Tytuł: Kot w worku

Autorzy: Zuzanna Kasperek (ZK), Joanna Iwanicka (JI)

Ostatnia modyfikacja: 28.08.2019

Spis treści

[1. Wstęp 1](#__RefHeading___Toc375_832888478)

[2. Specyfikacja 1](#__RefHeading___Toc377_832888478)

[2.1. Opis ogólny algorytmu 1](#__RefHeading___Toc379_832888478)

[2.2. Tabela zdarzeń 2](#__RefHeading___Toc381_832888478)

[3. Architektura 2](#__RefHeading___Toc383_832888478)

[3.1. Moduł: top 2](#__RefHeading___Toc385_832888478)

[3.1.1. Schemat blokowy 2](#__RefHeading___Toc387_832888478)

[3.1.2. Porty 2](#__RefHeading___Toc389_832888478)

[a) mou – mouse\_ctl, input 2](#__RefHeading___Toc391_832888478)

[b) vga – vga\_ctl, output 2](#__RefHeading___Toc393_832888478)

[3.1.3. Interfejsy 3](#__RefHeading___Toc453_832888478)

[a) m2c – mouse\_ctl to core 3](#__RefHeading___Toc395_832888478)

[3.2. Moduł: mouse\_ctl (external IP) 3](#__RefHeading___Toc397_832888478)

[3.3. Moduł: core 3](#__RefHeading___Toc399_832888478)

[3.3.1. Schemat blokowy 3](#__RefHeading___Toc455_832888478)

[3.3.2. Porty 3](#__RefHeading___Toc467_832888478)

[3.3.3. Interfejsy 3](#__RefHeading___Toc457_832888478)

[3.4. Moduł: core:timer 3](#__RefHeading___Toc1113_832888478)

[3.4.1. Schemat blokowy 3](#__RefHeading___Toc455_8328884781)

[3.4.2. Porty 3](#__RefHeading___Toc467_8328884781)

[3.4.3. Interfejsy 3](#__RefHeading___Toc457_8328884781)

[3.5. Rozprowadzenie sygnału zegara 3](#__RefHeading___Toc1150_832888478)

[4. Implementacja. Zaawansowanie na 14.03.2018 – 0% 4](#__RefHeading___Toc996_1436539759)

[5. Film. Zaawansowanie na 14.03.2018 – 0% 4](#__RefHeading___Toc1249_1436539759)

# Repozytorium git

Adres repozytorium GITa (jeżeli używane):

https://

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

# Wstęp

*Gra polega na złapaniu kota do pudełka sterowanego za pomocą myszki. Wiele popularnych gier, na których się wzorowałyśmy ma podobny sposób działania, który pozwala na poprawienie zręczności oraz pomaga się odstresować.*

# Specyfikacja

## Opis ogólny algorytmu

*By rozpocząć grę klikamy LPM, następnie widzimy kota spadającego z górnej części ekranu. Za pomocą myszki próbujemy go złapać. Gdy to się uda uzyskujemy punkt, gdy kot spadnie poza pudełko gracz przegrywa. By zacząć grę od początku należy wcisnąć PPM.*

*Uproszczony schemat działania implementowanego algorytmu. Co się dzieje po starcie, jak wygląda przebieg działania, kiedy i pod jakimi warunkami się kończy.*

*Ewentualnie przykładowe screen-shoty tego, co w przybliżeniu chcielibyśmy uzyskać.*

## 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 | Początek gry | Uruchomienie gry |
| Kot zostaje złapany do worka/pudełka | Gra | Gracz uzyskuje punkt |
| Kot spada poza obszarem worka/pudełka i uderza w dolny koniec? ekranu | Gra | Koniec gry |
| PPM | Koniec gry | Liczba punktów jest resetowana, użytkownik przechodzi do początku gry |

# Architektura

Uwaga: dobrze zrobiony projekt zawiera tylko moduły strukturalne (zbudowane z innych modułów) i funkcjonalne (zawierające bloki proceduralne always @). Staramy się nie generować bloków mieszających te dwa typy, o ile to możliwe.

Uwaga: opisujemy architekturę **tylko głównego modułu oraz rozprowadzenie sygnału zegara**.

## Moduł: top

Osoba odpowiedzialna: JK

### Schemat blokowy

Uwaga: Schemat blokowy to nie jest schemat z Vivado! Nie zawiera on sygnałów, tylko interfejsy. Interfejs oznacza tutaj grupę sygnałów. Schemat blokowy pokazuje moduły składowe, oraz łączące je interfejsy.

*Przykładowy schemat blokowy modułu głównego*

*Uwaga:*

* *interfejsy dwukierunkowe rozbijamy na 2 interfejsy jednokierunkowe*
* *nazwa interfejsu stanowi prefiks nazwy sygnałów składowych*
* *w interfejsach nie uwzględniamy sygnałów globalnych (np. clk i rst).*

### Porty

#### mou – mouse\_ctl, input

|  |  |
| --- | --- |
| **nazwa portu** | **opis** |
| mou\_si | szeregowe wejście danych |
|  |  |

#### vga – vga\_ctl, output

|  |  |
| --- | --- |
| **nazwa portu** | **opis** |
| vga\_vs | sygnał synchronizacji pionowej VGA |
|  |  |

### Interfejsy

#### m2c – mouse\_ctl to core

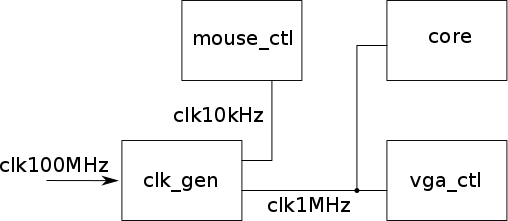
|  |  |
| --- | --- |
| **nazwa sygnału** | **opis** |
| m2c\_x[9:0] | horyzontalna pozycja kursora myszy na ekranie |
| m2c\_y[9:0] | wertykalna pozycja kursora myszy na ekranie |
|  |  |

## Rozprowadzenie sygnału zegara

Osoba odpowiedzialna: JK

*Informacja na temat źródła sygnału zegarowego, używanych częstotliwości zegara w całym układzie.*

*Moduł generatora zegara umieszczamy w module głównym projektu. W pozostałych modułach używamy tylko i wyłącznie sygnały zegara wygenerowane przez ten moduł.*

*Uwaga: jeżeli używamy różnych częstotliwości zegara w układzie, to należy je tak dobrać, aby były wielokrotnościami siebie (umożliwia to wygenerowanie tych sygnałów z jednego IP core generatora zegara i zapobiega problemom z synchronizacją).*

# Implementacja. Zaawansowanie na 14.03.2018 – 0%

# Film. Zaawansowanie na 14.03.2018 – 0%

Link do ściągnięcia filmu:

https://