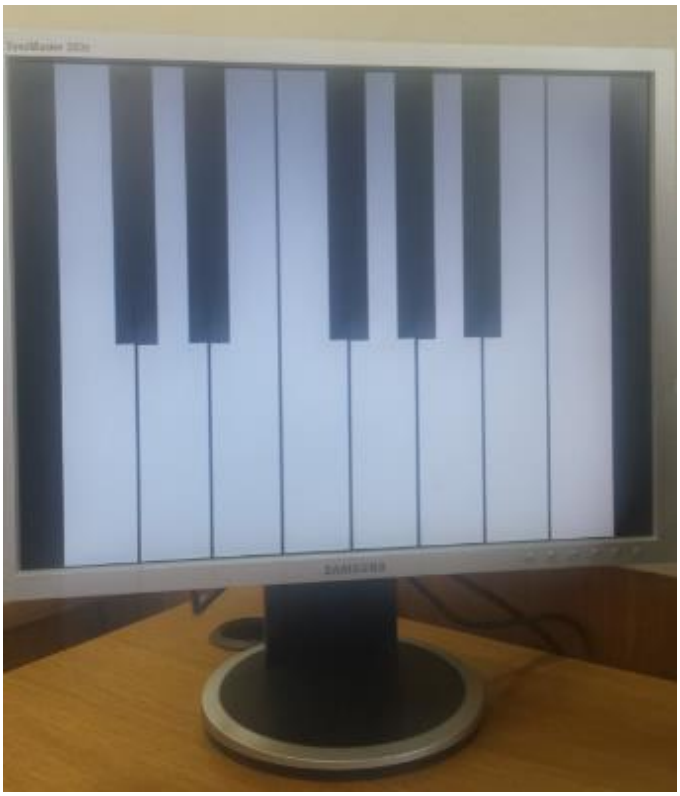
Rysunek 10 – podłączony monitor i klawiatura

Głośniczek należy podłączyć do przetwornika LTC2624 w sposób pokazany poniżej:



Rysunek 11 – podłączony głośnik

Układ należy uruchomić włączając przycisk *ON/OFF* w lewym górnym rogu Spartana. Gdy wszystko jest przygotowane potrzeba jeszcze uruchomić zaimplementowany program.



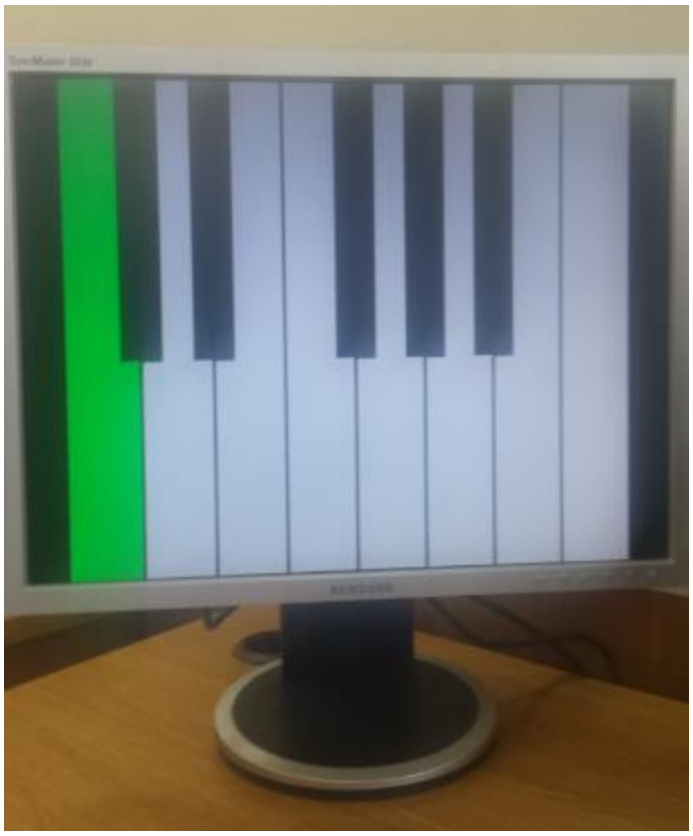
Rysunek 12 – organy

Gdy na monitorze ukażą się organy, można przystąpić do odgrywania dźwięków i wizualizacji ich przyciskania. Poniżej są odpowiednie klawisze klawiatury oraz dźwięki, które symulują.

Dźwięk	C^3	C#	D	D#	E	F	F#	G	G#	A	A#	H	C^4
Klawisz	A	W	S	E	D	F	T	G	Y	H	U	J	K

Tabela 1 – dźwięki i ich odpowiedniki na klawiaturze

Klawisz na organach powinien zmienić się po wybraniu odpowiadającej mu litery na klawiaturze.



Rysunek 13 – grające organy

4. Podsumowanie

4.1 Ocena krytyczna

Projekt spełnia początkowe wymagania i działa zgodnie z założeniami. Największym problemem podczas tego projektu okazało się samo wydobycie dźwięku z zestawu na początku pracy. Przydatna okazała się wiedza z laboratorium Układów Cyfrowych i Systemów Wbudowanych. Dużą korzyścią była przede wszystkim wyniesiona z laboratorium wiedza o monitorze VGA, która znacznie przyspieszyła i ułatwiła pracę nad ostatnim modułem, czyli tym odpowiadającym za wyświetlanie.

4.2 Dalsze prace

Projekt można rozszerzyć w łatwy sposób dodając kolejne oktawy. Za przełączanie między oktawami mógłby odpowiadać jakiś klawisz klawiatury, do którego nie został przypisany żaden dźwięk. Można również zaimplementować moduł przypominający pozytywkę, który grałby wybrane melodie. Inną kwestią, którą można rozbudować, jest moduł wyświetlania - można go rozszerzyć na przykład o wyświetlanie nazwy dźwięku w momencie jego zagrania.

5. Spis literatury

[1] https://www.xilinx.com/support/documentation/boards_and_kits/ug230.pdf

[2] http://www.zsk.ict.pwr.wroc.pl/zsk_ftp/fpga/

[3] http://staff.iiar.pwr.wroc.pl/antoni.sterna/ucsw/vga_driver.pdf

[4] https://www.fpga4fun.com/MusicBox1.html

[5] https://method-behind-the-music.com/mechanics/physics/