Urządzenia cyfrowe i systemy wbudowane 2

Projekt

Generator akustyczny przy użyciu modułu DACWrite

Organy z sygnałem wejściowym z PS2_Kbd + wyświetlanie "nut" poprzez VGAtxt48x20

Wtorek TN, 12:15

Bartosz Szymczak
252 734

Bartosz Górecki
248 846

prowadzący
Dr inż. Jarosław Sugier

Informatyka Techniczna
Wydział Informatyki i Telekomunikacji

6 czerwca 2022

Spis treści

1 Wprowadzenie

1.1 Cel i zakres projektu

Celem zadania projektowego było napisanie oprogramowania umożliwiającego symulację organów na układzie FPGA, na których można zagrać poprzez wciśnięcie odpowiednich przycisków na klawiaturze oraz wyświetlanie zagranych nut na monitorze. Zasięg naszych prac obejmował napisanie kodu obsługującego FPGA w języku VHDL, przeprowadzenie symulacji behawioralnej w celu zbadania zachowania modułów i ostateczne przetestowanie go w warunkach laboratoryjnym na dostępnym sprzęcie.
Projekt implementuje:

- moduł PS2_Kbd obsługujący konwersje sygnałów klawiatury na kody użytych klawiszy,
- moduł VGAtxt48x20 obsługujący wyświetlanie podawanych liter,
- moduł DACWrite obsługujący wysyłanie danych do przetwornika cyfrowo-analogowego,
- moduł konwertujący kody użytych klawiszy na odpowiadające im ustalone kody nut (Tabela 1),
- moduł konwertujący kody nut na odpowiednie częstotliwości, reprezentujące poszczególne nuty(Tabela 1),
- moduł konwertujący kody nut na odpowiednie kody ASCII[1],

1.2 Zagadnienia teoretyczne

1.2.1 Podstawowe założenia

Układ korzysta z jednego sygnału zegarowego o częstotliwości 50 MHz. Sygnał ten dostępny jest na wyprowadzeniu C9 i traktowany jest jako podstawowe źródła synchronizacji układu. Plik GenIO.UCF zawiera kompletną definicję tego wyprowadzenia.

1.2.2 Wytwarzanie dźwięku

Na początku zajęć projektowych otrzymaliśmy zadanie wytworzenia piło-kształtnego sygnału o częstotliwości 1 kHz. Należało stworzyć sygnał z 16 "schodków", uwzględniając informację o taktowaniu zegara płyty FPGA wynoszącym 20 ns. W celu wyliczenia po ilu cyklach zegara należy iterować "schodek" przekonwertowano wymaganą częstotliwość z Hz na okres w ns. Dalej podzielono otrzymaną wartość przez 20 (taktowanie płyty) i dalej podzielono przez 16 (ilość "schodków"). Następnie na bazie stworzonego sygnału należało wytwarzać kolejne dźwięki o 12 różnych częstotliwościach. Wybraliśmy następujące częstotliwości nut[3] oraz określiliśmy ich właściwości:

nuta	częstotliwość	ilość cykli w iteracji	kod nuty	przypisany klawisz	wyświetlona litera
c4	523,5 Hz	5969	0001	A	C
c#4	554,37 Hz	5637	0010	W	c
d4	587,33 Hz	5320	0011	S	D
d#4	622,85 Hz	5017	0100	E	d
e4	659,26 Hz	4740	0101	D	E
f4	698,46 Hz	4474	0110	F	F
f#4	739,99 Hz	4223	0111	R	f
g4	783,99 Hz	3986	1000	G	G
g#4	830,61 Hz	3762	1001	T	g
a4	880 Hz	3551	1010	H	A
a#4	932,33 Hz	3351	1011	Y	a
h4	987,77 Hz	3163	1100	J	H

Tabela 1: Wyznaczone właściwości określonych dźwięków

1.2.3 Moduł DAC

W celu użycia głośnika należało użyć modułu DAC. Moduł DAC to czterotaktowy, szeregowy, konwerter cyfrowo-analogowy[6]. Moduł umożliwia nam podłączenie się do czterech wyjść (od A do D, Rysunek 1), które widoczne są nad złączem Ethernet RJ-45. FPGA początkowo wysyła do układu osiem fikcyjnych bitów, a następnie 4-bitowe polecenie. Do stworzenia programu wykorzystaliśmy najpopularniejsze polecenie "0011", które przekłada się na natychmiastową aktualizację wybranych wyjść. Dalej wysyłany jest 4-bitowy adres, który określiliśmy jako "1111", czyli adres wszystkich wyjść. Następnie FPGA wysyła 12 bitową wartość danych, którą DAC konwertuje na wybranych wyjściach. Następnie wysyła cztery fikcyjne bity na koniec (Rysunek 2).

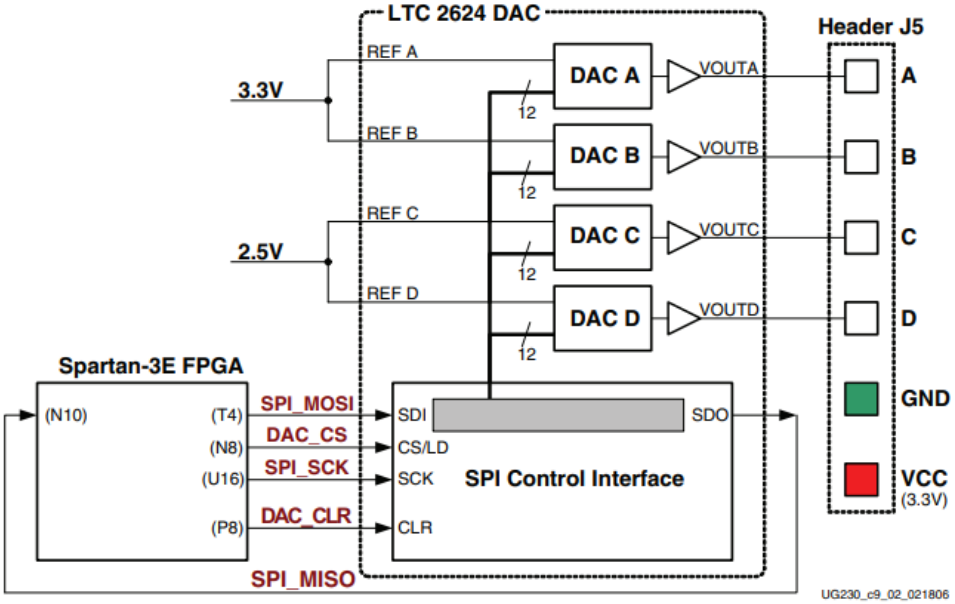
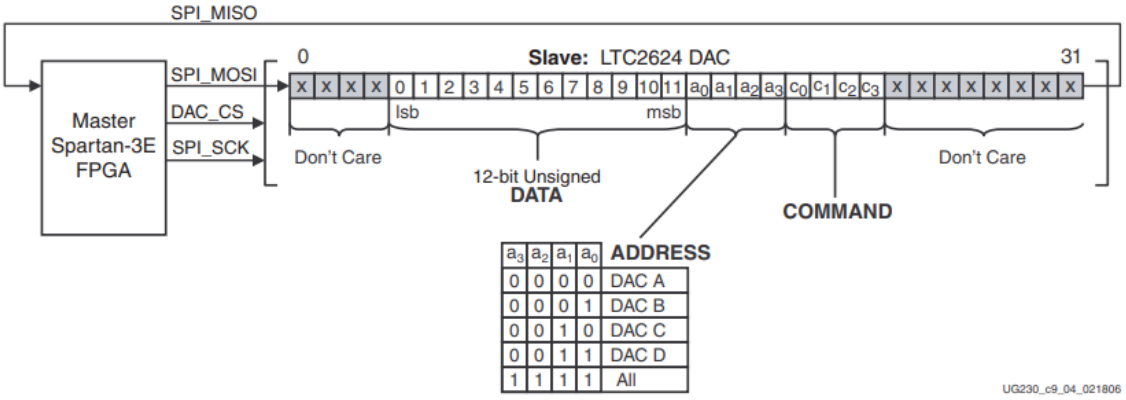


