SPRAWOZDANIE Z LABORATORIUM		
Przedmiot **Modelowanie i analiza systemów**		Rok akademicki **2015/16**
Temat ćwiczenia **Dice Game**		Termin zajęć: **środa** **12:45–15:00**
Wydział **Wydział Informatyki**	Kierunek, specjalność **Informatyka, Mikrosystemy Informatyczne**	
Semestr **Semestr 1**	Skład grupy **Jakub Kasznia, Paulina Wróbel**	Data wykonania **31.05.2016r.**

1 Reguły gry

Celem ćwiczenia jest zaprojektowanie i przetestowanie w zaproponowanym testbench'u modelu gry Dice Game.

Wejściami do układu są sygnały:

- `Reset` – przycisk inicjujący grę,
- `Rb` – *Roll button* – przycisk symulujący wyrzut kostek.

Wyjściami z układu są sygnały:

- `Win` – dioda sygnalizująca wygraną,
- `Lose` – dioda sygnalizująca przegraną,
- wyświetlacz pokazujący wyrzucone liczby i ich sumę.

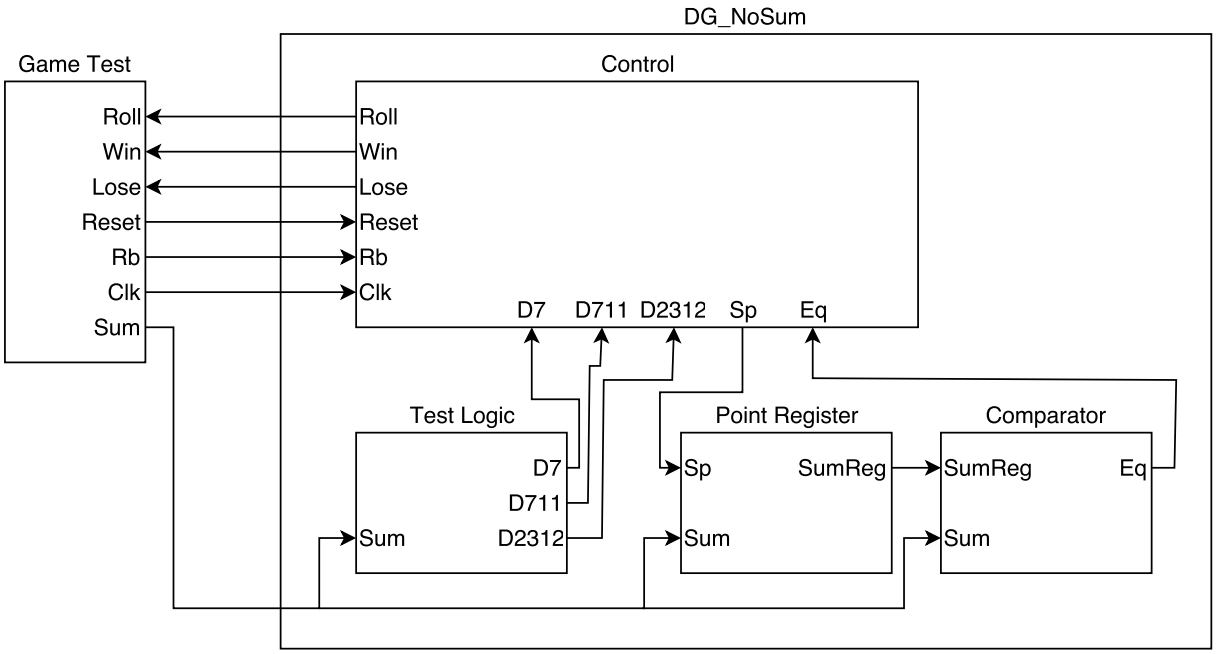
Symulacja wyrzutu kostek jest realizowana przez dwa liczniki o różnych częstotliwościach. Poniżej przedstawiono reguły gry. Wygnana i przegrana zależy od numeru iteracji:

- pierwszy wyrzut kostek:
  - wygrana – wylosowanie sumy 7 lub 11,
  - przegrana – wylosowanie sumy 2, 3 lub 12,
  - w innych przypadkach można losować jeszcze raz,
- drugi wyrzut kostek i kolejne:
  - wygrana – wylosowanie takiej samej sumy jak poprzednio,
  - przegrana – wylosowanie sumy 7,
  - w innych przypadkach można losować jeszcze raz.

2 Model bez liczników i sumatora

2.1 Struktura układu

Na rys. 1 przedstawiona jest struktura, na podstawie której napisano kod w VHDL.

