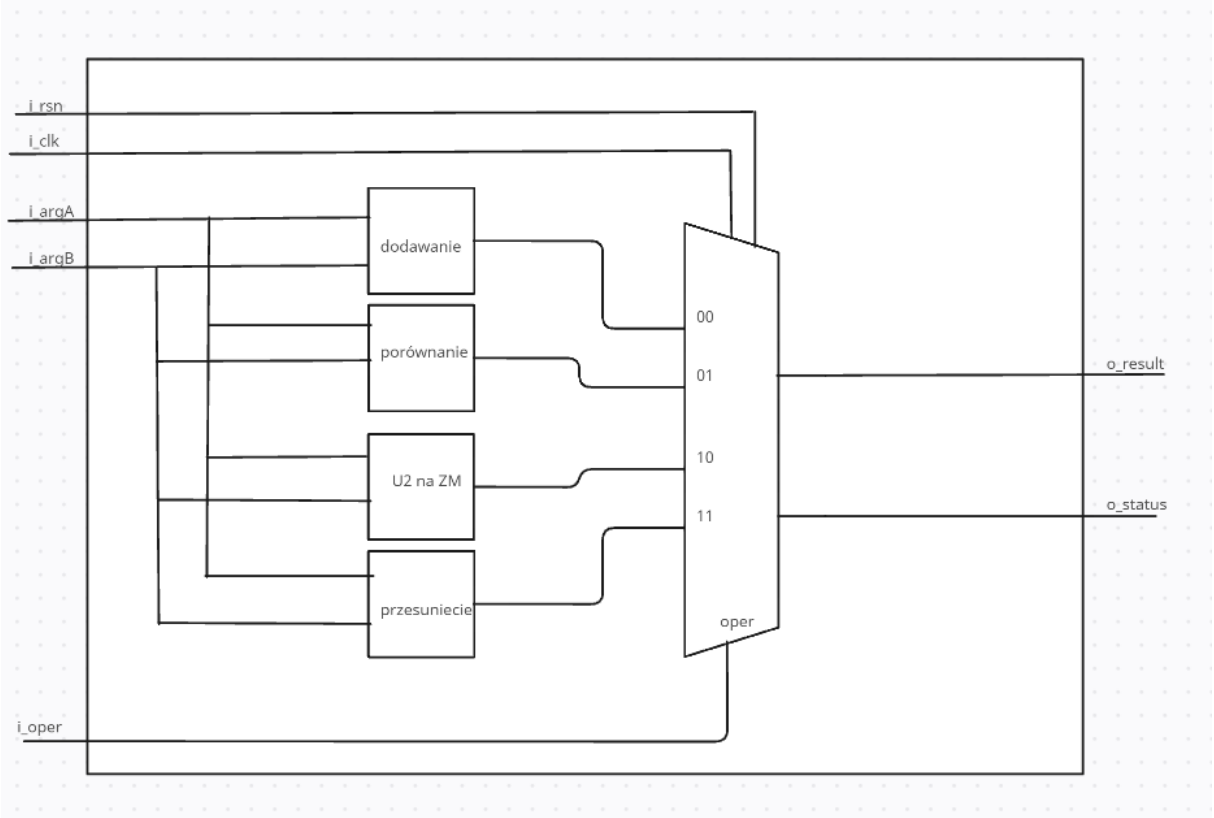
SCK		
Nazwa projektu:	Exe_unit_w12	
Wykonał	Przemysław Ziencik	Grupa dziekańska: E3
Data:	15.12.2023	
Nr. indeksu	325104	

- Schemat blokowy układu



Rysunek nr 1 – Schemat blokowy układu

- Pliki potrzebne do działania ALU

Nazwa pliku:	**Opis pliku**	**Moduły**
exe_unit_w12.sv	Plik w którym znajduje się ALU	exe_unit_w12
exe_unit_w12_tb.sv	Plik weryfikacji	exe_unit_w12_tb
run.ys	Plik służący do syntezy logicznej	----
exe_unit_w12_rtl.sv	wynik syntezy logicznej exe_unit_w12	exe_unit_w12_rtl
makefile	Skrypt symulacji	----
exe_unit_w12.vcd	Wynikowy plik symulacji	----
exe_unit_w12.iveri.log	Wynik syntezy iverilog	----
synth.log	Wynik syntezy logicznej	----
exe_unit_w12.iveri.run	plik skompilowany	----