Figure 9-2: **Digital-to-Analog Connection Schematics**

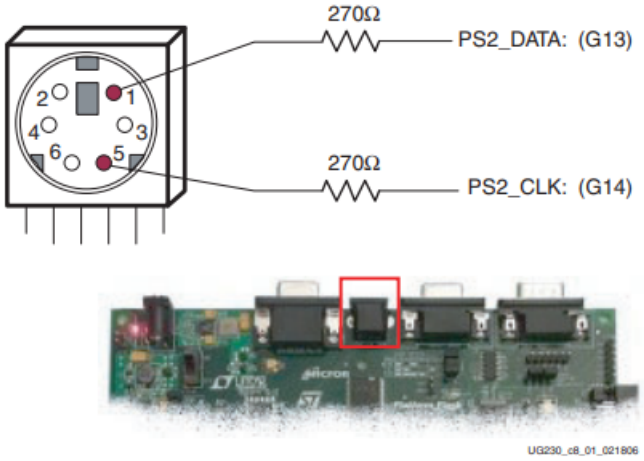
Rysunek 1: Schemat modułu DAC



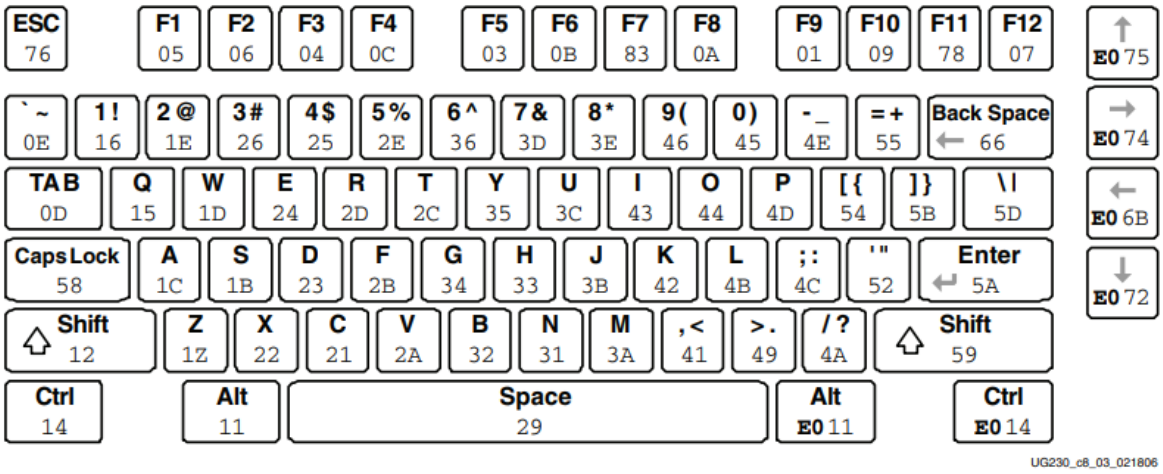
Rysunek 2: Protokół komunikacyjny modułu DAC

1.2.4 Interfejs PS/2

W celu wprowadzania sygnałów wejściowych, do płytki podłączono klawiaturę poprzez złącze PS/2 (Rysunek 3). Interfejs PS/2 [2] jest 6-pinowym portem używanym do połączenia myszki bądź klawiatury. Oba możliwe urządzenia używają tych samych czasów sygnału i 11-bitowych poleceń, zawierających bit początkowy, stop i bit parzystości. Klawiatura w stylu PS/2 (prawie każda obecnie używana) używa kodów skanowania do przekazywania danych o naciśniętych klawiszach (Rysunek 4). Każdy klawisz posiada unikalny klucz, który zostaje przesłany w sytuacji jego wciśnięcia. Jeśli klawisz zostanie naciśnięty i przytrzymany, klawiatura wielokrotnie wysyła kod skanowania co 100 ms. Po zwolnieniu klawisza klawiatura wysyła kod „F0", a następnie zeskanowany kod opuszczonego klawisza.



Rysunek 3: Lokalizacja oraz sygnały złącza PS/2



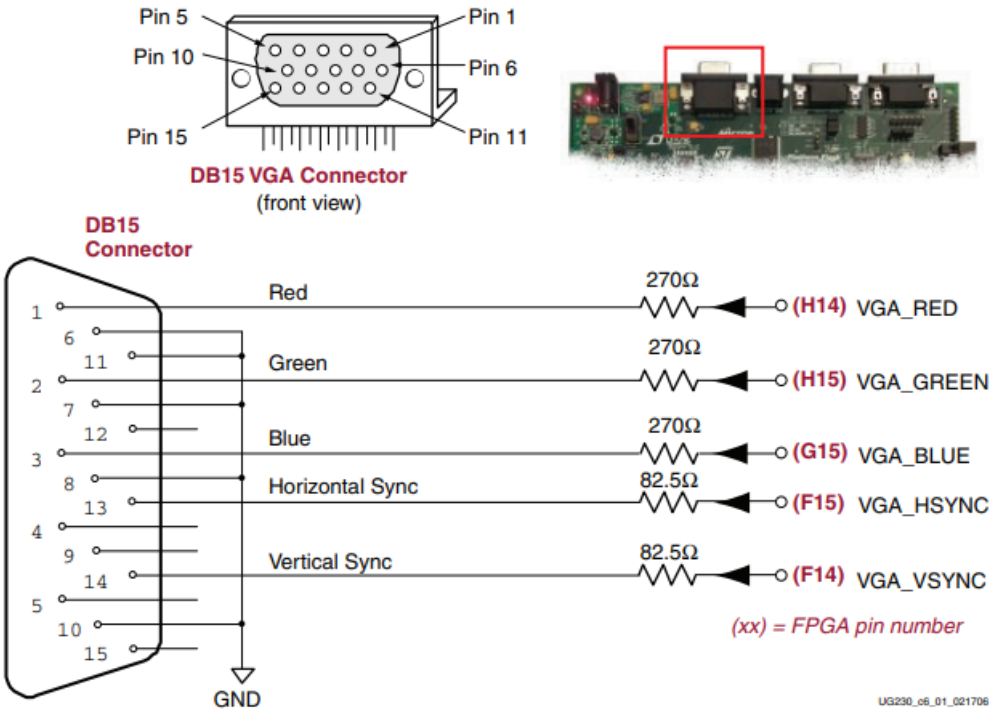
Rysunek 4: Kodowanie poszczególnych klawiszy PS/2