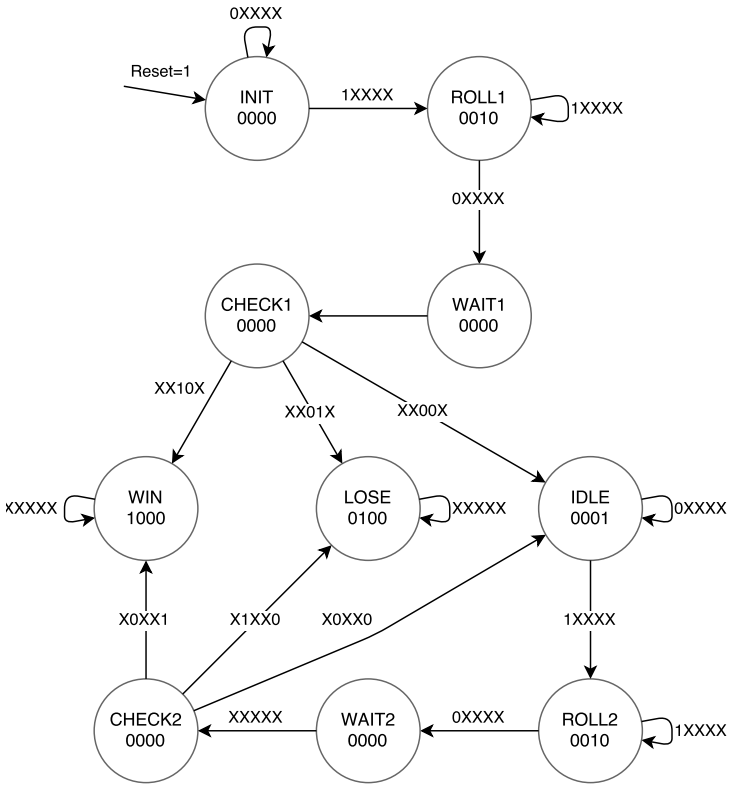


Rysunek 1: Struktura układu bez liczników i sumatora

Blok `DG_NoSum` jest modelem dice game bez liczników i sumatora. Składa się z następujących bloków:

- `Point Register` – rejestr do zapamiętywania sumy,
- `Comparator` – porównanie aktualnej sumy z poprzednią (jeżeli są równe, to sygnał `Eq` jest równy 1),
- `Test Logic` – ustawianie następujących sygnałów:
  - `D7` – równy 1, gdy suma jest równa 7,
  - `D711` – równy 1, gdy suma jest równa 7 lub 11,
  - `D2312` – równy 1, gdy suma jest równa 2, 3 lub 12,
- `Control` – blok sterujący (jego działanie jest opisane poniżej).

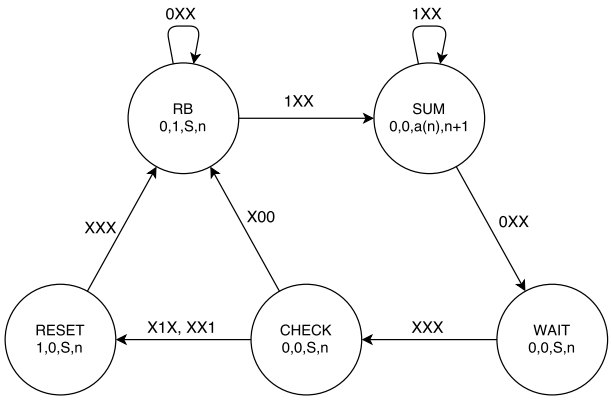
Diagram maszyny stanów bloku `Control` jest przedstawiony na rys. 2. Sygnał `Reset` równy 1 (wciśnięcie przycisku Reset) powoduje przejście maszyny do stanu `INIT`. Układ pozostaje w nim dopóki sygnał `Rb` nie będzie równy 1 (przycisk Roll Button wciśnięty) – wtedy przechodzi do stanu ROLL1. Pozostaje w tym stanie aż do puszczenie przycisku Roll Button, a wtedy przechodzi do stanu `WAIT1`. Stan `WAIT1` został dodany, aby przed sprawdzeniem wyników poczekać na ich poprawne wartości. W kolejnym cyklu układ przechodzi do stanu `CHECK1` i tam sprawdzane są sygnały z bloków `Test Logic` i `Comparator`. W zależności od ich wartości układ przechodzi do stanu `WIN`, `LOSE` albo `IDLE`. Stany `WIN` i `LOSE` kończą grę, aby rozegrać kolejną należy najpierw nacisnąć przycisk Reset. Natomiast stan `IDLE` pozwala na wylosowanie kolejnych liczb w ramach tej samej gry. Kolejne stany i przejścia są analogiczne. Przejście ze stanu `IDLE` do `ROLL2` jest takie samo jak ze stanu `INIT` do `ROLL1`. Stany `WAIT2` i `CHECK2` odpowiadają stanom `WAIT1` i `CHECK1`.



