Systemy cyfrowe i komputerowe

-

Projekt indywidualny - sync_arith_unit_4

Adam Szajgin s319821

Politechnika Warszawska

11 grudnia 2023

1. Opis modułu

1. Moduł 'sync_arith_unit_4' reprezentuje symulacyjną jednostkę arytmetyczno-logiczną (ALU), synchronizowaną z sygnałem zegarowym. Jednostka zdolna jest do wykonania czterech różnych operacji na dwóch wektorach wejściowych A i B. Zaimplementowane ALU działa w kodzie U2, natomiast zmiany zachodzące w układzie następują na zboczu narastającym zegara.

2. Porty układu:

   W skład portów układu wchodzą:

Nazwa	WE/WY	Funkcja portu
i_op	WE	n-bitowe wejście określające kod operacji
i_arg_A	WE	m-bitowe wejście wektora binarnego A
i_arg_B	WE	m-bitowe wejście wektora binarnego B
i_clk	WE	Wejście sygnału zegarowego
i_reset	WE	Wejście resetu synchronicznego wyzwalanego stanem niskim
o_result	WY	Wyjście synchroniczne zwracające wynik operacji na wektorach wejściowych
o_status	WY	4-bitowe wyjście synchroniczne dostarczające informacje o statusie wyniku operacji

i_reset:
Odpowiada za zerowanie wszystkich rejestrów wyjściowych Proces resetowania jest zsynchronizowany z sygnałem zegarowym. Aktywacja tego wejścia występuje w stanie niskim.

Natomiast port **o_status** składa się z bitów mających poniższe znaczenie:

Nazwa	Znaczenie bitu
bit ERROR	Sygnalizacja o tym, iż wynik został określony niepoprawnie
bit EVEN_1	Sygnalizuje parzystą liczbę jedynek w wyniku. Ustawiany na 0, gdy jest sygnalizowany błąd operacji
bit ONES	Sygnalizuje, że wszystkie bity wyniku są ustawione na 1. Ustawiany na 0, gdy jest sygnalizowany błąd
bit OVERFLOW	Sygnalizuje, że nastąpiło przepełnienie i wynik operacji wykracza poza szerokość wektora wejściowego

3. Parametry układu

   By zachować elastyczność i uniwersalność kodu, do realizacji układu użyte zostały parametry:

   - N = 2, do określenia wielkości wejścia określającego kod operacji
   - M = 4, do określenia wielkości argumentów A oraz B

4. Uwagi do opisu modułu

2. Lista realizowanych operacji

Realizowane operacje:

Na cztery operacje wykonywalne przez jednostkę 'sync_arith_unit_4' składają się:

- $A - 2 * B$, dla **i_op = 1'b00**
Odejmowanie dwukrotności liczby B od A

- $A < B$, dla **i_op = 1'b01**
Sprawdzenie czy liczba A jest mniejsza od liczby B. Gdy warunek jest spełniony, układ ma wystawić na wyjściu liczbę większą od zera, w przeciwnym wypadku ma to być liczba 0.

- $(A + B)[B] = 0$, dla **i_op = 1'b10**
Wynikiem operacji ma być liczba będąca sumą A i B z bitem o index B ustawionym na wartość 0. Jeżeli liczba B jest mniejsza od zera lub większa od szerokości wektora A, układ ma zgłaszać błąd.

- $U2(A) \Rightarrow ZM(A)$, dla **i_op = 1'b11**
Zamiana liczby A zapisanej w kodzie U2 na zapis w kodzie ZM. Jeżeli nie można dokonać poprawnej konwersji, należy zgłosić błąd, a wyjście układu powinno pozostać nieokreślone.

3. Schemat blokoway realizowanego modułu

4. Synteza logiczna

Do syntezy mojej jednostki arytmetyczno logicznej używałem yosys'a. Pobrałem program z oficjalnego GitHuba projektu Yosys i używałem zgodnie z dokumentacją. By przeprowadzić popraną syntezę odpalałem Yosys'a z dowolnego miejsca w drzewie plików (gdyż dodałem wczesniej program do PATH'a) . Do tego użyłem w kolejności komend:

- Wczytanie pliku do oprogramowania:

$read_verilog -sv alu.sv

- Skopiowanie jednostki sync_4 do sync_4_synthesis:

$copy sync_arith_unit_4 sync_arith_unit_4_synthesis

- Ustalenie hierarchii, gdzie jednostka sync_4_synthesis jest najwyższym poziomem:

$hierarchy -top sync_arith_unit_4_synthesis

- Synteza projektu:

$synth

- Optymalizacja i czyszczenie projektu:

$opt_clean

- Zapisanie projektu w formie pliku Verilog bez atrybutów:

$write_verilog -noattr alu_synthesis.sv

- Wyświetlenie statystyk dotyczących projektu:

$stat

W wyniku powyższej syntezy uzyskiwałem wiele linijek raportu. Jednak załączanie całego wydruku tutaj, zajęłoby ok. 10 stron, dlatego też załączam jedynie wydruk statystyk po syntezie mojej jednostki ALU:

```
7. Printing statistics.

=== sync_arith_unit_4_synthesis ===

   Number of wires:                197
   Number of wire bits:            220
   Number of public wires:          10
   Number of public wire bits:      33
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                201
     $_ANDNOT_                      42
     $_AND_                          5
     $_AOI3_                        17
     $_AOI4_                         1
     $_DFF_P_                        7
     $_MUX_                         11
     $_NAND_                         8
     $_NOR_                         12
     $_NOT_                         14
     $_OAI3_                        14
     $_OAI4_                         2
     $_ORNOT_                       16
     $_OR_                          26
     $_XNOR_                         9
     $_XOR_                         17
```

Rys. 1. Wydruk statystyk jednostki po syntezie

5. Przeprowadzone symulacje - testy

By skompliować plik testbench.sv używałem komendy: $iverilog -g2005-sv testbench.sv -o test_compiled. Następnie egzekwowałem skompilowany plik komendą: $vpp test_compiled. Na wskutek czego uzyskiwałem przebieg testbenchu który odpalałem w gtkwave komendą: gtkwave signals.vcd

1. Operacja pierwsza $A - 2 * B$

   W ramach testów tej operacji użyłem poniższych zestawień:

   - A = 3 (U2: 0011), B = 2 (U2: 0010)
   Oczekiwany wynik: 1111
   Oczekiwany status: 0110

   - A = -1 (U2: 1111), B = 3 (U2: 0011)
   Oczekiwany wynik: 1001
   Oczekiwany status: 0100

   - A = -3 (U2: 1101), B = 3 (U2: 0011)
   Oczekiwany wynik: WHATEVER (xxxx)
   Oczekiwany status: 1001

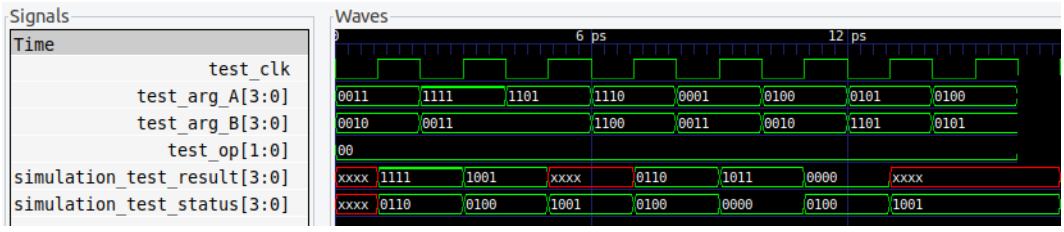
   - A = -2 (U2: 1110), B = -4 (U2: 1100)
   Oczekiwany wynik: 0110
   Oczekiwany status: 0100

   - A = 1 (U2: 0001), B = 3 (U2: 0011)
   Oczekiwany wynik: 1011
   Oczekiwany status: 0000

   - A = 4 (U2: 0100), B = 2 (U2: 0010)
   Oczekiwany wynik: 0000
   Oczekiwany status: 0100

   - A = 5 (U2: 0101), B = -3 (U2: 1101)
   Oczekiwany wynik: WHATEVER (xxxx)
   Oczekiwany status: 1001

   - A = 4 (U2: 0100), B = 5 (U2: 0101)
   Oczekiwany wynik: WHATEVER (xxxx)
   Oczekiwany status: 1001



Rys. 2. Wynik symulacji operacji nr. 1

2. Operacja druga $A < B$

   W ramach testów tej operacji użyłem poniższych zestawień:

   - A = 3, B = 5
   Oczekiwany wynik: 1
   Oczekiwany status: 0000

   - A = 7, B = 4

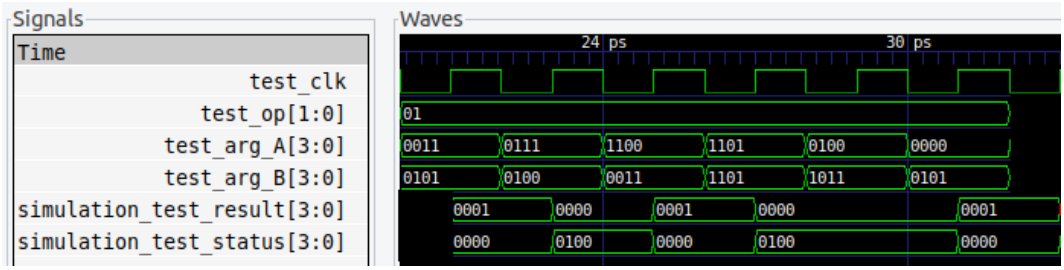
Oczekiwany wynik: 0
Oczekiwany status: 0100

- A = -4, B = 3
Oczekiwany wynik: 1
Oczekiwany status: 0000

- A = -3, B = -3
Oczekiwany wynik: 0
Oczekiwany status: 0100

- A = 4, B = -5
Oczekiwany wynik: 0
Oczekiwany status: 0100

- A = 0, B = 5
Oczekiwany wynik: 1
Oczekiwany status: 0000



Rys. 3. Wynik symulacji operacji nr. 2

Jak widać, dla każdej z testowanych operacji, wynik pokrywa się z oczekiwanym. Zarówno o_result jak i o_status.

