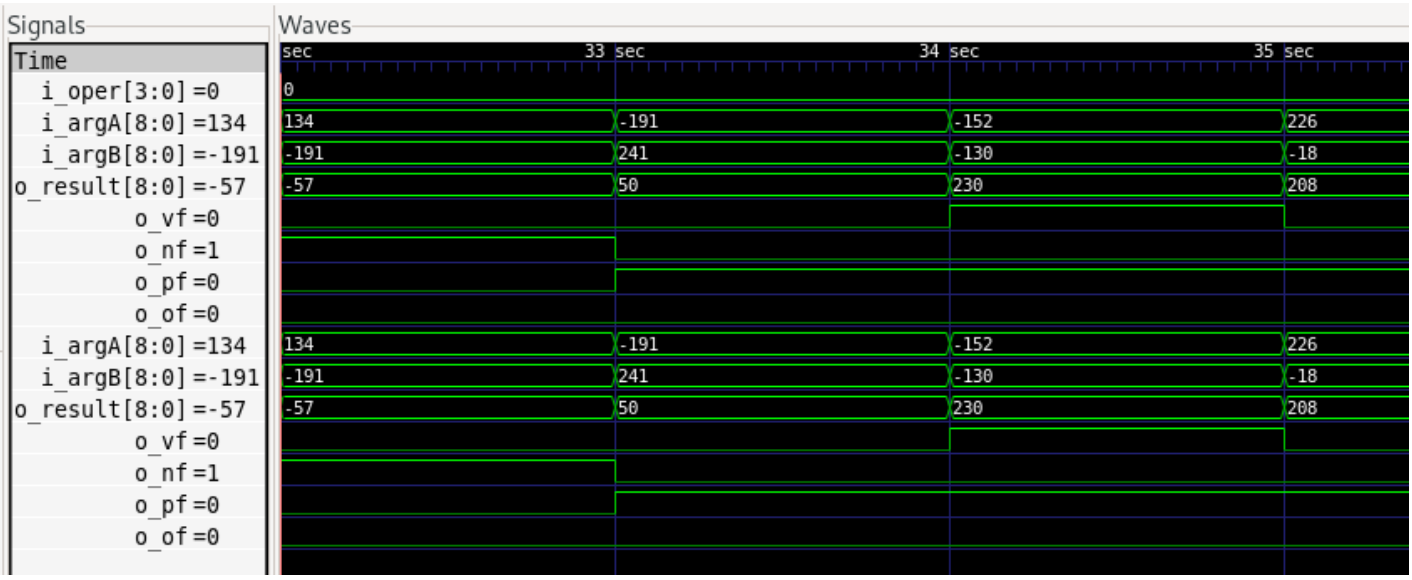
Dokumentacja Projektu 2 SCK- testy jednostki exe_unit

Karol Bogumił 310 430

Testy poszczególnych operacji:
Poniżej znajdują się testy każdej operacji realizowanej w exe_unit. Dalej zamieściłem kolejne testy skoncentrowane wyłącznie na flagach. Jeśli moduł nie dawał wyników zgodnych z prawdą, wprowadzałem w nim zmiany.

1) operacja dodawania (adder):



Na zrzutach ekranu występujących w dokumentacji, sygnały wyżej są sygnałami z jednostki po syntezie, a te poniżej są wprost z opisu exe_unit.

Sygnały są ze sobą zgodne, a wyniki pokrywają się z oczekiwanymi od sumatora wartościami.

Flaga przeniesienia poprawnie wskazuje wartość 1, jeśli wynikiem dodawania dwóch ujemnych liczb jest liczba dodatnia lub wynikiem dodawania dwóch dodatnich liczb jest liczba ujemna. Dochodzi wówczas do wyczerpania miejsc na wektorze.

Flagi dopełnienia do parzystej i nieparzystej ilości 1 w wektorze wyjściowym działają prawidłowo, chociaż tutaj dla wygody zastosowałem zapis dziesiętny, jednak widać, że flagi pf oraz nf nigdy nie przyjmują tej samej wartości.

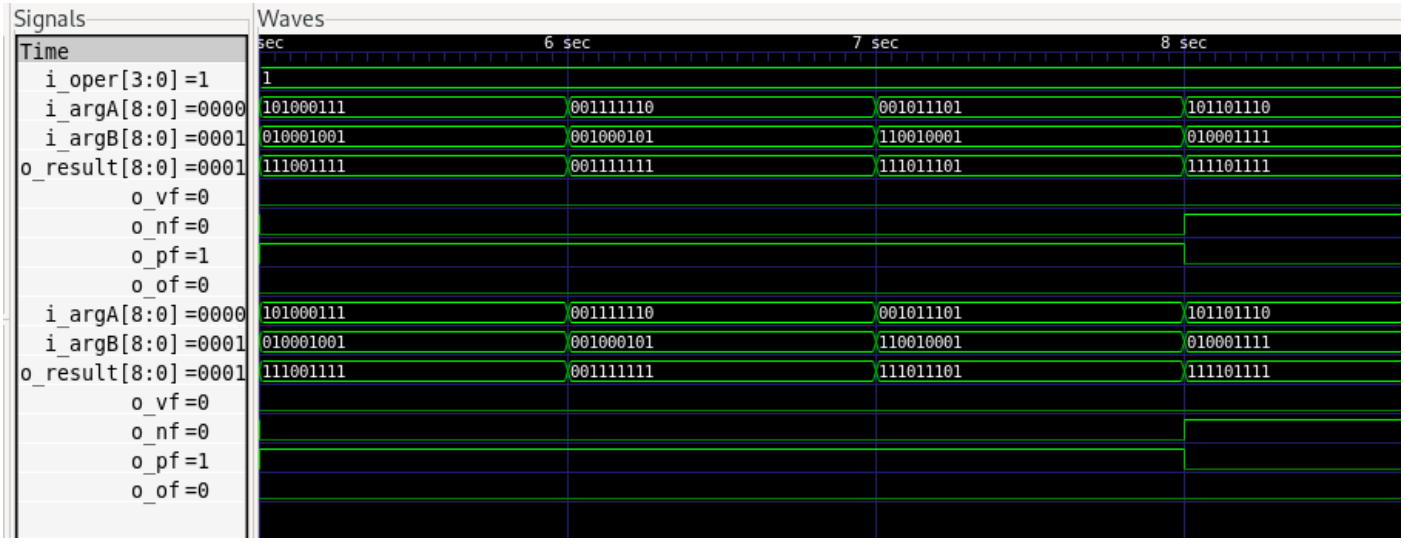
Flaga informująca, że wynik to same jedynki również działa, jednak aby natrafić na taki wynik najlepiej przetestować to w ramach specyficznych wartości dostępnych w testbench'u.