Rysunek 2: Diagram maszyny stanów bloku `Control`. Wejścia: `Rb`, `D7`, `D711`, `D2312`, `Eq` oraz `Reset`, wyjścia: `Win`, `Lose`, `Roll`, `Sp`.

2.2 Testbench i wyniki symulacji

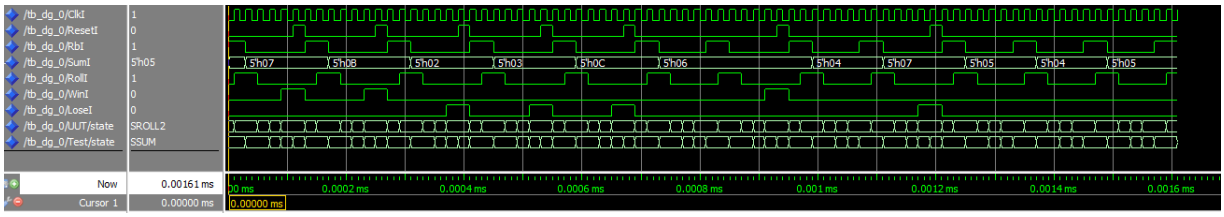
Do sprawdzenia poprawności działania został dodany blok `Game Test`. Diagram maszyny stanów tego bloku został przedstawiony na rys. 3.



Rysunek 3: Diagram maszyny stanów bloku `Game Test`. Wejścia: `Roll`, `Win`, `Lose`, wyjścia: `Reset`, `Rb`, `Sum`. `n`

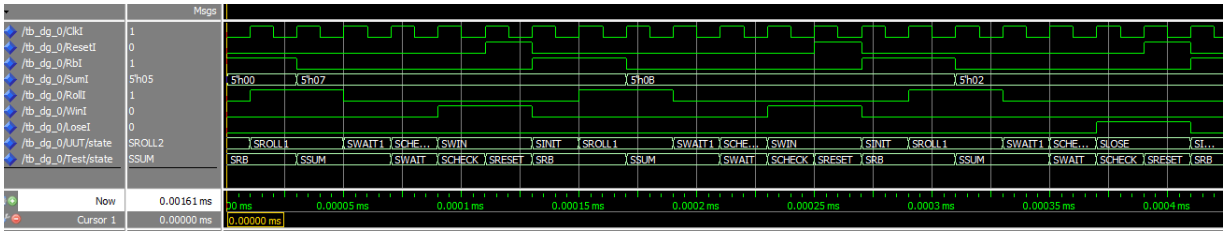
Na początku układ znajduje się w stanie `RB` (w tym stanie symulowane jest naciśnięcie przycisku Roll Button) i przechodzi do stanu `SUM` po otrzymaniu sygnału `Roll` równego 1. Stan `SUM` odpowiada za pobranie n–tego elementu z przygotowanej wcześniej tablicy i przypisanie $n = n + 1$. Po otrzymaniu sygnału `Roll` równego 0 układ przechodzi do stanu `WAIT`, gdzie czeka na otrzymanie poprawnych wyników z bloku `DG_NoSum`. Następnie przechodzi do stanu `CHECK`, a kolejne przejście zależy od sygnałów `WIN` i `LOSE`. Jeżeli którykolwiek z nich będzie równy 1, to układ przejdzie do stanu `RESET`, aby móc ustawić sygnał `Reset` na 1. W przeciwnym wypadku reset nie jest konieczny, więc układ przechodzi do stanu `RB`.

Poniżej przestawiono wyniki symulacji przeprowadzonej w programie *Modelsim*. Rys. 4 przedstawia widok na całość przebiegów, a rys. 5 przybliża kilka pierwszych przebiegów.



