

Tytuł: Duck Hunt - multiplayer

Autorzy: Łukasz Gąsecki (ŁG),
Oliwia Szewczyk (OS)

Ostatnia modyfikacja: 09.06.2025

Spis treści

| | |
|--|---|
| 1. Repozytorium git..... | 1 |
| 2. Wstęp | 1 |
| 3. Specyfikacja | 1 |
| 3.1. Opis ogólny algorytmu..... | 1 |
| 3.2. Tabela zdarzeń | 2 |
| 4. Architektura..... | 2 |
| 4.1. Moduł: top | 2 |
| 4.1.1. Schemat blokowy | 2 |
| 4.1.2. Porty..... | 3 |
| a) mouse – mouse_ctl, input | 3 |
| b) vga – vga_ctl, output | 3 |
| 4.1.3. Interfejsy..... | 3 |
| a) vga_if –vga_signal..... | 3 |
| 4.2. Rozprowadzenie sygnału zegara | 3 |
| 5. Implementacja | 4 |
| 5.1. Lista zignorowanych ostrzeżeń Vivado..... | 4 |
| 5.2. Wykorzystanie zasobów | 4 |
| 5.3. Marginesy czasowe | 4 |
| 6. Film. | 4 |

1. Repozytorium git

Adres repozytorium GITa:

https://github.com/LGasecki/UEC2_MTM_Project_Duck_Hunt.git

W przypadku repozytorium prywatnego należy zaprosić użytkownika zewnętrznego o adresie mailowym:
kaczmarczyk@agh.edu.pl

