Tytuł: **Basys Invaders**

Autorzy: **Tomasz Sieja (TS), Antoni Sus (AS)**

Ostatnia modyfikacja: 31.08.2025

Spis treści

1. Repozytorium git....1
2. Wstęp....2
3. Specyfikacja....2
   3.1. Opis ogólny algorytmu....2
   3.2. Tabela zdarzeń....3
4. Architektura....4
   4.1. Moduł: top....4
      4.1.1. Schemat blokowy....4
      4.1.2. Porty....5
         a) keyboard – keyboard_ctl, input....5
         b) vga – vga_ctl, output....5
         c) uart – uart_ctl, input....5
         d) uart – uart_ctl, output....5
      4.1.3. Interfejsy....5
         a) vga_if – vga signal pipeline....5
   4.2. Rozprowadzenie sygnału zegara....6
5. Implementacja....6
   5.1. Lista zignorowanych ostrzeżeń Vivado....6
   5.2. Wykorzystanie zasobów....7
   5.3. Marginesy czasowe....7
6. Konfiguracja sprzętu....8
7. Film....9

1. Repozytorium git

Adres repozytorium GITa:

https://github.com/Antoni-S/basys-invaders

2. Wstęp

Pomysł na tę grę pojawił się w momencie gdy dowiedzieliśmy się, że projekt może być prostą grą na płytkę FPGA. Jest to dobra okazja aby powrócić do gier klasycznych i spróbować naszych sił w odwzorowaniu ich. W ramach tego projektu staramy się odwzorować Space Invaders, ale z dodanym trybem kooperacyjnym i lekko zmienionym wyglądem.

3. Specyfikacja

3.1. Opis ogólny algorytmu

Algorytm gry jest strukturalnie prosty. Działa na zasadzie maszyny stanów, która przechodzi pomiędzy stanami TITLE, GAME i ENDSCREEN.

Po włączeniu programu przechodzimy do stanu TITLE, w którym czekamy na sygnał wciśniętego klawisza Enter, aby rozpocząć rozgrywkę i przejść do stanu GAME.

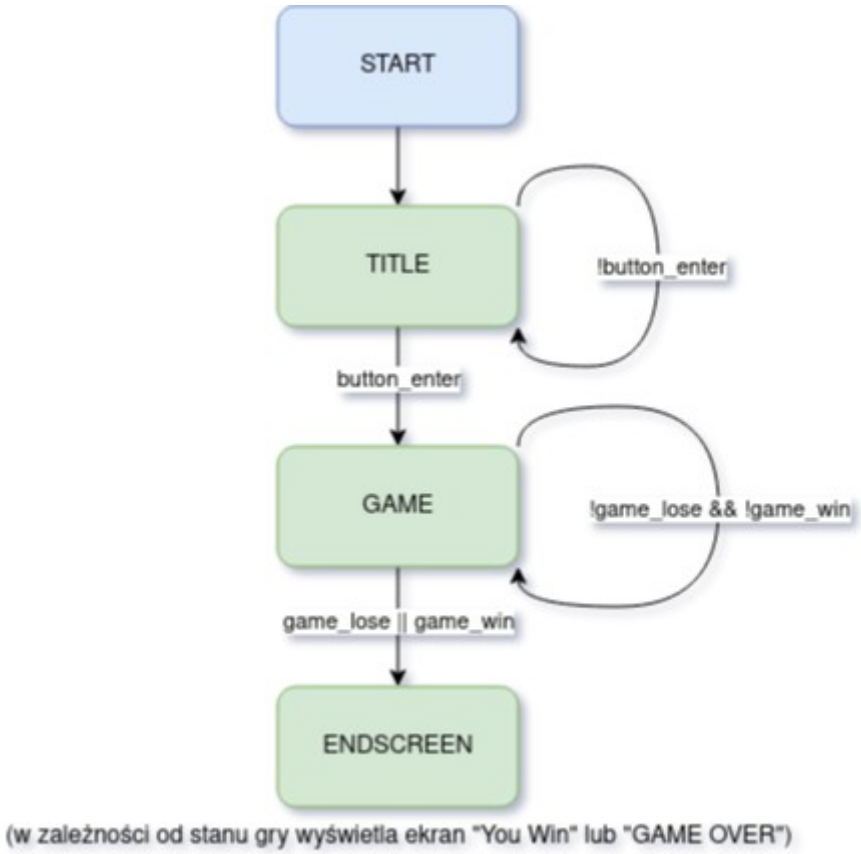
W tym stanie dzieje się kilka rzeczy, mianowicie przeciwnicy ciągle poruszają się po ekranie schematem PRAWO → DÓŁ → LEWO → DÓŁ, w tym czasie gracz może kontrolować swoją postać – statek kosmiczny, którym może poruszać się w prawo i w lewo dzięki klawiszom A i D. Gracz jest w stanie też wystrzelić pocisk, który leci w pozycji X odpowiadającej pozycji X gracza w momencie, którym gracz wcisnął przycisk W. Pocisk ten leci po prostej linii do góry ekranu i koliduje albo z górną krawędzią wyświetlacza, albo przeciwnikiem. W tym czasie gracz nie jest w stanie wystrzelić nowego pocisku. W przypadku „nałożenia się" pocisku i przeciwnika, dochodzi do kolizji i kosmita oraz rakieta są usuwane z ekranu a graczowi odblokowuje się możliwość wystrzelenia pocisku, co może poskutkować szybszym „wyczyszczeniem" przeciwników z planszy.

