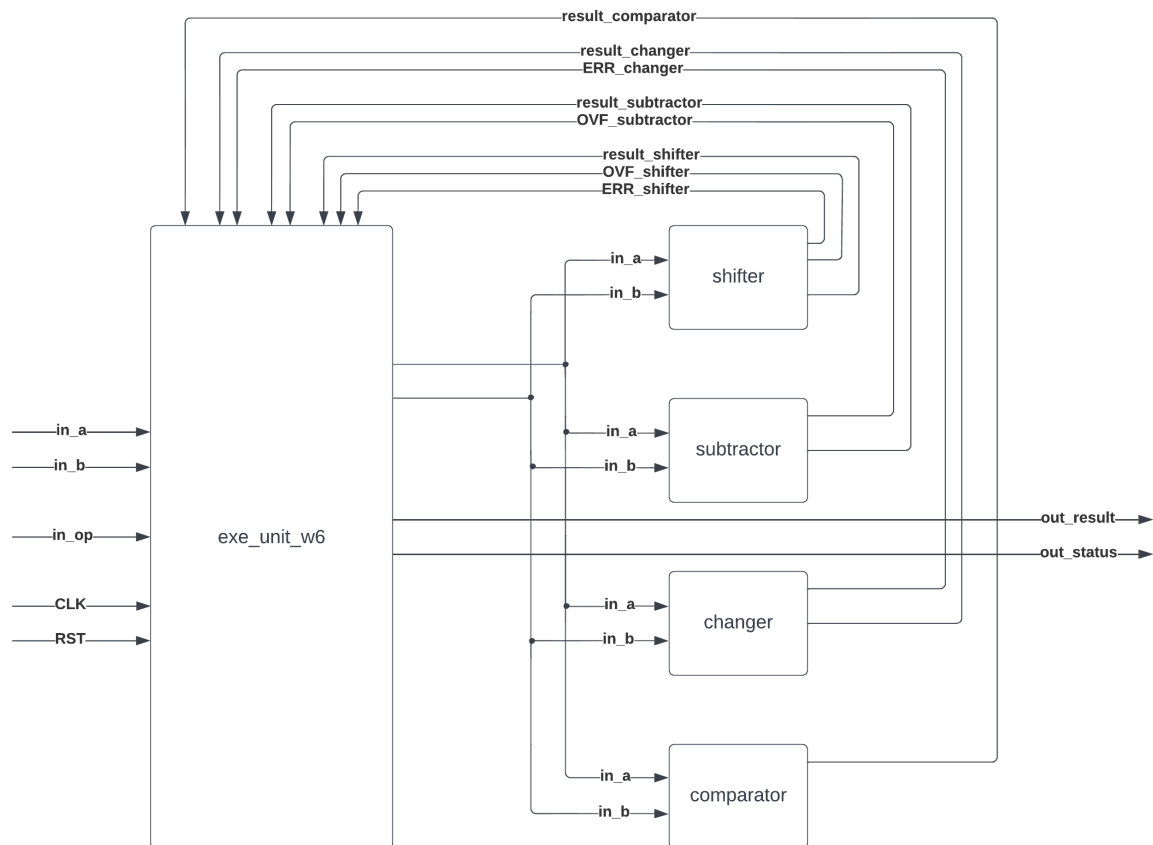


Specyfikacja modułu exe_unit_w6

Patryk Korycki, nr albumu 318529

8 grudnia 2022 r.



Schemat blokowy jednostki **exe_unit_w6**

Opis jednostki

Zadanie polegało na implementacji modułu **exe_unit_w6**. Zadaniem układu jest wykonywanie kilku zdefiniowanych operacji matematycznych i logicznych w systemie liczb Znak-Moduł

Wejścia

Działania są wykonywane na dwóch n-bitowych liczbach A i B (wejścia **in_a** i **in_b**) podanych na wejście. Wyboru operacji dokonuje się za pomocą 2-bitowego wejścia **in_op**.