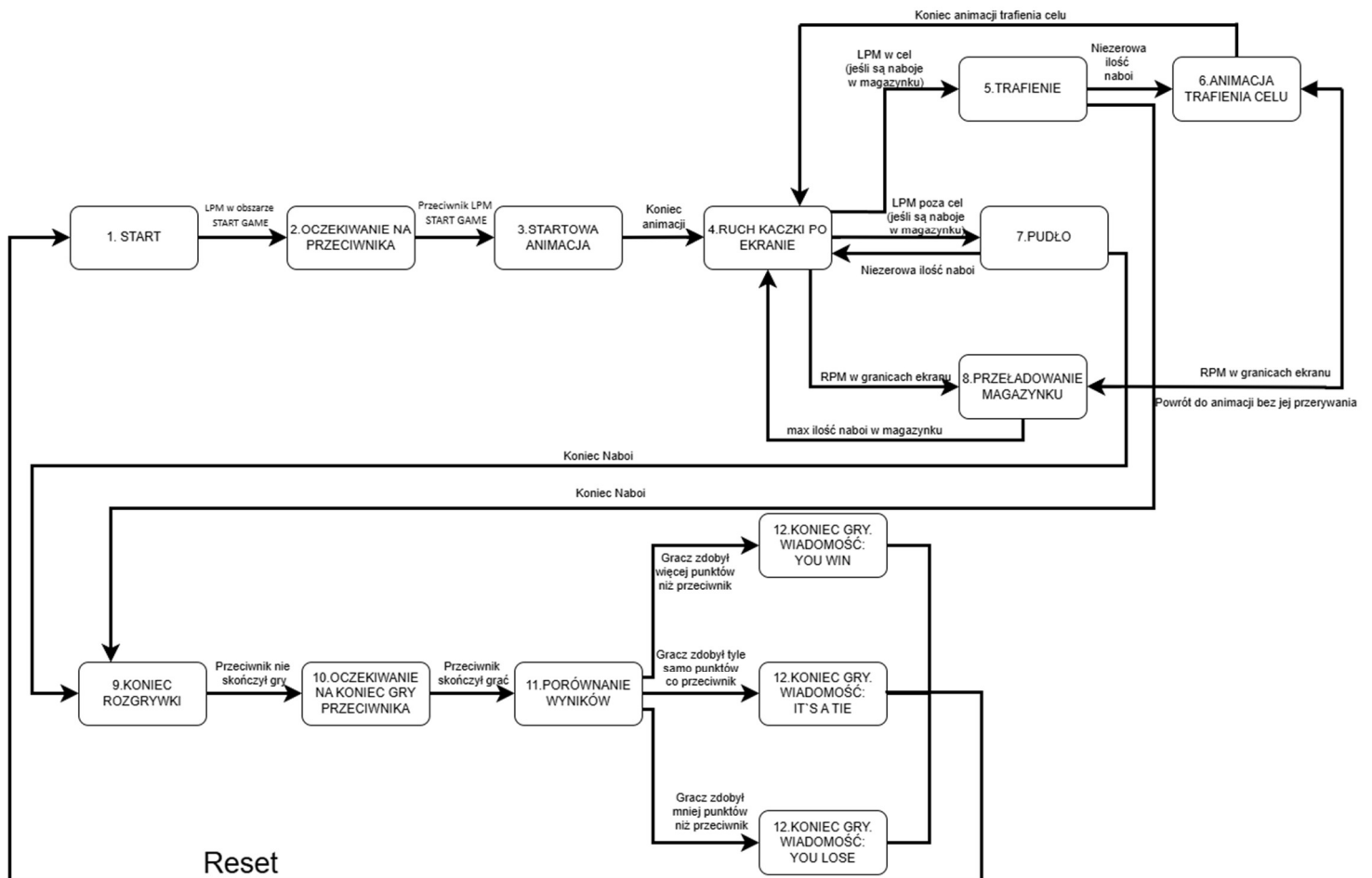
2. Wstęp

Pomysł na projekt wziął się z nostalgii do klasycznej gry z dzieciństwa – Duck Hunt. Postanowiliśmy stworzyć jej wieloosobową wersję od zera, tak aby przypominała oryginał. Gra została zaimplementowana w SystemVerilogu na platformie FPGA Basys3, gdzie wcielamy się w łowców, których celem jest zestrzelenie wszystkich kaczek. Projekt łączy zabawę z praktyczną nauką projektowania systemów cyfrowych.

3. Specyfikacja

3.1. Opis ogólny algorytmu

Uproszczony schemat blokowy działania implementowanego algorytmu.



1. Wyświetlany jest ekran startowy i przycisk *START GAME*
2. Graczowi który pierwszy kliknie *START GAME* pokaże się komunikat *WAITING FOR ENEMY*. Jak przeciwnik naciśnie *Start* gra się rozpoczyna.
3. Animacja zawiera psa który wskakuje za trawę rozpoczynając tym rozgrywkę
4. Losowanie pozycji celu, losowa pozycja *X* i kąt początkowy. Cel porusza się w linii prostej z różną prędkością i odbija się od ścian z losowym kątem,
5. Trafiając w cel(Lewy przycisk myszy) zdobywamy punkt i tracimy nabój, włączając animację.
6. W trakcie animacji można przeładować magazynek nie przerywając jej. Potem pojawia się nowy cel
7. Nie trafiając w cel(Lewy przycisk myszy), tracimy nabój.
8. Magazynek zawiera max 3 naboje, jak zużyjemy je wszystkie i spróbujemy wystrzelić ponownie pojawi się komunikat aby przeładować magazynek i nie pozwoli na strzał dopóki nie przeładujemy. Przeładować można z dowolną liczbą kul mniejszą niż 3.
9. Jeśli Gracz zużyje wszystkie naboje (ilość = 0), to kończy rozgrywkę i przechodzi do stanu oczekiwania na koniec rozgrywki przeciwnika
10. Jeśli przeciwnik dalej gra pokaże się komunikat *WAITING FOR ENEMY* dopóki przeciwnik nie skończy rozgrywki.
11. gdy obaj gracze skończą rozgrywkę następuje porównanie punktów i przejście do odpowiedniego ekranu końcowego
12. Jeśli gracz ma więcej punktów niż przeciwnik dostanie komunikat: *YOU WIN*.
Jeśli ma mniej punktów to dostanie komunikat: *YOU LOSE*.
W razie remisu pojawi się: *IT'S A TIE*.
Jeśli gracze chcą zagrać jeszcze raz to klikają *RESET*.

Płytki każdego z graczy komunikują się za pomocą protokołu *UART*. Jest to bardzo prosty protokół komunikacji oparty na zamianie danych równoległych na szeregowy. W czasie rzeczywistym przesyłane są informacje o chęci rozpoczęcia oraz o zakończeniu rozgrywki wraz w 6-bitową wartością wyniku każdego z graczy. Dzięki temu nie musieliśmy używać, więcej niż jedno złącze *UART*.