Rysunek 4: Symulacja układu bez liczników i sumatora – widok na całość

Widać, że sumy i odpowiadające im reakcje bloku `DG_NoSum` były zgodne z założeniami. W pierwszym cyklu po resecie i ustawieniu sumy na 7 (sytuacja: wygrana), więc układ ustawił sygnał `WinI` na 1. Kolejny cykl to reset i ustawienie sumy na 11. Również nastąpiła wygrana i ustawienie `WinI` na 1. Następnie były testowane przegrane. W kolejnych trzech cyklach sprawdzono 3 przypadki wartości sumy: 2, 3 i 12. Wtedy następowało rozpoznanie przegranej i ustawienie sygnału `LoseI` na 1. Następnie przetestowano wygraną poprzez wylosowanie 2 razy pod rząd tej samej sumy – zgodnie z oczekiwaniami otrzymano `WinI` równe 1. Kolejny cykl to suma równa 4, więc układ pozwolił na następne losowanie w ramach tej samej gry. Po wylosowaniu sumy równej 7 nastąpiła przegrana (sygnał `LoseI` równy 1). Ostatnie cykle pokazują, że jeżeli nie nastąpi ani przegrana, ani wygrana, to można wykonywać kolejne losowania w ramach tej samej gry.



Rysunek 5: Symulacja układu bez liczników i sumatora – widok na pierwszy cykl

Do sprawdzenia poprawności działania maszyn stanów blocków `DG_NoSum` i `Game Test` przybliżono przebiegi. Widać, że zachodzą one zgodnie z założeniami.

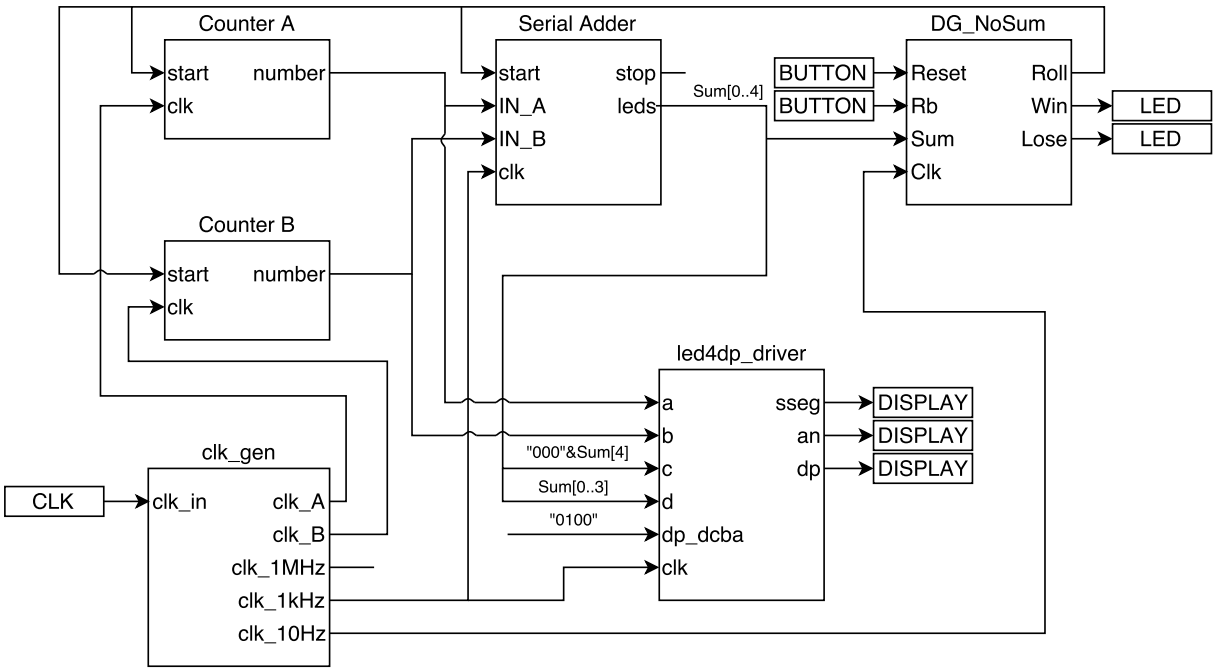
3 Model kompletny – z licznikami i sumatorem

3.1 Struktura układu

Na rys. 6 przedstawiona jest struktura, na podstawie której napisano kod w VHDL.
Jest to model dice game z licznikami i sumatorem. Składa się z następujących bloków:

- `DG_NoSum` – układ dice game bez liczników i sumatora,
- `Counter A` i `Counter B` – liczniki od 1 do 6, zmieniają wartość, gdy wciśnięty zostanie przycisc Roll Button,
- `Serial Adder` – sumator szeregowy przygotowany na poprzednich zajęciach,
- `clk_gen` – układ generujący sygnały o różnych częstotliwościach,
- `led4dp_driver` – układ umożliwiający przedstawienie wyników na wyświetlaczu.

