Tytuł: **Gra w Statki**

Autorzy:

Tomasz Ochmanek, Jan Panek

Ostatnia modyfikacja: 02.09.2024

Spis treści

1. Repozytorium git

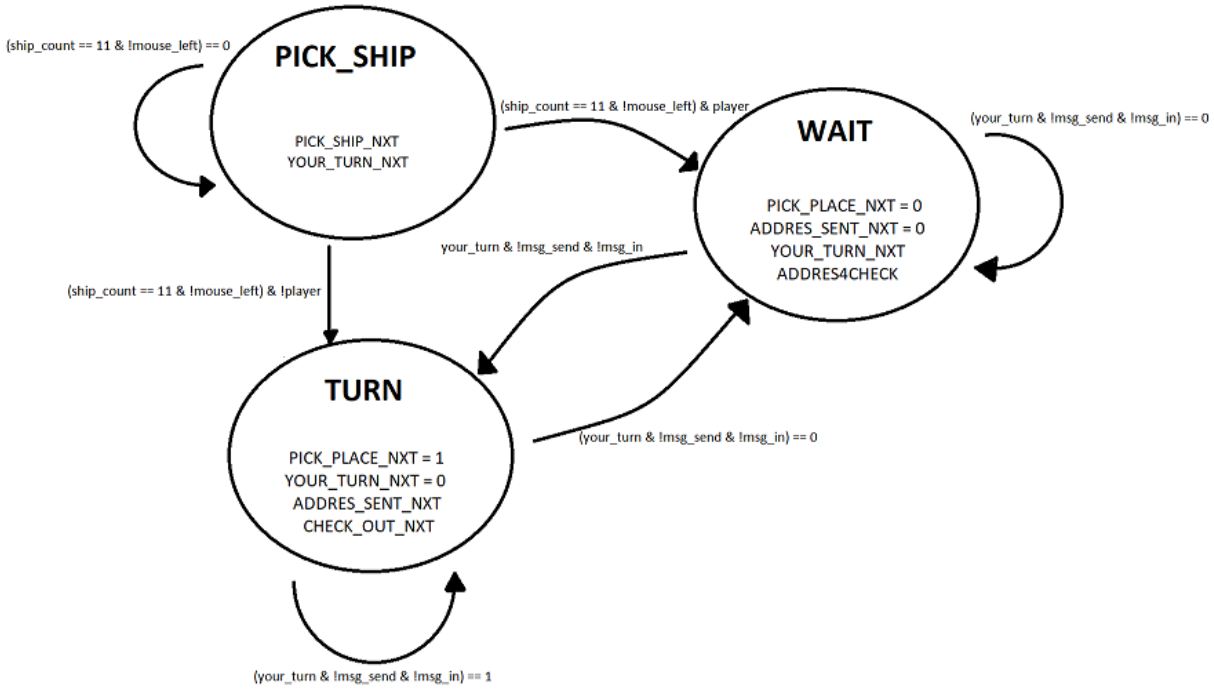
https://github.com/Tomsz1337/UEC2_PROJEKT

2. Wstęp

W ramach projektu została stworzona gra w statki dla dwóch osób, każda z osób do gry używa osobnej płytki BASYS3. Gra w statki to popularna gra planszowa. W grze zostały zaimplementowane wszystkie podstawowe zasady gry w statki takie jak ustawienie statków na początku rozgrywki oraz trakcie swojej tury wybieramy pole na planszy przeciwnika, w które chcemy trafić, następnie czekamy na odpowiedz czy na danym polu znajdował się statek i pokazuje się nam trafienie albo pudło.