1.2.5 VGA

Do podłączenia monitora należało użyć złącza VGA (Rysunek 6). VGA (ang. Video Graphics Array)[4] to standardowy typ połączenia dla urządzeń wideo, takich jak monitory i projektory. Jest to analogowy standard wideo o wysokiej rozdzielczości używany w miejscach, gdzie potrzebna jest możliwość przesyłania ostrego, dokładnego obrazu. Układ Spartan-3E posiada port VGA (złącze DB15). Port ten można bezpośrednio połączyć do większości monitorów komputerowych lub płaskich wyświetlaczy LCD przy pomocy standardowego kabla. Spartan-3E steruje bezpośrednio pięcioma sygnałami VGA za pośrednictwem rezystorów: VGA HSYNC, VGA VSYNC, VGA RED, VGA GREEN oraz VGA BLUE. VGA wykorzystuje trzy oddzielne przewody do przesyłania, składających się z trzech kolorów (R, G i B), piono- wo i poziomo synchronizujących sygnałów 0V/5V. Każda linia koloru ma szeregowy rezystor, z jednym bitem dla VGA RED, VGA GREEN oraz VGA BLUE 2. Odpowiednie sterowanie sygnałami VGA RED, VGA GREEN i VGA BLUE pozwoli na wygenerowanie ośmiu kolorów przedstawionych na Rysunku 5.

VGA_RED	VGA_GREEN	VGA_BLUE	Resulting Color
0	0	0	Black
0	0	1	Blue
0	1	0	Green
0	1	1	Cyan
1	0	0	Red
1	0	1	Magenta
1	1	0	Yellow
1	1	1	White

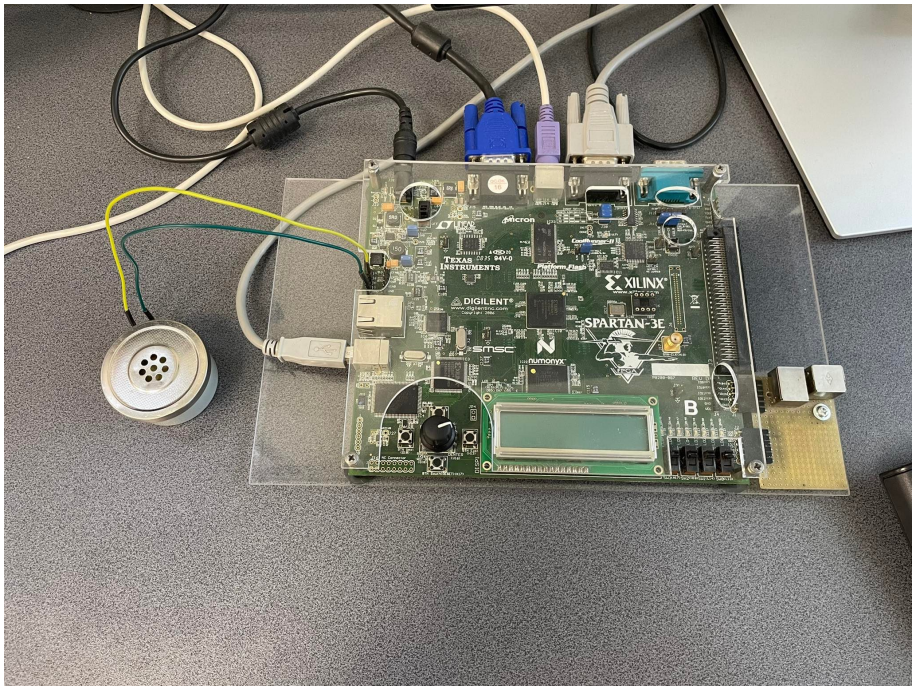
Rysunek 5: 3-bitowe kody kolorów wyświetlacza VGA



Rysunek 6: Połączenia VGA z Spartan-3E

1.3 Sprzęt

Do realizacji tematu wykorzystaliśmy układ **Digilent Spartan-3E Starter Board**[6] z matrycą logiczną FPGA marki **Xilinx**. Do układu podłączone zostały prosta klawiatura Genius poprzez złącze PS/2, monitor Samsung SyncMaster203 przez złącze VGA oraz prosty głośnik Tonsil W66 poprzez dwa piny DAC (A oraz uziemienie).

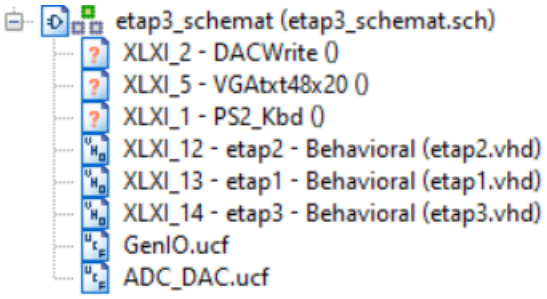


Rysunek 7: Płyta Spartan-3E oraz głośnik Tonsil W66

Rysunek 8: Płyta Spartan-3E, głośnik Tonsil W66, monitor Samsung SyncMaster203 oraz klawiatura Genius

