Tytuł: **Donkey Vs. Kong**

Autorzy: **Dawid Bodzek (DB), Jakub Bukowski (JB)**

Ostatnia modyfikacja: 31.08.2025

Spis treści

1. Repozytorium git

Adres repozytorium GITa:

https://github.com/Foggallon/DonkeyVsKong

2. Wstęp

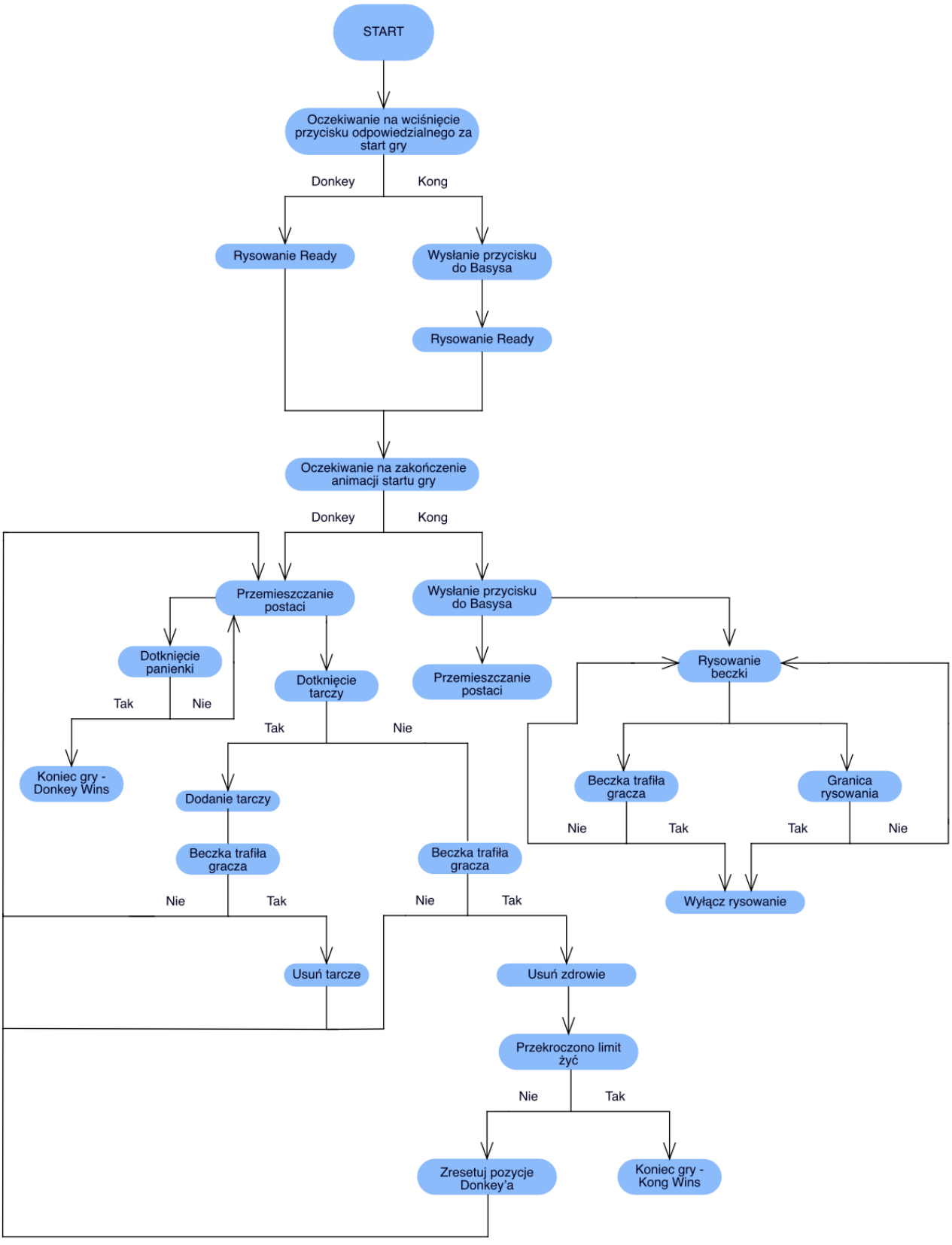
W ramach projektu przygotowaliśmy grę, a mianowicie twist na klasycznym tytule Donkey Kong. Zamysł gry fundamentalnie opiera się na zamyśle retro oryginału, który dał nam inspirację, nasz twist polega na możliwości cieszenia się z tego klasyka we dwie osoby. W taki oto sposób jeden gracz wciela się w rolę Kong'a, który ma na celu uniemożliwienie Donkey'emu dotarcie do księżniczki.