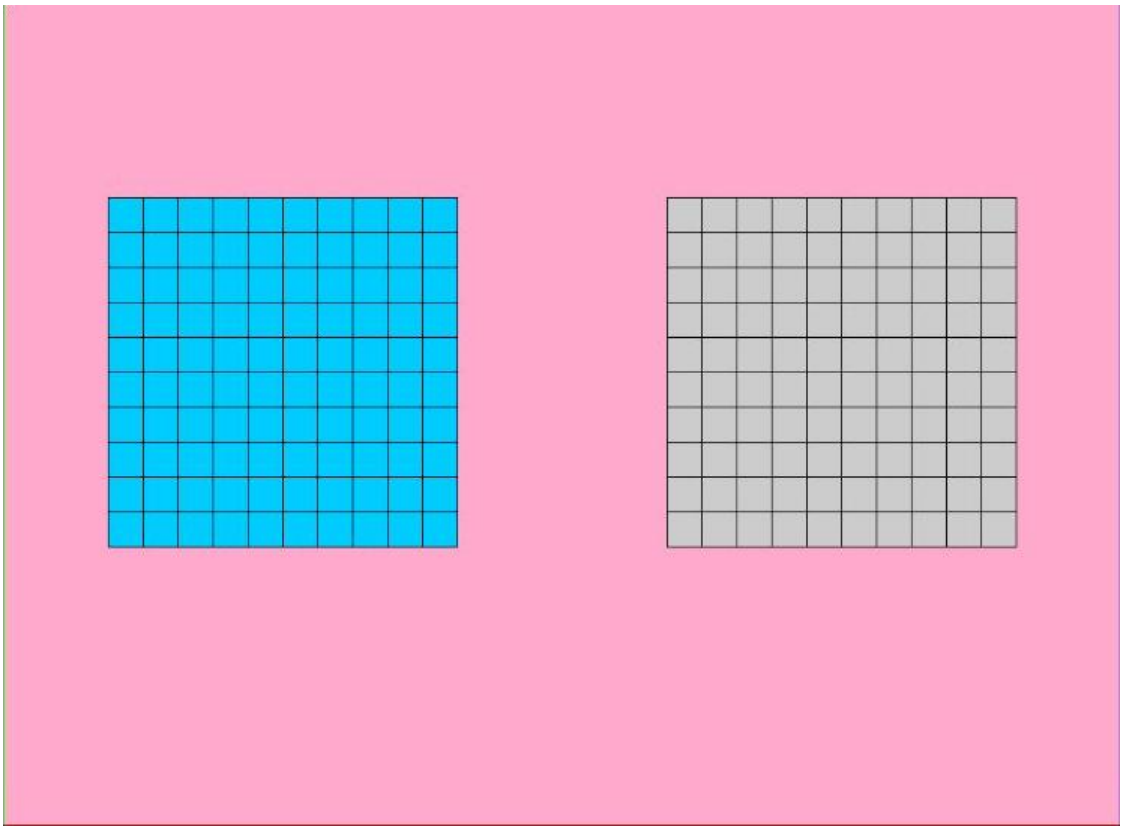
3. Specyfikacja

3.1. Opis ogólny algorytmu



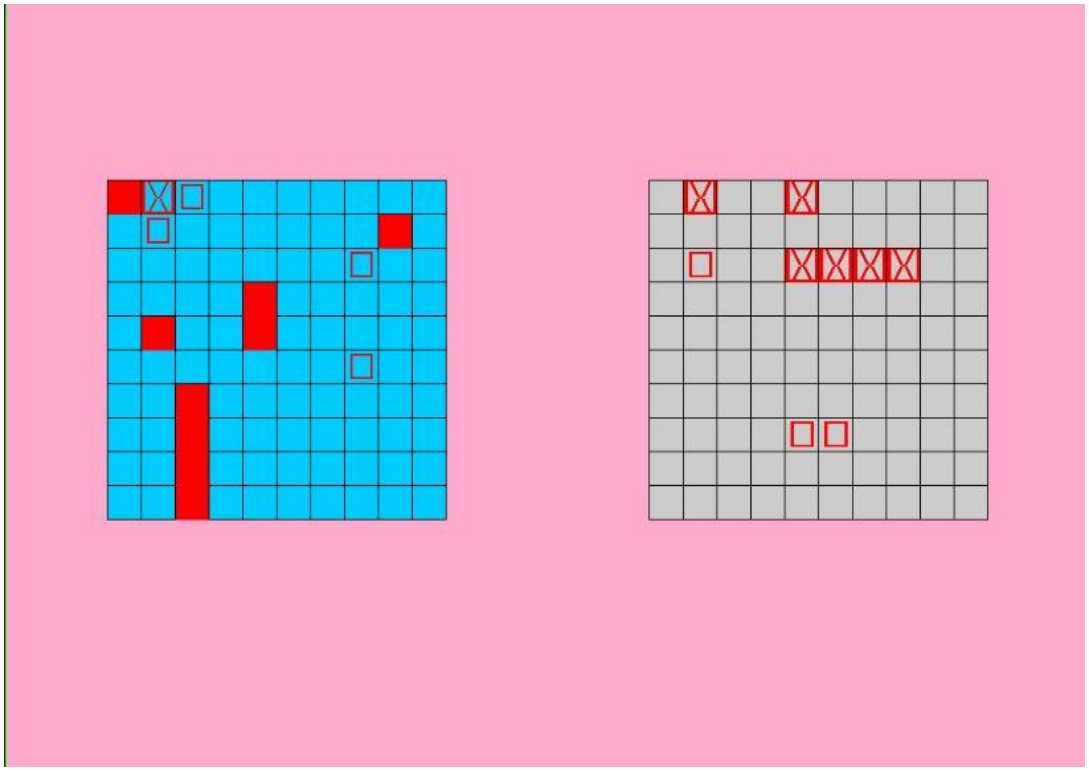
Schemat 1. Uproszczony schemat blokowy działania implementowanego algorytmu