Dodatkowo układ jest wyposażony w wejście zegarowe wyzwalane zboczem narastającym **CLK** i resetu synchronicznego **RST**. Stan niski na wejściu reset skutkuje przywróceniem stanu układu do stanu początkowego, czyli ustawienia operacji odejmowania dla obydwu wejść wynoszących zero. Pełna lista wejść układu jest przedstawiona w Tabeli 1. Poszczególne operacje wraz z wartościami wejścia **i_op** zostały wymienione w Tabeli 2.

Wejście	Funcja	Ilość bitów wejściowych
in_a	Pierwszy składnik obliczeń	N-bitów
in_b	Drugi składnik obliczeń	N-bitów
in_op	Wybór operacji	2-bity
RST	Reset synchroniczny	1-bit
CLK	Taktowanie układu	1-bit

Tabela 1: Lista przedstawiająca wszystkie wejścia jednostki

Wartość wejścia in_op	Operacja	Flagi wyjściowe
0b00	Odejmowanie Liczby A od liczby B	OVF, EVEN, SINGLE
0b01	Porównanie, czy liczba $A \geq B$. Jeśli tak, to wyjście jest dodatnie. Jeśli nie to wyjście jest zerem	EVEN
0b10	Przesunięcie liczby A o B bitów w lewo (z zachowaniem znaku). Gdy B ma wartość ujemną lub jest większe od liczby bitów liczby A, zwróć błąd.	OVF, ERROR, EVEN, SINGLE
0b11	Zmiana bitu liczby A na pozycji B. Gdy B ma wartość ujemną lub jest większe od liczby bitów liczby A, zwróć błąd.	ERROR, EVEN, SINGLE

Tabela 2: Opis poszczególnych operacji wraz z kodami wejściowymi i dostępnymi flagami

Wyjścia

Na wyjściu modułu dostępne są dwie wartości: **out_result** i **out_status**. Wyjście **out_result** zawiera wynik ostatnio wykonywanej operacji. Na wyjściu **out_status** pojawiają się flagi informacyjne dotyczące ostatnio wykonanej operacji. Każda z wykonywanych operacji ma możliwość zmiany wyłącznie wybranych flag (opisanych w Tabeli 2). Pełna lista wyjść znajduje się w Tabeli 3. Flagi wyjściowe z każdej operacji są zawarte w Tabeli 4.

Flagi

Rejestr wyjściowy **out_status** składa się z 4 flag sygnalizujących stan wyjścia układu. Dodatkowo należy wspomnieć, że podczas załączenia flagi **ERROR** wyjście z układu jest

Wyjście	Funkcja	Ilość bitów wyjściowych
out_result	Wynik ostatnio wykonanej operacji	N-bitów
out_status	Rejestr z flagami informacyjnymi	4-bity

Tabela 3: Lista przedstawiająca wszystkie wyjścia układu

Bit	3	2	1	0
	SINGLE	OVF	EVEN	ERROR

Tabela 4: Bity dostępne w wektorze wyjściowym **out_status**

nieokreślone i wynik będący wtedy na wyjściu w ogóle nie powinien być brany pod uwagę.

- **SINGLE** - Flaga informująca, że w wyniku jest tylko jedno zero
- **OVF** - Flaga informująca o przepełnieniu podczas operacji
- **EVEN** - Flaga informująca, że liczba zer w wyniku jest parzysta.
- **ERROR** - Flaga informująca o błędzie podczas wykonywania operacji.

Instancjonowanie

W kodzie 1 pokazano ostateczne zainstancjonowanie jednostki w użyciu. W przypadku układu przed syntezą konieczne może być zdefiniowanie liczby **N**, czyli ilości bitów rejestrów wejściowych i wyjściowych. Wartość ta jest przechowywana w module testowym jako parametr **BITS**. Nazwy podłączonych sygnałów wewnętrznych z przedrostkiem **s_** są jedynie przykładowe i zostały użyte podczas testowania jednostki.

