Projekt indywidualny – wariant 22

Grzegorz Fabisiak 325000

Specyfikacja jednostki głównej:

Układ składa się z czterech modułów, wykonujących osobne operacje. Wszystkie operacje odbywają się w kodzie U2. Wyboru operacji dokonujemy poprzez zadanie konkretnego sygnału na N-bitowe wejście **i_oper.** W naszym przypadku wejście to ma 2 bity i wybiera spośród 4 operacji.

00: Negacja

Zmienia znak danej wartości na wejściu A w U2.

Lista wejść i wyjść:

i_argA – m-bitowe wejście pierwszego argumentu

o_result – m-bitowe wyjście synchroniczne

error – wyjście sygnalizujące niepoprawne działanie

01: Porównanie argumentów

Wyświetla na wyjściu jedynkę jeśli A jest większe bądź równe B lub na każdym bicie zera jeśli warunek nie jest spełniony.

Lista wejść i wyjść:

i_argA - m-bitowe wejście pierwszego argumentu

i_argB – m-bitowe wejście drugiego argumentu

o_result – m-bitowe wyjście synchroniczne

10: Ustawienie bitu

Operacja ustawia B-bit argumentu A na 1

Lista wejść i wyjść:

i_argA – m-bitowe wejście pierwszego argumentu

i_argB – m-bitowe wejście drugiego argumentu

o_result – m-bitowe wyjście synchroniczne

error – wyjście sygnalizujące niespełnienie warunków działania

11: Zmiana kodu

Operacja zmienia format kodu liczby argumentu A z U2 na znak moduł.

Lista wejść i wyjść:

i_argA – m-bitowe wejście pierwszego argumentu

o_result – m-bitowe wyjście synchroniczne

error – wyjście sygnalizujące niepoprawne działanie

Jednostka główna:

Lista wejść, wyjść oraz sygnałów wewnętrznych:

i_argA – m-bitowe wejście pierwszego argumentu

i_argB – m-bitowe wejście drugiego argumentu

i_oper – n-bitowe wejście wyboru operacji

i_clk – zbocze zegara z aktywnym zboczem narastającym

i_rsn – wejście resetu synchronicznego (przy wartości 0 ustawia rejestry na 0)

o_result – m-bitowe wyjście synchroniczne będące wynikiem danej operacji

o_status – 4 bitowe wyjście synchroniczne sygnalizujące o danych następstwach: bit 3 – ERROR, bit 2 – OEVRFLOW, bit 1 – ODD, bit 0 – ZERO

error_0, error_2, error_3 – odbierają sygnały z errorów wewnątrz modułów

Sygnały wewnętrzne error oraz overflow – odbierają sygnały z errorów wewnętrznych jednostki głównej i dzielą według przyczyn na wyjście statusu

Wszystkie moduły są zawarte w katalogu **Modules** w repozytorium gitlab pod nazwami kolejno tak jak wyżej:

- negation.sv

- comparison.sv

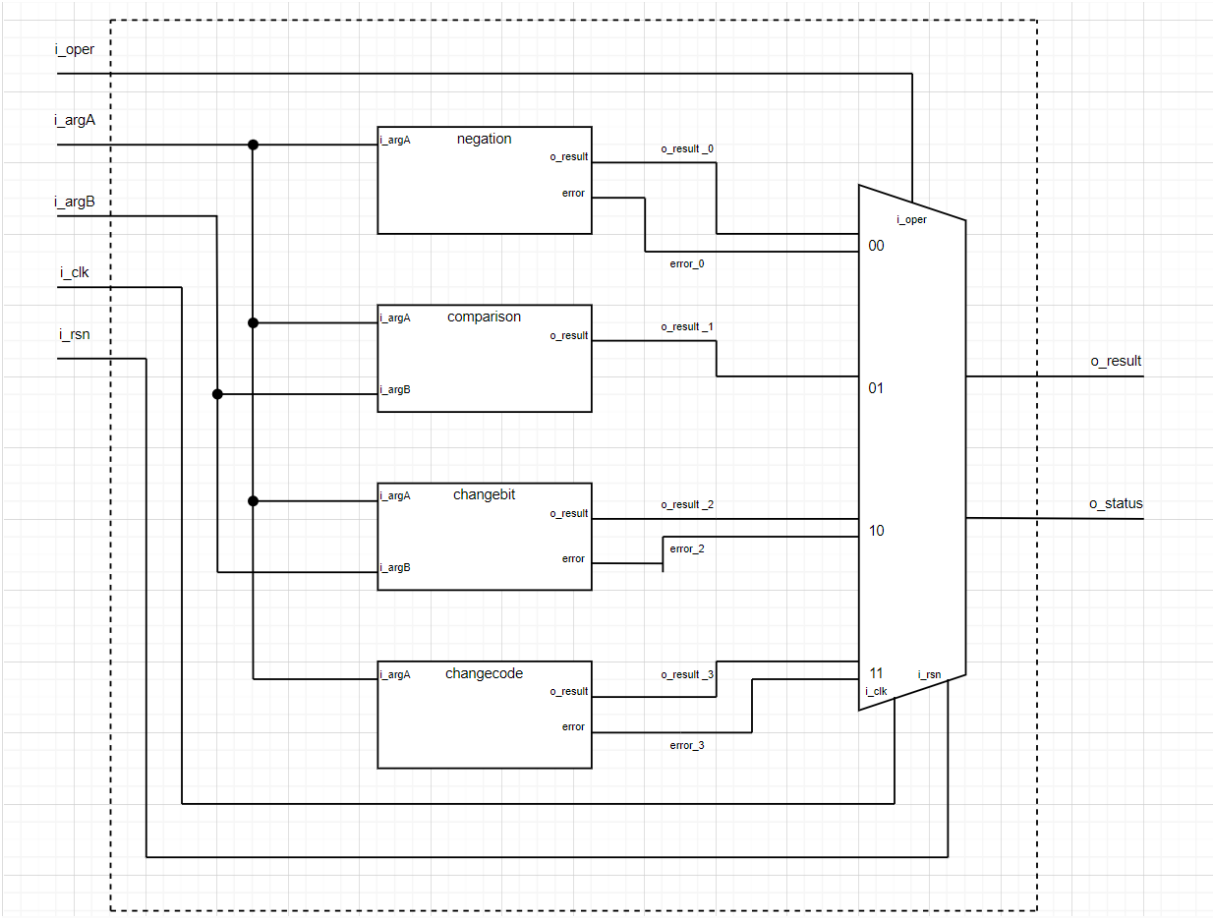
- changebit.sv

- changecode.sv

Oraz jednostka główna łącząca wszystkie poprzednie: **exe_unit_w22.sv**

W katalogu **Synthesis** znajdziemy wszystkie te moduły po syntezie, zaś w katalogu **Testbench** wszystkie pliki testbench sterujące sygnałami do gtkwave.

Schemat blokowy struktury jednostki:



Wyniki syntez oraz symulacji przed i po syntezie dla każdego modułu:

