MaxPlusII – Opis projektu za pomocą języka AHDL

Opis projektu za pomocą języka AHDL rozpoczyna się od wyboru w środowisku MaxPlusII edytora tekstowego z menu ***Max+PlusII*** funkcja ***Text Editor*** lub w menu ***File*** należy wybrać opcję ***New…*** wskazując **Text Editor File**. Po otwarciu okienka „Untitled1 – Text Editor" można rozpocząć opis projektu. Warto jest od razu nazwać ten plik wskazując rozszerzenie *.tdf dzięki czemu różne struktury językowe zaznaczane są różnymi kolorami co ułatwia tworzenie opisu projektu.

Podstawowe struktury języka

Przykładowy opis rewersyjnego licznika modulo 64.

```
TITLE "Licznik rewersyjny modulo 64 z zerowaniem i zapisem";
%------------------------------------------------------------
Zbior  : cnt_txt.tdf
Jezyk  : AHDL
Autor  : Marek Pawlowski
Wersja : 1
Data   : 2004.12.14
------------------------------------------------------------%
SUBDESIGN cnt_txt
(
   WE[5..0] : input;
   WY[5..0] : output;
         LD : input;
            CR : input;
            UD : input;
            CK : input;
)
VARIABLE
   WY[5..0] : DFF;
BEGIN
   WY[].clk  = CK;
   WY[].clrn = CR;
   if    (LD==0) then WY[]=WE[];
   elsif (UD==0) then WY[]=WY[]+1;
   else               WY[]=WY[]-1;
   end if;
END;
```

W pierwszym wierszu pliku **.tdf* występuje dyrektywa **TITLE** zawierającą nagłówek umieszczany w raporcie z kompilacji. Następujące po nim wiersze są komentarzem opisującym podstawowe dane projektu.

- **Komentarz** rozpoczyna znak i kończy znak '**%**'.
- **SUBDESIGN**

Dyrektywa **SUBDESIGN** z nazwą projektowane układu wyznacza zmienne wejściowe '**input**', wyjściowe '**output**' i dwukierunkowe '**bidir**' projektowanego układu. Nazwa projektowanego układu musi być identyczna jak nazwa zbioru **.tdf*, w którym została dyrektywa subdesign umieszczona.

Słowo kluczowe określające typ zmiennej poprzedza lista zmiennych oddzielonych przecinkami zakończona dwukropkiem. Deklaracje zmiennych we-wy umieszczane są pomiędzy nawiasami '**(**' i '**)**'. Zapis w BNF (ang. Backus-Naur Form) tej konstrukcji jest następujący:

```
SUBDESIGN nazwa_zbioru
(
   <lista zmiennych> : <typ zmiennej>;
   {<lista zmiennych> : <typ zmiennej>;}
)
```

gdzie

```
<lista zmiennych>   ::= <zmienna> {,<zmienna>}
  <typ zmiennej>      ::= INPUT | OUTPUT | BIDIR
```

dla przypomnienia oznaczeń notacji BNF:

`<....>`	–	zawiera identyfikator
`{....}`	–	opcjonalne powtórzenie obiektu (zero bądź dowolną liczbę razy)
`(....)`	–	grupowanie listy obiektów
`..|..|..`	–	wybór jednej z wymienionych możliwości
`n:m`	–	zakres obiektu
`::=`	–	definiowanie znaczenia lewej strony wyrażenia

Słowa kluczowe mogą być pisane dużymi bądź małymi literami.

- **VARIABLE**

Słowo kluczowe **VARIABLE** pozwala wskazać realizację zmiennych jako rejestrowe a także zdefiniować zmienne wewnętrzne układu nie wyprowadzane na jego końcówki.

```
VARIABLE
   <lista zmiennych> : <typ zmiennej wewnętrznej>;
  {<lista zmiennych> : <typ zmiennej wewnętrznej>;}
```

gdzie

```
<lista zmiennych> ::= <zmienna> {,<zmienna>}
<typ zmiennej wewnętrznej> ::= NODE | TRI_STATE_NODE |
                               <podprojekt> |
                               <element podstawowy> |
                               <maszyna stanów>
<podprojekt> ::= <nazwa zbioru z projektem>
<element podstawowy> ::= DFF | DFFE | JKFF | LATCH | TRI
<maszyna stanów> ::=
      MACHINE [OF BITS <rejestr>] WITH STATES (<stan>{,<stan>})
      <rejestr> ::= <lista nazw> | <nazwa>[<zakres>]
      <zakres> ::= <liczba dziesiętna> .. <liczba dziesiętna>
      <stan> ::= <nazwa> [= <liczba dziesiętna> | <nazwa>]
      <liczba> ::= <kod liczbowy> ”<ciąg cyfr>”
      <kod liczbowy> ::= B | O | Q | H | X
```

uwagi:

- nazwa zbioru z podprojektem nie powinna przekroczyć 32 znaków;
- elementy podstawowe (ang. primitives – patrz help programu MAX+PlusII) pogrupowane są w następujące kategorie: bufory, przerzutniki i zatrzaski, układy logiczne i pozostałe (w definicji powyższej wymieniono tylko najczęściej wykorzystywane np. DFF – przerzutnik D, DFFE – przerzutnik D z blokadą zegara, JKFF – przerzutnik JK itp.);
- w deklaracji maszyny stanów lub inaczej automatu po słowie kluczowym MACHINE określa się liczbę bitów rejestru stanu automatu a kończy ją lista stanów wewnętrznych opisanych liczbami bądź nazwami z opcjonalnie przypisanymi kodami;
- liczby występujące w AHDL mogą być zapisane w następujących kodach: B – dwójkowo, dziesiętnie (format domniemany nie wymagający kodu liczbowego), O lub Q – ósemkowo, H lub X – szesnastkowo (format domyślny liczby to dziesiętny od jednej do 10 cyfr); w zapisie dwójkowym można używać również znaku X jako wartość dowolna danej cyfry.

W omawianym przykładzie zmienna WY została zadeklarowana jako zmienna realizowana na przerzutnikach D.

Zmienne rejestrowe wyposażone są w atrybuty opisujące wejścia przerzutników. Do najczęściej używanych należą:

.clk- wejście taktujące;

.clrn - wejście asynchronicznego zerowania przerzutnika;

.prn - wejście asynchronicznego ustawiania przerzutnika w stan '1'.

Na wejścia te podawane są najczęściej sygnały globalne ale mogą być one sterowane dowolną funkcją logiczną.

W przypadku omawianego przykładu przerzutniki WY taktowane są sygnałem CK a zerowane asynchronicznie sygnałem CR.

- **BEGIN ... END**

Słowo kluczowe **BEGIN** rozpoczyna sekcję opisu działania projektowanego układu a **END** ją kończy. W sekcji tej mogą być umieszczane min.:

- Równania i funkcje złożone;
- tablice wartości;
- opisy automatów.

Równania i funkcje złożone

- Równania pozwalają na wykonanie prostych operacji podstawień. Uproszczona definicja zapisu w notacji BNF jest następująca:

```
<równanie> ::= <zmienna wyjściowa> = <zmienna> [<operator> <zmienna>];
```

gdzie

```
<zmienna wyjściowa> :: = <zmienna>
<zmienna> ::= <nazwa> | '('<nazwa> [,<nazwa>]')'
<nazwa> ::= <nazwa prosta> | <nazwa indeksowana>
<nazwa prosta> ::= <litera> [<litera> | <cyfra>]
<nazwa indeksowana> ::= <nazwa prosta>
     '[' [<liczba dziesiętna> [..<liczba dziesiętna>]] ']'
<operator> ::= <operator logiczny> | <operator relacji> |
                <operator arytmetyczny>
<operator logiczny> ::= ! | NOT | & | AND | !& | NAND |
                        # | OR | !# | NOR | $ | XOR |
                       !$ | XNOR
<operator relacji> ::= == | != | > | >= | < | <=
<operator arytmetyczny> ::= + | - | * | DIV | MOD | ^ | LOG2
```

```
ZZ[] = (XX[] + YY[]) & (S[]==1);
          % operacja arytmetyczna pod warunkiem S[]==1%
(ZZ[7..0],ZZ[15..8])= (XX[15..8],YX[7..0]) AND (S[]==2);
           % konkatenacja słów binarnych %
ZZ[] = (ZZ[14..0],ZZ15);
           % przesunięcie cykliczne w lewo %
```

- **IF THEN ELSE** pozwala na warunkowe wykonanie operacji podstawień. Opis struktury w notacji BNF jest następujący:

```
IF <wyrażenie warunkowe> THEN
   <równanie>;
ELSE <równanie>;
END IF;
```

lub

```
IF <wyrażenie warunkowe> THEN
   <równanie>;
[ ELSIF <wyrażenie warunkowe> THEN
```

```
  <równanie>;]
ELSE <równanie>;
END IF;
```

gdzie

```
<wyrażenie warunkowe> :: = <zmienna> <operator logiczny>
                                  <zmienna> | <stała>
<stała> ::= <liczba> | <ciąg alfanumeryczny>
<ciąg alfanumeryczny> ::= ” [<znak alfanumeryczny>] ”
```

```
if S[]==0 then
    ZZ[] = XX[]; % proste podstawienie pod warunkiem S=0 %
elsif s[]==1 then
    ZZ[] = (XX[] + YY[]); % operacja arytmetyczna %
elsif s[]==2 then
   (ZZ[7..0],ZZ[15..8])= (XX[15..8],YY[7..0]);
else
    ZZ[] = (ZZ[14..0],ZZ15); % przesunięcie cykliczne w lewo %
end if
```

