**UCiSW 2 projekt – gra Pong**

Marek Machliński (241308)

Daniel Król (241399)

Prowadzący: dr inż. Jacek Mazurkiewicz

Spis treści

[Cel projektu 2](#_Toc39943932)

[Opis komponentów 2](#_Toc39943933)

[Podsumowanie 2](#_Toc39943934)

# Cel projektu

Celem projektu było wykonanie programu w VHDL realizującego grę Pong, który powinien zostać uruchomiony na zestawie Spartan3e Starter Kit. Gra oferuje możliwość rozgrywki dla dwóch graczy za pomocą jednej klawiatury. Sterowanie dla każdego z graczy odbywa się z użyciem dwóch klawiszy: W i S oraz strzałka w górę i strzałka w dół. Dodatkowo istnieje opcja resetu rozgrywki za pomocą klawisza R i pauzy za pomocą klawisza P. Każdy z graczy dysponuje paletkę, którą może poruszać w górą lub w dół ekranu. Między graczami odbija się piłka, która jeśli przekroczy linię paletki któregoś z nich, to przeciwnik zdobywa punkt. Zadaniem graczy jest wprowadzanie paletki na kurs kolizyjny z piłką. Dodatkowo piłka może odbijać się od górnej i dolnej krawędzi ekranu. Po osiągnięciu maksymalnej liczby punktów jeden z graczy wygrywa. W grze pojawiają się także losowe ulepszenia, które gracze mogą zebrać, aby zmodyfikować rozgrywkę (np. powiększenie piłki, powiększenie paletki, przyspieszenie paletki itd.).

# Opis komponentów

Każdy z komponentów został przetestowany przy użyciu symulacji behawioralnej. Nazwa każdego z obrazów poniżej nawiązuje do nazwy jego pliku testowego.

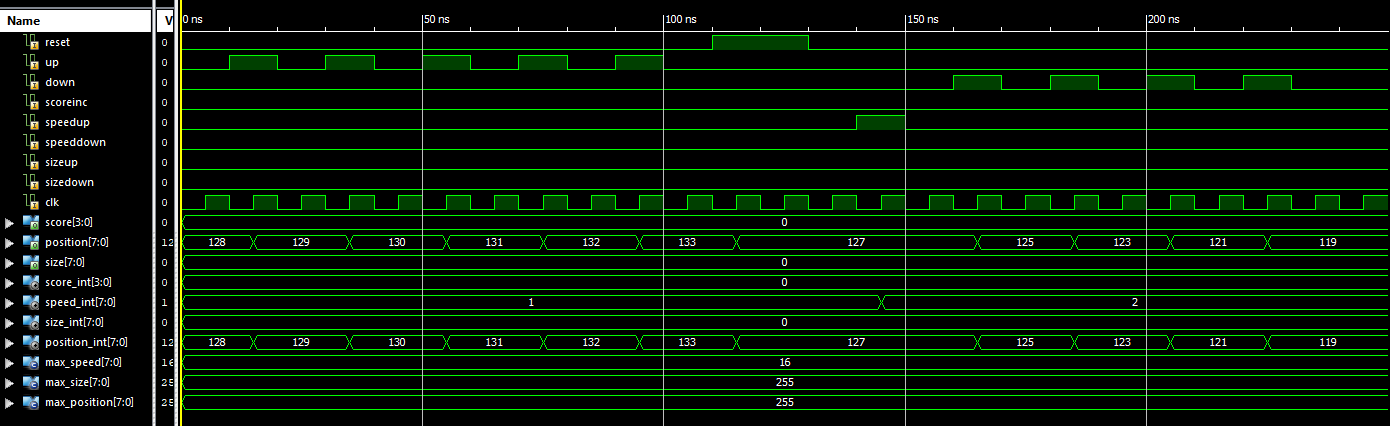
## Input Manager

InputManager odpowiada za obsługę odczytanych klawiszy, które zostały wciśnięte. Korzysta on z modułu PS2\_Kbd do odczytu kodów klawiszy z klawiatury. Jego logika polega na rozpoznaniu kodu wciśniętego klawisza za pomocą klauzuli case … when.