Tabela nr 1 – Pliki

- Wejścia i wyjścia

Nazwa	Liczba bitów	Funkcja
i_argA	4	Argument A
i_argB	4	Argument B
i_oper	2	Wybór operacji
i_rsn	1	Zbocze narastające
i_clk	1	Cykl zegara

Tabela nr 2 – wejścia

Nazwa	Liczba bitów	Funkcja
o_result	4	wynik
o_status	4	Wyjście stanów
s_result	4	Wyjście – wynik-wewnętrzne
s_status	4	Wyjście – stanów-wewnętrzne

Tabela nr 3 – wyjścia

- Parametry zawarte w programie

Nazwa	Liczba bitów
M	4
N	2

Tabela nr 4 – parametry

- Zmienne zawarte w programie

Nazwa	Liczba bitów	Funkcja
k	-	integer
p	-	integer
b	4	zmienna
P	4	zmienna

Tabela nr 5 – zmienne

- Funkcje i operacje zawarte w programie

Funkcja:	i_oper (s_oper w tb)
dodawanie	00
porównanie	01
U2 na ZM	10
Zamiana bitu na 1	11

Tabela nr 6 –funkcje i operacje

- Flagi zawarte w programie

Flaga:	Bit błędu	Przykład dla wartości 1		Przykład dla wartości 0
ERROR	**0**	**1111111111(Zmiana bitu na 1)**		**1111 (Zmiana bitu na 1)**
ZERO	**1**	**0000**		**1111**
NEG	**2**	**1001(ZM)**	**1000(ZM)**	**0010(ZM)**
EVEN	**3**	**1010**	**1111**	**0010**

Tabela nr 7 – Flagi

Powyższe flagi są odpowiednimi bitami wyjścia o_status.

Bit 0:ERROR – operacja nie została wykonana wartość o_result jest nie określona.

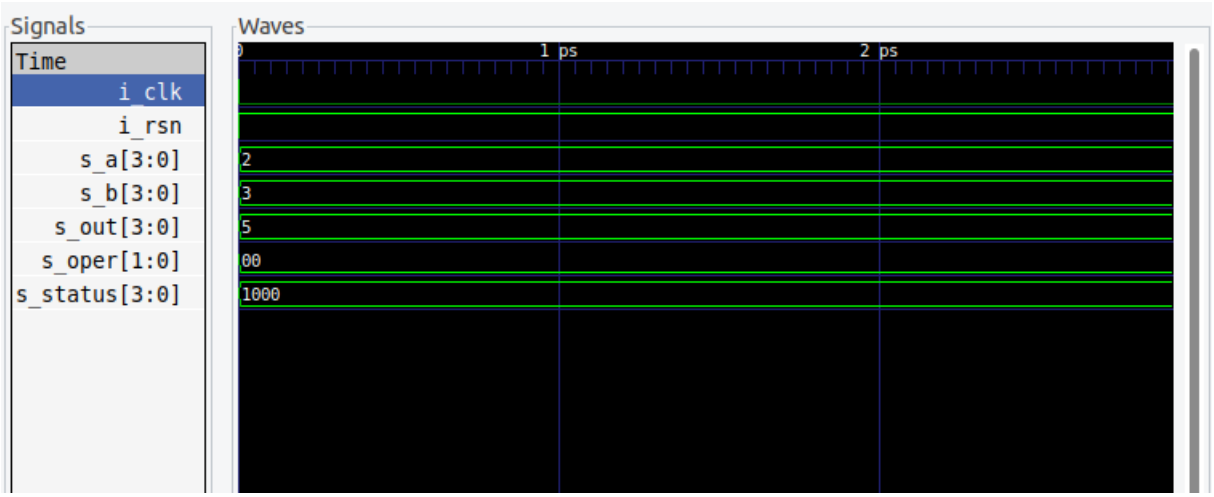
Bit 1: ZERO – wszystkie bity wyniku są ustawione na 0.

Bit 2: NEG – wynik jest liczbą ujemną.

Bit 3: EVEN – w wyniku jest parzysta liczba jedynek.