Aby przejść do następnego stanu, musi się wydarzyć jedna z dwóch rzeczy:

1. *Graczowi powiodło się usunięcie wszystkich przeciwników z pola gry*
2. *Najniższy żywy przeciwnik przedostał się na dolną część ekranu (odległość od dolnej krawędzi ekranu wynosi 70px)*

Po spełnieniu jednego z tych warunków, gra przechodzi w stan ENDSCREEN, który wyświetla na ekranie słowa „You Win!" w wypadku wygranej lub „GAME OVER" jeśli gracz przegrał grę.

W grze wieloosobowej gracze „wyścigują się" w tym, kto pierwszy jest w stanie pokonać nadchodzącą inwazję. Na obydwu ekranach wyświetla się „duch" drugiego gracza.



3.2. Tabela zdarzeń

Zdarzenie	**Kategoria**	**Reakcja systemu**
Przycisk Enter	Ekran startowy	Uruchomienie gry
Przycisk „A”	Klawiatura	Statek porusza się w lewo
Przycisk „D”	Klawiatura	Statek porusza się w prawo
Przycisk „W”	Klawiatura	Statek wypuszcza pocisk, zapisywana jest pozycja gracza
Wystrzelony pocisk	Gracz	Pocisk porusza się w prostej linii do góry, w pozycji X w której gracz kliknął przycisk „W”. Gracz nie może wystrzelić kolejnego pocisku
Pocisk zderza się z przeciwnikiem	Gracz	Przeciwnik zostaje usunięty z „planszy”. Gracz może wystrzelić kolejny pocisk
Pocisk zderza się z górną krawędzią ekranu	Gracz	Pocisk zostaje usunięty. Gracz może wystrzelić kolejny pocisk
Wszyscy przeciwnicy usunięci z „planszy”	Gra	Zakończenie gry, ekran zmienia się na ekran wygranej
Przeciwnik zdążył przejść na dolną część ekranu (odległość od dolnej	Gra	Zakończenie gry, ekran zmienia się na ekran przegranej

części ekranu 70px)		