- Po uruchomieniu płytek z wgranym bitstreamem gry na ekranie pojawia się plansza należy przełączyć przełącznik sw[0] na jednej z płytek aby zdecydować kto zaczyna. Później można wybrać gdzie ustawić swoje statki na niebieskiej planszy. Gra rozpocznie się gdy każdy gracz wybierze 11 pól dla swoich statków.



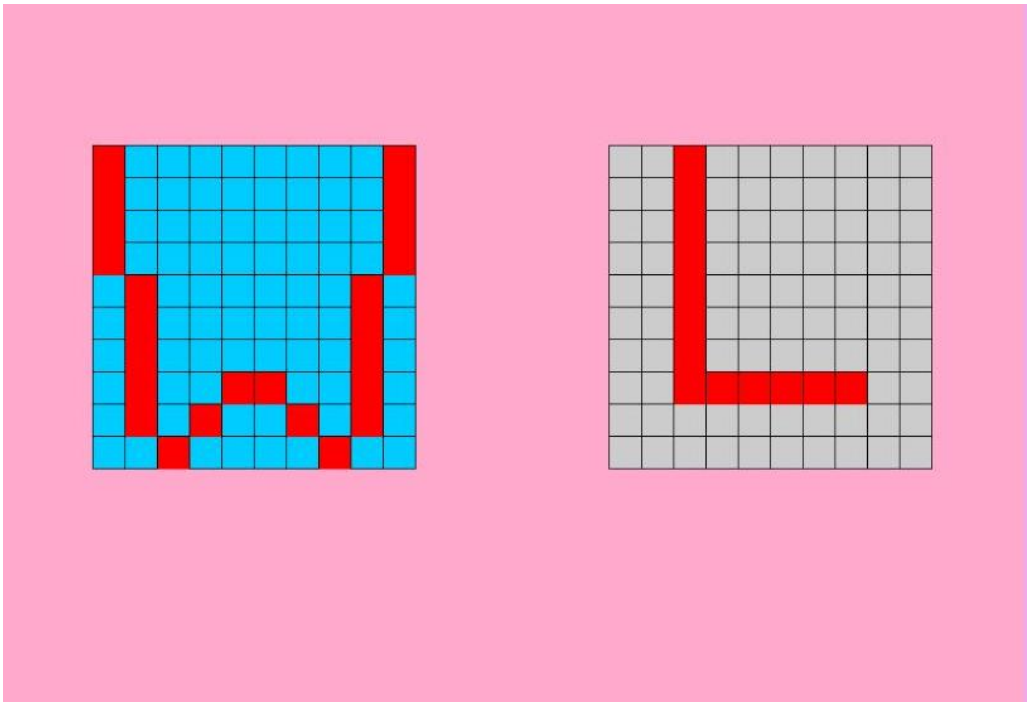
Obraz 1. Wygląd planszy podczas startu.

