Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej Nr 20

XIV Seminarium
ZASTOSOWANIE KOMPUTERÓW W NAUCE I TECHNICE' 2004
Oddział Gdański PTETiS

PROJEKTOWANIE UKŁADÓW CYFROWYCH Z WYKORZYSTANIEM PAKIETU MULTISIM 2001 ORAZ JĘZYKA OPISU SPRZĘTU VHDL

Krystyna NOGA[1], Łukasz ŻYLIŃSKI[2]
Akademia Morska w Gdyni, Katedra Automatyki Okrętowej
1. e-mail: jagat@am.gdynia.pl, 2. e-mail: raskol@pf.pl

W artykule zostanie przedstawiony krótki opis oprogramowania Multisim VHDL Education wchodzący w skład pakietu Multisim 2001 oraz przykłady jego wykorzystania do nauczania podstaw projektowania programowalnych układów logicznych opartych na języku VHDL. Multisim VHDL to zaawansowane narzędzie przeznaczone do nauki, projektowania i symulacji układów programowalnych (FPGA / CPLD) jak i innych cyfrowych urządzeń (procesory / pamięci). Posiada on szeroką gamę gotowych narzędzi pomagających stworzyć i edytować projekt. W pracy zostaną omówione możliwości budowy projektów w programie Multisim VHDL. Zostaną również przedstawione przykładowe symulacje.

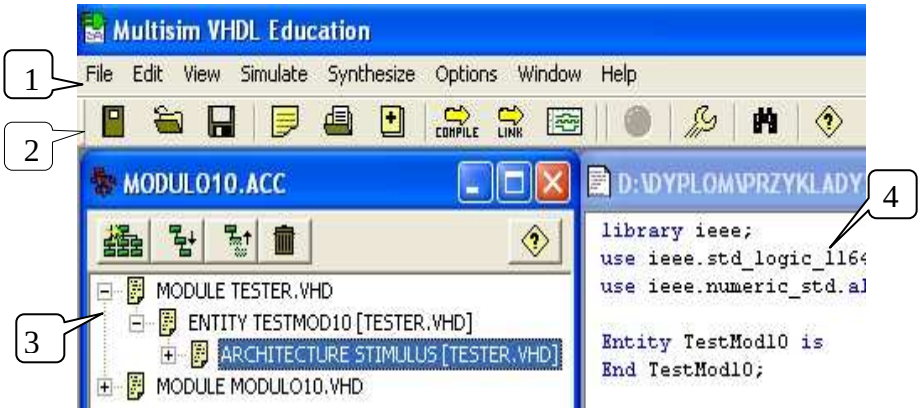
1. WSTĘP

Program Multisim VHDL Education wchodzi w skład pakietu Multisim 2001 firmy *Interactive Image Technologies* [2]. Multisim VHDL to zaawansowane narzędzie przeznaczone do nauki, projektowania i symulacji układów programowalnych (FPGA / CPLD) jak i innych cyfrowych urządzeń (procesory / pamięci). W skład programu wchodzą gotowe narzędzia wspomagające tworzenie i edytowanie projektu. Do najważniejszych z nich zaliczyć można symulator, edytor kodu źródłowego i kreatory *Test Bench Wizard* i *Design Wizard*. Zaletą stosowania języka VHDL jest możliwość tworzenia projektu bez konieczności uprzedniego wyboru układu, w którym będzie on realizowany [1, 3, 4]. W programie istnieje możliwość określenia normy języka VHDL (IEEE 1076, IEEE 1164), parametrów symulacji, czcionki oraz sposobu wyświetlania okien.

2. STRUKTURA PROGRAMU MULTISIM VHDL EDUCATION

W celu uruchomienia programu Multisim VHDL Education należy wybrać w programie Multisim 2001, z zakładki *Simulate*, polecenie *VHDL Simutation*, bądź z paska narzędziowego wybrać symbol układu scalonego. Po uruchomieniu programu na ekranie ukaże się menu główne (rys. 1), które zawiera pasek menu (1) i pasek narzędziowy (2). Po utworzeniu nowego projektu, bądź otwarciu istniejącego, dostępne jest okno modułów (3).