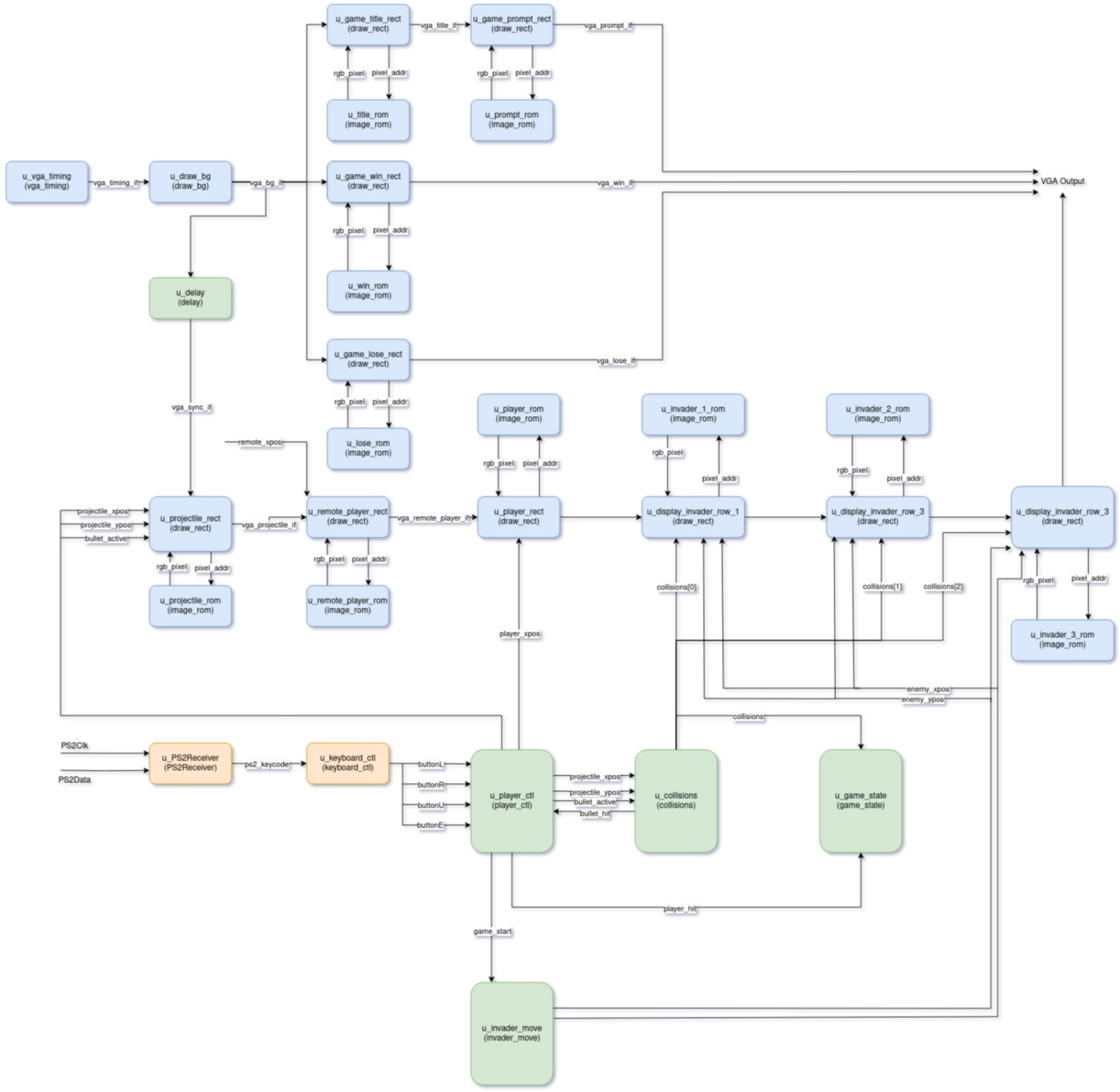
4. Architektura

4.1. Moduł: top

Osoba odpowiedzialna: AS, TS

Moduł top_vga jest złożony z podmodułów odpowiedzialnych za wyświetlanie elementów na ekranie, kontrolowanie ich oraz z modułów pomocniczych (np. pamięci ROM lub modułu delay).