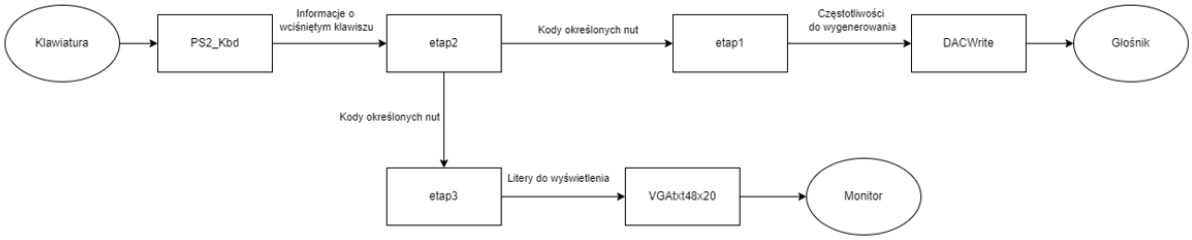
2 Projekt

2.1 Hierarchia projektu



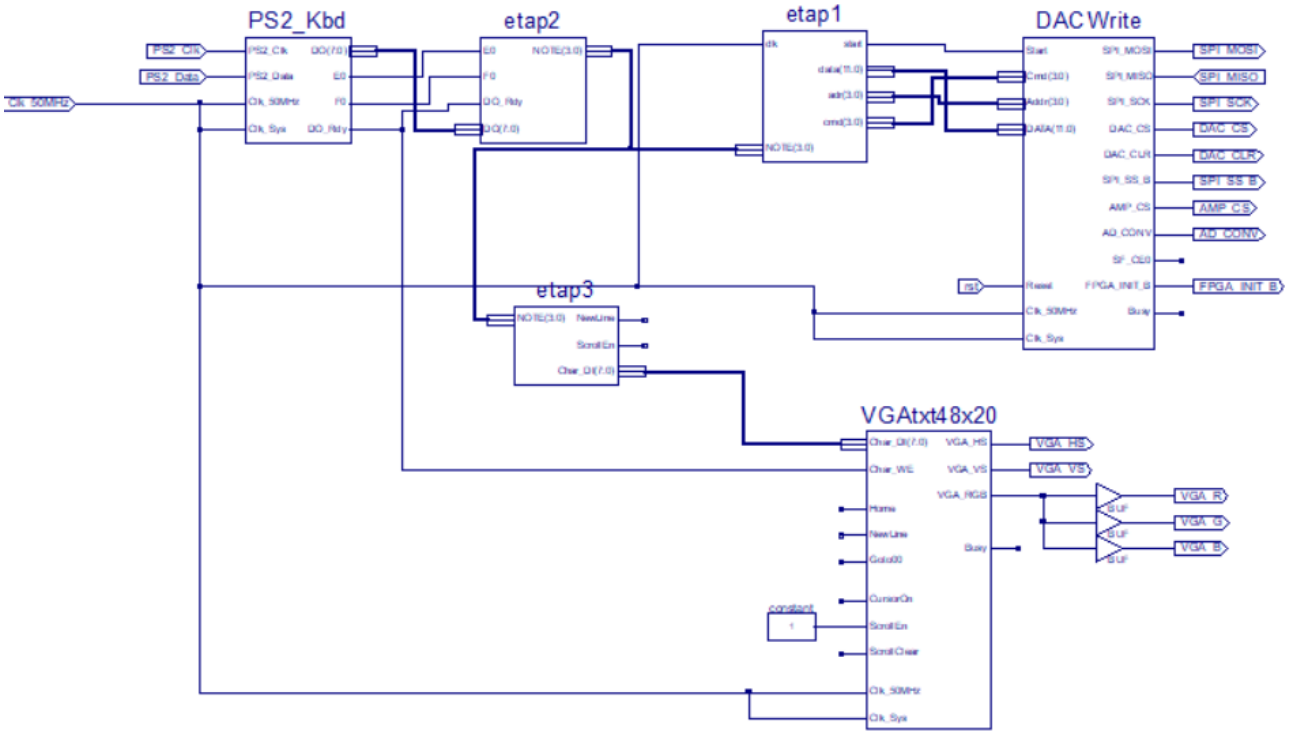
Rysunek 9: Hierarchia projektu

2.2 Przepływ danych



Rysunek 10: Diagram przepływu danych

2.3 Schemat logiczny



Rysunek 11: Schemat projektu

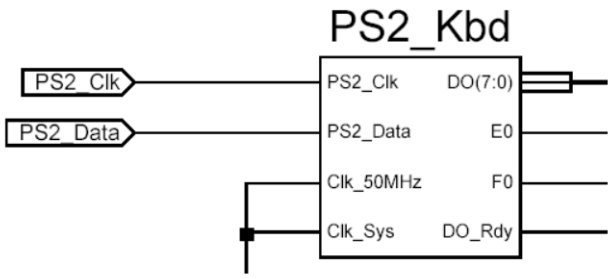
2.4 Moduły

Wykonany projekt składa się z następujących modułów:

- PS2_Kbd[5]
- DACWrite[5]
- VGAtxt48x20[5]
- etap1
- etap2
- etap3

2.4.1 PS2_Kbd

Moduł PS2_Kbd (Rysunek 12) jest pierwszym z użytych gotowych modułów[5] stworzonych przez prowadzącego na potrzeby kursu. Moduł ten jest odbiornikiem kodów wysyłanych przez klawiaturę PS/2. Sygnał DO_Rdy = '1' sygnalizuje zakończenie odbioru kodu (impuls jednotaktowy), na DO(7:0) podawana jest wówczas jego wartość, a wyjścia E0 oraz F0 sygnalizują, czy był on poprzedzony bajtami X"E0" (tzw. kod rozszerzony) oraz X"F0" (kod zwolnienia klawisza). Odbiór samych bajtów X"E0" oraz X"F0" nie jest sygnalizowany. Sygnały DO, E0 oraz F0 pozostają stabilne do czasu następnej transmisji klawiatury, w praktyce przez co najmniej kilkadziesiąt milisekund.



Rysunek 12: Schemat modułu PS2_Kbd

Wejścia układu:

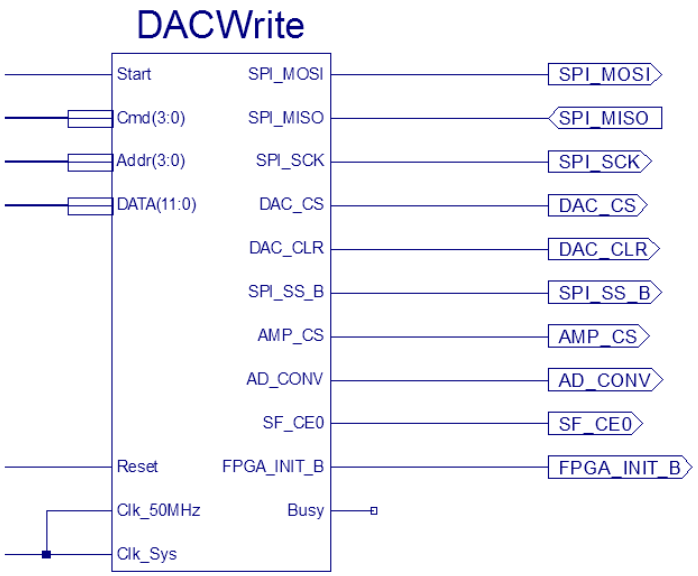
- PS2_Clk - wejście sygnału zegarowego klawiatury,
- PS2_Data - wejście zawierające dane z klawiatury,
- Clk_50Mhz i Clk_Sys - wejścia sygnału zegarowego 50Mhz,

Wyjścia układu:

- DO(7:0) - wektor 8-bitowy, zawierający kod użytego klawisza,
- E0 - wyjście sygnalizujące fakt poprzedzenia kodu klawisza bajtem X"E0",
- F0 - wyjście sygnalizujące fakt poprzedzenia kodu klawisza bajtem X"F0",
- DO_Rdy - sygnał określający zakończenie odbioru kodu.