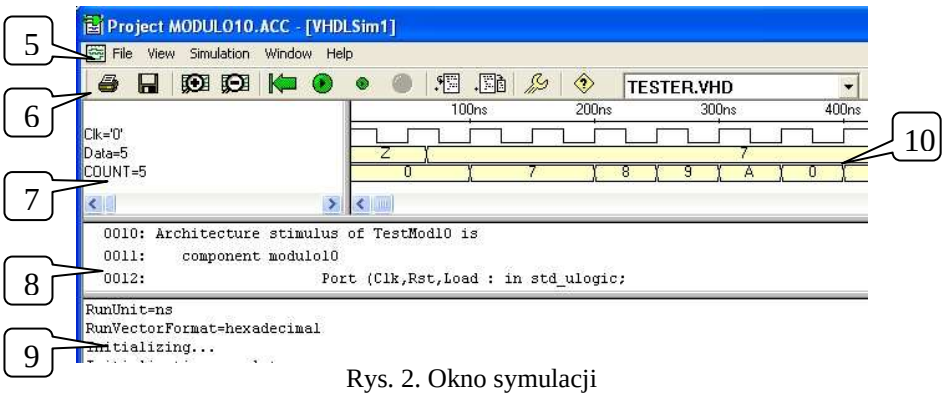
Po dwukrotnym naciśnięciu lewego przycisku myszki, na otwartym module, ukaże się pole edycyjne (4).



Rys. 1. Menu główne programu Multisim VHDL Education

Pasek menu zawiera następujące zakładki: *File, Edit, View, Simulate, Synthesize, Options, Window, Help*. Inne najczęściej używane zakładki, tzn. *Compile* oraz *New Module*, zostały zgrupowane na pasku narzędziowym. Podstawowy szkielet projektu tworzy się w oknie modułów, które zawiera moduł programu i moduł symulacji. Ponadto w polu edycyjnym można edytować wybrany kod źródłowy programu lub samodzielnie go napisać wykorzystując język VHDL. Dla użytkownika ułatwieniem jest rozróżnianie przez program słów kluczowych (np. *entity, port, end*) stosowanych w języku VHDL, są one wyróżnianie kolorem niebieskim. Pozostałe słowa wprowadzone przez użytkownika są identyfikatorami i są oznaczane kolorem czarnym. W celu umieszczenie komentarza wprowadza się dwie poziome kreski (--), a następnie treść, co zostanie wyróżnione na zielono. Cały tekst w takim wierszu, aż do końca, jest komentarzem.

Aby przeprowadzić symulację w oknie modułów zaznacza się, poprzez podświetlenie, jednostkę projektową (*entity*), która ma podlegać symulacji. W kolejnym kroku należy wybrać z zakładki *Simulate* polecenie *Load Selected*. W przypadku prawidłowej kompilacji pojawi się okno, w którym należy dokonać wyboru sygnałów i zmiennych. Dla nich zostaną przedstawione przebiegi czasowe. Po dokonaniu powyższych czynności i ich zatwierdzeniu na ekranie pokazuje się okno symulacji (rys. 2). Po wybraniu z paska menu (5 na rys. 2) zakładki *Simulation*, a następnie polecenia *Go*, bądź po wyborze ikony z czarnym trójkątem (w zielonym kole) na pasku narzędziowym (6), zostanie przeprowadzona symulacja. Na okno symulacji składa się wspomniany już pasek menu i pasek narzędziowy. Polecenia dostępne w pasku narzędziowym to między innymi powiększenie i zmniejszenie rysunków przedstawiających przebiegi czasowe, start i określenie kroku symulacji. W oknie sygnałów (7) można dodać lub usunąć sygnał lub zmienną. Przebiegi czasowe można zaobserwować w oknie (10). Okno to zawiera skalę czasową, ponadto pod przebiegami wyświetlane są wartości sygnałów. Ułatwia to weryfikowanie projektu i sprawia, że wszystko dla projektanta jest czytelne. Efekty symulacji można obserwować w polu (8), a proces kompilacji kodu w oknie (9). Gdy podczas próby kompilacji wystąpią błędy pojawia się komunikat, który informuję użytkownika o ilości błędów, jak również o miejscu ich wystąpienia (podany zostaje numer błędnej linijki w kodzie programu).



Rys. 2. Okno symulacji

3. ZASADY TWORZENIA PROJEKTÓW

Multisim VHDL Education oferuję gotowe narzędzia do stworzenia projektu. Przy każdej próbie utworzenia nowego projektu program daje możliwość wyboru. Użytkownik musi podjąć decyzję, czy chce korzystać z kreatora czy tylko chce utworzyć puste okno modułu (*Create Blank Module*). Dostępne dla użytkownika kreatory to *Module Wizard* i *Test Bench Wizard*.