4.1.1. Schemat blokowy



4.1.2. Porty

a) keyboard – keyboard_ctl, input

nazwa portu	opis
PS2Data	szeregowe wejście danych z klawiatury
PS2Clk	zegar klawiatury

b) 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

c) uart – uart_ctl, input

nazwa portu	opis
test_Rx	sygnał wejściowy UART używany do testu poprawności działania
JB1	sygnał wejściowy UART odpowiedzialny za przekazywanie pozycji gracza

d) uart – uart_ctl, output

nazwa portu	opis
test_Tx	sygnał wyjściowy UART używany do testu poprawności działania
JC1	sygnał wyjściowy UART odpowiedzialny za przekazywanie pozycji gracza

4.1.3. Interfejsy

a) vga_if – vga signal pipeline

nazwa sygnału	opis
hcount [10:0]	licznik horyzontalny VGA
vcount [10:0]	licznik wertykalny VGA
hsync	synchronizacja pozioma VGA
vsync	synchronizacja pionowa VGA
hblnk	sygnał blank poziomy VGA
vblnk	sygnał blank pionowy VGA
rgb [11:0]	sygnał z informacją o kolorze

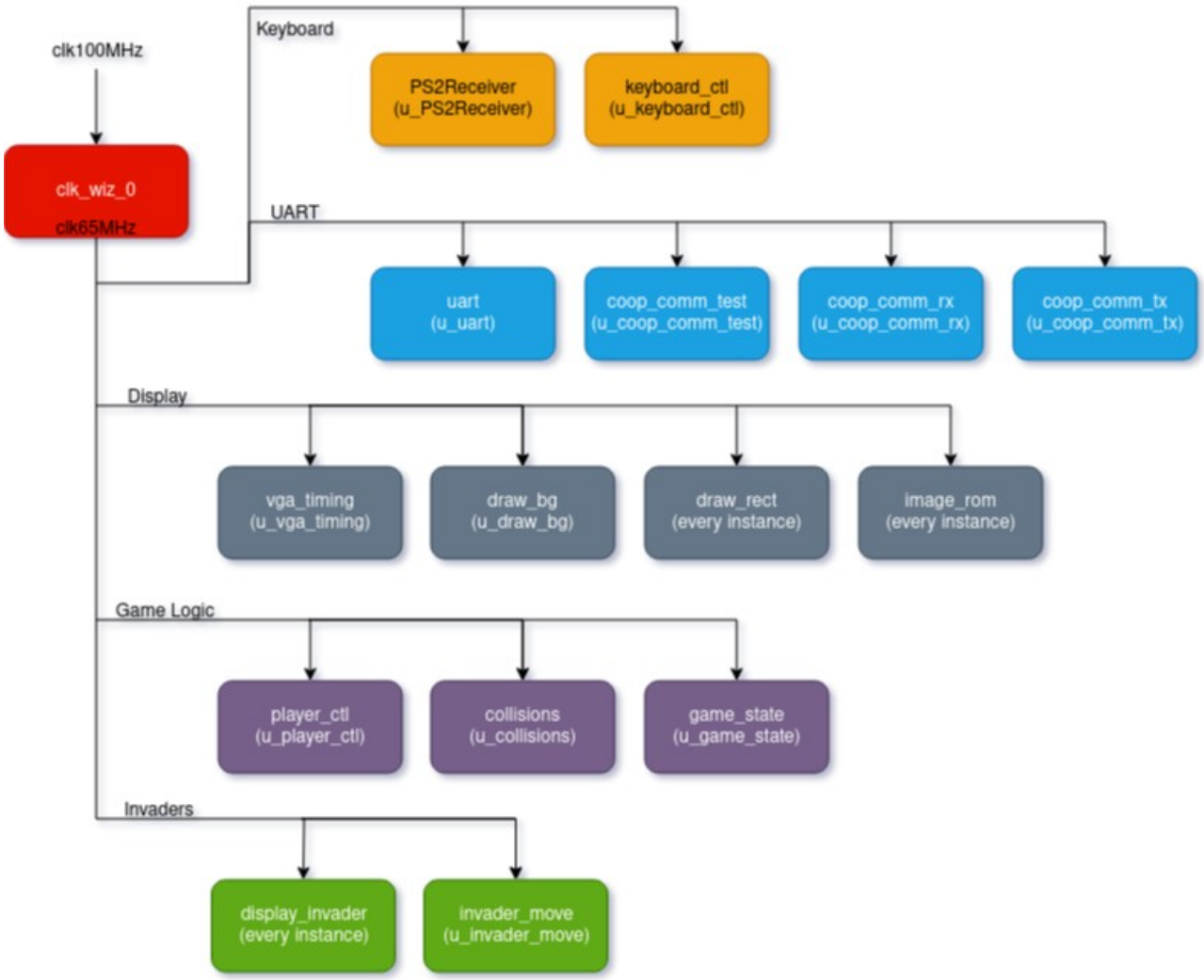
Każdy moduł, który w jakikolwiek sposób jest rysowany (wyświetlany) na ekranie korzysta z powyższego interfejsu vga_if. Dla czytelności, każdy interfejs w projekcie jest nazywany zgodnie z