- Po wybraniu statków jeden z graczy rozpoczyna rozgrywkę, a drugi oczekuje na jego ruch, gracze wykonują ruchy na zmianę, wysyłając miedzy sobą informacje o wykonanym ruchu.
- Po wybraniu przez gracza pola na szarej planszy, które chce trafić, do drugiego gracza wysyłana jest informacja z wybranymi współrzędnymi. Moduł game_board sprawdza czy ruch trafił czy spudłował na planszy przeciwnika, która jest zapisana w board_host przeciwnika następnie zostaje wtedy wpisany odpowiedni kod znaku do board_guest i wyświetla na tej planszy „X” (trafiony) albo „O” (pudło).



Obraz 2. Wygląd planszy w trakcie rozgrywki.

- Gra kończy się w momencie kiedy wszystkie statki zostaną zestrzelone wtedy na ekranie wyświetla się menu końca gry, na którym wyświetla się kto wygrał i kto przegrał. Statki ustawiają się w kształt litery W dla wygranego i L dla przegranego.



Obraz 3. Wygląd menu końcowego gdy się wygrywa.

3.2. Tabela zdarzeń

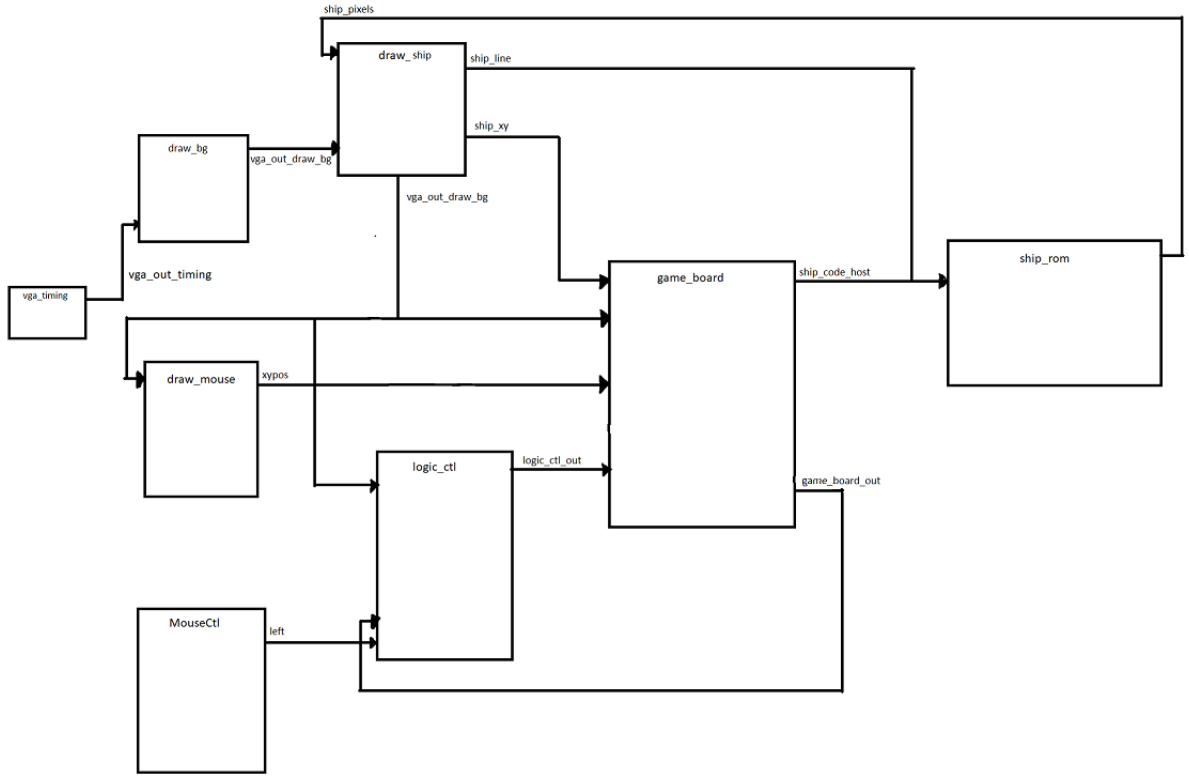
Opis zdarzeń występujących podczas działania programu/urządzenia, zarówno zewnętrznych (interakcje z użytkownikiem), jak i wewnętrznych (specyficzne stany w algorytmie). Zdarzenia podzielone są na kategorie dotyczący różnych stanów działania programu. Kategorie powinny odpowiadać stanom ze schematu z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