Kod 1 Przykładowe zainstancjonowanie jednostki **exe_unit_w6** w ostatecznym kodzie

```
exe_unit_w6 #(.BITS(N_BITS)) exe_unit_w6_model (.in_a(s_a), .in_b(s_b),
.i_op(s_op), .i_clk(s_clk), .i_rst(s_rst), .o_out(s_out_model),
.o_status(s_status_model));    // model przed syntezą

exe_unit_w6_rtl exe_unit_w6_synth (.in_a(s_a), .in_b(s_b),
.i_op(s_op), .i_clk(s_clk), .i_rst(s_rst), .o_out(s_out_synth),
.o_status(s_status_synth));    // model po syntezie
```

Opis Podmodułów

Jednostka składa się z modułu sterującego i 4 modułów wykonawczych:

- **exe_unit_w6** - Moduł sterujący
- **subtractor** - Moduł odejmujący
- **comparator** - Moduł porównujący
- **shifter** - Moduł wykonujący operację przesunięcia
- **changer** - Moduł zmieniający bit na 1 na danej pozycji

Każdy z modułów wykonawczych odpowiada za daną operację. Z podmodułów wychodzą sygnały zawierające wynik operacji i flagi informacyjne. W module sterującym generowane są flagi **EVEN** oraz **SINGLE** i wybierane jest wyjście odpowiedniego podmodułu.

Synteza logiczna

Synteza została przeprowadzona w programie **yosys 0.22**. Niestety napotymano problemy podczas użytkowania programu w wersji 0.9. Prawdopodobnie było to spowodowane błędami we wcześniejszej wersji oprogramowania do syntezy.

Raport z syntezy

Pełny raport z syntezy znajduje się w pliku **Pliki_projektu/synth.log**. Poniżej znajduje się summaryczne podsumowanie syntezy w postaci wyliczenia ilości użytych bramek i oszacowanie ilości potrzebnych tranzystorów do budowy układu.

Wyliczenie ilości użytych bramek i oszacowanie ilości tranzystorów użytych do budowy jednostki

=== design hierarchy ===

[illegible]

Number of wires:	461
Number of wire bits:	636
Number of public wires:	38
Number of public wire bits:	213
Number of memories:	0
Number of memory bits:	0
Number of processes:	0
Number of cells:	480
\$_AND_	200
\$_NOT_	61
\$_OR_	178
\$_SDFF_PNO_	18
\$_XOR_	23

Estimated number of transistors: 2666+

Symulacja i testy

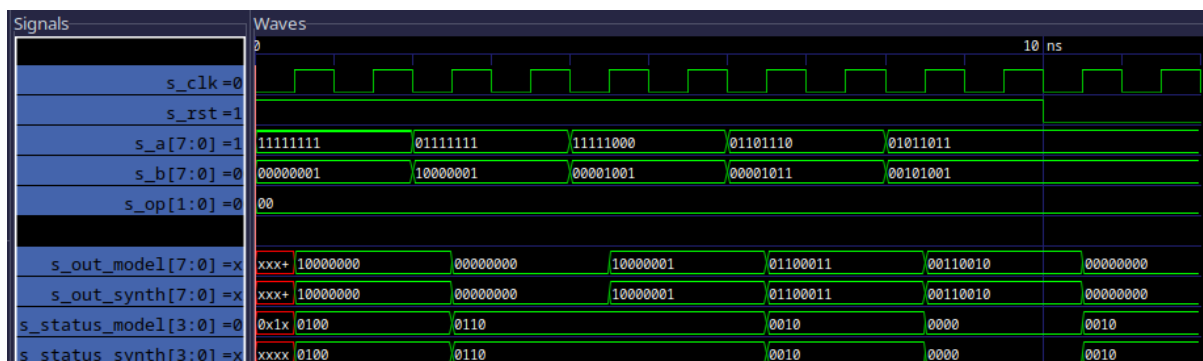
Z powodu braku możliwości przeprowadzenia testów dla wszystkich możliwych przypadków wybrano kilka skrajnych przypadków testowych (np. zmieniających daną flagę). W przypadkach testowych literą X oznaczono nieznaczone bity, czyli o dowolnej wartości. Testy podzielono na scenariusze dla każdej operacji.