- Przykładowe przebiegi

Przykładowe Przebiegi



Zdjęcie nr 1 – Operacja dodawanie

Dla zdjęcia powyżej:

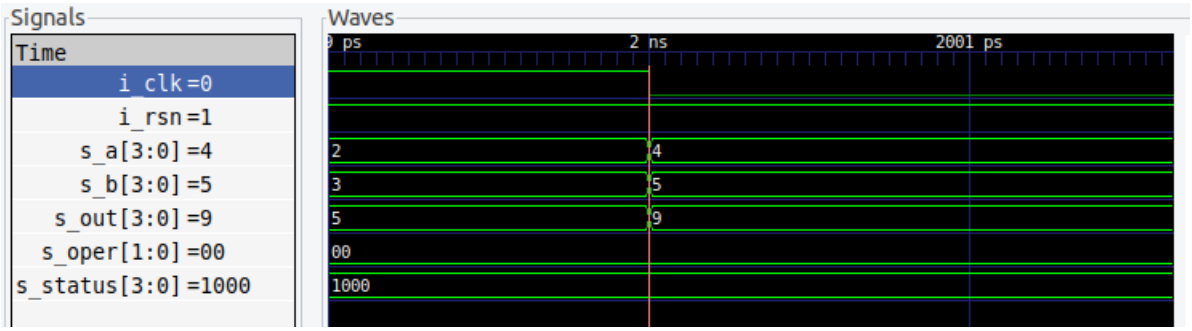
i_oper = 00 (binary)

i_argA = 2 (decimal)

i_argB = 3 (decimal)

o_result = 5 (decimal)

o_status = 1000 (binary), bit 3 = 1: w wyniku jest parzysta liczba jedynek



Zdjęcie nr 2 – Operacja dodawanie

Dla zdjęcia powyżej:

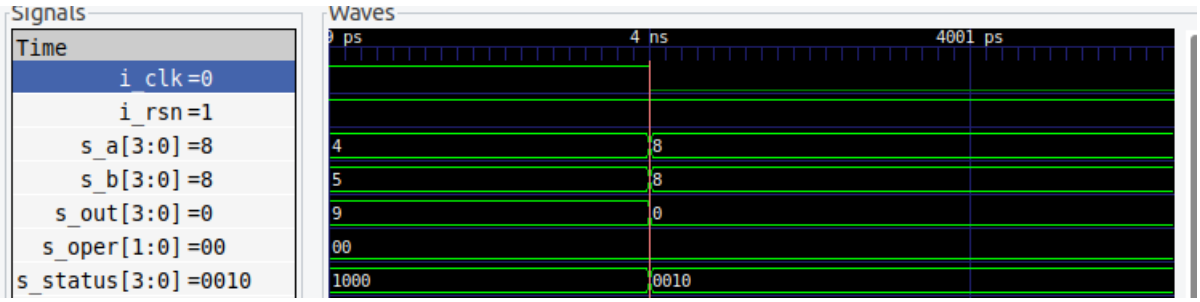
i_oper = 00 (binary)

i_argA = 4 (decimal)

i_argB = 5 (decimal)

o_result = 9 (decimal)

o_status = 1000 (binary), bit 3 = 1: w wyniku jest parzysta liczba jedynek



Zdjęcie nr 3 – Operacja dodawanie

Dla zdjęcia powyżej:

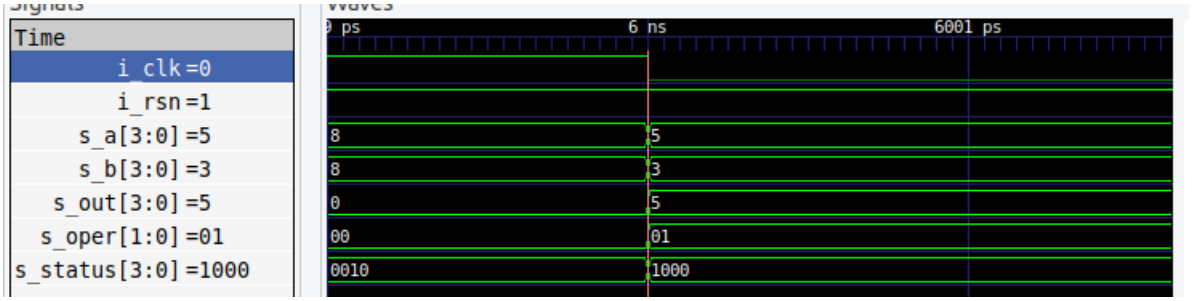
i_oper = 00 (binary)

i_argA = 8 (decimal)

i_argB = 8 (decimal)

o_result = 0 (decimal), dla 4 bitów maksymalną liczbą jest 15, naszym wynikiem jest 16 więc wyjście przyjmuje wartość 0.

o_status = 0010 (binary), bit 1 jest równy 1 ponieważ wszystkie bity w wyniku są 0, 4’b0000.