3. Operacja trzecia $(A + B)[B] = 0$

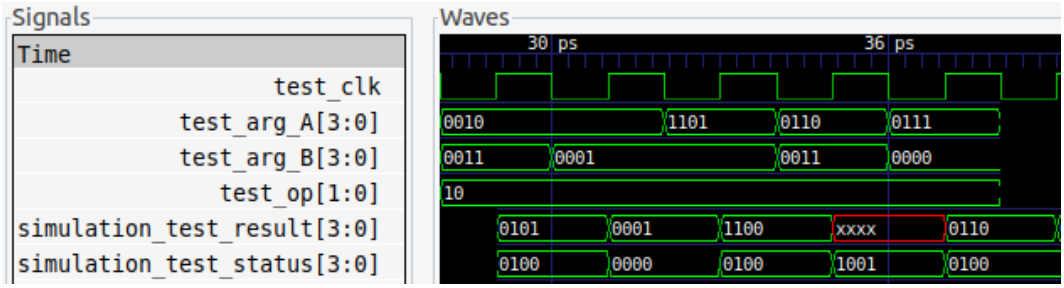
- A = 2 (U2: 0010), B = 3 (U2: 0011)
Oczekiwany wynik: 0101
Oczekiwany status: 0100

- A = 2 (U2: 0010), B = 1 (U2: 0001)
Oczekiwany wynik: 0001
Oczekiwany status: 0000

- A = -3 (U2: 1101), B = 1 (U2: 0001)
Oczekiwany wynik: 1100
Oczekiwany status: 0000

- A = 6 (U2: 0110), B = 3 (U2: 0011)
Oczekiwany wynik: WHATEVER (xxxx)
Oczekiwany status: 1001

- A = 7 (U2: 0111), B = 0 (U2: 0000)
Oczekiwany wynik: 0110
Oczekiwany status: 0100



Rys. 4. Wynik symulacji operacji nr. 3

4. Operacja czwarta $U2(A) \Rightarrow ZM(A)$

   - A = 3 (U2: 0011)
   Oczekiwany wynik: 0011
   Oczekiwany status: 0100

   - A = 7 (U2: 0111)
   Oczekiwany wynik: 0111
   Oczekiwany status: 0000

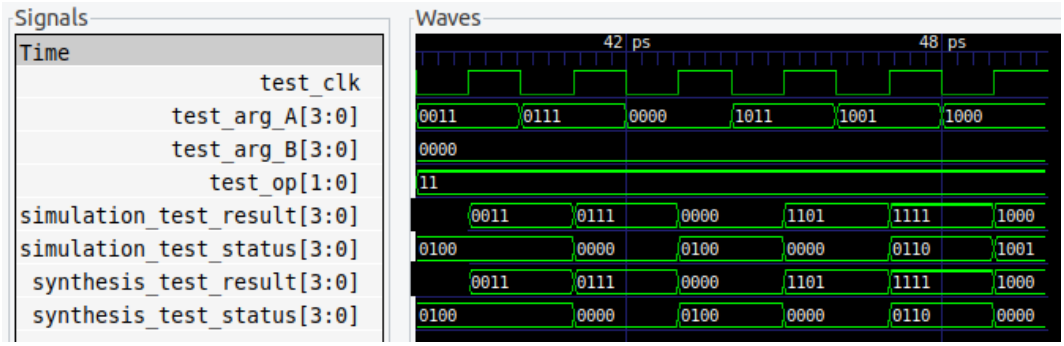
   - A = 0 (U2: 0000)
   Oczekiwany wynik: 0000
   Oczekiwany status: 0100

   - A = -5 (U2: 1011)
   Oczekiwany wynik: 1101
   Oczekiwany status: 0000

   - A = -7 (U2: 1001)
   Oczekiwany wynik: 1111
   Oczekiwany status: 0110

   - A = -8 (U2: 1000)
   Oczekiwany wynik: 1000
   Oczekiwany status: 1001

   Uwaga: W ZM nie występuje -8 (w zakresie 4 bitów), natomiast mamy dwa "0". Wynik będzie liczony dalej tak samo, jednak będzie nie prawidłowy, dlatego flaga statusu będzie 1001.



Rys. 5. Wynik symulacji operacji nr. 4

Jak widać, dla każdej z testowanych operacji, wynik pokrywa się z oczekiwanym. Zarówno o_result jak i

o_status.