Projekt
Urządzenia cyfrowe i systemy wbudowane 2

Adam Troszczyński 263801, Kamil Pawelski 263795

20 maja 2024

Spis treści

1. Wprowadzenie

1..1 Cel i zakres projektu

Celem projektu było zaprojektowanie oraz implementacja odtwarzacza pliku WAV z karty SD, który będzie generował efekty wizualne na monitorze poprzez interfejs VGA. Projekt miał zostać stworzony na układzie Spartan-3E, z użyciem języka VHDL.

1..2 Zagadnienia teoretyczne

Interfejs VGA (Video Graphics Array)

VGA to kontroler wyświetlania wideo i towarzyszący mu standard grafiki, po raz pierwszy wprowadzony wraz z komputerami IBM w 1987r [1].
Główne specyfikacje VGA to:

- Wybieralny zegar pikseli na poziomie 25,175 MHz lub 28,322 MHz.
- Maksymalnie 640 pikseli poziomych
- Tryb planarny do 16 kolorów (4 płaszczyzny bitowe)

Synchronizacja sygnałów

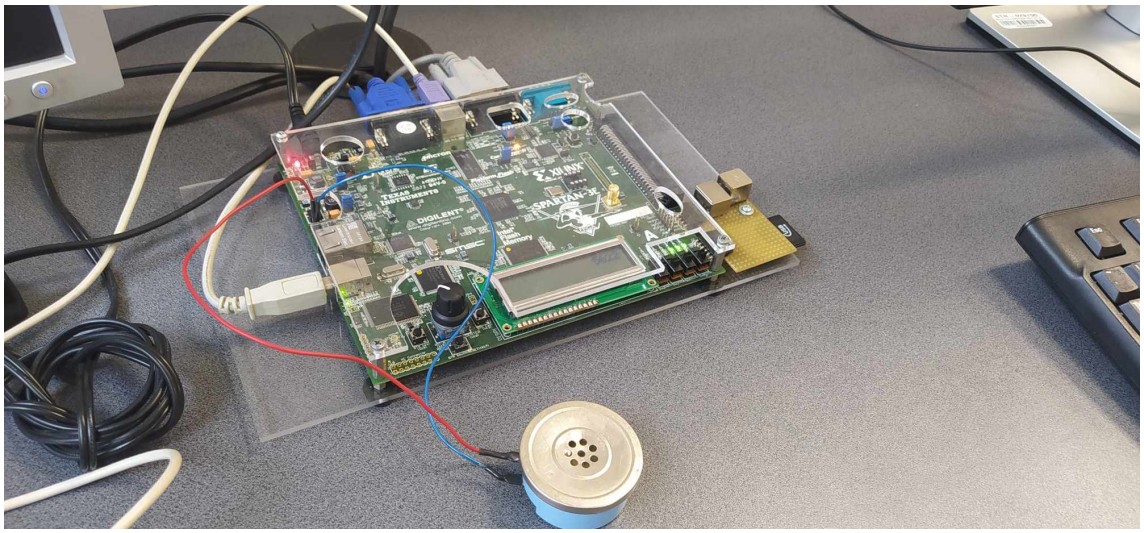
Najczęstszy tryb synchronizacji dla VGA to 640 x 480, 60 Hz. Dla naszego projektu ze względu na zegar 50 MHz, wybraliśmy tryb 800 x 600 oraz 72 Hz [2].

Format WAV (Waveform audio format)

Format plików dźwiękowych stworzonych przez Microsoft oraz IBM. Bazuje na formacie RIFF [3], poszerzając go o informacje o strumieniu audio, takie jak użyty kodek, częstotliwość próbkowania czy liczba kanałów. Format zapisu danych jest uzależniony od przyjętej rozdzielczości. Jeśli ta jest równa 8, wówczas zbiór wartości próbki wynosi od -128 do 127, przy czym taką wartość należy powiększyć o 128, otrzymując w ten sposób wartości pomiędzy 0 a 255 [4].

1..3 Sprzęt

Zdjęcie użytego sprzętu można zobaczyć na Rysunek 1



Rysunek 1: Spartan-3E, głośnik, karta SD

Spartan-3E

Projekt został zaimplementowany na płytce Spartan-3E [5].
Do realizacji projektu użyliśmy układów:

- FPGA XC3S500E [6]
- Port VGA
- Czytnik kart SD

Głośnik analogowy

W celu usłyszenia odczytanych dźwięków, wymagany jest głośnik analogowy, podłączany 2 pinami.

Karta SD

Pliki WAV muszą być umieszczone na karcie SD. Nazwy plików w postaci np. 1.wav, 2.wav.

Monitor

W celu wizualizacji odczytanych dźwięków, wymagany jest monitor działający w standardzie VGA

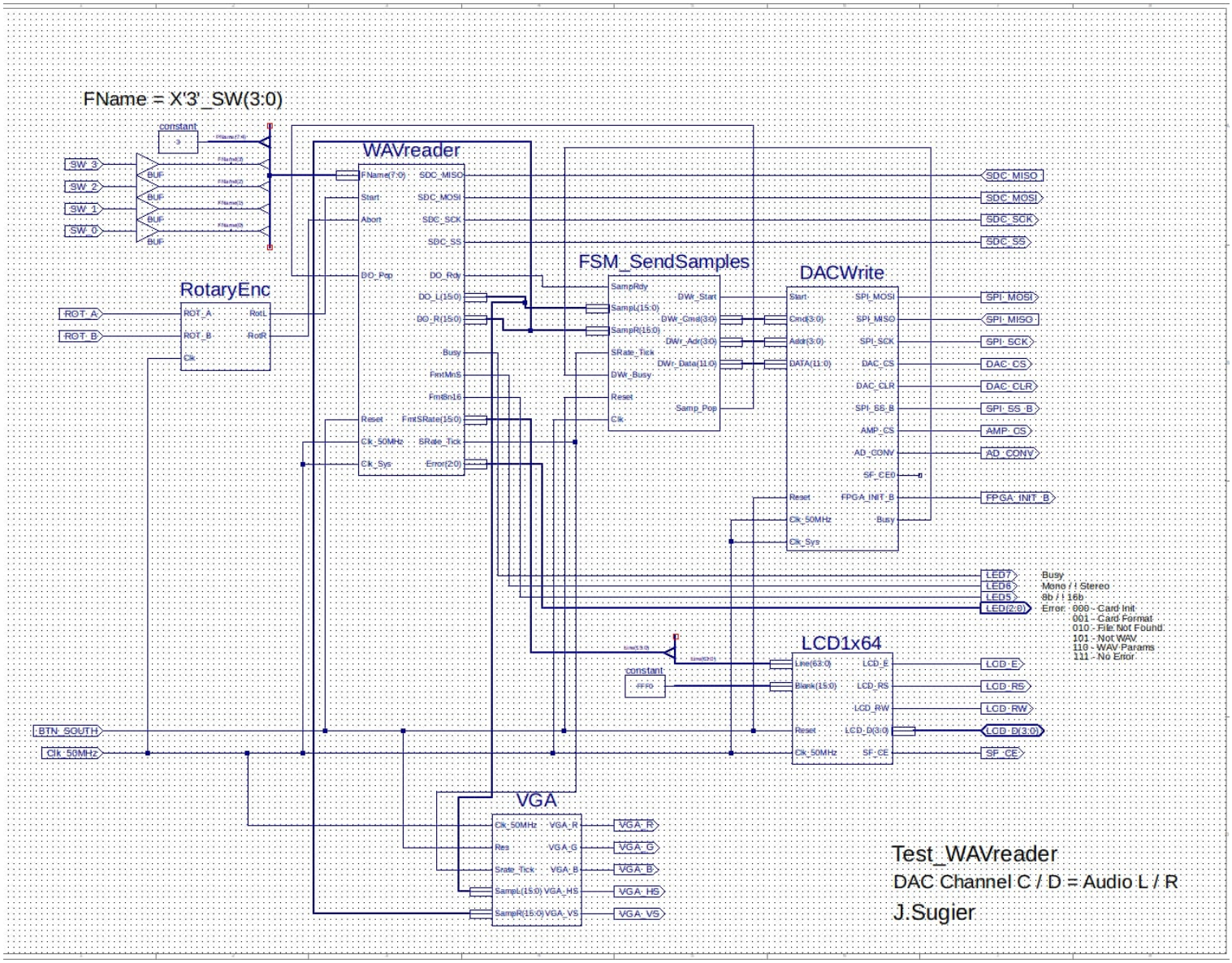
2. Projekt

2..1 Hierarchia źródeł

Moduły jakich użyliśmy w naszej implementacji to:

- RotaryEnc [7]
- WAVReader [8]
- FSM_SendSamples [9]
- DACWrite [10]
- LCD1x64 [11]
- VGA

Główny przepływ danych polega na tym, że WAVReader zczytuje dane z pliku wav, które następnie są używane przez FSM_SendSamples i DACWriter, które sterują dźwiękiem jaki wydobywa się z głośnika. Dane odczytane przez WAVReader trafiają również do naszego modułu VGA (dokładnie SampL, SampR oraz SRate_Tick). Następnie wygenerowane na podstawie tych danych sygnały przesyłamy bezpośrednio do portów odpowiadających za interfest VGA. Pełny schemat znajduje się na Rysunek 2

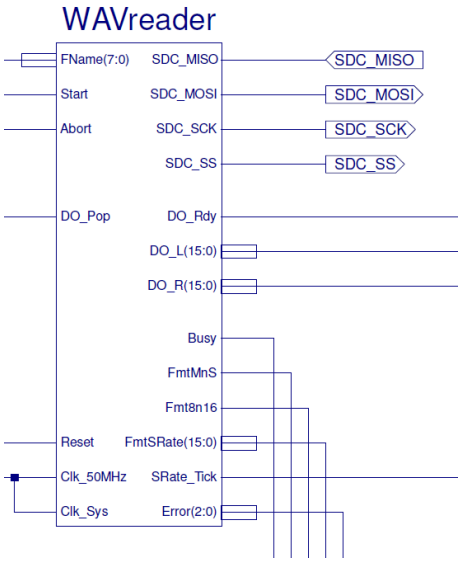


Rysunek 2: Schemat modułów

2..2 Najistotniejsze moduły

WAVReader

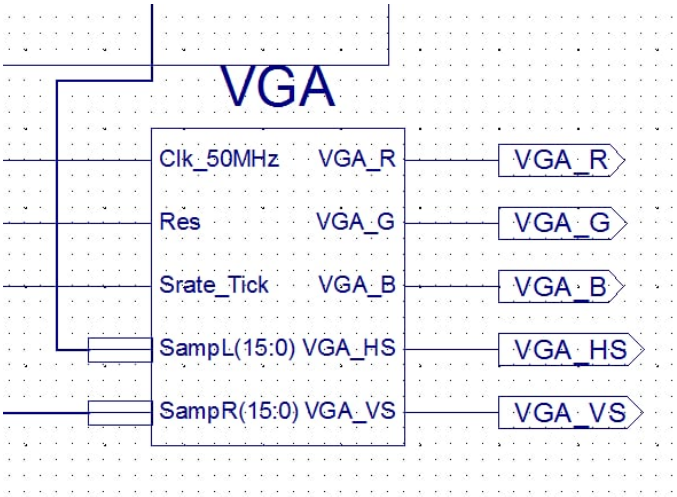
Moduł WAVReader został pobrany ze strony kursu [8]. W projekcie pełni funkcje odczytywania danych z plików z karty SD. Następnie dane są przesyłane do kolejnych modułów. Pełny schemat modułu znajduje się na Rysunek 3



Rysunek 3: Moduł WAVReader

VGA

Moduł VGA jest naszym autorskim modułem, został stworzony w celu generowania fali na monitorze poprzez interfejs VGA. Bazuje on na sygnałach SampL oraz Srate_Tick, na podstawie których generuje sygnały VGA_R, VGA_G, VGA_B, VGA_HS, VGA_VS przekazywane bezpośrednio do interfejsu VGA. Poniżej znajduje sie dokładny opis działania modułu wraz z fragmentami kodu. Schemat modułu znajduje się na Rysunek 4



Rysunek 4: Moduł WAVReader

1. **Inicjalizacja wartości początkowych**
   Początkową fazą programu jest inicjalizacja danych wejściowych, oraz ustawienie najistotniejszych rzeczy dla Horizontal timing (H_PX, H_BS, H_AS) oraz dla Vertical timing (V_PX, V_BS, V_AS) [12]. Dodatkowo następuje inicjalizacja liczników i zmiennych pomocniczych. Kod dostępny na listingu listing 1.

Listing 1: Inicjalizacja zmiennych

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

entity VGA is
    Port ( Clk_50MHz : in  STD_LOGIC;
           Res : in  STD_LOGIC;
           SampL : in  STD_LOGIC_VECTOR (15 downto 0);
           SampR : in  STD_LOGIC_VECTOR (15 downto 0);
           Srate_Tick : in  STD_LOGIC;
           VGA_R : out  STD_LOGIC;
           VGA_G : out  STD_LOGIC;
           VGA_B : out  STD_LOGIC;
           VGA_HS : out  STD_LOGIC;
           VGA_VS : out  STD_LOGIC);
end VGA;

architecture Behavioral of VGA is
    constant H_PX : unsigned(10 downto 0) := to_unsigned(1040, 11);
    constant H_BS : unsigned(10 downto 0) := to_unsigned(856, 11);
    constant H_AS : unsigned(10 downto 0) := to_unsigned(976, 11);

    constant V_PX : unsigned(10 downto 0) := to_unsigned(666, 11);
    constant V_BS : unsigned(10 downto 0) := to_unsigned(637, 11);
    constant V_AS : unsigned(10 downto 0) := to_unsigned(643, 11);

    signal H_COUNTER : unsigned(10 downto 0) := (others => '0');
    signal V_COUNTER : unsigned(10 downto 0) := (others => '0');
    signal WRITE_COUNTER : unsigned(10 downto 0) := (others => '0');

    signal V_ENABLE : std_logic := '0';
    signal VIDEO_ON : std_logic := '0';
```

2. **Pamięć ROM**
   Ważnym elementem naszego programu jest pamieć ROM 800 bitowej, która będzie przechowywać dane pobrane z WAVReadera. Fragment kodu dostepny w listingu listing 2.

Listing 2: Pamięć ROM

```
type rom_type is array (799 downto 0) of std_logic_vector (8 downto 0)
    ;
    signal ROM : rom_type := (others => "011111111");

     signal WRITE_ENABLE : std_logic := '0';
```

3. **Proces zapisu do pamięci ROM**
   Proces ten monitoruje sygnały Clk_50MHz (zegar) oraz Res (reset). Gdy Res jest aktywny (1), sygnał WRITE_ENABLE jest ustawiany na 0, co wyłącza zapis do pamięci ROM. Jeśli sygnał Srate_Tick jest aktywny (1), WRITE_ENABLE jest ustawiany na 1, co włącza zapis do pamięci ROM. W przeciwnym razie WRITE_ENABLE pozostaje wyłączony (0). Kod dostępny na listingu listing 3.

Listing 3: Proces zapisu do pamięci ROM