Zdjęcie nr 4 – Operacja porównanie

Dla zdjęcia powyżej:

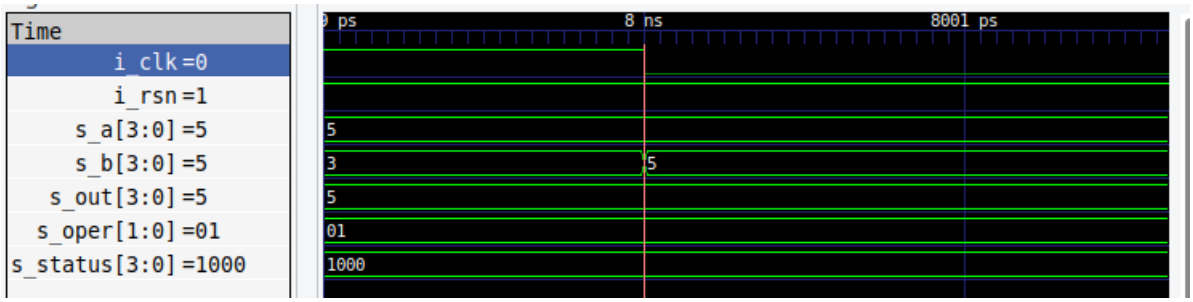
i_oper = 01 (binary)

i_argA = 5 (decimal)

i_argB = 3 (decimal)

o_result = 5 (decimal), liczbą większą jest 5 (i_argA)

o_status = 1000 (binary), bit 3 = 1: w wyniku jest parzysta liczba jedynek



Zdjęcie nr 5 – Operacja porównanie

Dla zdjęcia powyżej:

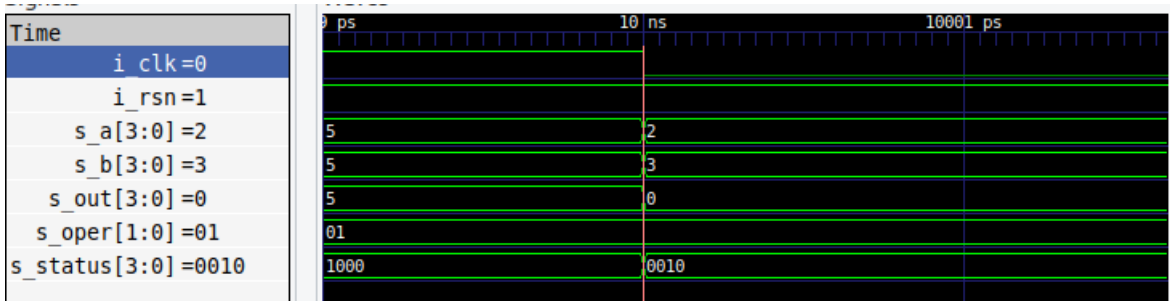
i_oper = 01 (binary)

i_argA = 5 (decimal)

i_argB = 5 (decimal)

o_result = 5 (decimal), liczbą dla i_argA i i_argB są równie sobie.

o_status = 1000 (binary), bit 3 = 1: w wyniku jest parzysta liczba jedynek.



Zdjęcie nr 6 – Operacja porównanie

Dla zdjęcia powyżej:

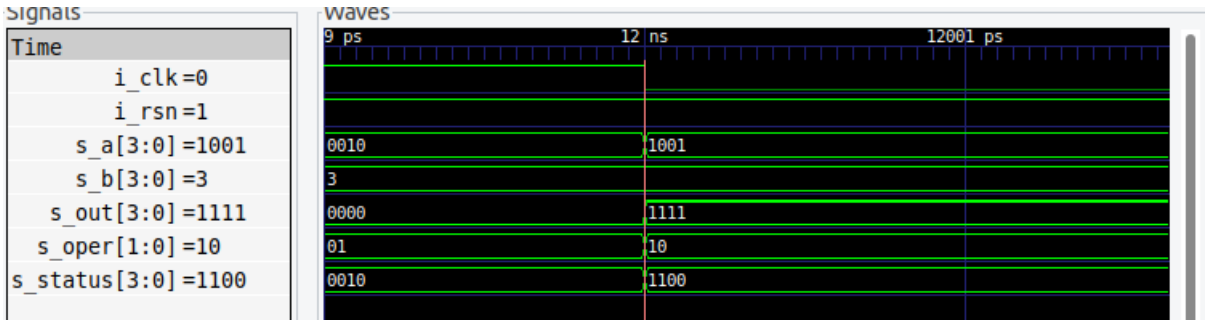
i_oper = 01 (binary)