Tabela adder:

ADDER	Dane 1	Dane 2	Dane 3	Dane 4
i_argA	134	-191	-152	226
i_argB	-191	241	-130	-18
Wynik oczekiwany	-57	50	-282: nastąpi przepełnienie	208
Wynik exe_unit	-57	50	230	208
Wynik exe_unit_rtl	-57	50	230	208
Flaga o_vf	0	0	1	0
Flaga o_nf	1	0	0	0
Flaga o_pf	0	1	1	1
Flaga o_of	0	0	0	0

2) operacja or:

Zrzut ekranu:



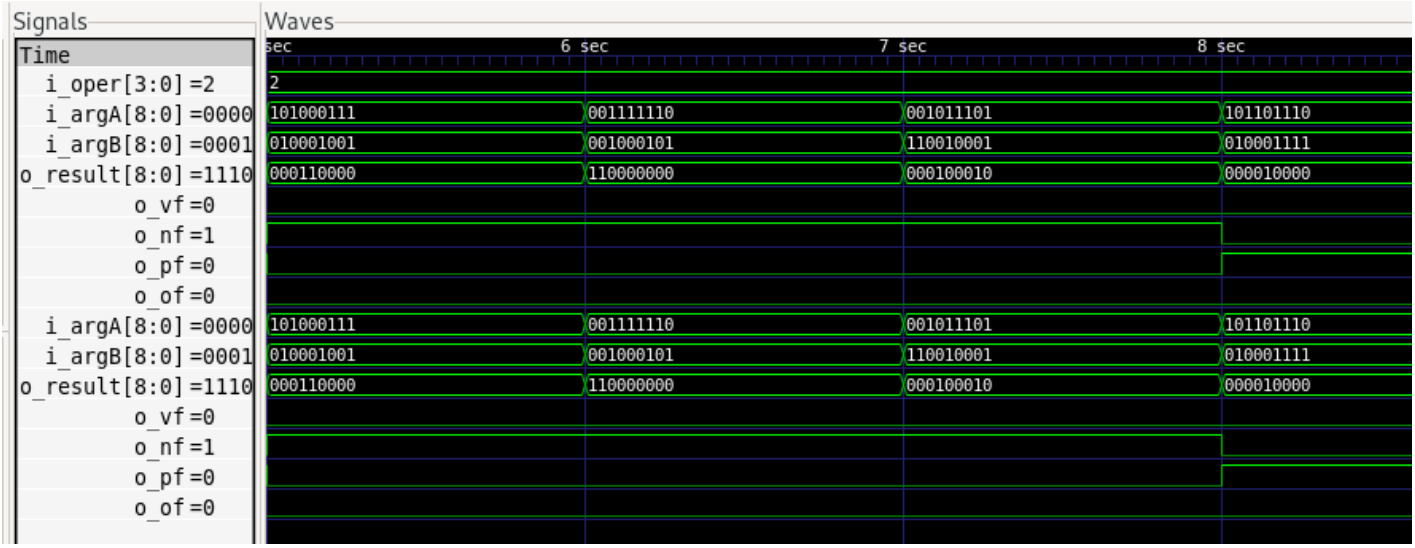
Jednostka poprawnie wykonuje sumę logiczną or. Przepełnienie nie ma prawa zajść oraz widać poprawność działania flag o_nf oraz o_pf.

Tabela or:

OR	Dane 1	Dane 2	Dane 3	Dane 4
i_argA	101000111	001111110	001011101	10101110
i_argB	010001001	001000101	110010001	01000111
Wynik oczekiwany	111001111	001111111	111011101	11101111
Wynik exe_unit	111001111	001111111	111011101	11101111
Wynik exe_unit_rtl	111001111	001111111	111011101	11101111
Flaga o_vf	0	0	0	0
Flaga o_nf	0	0	0	1
Flaga o_pf	1	1	1	0
Flaga o_of	0	0	0	0

3) operacja nor:

zrzut ekranu:



Ta operacja, jak i flagi w przypadku jej wywołania również działa poprawnie, stanowi ona negację wyniku poprzedniej operacji a zrzut ekranu wykonałem dla tych samych wartości argumentów wejściowych, zaobserwować można, jak wyniki tych operacji są dla siebie przeciwieństwami

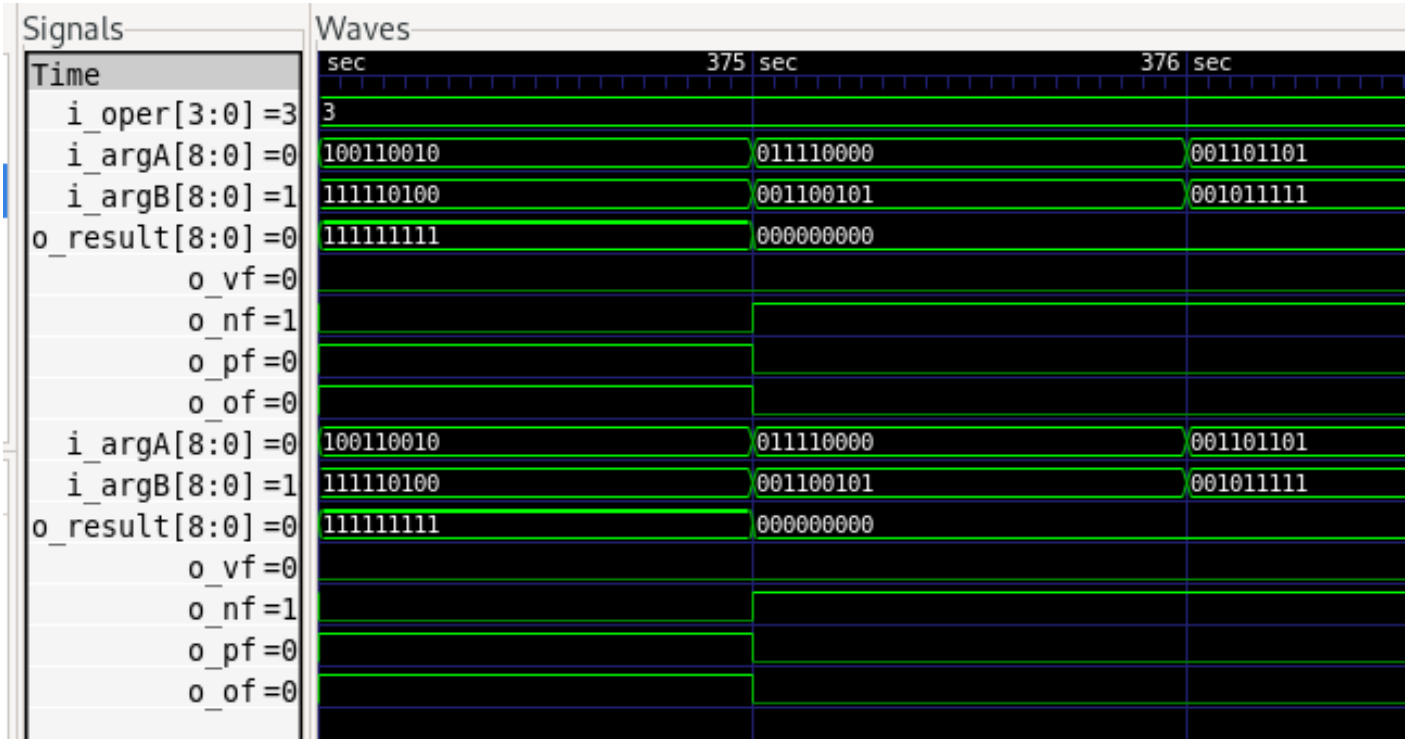
Tabela nor:

NOR	Dane 1	Dane 2	Dane 3	Dane 4
i_argA	101000111	001111110	001011101	101101110
i_argB	010001001	001000101	110010001	010001111
Wynik oczekiwany	000110000	110000000	000100010	000010000
Wynik exe_unit	000110000	110000000	000100010	000010000
Wynik exe_unit_rtl	000110000	110000000	000100010	000010000
Flaga o_vf	0	0	0	0
Flaga o_nf	1	1	1	0
Flaga o_pf	0	0	0	1
Flaga o_of	0	0	0	0

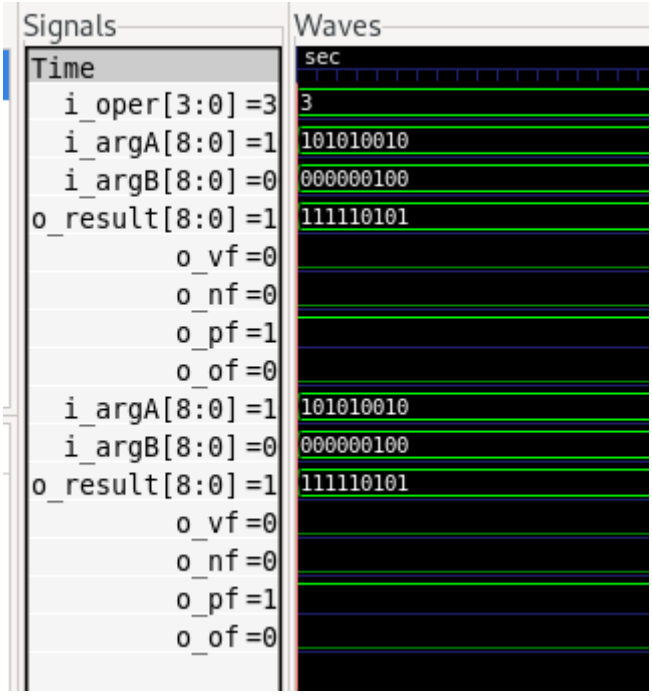
4) operacja arytmetycznego przesunięcia w prawo argumentu A o ilość bitów z argumentu B.

Ponieważ dla 9-bitowego wektora na dane wejściowe i wyjściowe, przesunięcie arytmetyczne o ilość bitów równą wartości argumentu B najczęściej owocuje wynikiem ‘0 lub ‘1, dokonałem symulacji dla specyficznej wartości tak, aby widać było pełną poprawność tego modułu. Tę specyficzną wartość umieściłem na oddzielnym screenie i wypełniłem w tabeli jako czwartą wartość. Operacja ta dobrze obrazuje poprawność działania flagi o_of informującej, że na wyjściu są same jedynki.

Zrzut ekranu (3 wartości losowe):



Zrzut ekranu(specyficzna wartość):



Na tym screenshocie widać, że operacja faktycznie wykonuje się poprawnie.

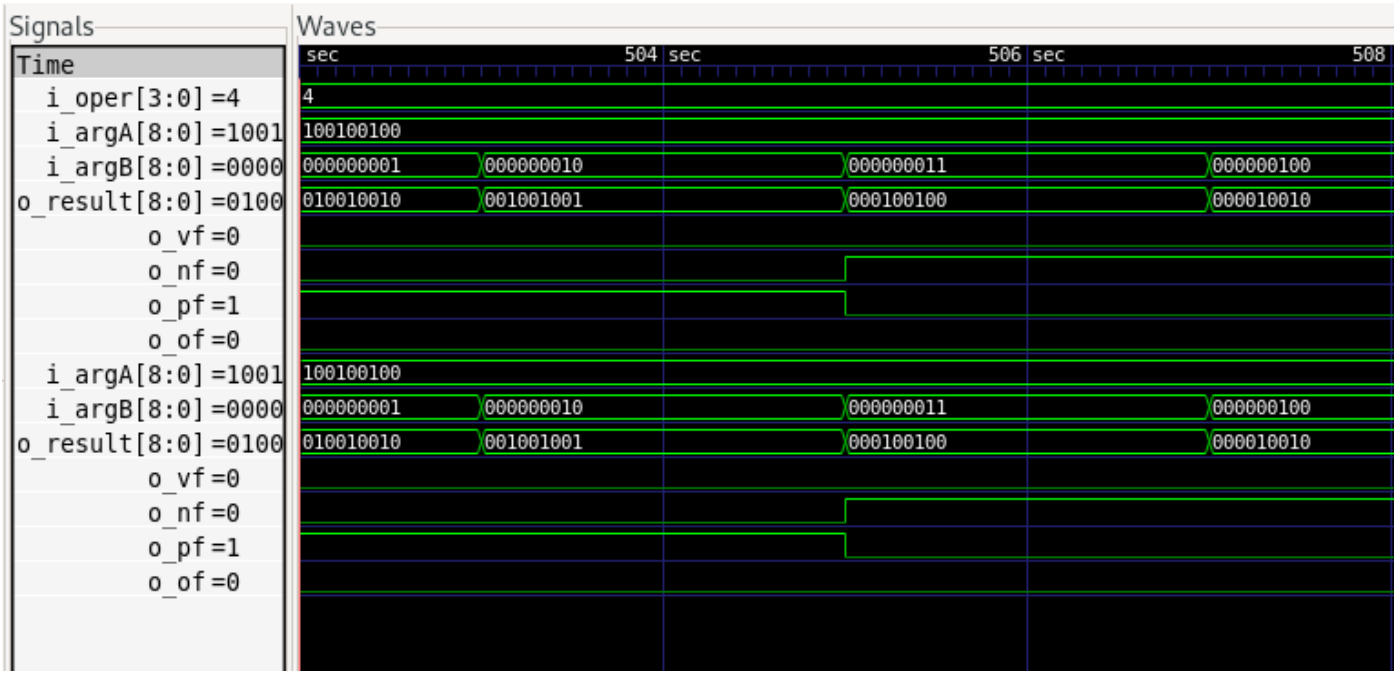
Tabela arytmetycznego przesunięcia w prawo:

> > >	Dane 1	Dane 2	Dane 3	Dane 4
i_argA	100110010	011110000	001101101	101010010
i_argB	111110100	001100101	001011111	000000100
Wynik oczekiwany	111111111	000000000	000000000	111110101
Wynik exe_unit	111111111	000000000	000000000	111110101
Wynik exe_unit_rtl	111111111	000000000	000000000	111110101
Flaga o_vf	0	0	0	0
Flaga o_nf	0	1	1	0
Flaga o_pf	1	0	0	1
Flaga o_of	1	0	0	0

5) operacja logicznego przesunięcia w prawo argumentu A o ilość bitów z argumentu B.

Taka realizacja tej operacji dla losowych wartości argumentów powoduje najczęściej wynik operacji = ‘0, spośród losowych wartości znalazłem jeden moment, gdzie argA było logicznie przesuwane tylko o 2 bity, bo argB przyjął wartość (bin)10. Aby udowodnić że poprawność tego wyniku nie jest przypadkiem dodałem 4 specyficzne wartości w testbench’u, które w tabeli umieszczę jako cztery dane.