```
process(Clk_50MHz, Res)
begin
    if Res = '1' then
        WRITE_ENABLE <= '0';
    elsif rising_edge(Clk_50MHz) then
        if Srate_Tick = '1' then
            WRITE_ENABLE <= '1';
        else
            WRITE_ENABLE <= '0';
        end if;
    end if;

end process;
```

4. **Zapis próbek do pamięci ROM**
   Proces ten jest wyzwalany sygnałem Clk_50MHz. Jeśli WRITE_ENABLE jest aktywny (1), WRITE_COUNTER jest inkrementowany. Najbardziej znaczące bity (MSB) sygnału SampL (od 15 do 7) są zapisywane do pamięci ROM w pozycji wskazanej przez WRITE_COUNTER. Kod dostępny na listingu listing 4.

Listing 4: Zapis próbek do pamięci ROM

```
process(Clk_50MHz)
begin
    if rising_edge(Clk_50MHz) and WRITE_ENABLE = '1' then
        WRITE_COUNTER <= WRITE_COUNTER + 1;
        ROM(to_integer(WRITE_COUNTER)) <= SampL(15 downto 7);  --
            Zapisanie 9 MSB do ROM
    end if;
end process;
```

5. **Proces synchronizacji poziomej (H_COUNTER)**
   Proces ten monitoruje sygnały Clk_50MHz i Res. Gdy Res jest aktywny (1), licznik H_COUNTER jest resetowany do zera. Sygnał Clk_50MHz powoduje, że H_COUNTER jest inkrementowany. Jeśli H_COUNTER osiągnie wartość H_PX - 1 (1040 - 1), jest resetowany do zera, a sygnał V_ENABLE jest ustawiany na 1, sygnalizując konieczność inkrementacji licznika pionowego (V_COUNTER). Kod dostępny na listingu listing 5.

Listing 5: Proces synchronizacji poziomej

```
process(Clk_50MHz, Res)
begin
    if Res = '1' then
        H_COUNTER <= (others => '0');
    elsif rising_edge(Clk_50MHz) then
        if H_COUNTER = H_PX - 1 then
            H_COUNTER <= (others => '0');
            V_ENABLE <= '1';
        else
            H_COUNTER <= H_COUNTER + 1;
            V_ENABLE <= '0';
        end if;
    end if;
end process;
```

6. **Proces synchronizacji pionowej (V_COUNTER)**
Proces ten monitoruje sygnały Clk_50MHz i Res. Gdy Res jest aktywny (1), licznik V_COUNTER jest resetowany do zera. Sygnał CLK_50MHz powoduje, że V_COUNTER jest inkrementowany, jeśli sygnał V_ENABLE jest aktywny (1). Jeśli V_COUNTER osiągnie wartość V_PX - 1 (666 - 1), jest resetowany do zera. Kod dostępny na listingu listing 6.

Listing 6: Proces synchronizacji pionowej

```
process(Clk_50MHz, Res)
begin
    if Res = '1' then
        V_COUNTER <= (others => '0');
    elsif rising_edge(Clk_50MHz) and V_ENABLE = '1' then
        if V_COUNTER = V_PX - 1 then
            V_COUNTER <= (others => '0');
        else
            V_COUNTER <= V_COUNTER + 1;
        end if;
    end if;
end process;
```

7. **Generowanie sygnałów synchronizacji poziomej (VGA_HS) i pionowej (VGA_VS)**
Sygnał synchronizacji poziomej (VGA_HS) jest aktywny (0) gdy H_COUNTER znajduje się w zakresie od H_BS (856) do H_AS (976). Sygnał synchronizacji pionowej (VGA_VS) jest aktywny (0) gdy V_COUNTER znajduje się w zakresie od V_BS (637) do V_AS (643). Kod dostępny na listingu listing 7.

Listing 7: Generowanie sygnałów synchronizacji poziomej i pionowej

```
VGA_HS <= '0' when (H_COUNTER >= H_BS and H_COUNTER < H_AS) else '1';
VGA_VS <= '0' when (V_COUNTER >= V_BS and V_COUNTER < V_AS) else '1';
```

8. **Kontrola sygnału wyjścia wideo (VIDEO_ON)**
Sygnał VIDEO_ON jest aktywny (1) gdy H_COUNTER i V_COUNTER są w zakresie widocznych pikseli (800x600). Poza tym zakresem VIDEO_ON jest nieaktywny (0). Kod dostępny na listingu listing 8.

Listing 8: Kontrola sygnału wyjścia wideo

```
VIDEO_ON <= '1' when (((to_integer(H_COUNTER) >= 0) and (to_integer(
    H_COUNTER) < 800))
                  and ((to_integer(V_COUNTER) >= 0)  and (to_integer(
                      V_COUNTER) < 600))) else '0';
```

9. **Generowanie sygnałów RGB (VGA_R, VGA_G, VGA_B)**
Proces ten kontroluje sygnały VGA_R, VGA_G i VGA_B na podstawie wartości H_COUNTER, V_COUNTER i VIDEO_ON. Gdy VIDEO_ON jest aktywny (1), sygnały RGB są ustawiane na 1 lub 0 w zależności od wartości H_COUNTER i V_COUNTER w stosunku do danych zapisanych w pamięci ROM. Gdy VIDEO_ON jest nieaktywny (0), sygnały RGB są ustawiane na 0. Kod dostępny na listingu listing 9.

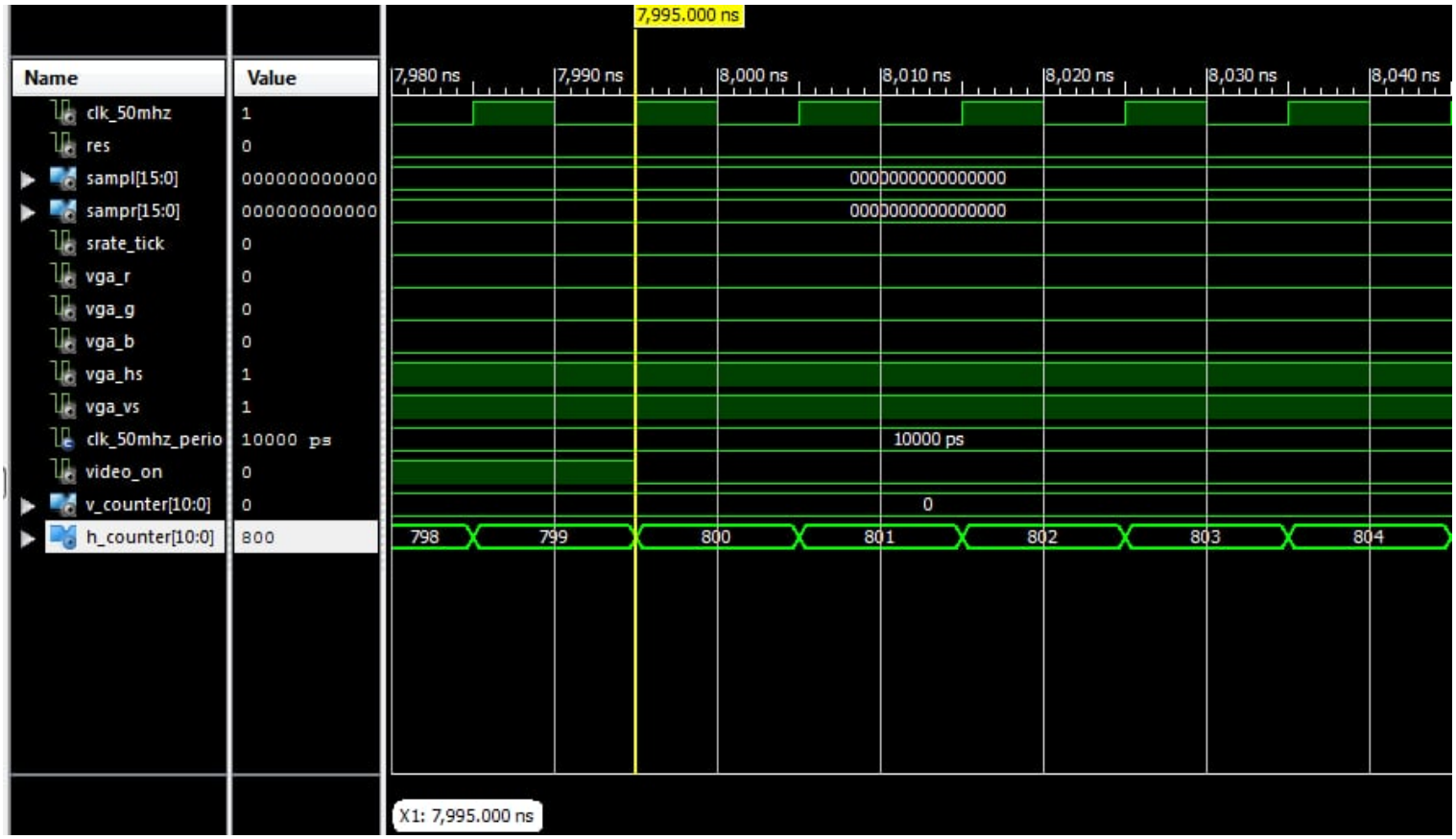
Listing 9: Generowanie sygnałów RGB