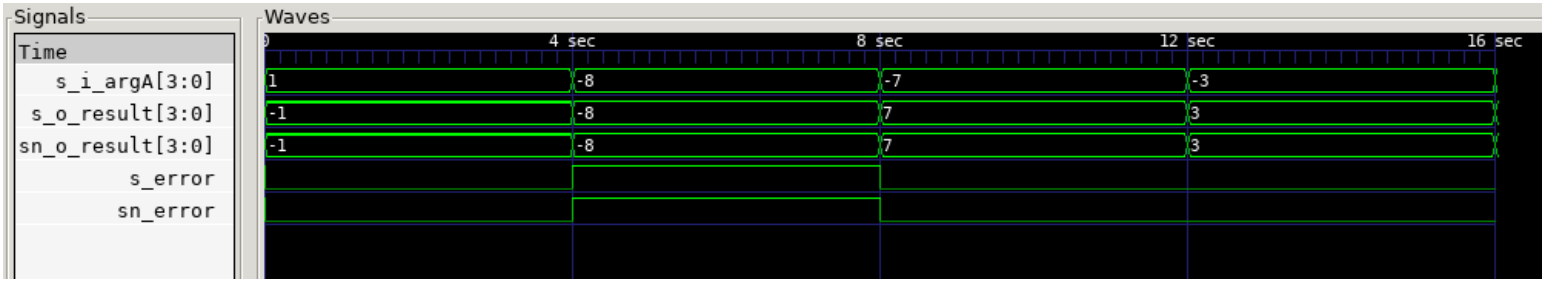
negation.sv

```
=== negation_rtl ===

   Number of wires:                 10
   Number of wire bits:             16
   Number of public wires:           3
   Number of public wire bits:       9
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 12
     $_ANDNOT_                       4
     $_NOT_                          1
     $_OAI3_                         2
     $_ORNOT_                        1
     $_OR_                           2
     $_XOR_                          2
```

Dla lepszego podglądu, ustawiłem format wyświetlanych sygnałów na signed decimal.

Jak widzimy error pojawia się dla zadania na wejście -8, ponieważ nie da się zapisać liczby 8 w czterech bitach.

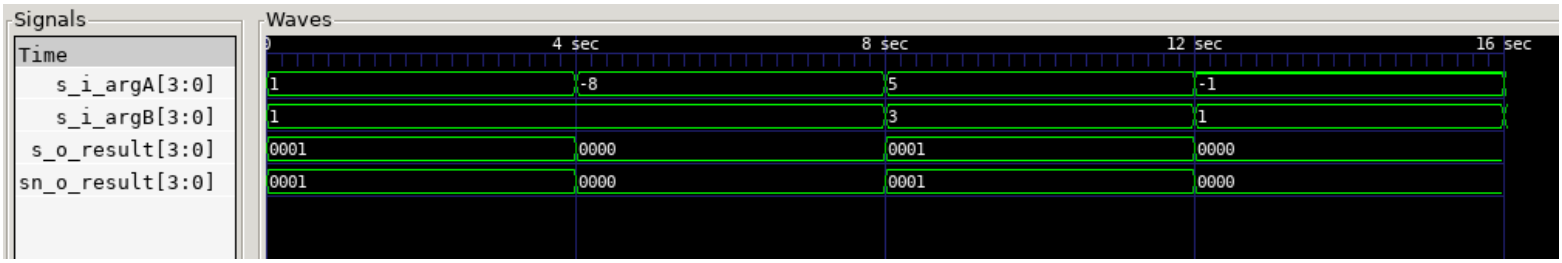


comparison.sv

```
=== comparison_rtl ===

   Number of wires:                 18
   Number of wire bits:             27
   Number of public wires:           3
   Number of public wire bits:      12
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 16
     $_ANDNOT_                       2
     $_AOI3_                         1
     $_NOR_                          1
     $_OAI3_                         2
     $_ORNOT_                        3
     $_OR_                           2
     $_XOR_                          5
```

Tutaj również użyłem poglądowo formatu signed decimal aby można było łatwiej porównać liczby. W tym przypadku nie ma wyjścia dla błędu, ponieważ liczba jest mniejsza, równa, bądź większa. Nie ma żadnego warunku, którego niespełnienie nie pozwoliłoby wykonać operacji.