3.2. Tabela zdarzeń

Opis zdarzeń występujących podczas działania programu/urządzenia, zarówno zewnętrznych (interakcje z użytkownikiem), jak i wewnętrznych (specyficzne stany w algorytmie). Zdarzenia podzielone są na kategorie dotyczące różnych stanów działania programu. Kategorie powinny odpowiadać stanom ze schematu z pkt. 2.1.

| Zdarzenie | Kategoria | Reakcja systemu |
|---|------------------------------------|---|
| LPM w obszarze napisu START GAME | Ekran startowy | Przejsie do stanu czekania na przeciwnika, informacja przeciwnika o gotowości gry |
| LPM w obszarze START GAME przez przeciwnika | Czekanie na start przeciwnika | Rozpoczęcie gry |
| Animacja startowa | Gra | Animacja wchodzącego psa i wskazującego za trawę |
| Losowanie startowej pozycji X celu | Gra | Ustawienie celu na wysokości startowej oraz losowej pozycji wzdłuż linii tej wysokości |
| Cel uderza w ścianę | Gra | Zmiana kierunku lotu celu z losową prędkością wektorową |
| Trafienie celu (LPM) | Gra | Dodanie punktu do naszego wyniku, animacja zabicia celu, odjęcie naboju, animacja psa trzymającego zabity cel |
| Strzelenie poza celem / Pudło (LPM) | Gra | Odjęcie naboju, gra toczy się dalej |
| Przeładowanie magazynku (RPM) | Gra | Załadowanie maksymalną liczbę naboji (3) do magazynku, aktualizacja dostępnych wszystkich naboji |
| Liczba wszystkich naboji = 0 | Gra | Koniec gry, przejście do stanu czekania na koniec gry przeciwnika |
| Przeciwnik skończy grę (liczba jego naboji = 0) | Czekanie na koniec gry przeciwnika | Porównanie wyników, przejście do ekranu końca gry. |
| Wyświetlanie wyniku | Ekran końcowy | Wyświetlanie wyniku graczy wraz z informacją kto wygrał |
| LPM w obszar napisu RESET GAME | Ekran końcowy | Powrót na ekran startowy |

