

Termin zajęć	Godzina	Imię i Nazwisko	Nr albumu
Czwartek TP	10:45-13:00	<b>Łukasz Sztuka</b>	<b>243168</b>

## TEMAT: JEDNOSTKA ARYTMETYCZNO-LOGICZNA 8BIT

### 1. Specyfikacja projektu

Jak sama nazwa wskazuje układ wykonuje działania arytmetyczne i logiczne na dwóch ośmiobitowych słowach wejściowych. Jednostka jest w stanie realizować jedenaście różnych operacji. Wybór operacji dokonuje się poprzez podanie czterobitowego słowa na wejście "op". Wynik otrzymujemy na wyjściu "u\_result" w postaci szesnastobitowego słowa.

Należy sprawdzić poprawność działania wszystkich dostępnych operacji arytmetycznych i logicznych. Sprawdzenie wszystkich możliwych kombinacji bitów słów wejściowych ze względu na bardzo dużą ilość jest nie efektywne, dlatego wystarczającym będzie wybranie kilku sekwencji (np. najmniejszej, największej oraz losowej).

### 2. Protokół weryfikacji systemu

a. opis oznaczeń, modułów, bloków funkcjonalnych, itp.

Zaprojektowany układ testujący posiada funkcję odczytu danych wejściowych z pliku o nazwie data\_in.txt. W pliku tym znajdują się dane podawane na wejścia układu testowanego. Algorytm testujący został napisany w taki sposób aby co takt zegara odczytywał kolejny wiersz pliku. Struktura pliku wejściowego przedstawiona została w Tab. 1. Kolejne słowa wejściowe zostały odseparowane przy użyciu spacji.

	data in			
Opis	Słowo wejściowe A	Słowo wejściowe B	Słowo wyboru operacji	Bit restartu
Długość	8 bitów	8 bitów	4 bity	1 bit
Wejście	u a	u b	op	nreset

*Tab. 1. Struktura pliku data\_in*

```

00000000 00000000 0000 1
01010101 11001100 0000 1
11111111 11111111 0000 1
00000000 00000000 0001 1
01010101 11001100 0001 1
11111111 11111111 0001 1
00000000 00000000 0010 1

```

*Rys. 1. Przykładowy wygląd zawartości pliku wejściowego*

W celu uproszczenia kodu algorytmu testującego zdecydowałem się dodać zmienne pomocnicze, przechowujące wartości wejściowe i wyjściowe. Ze względu, że wynik na wyjściu pojawiał się jeden cykl zegarowy później niż słowa wejściowe konieczne było ich zsynchronizowanie. Lista wszystkich zmiennych znajduje się w Tab. 2.

Typ zmiennej	Nazwa Zmiennej	Opis
integer	plik_we	Przechowuje plik wejściowy data_in
integer	plik_wy	Przechowuje plik wyjściowy Raport
integer	pop	Poprawność wykonanej operacji 1- poprawnie 0-nie poprawnie
integer	a2	Słowo wejściowe A opóźnione o pół taktu zegara
integer	b2	Słowo wejściowe B opóźnione o pół taktu zegara
integer	a3	Słowo wejściowe A opóźnione o takt zegara
integer	b3	Słowo wejściowe B opóźnione o takt zegara
integer	a4	Dodatkowa zmienna używana w operacji NOT
integer	res3	Wynik w formacie pozwalającym na porównanie
integer	test	Wartość obliczona przez jednostkę testującą

Tab. 2. Lista zmiennych pomocniczych

```





















94 | always @(negedge clk) //wczytywanie linijki z pliku co zbocze opadające zegara
95 | begin
96 |     $fscanf(plik_we,"%b %b %b %b\n", u_a, u_b, op, nreset);
97 |     a3 = a2; // Przepisanie wartości w celu synchronizacji
98 |     b3 = b2;
99 |     op3 = op2;
100 | end
101 |
102 | always @(posedge clk) //co zbocze rosnące zegara
103 | begin
104 |     a2 = u_a;
105 |     b2 = u_b;
106 |     op2 = op;
107 | end
108 |
109 | always @(posedge clk2 && clk == 0) //co zbocze rosnące zegara2 i 0 na lini zegara
110 | begin
111 |     res3 = u_result;
112 | end