Zrzut ekranu:



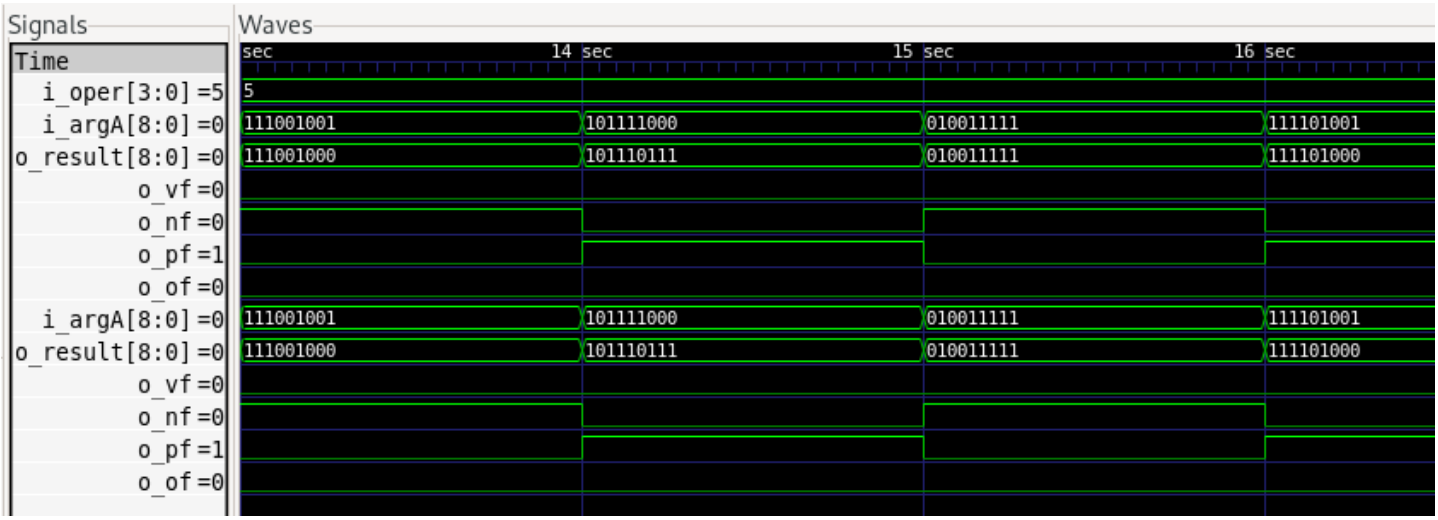
widać tutaj jak ten sam sygnał wejściowy argA został logicznie przesuwany kolejno o 1, 2, 3 i finalnie 4 bity.

Tabela logicznego przesunięcia w prawo:

> >	Dane 1	Dane 2	Dane 3	Dane 4
i_argA	100100100	100100100	100100100	100100100
i_argB	000000001	000000010	000000011	000000100
Wynik oczekiwany	010010010	001001001	000100100	000010010
Wynik exe_unit	010010010	001001001	000100100	000010010
Wynik exe_unit_rtl	010010010	001001001	000100100	000010010
Flaga o_vf	0	0	0	0
Flaga o_nf	0	0	1	1
Flaga o_pf	1	1	0	0
Flaga o_of	0	0	0	0

6) operacja konwersji argumentu A z kodu U2 na kod U1.

Zrzut ekranu:

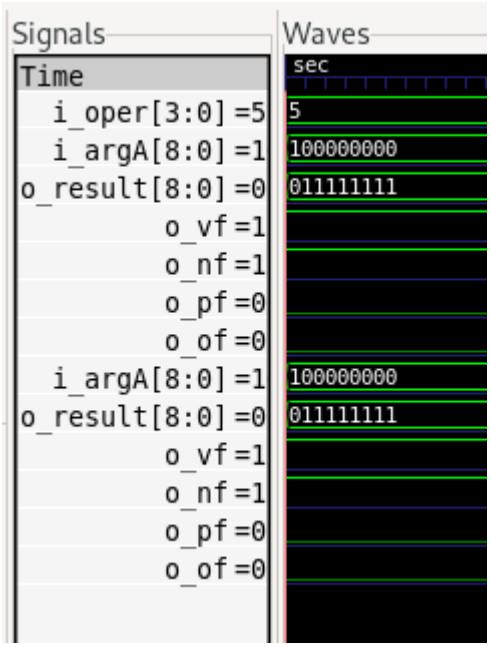


W tej operacji do przepełnienia może dojść w bardzo specyficznej sytuacji, kiedy pierwszy bit ma wartość 1, a reszta bitów jest wyzerowana. Przebieg z takiej sytuacji i dowód, że flaga overflow działa dla tej operacji umieszczę pod tabelą.

Tabela konwersji U2 na U1:

U2>U1	Dane 1	Dane 2	Dane 3	Dane 4
i_argA	111001001	101111000	010011111	111101001
i_argB	Nieistotne	Nieistotne	Nieistotne	Nieistotne
Wynik oczekiwany	111001000	101110111	010011111	111101000
Wynik exe_unit	111001000	101110111	010011111	111101000
Wynik exe_unit_rtl	111001000	101110111	010011111	111101000
Flaga o_vf	0	0	0	0
Flaga o_nf	1	0	1	0
Flaga o_pf	0	1	0	1
Flaga o_of	0	0	0	0

Przypadek przepełnienia:



Widać, że w tym przypadku flaga przepełnienia osiąga wartość 1

7) operacja konwersji argumentu A z kodu znak-moduł na kod U2

Zrzut ekranu:

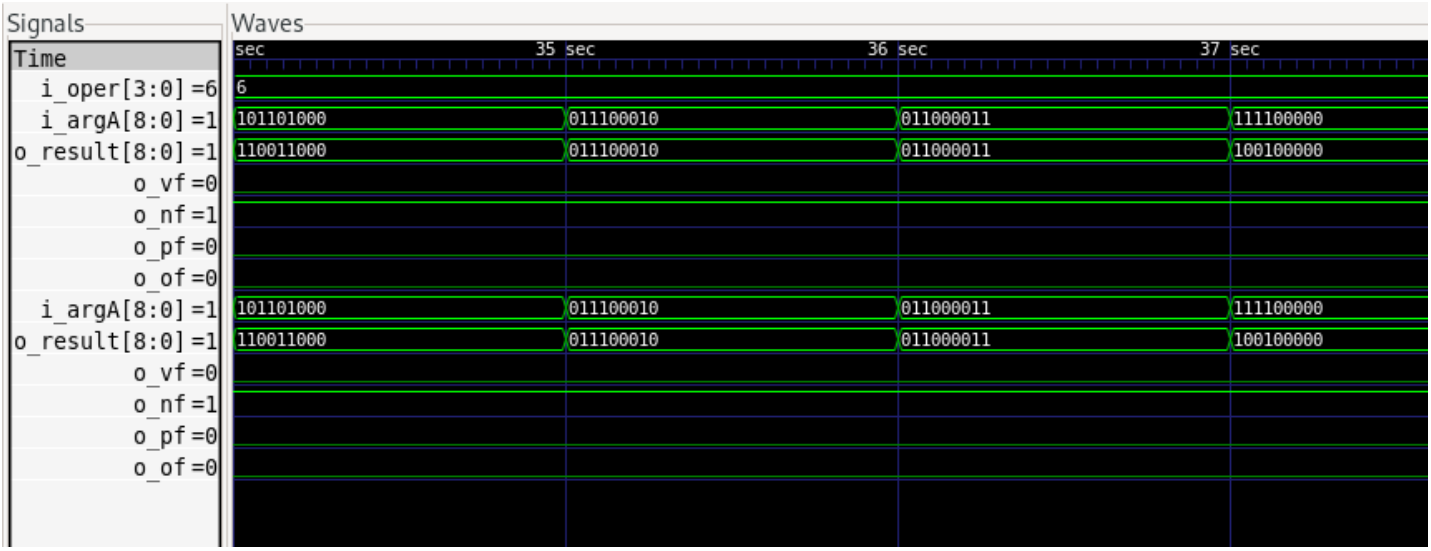


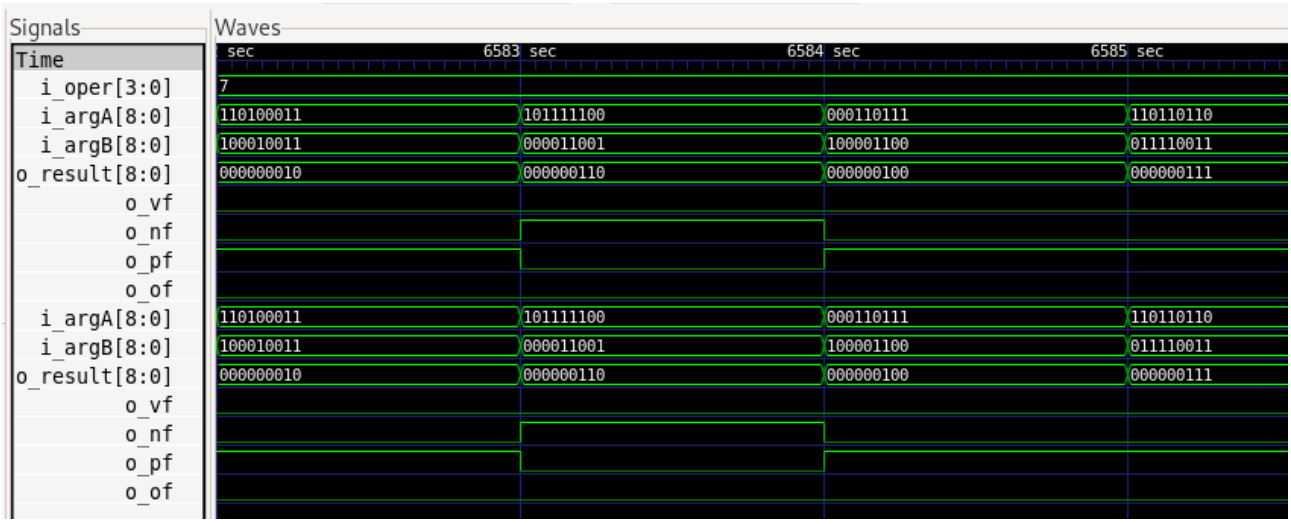
Tabela SM to U2:

SM>U2	Dane 1	Dane 2	Dane 3	Dane 4
i_argA	101101000	011100010	011000011	111100000
i_argB	nieistotne	nieistotne	nieistotne	nieistotne
Wynik oczekiwany	110011000	011100010	011000011	100100000
Wynik exe_unit	110011000	011100010	011000011	100100000
Wynik exe_unit_rtl	110011000	011100010	011000011	100100000
Flaga o_vf	0	0	0	0
Flaga o_nf	1	1	1	1
Flaga o_pf	0	0	0	0
Flaga o_of	0	0	0	0

8) wyznaczanie kodu CRC-3

W mojej jednostce exe_unit moduły oparte o crc polegały wyłącznie na wyznaczaniu kodu crc. Aby poprawić komfort użytkowania tych modułów zmieniłem trochę to, w jaki sposób działają. Od teraz 3 bity wyzerowane (lub dla crc4 4 bity wyzerowane) nie stanowią części argumentu B, tylko są dostarczane zawsze, a wielomian stanowi ostatnie 4 lub 5 bitów argumentu B.

Zrzut ekranu:



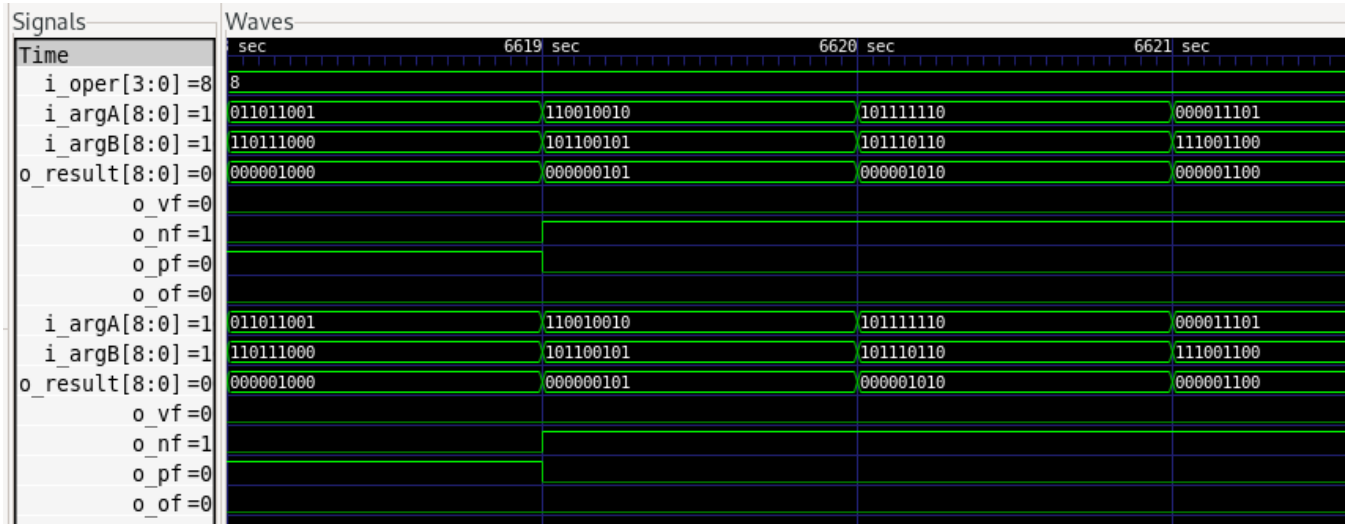
Widać, że wynikiem są zawsze 3 bitowe kody.
W tym module do przepełnienia nie dochodzi, widać również antagonistyczny charakter flag pf oraz nf.