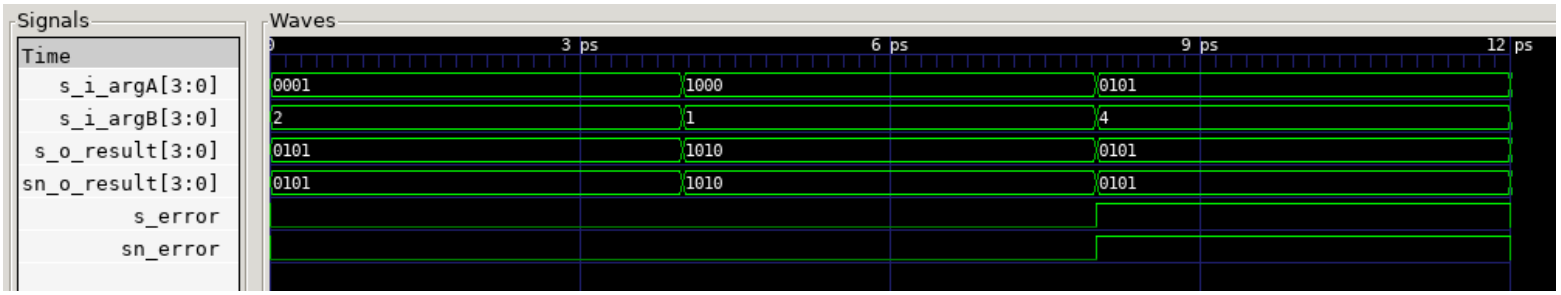


changebit.sv

```
=== changebit_rtl ===

   Number of wires:                 19
   Number of wire bits:             28
   Number of public wires:           4
   Number of public wire bits:      13
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 20
     $_NOT_                          5
     $_OAI3_                         4
     $_ORNOT_                        1
     $_OR_                          10
```

Jak widzimy, błąd pojawia się gdy chcemy ustawić czwarty bit na 1. Oczywiście bity indeksuje się od 0, dlatego pojawia się błąd, ponieważ najbardziej znaczący ma indeks 3.

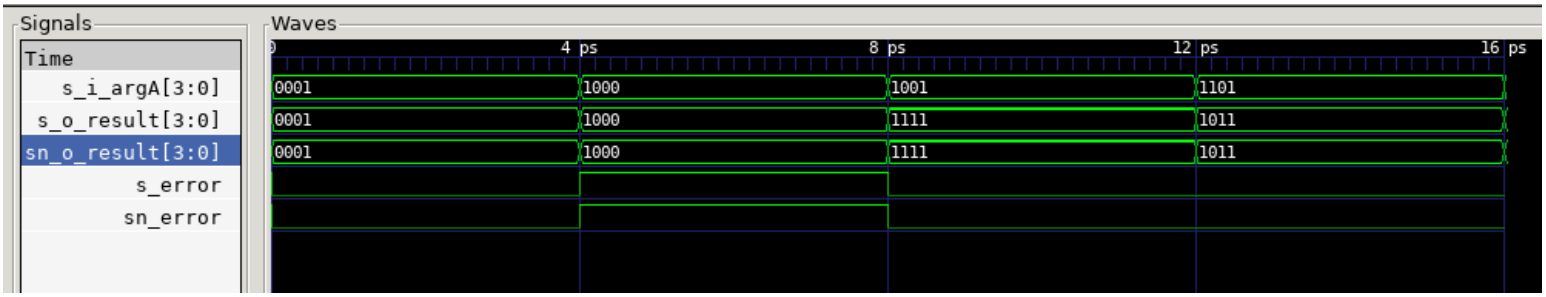


changecode.sv

```
=== changecode_rtl ===

   Number of wires:                 15
   Number of wire bits:             21
   Number of public wires:           3
   Number of public wire bits:       9
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                 17
     $_ANDNOT_                       4
     $_AND_                          1
     $_MUX_                          3
     $_NOR_                          1
     $_NOT_                          1
     $_OAI3_                         1
     $_ORNOT_                        1
     $_OR_                           3
     $_XNOR_                         1
     $_XOR_                          1
```

Błąd pojawia się podczas ustawienia na wejście 1000, czyli w U2 liczby -8. W ZM nie ma takiej wartości w zapisie 4 bitowym, dlatego operacja nie może zostać wykonana.