```
process (H_COUNTER, V_COUNTER, VIDEO_ON)
begin
     if VIDEO_ON = '1' then
          if H_COUNTER < 800 and V_COUNTER < 600 then
                if V_COUNTER = unsigned(ROM(to_integer(H_COUNTER)))
                    then
                      VGA_R <= '1';
                      VGA_G <= '1';
                      VGA_B <= '1';
                else
                      VGA_R <= '0';
                      VGA_G <= '0';
                      VGA_B <= '0';
                end if;
          else
                VGA_R <= '0';
                VGA_G <= '0';
                VGA_B <= '0';
          end if;
     else
          VGA_R <= '0';
          VGA_G <= '0';
          VGA_B <= '0';
     end if;
end process;
```

2..3 Symulacje

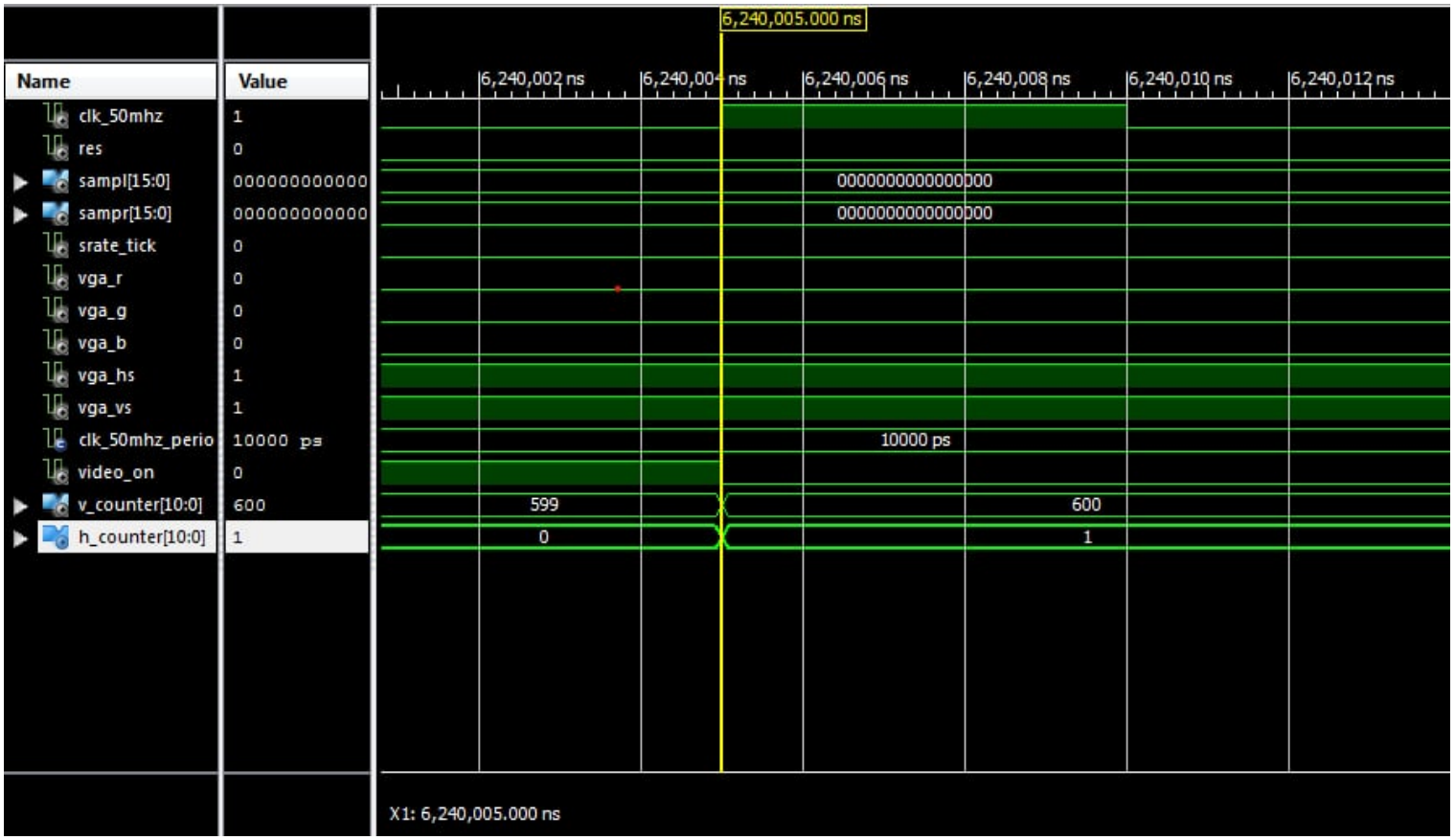
Przykładowe symulacje, które pozwoliły nam sprawdzić poprawność działania programu:

- Symulacja przedstawiona na Rysunek 5 sprawdza poprawność generowania sygnału video_on dla H_Counter .



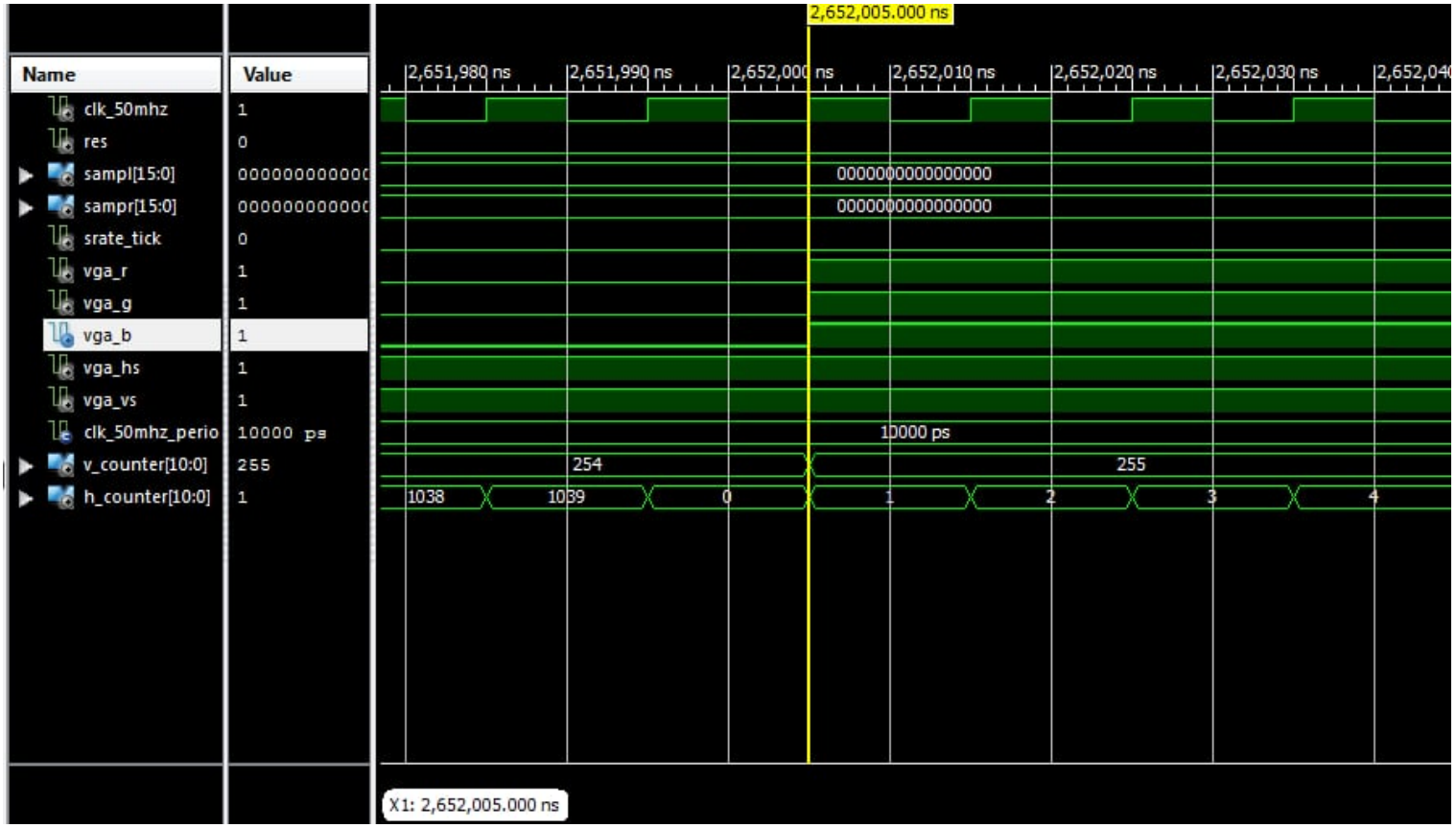
Rysunek 5: Symulacja sprawdzająca poprawność generowania sygnału video_on dla H_Counter

- Symulacja przedstawiona na Rysunek 6 sprawdzająca poprawność generowania sygnału video_on dla V_Counter.



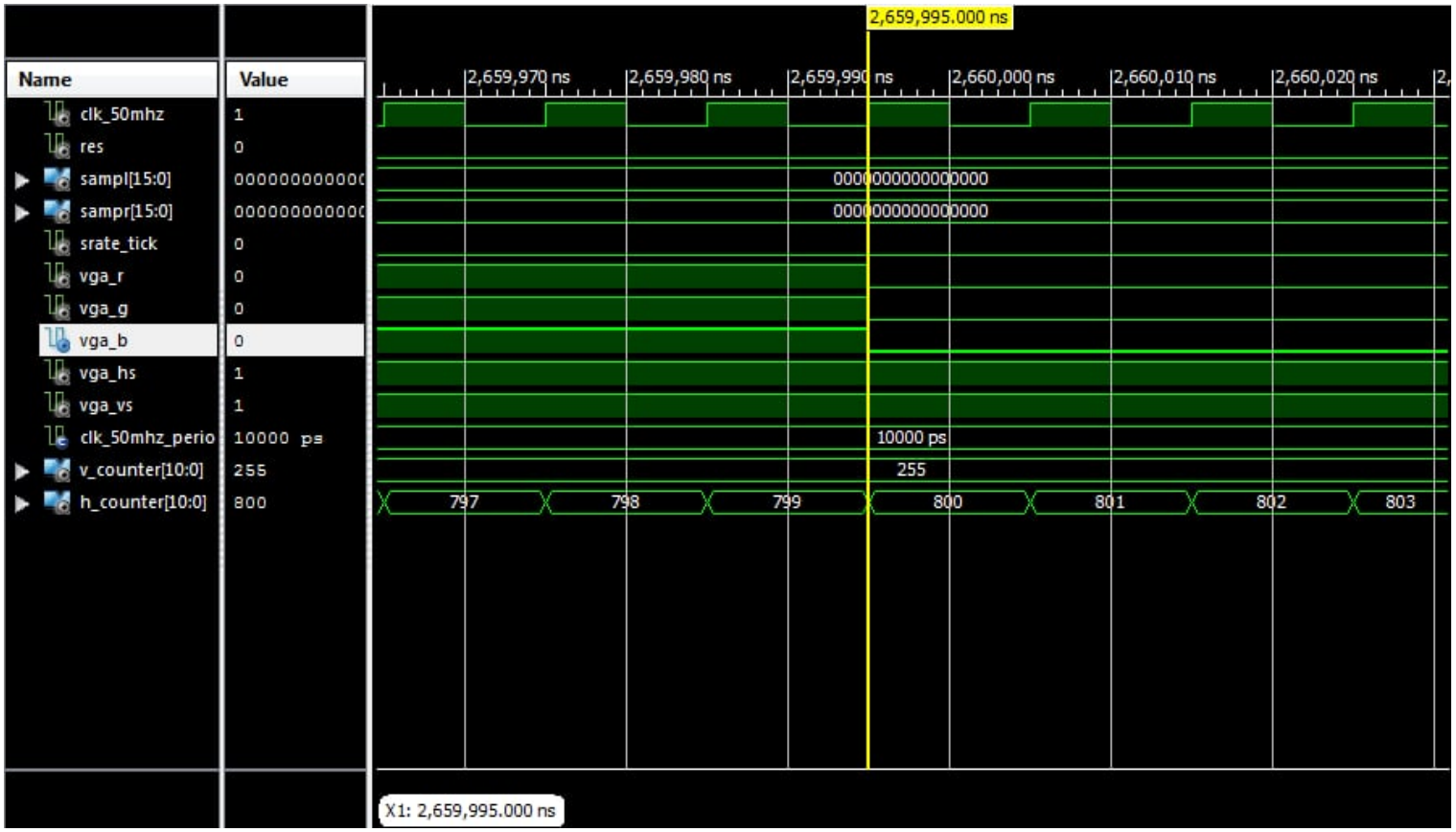
Rysunek 6: Symulacja sprawdzająca poprawność generowania sygnału video_on dla V_Counter

- Symulacja przedstawiona na Rysunek 7 rozpoczyna rysowanie na danej wysokości testowej 255.



Rysunek 7: Symulacja rozpoczynająca rysowanie linii na wysokości 255

