Urządzenia Cyfrowe i Systemy Wbudowane 2 - projekt

Michał Chojnacki 225936
Maciej Chyrc - 222112

Prowadzący:
dr inż. Jacek Mazurkiewicz
Grupa : Poniedziałek 10:00 tydzien nieparzysty

17 czerwca 2018

Spis treści

1 Wstęp

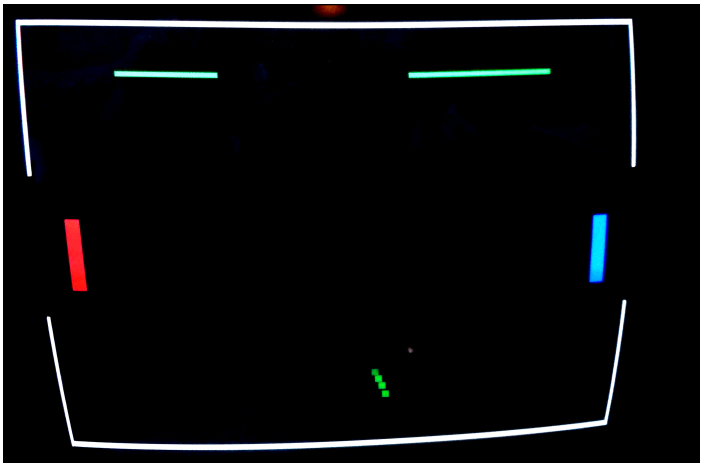
Celem projektu była implementacja gry PONG z użyciem języka VHDL i środowiska Xilinx na matrycy FPGA Spartan3E Starter Kit. Gra jest sterowana przy pomocy klawiatury, podłączonej do układu przy pomocy portu PS2. Pong jest grą imitującą tenis stołowy.

1.1 Założenia dotyczące rozgrywki

Dwaj gracze sterują w płaszczyźnie pionowej widzianymi "z góry" paletkami (ang. paddles) tak by odbijać piłkę w kierunku ograniczonego obszaru, nazywanego dalej bramką przeciwnika. Obaj gracze posiadają własny status żywotności, reprezentowany poziomą zieloną linią. Domyślnie gracze mają 3 "życia", które ulegają dekrementacji przy "stracie" bramki. Gra zostaje przywrócona do początkowego stanu, w przypadku gdy któryś z graczy straci wszystkie 3 "życia" co jest jednoznaczne z przegraną danego gracza.

1.2 Sterowanie

Sterowanie paletką 1 (gracz z lewej strony) odbywa się za pomocą klawiszy W i S. Sterowanie paletką 2 (gracz z prawej strony) odbywa się za pomocą klawiszy I i K.



Rysunek 1: Zrzut ekranu z gry. Widoczne są dwie paletki (czerwona i niebieska) odbijające zieloną piłkę. Białe linie wyznaczają obszar, w którym porusza się piłka. Zielone paski nad graczami są ich statusami żywotności.

1.3 Działanie gry

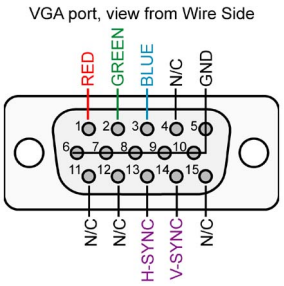
Piłka odbija się zarówno od paletek jak i od ścian. W zależności od odległości od środka paletki piłka może zmienić tor swojego ruchu z różną prędkością i z różnym kątem. Ten efekt można osiągnąć przez odpowiednie działania na zmiennych,reprezentujących współrzędne i prędkość piłki.

2 Implementacja gry

Przy implementacji gry można było wykorzystać gotowe moduły do obsługi klawiatury i wyświetlacza VGA, dostępnych na stronie Zakładu Systemów Komputerowych i Dyskretnych. W projekcie, będącym przedmiotem tej dokumentacji obsługa klawiatury i monitora zostały napisane ręcznie w celu dogłębnego zrozumienia ich działania oraz możliwości dostosowania sprzętu pod konkretne wymagania.

2.1 Wyświetlacz VGA

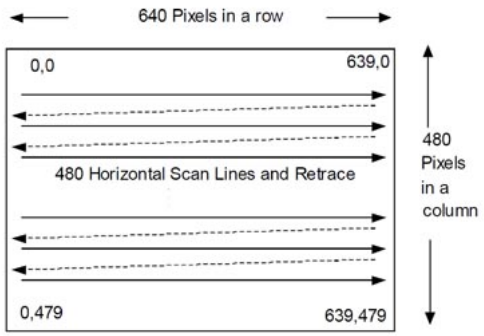
VGA (ang. Video Graphics Array) to standard kart graficznych, służący do wyświetlania kolorowych obrazów. Interfejs VGA przesyła 5 podstawowych sygnałów: H-sync, V-sync, RED, GREEN, BLUE. Przy rozdzielczości ekra-