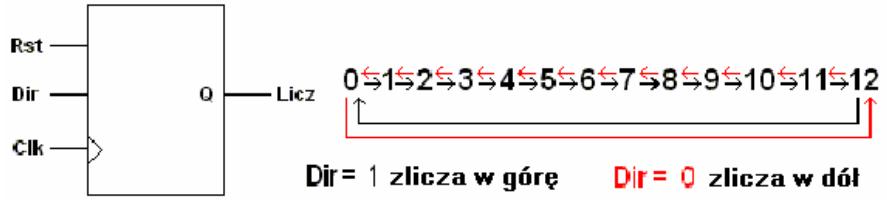
Po wybraniu *Module Wizard* pojawia się pytanie o podstawowe informacje, dzięki którym stworzony zostanie projekt. W polu *Entity name* należy wpisać identyfikator jednostki projektowej, a w polu *Architecture name* nazwę trzonu architektury. Ponadto w *Port Name* należy wpisać sygnały wejścia / wyjścia wykorzystywane w projekcie. Tryb portu (*Mode*) wybiera się z rozwijanego menu. Dostępnych jest pięć typów: in, out, inout, buffer, linkage. Oprócz podania nazwy portu i trybu konieczne jest również zadeklarowanie typu danych. W przypadku wyboru standardu IEEE dostępne są typy: std_logic, std_ulogic, std_logic_-vector i std_ulogic_vector. Wyboru dokonuje się zaznaczając opcję *Use IEEE 1164 standard logic* w oknie kreatora. W przypadku nie zaznaczenia tej opcji dostępne będą typy: boolean, bit, bit_vector i integer. Po wypełnieniu wszystkich pól i naciśnięciu przycisku *Add Port* zadeklarowany zostanie port. Aby stworzyć nowy moduł należy wybrać polecenie *Create*. Po zapisaniu utworzony zostaje plik z rozszerzeniem vhd, który widoczny jest w oknie modułów. W stworzonym kodzie programu należy zamienić tymczasowe sygnały *CLOCK* i *RESET*, utworzone przez kreator, na zadeklarowane wcześniej podczas tworzenia projektu (np. na Clk, Reset). Kolejnym krokiem niezbędnym do stworzenia projektu jest przeprowadzenie kompilacji, co uzyskujemy po wyborze zakładki *Simulate*, a następnie polecenia *Compile Selected*. Jeżeli nie wystąpią błędy w kodzie źródłowym, to należy przebudować składnię programu używając polecenia *File* / *Rebuild Hierarchy*. Utworzona zostanie w ten sposób podstawowa struktura programu.

Po przygotowaniu pliku z opisem programu należy przejść do kreatora *Test Bench Wizard*. Przy jego pomocy przygotowuje się kod źródłowy programu odpowiedzialny za przebieg symulacji. Utworzony wcześniej plik z programem widoczny jest w oknie modułów. Następnie należy plik zaznaczyć, poprzez podświetlenie, i wybrać kreator *Test Bench Wizard*. Wówczas też zostanie otwarte nowe okno, w którym będą już uzupełnione pola *Entity name* i *Architecture name*. Zadaniem użytkownika jest uzupełnienie pola *Port declaration* zmiennymi, takimi samymi danymi jak w programie głównym. Po naciśnięciu przycisku *Create* pojawi się pytanie o miejsce zapisu i nazwę programu. Utworzony plik posiada

rozszerzenie vhd i jest również widoczny w oknie modułów. Ponadto w polu edycyjnym (4) należy określić cykl zegara (*Clock*) oraz parametry symulacji (*Stimulus*). Po wykonaniu powyższych czynności należy przejść do kompilacji i przebudowy składni programu. Tak przygotowany, za pomocą kreatorów, program nadaje się do przeprowadzenia symulacji.