2.4.2 DACWrite

Moduł DACWrite (Rysunek 13) to drugi z użytych gotowych modułów[5] stworzonych przez prowadzącego na potrzeby kursu. Obsługuje wysyłanie danych do przetwornika cyfra/analog LTC2624. Impuls Start = '1' zatrzaskuje dane na wejściach Cmd, Addr i DATA oraz rozpoczyna ich szeregowe wysłanie do układu. Czas trwania operacji sygnalizuje wyjście Busy.



Rysunek 13: Schemat modułu DACWrite

Wejścia układu:

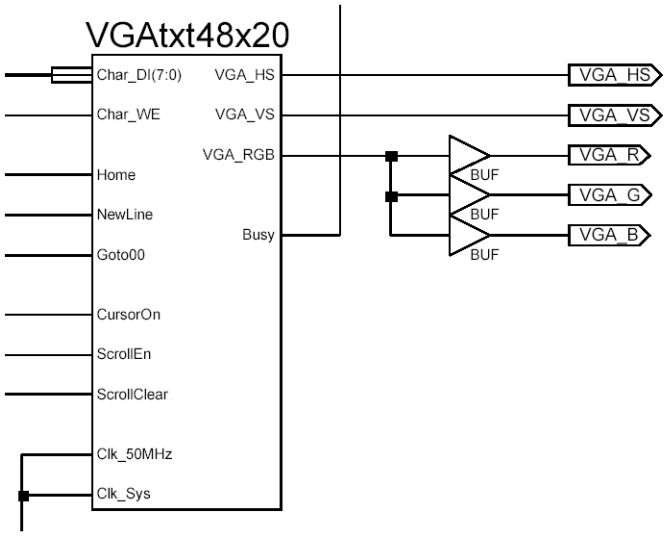
- Start - impuls zatrzaskujący dane na następnych wejściach,
- Cmd(3:0) - wektor 4-bitowy, zawierający polecenie[6]
- Addr(3:0) - wektor 4-bitowy, zawierający adres[6],
- DATA(11:0) - wektor 12-bitowy zawierający dane, które DAC przekonwertuje na określonych wyjściach[6],
- Reset - sygnał resetujący układ,
- Clk_50Mhz i Clk_Sys - wejścia sygnału zegarowego 50Mhz,
- SPI_MISO - dane szeregowe, wejście główne,

Wyjścia układu:

- SPI_MOSI - dane szeregowe, wyjście główne,
- SPI_SCK - zegar,
- DAC_CS - wybór chipa jako aktywnego,
- DAC_CLR - asynchroniczne wejście o niskim resecie aktywnym,
- SPI_SS_R - flash szeregowy SPI,
- AMP_CS - programowalny wzmacniacz,
- AD_CONV - konwerter analogowo-cyfrowy,
- SF_CE0 - pamięć StrataFlash Parallel Flash PROM,
- FPGA_INIT_B - pamięć Flash PROM,

2.4.3 VGAtxt48x20

Do wyświetlania nut użyliśmy kolejnego gotowego modułu, czyli VGAtxt48x20[5] (Rysunek 14). Moduł ten jest prostym terminalem znakowym wyświetlającym 20 linie po 48 znaków każda. Możliwe są podstawowe operacje przenoszenia kursora oraz automatyczne przewijanie ekranu. Początkowo pozycją bieżącą jest (0, 0). Jednotaktowy impuls Char_WE powoduje zapis na niej znaku o kodzie ASCII z Char_DI oraz jej automatyczną inkrementację. Po dojściu do ostatniej kolumny pozycja bieżąca automatycznie przechodzi do następnej linii, ponadto można ją zmieniać jednotaktowymi impulsami sygnałów Home, NewLine oraz Goto00. Gdy CursorOn = '1', na bieżącej pozycji wyświetlany jest kursor. Wyświetlany jest zestaw 128 znaków ASCII, bit Char_DI(7) jest ignorowany. Przewijanie ekranu jest wykonywane, gdy bieżąca pozycja opuszcza ostatnia linię i ScrollEn = '1'. Gdy ScrollEn = '0', pozycją następną po ostatniej linii jest (0, 0). Ponadto gdy ScrollEn = ScrollClear = '1', przewinięcie ekranu powoduje automatyczne wyczyszczenie nowej linii, które trwa 48 taktów i na ten przedział ustawia flagę Busy = '1'. W tym czasie ignorowane są wszystkie polecenia. Gdy ScrollEn = '1' i ScrollClear = '0', w nowej linii pokaże się stara zawartość; bufor cykliczny ma pojemność 32 linii. Za wyjątkiem czyszczenia nowej linii (ScrollEn = ScrollClear = '1'), wszystkie operacje wykonywane są w 1 takcie i flaga Busy nie jest ustawiana.



Rysunek 14: Schemat modułu VGAtxt48x20

Wejścia układu:

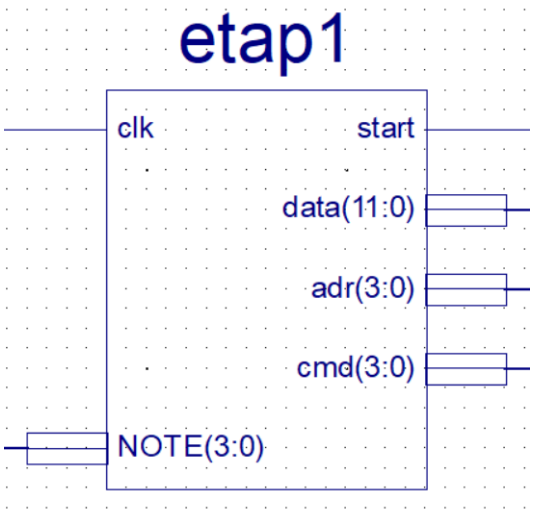
- Char_DI(7:0) - 8-bitowy sygnał, będący kodem ASCII zapisywanego znaku,
- Char_WE - impuls powodujący zapis znaku na ekranie,
- Home - powrót do początku linii,
- NewLine - przejście do następnej linii,
- Goto00 - przejście do początku ekranu,
- CursorOn - wyświetlanie kursora,
- ScrollEn - umożliwienie przewijania strony,
- ScrollClear - umożliwienie czyszczenia strony po przepełnieniu,
- Clk_50Mhz i Clk_Sys - wejścia sygnału zegarowego 50Mhz,

Wyjścia układu:

- VGA_HS - synchronizacja pozioma
- VGA_VS - synchronizacja pionowa,
- VGA_RGB - bity kolorów,
- Busy - impuls informujący o trwającej pracy układu.

2.4.4 etap1

Moduł etap1 został przygotowany na potrzeby realizacji tematu projektu, jest odpowiedzialny za pobieranie kodu nuty i przekonwertowanie go na odpowiednią częstotliwość dźwięku i wyprowadzenie go na głośnik.