Rysunek 2: Wyjście VGA z zaznaczonymi sygnałami

nu 640x480 potrzebujemy liczników modulo 800 dla współrzędnej x i modulo 525 dla współrzędnej y. W każdym cyklu odświeżania poziomego i pionowego, oprócz części widocznej, mamy obszar niewidoczny, na który składają się: back porch, sync pulse i front porch. Na początku każdego cyklu na początku odliczamy back porch i sync pulse, następnie jest obszar widoczny, gdzie ustawiamy barwy pikseli według naszego uznania i na końcu mamy front porch. Z tego względu aby łatwiej było wyliczać koordynaty obiektów na

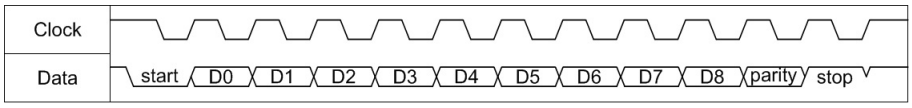
ekranie, stosujemy rekalkulację koordynatów w osi x odejmując 144 od wartości licznika pozycji x, a w osi y odejmując 35 od wartości licznika pozycji y. Zasadę odświeżania ekranu najlepiej ilustruje poniższa grafika.



Rysunek 3: Schemat ilustrujący generowanie obrazu na monitorze VGA o rozdzielczości 640x480 (widzialny obszar obrazu)

2.2 Klawiatura PS2

PS2 jest szeregowym portem komunikacyjnym, służącym do podłączania klawiatury i myszki. Jego działanie jest dużo prostsze niż działanie interfejsu VGA, ponieważ w przypadku PS2 są tylko dwa istotne sygnały - Clock i Data. Dane wczytywane z klawiatury kodowane są szesnastkowo. Na rzecz stero-



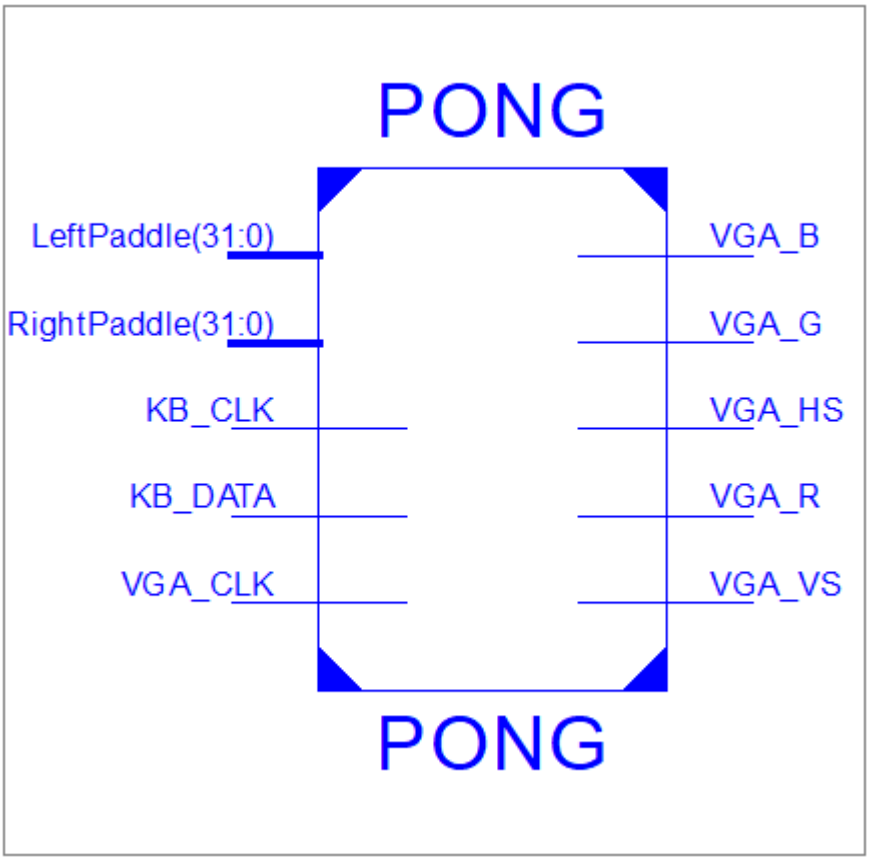
Rysunek 4: Czytanie danych z klawiatury PS2

wania paletek potrzeba jedynie 4 klawiszy W(1D), S(1B), I(43) oraz K(42). By interpretować zdarzenia z klawiatury należy zaimplementować również funkcję dekodowania na kod binarny.