4. PRZYKŁADOWE SYMULACJE UKŁADÓW CYFROWYCH

Jako przykład symulacji zaprezentowany zostanie licznik rewersyjny modulo 13 [5]. Przy pomocy sygnału podanego na wejście *Dir* dokonuje się wyboru kierunku zliczania. Stan wysoki to zliczanie „w górę", stan niski „w dół". Licznik wyposażony jest również w wejście kasowania *Rst*, przy czym stanem aktywnym jest jedynka logiczna. Zawarta w kodzie programu zmienna *Q* wymaga zastosowania biblioteki *NUMERIC_STD*. Nie stosując tej biblioteki można się narazić na fałszywe przypisywanie wartości do zmiennej Q, ponieważ rozmiar tego sygnału oraz łańcucha, z którym jest porównywany, są różne. Przy tworzeniu projektu w Multisim VHDL za pomocą kreatorów domyślnie dopisywana jest do kodu wspomniana biblioteka *NUMERIC_STD*. Na rysunku 3 przedstawiono symbol graficzny licznika modulo 13 i odpowiadający mu graf przejść.

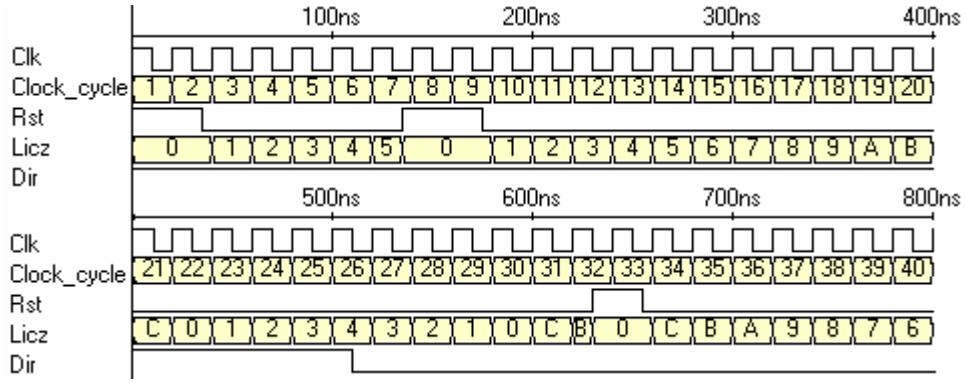


Rys. 3. Symbol graficzny licznika modulo 13 i jego graf przejść

Przebieg symulacji rozpoczyna się od określenia parametrów początkowych. Dla kierunku zliczania w „przód" na wejście *Dir* podajemy stan wysoki. Początkowo na wejście Rst podano jedynkę logiczną, licznik został więc wyzerowany. Następnie po 35 ns od początku pracy na wejście *Rst* podano stan niski, co zapoczątkowało zliczanie. Prawidłowość działania układu kasującego zaobserwowano między 135 ns a 175 ns. Zmiany kierunku liczenia dokonano w 335 ns procesu symulacji. Poniżej zamieszczono fragment kodu źródłowego dla kierunku zliczania „w górę" oraz opis pracy zegara. Przebiegi czasowe licznika zostały przedstawione na rysunku 4.

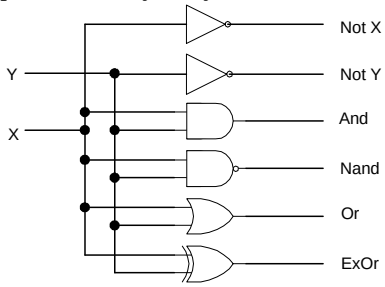
```
variable Q: unsigned (3 downto 0);                    -- zmienna Q odpowiedzialna za etapy licznika
   begin
      if Dir = '1' then                               -- kierunek zliczania "w gore"
      if Rst = '1' then Q:= "0000";                   -- kasowanie stanu licznika
            elsif rising_edge(Clk) then Q:=Q+"0001";  -- zmiany reagujące na zbocze rosnące
       if Q = 13 then Q:="0000";
      end if;
      end if;
      end if;

-- opis pracy zegara

Clock: process
   begin
   Clock_cycle <=Clock_cycle +1;
   Clk<='1';
```

```
  wait for 10 ns;
  Clk<='0';
      wait for 10 ns;
end process;
```



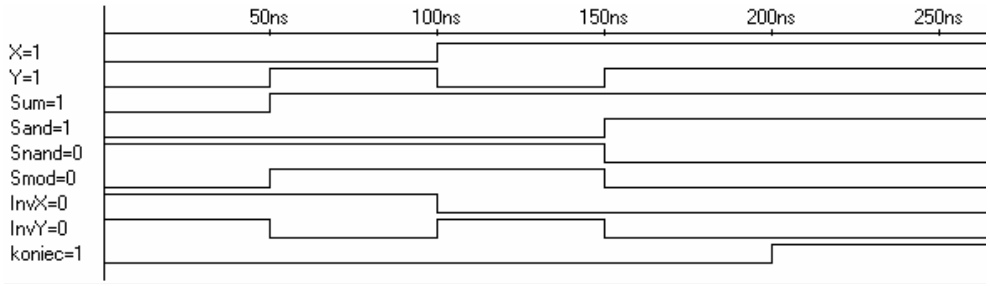
Rys. 4. Przebiegi czasowe licznika modulo 13