Ponadto do sygnałów wejściowych i wyjściowych układy dodano atrybut `LOC`, aby móc zaimplementować układ na płycie Nexys. Z rys. 6 widać, że sygnał zegarowy zostanie podpięty pod



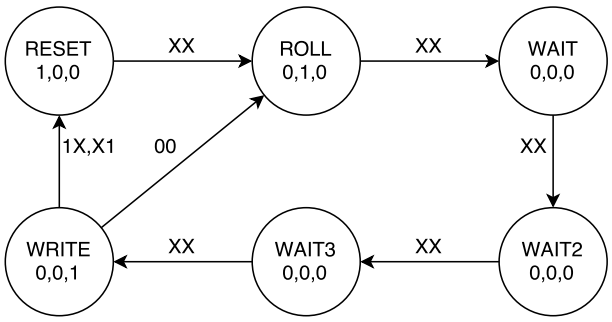
Rysunek 6: Struktura układu bez liczników i sumatora

`clk_in` (CLK), sygnały `Win` i `Lose` zostaną przedstawione na diodach (LED), sygnały `Reset` i `Rb` to przyciski (BUTTON), a `sseg`, `an` i `dp` umożliwią odpowietnie ustawienie wyświetlacza (DISPLAY).

Do zasymulowania losowości otrzymywanej sumy do bloków `Counter A` i `Counter B` podpięto sygnały zegarowe o różnych częstotliwościach. Inne bloki korzystają jeszcze z innych częstotliwośc, zatem niezbędne było napisanie bloku `clk_gen`. Ten blok z sygnału wejściowego o częstotliwości 50 MHz generuje sygnały o potrzebnych częstotliwościach.

3.2 Testbench i wyniki symulacji

Do sprawdzenia poprawności działania układu w pliku testbench została zaimpementowana maszyna stanów, której diagram został przedstawiony na rys. 7.

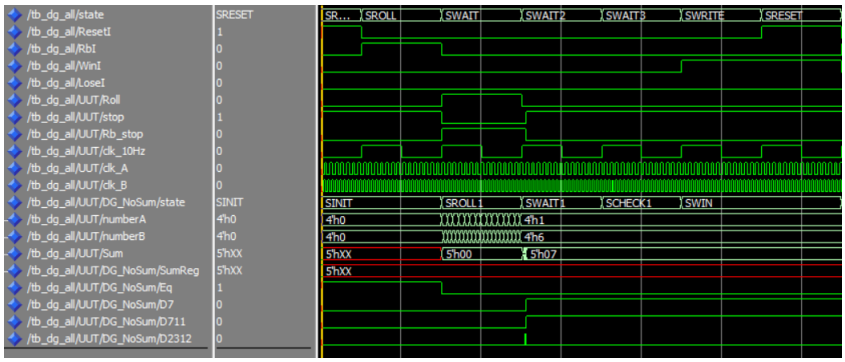


Rysunek 7: Diagram maszyny stanów testbenchu. Wejścia: `Win`, `Lose`, wyjścia: `Reset`, `Rb`, `write_enable`

Stanem początkowym jest stan `RESET`. Wtedy ustawienie sygnału `Reset` na 1 spowoduje rozpoczęcie nowej gry. Następnie układ przejdzie do stanu `ROLL`, gdzie zostanie ustawiony na 1 sygnał `Rb`. Następne 3 stany `WAIT`, `WAIT2` i `WAIT3` nic nie ustawiają, ale są niezbędne do poczekania na poprawne wyniki z układu dice game. Ze stanu `WAIT3` testbench przechodzi do stanu `WRITE`. Tam ustawiany jest sygnał `write_enable` inicjujący zapis do pliku `testbench_results.txt`. Następnie układ przechodzi do stanu `RESET` jeżeli wykryto przegraną lub wygraną. W przeciwnym

wypadku przechodzi do stanu `ROLL`.

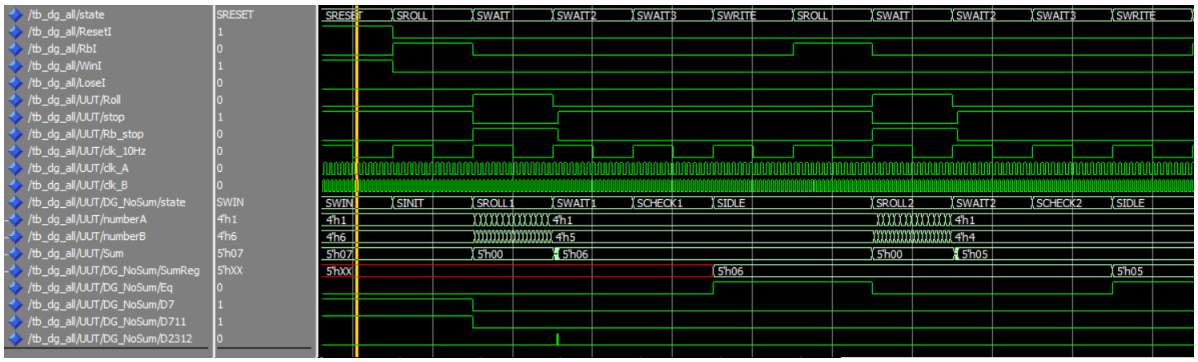
Poniżej przestawiono wyniki symulacji przeprowadzonej w programie *Modelsim*. Rys. 8–13 przedstawiają przebiegi otrzymane podczas symulacji.