i_argA = 2 (decimal)

i_argB = 3 (decimal)

o_result = 0 (decimal), wynikiem jest 0 ponieważ warunek nie został spełniony.

o_status = 0010 (binary), bit 1 = 1: wszystkie bity na wyjściu są 0, 4'b0000.



Zdjęcie nr 7 – U2 na ZM

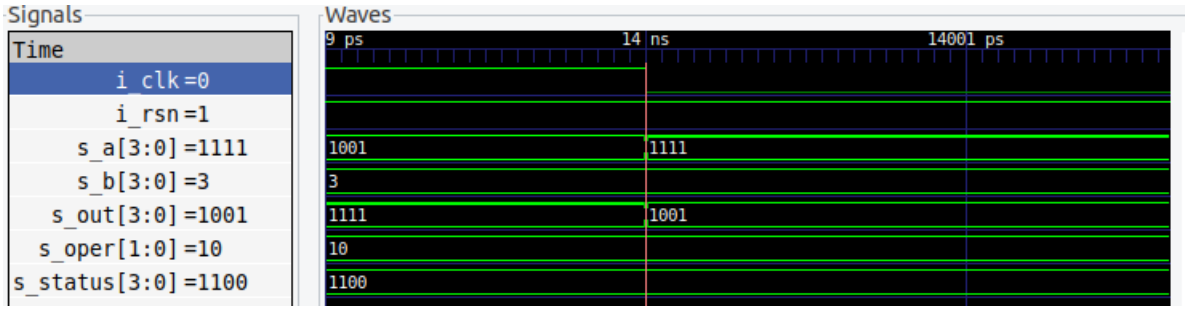
Dla zdjęcia powyżej:

i_oper = 10 (binary)

i_argA = 1001 (binary)

o_result = 1111 (decimal)

o_status = 1100 (binary), bit 3 = 1 i bit 2 = 1: w wyniku jest parzysta liczba jedynek, wynik jest liczbą ujemną



Zdjęcie nr 8 – U2 na ZM

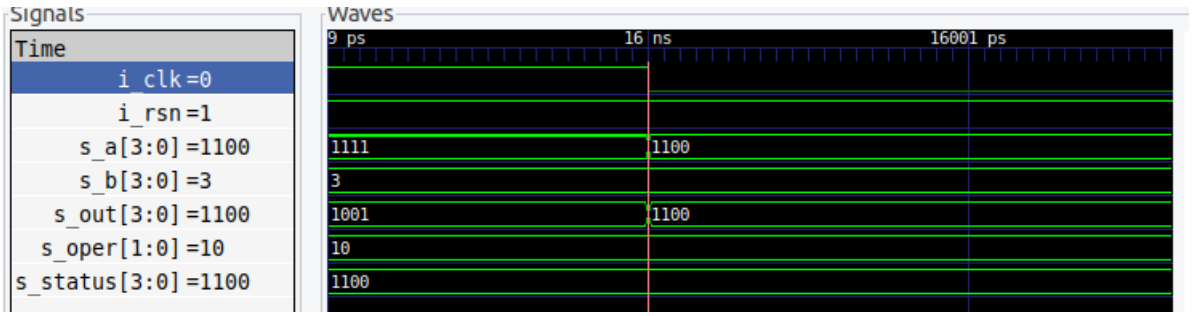
Dla zdjęcia powyżej:

i_oper = 10 (binary)

i_argA = 1111 (binary)

o_result = 1001 (decimal)

o_status = 1100 (binary), bit 3 = 1 i bit 2 = 1: w wyniku jest parzysta liczba jedynek, wynik jest liczbą ujemną