Odejmowanie

Przypadki testowe

A	B	Oczekiwane wyjście	Flagi	Cel testu
-127	1	X000 0000	01X0	Flaga OVF
127	-1	X000 0000	01X0	Flaga OVF
-120	-9	1110 1111	1000	Flaga SINGLE
110	11	1100 0011	0010	Flaga EVEN
91	41	0011 0010	0000	Obliczenie próbne
91	41	0000 0000	0010	Reset

Przebieg czasowy



Przebieg czasowy z testów odejmowania

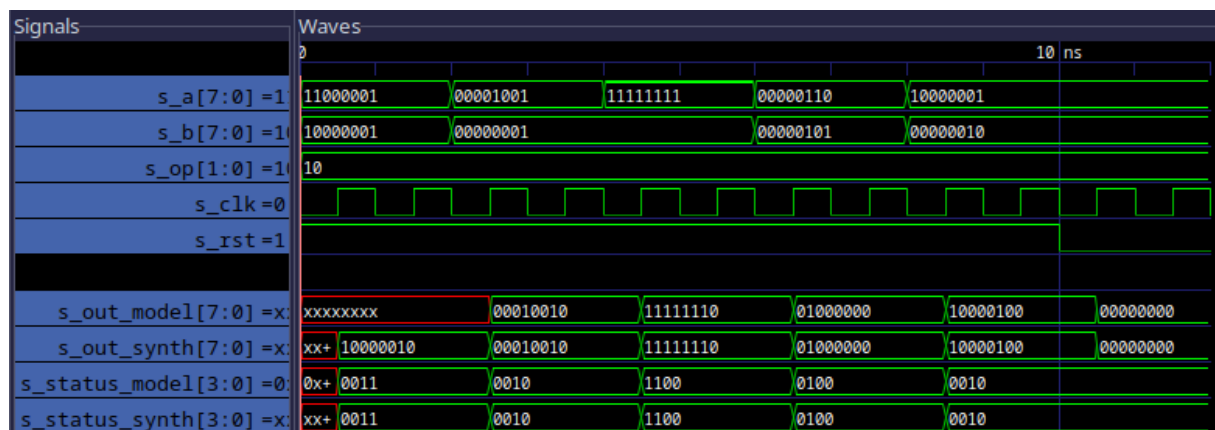
Pokrycie scenariusza testowego dla odejmowania: 100%

Przesuwanie bitów

Przypadki testowe

A	B	Oczekiwane wyjście	Flagi	Cel testu
1100 0001	1000 0001	XXXX XXXX	X0X1	Flaga ERROR
0000 1001	0000 0001	0001 0010	0010	Obliczenie próbne
1111 1111	0000 0001	1111 1110	1100	Flaga SINGLE i OVF
0000 0110	0000 0101	0100 0000	0100	Flaga OVF
1000 0001	0000 0010	1000 0100	0010	Zachowanie znaku
1000 0001	0000 0010	0000 0000	0010	Reset