Rysunek 8: Symulacja układu z licznikami i sumatorem – pierwsza gra

Na powyższym zrzucie ekranu (rys. 8) widać przebiegi dotyczące rozpoczęcia gry i pierwszego rzutu kośćmi. Rzut poprzedzony jest stanem wysokim na sygnale `ResetI` – układ jest w tym czasie resetowany, a maszyna stanów gry znajduje się w stanie `INIT`. Następnie wykonywany jest pierwszy rzut – sygnał `Roll` jest w stanie wysokim, sygnał `stop` przechodzi w stan niski(czyli nieaktywny), a maszyna stanów gry znajduje się w stanie `ROLL1`. Można zaobserwować, że w tym czasie z różną częstotliwością zmieniają się wartości sygnałów `numberA` i `numberB`, które zależą od przebiegów zegarów `clk_A` i `clk_C`. W ten sposób losowany jest wynik rzutu obu kości. Po puszczeniu przycisku Roll Button układ przechodzi w stan `WAIT1`, w którym wartości `numberA` i `numberB` są już ustalone i wynoszą odpowiednio 1 i 6. Suma, która przechowywana jest w rejestrze `Sum`, wynosi 7, więc sygnały `D7` oraz `D711` są w stanie wysokim. W stanie `CHECK1` następuje sprawdzenie rezultatu. Ze względu na to, że jest to pierwszy rzut w grze, wylosowanie sumy 7 oznacza wygraną i przejście do stanu `WIN`. Wówczas nie ma możliwości dalszego „rzucania" kośćmi – trzeba wykonać reset, aby rozpocząć grę od nowa. Warto tu również zwrócić uwagę na stany w testbenchu. Jak widać wyprzedzają one stany układu odpowiadającego za grę. Wcześniej omówiony stan `RESET` odpowiada za wyzerowanie układu, `ROLL za rzucenie kośćmi`, `WAIT`, `WAIT2`, `WAIT3` odpowiadają za oczekiwanie na wynik, a `WRITE` za zapis wyniku do pliku tekstowego.

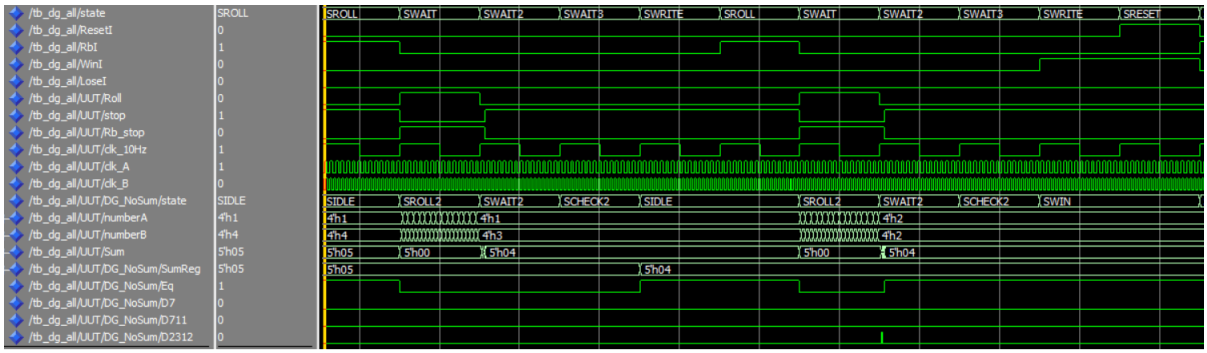
Ze względu na fakt, iż powyżej zostały omówione wszystkie najważniejsze elementy testbenchu, tj. stany i omówienie znaczenia wartości poszczególnych sygnałów, dla kolejnych gier zostaną omówione tylko najistotniejsze różnice.