- Symulacja przedstawiona na Rysunek 8 kończy rysowanie przykładowej linii na wysokości 255.



Rysunek 8: Symulacja kończąca rysowanie przykładowej linii na wysokości 255

3. Implementacja

3..1 Raporty

Rozmiar

Najważniejsze dane z raportu to:

- **LUTs** - 3216kb
- **Slices** - 1794kb

Pełny raport pamięci można zobaczyć na Rysunek 9

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	835	9,312	8%	
Number of 4 input LUTs	3,032	9,312	32%	
Number of occupied Slices	1,794	4,656	38%	
Number of Slices containing only related logic	1,794	1,794	100%	
Number of Slices containing unrelated logic	0	1,794	0%	
Total Number of 4 input LUTs	3,216	9,312	34%	
Number used as logic	2,130			
Number used as a route-thru	184			
Number used for Dual Port RAMs	900			
Number used as Shift registers	2			
Number of bonded IOBs	40	232	17%	
Number of RAMB16s	1	20	5%	
Number of BUFGMUXs	1	24	4%	
Average Fanout of Non-Clock Nets	4.36			

Rysunek 9: Raport rozmiaru użytej pamięci

Prędkość

Najważniejsze dane z raportu to:

- **Worst Case Slack** - 8.210ns
- **Best Case Achievable** - 11.790ns

Pełny raport prędkości mozna zobaczyć na Rysunek 10

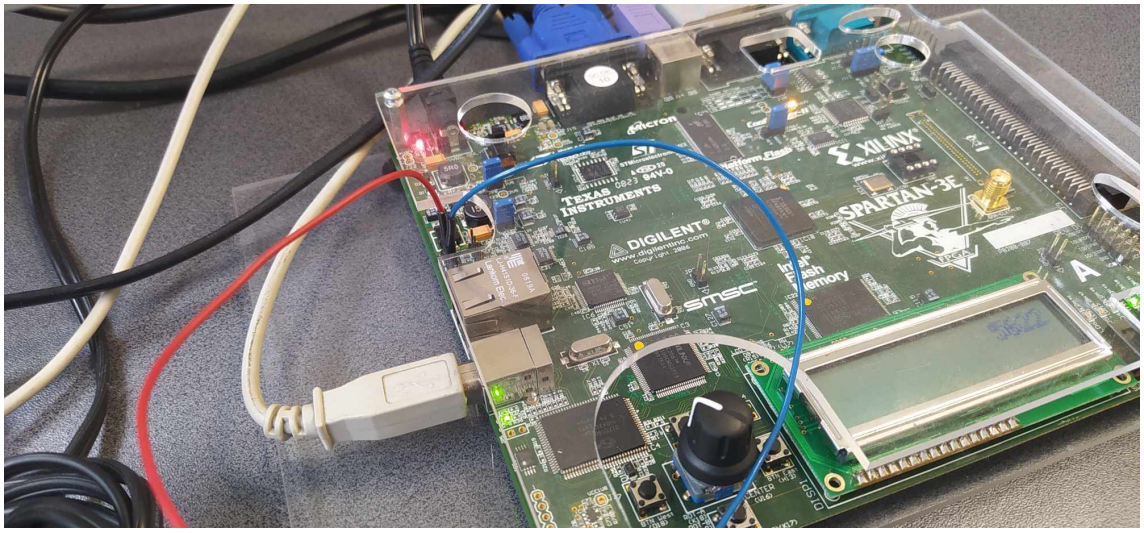
	Constraint	Check	Worst Case Slack	Best Case Achievable	Timing Errors	Timing Score
Yes	NET "Clk_50MHz_BUFGP/IBUFG" PERIOD = 20 ns HIGH 50%	SETUP HOLD	8.210ns 0.815ns	11.790ns	0 0	0 0

Rysunek 10: Raport prędkości implementacji

3..2 Podręcznik obsługi urządzenia

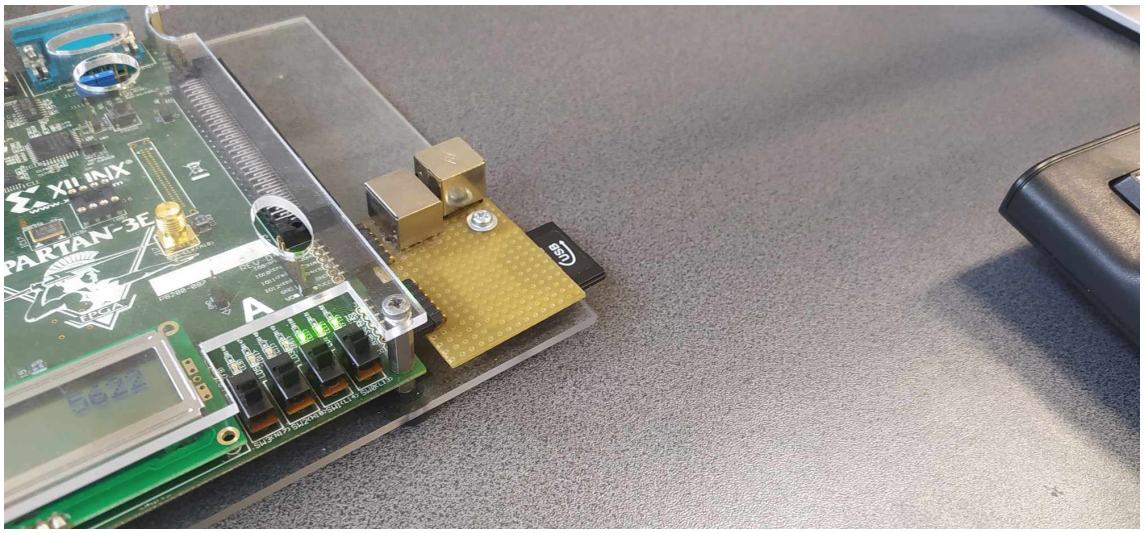
Podłączenie sprzętu