3. Specyfikacja

3.1. Opis ogólny algorytmu

Uproszczony schemat blokowy działania implementowanego algorytmu.

Rozpoczęcie gry ma miejsce po zgłoszeniu przez obu graczy gotowości, po otrzymaniu informacji na temat gotowości rysowany jest o tym komunikat pod odpowiednią nazwą postaci. Gdy obaj gracze zgłoszą gotowość, zaczyna się animacja początkowa i to po jej zakończeniu zaczyna się właściwa gra. Obaj gracze mają możliwość przemieszczania się po planszy, obszar jednego z nich jest ograniczony lecz wynika to jedynie ze względów balansu trudności rozgrywki. Podczas gdy gracz "Donkey" przemieszcza się wykonywane jest sprawdzenie czy znajduje się on w obszarze tarczy, jeśli do tego dojdzie tarcza zostaje dodana do paska życia oraz następnie przestaje być rysowana na planszy gry. Podczas rozgrywki "Donkey" może zostać trafiony beczką przez "Konga" co spowoduje utratę jednego z życia z paska zdrowia, co również doprowadzi do resetu jego pozycji do położenia początkowego, jeśli straci on wszystkie ze swoich życi rozgrywka kończy się i zwracany jest komunikat w postaci wizualnej informujący o zwycięstwie "Konga". W przypadku gdy "Donkey" zostanie trafiony beczką przez "Konga" a posiada on tarczę nie zostanie on zresetowany na pozycję startową lecz straci on za to tarczę ze swojego paska zdrowia. Drugi gracz ("Kong") natomiast aby jego rozgrywka była choć minimalnie ciekawa i wciągająca ma możliwość rzucania wymienionymi wcześniej beczkami na dwa sposoby. Jeden z nich to rzuty horyzontalne, które powodują spadanie beczek przez całą mapę po linii platform. Następuję to po naciśnięciu przez "Konga" odpowiedniego przycisku, co powoduje, że rysowana jest beczka. Beczek na planszy może znajdować się maksymalnie pięć w tym samym czasie, tego samego rodzaju, po osiągnięciu określonej w kodzie granicy ich spadania przestają być rysowane. Beczki również przestają być rysowane w momencie kontaktu z "Donkey'm". Oczywiście "Kong" nie jest jedynym, który może wygrać, a warunkiem na zwycięstwo "Donkey'ego" jest uratowanie księżniczki. Następuję to przez osiągnięcie szczytu mapy i znalezienie się postaci w obszarze określonym jako obszar dotknięcia księżniczki. Spełnienie tego warunku skutkuje w otrzymaniu wizualnej informacji zwrotnej na temat zwycięstwa "Donkey'ego".



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.