Rysunek 15: Schemat modułu etap1

Wejścia układu:

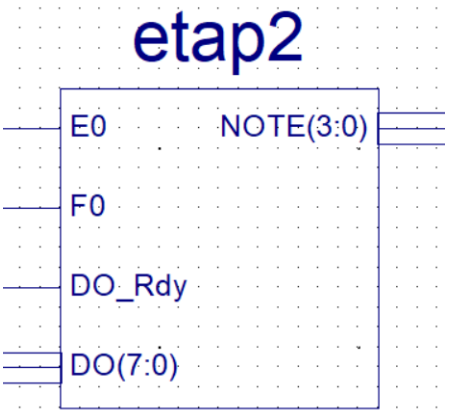
- clk - wejście zegarowe,
- NOTE(3:0) - wektor 4-bitowy zawierający ustalony kod nuty (Tabela 1),

Wyjścia układu:

- start - impuls zatrzaskujący dane na następnych wejściach,
- cmd(3:0) - wektor 4-bitowy, zawierający odpowiednie polecenie[6]
- addr(3:0) - wektor 4-bitowy, zawierający odpowiedni adres[6],
- data(11:0) - wektor 12-bitowy zawierający dane, które DAC przekonwertuje na określonych wyjściach[6],

2.4.5 etap2

Moduł etap2 został przygotowany na potrzeby realizacji tematu projektu, jest odpowiedzialny za pobieranie wartości wciśniętych klawiszy i konwertowanie ich na określone kody nuty (Tabela 1).



Rysunek 16: Schemat modułu etap2

Wejścia układu:

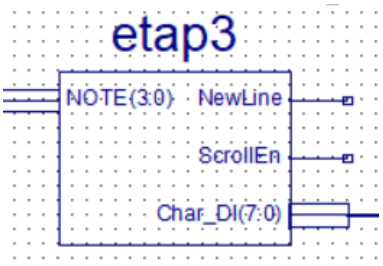
- DO(7:0) - wektor 8 bitów, zawierający kod użytego klawisza,
- E0 - wyjście sygnalizujące fakt poprzedzenia kodu klawisza bajtem X"E0",
- F0 - wyjście sygnalizujące fakt poprzedzenia kodu klawisza bajtem X"F0",
- DO_Rdy - sygnał określający zakończenie odbioru kodu.

Wyjścia układu:

- NOTE(3:0) - wektor 4-bitowy zawierający ustalony kod nuty,

2.4.6 etap3

Moduł etap3 został przygotowany na potrzeby realizacji tematu projektu, jest odpowiedzialny za pobieranie kodów nut i konwertowanie ich na odpowiednie kody ASCII w celu wyświetlenia ich na monitorze.



Rysunek 17: Schemat modułu etap3

Wejścia układu:

- NOTE(3:0) - wektor 4-bitowy zawierający ustalony kod nuty,

Wyjścia układu:

- Char_DI(7:0) - 8-bitowy sygnał, będący kodem ASCII zapisywanego znaku,
- ScrollEn - umożliwienie przewijania strony, (ostatecznie niewykorzystane)
- NewLine - przejście do następnej linii, (ostatecznie niewykorzystane)

2.5 Procesy

2.5.1 etap1

Kluczowymi składowymi algorytmu w etapie pierwszym są dwa liczniki. Licznik **innercounter** zlicza cykle zegara po których następuje inkrementacja "schodka" sygnału. Licznik **counter** zostaje zwiększony w sytuacji inkrementacji schodka. Te obliczenia mają miejsce w procesie **count**.

```
count: process (clk)
begin
if rising_edge(clk) then
  innercounter <= innercounter + 1;
  if(innercounter = notecode) then
    nextcounter <= std_logic_vector(unsigned(counter) + 1);
    innercounter <= "0000000000000000";
  end if;
end if;
end process;
```

Inkrementacja "schodka" przekłada się na przesłanie danych, czyli ustawienie sygnału start na "1". Licznik **counter** zostaje przekazany jako **data** po dodaniu do niego dodatkowych bitów, w celu spełnienia wymagań modułu DAC[6]. Działanie to wykonuje się w procesie **send**.

```
send: process(clk, counter, innercounter, nextcounter)
begin
adr <= "1111";
cmd <= "0011";
counter <= nextcounter;
data <= counter & "00000000";

if rising_edge(clk) then
    if innercounter = notecode then
      start <= '1';
    else
      start <= '0';
    end if;
  end if;
end process;
```

Konwertowanie kodu nuty **NOTE** na ilość potrzebnych cykli zegara do inkrementacji "schodka" **notecode** odbywa się w procesie **readNote**. Ilość cykli zegara została zapisana jako liczba binarna.

```
readNote: process (NOTE, notecode)
begin
  if NOTE = "0000" then
    notecode <= "0000000000000000";
  elsif NOTE = "0001" then
    notecode <= "0001011101010001";
  elsif NOTE = "0010" then
    notecode <= "0001011000000101";
  elsif NOTE = "0011" then
    notecode <= "0001010011001000";
  elsif NOTE = "0100" then
    notecode <= "0001001110011001";
  elsif NOTE = "0101" then
    notecode <= "0001001010000100";
  elsif NOTE = "0110" then
    notecode <= "0001000101111010";
  elsif NOTE = "0111" then
    notecode <= "0001000001111111";
  elsif NOTE = "1000" then
    notecode <= "0000111110010010";
  elsif NOTE = "1001" then
    notecode <= "0000111010110010";
  elsif NOTE = "1010" then
    notecode <= "0000110111011111";
  elsif NOTE = "1011" then
    notecode <= "0000110100010111";
  elsif NOTE = "1100" then
    notecode <= "0000110001011011";
  end if;
end process;
```

2.5.2 etap2

Program na poziomie etapu drugiego konwertuje wciśnięte klawisze **DO** na określone kody nut **NOTE** (Tabela 1 i Rysunek 3). Konwersja zostaje wykonana jedynie w przypadku sygnałów DO_Rdy i F0 ustawionych na "1" (zakończenie odbioru, na końcu bajt "F0").

```
readCode: process(DO,F0,DO_Rdy)
begin
  if DO_Rdy = '1' then
    if F0 = '0' then
      if DO = X"1C" then
        NOTE <= "0001";
      elsif DO = X"1D" then
        NOTE <= "0010";
      elsif DO = X"1B" then
        NOTE <= "0011";
      elsif DO = X"24" then
        NOTE <= "0100";
      elsif DO = X"23" then
        NOTE <= "0101";
      elsif DO = X"2B" then
        NOTE <= "0110";
      elsif DO = X"2D" then
        NOTE <= "0111";
      elsif DO = X"34" then
        NOTE <= "1000";
      elsif DO = X"2C" then
        NOTE <= "1001";
      elsif DO = X"33" then
        NOTE <= "1010";
      elsif DO = X"35" then
        NOTE <= "1011";
      elsif DO = X"3B" then
        NOTE <= "1100";
      end if;
    elsif F0 = '1' then
        NOTE <= "0000";
    end if;
  end if;
end process;
```