Zdarzenie	**Kategoria**	**Reakcja systemu**
LPM w obszarze planszy niebieskiej	pick_ship	Ułożenie statków
Ship_count == 11 & player == 0	Turn	Rozpoczęcie rozgrywki
Ship_count == 11 & player == 0	Wait	Rozpoczęcie rozgrywki
LPM w obszarze szarej planszy	Turn	Oddanie strzału
hit_counter == 0	Turn/Wait	Koniec gry
Your_turn = 0	Wait	Oczekiwanie na ruch przeciwnika
Your_turn = 1	Turn	Nasz ruch
msg_send = 2’b00	Turn/Wait	Brak transmisji
msg_send = 2’b01	Turn/Wait	Transmisja danych i przejście tury do przeciwnika

4. Architektura

4.1. Moduł: top

4.1.1. Schemat blokowy



Porty

a) mou – mouse_ctl, input

nazwa portu	opis
mou_si	Szeregowe wejście danych

b) vga – vga_ctl, output

nazwa portu	opis
vga_vs	sygnał synchronizacji pionowej VGA
vga_hs	sygnał synchronizacji poziomej VGA
vga_red	Kolor czerowny nasycenie
vga_blue	Kolor niebieski nasycenie
vga_green	Kolor zielony nasycenie

4.1.2. Interfejsy

a) m2c – mouse_ctl to core

nazwa sygnału	opis
m2c_x[9:0]	Horyzontalna pozycja kursora myszy na ekranie
m2c_y[9:0]	Wertykalna pozycja kursora myszy na ekranie

b) ship_xy

nazwa sygnału	opis
ship_xy_host	Pozycja statku na planszy niebieskiej
ship_xy_guest	Pozycja statku na planszy szarej

c) ship_code_host

nazwa sygnału	opis
ship_code_host	Adres rysownego znaku
ship_code_guest	Adres rysownego znaku

d) vga_out

nazwa sygnału	opis
vsync	Synchronizacja pionowa
vcount	Licznik pionowy
vblnk	Obszar bez rysowania niczego
hcount	Licznik poziomy
hsync	Synchronizacja pozioma
hblnk	Obszar bez rysowania niczego
rgb	Kolory

e) vga_in

nazwa sygnału	opis
vsync	Synchronizacja pionowa
vcount	Licznik pionowy
vblnk	Obszar bez rysowania niczego
hcount	Licznik poziomy
hsync	Synchronizacja pozioma
hblnk	Obszar bez rysowania niczego
rgb	Kolory

f) xypos

nazwa sygnału	opis
ypos	Pozycja y myszki
xpos	Pozycja x myszki

g) logic_ctl_out

nazwa sygnału	opis
mouse_position	Wysyła dane zawierające aktulane położenie myszki
addres4check	Adres do sprawdzenia czy jest na danym polu statek
Pick_place	Wybór pola na planszy
Pick_ship	Wybór pola w którym ma zostać postawiony statek