Tabela wyznaczania kodu CRC-3:

CRC-3	Dane 1	Dane 2	Dane 3	Dane 4
i_argA	110100011	101111100	000110111	110110110
i_argB	100010011	000011001	100001100	011110011
Wynik oczekiwany	010	110	100	111
Wynik exe_unit	010	110	100	111
Wynik exe_unit_rtl	010	110	100	111
Flaga o_vf	0	0	0	0
Flaga o_nf	0	1	0	0
Flaga o_pf	1	0	1	1
Flaga o_of	0	0	0	0

9) wyznaczanie kodu CRC-4

Zrzut ekranu:



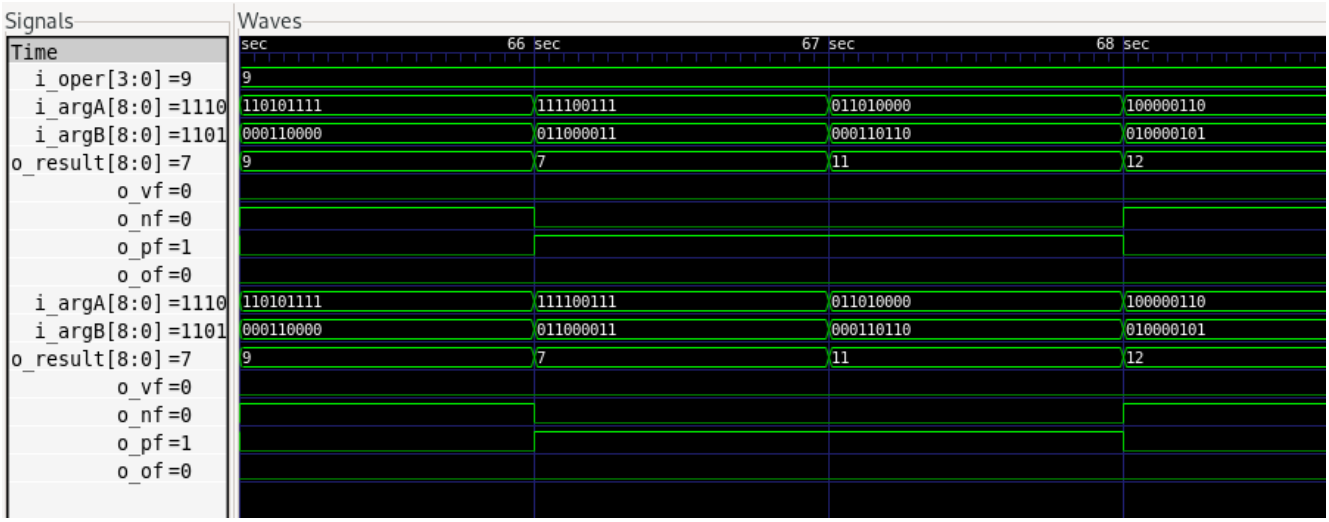
Tym razem widać, że wynikiem są zawsze 4 bitowe kody.
W tym module do przepełnienia nie dochodzi, widać również antagonistyczny charakter flag pf oraz nf.

Tabela wyznaczania kodu CRC-4:

CRC -4	Dane 1	Dane 2	Dane 3	Dane 4
i_argA	011011001	110010010	101111110	000011101
i_argB	110111000	101100101	101110110	111001100
Wynik oczekiwany	1000	0101	1010	1100
Wynik exe_unit	1000	0101	1010	1100
Wynik exe_unit_rtl	1000	0101	1010	1100
Flaga o_vf	0	0	0	0
Flaga o_nf	0	1	1	1
Flaga o_pf	1	0	0	0
Flaga o_of	0	0	0	0

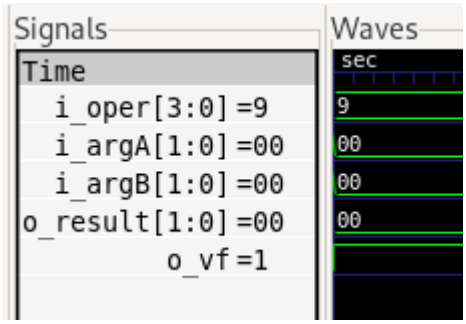
10) zliczanie zer w obu argumentach wejściowych

Zrzut ekranu:



W tym module jest szansa aby osiągnąć przepełnienie, jednak wymaga to specyficznych wartości, dlatego działanie flagi przepełnienia udowodnię za pomocą opcji specyficznego przypadku w testbench'u.
Jeśli ilość bitów na wejściach jest mniejsza od 3, to w przypadku gdy wszystkie argumenty będą miały wartość 0, ilość zer nie będzie mogła być reprezentowana przez wyjście.
Np 4 zer nie da się zapisać na 2 bitach, musi dojść do przepełnienia.

Zrzut ekranu (specyficzna wartość):



Na powyższym obrazku widać, że flaga przepełnienia działa dla tego modułu.
Tabela zliczania zer:

COUNT- 0	Dane 1	Dane 2	Dane 3	Dane 4
i_argA	110101111	111100111	011010000	100000110
i_argB	000110000	011000011	000110110	010000101
Wynik oczekiwany	9	7	11	12
Wynik exe_unit	9	7	11	12
Wynik exe_unit_rtl	9	7	11	12

Flaga o_vf	0	0	0	0
Flaga o_nf	1	0	0	1
Flaga o_pf	0	1	1	0
Flaga o_of	0	0	0	0

11) konwersja z kodu termometrowego na kod nkb

Moduł ten wymaga od użytkownika wprowadzenia argumentu A w poprawny sposób, dlatego weryfikację przeprowadziłem na własnych testach.

Zrzut ekranu:

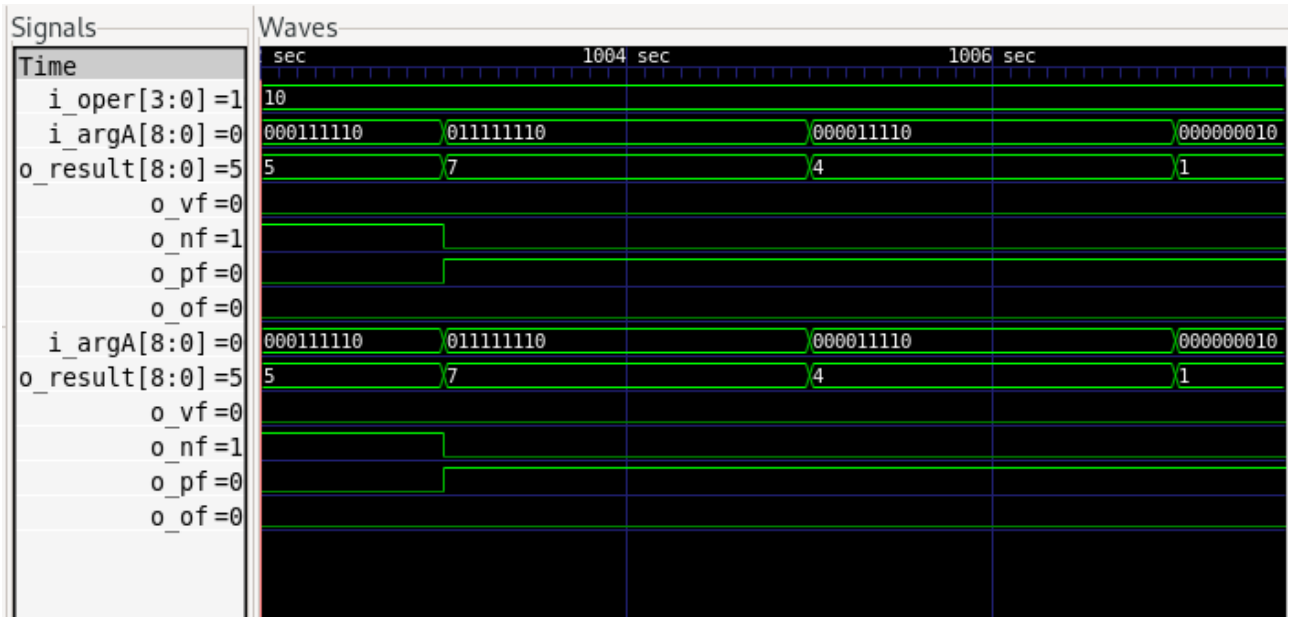


Tabela konwersji kodu termometrowego na kod nkb:

THERMO2NKB	Dane 1	Dane 2	Dane 3	Dane 4
i_argA	000111110	011111110	000011110	000000010
i_argB	nieistotne	nieistotne	nieistotne	nieistotne
Wynik oczekiwany	5	7	4	1
Wynik exe_unit	5	7	4	1
Wynik exe_unit_rtl	5	7	4	1
Flaga o_vf	0	0	0	0
Flaga o_nf	1	0	0	0
Flaga o_pf	0	1	1	1
Flaga o_of	0	0	0	0

12) konwersja z kodu nkb na kod gorącej jedynki

Ponieważ wektor wyjścia ograniczony jest do liczby bitów równej szerokości wektora wejściowego, z uwagi na charakter kodu gorącej jedynki, większość przypadków owocuje w przepełnienie.