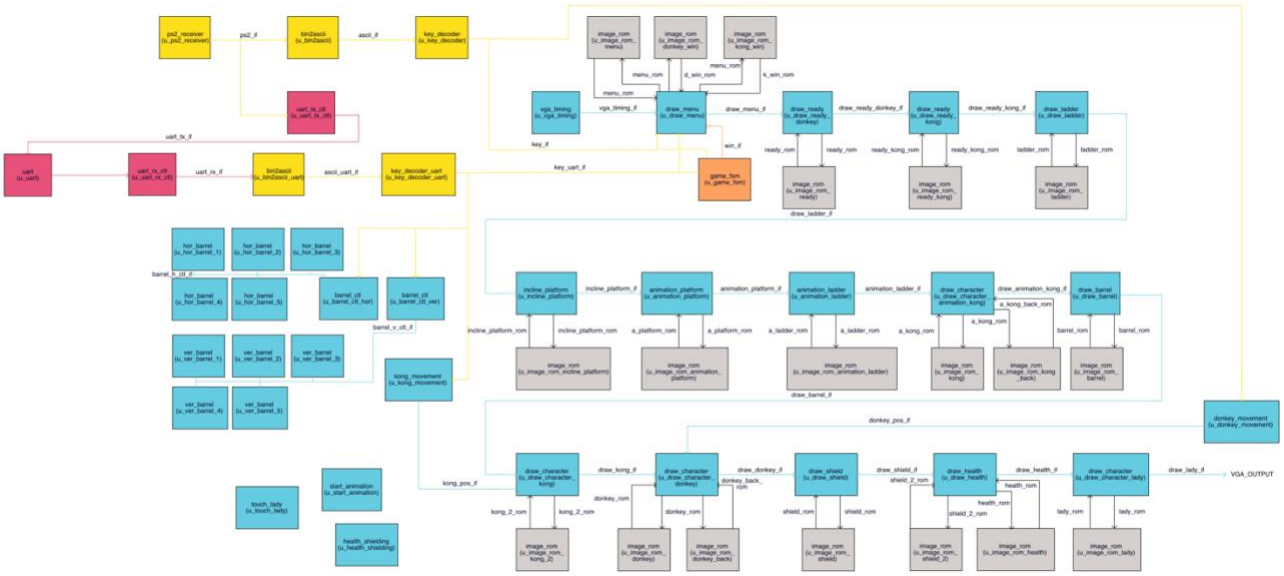
Zdarzenie	Kategoria	Reakcja systemu
Reset / Uruchomienie sprzętu	Start	System inicjalizuje rejestry i pamięć wartościami domyślnymi, a następnie wyświetla ekran startowy z menu.
Naciśniecie klawisza Enter przez gracza	Menu, ekran startowy	Na ekranie pojawia się komunikat *„Ready”*. System oczekuje na reakcję drugiego gracza. W przypadku gracza **Kong** sygnał klawisza przekazywany jest poprzez interfejs **UART**.
Gracz Donkey wciska klawisz (W, S, A, D, spacja)	Przemieszczanie postaci	Postać **Donkey** porusza się po planszy: wspina się lub schodzi po drabinie, przemieszcza w lewo lub prawo, a po naciśnięciu spacji wykonuje skok.
Gracz Kong wciska klawisz (A, D, W, S)	Przemieszczanie postaci	Naciśnięty klawisz przesyłany jest przez **UART**, a postać **Kong** reaguje odpowiednio: porusza się w lewo lub prawo, bądź rzuca beczkę – po platformie albo w dół ekranu.
is_shielded == 1	Dotknięcie tarczy	Gracz otrzymuje dodatkowe życie w postaci tarczy (parasolka), a jego postać staje się odporna na trafienie beczką.
hit == 1	Beczka trafia gracza	Postać **Donkey** traci jedno życie, a jego pozycja na planszy zostaje zresetowana.
done == 1	Granica rysowania	Rysowanie beczki zostaje zakończone, a zwolnione miejsce w sygnale *barrel* umożliwia wygenerowanie nowej beczki.
touch_lady == 1	Dotknięcie panienki	Gra kończy się, a na ekranie wyświetlana jest informacja o zwycięstwie gracza **Donkey**.
health_en == 0	Przekroczono limit żyć	Gra kończy się, a na ekranie pojawia się informacja o zwycięstwie gracza **Kong**.