Za pomocą Multisim VHDL można zaimplementować różne układy cyfrowe. Proste komendy opisujące bramki logiczne ułatwiają budowę projektu. W zaprezentowanym, w dalszej części artykułu, projekcie przedstawiono pracę sześciu bramek (2 x NOT, AND, NAND, OR, ExOR) (rys.5). Opis podstawowych funkcji logicznych jest zgodny z językiem VHDL, a zastosowane zmienne są typu *BIT*. Przebieg symulacji wykonano dla wszystkich możliwych przypadków jakie mogą przybrać wejścia *X* i *Y*. Cykl zmian zmiennych wejściowych zdefiniowano na 50 ns, przyporządkowując zmiennej *PERIOD* zadeklarowany czas. Zakończenie procesu symulacji sygnalizowane jest stanem wysokim na wyjściu *koniec*. Przebieg symulacji został przedstawiony na rysunku 6.



Rys. 5. Schemat logiczny projektu bramek

```
architecture behavioral of Bramki is
begin
   -- opisy bramek
   InvX <= not X;            -- negacja X
   InvY <= not Y;            -- negacja Y
   Sand <= X and Y;          -- koniunkcja X i Y
   Snand <= X nand Y;        -- negacja koniunkcji X i Y
   Sum <= X or Y;            -- alternatywa X i Y
   Smod <= X xor Y;          -- suma modulo2 X i Y
end ;
```



Rys. 6. Przebiegi czasowe programu „Bramki”

5. PODSUMOWANIE

Przedstawiony program Multisim VHDL Education, wchodzący w skład pakietu Multsim 2001, jest odpowiednim narzędziem dla początkujących projektantów. Rozbudowane kreatory pozwalają stworzyć projekt od podstaw. Komentarze, umieszczone w programie źródłowym, ułatwiają analizę i korektę projektu. Czytelny edytor kodów wyposażony jest w funkcję rozróżniania (za pomocą kolorów) komend języka VHDL od pozostałych słów umieszczonych w programie. Okno z przebiegami symulacji, wyposażone w linijkę, ułatwia podgląd wartości zmiennych. Ponadto Multisim VHDL posiada wiele rozbudowanych funkcji, poczynając od zaawansowanych kreatorów do prostych opcji symulacyjnych. Pakiet posiada również wady, do których należy zaliczyć błąd przy tworzeniu projektu w kreatorach. Polega on na tym, że zamiast trybu portu proponowanego przez kreatora, umieszczany jest tryb inny niż żądany. Ponadto zmiana czasu przeprowadzanej aktualnie symulacji na mniejszy wymaga ponownego uruchomienia programu symulacyjnego, co sprowadza się do niepotrzebnego „restartu”. Kolejna wada dotyczy wydruku przebiegów dla standardowych parametrów, który rozciąga się aż do 6 stron formatu A4.

6. BIBLIOGRAFIA

1. Skahill K. – Język VHDL, projektowanie programowalnych układów cyfrowych, WN, Warszawa 2001
2. http://www.electronicsworkbench.com
3. Zwoliński M. – Projektowanie układów cyfrowych z wykorzystaniem języka VHDL, WKiŁ, Warszawa 2002
4. VHDL Design, Simulation and Debug, Multisim 2001 User Quide, dokumentacja Electronics Workbench
5. Żyliński Ł. – Projektowanie układów cyfrowych z wykorzystaniem pakietu Multisim 2001 oraz języka opisu sprzętu, przygotowywana praca dyplomowa, AM Gdynia

DESIGNING DIGITAL PROJECTS USING SOFTWARE MULTISIM 2001 AND HARDWARE DESCRIPTION LANGUAGE VHDL

In the paper software Multisim VHDL Education is presented. Multisim VHDL is an advanced software product intended to help and learn users to use VHDL for digital design projects. Multisim VHDL includes an integrated simulator, source file editor, hierarchy browser and other resources for VHDL users. In article are discussed possibilities of building projects using Multisim VHDL and example and simulations are presented.