```

Rys.2. Funkcje synchronizujące

Testbench posiada również funkcję wykrywającą błędów wczytania plików. Jeśli plik lub pliki nie zostaną wczytane w konsoli Tcl zostanie wyświetlony odpowiedni komunikat.

```

69    :      if (plik_we==0)      // Test poprawności otworzenia plików
70         begin
71             $display("Błąd pliku wejściowego!");
72             $finish;
73         end
74         if (plik_wy==0)
75         begin
76             $display("Błąd pliku wyjściowego!");
77             $finish;
78         end

```

Rys. 3. Funkcja testująca wczytanie plików

Po wykonaniu testu zostaje wygenerowany plik tekstowy Raport.txt zawierający wszystkie najważniejsze dane oraz ocenę poprawności. Struktura wygenerowanego pliku została przedstawiona na Rys. 4.

```

Test monzenie
We: 00000000000000000000000000000000, 00000000000000000000000000000000; Wy: 00000000000000000000000000000000; Poprawność: 00000000000000000000000000000001
Test monzenie
We: 000000000000000000000000000000001010101, 0000000000000000000000000000000011001100; Wy: 00000000000000000000000000000000100001110111100; Poprawność: 00000000000000000000000000000001
Test monzenie
We: 0000000000000000000000000000000011111111, 0000000000000000000000000000000011111111; Wy: 00000000000000000000000000000000111111100000001; Poprawność: 00000000000000000000000000000001
Operator niepoprawny
Operator niepoprawny
Operator niepoprawny
Test dodawanie
We: 0000000000000000000000000000000001, 0000000000000000000000000000000001; Wy: 00000000000000000000000000000000010; Poprawność: 00000000000000000000000000000001
Test dodawanie
We: 0000000000000000000000000000000001010101, 00000000000000000000000000000000011001100; Wy: 00000000000000000000000000000000010010001; Poprawność: 00000000000000000000000000000001
Test dodawanie
We: 00000000000000000000000000000000011111111, 00000000000000000000000000000000011111111; Wy: 0000000000000000000000000000000000000000; Poprawność: 00000000000000000000000000000000
RESET WŁĄCZONY
Test dodawanie
We: 0000000000000000000000000000000001, 0000000000000000000000000000000001; Wy: 0000000000000000000000000000000000000000; Poprawność: 00000000000000000000000000000000
RESET WŁĄCZONY
Test dodawanie
We: 0000000000000000000000000000000001010101, 00000000000000000000000000000000011001100; Wy: 0000000000000000000000000000000000000000; Poprawność: 00000000000000000000000000000000

```

Rys. 4. Struktura pliku wyjściowego Raport

W pierwszym wersie wyświetlony zostaje rodzaj wykonywanej operacji. Następnie wypisane zostają dane wejściowe, wyjściowe oraz ocena poprawności. W przypadku podania nieprawidłowych danych na wejście ustalające przeprowadzaną operację, zostaje wyświetlony komunikat „Operator niepoprawny”. Jeśli w trakcie działania układu uruchomiony zostanie Reset, również wyświetli się odpowiedni komunikat „RESET WŁĄCZONY” należy wówczas zignorować ocenę poprawności. Kompletny plik Raport.txt znajduje się w załączniku i zawiera test wszystkich możliwych trybów działania układu.