4. Architektura

4.1. Moduł: top

Osoba odpowiedzialna: DB, JB

4.1.1. Schemat blokowy



4.1.2. Porty

a) keyboard – keyboard_ctl, input

nazwa portu	opis
ps2_data	szeregowe wejście danych klawiatury
ps2_clk	zegar klawiatury

b) UART – uart_ctl, input, output

nazwa portu	opis
rx	sygnał odbioru UART
tx	sygnał transmisji UART

c) vga – vga_ctl, output

nazwa portu	opis
vga_vs	sygnał synchronizacji pionowej VGA
vga_hs	sygnał synchronizacji poziomej VGA
vga_r[3:0]	sygnał koloru czerwonego VGA
vga_g[3:0]	sygnał koloru zielonego VGA
vga_b[3:0]	sygnał koloru niebieskiego VGA

4.1.3. Interfejsy

d) vga_timing_if – vga_timing to draw_menu

nazwa sygnału	opis
hcount[10:0]	Licznik poziomy VGA
vcount[10:0]	Licznik pionowy VGA
hsync	Sygnał synchronizacji poziomej VGA
vsync	Sygnał synchronizacji pionowej VGA
hblnk	Sygnał blank poziomy VGA
vblnk	Sygnał blank pionowy VGA
rgb[11:0]	Sygnał rgb VGA

e) draw_menu_if – draw_menu to draw_ready_donkey

nazwa sygnału	opis
hcount[10:0]	Licznik poziomy VGA
vcount[10:0]	Licznik pionowy VGA
hsync	Sygnał synchronizacji poziomej VGA
vsync	Sygnał synchronizacji pionowej VGA
hblnk	Sygnał blank poziomy VGA
vblnk	Sygnał blank pionowy VGA
rgb[11:0]	Sygnał rgb VGA

f) draw_ready donkey_if – draw_ready_donkey to draw_ready_kong

nazwa sygnału	opis
hcount[10:0]	Licznik poziomy VGA
vcount[10:0]	Licznik pionowy VGA
hsync	Sygnał synchronizacji poziomej VGA
vsync	Sygnał synchronizacji pionowej VGA
hblnk	Sygnał blank poziomy VGA
vblnk	Sygnał blank pionowy VGA
rgb[11:0]	Sygnał rgb VGA

g) draw_ready_kong_if – draw_ready_kong to draw_ladder

nazwa sygnału	opis
hcount[10:0]	Licznik poziomy VGA
vcount[10:0]	Licznik pionowy VGA
hsync	Sygnał synchronizacji poziomej VGA
vsync	Sygnał synchronizacji pionowej VGA
hblnk	Sygnał blank poziomy VGA
vblnk	Sygnał blank pionowy VGA
rgb[11:0]	Sygnał rgb VGA

h) draw_ladder_if – draw_ladder to incline_platform

nazwa sygnału	opis
hcount[10:0]	Licznik poziomy VGA
vcount[10:0]	Licznik pionowy VGA
hsync	Sygnał synchronizacji poziomej VGA
vsync	Sygnał synchronizacji pionowej VGA
hblnk	Sygnał blank poziomy VGA
vblnk	Sygnał blank pionowy VGA
rgb[11:0]	Sygnał rgb VGA

i) incline_platform_if – incline_platform to animation_platform

nazwa sygnału	Opis
hcount[10:0]	Licznik poziomy VGA
vcount[10:0]	Licznik pionowy VGA
hsync	Sygnał synchronizacji poziomej VGA
vsync	Sygnał synchronizacji pionowej VGA
hblnk	Sygnał blank poziomy VGA
vblnk	Sygnał blank pionowy VGA
rgb[11:0]	Sygnał rgb VGA

j) animation_platform _if – animation_platform to animation_ladder

nazwa sygnału	opis
hcount[10:0]	Licznik poziomy VGA
vcount[10:0]	Licznik pionowy VGA
hsync	Sygnał synchronizacji poziomej VGA
vsync	Sygnał synchronizacji pionowej VGA
hblnk	Sygnał blank poziomy VGA
vblnk	Sygnał blank pionowy VGA
rgb[11:0]	Sygnał rgb VGA

k) animation_ladder _if – animation_ladder to draw_animation_kong

nazwa sygnału	opis
hcount[10:0]	Licznik poziomy VGA
vcount[10:0]	Licznik pionowy VGA
hsync	Sygnał synchronizacji poziomej VGA
vsync	Sygnał synchronizacji pionowej VGA

hblnk	Sygnał blank poziomy VGA
vblnk	Sygnał blank pionowy VGA
rgb[11:0]	Sygnał rgb VGA