Przebieg czasowy



Przebieg czasowy z testów przesuwania bitów

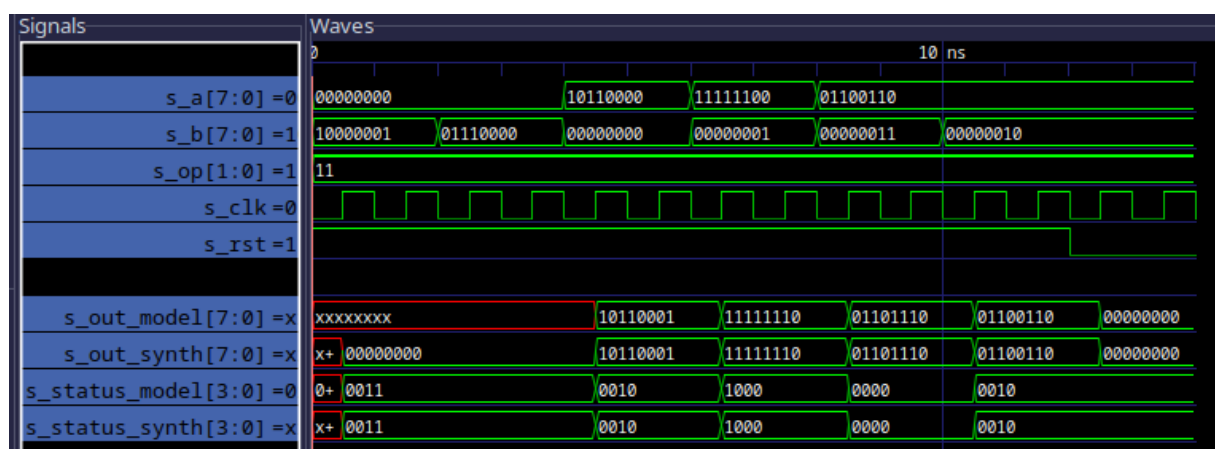
Pokrycie scenariusza testowego dla przesuwania bitów: 100%

Zamiana bitu

Przypadki testowe

A	B	Oczekiwane wyjście	Flagi	Cel testu
0000 0000	1000 0001	XXXX XXXX	X0X1	Flaga ERROR
0000 0000	0111 0000	XXXX XXXX	X0X1	Flaga ERROR
1011 0000	0000 0000	1011 0001	0010	Flaga EVEN
1111 1100	0000 0001	1111 1110	1000	Flaga SINGLE
0110 0110	0000 0011	0110 1110	0000	Obliczenie próbne
0110 0110	0000 0010	0110 0110	0010	Obliczenie próbne
0110 0110	0000 0010	0000 0000	0000	Reset

Przebieg czasowy



Przebieg czasowy z testów zmiany bitu

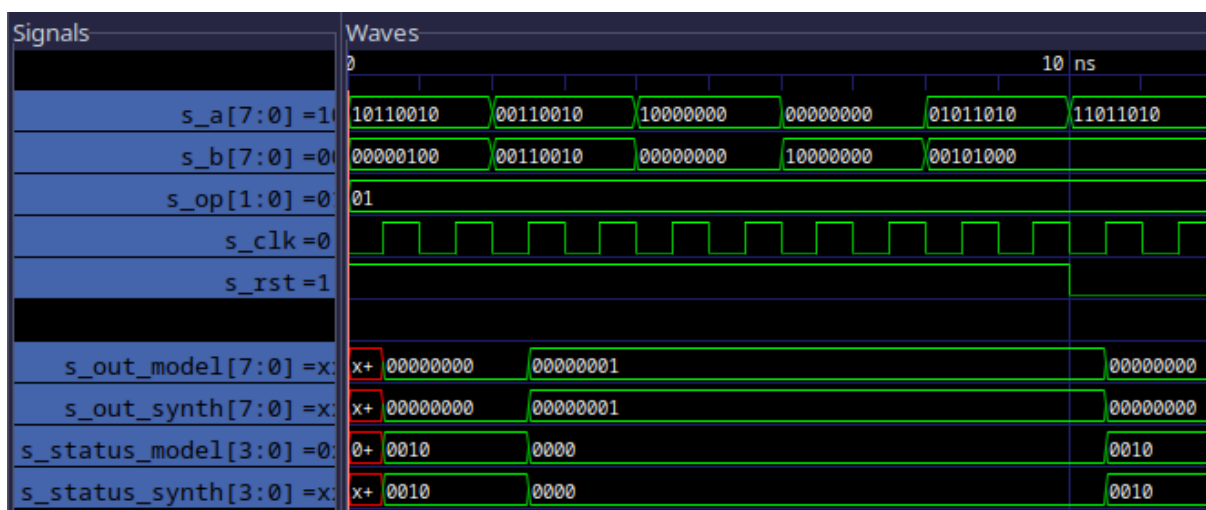
Pokrycie scenariusza testowego dla zmiany bitu: 100%

