Układy Cyfrowe i Systemy Wbudowane 2

Prowadzący: Dr inż. Dariusz Banasiak, wtorek 11:15 TP

Gra Pong

Michał Madarasz 238903

Piotr Borowski 234996

04 czerwiec 2019

Spis treści

1 Wprowadzenie

1.1 Cel i zakres

Celem projektu jest zaprojektowanie oraz zaimplementowanie przy użyciu języka VHDL prostej gry działającej na układzie Spartan-3E (XC3S500E). Sterowanie w grze będzie odbywać się za pomocą enkodera, a wyświetlanie obrazu gry przy użyciu monitora VGA.

Projekt zostanie przeprowadzony w sposób iteracyjny, z podziałem na etapy mające na celu stopniowe poznawanie potrzebnych technologii i ich zastosowanie w projekcie. Można wyróżnić następujące etapy:

1. Nawiązanie komunikacji z monitorem VGA
2. Nawiązanie komunikacji z enkoderem
3. Określenie oraz implementacja mechaniki gry

Każdemu etapowi będą towarzyszyły testy, mające na celu sprawdzenie, czy kolejne funkcjonalności zostały zrealizowane poprawnie oraz testy weryfikujące grywalność gry.

1.2 Mechanika gry

Podczas projektu postaramy się utworzyć grę zręcznościową, w której gracz będzie przy pomocy enkodera poruszał paletką.

1.2.1 Cel gry

Gracz będzie miał na celu odbijanie wygenerowanej piłeczki tak, by nie znalazła się ona poniżej położenia paletki. Piłeczka będzie odbijała się od krawędzi ekranu. Po przegranej gra zaczyna się od początku.

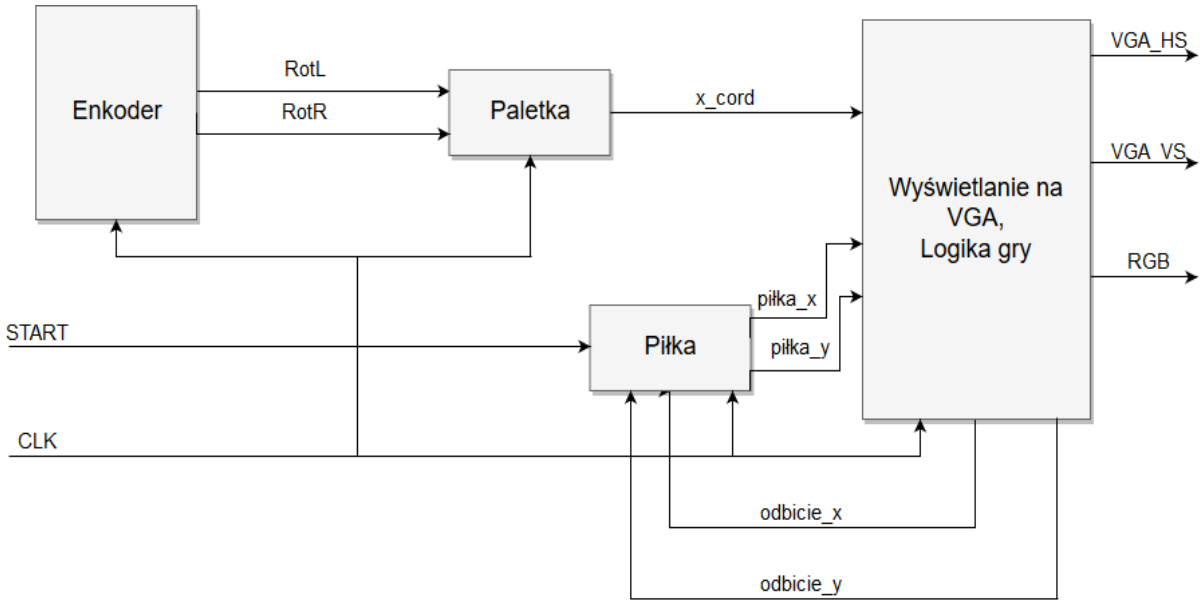
1.2.2 Statystyki

Gra będzie wyświetlała na ekranie statystyki gry: liczbę odbić, liczbę przegranych.

1.3 Sprzęt

Projekt został zaimplementowany na płycie Spartan-3E (UG230). Z punktu widzenia projektu istotnymi układami umieszczonymi na tej płycie były: układ FPGA XC3S500E, zegar 50MHz, port VGA oraz RotaryEncoder.