Zrzut ekranu:

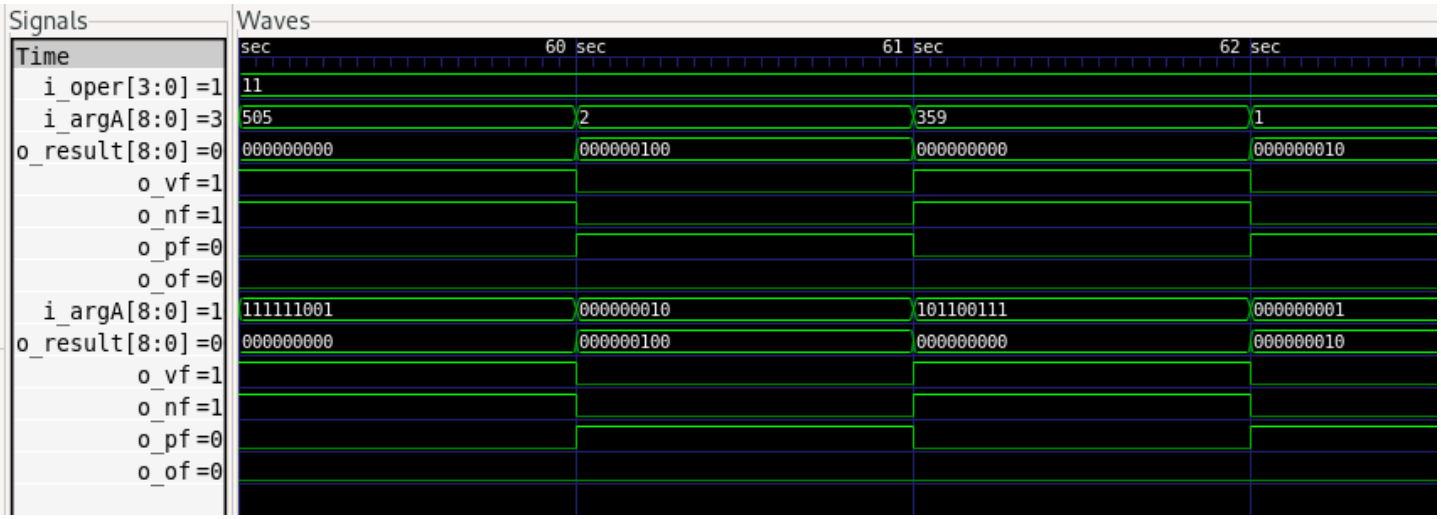


Tabela nkb na onehot

Nkb>onehot	Dane 1	Dane 2	Dane 3	Dane 4
i_argA	505	2	359	1
i_argB	nieistotne	nieistotne	nieistotne	nieistotne
Wynik oczekiwany	przepełnienie	000000100	przepełnienie	000000010
Wynik exe_unit	000000000	000000100	000000000	000000010
Wynik exe_unit_rtl	000000000	000000100	000000000	000000010
Flaga o_vf	1	0	1	0
Flaga o_nf	1	0	1	0
Flaga o_pf	0	1	0	1
Flaga o_of	0	0	0	0

Testy skoncentrowane na flagach:

1) flaga przepełnienia o_vf

flagę przepełnień testowałem osobno dla każdego modułu, który przewidywał możliwość overflow. Są to: sumator, konwersja z U2 na U1, zliczanie zer oraz konwersja z nkb na onehot

Testy się powiodły, tam gdzie przepełnienie faktycznie zachodzi, flaga przyjmuje wartość 1, poniżej znajduje się zrzut ekranu oraz tabela dla przepełnienia w module sumatora:
(o_vf=1, jeśli wynik dodawania dwóch dodatnich liczb jest ujemny lub jeśli wynik dodawania dwóch ujemnych liczb jest dodatni)

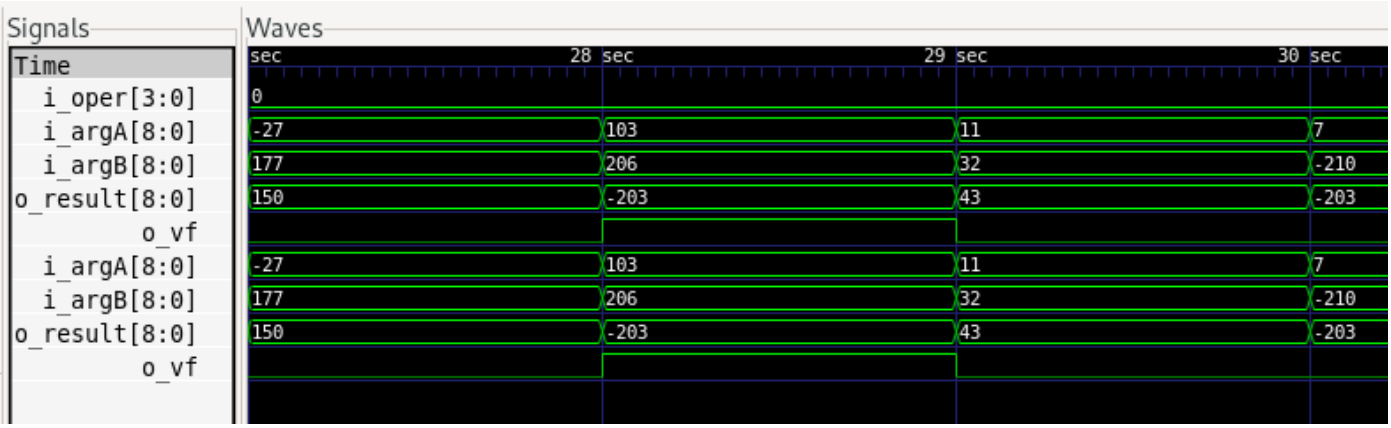


Tabela o_vf dla sumatora:

SUMATOR_VF	Dane 1	Dane 2	Dane 3	Dane 4
i_argA	-27	103	11	7
i_argB	177	206	32	-210
o_result	150	-203	43	-203
Wartość oczekiwana flagi	0	1	0	0
Flaga exe_unit	0	1	0	0
Flaga exe_unit_rtl	0	1	0	0

2) flaga dopełnienia do nieparzystej ilości jedynek w wyniku o_nf

Jeśli liczba jedynek na wyjściu jest parzysta, flaga ta wyświetla wartość 1 dopełniając liczbę jedynek tak, aby ta była nieparzysta.
Flagi dopełnień nf oraz pf w przeciwieństwie do flagi overflow występują zawsze i są sobie zaprzeczeniem tzn. zawsze uruchamia się tylko jedna z nich.