4. Architektura

4.1. Moduły top:

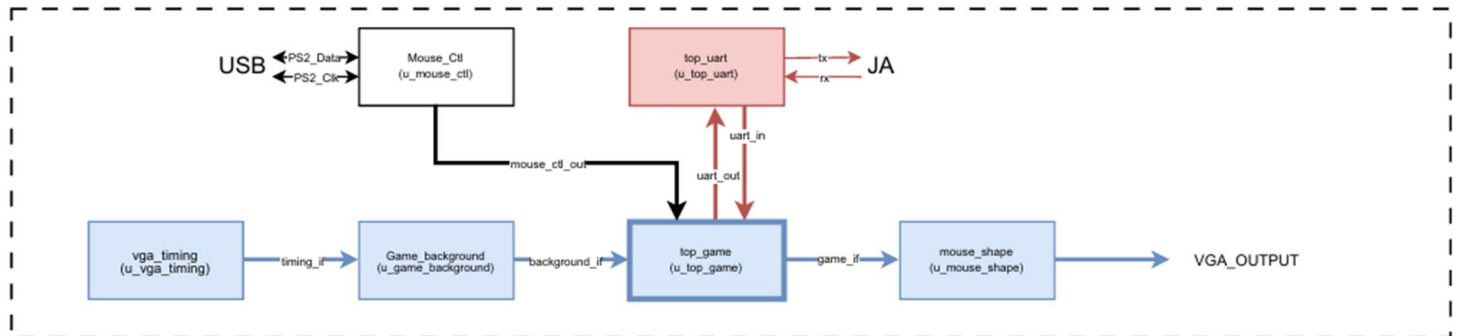
top_vga: Osoba odpowiedzialna: ŁG

top_game: Osoby odpowiedzialne: ŁG + OS

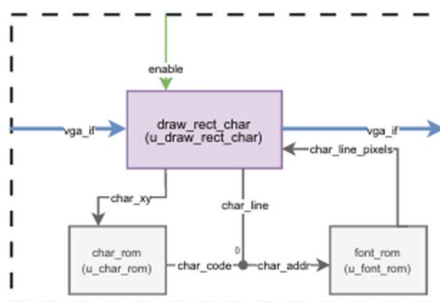
top_uart: Osoba odpowiedzialna: ŁG

4.1.1. Schemat blokowy

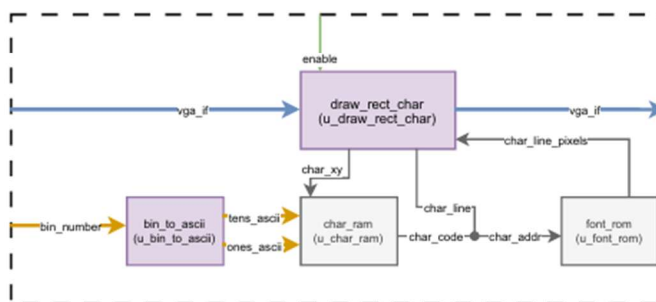
top_vga:



draw_string:



draw_2_numbers:



- moduły dotyczące sygnału vga



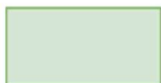
- moduł wyjściowy UART



- moduły fsm kontrolujące poruszanie się obiektów



- moduł wyjściowy USB / mouse PS2



- główny moduł fsm, ustala etapy gry

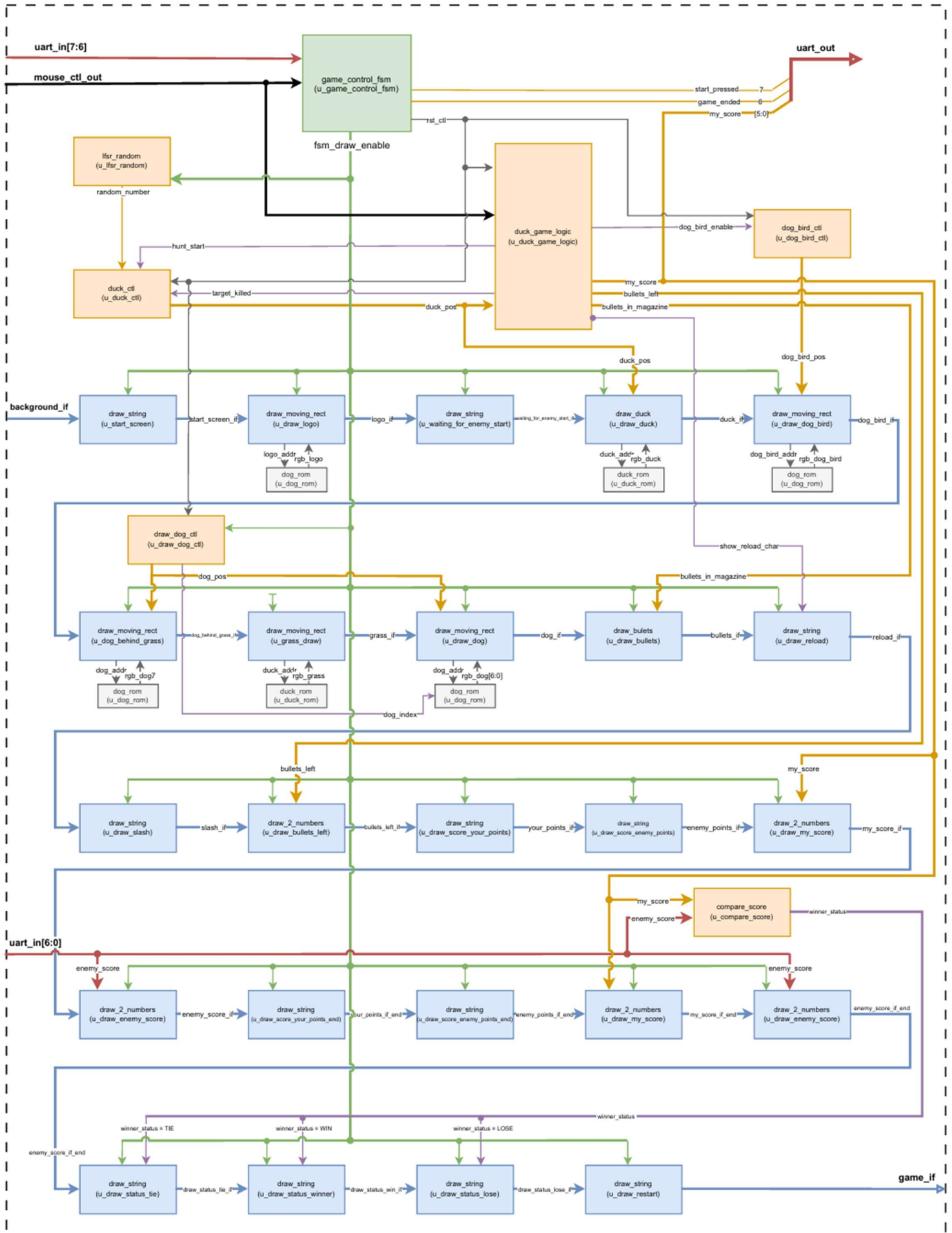


- moduły inne/różne



- moduły pamięci ram/rom

TOP_GAME



4.1.2. Porty

a) *mouse – mouse_ctl_out, output*

| nazwa portu | opis |
|-------------|---|
| ps2_data | szeregowe wejście/wyjście danych muszki |
| ps2_clk | zegar myszki |

b) *vga – vga_ctl, output*

| nazwa portu | opis |
|---------------|--|
| Vsync | sygnał synchronizacji pionowej VGA |
| Hsync | Sygnał synchronizacji poziomej VGA |
| vgaGreen[3:0] | Sygnał natężenia koloru zielonego VGA |
| vgaBlue[3:0] | Sygnał natężenia koloru niebieskiego VGA |
| vgaRed[3:0] | Sygnał natężenia koloru czerwonego VGA |

c) *uart – top_uart, inout*

| nazwa portu | opis |
|-------------|-------------------------------------|
| rx | Szeregowe wejście danych uart |
| tx | Szeregowe wyjście danych uart |
| gnd | Wspólna masa dla połączonych płytek |

4.1.3. Interfejsy

a) *Vga_if:*

timing_if / background_if / draw_mouse_if / mouse_shape_if / game_if / start_screen_if / logo_if / waiting_for_enemy_start_if / duck_if / dog_bird_if / dog_behind_grass_if / grass_if / dog_if / bullets_if / reload_if / slash_if / bullets_left_if / your_points_if / enemy_points_if / my_score_if / enemy_score_if / your_points_if_end / enemy_points_if_end / my_score_if_end / enemy_score_if_end / draw_status_tie_if / draw_status_win_if / draw_status_lose_if

output

| nazwa sygnału | opis |
|---------------|------------------------------------|
| hcount [10:0] | horyzontalny licznik VGA |
| vcount [10:0] | wertykalny licznik VGA |
| hsync | sygnał synchronizacji poziomej VGA |
| vsync | sygnał synchronizacji pionowej VGA |
| hblink | sygnał horyzontalny blank VGA |
| vblink | sygnał wertykalny blank VGA |
| rgb [11:0] | sygnał koloru rgb VGA |

b) fsm_draw_enable

| nazwa sygnału | opis |
|----------------------|--|
| start_screen_enable | bit enable sygnalizujący ekran startowy gry |
| start_pressed | bit sygnalizujący gotowość do gry |
| game_enable_posedge | bit sygnalizujący moment rozpoczęcia gry (dla animacji) |
| game_enable | bit sygnalizujący rozpoczęcie rozgrywkę gry |
| game_finished | bit sygnalizujący o skończeniu rozgrywki przez gracza (ekran czekania) |
| end_screen_enable | bit sygnalizujące ekran końcowy z wynikami |

c) duck_pos

| nazwa sygnału | opis |
|----------------------|-----------------------------|
| duck_xpos[11:0] | pozycja x celu (kaczki) |
| duck_ypos[11:0] | pozycja y celu (kaczki) |
| duck_direction | kierunek lotu celu (kaczki) |

d) dog_pos

| nazwa sygnału | opis |
|----------------------|------------------------|
| dog_xpos[11:0] | pozycja x animacji psa |
| dog_ypos[11:0] | pozycja y animacji psa |

e) dog_bird_pos

| nazwa sygnału | opis |
|----------------------|---------------------------------|
| dog_bird_xpos[11:0] | pozycja x psa ze złapanym celem |
| dog_bird_ypos[11:0] | pozycja y psa ze złapanym celem |