2.5.3 etap3

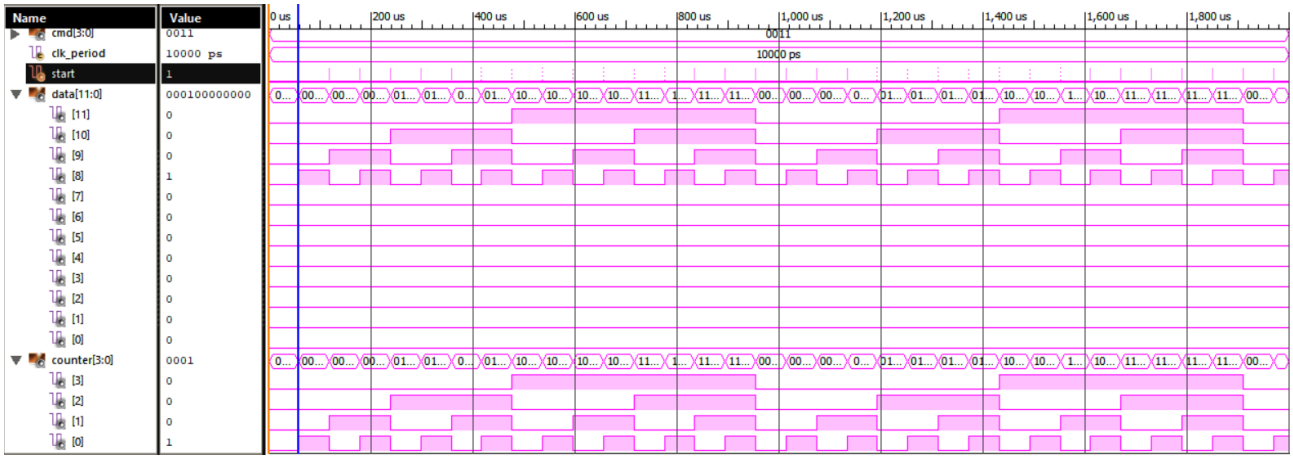
Kod etapu trzeciego konwertuje określone kody nut **NOTE** na kody ASCII odpowiadających im liter **Char_DI** (Tabela 1 i Rysunek 4).

```
Code: process (NOTE)
begin
  if NOTE = "0001" then
    Char_DI <= X"43";
  elsif NOTE = "0010" then
    Char_DI <= X"63";
  elsif NOTE = "0011" then
    Char_DI <= X"44";
  elsif NOTE = "0100" then
    Char_DI <= X"64";
  elsif NOTE = "0101" then
    Char_DI <= X"45";
  elsif NOTE = "0110" then
    Char_DI <= X"46";
  elsif NOTE = "0111" then
    Char_DI <= X"66";
  elsif NOTE = "1000" then
    Char_DI <= X"47";
  elsif NOTE = "1001" then
    Char_DI <= X"67";
  elsif NOTE = "1010" then
    Char_DI <= X"41";
  elsif NOTE = "1011" then
    Char_DI <= X"61";
  elsif NOTE = "1100" then
    Char_DI <= X"48";
  end if;
end process;
```

2.6 Symulacja behawioralna

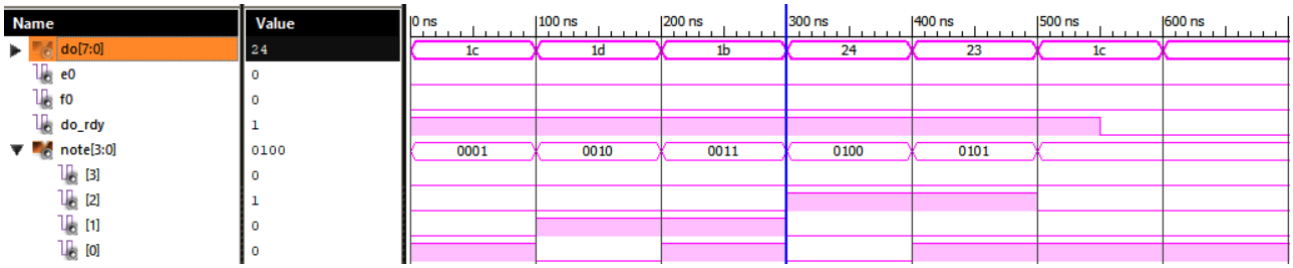
Symulacja behawioralna została ograniczona do poszczególnych modułów stworzonych na potrzeby projektu. Wynik symulacji dla całego projektu byłby stosunkowo ciężki do rozczytania, z racji faktu, że ostatecznie otrzymujemy sygnały potrzebne do mechanicznego wytworzenia dźwięku oraz wypisania na ekranie monitora określonych liter w odpowiednim miejscu.

Symulacja programu z etapu 1 (Rysunek 18) zwraca oczekiwane wyniki. Wewnętrzny licznik po określonej ilości cykli powoduje inkrementacje licznika głównego, który aktualizuje bity przesłania danych oraz powoduje zmianę sygnału start na ”1”.



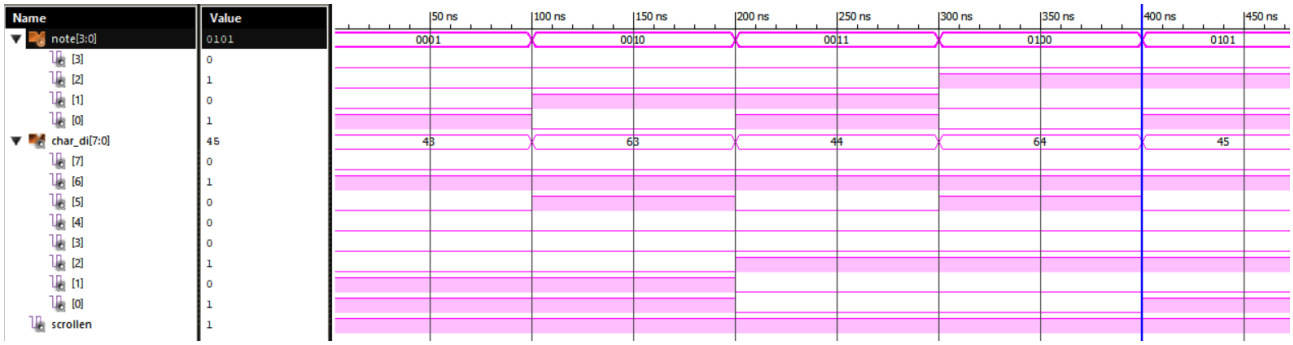
Rysunek 18: Symulacja behawioralna etapu 1

Symulacja programu z etapu 2 zwraca oczekiwane wyniki. Kolejne kody klawiszy konwertowane są na właściwe kody nut, w sytuacji gdy sygnał DO_Rdy jest ustawiony na ”1”. Dla przykładu klawisz o kodzie X”24” (E) został przekonwertowany na kod ”0100” zgdonie z Tabelą 1 (Rysunek 19).