Porównywanie

Przypadki testowe

A	B	Oczekiwane wyjście	Flagi	Cel testu
-50	4	XXXX XXX0	0010	Flaga EVEN
50	50	XXXX XXX1	0000	Warunek gdy A = B
+0	-0	XXXX XXX1	0000	Wynik dla różnych reprezentacji zera
-0	+0	XXXX XXX1	0000	Wynik dla różnych reprezentacji zera
90	40	XXXX XXX1	0000	Obliczenie próbne
-90	40	XXXX XXX0	0010	Reset

Przebieg czasowy



Przebieg czasowy z testów porównywania

Pokrycie scenariusza testowego dla porównywania: 100%

Komentarz

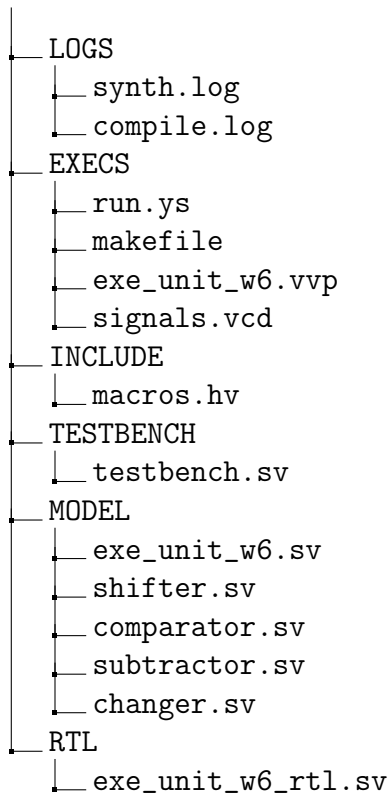
Wszystkie testy przebiegły zgodnie z przewidywaniami założonymi podczas projektowania. Co więcej na przebiegach czasowych wszystkich testów możemy zauważyć że wyjścia z modelu przed synteza i modelu zsyntetyzowanego są identyczne (poza nieokreślonościami, których jednoznacznie nie da się przedstawić po syntezie). Oznacza to, że proces syntezy odbył się poprawnie.

Struktura plików

Wszystkie pliki modelu i syntezy są umieszczone w katalogu **Pliki projektu**. Wewnątrz katalogu utworzono podkatalogi zgodnie z przeznaczeniem plików w nich przechowywanych:

- **LOGS** - Raport z syntezy i wyjście ewentualnych błędów z kompilatora.
- **EXECS** - Pliki odpowiedzialne za wykonanie syntezy, zarządzaniem procesem kompilacji, plik wynikowy do symulacji i plik z zapisanymi sygnałami wewnętrznymi z modułu testowego.
- **INCLUDE** - Plik nagłówkowy zawierający kilka makr usprawniających pracę.
- **MODEL** - Pliki z kodem modelu przed syntezą logiczną.
- **TESTBENCH** - Moduł testowy, wykorzystywany w symulacji.
- **RTL** - Plik z kodem jednostki po syntezie logicznej.

Wszystkie niezbędne pliki **.sv** znajdują się w katalogach **MODEL**, **TESTBENCH** i **RTL**. Pliki modelu zostały nazwane tak samo jak moduły w nich przechowywane.



Struktura katalogów projektu