- **CASE WHEN** pozwala podobnie jak poprzednia konstrukcja na warunkowe wykonanie operacji podstawień. Opis struktury w notacji BNF jest następujący:

```
CASE <zmienna> IS
   WHEN  <stała> => <równanie>;
  [WHEN  <stała> => <równanie>;]
END CASE;
```

```
case S[] is
  when 0 => ZZ[] = XX[];
   % proste podstawienie pod warunkiem S=0 %
  when 1 => ZZ[] = (XX[] + YY[]); % operacja arytmetyczna %
  when 2 => (ZZ[7..0],ZZ[15..8])= (XX[15..8],YY[7..0]);
  when 3 => ZZ[] = (ZZ[14..0],ZZ0);
      % przesunięcie cykliczne w lewo %
end case;
```

Tabele funkcji

Tabele funkcji służą do czytelnego opisu układów koderów, dekoderów i innych bloków funkcjonalnych, które w sposób jednoznaczny przypisują stan wyjść stanom wejść. Postać tabeli w zapisie BNF jest następująca:

```
<tabela> ::=
    TABLE <lista wejść> => <lista wyjść>;
          <lista wejść> => <lista wyjść>;
         {<lista wejść> => <lista wyjść>;}
    END TABLE;
```

```
TABLE WE[]  => A, B, C, D, E, F, G;
   H"0"     => 1, 1, 1, 1, 1, 1, 0;
   H"1"     => 0, 1, 1, 0, 0, 0, 0;
     2      => 1, 1, 0, 1, 1, 0, 1;
   Q"3"     => 1, 1, 1, 1, 0, 0, 1;
   H"4"     => 0, 1, 1, 0, 0, 1, 1;
   H"5"     => 1, 0, 1, 1, 0, 1, 1;
END TABLE;
```

Maszyna stanów – automat

Maszyny stanów służą do opisywania automatów w projektowanych układach. Sposób zadeklarowania maszyny stanów został przedstawiony przy okazji dyrektywy VARIABLE. Warto

zwrócić uwagę na fakt, że wymieniony rejestr stanu w definicji automatu nie musi być deklarowany ponownie.

Do opisu funkcji przejść i wyjść automatu można użyć tablic, case-when lub if-then-else. Na początek przykład automatu realizującego funkcję licznika modulo 5 generującego na wyjściu WY kolejne stany kodu NKB. Kodowanie stanów jest dowolne co pokazano w przykładzie.

```
TITLE "Automat - Licznik rewersyjny modulo 5 bez ladowania";
SUBDESIGN automat
(
   WY[2..0] : output;
            CR : input;
            UD : input;
            CK : input;
)
VARIABLE
   AUT : machine of bits (Q[2..0])
    with states(S0=B"000",S1=B"001",S2=B"010",S3=B"111",S4=B"101");
BEGIN
   AUT.clk   = CK;
   AUT.reset = !CR;
    case AUT is
      when S0 => WY[]=B"000"; if (UD==0) then AUT=S1; else AUT=S4; end if;
      when S1 => WY[]=B"001"; if (UD==0) then AUT=S2; else AUT=S0; end if;
      when S2 => WY[]=B"010"; if (UD==0) then AUT=S3; else AUT=S1; end if;
      when S3 => WY[]=B"011"; if (UD==0) then AUT=S4; else AUT=S2; end if;
      when S4 => WY[]=B"100"; if (UD==0) then AUT=S0; else AUT=S3; end if;
    end case;
END;
```

Automat ma zdefiniowane następujące wejścia sterujące:

.clk - wejście taktujące;

.reset - wejście zerujące aktywne poziomem wysokim (inaczej niż przerzutniki);

.ena - wejście blokujące wpływ sygnału taktującego na automat.

Ten sam automat opisany za pomocą tabeli jak pokazano poniżej zachowa się w sposób identyczny do poprzednio zaprojektowanego. W poniższym przykładzie pominięto sekcję deklaracyjną identyczną do poprzedniego opisu.

```
BEGIN
   AUT.clk   = CK;
   AUT.reset = !CR;
      table
            AUT, UD  => AUT, WY[];
             S0,  0  =>  S1, B"000";
             S0,  1  =>  S4, B"000";
             S1,  0  =>  S2, B"001";
             S1,  1  =>  S0, B"001";
             S2,  0  =>  S3, B"010";
             S2,  1  =>  S1, B"010";
             S3,  0  =>  S4, B"011";
             S3,  1  =>  S2, B"011";
             S4,  0  =>  S0, B"100";
             S4,  1  =>  S3, B"100";
   end table;
END;
```

Pojedyncze sygnały nie mogą przyjmować wartości liczbowych 1 lub 0. Zastępują je predefiniowane zmienne odpowiednio VCC i GND.