## Player

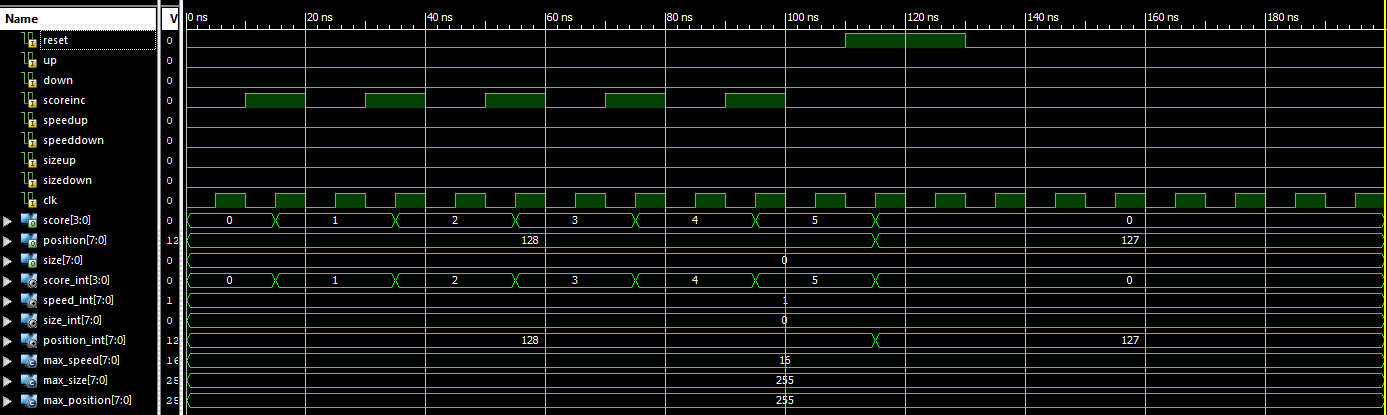
Instancja modułu Player jest osobna dla każdego z graczy. Posiada on proces odpowiedzialny za zwiększanie punktów, który może resetować liczbę punktów do zera w przypadku naciśnięcia resetu lub zwiększać liczbę punktów gracza w przypadku otrzymania sygnału zwiększenia punktów. Limitem punktów jest stała liczbowa. Drugi proces odpowiada za modyfikowanie prędkości poruszania się paletki gracza, która może zostać zwiększona do wartości maksymalnej lub zmniejszona do domyślnej (1 jednostka odległości na 1 jednostkę czasu). Analogicznie zachowuje się trzeci proces odpowiadający za modyfikację rozmiaru paletki gracza: może zostać powiększona do rozmiaru maksymalnego lub zmniejszona do rozmiaru minimalnego. Ostatni proces odpowiada za modyfikację pozycji paletki gracza, która jest obliczana na podstawie obecnej pozycji, docelowego kierunku poruszenia się i obecnej prędkości poruszania się paletki danego gracza.

Testowanie modułu polegało na obserwacji zmian wartości pozycji gracza zależnie od aktualnej prędkości poruszania się, pozycji i otrzymanego docelowego kierunku poruszania się. Paletka gracza powinna poruszać się od aktualnej pozycji gracza o wartość szybkości poruszania się w stronę zdefiniowaną przez sygnał kierunku poruszania. Dodatkowo w momencie odebrania sygnału przyspieszenia gracz powinien zacząć poruszać się ze zwiększoną szybkością:



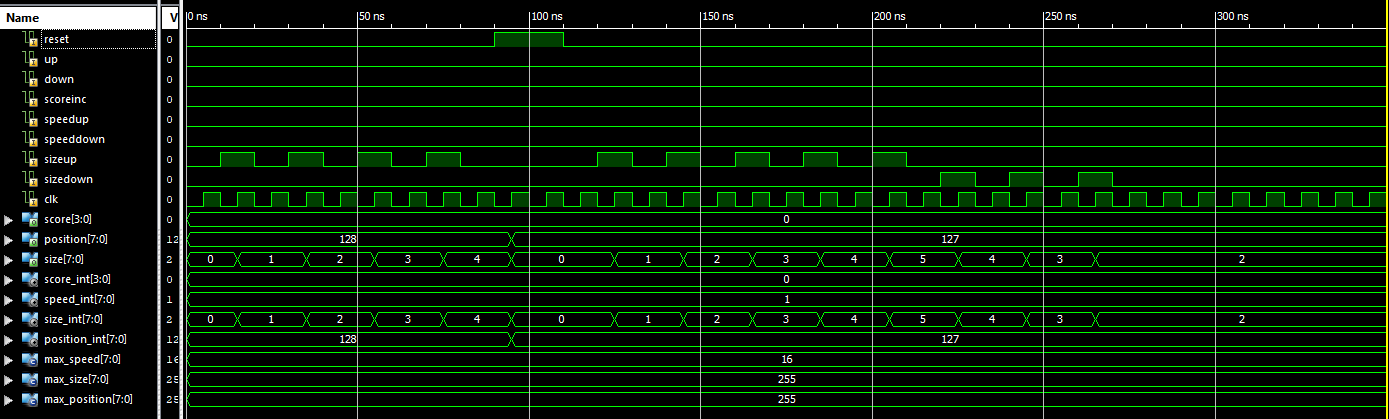
Player\_shouldMoveUpAndDown

Moduł gracza powinien reagować na sygnał resetu poprzez zresetowanie aktualnej wartości punktów gracza i przesunięciu go do pozycji domyślnej:



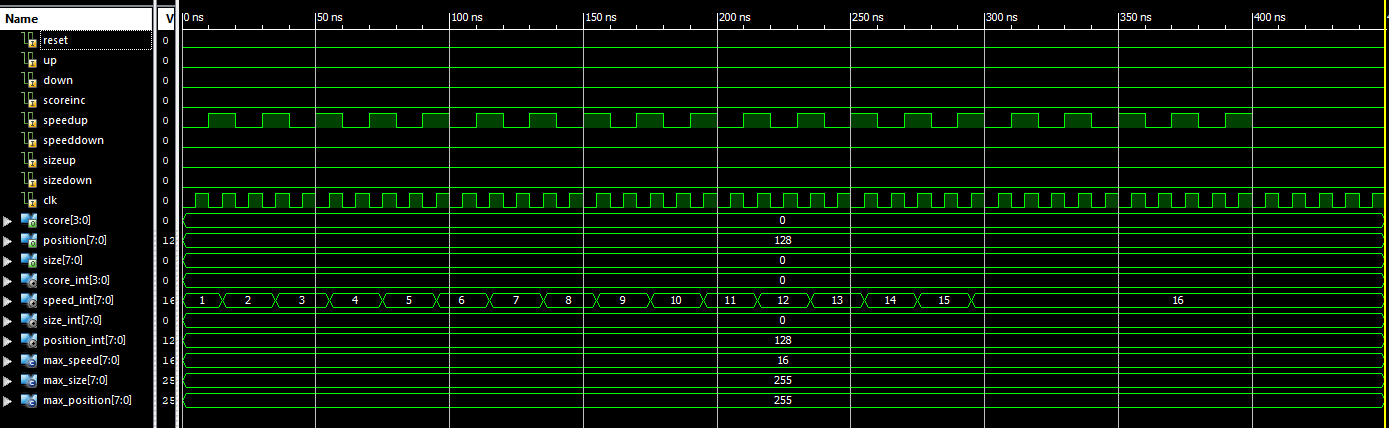
2 Player\_shouldResetScore

Gracz powinien również reagować na sygnały zwiększania i zmniejszania się rozmiaru jego paletki. Dla resetu wielkość paletki powinna być przywracana do domyślnej:



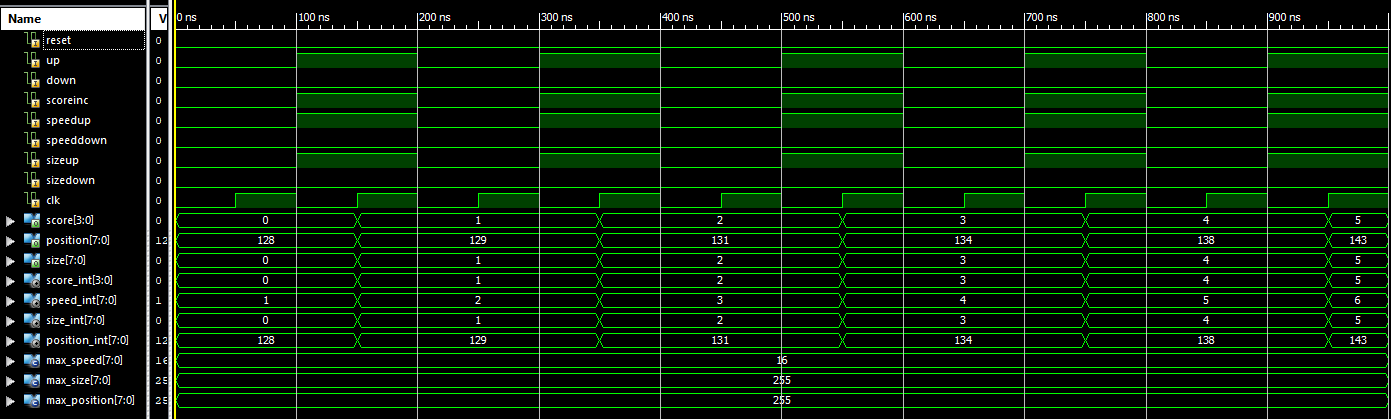
3 Player\_shouldSizeDown

Gracz powinien reagować na zmiany jego szybkości poruszania się uwzględniając maksymalną wartość prędkości, po osiągnięciu której nie zostaje ona zwiększana:



4 Player\_shouldSpeedUpToMaxSpeed

Ostatecznie gracz powinien reagować na wszystkie podane wcześniej sygnały tak samo w przypadku gdy występują one osobne jak i w momencie, gdy występują jednocześnie. Test obrazujący zachowanie tego modułu pod wpływem wszystkich sygnałów go dotyczących:

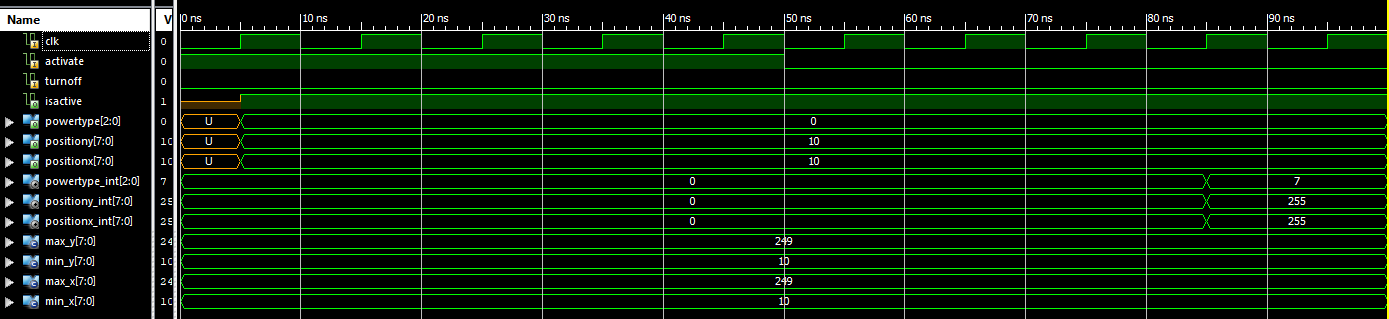


Player\_behavior

## PowerUp

Ten moduł odpowiada za zarządzanie stanem ulepszeń rozmieszczonych na mapie. Ulepszenia rozmieszczane są w losowych punktach, a w przypadku wylosowaniu pozycji dla ulepszenia poza obszarem gry pozycja ta zostaje przesunięta do krawędzi, poza którą ta pozycja wykracza. Dane ulepszenie może być aktywowane lub dezaktywowane. Moduł ten posiada submoduł RandGen, którego instancje odpowiadają kolejno za generowanie losowego typu ulepszenia (typ ulepszenia określa, czy ulepszeniem będzie np. powiększenie paletki czy powiększenie piłki), a także za generowanie losowej wartości pozycji ulepszenia osobno w płaszczyźnie X i Y.

Przykładowa generacja losowego ulepszenia:

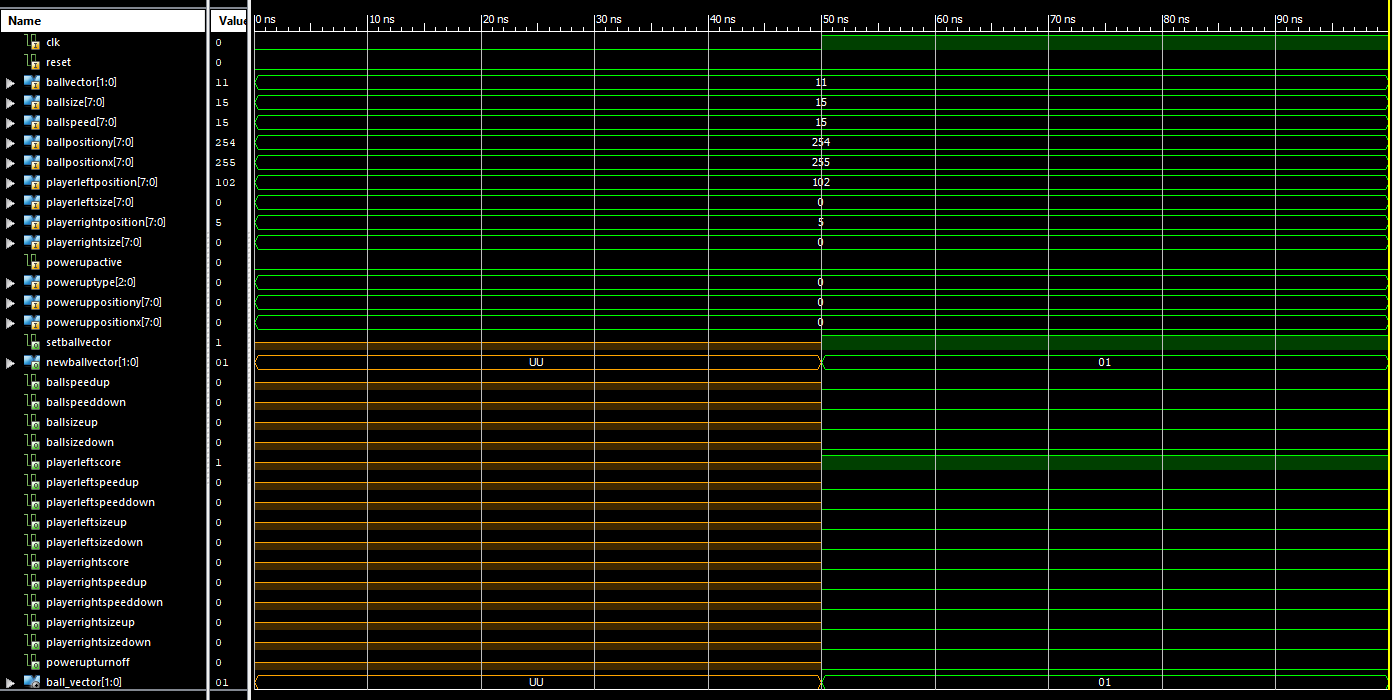


PowerUp\_shouldGenerateRandomPowerUp

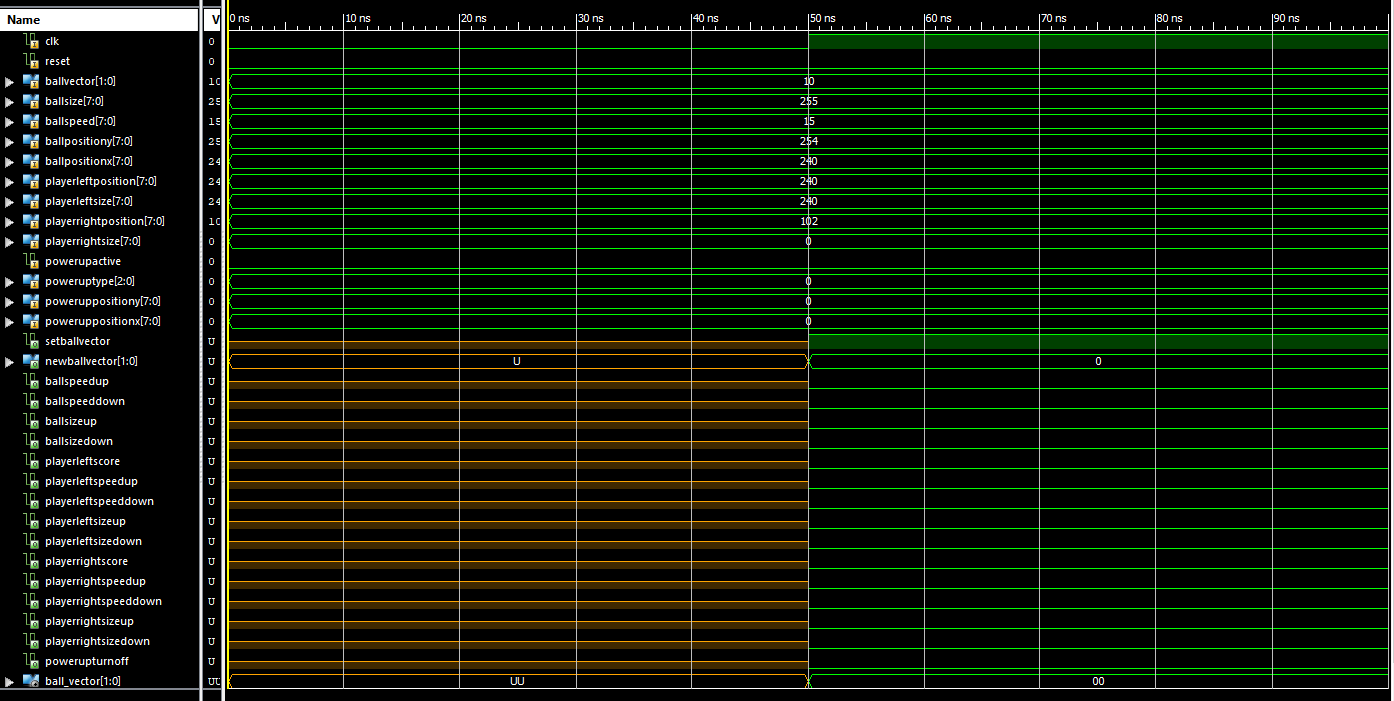
## CollisionManager

Jest to moduł odpowiadający za sprawdzanie kolizji podczas poruszania się piłki. Na początku ustalany jest rozmiar piłki na podstawie jej pozycji i aktualnego rozmiaru. Następnie sprawdzane są warunki matematyczne związane z pozycją paletki gracza (także z uwzględnieniem ulepszeń), pozycji piłki i wektora poruszania się piłki. Na podstawie tego najpierw wykrywana jest kolizja z prawą ścianą, a następnie sprawdzanie, czy w tym miejscu ściany znajdowała się paletka gracza. Jeśli nie było tam paletki, to punkt zyskuje gracz po lewej stronie. Analogiczne warunki są sprawdzane dla kolizji z lewą ścianą. Ostatnimi opcjami jest sprawdzanie kolizji z górną i dolną krawędzią ekranu, jednak wtedy ustalany jest jedynie kolejny wektor odbicia piłki bez sprawdzania warunków dotyczących punktacji. Drugi z procesów tego modułu odpowiada za sprawdzanie kolizji z ulepszeniami. Odbywa się to na podstawie obecnej pozycji piłki, obecnej szybkości poruszania się piłki, docelowego miejsca poruszenia się piłki i pozycji ulepszenia. W przypadku kolizji z ulepszeniem sprawdzany jest jego typ, a na podstawie tego typu aktywowane jest konkretne ulepszenie. Taka kolizja możliwa jest w czterech przypadkach, dlatego za sprawdzanie każdej z możliwych pozycji piłki względem ulepszenia odpowiedzialna jest osobna wartość w klauzuli case … when.