1.4 Schemat logiczny

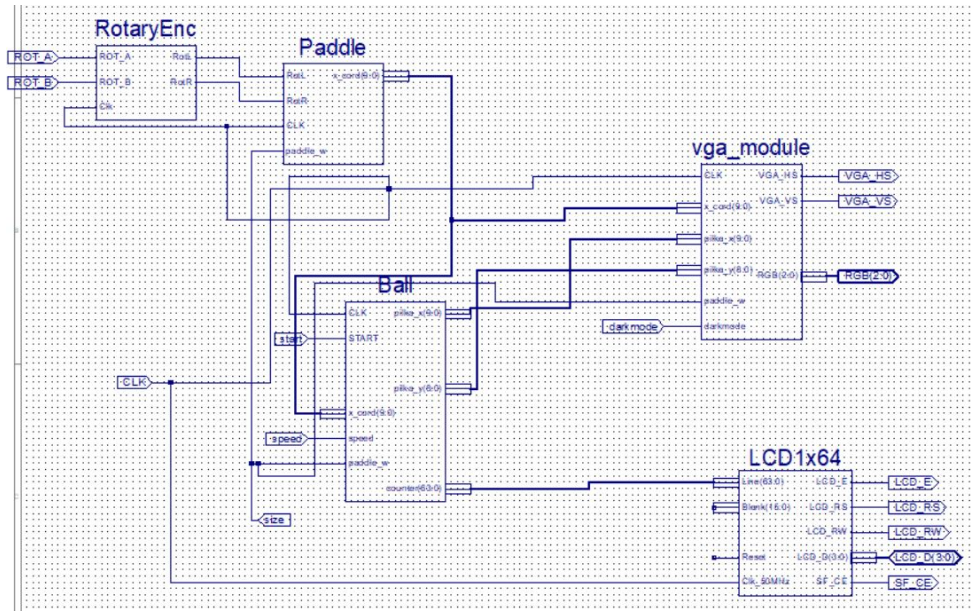


Sygnały:

- START (std_logic) – sygnał określający czy piłka ma się poruszać
- CLK (std_logic) – zegar
- RotL (std_logic) – obrót w lewo
- RotR (std_logic) – obrót w prawo
- x_cord (std_logic_vector(9 downto 0)) – położenie paletki w osi X
- pilka_x (std_logic_vector(9 downto 0)) – położenie piłki w osi X
- pilka_y (std_logic_vector(8 downto 0)) – położenie piłki w osi Y
- odbicie_x (std_logic) - zmiana kierunku w osi X
- odbicie_y (std_logic) - zmiana kierunku w osi Y
- VGA_HS (std_logic) – synchronizacja pozioma
- VGA_VS (std_logic) – synchronizacja pionowa
- RGB (std_logic_vector(2 downto 0)) – kolor piksela

2 Implementacja

2.1 Główny Schemat



Rysunek 1: Główny schemat

2.2 Moduł Ball

2.2.1 Opis

Moduł Ball odpowiada za mechanikę piłki, to znaczy:

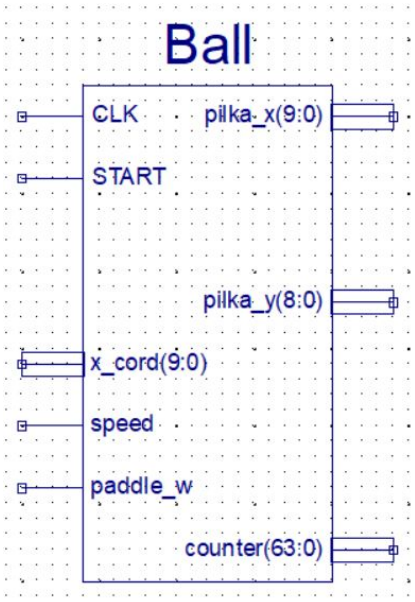
- poruszanie się piłki po ekranie;
- sprawdzanie odbicia z paletką;
- sprawdzanie przegranej.
- zatrzymanie piłki

2.2.2 Schemat

```
entity Ball is
    Port ( CLK : in  STD_LOGIC;
           START : in  STD_LOGIC;
           x_cord : in STD_LOGIC_VECTOR (9 downto 0);
           speed : in STD_LOGIC;
```

```
        paddle_w: in STD_LOGIC;
        pilka_x : out  STD_LOGIC_VECTOR (9 downto 0);
        pilka_y : out  STD_LOGIC_VECTOR (8 downto 0);
        counter : out STD_LOGIC_VECTOR (63 downto 0)
        );
end Ball;
```

Listing 1: Sygnały modułu Ball



Rysunek 2: Schemat modułu Ball

- CLK - zegar 50 MHz
- START - start/zatrzymanie ruchu piłki
- x_cord - współrzędna paletki
- speed - wybór szybkości piłki
- paddle_w - szerokość paletki
- pilka_x - współrzędna x piłki
- pilka_y - współrzędna y piłki
- counter - licznik odbić