Pierwszą czynnością jaką należy zrobić, jest podłączenie głośnika analogowego do płyty Spartan-3E. Należy podłączyć przewód sygnałowy do pinu 5 lub 6 znajdującego się na lewej stronie płyty. Przewód masowy należy podłączyć pod pin oznaczony napisem 'GND'. Obrazek podglądowy poprawnego podłączenia znajduje się na Rysunek 11.



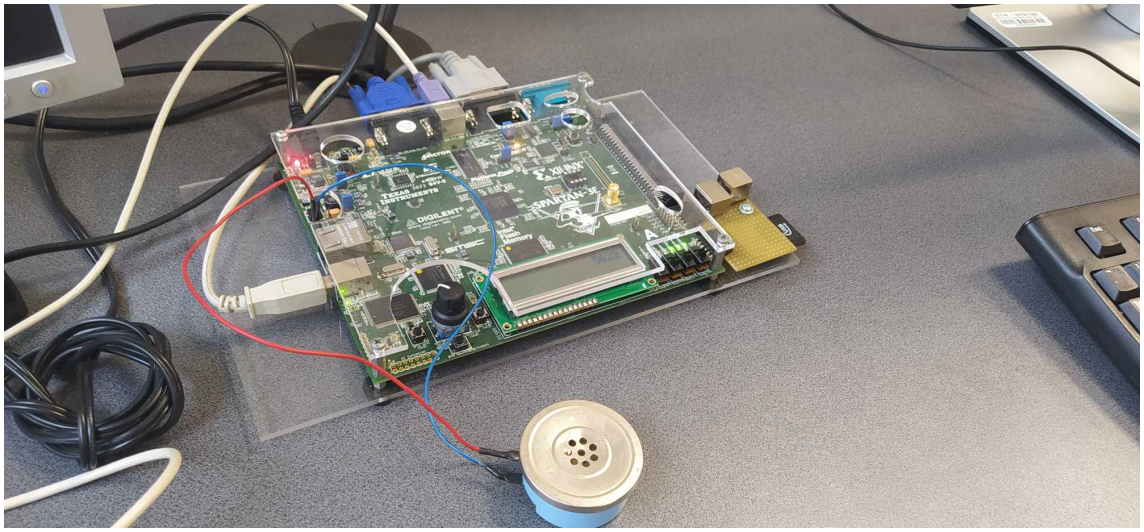
Rysunek 11: Poprawne podłączenie głośnika

Następną czynnością, którą należy zrobić w celu poprawnego podłączenia sprzętu, jest umieszczenie karty SD, z wgranymi plikami .wav do czytnika kart SD. Czytnik ten znajduje się z prawej strony płyty. Kartę wkładamy odwrotnie do jej neutralnej pozycji. Obrazek podglądowy poprawnego podłączenia znajduje się na Rysunek 12



Rysunek 12: Poprawne podłączenie karty SD

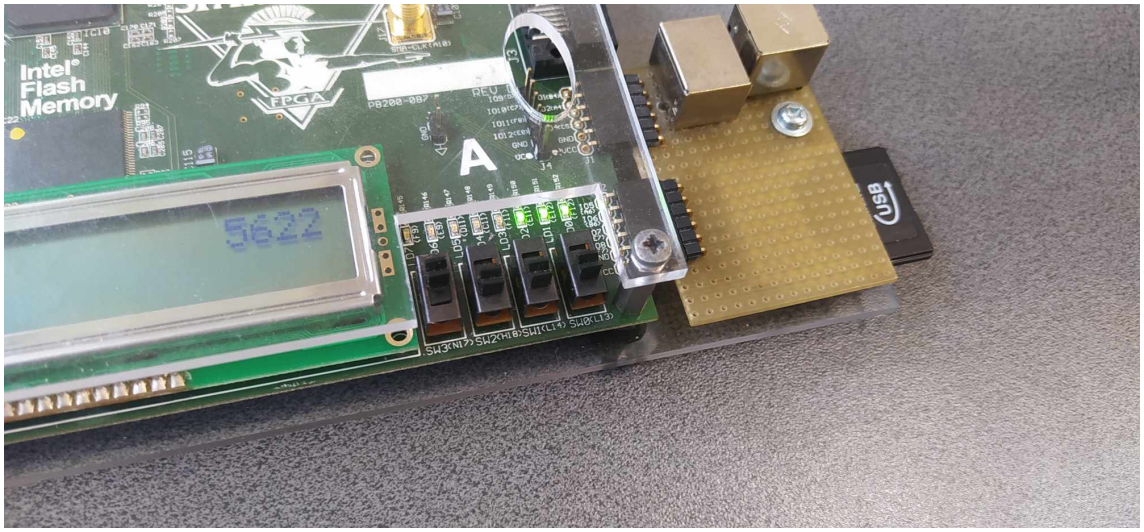
Ostatnią ważną czynnością jest podłączenie monitora do portu VGA na płycie. Port znajduje się na samej górze płyty. Obrazek podglądowy z poprawnie podłączonymi wszystkimi urządzeniami znajduje się na Rysunek 13



Rysunek 13: Poprawne podłączone urządzenia

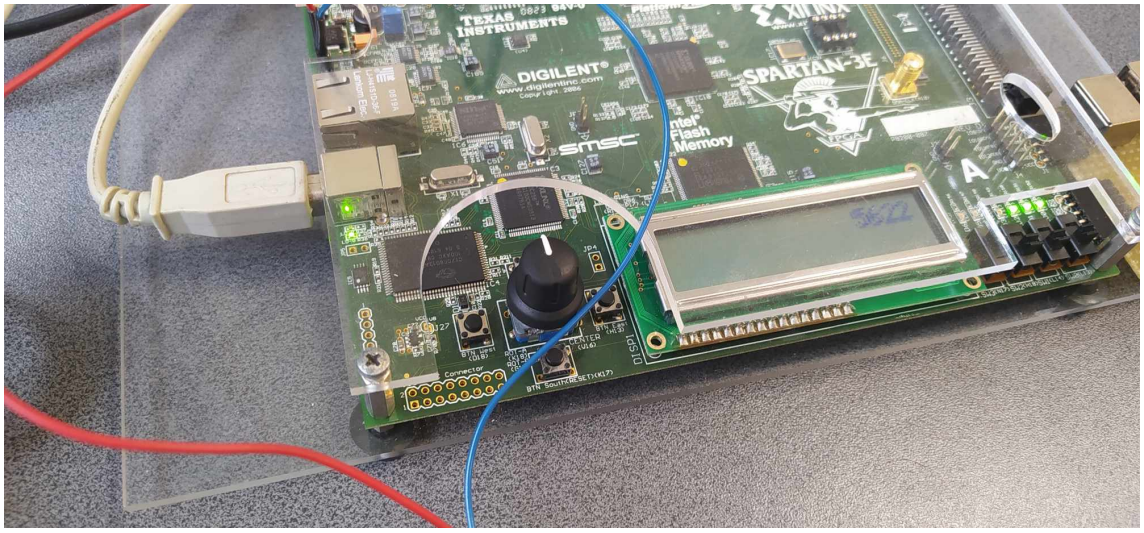
Obsługa programu

Po poprawnym podłączeniu urządzeń zewnątrznych, można przejść do obsługi programu. W tym celu pierwszą czynnością jest wybór pliku do wczytania z karty SD. Służą do tego cztery przełączniki znajdujące się w prawym dolnym rogu płyty. Wybór pliku polega na włączeniu odpowiedniej ilości przełaczników. Liczenie opiera sie o system binarny, przy czym lewy skrajny przełącznik oznacza najstarszy bit 2^3. Przykładowo włączony przełącznik 1 i 3 binarnie da sumę 5, czyli plik o nazwie 5.wav zostanie wybrany do wczytania danych. Zdjęcie podglądowe znajduje się na Rysunek 14