konwencją vga_[element_wyświetlany]_if. Dla przejrzystości tego raportu, nie będzie wymieniana każda instancja tego interfejsu.

4.2. Rozprowadzenie sygnału zegara

Osoba odpowiedzialna: TS

Cały projekt działa na zegarze o częstotliwości 65 MHz, który został wygenerowany za pomocą Clock Wizard-a w Vivado. Wartość tej częstotliwości wynika z wymagania projektu dyktującego minimalną rozdzielczość monitora jako 1024x768 pikseli.



5. Implementacja

5.1. Lista zignorowanych ostrzeżeń Vivado.

Identyfikator ostrzeżenia	Liczba wystąpień	Uzasadnienie
8-7080	1	Projekt nie jest na tyle rozbudowany, aby spełniać zasady

		równoległej syntezy
8-7129	2	Zegar 100 MHz nie jest używany w projekcie
8-6014	1	Element FIFO_Rx instancji modułu u_uart_test nie jest wykorzystywany ponieważ ten moduł UART służy tylko do wysyłania danych do terminala w celu debugowania
8-3332	2	Element FSM_sequential_state_reg w instancji modułu u_uart_rx_unit nie jest wykorzystywany ponieważ ten moduł UART służy tylko do wysyłania danych do terminala w celu debugowania

5.2. Wykorzystanie zasobów

Tabela z wykorzystaniem zasobów z Vivado

Resource	Utilization	Available	Utilization %
LUT	2259	20800	10.86
LUTRAM	40	9600	0.42
FF	975	41600	2.34
BRAM	11	50	22.00
IO	23	106	21.70
BUFG	2	32	6.25
MMCM	1	5	20.00

Marginesy czasowe

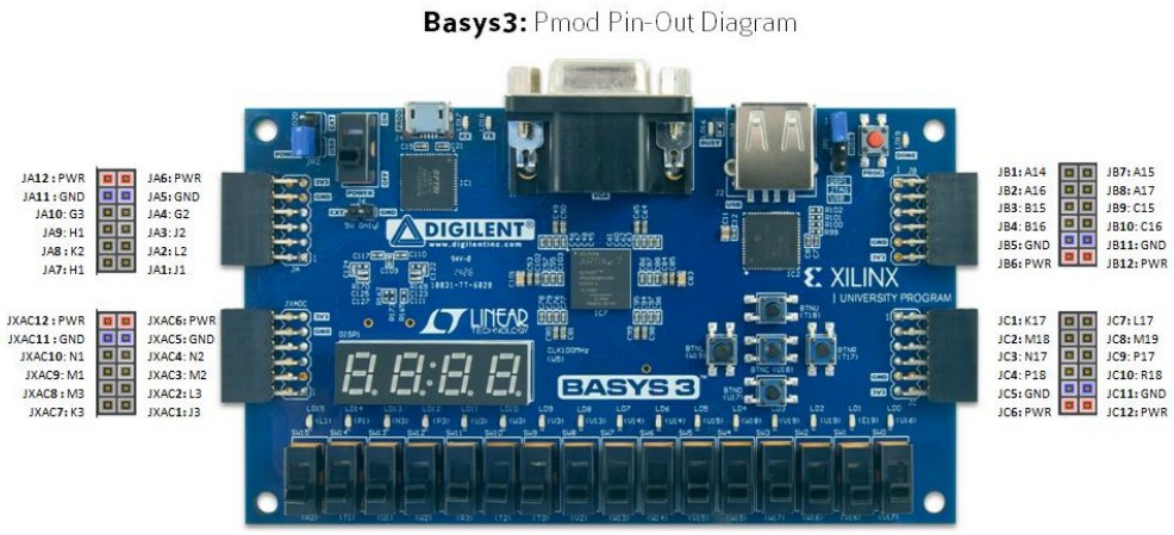
Marginesy czasowe (WNS) dla setup i hold.