Rysunek 9: Symulacja układu z licznikami i sumatorem – druga gra

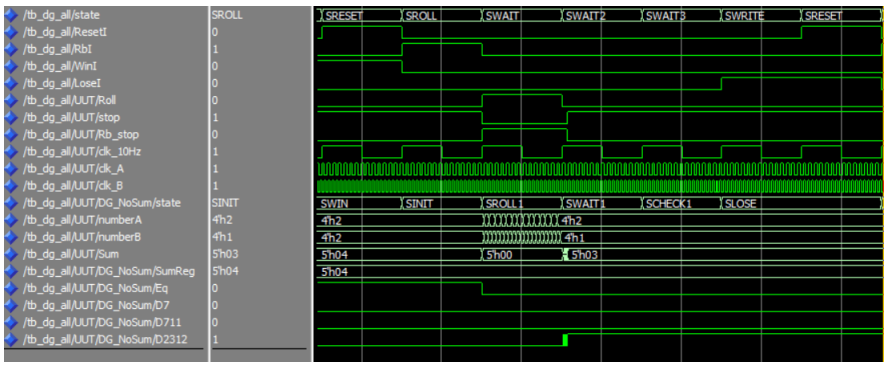
Zamieszczony kolejny zrzut ekranu (rys. 9) prezentuje rozpoczęcie drugiej gry i wykonanie dwóch rzutów kośćmi. Po pierwszym rzucie otrzymano wyniki dla `numberA` oraz `numberB` odpowiednio 1 i 5, co w sumie daje 6. Oznacza to konieczność wykonania kolejnego rzutu. W

kolejnym rzucie otrzymujemy wartości 1 i 4, co w sumie daje 5 i również nie pozwala stwierdzić, czy rozgrywka zakończyła się zwycięstwem czy porażką.



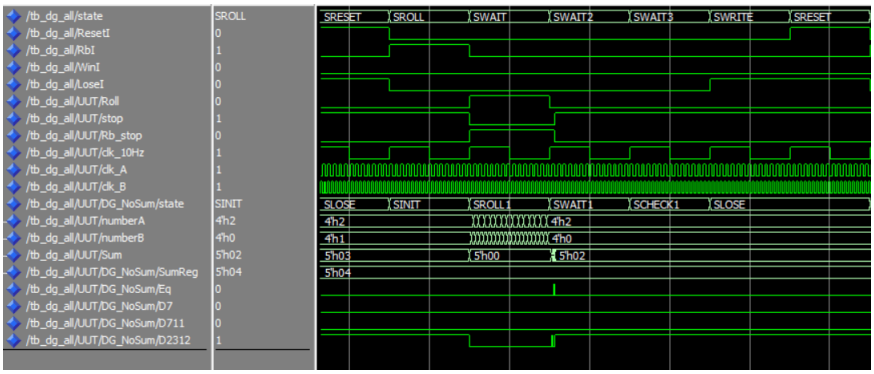
Rysunek 10: Symulacja układu z licznikami i sumatorem – druga gra, kolejne rzuty

Wykonywany jest zatem kolejny rzut (rys. 10). On również nie pozwala na określenie statusu gry, bowiem wylosowane wartości to 1 i 3, w sumie 4. Dopiero ostatnia kolejka przynosi oczekiwany rezultat, ponieważ wyrzucone zostają wartości 2 i 2, w sumie 4 – bieżąca suma jest równa poprzedniej, oznacza to zatem wygraną.



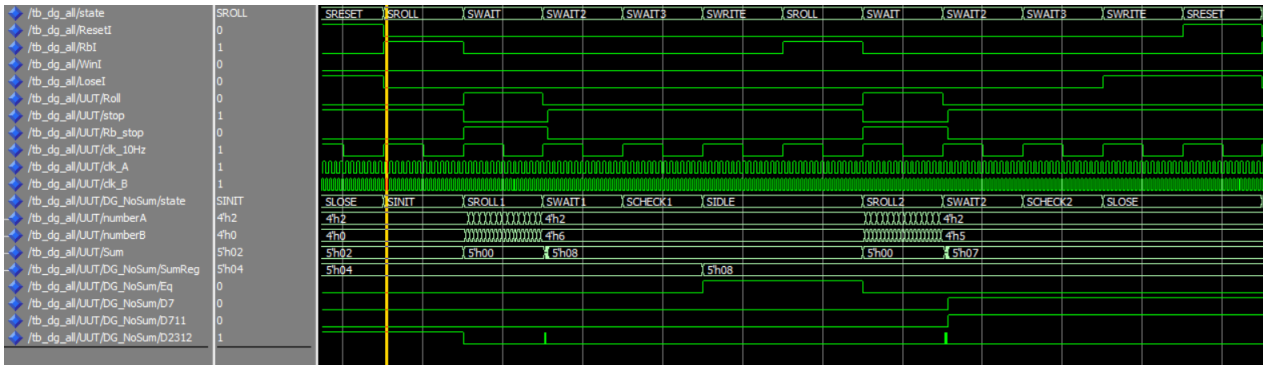
Rysunek 11: Symulacja układu z licznikami i sumatorem – trzecia gra

Zamieszczony następny zrzut (rys. 11) obrazuje trzecią grę wykonaną w ramach testbenchu. W tym przypadku w pierwszym rzucie wypadają wartości 2 i 1, czyli w sumie 3, co oznacza przegraną.