Rysunek 14: Przełączniki do wyboru pliku

Następnie wymagana jest obsługa potencjometru znajdującego się w lewym dolnym rogu płyty. Przesunięcie pokrętła w lewo rozpocznie czytanie danych z pliku oraz uruchomi wyświetlanie fali na monitorze. Natomiast przesunięcie pokrętła w prawo, przerwie działanie programu, co za tym idzie, czytanie pliku oraz wyświetlanie fali. Zdjęcie podglądowe znajduje się na Rysunek 15



Rysunek 15: Potencjometr do sterowania programem

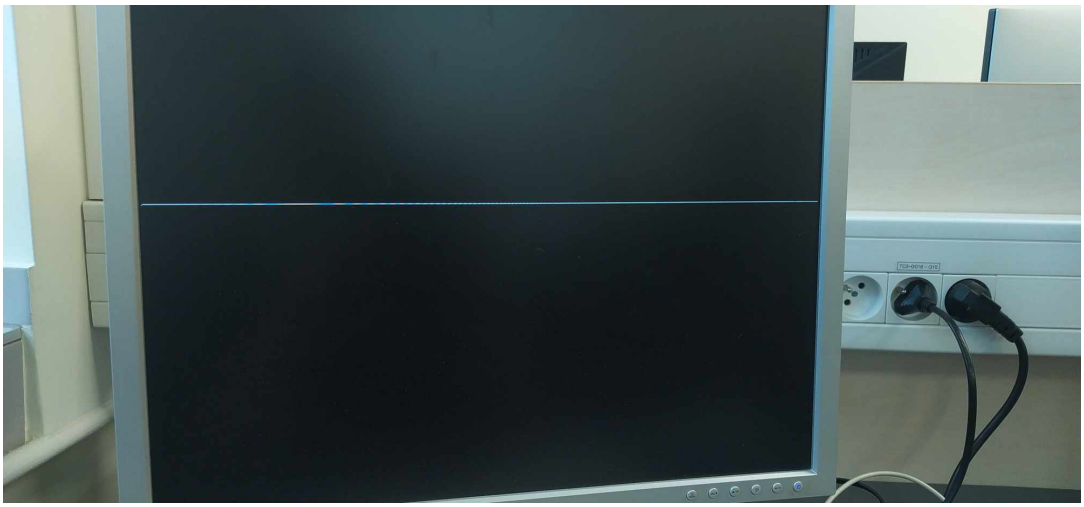
4. Podsumowanie

4..1 Ocena krytyczna

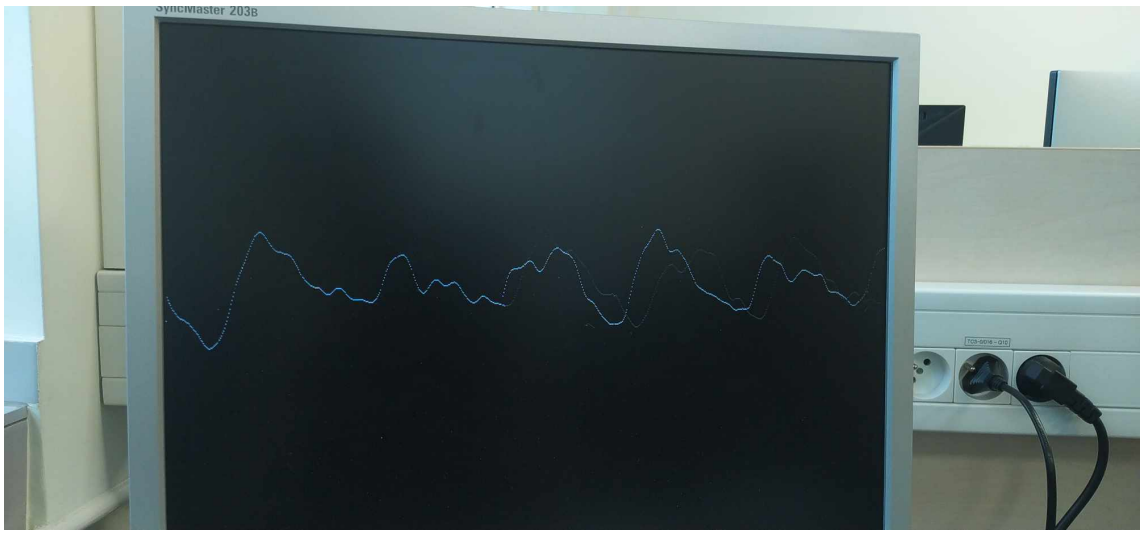
Podsumowując naszą pracę, udało nam się zaimplementować większość planowanych funkcjonalności takich jak:

- Obsługa VGA
- Wczytywanie danych z WAV do pamięci ROM
- Generowanie fali na podstawie danych oraz wyświetlanie jej na monitorze

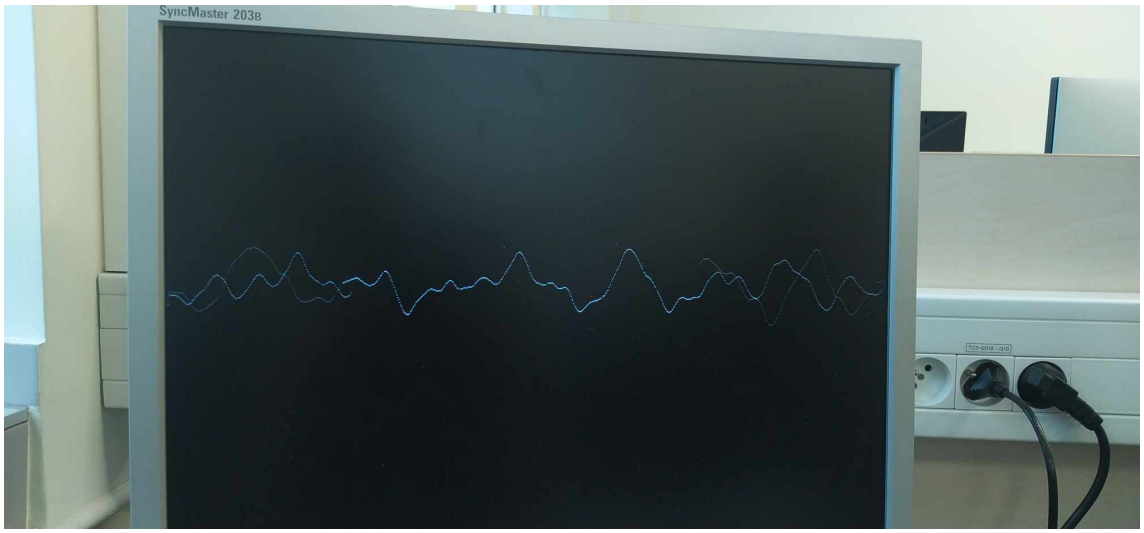
Jeżeli byśmy mogli zacząć projekt jeszcze raz, z aktualną wiedzą, zrobilibyśmy kilka rzeczy lepiej i sprawniej. Dużo czasu zmarnowaliśmy na próbie wyświetlania danych zczytanych z WAV bezpośrednio na port VGA, zamiast skorzystać z pamięci ROM. Dużo czasu zajęło nam też, zrozumienie i opanowanie działania interfejsu VGA.
Efekty naszej pracy można zobaczyć na Rysunek 16, Rysunek 17, Rysunek 18



Rysunek 16: Wyświetlanie fali - Stan bezczynności



Rysunek 17: Wyświetlanie fali - W trakcie



Rysunek 18: Wyświetlanie fali - W trakcie

4..2 Dalsze prace