f) winner_status

| nazwa sygnału | opis |
|----------------------|--------------------------|
| draw | sygnał remisu graczy |
| win | sygnał przegranej gracza |
| lose | sygnał wygranej gracza |

g) my_score, enemy_score

| nazwa sygnału | opis |
|----------------------|----------------------------|
| my_score [5:0] | liczba punktów gracza |
| enemy_score [5:0] | liczba punktów przeciwnika |

h) bullets_left

| nazwa sygnału | opis |
|-------------------|---|
| bullets_left[6:0] | liczba naboí pozostałych możliwych do przeładowania |

i) bullets_in_magazine

| nazwa sygnału | opis |
|------------------------------|--|
| bullets_in_magazine [2:0] | liczba naboí możliwych do wystrzelenia (max 3) |

4.2. Rozprowadzenie sygnału zegara

Wszystkie moduły korzystają z jednego zegara o częstotliwości 65 MHz.

5. Implementacja**5.1. Lista zignorowanych ostrzeżeń Vivado.**

| Identyfikator ostrzeżenia | Liczba wystąpień | Uzasadnienie |
|--|------------------|--|
| Synth 8-7080 Parallel synthesis criteria is not met | 1 | Pojawia się zawsze, nie ma wpływu na bitstream |

5.2. Wykorzystanie zasobów

Tabela z wykorzystaniem zasobów z Vivado

| Resource | Utilization | Available | Utilization % |
|----------|-------------|-----------|---------------|
| LUT | 5703 | 20800 | 27.42 |
| LUTRAM | 108 | 9600 | 1.13 |
| FF | 3766 | 41600 | 9.05 |
| BRAM | 41 | 50 | 82.00 |
| DSP | 2 | 90 | 2.22 |
| IO | 21 | 106 | 19.81 |
| BUFG | 2 | 32 | 6.25 |
| MMCM | 1 | 5 | 20.00 |

5.3. Marginesy czasowe

Marginesy czasowe (WNS) dla setup i hold.

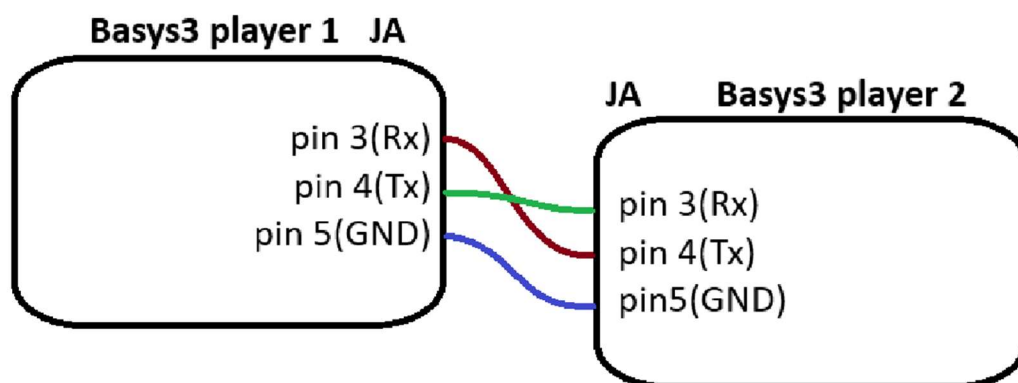
Design Timing Summary

| Setup | Hold | Pulse Width |
|--------------------------------------|----------------------------------|---|
| Worst Negative Slack (WNS): 1,865 ns | Worst Hold Slack (WHS): 0,035 ns | Worst Pulse Width Slack (WPWS): 3,000 ns |
| Total Negative Slack (TNS): 0,000 ns | Total Hold Slack (THS): 0,000 ns | Total Pulse Width Negative Slack (TPWS): 0,000 ns |
| Number of Failing Endpoints: 0 | Number of Failing Endpoints: 0 | Number of Failing Endpoints: 0 |
| Total Number of Endpoints: 5707 | Total Number of Endpoints: 5707 | Total Number of Endpoints: 3962 |

All user specified timing constraints are met.

6. Konfiguracja sprzętu

Schemat połączenia ze sobą płytek Basys3 w trybie multiplayer.



7. Film.