Rysunek 19: Symulacja behawioralna etapu 2

Symulacja programu z etapu 3 zwraca oczekiwane wyniki. Kolejne kody nut konwertowane są na właściwe kody ASCII[1] reprezentujących ich liter (Tabela 1). Dla przykładu kod nuty ”0101” został przekonwertowany na kod ASCII X”45” (E) zgodnie z Tabelą 1 (Rysunek 20).



Rysunek 20: Symulacja behawioralna etapu 3

3 Implementacja

Rysunek 21 przedstawia dane na temat rozmiaru projektu – LUT, Slice, BRAM, użyciu DCMs i pozostałych komponentów:

3.1 Rozmiar projektu

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Total Number Slice Registers	218	9,312	2%
Number used as Flip Flops	195		
Number used as Latches	23		
Number of 4 input LUTs	285	9,312	3%
Number of occupied Slices	214	4,656	4%
Number of Slices containing only related logic	214	214	100%
Number of Slices containing unrelated logic	0	214	0%
Total Number of 4 input LUTs	350	9,312	3%
Number used as logic	282		
Number used as a route-thru	65		
Number used as Shift registers	3		
Number of bonded IOBs	18	232	7%
Number of RAMB16s	2	20	10%
Number of BUFGMUXs	1	24	4%
Average Fanout of Non-Clock Nets	3.14		

Rysunek 21: Informacje o wykorzystaniu zasobów urządzenia

3.2 Szybkość

	Met	Constraint	Check	Worst Case Slack	Best Case Achievable	Timing Errors	Timing Score
1	Yes	NET "Clk_50MHz_BUFGP/IBUFG" PERIOD = 20 ns HIGH 50%	SETUPHOLD	10.917ns0.788ns	9.083ns	00	00

Rysunek 22: Szybkość implementacji

3.3 Plik ucf

Do poprawnego działania programu potrzebne są pliki o następujących zawartościach:
GenIO.ucf

```
NET "Clk_50MHz" LOC = "C9" | IOSTANDARD = LVTTL;
NET "Clk_50MHz" PERIOD = 20.0ns HIGH 50%;

NET "PS2_Data" LOC = "G13" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8;
NET "PS2_Clk"  LOC = "G14" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8;

NET "VGA_R"  LOC = "H14" | IOSTANDARD = LVTTL | SLEW = FAST | DRIVE = 8;
NET "VGA_G"  LOC = "H15" | IOSTANDARD = LVTTL | SLEW = FAST | DRIVE = 8;
NET "VGA_B"  LOC = "G15" | IOSTANDARD = LVTTL | SLEW = FAST | DRIVE = 8;
NET "VGA_HS" LOC = "F15" | IOSTANDARD = LVTTL | SLEW = FAST | DRIVE = 8;
NET "VGA_VS" LOC = "F14" | IOSTANDARD = LVTTL | SLEW = FAST | DRIVE = 8;
```

ADC_DAC.ucf

```
NET "SPI_MISO" LOC = "N10" | IOSTANDARD = LVCMOS33 ;
NET "SPI_MOSI" LOC = "T4" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 6 ;
NET "SPI_SCK" LOC = "U16" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 6 ;

NET "SPI_SS_B" LOC = "U3" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 6 ;
NET "FPGA_INIT_B" LOC = "T3" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 4 ;

NET "DAC_CLR" LOC = "P8" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
NET "DAC_CS" LOC = "N8" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;

NET "AD_CONV" LOC = "P11" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 6 ;

NET "AMP_CS" LOC = "N7" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 6 ;
```

3.4 Podręcznik obsługi urządzenia

Użytkownik urządzenia Spartan 3E z uruchomionym opisywanym programem jest prosty w użyciu. Program pozwala na wytworzenie podstawowych dźwięków organów oraz wyświetleniu ich na monitorze. W celu wytworzenia określonego dźwięku należy podłączyć płytę FPGA do zasilania i komputera, załadować na płytę opisywany program, połączyć płytę z monitorem przez złącze VGA, z klawiaturą przez złącze PS/2 oraz głośnik przez piny DAC (Rysunek 7).

Stan początkowy zestawu:

- Brak wytwarzanego dźwięku,
- Brak wyświetlanych liter na monitorze,

Stan zestawu przy wciśnięciu obsługiwanego klawisza klawiatury (Tabela 1):

- Przez czas wciskania klawisza zostaje wytworzony dźwięk o określonej częstotliwości (Tabela 1),
- Na ekranie zostaje zapisana, powiązana z określoną częstotliwością, litera reprezentująca daną nutę (Tabela 1).

4 Podsumowanie

4.1 Wnioski

W trakcie tworzenia oraz po zakończeniu projektu doszliśmy do następujących wniosków:

- Powstały w trakcie zajęć projekt spełnia podstawowe założenia i posiada główne funkcjonalności, tzn. potrafi współpracować z określonymi urządzeniami oraz potrafi konwertować wytworzone sygnały na różne dźwięki oraz zapis liter na monitorze,
- Przeprowadzone operacje obsługi i konwersji sygnałów odbywa się w łatwy i przejrzysty sposób,
- Projekt jest gotowy do rozszerzeń o dodatkowe funkcjonalności np. obliczanie czasu trwania określonego czasu, wykorzystanie niewykorzystanych wyjść modułów (Rysunek 11) itp.

4.2 Ocena krytyczna i możliwe ulepszenia

Mimo spełnienia podstawowych założeń nie udało się zrealizować dodatkowych funkcjonalności. Dużą część czasu spędziliśmy na szukaniu rozwiązań problemów, które wynikały z popełniania prostych błędów i braku większego doświadczenia w środowisku ISE Xilinx, które nie jest intuicyjne dla początkujących użytkowników. Punkty do zrealizowania lub ulepszania to np:

- Dodanie wyświetlania na ekranie monitora czasu, przez który wytwarzano określone nuty,
- Wykorzystanie niewykorzystanych wyjść modułu etap3,

Bibliografia

[1] *ASCII*. URL: https://pl.wikipedia.org/wiki/ASCII. (dostępne w dniu 04.06.2022r.)

[2] Adam Chapweske. *The PS/2 Mouse/Keyboard Protocol*. URL: https://www.avrfreaks.net/sites/default/files/PS2%20Keyboard.pdf. (dostępne w dniu 04.06.2022r.)

[3] *Częstotliwości nut*. URL: https://muzycznelaboratorium.pl/8-czestotliwosci-nut. (dostępne w dniu 04.06.2022r.)

[4] Digi-Key. *VGA Controller (VHDL)*. URL: https://forum.digikey.com/t/vga-controller-vhdl/12794. (dostępne w dniu 04.06.2022r.)

[5] Jarosław Sugier. *Moduły Spartan 3E Starter*. URL: http://www.zsk.iiar.pwr.edu.pl/zsk_ftp/fpga. (dostępne w dniu 04.06.2022r.)

[6] Xilinx. *Spartan-3E FPGA Starter Kit Board User Guide*. 2011. URL: https://docs.xilinx.com/v/u/en-US/ug230. (dostępne w dniu 04.06.2022r.)