- b. zastosowany algorytm testowania poszczególnych modułów/bloków (opis, diagram blokowy)

Aby zweryfikować poprawność poszczególnych operacji wykonywanych przez testowaną jednostkę, napisałem algorytm realizujący bliźniacze funkcje i porównujący uzyskane wyniki. Algorytm rozpoznaje wykonywaną operację poprzez porównanie wartości wpisanej do zmiennej pomocniczej op3 oraz predefiniowanej wartości zgodnej ze specyfikacją testowanej jednostki (dla operacji logicznej AND jest to 0). Przykładowy blok testujący widoczny na Rys. 5.

```

117 ○ //Rozpoznanie wykonywanej operacji
118 ○
119 ○ begin
120 ○   $fdisplay(plik_wy, "Test and");
121 ○   test = a3 & b3;
122 ○   if (res3 == test) // Porównanie wartości obliczonej przez układ testowany i układ testujący
123 ○     begin
124 ○       pop = 1;
125 ○     end
126 ○   else begin
127 ○     pop = 0;
128 ○   end
129 ○   $fdisplay(plik_wy, "We: %b, %b; Wy: %b; Poprawność: %b", a3, b3, res3, pop);
130 ○ end

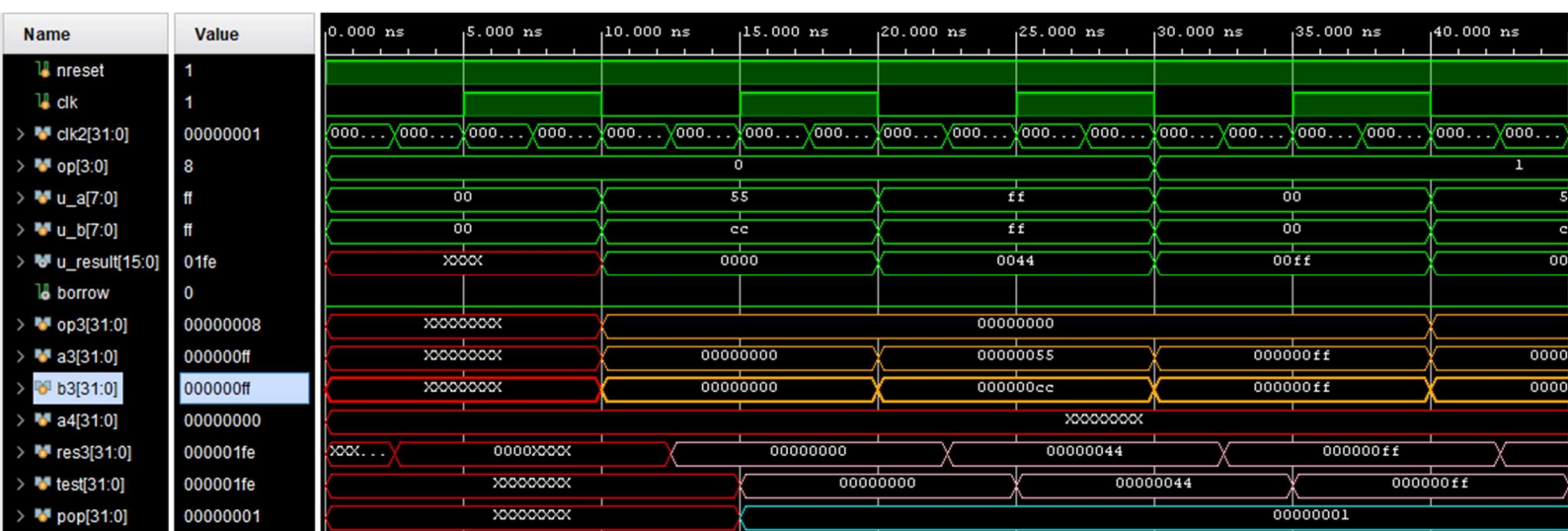
```

Rys. 5. Kod odpowiadający za test operacji logicznej AND

Jeśli wartości uzyskane przez układ testowany zapisane w zmiennej res3 i wartości uzyskane przez jednostkę testującą zapisane w zmiennej „test” są takie same wartość zmiennej „pop” zostaje ustawiona na 1.

Kolejne bloki kodu testującego zostały napisane w sposób analogiczny i można znaleźć je w załączniku.