2.3 Ważne segmenty kodu

Sygnały wejściowe

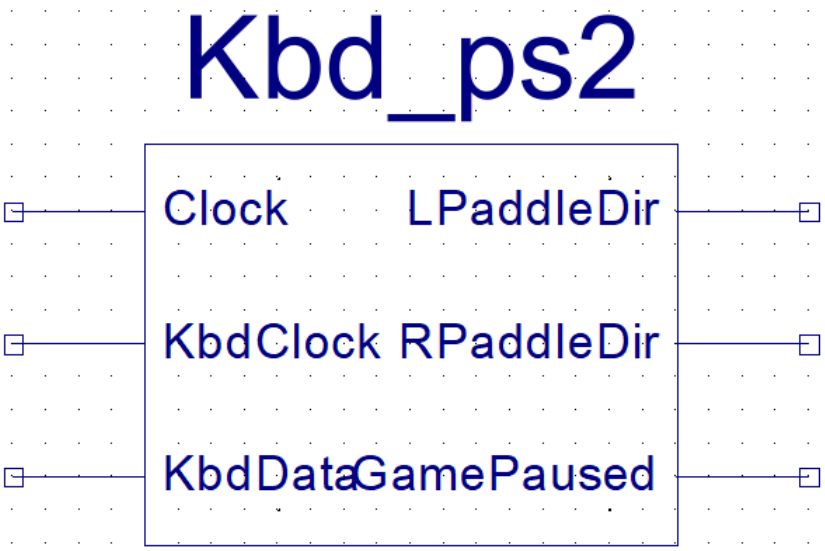
```
PORT (
   VGA_CLK : in std_logic;
   VGA_HS, VGA_VS : out std_logic;
   VGA_R : out std_logic;
   VGA_G : out std_logic;
   VGA_B : out std_logic;
   KB_CLK : in std_logic;
   KB_DATA : in std_logic
);
end PONG;
```



Rysunek 5: Schematy bloku gry, wygenerowany na podstawie kodu VHDL

Komponent klawiatury

```
component Kbd_ps2 is
    Port ( Clock : in STD_LOGIC;
           KbdClock : in STD_LOGIC;
           KbdData : in STD_LOGIC;
           LPaddleDir : out integer;
           RPaddleDir : out integer;
           GamePaused : out integer);
end component;
```



Rysunek 6: Schematy bloku klawiatury, wygenerowany na podstawie kodu VHDL

bity reprezentujące kolory RGB, współrzędne na ekranie, zmienne przechowujące przekalkulowane współrzędne oraz proces rekalkulacji współrzędnych by były bardziej intuicyjne dla autorów kodu przy działaniach arytmetycznych.

```
signal set_red, set_green, set_blue : std_logic;
signal pos_x : integer range 0 to 800:=0;
signal pos_y : integer range 0 to 525:=0;
signal real_x : integer range 0 to 640:=0;
signal real_y : integer range 0 to 480:=0;

calculateOffset : process(pos_x,pos_y)
```

```
  begin
     real_x <= pos_x -144;
     real_y <= pos_y -35;
     end process calculateOffset;
```

Segment kodu, odpowiedzialny za obliczanie kierunku i prędkości piłki w zależności od pozycji paletki. Liczba 8 jest szerokością (jak i również wysokością) piłki. `paddle_x1 i paddle_x2` to poziome położenie paletek, `paddle_width i paddle_height` to stałe określające wymiary paletek. `ball_x, ball_y` to współrzędne położenia piłki, natomiast `ballSpeed_x i ballSpeed_y` to składowe jej prędkości.

```
if ball_x + 8 > paddle_x2 and ball_x < paddle_x2 + paddle_width
    and ball_y + 8 > paddle_y2 and ball_y < paddle_y2 +
    paddle_height then
             if ballSpeed_x > 0 then
                ball_x <= paddle_x2 - 8;
             elsif ballSpeed_x < 0 then
                ball_x <= paddle_x2 + paddle_width;
             end if;
             ballSpeed_y <= (ball_y - (paddle_y2 + paddle_height /
                 2)) / 16;
             ballSpeed_x <= ballSpeed_y - ballSpeed_max / 2;
          elsif ball_x < paddle_x1 + paddle_width and ball_x + 8 >
              paddle_x1 and ball_y + 8 > paddle_y1 and ball_y <
              paddle_y1 + paddle_height then
             if ballSpeed_x > 0 then
                ball_x <= paddle_x1 - 8;
             elsif ballSpeed_x < 0 then
                ball_x <= paddle_x1 + paddle_width;
```

3 Wnioski

3.1 Podsumowanie

Celem projektu było zaimplementowanie gry Pong na układzie Spartan3E. Cel ten udało się zrealizować, w procesie testowania udało się również usunąć szereg potencjalnych błędów w zachowaniu gry.

3.2 Możliwości i kierunki rozwoju

Krokiem którego nie udało się zrealizować w czasie okazał się mechanizm zatrzymywania gry przy pomocy klawisza. W przypadku dalszej rozbudowy gry to mógłby być następny krok. Można również rozważyć mechanizm zmiennej prędkości i przyśpieszenia ruchu paletki paletek lub dopisanie większej liczby warunków, specyfikujących mechanikę ruchu piłki.

4 Bibliografia

- http://www.zsk.ict.pwr.wroc.pl/zsk/dyd/intinz/uc/
- http://www.digital-circuitry.com/VHDL_VGA_PONG.htm
- http://tinyvga.com/vga-timing
- https://eewiki.net/pages/viewpage.action?pageId=28278929