2.2.3 Kod

```
if pilka_x1 + pilka_r1 >= to_integer(unsigned(x_cord)) and
pilka_x1 - pilka_r1 <= to_integer(unsigned(x_cord)) + paletka_width and
pilka_y1 + pilka_r1 >= paletka_y and
pilka_y1 <= paletka_y + paletka_height and
moveunit_y > 0 then

    moveunit_y <= moveunit_y * (-1);
    point_counter <= std_logic_vector( unsigned(point_counter) + 1 );
    counter <= point_counter;

elsif ((pilka_y1 + pilka_r1 >= paletka_y and
pilka_y1 + pilka_r1 <= paletka_y + paletka_height) or
(pilka_y1 - pilka_r1 >= paletka_y and
pilka_y1 - pilka_r1 <= paletka_y + paletka_height) ) and
pilka_x1 + pilka_r1 >= to_integer(unsigned(x_cord)) and
pilka_x1 - pilka_r1 <= to_integer(unsigned(x_cord)) + paletka_width
and moveunit_y > 0
        then

    moveunit_x <= moveunit_x * (-1);
    point_counter <= std_logic_vector( unsigned(point_counter) + 1 );
    counter <= point_counter;

end if;
```

Listing 2: Sprawdzenie odbicia z paletką

2.3 Moduł Paddle

2.3.1 Opis

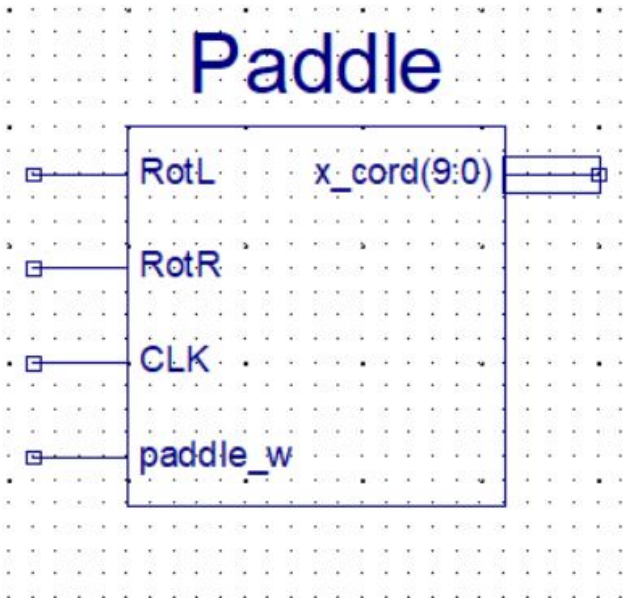
Moduł Paddle obsługuje mechanikę paletki, to znaczy:

- zmianę położenia paletki przy pomocy Rotary Encoder;
- przekazywanie współrzędnych paletki do modułu wyświetlającego.

2.3.2 Schemat

```
entity Paddle is
    Port ( RotL : in  STD_LOGIC;
           RotR : in  STD_LOGIC;
           CLK : in  STD_LOGIC;
           paddle_w : in STD_LOGIC;
           x_cord : out  STD_LOGIC_VECTOR (9 downto 0));
end Paddle;
```

Listing 3: Sygnały modułu Paddle



Rysunek 3: Schemat modułu Paddle

- RotL - sygnał ruchu w lewo
- RotR - sygnał ruchu w prawo
- CLK - zegar 50 MHz

- paddle_w - szerokość paletki
- x_cord - współrzędna paletki

2.3.3 Kod

```
if rising_edge(CLK) then
   if RotL = '1' then
      if xpaddle <= x_min then
         xpaddle <= xpaddle;
      else
         xpaddle <= xpaddle - 10;
      end if;
   end if;

   if RotR = '1' then
      if paddle_w = '1'
      then
         paddle_width := 80;
      else
         paddle_width := 50;
      end if;

      if xpaddle + paddle_width >= x_max then
         xpaddle <= xpaddle;
      else
         xpaddle <= xpaddle + 10;
      end if;
   end if;

   x_cord <= std_logic_vector(to_unsigned(xpaddle, x_cord'length));
end if;
```

Listing 4: Zmiana położenia paletki

2.4 Moduł VGA_Module

2.4.1 Opis

Moduł VGA_Module obsługuje wyświetlanie na monitorze VGA oraz zmianę kolorów wyświetlania.