Zdjęcie nr 9 – U2 na ZM

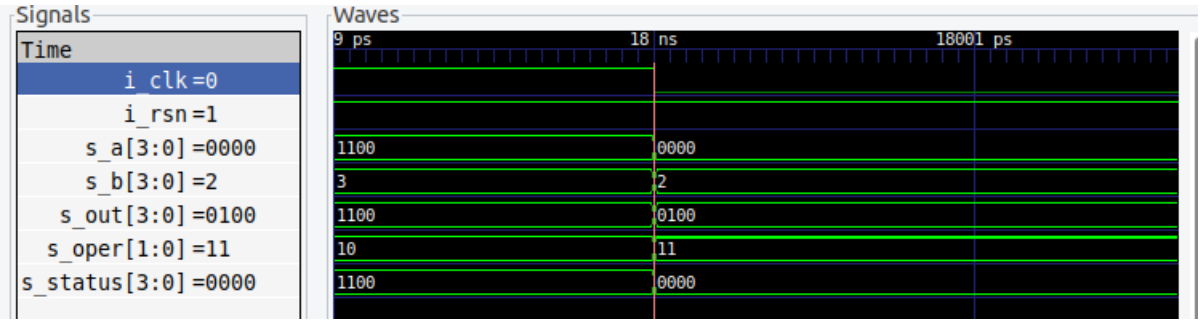
Dla zdjęcia powyżej:

i_oper = 10 (binary)

i_argA = 1100 (binary)

o_result = 1100 (decimal)

o_status = 1100 (binary), bit 3 = 1 i bit 2 = 1: w wyniku jest parzysta liczba jedynek, wynik jest liczbą ujemną



Zdjęcie nr 10 – Zamiana bitu na 1

Dla zdjęcia powyżej:

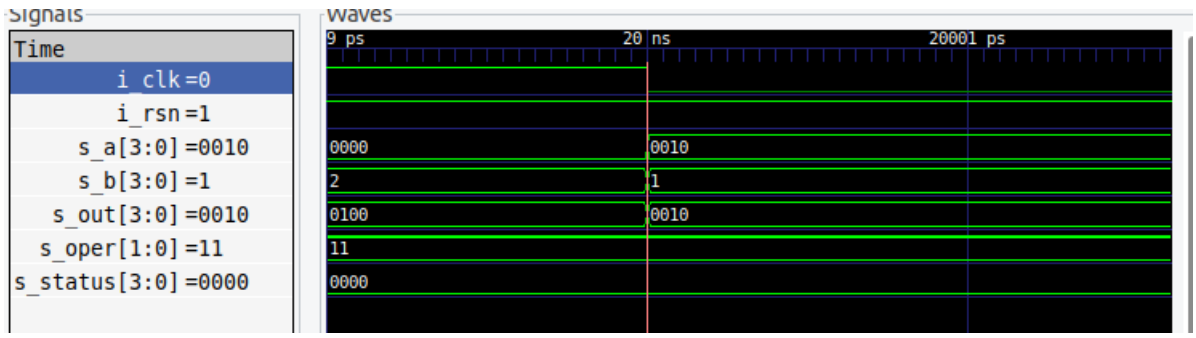
i_oper = 11 (binary)

i_argA = 0000 (binary)

i_argB = 2 (decimal)

o_result = 0100 (decimal)

o_status = 0000 (binary)



Zdjęcie nr 11 – Zamiana bitu na 1

Dla zdjęcia powyżej:

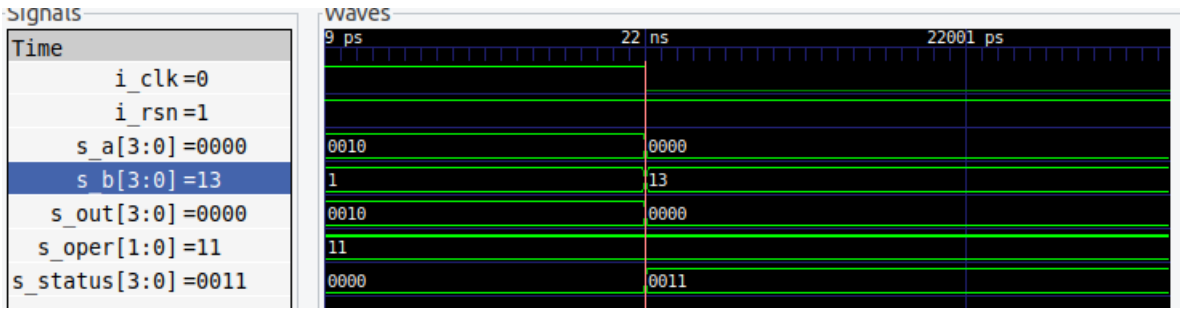
i_oper = 11 (binary)

i_argA = 0010 (binary)

i_argB = 1 (decimal)

o_result = 0010 (decimal)

o_status = 0000 (binary)