- c. przykładowe przebiegi czasowe - istotne fragmenty wykazujące prawidłowe/nieprawidłowe funkcjonowanie elementów systemu, jego ograniczenia, prezentacja elementów implementacji wymagających zmiany

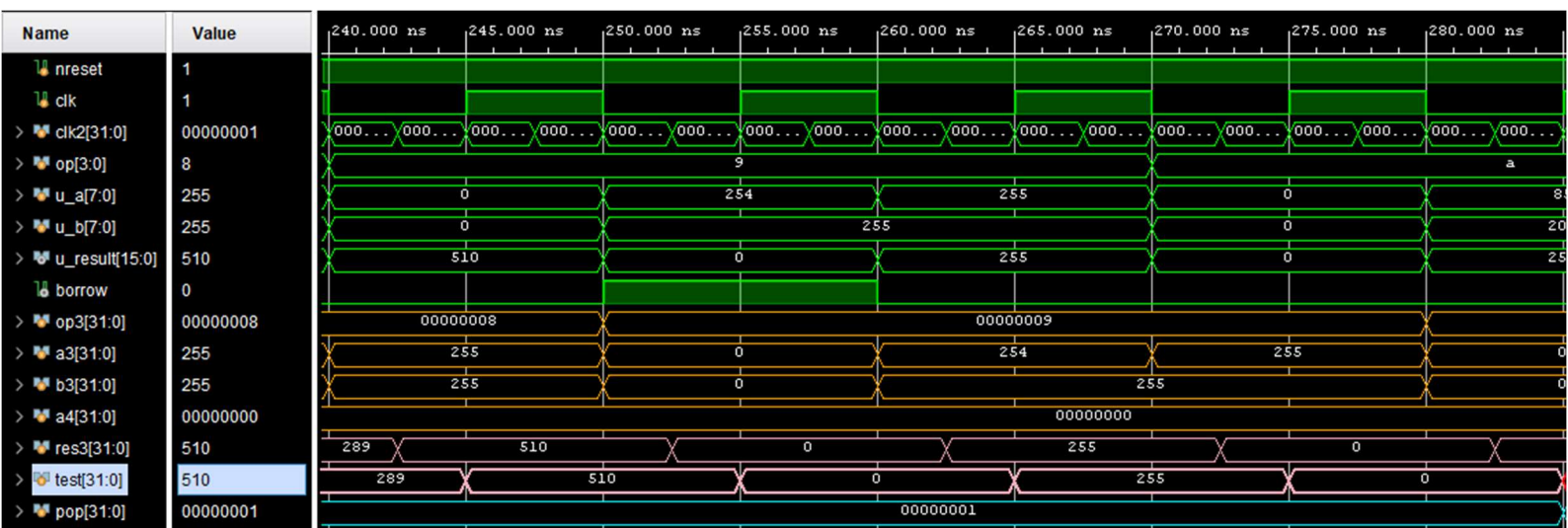


Rys. 6. Przebiegi czasowe w pierwszych 45 ns po uruchomieniu symulacji



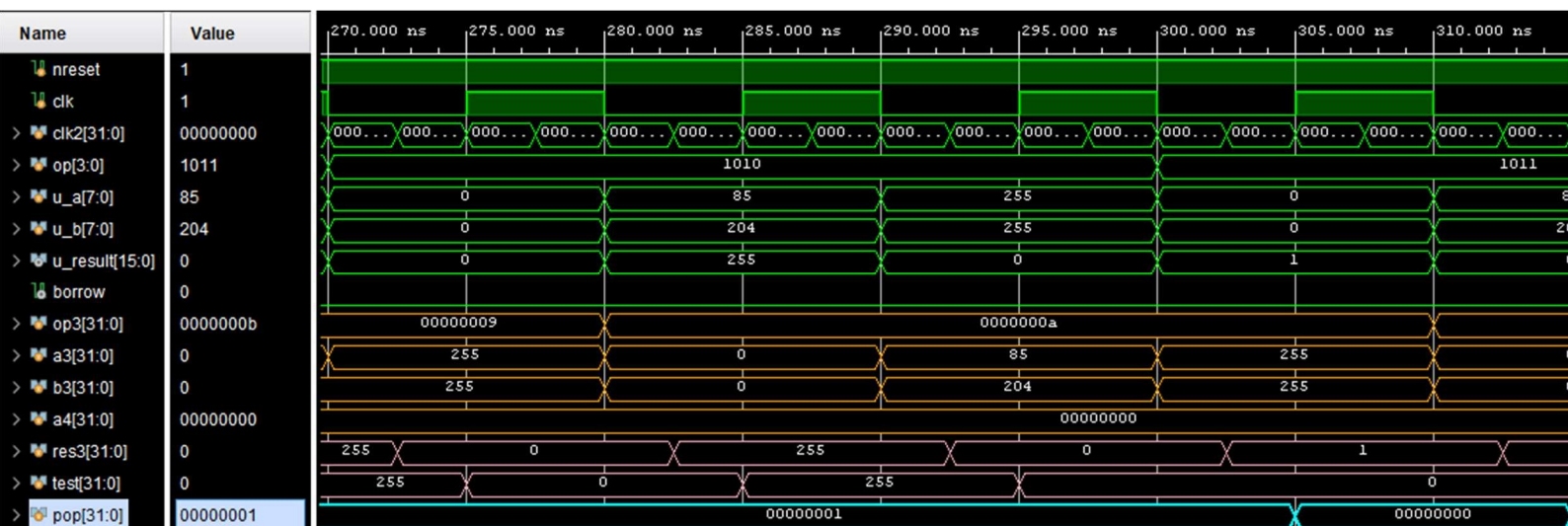
W pierwszym takcie zegara na zmiennych pomocniczych oraz wyjściu układu pojawia się stan nieustalony (oznaczony literą „X”) spowodowane jest to nie przypisaniem wcześniej żadnej wartości. Następnie testowana jest operacja logiczna AND. Kolorem zielonym zaznaczone są przebiegi sygnałów wejściowych i wyjściowych oraz zegary. Kolorem pomarańczowym oznaczone są zmienne pomocnicze. Różowym oznaczone są zmienne przechowujące wyniki operacji. Natomiast kolorem turkusowym oznaczono zmienną sygnalizującą poprawność wykonanej operacji.

Na wykresie zaobserwować możemy opóźnienia sygnałów wyjściowych względem wejściowych oraz ich synchronizację w zmiennych pomocniczych.



Rys. 7. Operacja odejmowania z pożyczaniem

Szczególną uwagę należałoby zwrócić na realizację odejmowania liczby większej od mniejszej. Jako iż układ nie jest w stanie operować na liczbach ujemnych konieczne jest dokonanie pożyczki w tym przypadku liczby 256.



Rys. 8. Operacja dzielenia