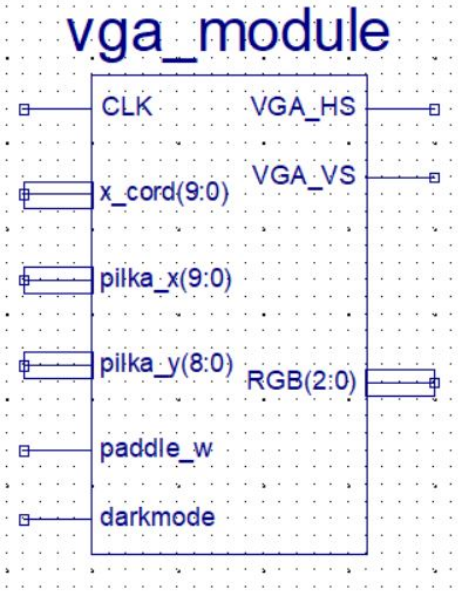
Moduł VGA_Module generuje sygnały czasowe oraz synchronizacyjne. Sygnały h_sync oraz v_sync są służą do kontrolowania poziomych oraz pionowych skanów monitora. Te dwa sygnały są dekodowane przez wewnętrzny licznik, którego wyjściem są sygnały h_counter x oraz v_counter. Sygnały wyjściowe wskazują na względną pozycję skanowania i określają położenie bieżącego piksela.

2.4.2 Schemat

```
entity vga_module is
    Port ( x_cord : in  STD_LOGIC_VECTOR (9 downto 0);
           pilka_x : in  STD_LOGIC_VECTOR (9 downto 0);
           paddle_w : in STD_LOGIC;
           pilka_y : in  STD_LOGIC_VECTOR (8 downto 0);
           CLK : in  STD_LOGIC;
           darkmode : in STD_LOGIC;
           VGA_HS : out  STD_LOGIC;
           VGA_VS : out  STD_LOGIC;
           RGB : out  STD_LOGIC_VECTOR (2 downto 0));
end vga_module;
```

Listing 5: Sygnały modułu VGA_Module

- CLK - zegar 50 MHz
- x_cord - współrzędna paletki
- pilka_x - współrzędna x piłki
- pilka_y - współrzędna y piłki
- paddle_w - szerokość paletki
- darkmode - wybór ciemnego motywu
- VGA_HS - sygnał synchronizacji poziomej
- VGA_VS - sygnał synchronizacji pionowej
- RGB - kolor piksela



Rysunek 4: Schemat modułu VGA_Module

2.4.3 Kod

```
horizontal_sync : process(h_counter)
begin

   if h_counter >= 0 and h_counter <= h_sync then
      VGA_HS <= '0';

   else
      VGA_HS <= '1';

   end if;

end process;
```

Listing 6: Synchronizacja pozioma

```
if rising_edge(clk25) then


   if h_counter >= h_size then
      h_counter <= 0;

      if v_counter >= v_size then
         v_counter <= 0;
      else
         v_counter <= v_counter + 1;
      end if;

   else
      h_counter <= h_counter + 1;
   end if;



end if;
```

Listing 7: Licznik położenia w poziomie

```
if h_counter > h_back + h_sync and h_counter < h_size - h_front and
v_counter > v_back + v_sync and v_counter < v_size - v_front then

  --rysowanie paletki
  if h_counter >= h_back + h_sync + x_cord1 and
  h_counter <= h_back + h_sync + x_cord1 + paletka_width and
  v_counter >= paletka_y and v_counter <= paletka_y + paletka_height t

      RGB <= "001";

  --rysowanie pilki
  elsif h_counter >= pilka_x1 - pilka_r1 + h_back + h_sync and
  h_counter <= pilka_x1 + pilka_r1 + h_back + h_sync and
  v_counter >= pilka_y1 - pilka_r1 + v_back + v_sync and
  v_counter <= pilka_y1 + pilka_r1 + v_back + v_sync then

      RGB <= "100";

  else
```

```
        if darkmode = '1' then
            RGB <= "000";
        else
            RGB <= "111";
        end if;
    end if;

else
    RGB <= "000";
end if;
```

Listing 8: Rysowanie obiektów na ekranie

2.5 Inne moduły

Dodatkowo w projekcie znalazły się gotowe moduły obsługujące enkoder rotacyjny (RotaryEnc), za pomocą którego poruszana jest paletka oraz wyświetlacz LCD (LCD1x64), za pomocą którego wyświetlana jest liczba odbić piłki. Oba moduły pochodzą ze strony internetowej *www.zsk.ict.pwr.wroc.pl/zsk_ftp/fpga/*.

3 Wnioski

Logika gry została przeniesiona do modułu Ball (początkowo miała znajdować się w module VGA_Module). Nie udało się wykonać zliczania punktów, jedynie zliczanie odbić w postaci hexadecymalnej na wyświetlaczu LCD. Podczas wykonywania projektu zmagaliśmy się z problemem z wyświetlaniem, który zajął nam 2 zajęcia projektowe. Problem był skomplikowany do wykrycia, ponieważ był on związany z konwersją STD_LOGIC_VECTOR na signed_integer. Podczas konwersji należy pamiętać, że zmienia się zakres liczby i zamiast np. liczby od 0 do 255 uzyskujemy liczbę od -127 do 128. Problem został ostatecznie rozwiązany ale zabrał dużo czasu.