ALU.sv

```
=== design hierarchy ===

   ALU_rtl                          1
     $paramod\changebit\BITS=4      1
     $paramod\changecode\BITS=4     1
     $paramod\comparison\BITS=4     1
     $paramod\negation\BITS=4       1

   Number of wires:               109
   Number of wire bits:           170
   Number of public wires:         27
   Number of public wire bits:     82
   Number of memories:              0
   Number of memory bits:           0
   Number of processes:             0
   Number of cells:               111
     $_ANDNOT_                     18
     $_AND_                         2
     $_AOI3_                        1
     $_AOI4_                        3
     $_DFF_PN0_                     8
     $_MUX_                         9
     $_NAND_                        3
     $_NOR_                         3
     $_NOT_                        11
     $_OAI3_                        9
     $_OAI4_                        1
     $_ORNOT_                      11
     $_OR_                         19
     $_XNOR_                        3
     $_XOR_                        10
```

Widzimy tu syntezę oraz wynik symulacji dla jednostki głównej wykonującej wszystkie operacje. Możemy również tutaj powiedzieć, w jakich przypadkach jaka flaga błędu jest uruchamiana.

ERROR – trzeci bit statusu

Dobrym przykładem pojawienia się flagi error w statusie jest pierwszy sygnał dla modułu zmiany bitu. Próbujemy ustawić w tym miejscu bit argumentem B, który jest ujemny i wskazuje bit o ujemnym indeksie. Jest to oczywisty błąd.

OVERFLOW – drugi bit statusu

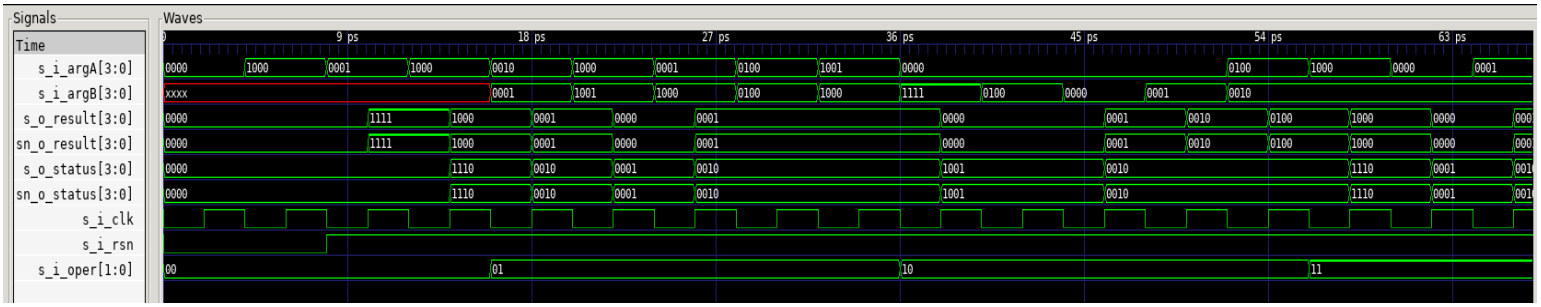
Zastosowanie tej flagi widać, gdy chcemy wykonać negację dla wartości 1000 czyli -8. W kodzie U2 na 4 bitach nie da się wypisać cyfry 8, ale na 5 bitach już się da, dlatego mamy do czynienia z przepełnieniem.

ODD – pierwszy bit statusu

Widzimy go większość czasu, w porównaniu liczb. Prawidłowo wykonana operacja wysyła na wyjście sygnał 0001, który posiada oczywistą nieparzystą liczbę jedynek.

ZERO – zerowy bit statusu

Wynikiem zmiany 0000 w kodzie U2, jest 0000 w kodzie ZM. Jest to przykład powstania warunku tej flagi



Repozytorium:

W repozytorium oprócz modułów, syntez i testbenchy znajduje się także mniej ważny folder z resztą plików symulacyjnych pod nazwą **Work**.