Timing	**Setup** | Hold |
Worst Negative Slack (WNS):	3.863 ns
Total Negative Slack (TNS):	0 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	1676

Timing	Setup | **Hold** |
Worst Hold Slack (WHS):	0.052 ns
Total Hold Slack (THS):	0 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	1676

6. Konfiguracja sprzętu

Schemat połączenia ze sobą płytek Basys3 w trybie multiplayer:

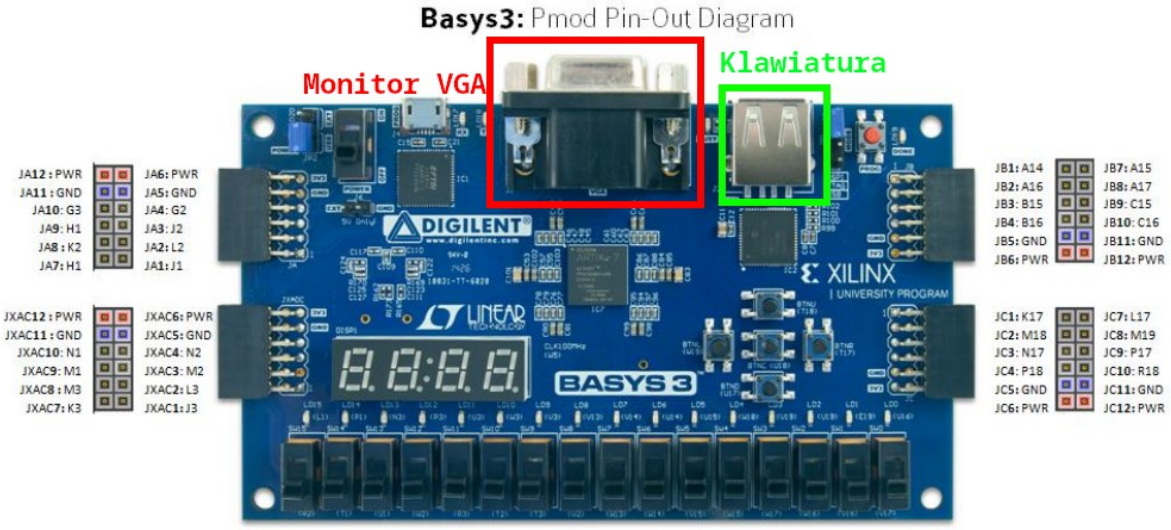


Źródło: https://digilent.com/reference/_media/basys3:basys3_rm.pdf

1. JB1 - UART linia RX
2. JC1 – UART linia TX

Podłączane piny na płytce Basys3	
Płytka 1	Płytka 2
JB1 (Rx)	JC1(Tx)
JB5 (GND)	JC5 (GND)
JC1 (Tx)	JB1 (Rx)
JC5 (GND)	JB5 (GND)

Schematy podłączenia dodatkowych urządzeń peryferyjnych:



7. Film.

Link do ściągnięcia filmu:

https://drive.google.com/file/d/13FWwOM_-nNacsuViIdWcqBM9EEG047PF