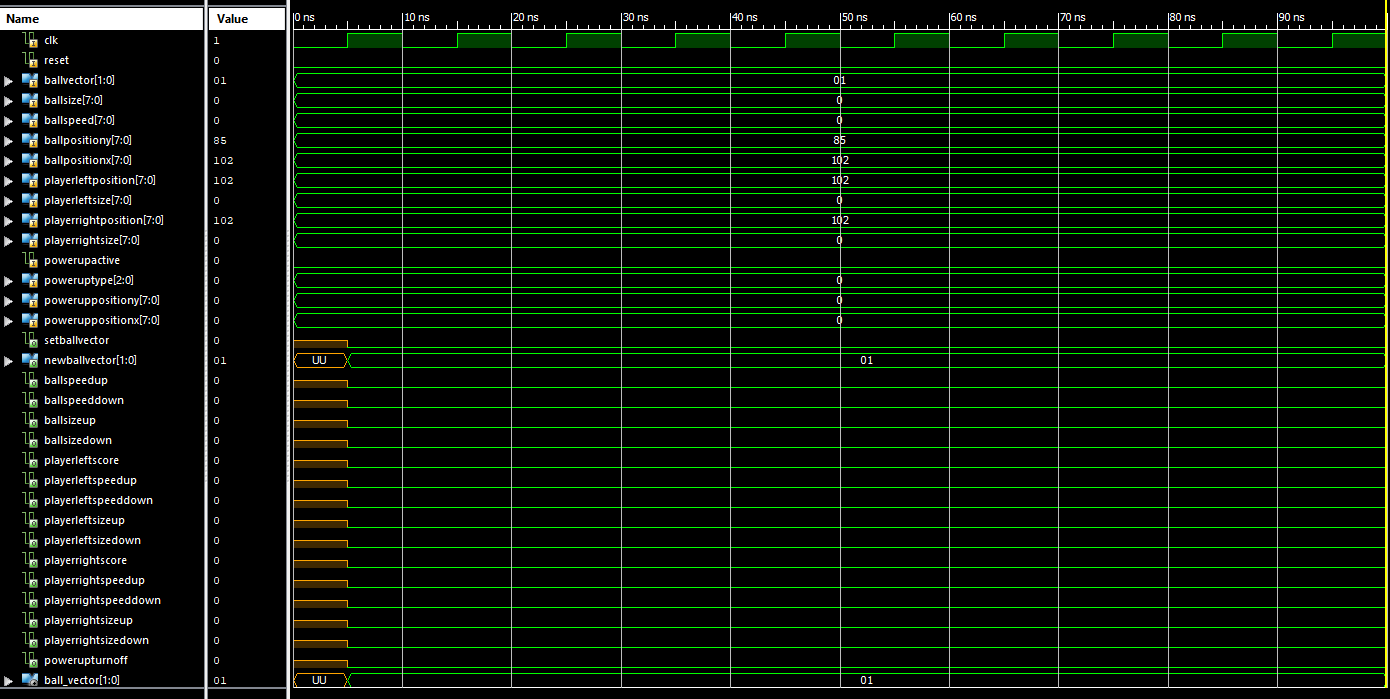
Testowanie tego modułu polegało na sprawdzeniu jaki wektor (czyli dwubitowy kierunek) odbicia piłki zostanie zwrócony dla konkretnych wartości pozycji paletki gracza, piłki i docelowego wektora poruszenia się. Dodatkowo w przypadku trafienia w jedną ze ścian graczy (gdy w tym momencie nie ma tam paletki) powinien być zdobywany punkt przez przeciwnika. Przykładowe wartości pozycji obiektów i reakcja na nie:



CollisionManager\_shouldHitRightWall



CollisionManager\_shouldBounceLeftWall



CollisionManager\_behavior

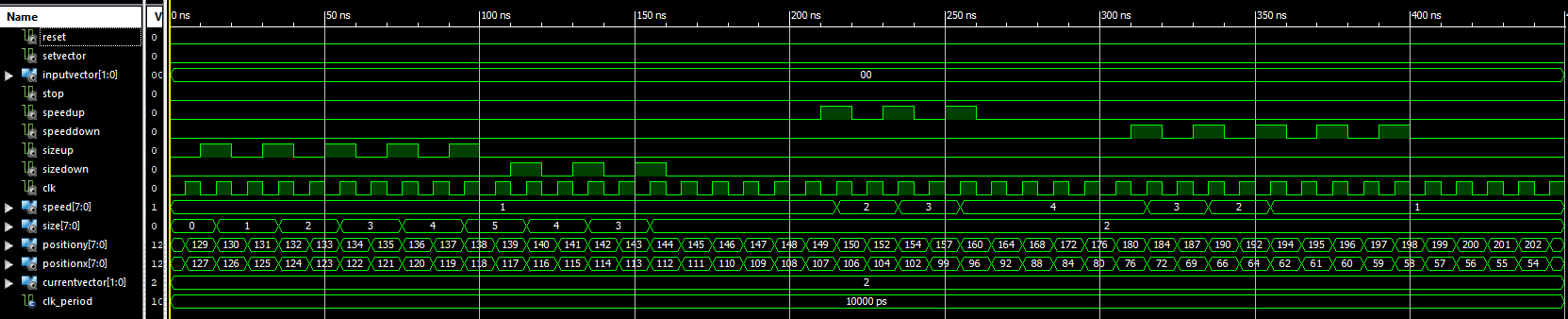
## RenderManager

Moduł ten nie został jeszcze zaimplementowany. W tym momencie pełni jedynie rolę interfejsu bez implementacji.

## Ball

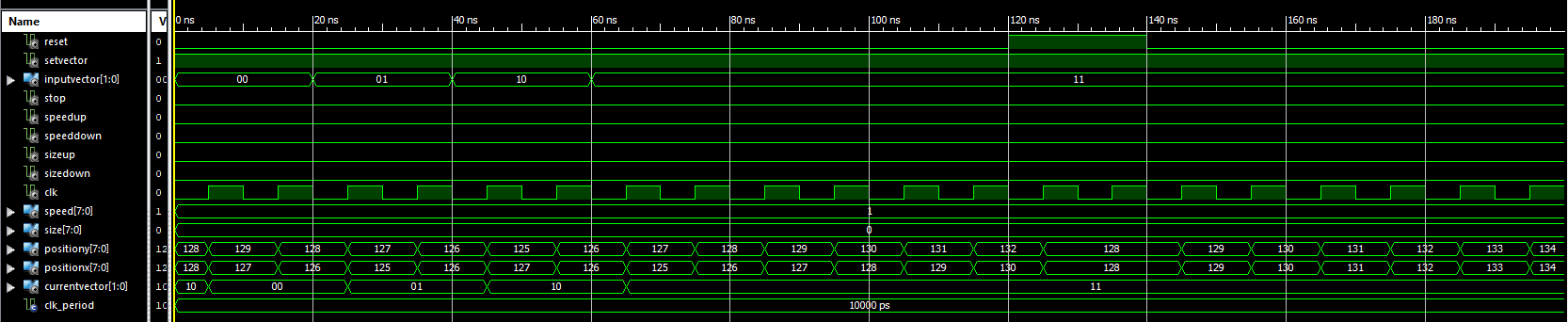
Moduł Ball posiada proces zarządzający prędkością poruszania się i rozmiarem piłki. Operuje on na sygnale Reset, w przypadku którego wystąpienia następuje zresetowanie tych dwóch wartości do wartości domyślnych. W przypadku aktywności któregoś z sygnału odpowiadającego za modyfikację prędkości czy rozmiaru piłki następuje obliczenie nowej wartości tej wielkości. Drugi proces zajmuje się odpowiada za zmianę pozycji piłki. W przypadku resetu zostaje ona ustawiona na wartość domyślną. W przypadku nastąpienia zmiany pozycji piłki jest ona obliczana na podstawie wektora kierunku przesunięcia piłki, który może przyjmować cztery wartości zależnie od kierunku.

Zmienianie się wartości pozycji piłki jest zależne od jej aktualnej pozycji, rozmiaru i prędkości. Piłka posiadając ciągle ten sam wektor poruszania się może zmieniać swoją pozycję w różny sposób zależnie od wartości prędkości czy rozmiaru:



10 Ball\_shouldChangeSpeedAndSize

W przypadku zmiany wartości wektora docelowego poruszenia się piłka powinna zaczynać poruszać się w kierunku zdefiniowanym przez aktualny wektor poruszania się:

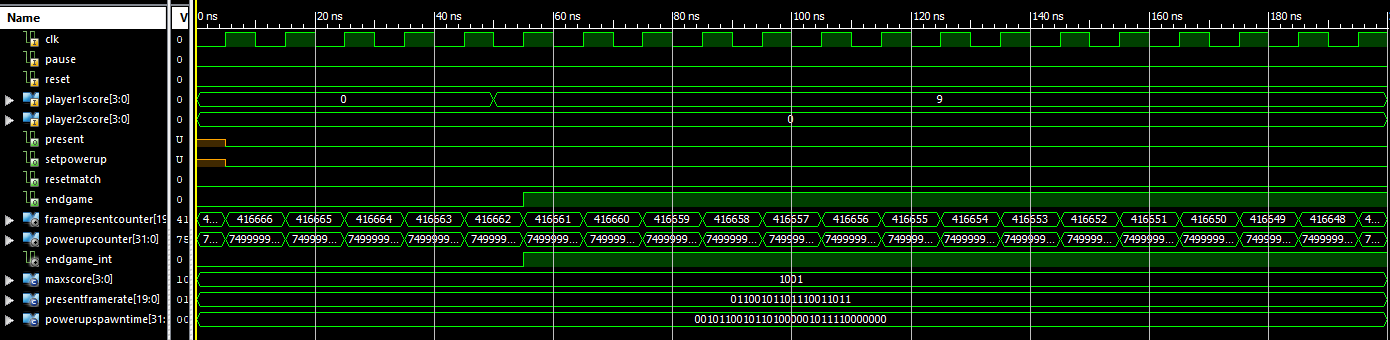


Ball\_shouldMoveByVector

## GameLogic

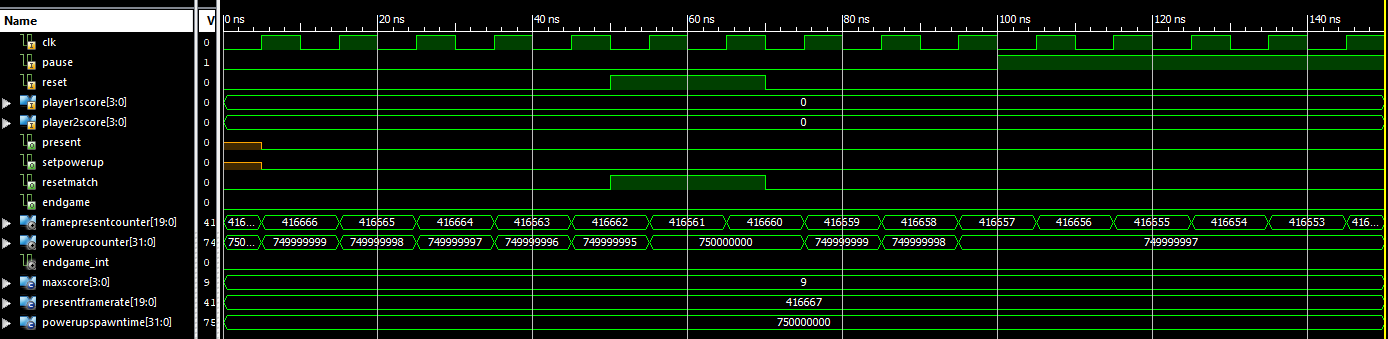
Ten moduł odpowiada za procesy związane z warunkami gry abstrahując od mechaniki jej działania. Pierwszy proces odpowiada za sprawdzanie warunku końca gry, którym jest zdobycie maksymalnej wartości punktów przez którego z graczy. Drugi proces odpowiada za ulepszenia podczas wystąpienia resetu lub pauzy w trakcie rozgrywki. Dla resetu licznik czasu stworzenia ulepszenia jest resetowany, a dla pauzy jest zmniejsza o jedną jednostkę. Ostatni proces zarządza obliczaniem momentu, w którym powinna zostać wyświetlona kolejna klatka prezentująca obraz gry. Działa on jedynie na podstawie sygnału zegarowego.

Moduł ten powinien reagować na zmianę wartości punktów graczy i kończyć grę w momencie osiągnięcia przez któregoś z nich maksymalnej liczby punktów:



GameLogic\_shouldEndGame

W przypadku resetu wartość licznika czasu do pojawienia się ulepszenia powinna zostać przywrócona do wartości domyślnej, a dla pauzy powinna zostać „zamrożona” wraz z licznikiem czasu wysyłania kolejnej klatki obrazu do ekranu podczas całego trwania pauzy:



GameLogic\_shouldResetGame

# Podsumowanie

Obecnie wszystkie moduły posiadają testy, na których zostało sprawdzone zachowanie poszczególnych komponentów. Odbyło się to za pomocą analizy wartości sygnałów podczas symulacji behawioralnej. Do implementacji pozostał jedynie moduł odpowiadający za wyświetlanie obrazu na ekranie przez VGA, a także dopasowanie niektórych stałych liczbowych w istniejących modułach w zależności od parametrów ekranu.