Rysunek 12: Symulacja układu z licznikami i sumatorem – czwarta gra

W czwartej grze (rys. 12), podobnie jak w trzeciej, również wynikiem jest przegrana. W pierwszym rzucie wypadają wartości 2 i 0, czyli łącznie 2. Taka suma po pierwszym rzucie oznacza przegraną, zgodnie z regułami gry.



Rysunek 13: Symulacja układu z licznikami i sumatorem – piąta gra

Piąta gra (rys. 13) także przedstawia przypadek przegranej, ale wynikającej z faktu, iż w drugim rzucie suma wylosowanych wartości była równa 7.

Wyniki symulacji zostają także zapisane do pliku tekstowego. Zawartość takiego pliku została przedstawiony w listingu 1. Struktura pliku składa się z następujących kolumn:

- `Time` – czas, w którym nastąpił zapis do pliku,
- `A` – wartość wyjścia z pierwszego licznika,
- `B` – wartość wyjścia z drugiego licznika,
- `SUM` – wartość wyjścia z sumatora,
- `Win` – wartość sygnału `WinI`,
- `Lose` – wartość sygnały `LoseI`,
- `Game` – opis mówiący, czy dana gra jest rozpoczynana (`NEW`), czy następuje kolejne losowanie w rammach tej samej gry (wtedy w kolumnie jest puste pole).

Listing 1: Zawartość pliku `testbench_results.txt` – wyniki symulacji

```
1          Time | A B SUM |  Win Lose| Game
2     450500520 ns | 1 6   7 |    1    0|
3    1050500520 ns | 1 5   6 |    0    0| NEW
4    1550500520 ns | 1 4   5 |    0    0|
5    2050500520 ns | 1 3   4 |    0    0|
6    2550500520 ns | 2 2   4 |    1    0|
7    3150500520 ns | 2 1   3 |    0    1| NEW
8    3750500520 ns | 2 0   2 |    0    1| NEW
9    4350500520 ns | 2 6   8 |    0    0| NEW
10   4850500520 ns | 2 5   7 |    0    1|
11   5450500520 ns | 2 4   6 |    0    0| NEW
```

Z listingu widać, że nastąpiło 10 rzutów kośćmi. Widać również, że pierwsze losowanie zakończyło się wygraną, więc kolejne losowanie jest rozpoczęciem nowej gry. Ta gra również kończy się wygraną – wygrana poprzez wylosowanie dwóch takich samych sum pod rząd. Następna nowa gra jest zapisana w linii 7 i kończy się ona przegraną (wylosowanie sumy 3). Podobna sytuacja jest w linii 8 – przegrana poprzez wylosowanie sumy 2. Kolejne 2 linie przedstawiają kolejną grę – widać, że została ona przegrana przez otrzymanie sumy równej 7 przy drugim rzucie. Ostatni rzut nie zakończył się ani porażką, ani wygraną, zatem kolejne losowanie nastąpiłoby w ramach tej samej gry.

3.3 Nexys

Przypisano odpowiednie piny płytki Nexys do sygnałów:

- `Clk` – B8,
- `Reset` – B18,
- `Rb` – D18,
- `Win` – J14,
- `Lose` – J15,
- `sseg` – H14, J17, G14, D16, D17, F18, L18,
- `an` – F17, H17, C18, F15,
- `dp` – C17.

W środowisku *ISE Xilinx* stworzono projekt, dokonano syntezy i wygenerowano plik `.bit`. Żadne błędy nie zostały wykryte podczas tych procesów. Po uruchomieniu programu na płytce Nexys wykonano testy działania. Nie wykryto zachowań niezgodnych z założeniami.

4 Wnioski

Zrealizowany projekt pozwolił studentom na utrwalenie swoich umiejętności z implementacji maszyn stanu w VHDL. Ponadto zapoznano się z tworzeniem projektu syntezowalnego, aby móc wgrać go na płytkę Nexys. Symulacja w programie *Modelsim* i działanie układu na płycie Nexys przebiegło zgodnie z założeniami projektu.