h) game_board_out

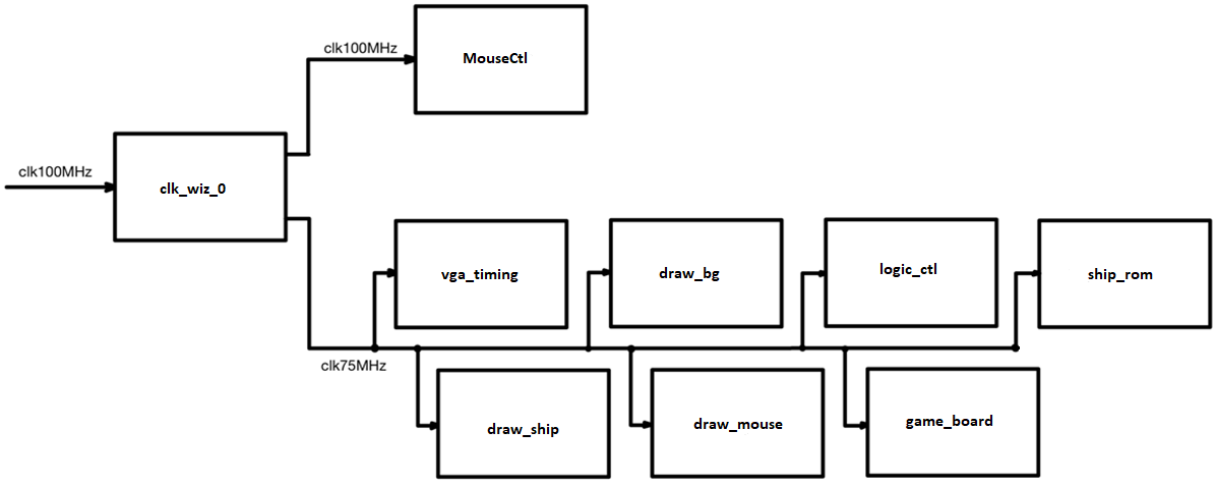
nazwa sygnału	opis
m2c_x[9:0]	Horyzontalna pozycja kursora myszy na ekranie
m2c_y[9:0]	Wertykalna pozycja kursora myszy na ekranie

b) ship_code_host

nazwa sygnału	opis
msg_out	Komunikuje się między płytkami

4.2. Rozprowadzenie sygnału zegara

W całym układzie używamy dwóch zegarów clk_75 – 75MHz oraz clk_100 – 100MHz. Zegara clk100 używa myszka pozostałe moduły używają zegara clk75.

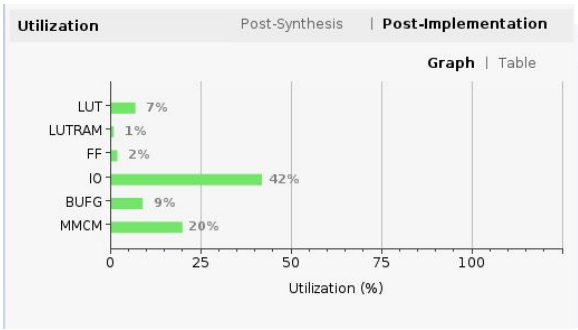


5. Implementacja

5.1. Lista zignorowanych ostrzeżeń Vivado.

Identyfikator ostrzeżenia	Liczba wystąpień	Uzasadnienie
[Synth 8-87]	2	Używamy bloku always comb do przypisania stałych lub domyślnych wartości.
[Synth 8-327]	2	Logika wymaga zatrzymania stanu pomiędzy cyklami zegara, a zatem wymaga przerzutników.
[Synth 8-7080]	1	Brak tej optymalizacji nie jest kluczowe dla obecnej wersji projektu.
[Synth 8-5856]	2	Użycie rejestrów zamiast pamięci 3D nie wpływa negatywnie na projekt pod względem wydajności, powierzchni logicznej, czy innych zasobów sprzętowych.
[Synth 8-6014]	4	Korzystamy tylko części sygnałów, które nam są potrzebne w module pozostawiając część sygnałów w interfejsie nie używanych
[Synth 8-3332]	26	Bez tego nasze statki i oznaczenia trafienia nie rysowałyby się w całości
[Synth 8-7129]	6	To ostrzeżenie musi być, żeby program zadziałał

5.2. Wykorzystanie zasobów



Utilization Post-Synthesis | **Post-Implementation**

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	1481	20800	7.12
LUTRAM	2	9600	0.02
FF	993	41600	2.39
IO	44	106	41.51
BUFG	3	32	9.38
MMCM	1	5	20.00

5.3. Marginesy czasowe

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0.350 ns	Worst Hold Slack (WHS):	0.039 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:	1289	Total Number of Endpoints:	1289	Total Number of Endpoints:	1005

All user specified timing constraints are met.

6. Film.

Link do ściągnięcia filmu:

https://drive.google.com/file/d/1oItu8FULwrDpVpCUsphZVd5o5aH3CyPh/view?usp=sharing