l) draw_animation_kong _if – draw_animation_kong to draw_kong

nazwa sygnału	opis
hcount[10:0]	Licznik poziomy VGA
vcount[10:0]	Licznik pionowy VGA
hsync	Sygnał synchronizacji poziomej VGA
vsync	Sygnał synchronizacji pionowej VGA
hblnk	Sygnał blank poziomy VGA
vblnk	Sygnał blank pionowy VGA
rgb[11:0]	Sygnał rgb VGA

m) draw_kong _if – draw_kong to draw_donkey

nazwa sygnału	Opis
hcount[10:0]	Licznik poziomy VGA
vcount[10:0]	Licznik pionowy VGA
hsync	Sygnał synchronizacji poziomej VGA
vsync	Sygnał synchronizacji pionowej VGA
hblnk	Sygnał blank poziomy VGA
vblnk	Sygnał blank pionowy VGA
rgb[11:0]	Sygnał rgb VGA

n) draw_donkey _if – draw_donkey to draw_shield

nazwa sygnału	opis
hcount[10:0]	Licznik poziomy VGA
vcount[10:0]	Licznik pionowy VGA
hsync	Sygnał synchronizacji poziomej VGA
vsync	Sygnał synchronizacji pionowej VGA
hblnk	Sygnał blank poziomy VGA
vblnk	Sygnał blank pionowy VGA
rgb[11:0]	Sygnał rgb VGA

o) draw_shield _if – draw_shield to draw_health

nazwa sygnału	opis
hcount[10:0]	Licznik poziomy VGA
vcount[10:0]	Licznik pionowy VGA
hsync	Sygnał synchronizacji poziomej VGA
vsync	Sygnał synchronizacji pionowej VGA
hblnk	Sygnał blank poziomy VGA
vblnk	Sygnał blank pionowy VGA
rgb[11:0]	Sygnał rgb VGA

p) draw_health _if – draw_health to draw_lady

nazwa sygnału	opis
hcount[10:0]	Licznik poziomy VGA
vcount[10:0]	Licznik pionowy VGA
hsync	Sygnał synchronizacji poziomej VGA
vsync	Sygnał synchronizacji pionowej VGA
hblnk	Sygnał blank poziomy VGA
vblnk	Sygnał blank pionowy VGA
rgb[11:0]	Sygnał rgb VGA