Nieintuicyjne zachowanie możemy zaobserwować również w przypadku dzielenia. W przypadku dzielenia przez 0 układ zwraca wynik 255. Natomiast kiedy dzielimy liczbę mniejszą przez większą wynikiem powinien być ułamek, oczywiście układ nie oferuje takiego rozwiązania i zwraca 0 co zostało uwzględnione w jednostce testującej. Najdziwniejszym

jednak jest zachowanie układu podczas dzielenia 255 przez 255. Układ zwraca wówczas wartość 0 co jest wartością błędną. Przypadek ten potwierdza poprawne działanie układu testującego, który wyświetla 0 na zmiennej oceniającej poprawność działania.

d. analiza pokrycia funkcjonalnego przez środowisko testowe

Jak wcześniej wspomniałem pełne pokrycie wszystkich możliwych kombinacji słów wejściowych oraz operacji jest niemożliwe do przetestowania ze względu na ich bardzo dużą liczbę. Jednak wystarczające okazuje się użycie tylko trzech kombinacji wejściowych dla każdego z trybów działania układu. Układ testujący niestety nie wykrywa poprawności działania trybu resetu i wartości podawanej na wyjście pożyczki (borrow) i należałoby dokonać tego manualnie śledząc przebiegi w środowisku symulującym.

Załączniki:

- Kod implementacji systemu w języku VHDL (dodatkowo kod VHDL który został zmodyfikowany)
- Kod środowiska testowego systemu w języku Verilog opatrzone komentarzami
- Pliki monitorowania wygenerowane za pomocą środowiska testowego (jeśli utworzono)