Następną rzeczą do wprowadzenia, na którą zabrakło nam już czasu, jest wyzwalacz zapisu, który pomógłby ustabilizować wyświetlaną falę. Dobrym pomysłem byłoby również wprowadzenie możliwości zapałzowania wyświetlania fali.
Kod źródłowy dostępny na github [13]

Spis rysunków

Listings

Bibliografia

[1] Wikipedia. Vga. https://en.wikipedia.org/wiki/Video_Graphics_Array, 2024.

[2] SECONS Ltd. Vga signal timing. http://www.tinyvga.com/vga-timing, 2008.

[3] Wikipedia. Resource interchange file format. https://en.wikipedia.org/wiki/Resource_Interchange_File_Format, 2024.

[4] Wikipedia. Wav. https://en.wikipedia.org/wiki/WAV, 2024.

[5] Xilinx. *Spartan-3E FPGA Starter Kit Board User Guide*. AMD, 2011. https://docs.amd.com/v/u/en-US/ug230.

[6] Xilinx. Xc3s500e. https://www.amd.com/en.html, 2008.

[7] dr. inż. Jarosław Sugier. Rotaryenc. https://fpga.iiar.pwr.edu.pl/Spartan3E/#_Toc99974101, 2008.

[8] dr. inż. Jarosław Sugier. Wavreader. https://fpga.iiar.pwr.edu.pl/Spartan3E/#_Toc99974115, 2008.

[9] dr. inż. Jarosław Sugier. Fsm sendsamples. https://fpga.iiar.pwr.edu.pl/Spartan3E/#_Toc99974133, 2008.

[10] dr. inż. Jarosław Sugier. Dacwrite. https://fpga.iiar.pwr.edu.pl/Spartan3E/#_Toc99974110, 2008.

[11] dr. inż. Jarosław Sugier. Lcd1x64. https://fpga.iiar.pwr.edu.pl/Spartan3E/#_Toc99974107, 2008.

[12] SECONS Ltd. Vesa signal 800 x 600 @ 72 hz timing. http://www.tinyvga.com/vga-timing/800x600@72Hz, 2008.

[13] Kamil Pawelski Adam Troszczyński. Kod źródłowy. https://github.com/Kamil-Pawelski/Uk-ady/tree/main, 2024.