Zdjęcie nr 12 – Zamiana bitu na 1

Dla zdjęcia powyżej:

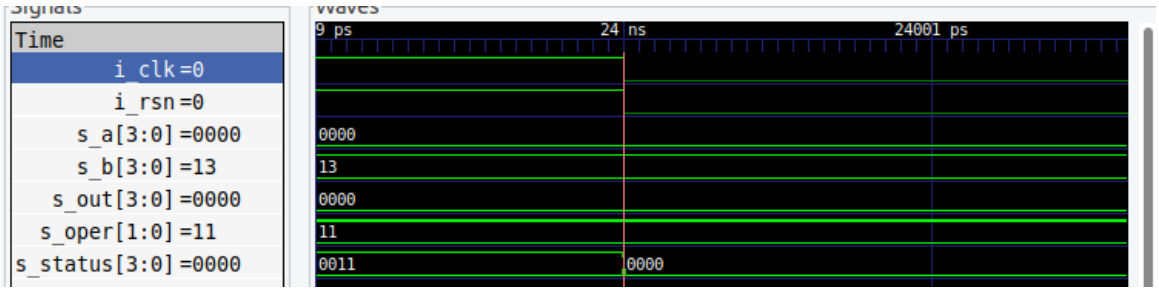
i_oper = 11 (binary)

i_argA = 0000 (binary)

i_argB = 13 (decimal)

o_result = 0000 (decimal)

o_status = 0011 (binary), bit 0 = 1 i bit 1 = 1, mamy error i wynikiem są 4 zera



Zdjęcie nr 13 – Koniec działania programu

- Raporty syntezy

```
3.23.1.2. Re-integrating ABC results.
ABC RESULTS:               AND cells:        8
ABC RESULTS:            ANDNOT cells:       22
ABC RESULTS:              AOI3 cells:       15
ABC RESULTS:              AOI4 cells:        1
ABC RESULTS:               MUX cells:        9
ABC RESULTS:              NAND cells:        3
ABC RESULTS:               NOR cells:        9
ABC RESULTS:               NOT cells:        6
ABC RESULTS:              OAI3 cells:        7
ABC RESULTS:              OAI4 cells:        3
ABC RESULTS:                OR cells:       13
ABC RESULTS:             ORNOT cells:        4
ABC RESULTS:              XNOR cells:       12
ABC RESULTS:               XOR cells:        4
ABC RESULTS:        internal signals:      170
ABC RESULTS:           input signals:       11
ABC RESULTS:          output signals:        8
Removing temp directory.
```

Zdjęcie nr 14 – Raport syntezy logicznej (plik synth.log)

```
=== exe_unit_w12_rtl ===

   Number of wires:                114
   Number of wire bits:            158
   Number of public wires:           8
   Number of public wire bits:      52
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                114
     $_ANDNOT_                      22
     $_AND_                          8
     $_AOI3_                        15
     $_AOI4_                         1
     $_MUX_                          9
     $_NAND_                         3
     $_NOR_                          9
     $_NOT_                          6
     $_OAI3_                         7
     $_OAI4_                         3
     $_ORNOT_                        4
     $_OR_                          13
     $_XNOR_                        10
     $_XOR_                          4
```

Zdjęcie nr 15 – Raport syntezy logicznej (plik synth.log)

```
4.1.2. Re-integrating ABC results.
ABC RESULTS:               AND cells:       60
ABC RESULTS:               NOT cells:       26
ABC RESULTS:                OR cells:       53
ABC RESULTS:               XOR cells:       10
ABC RESULTS:        internal signals:      106
ABC RESULTS:           input signals:       11
ABC RESULTS:          output signals:        8
Removing temp directory.
```

Zdjęcie nr 16 - Raport syntezy logicznej (plik synth.log)

```
exe_unit_w12.sv:21: sorry: constant selects in always_* processes are not currently supported (all bits will be included).
exe_unit_w12.sv:21: sorry: constant selects in always_* processes are not currently supported (all bits will be included).
```

Zdjęcie nr 17 – Raport z syntezy iverilog (plik