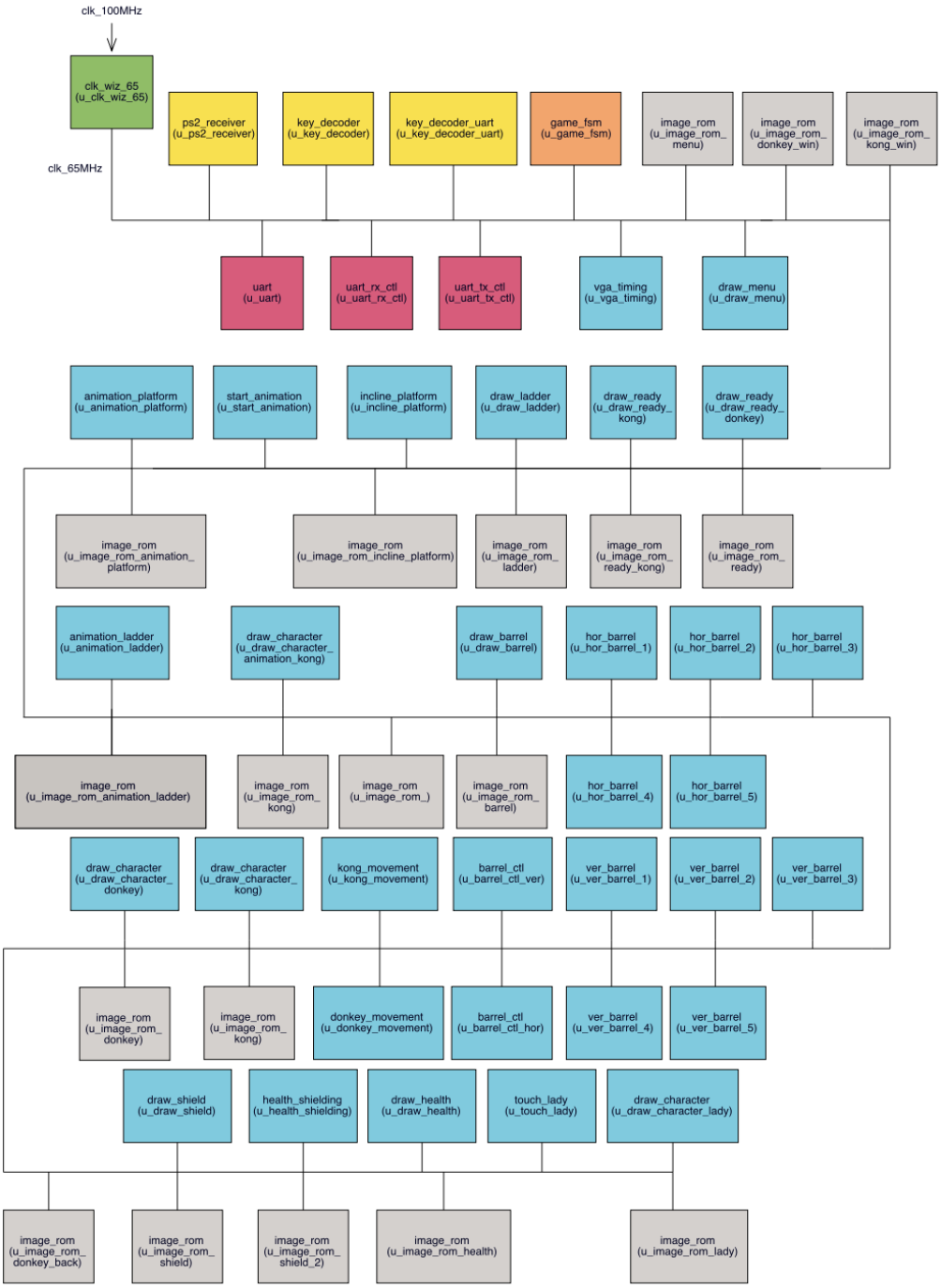
q) draw_lady _if – draw_lady

nazwa sygnału	opis
hcount[10:0]	Licznik poziomy VGA
vcount[10:0]	Licznik pionowy VGA
hsync	Sygnał synchronizacji poziomej VGA
vsync	Sygnał synchronizacji pionowej VGA
hblnk	Sygnał blank poziomy VGA
vblnk	Sygnał blank pionowy VGA
rgb[11:0]	Sygnał rgb VGA

4.2. Rozprowadzenie sygnału zegara

Osoba odpowiedzialna: DB, JB

W projekcie zastosowano jedną częstotliwość zegara 65MHz, która została wygenerowana w module clk_wiz_65.



5. Implementacja

5.1. Lista zignorowanych ostrzeżeń Vivado.

Identyfikator ostrzeżenia	Liczba wystąpień	Uzasadnienie
Synth 8-7080	1	Jest to jedynie ostrzeżenie, że zysk z równoległej syntezy byłby znikomy, więc Vivado nie rozdziela zadań na wątki.

5.2. Wykorzystanie zasobów

Tabela z wykorzystaniem zasobów z Vivado

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

Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	4123	20800	19.82
LUTRAM	25	9600	0.26
FF	2807	41600	6.75
BRAM	29	50	58.00
IO	20	106	18.87
BUFG	2	32	6.25
MMCM	1	5	20.00

5.3. Marginesy czasowe

Marginesy czasowe (WNS) dla setup i hold.

Timing **Setup** | Hold | Pulse Width

Worst Negative Slack (WNS): 5.122 ns

Timing Setup | **Hold** | Pulse Width

Worst Hold Slack (WHS): 0.023 ns

6. Konfiguracja sprzętu

Schemat połączenia ze sobą płytek Basys3 w trybie multiplayer.

Bassys 1

Bassys 2

7. Film.

Link do ściągnięcia filmu:

https://drive.google.com/drive/folders/1M9-373D5wedsBxJr0NkqhG6PQ1t-hx--?usp=sharing