Zrzut ekranu:

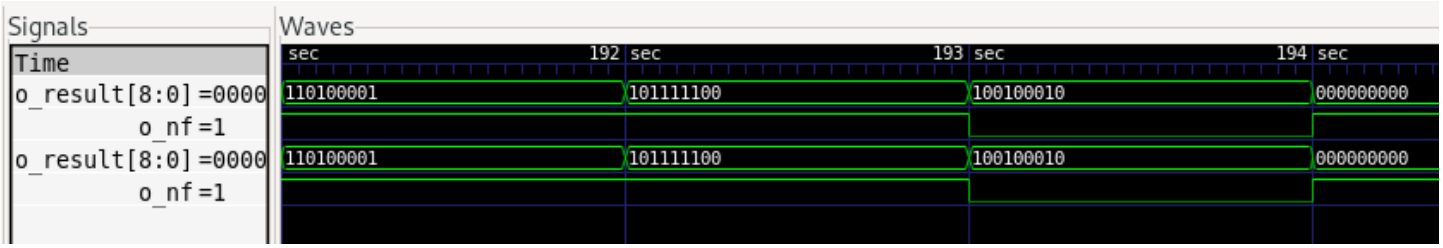


Tabela o_nf:

O_NF	Dane 1	Dane 2	Dane 3	Dane 4
Wynik o_result	110100001	101111100	100100010	000000000
Wartość oczekiwana flagi	1	1	0	1
Flaga exe_unit	1	1	0	1
Flaga exe_unit_rtl	1	1	0	1

3) flaga dopełnienia do parzystej ilości jedynek w wyniku o_pf

Flaga ta działa dualnie do flagi poprzedniej.
Jeśli liczba jedynek na wyjściu jest nieparzysta, flaga ta wyświetla wartość 1 dopełniając liczbę jedynek tak, aby ta była parzysta.

Zrzut ekranu:

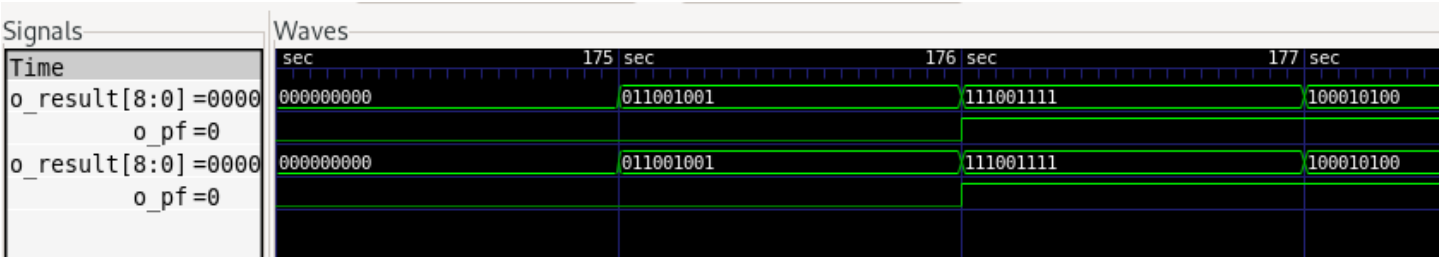


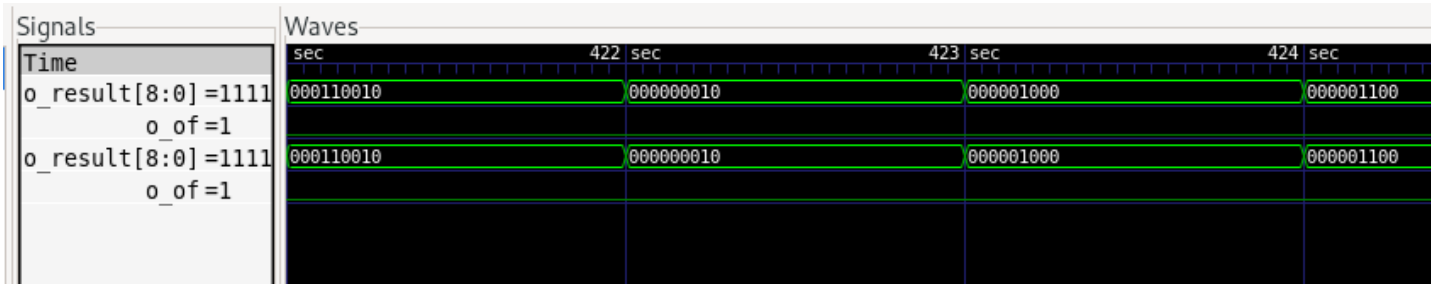
Tabela o_pf:

O_PF	Dane 1	Dane 2	Dane 3	Dane 4
Wynik o_result	000000000	011001001	111001111	100010100
Wartość oczekiwana flagi	0	0	1	1
Flaga exe_unit	0	0	1	1
Flaga exe_unit_rtl	0	0	1	1

4) flaga informująca, że wynikiem są same jedynki o_of

W bardzo wyjątkowych przypadkach, sygnał wyjściowy może osiągnąć wartość ‘1 i wtedy ta flaga się uruchamia. Do jej zaprezentowania użyję specyficznego przypadku.
(suma ‘0 oraz ‘1)

Zrzut ekranu:



Ta flaga najczęściej przyjmuje wartość 0, chyba że wynik jest równy ‘1:

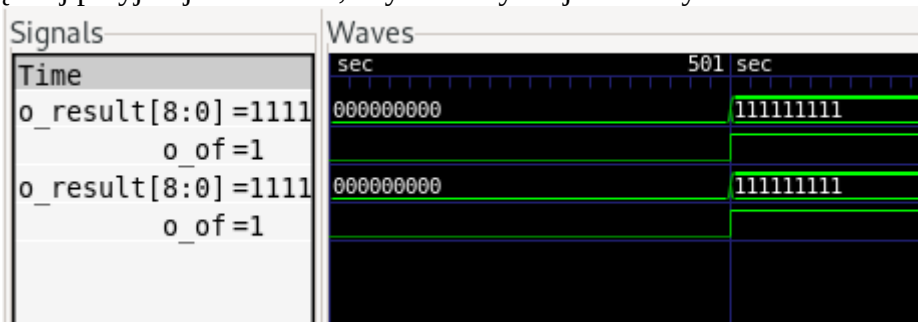


Tabela o_of:

O_OF	Dane 1	Dane 2	Dane 3	Dane 4
Wynik o_result	000110010	000000010	000001000	111111111
Wartość oczekiwana flagi	0	0	0	1
Flaga exe_unit	0	0	0	1
Flaga exe_unit_rtl	0	0	0	1

Powyższe zrzuty ekranu i tabele wykazują, że testy exe_unit przebiegły pomyślnie i dają oczekiwany rezultat- moduły, które nie spełniały swoich założeń zostały naprawione i teraz cała jednostka działa poprawnie.