Атрибути

Атрибути

В VHDL сигнали, змінні та інші об'єкти крім свого значення, також мають множину атрибутів (сталих ознак).

Атрибут об'єкта - це спеціальна функція, яка повертає його особливе значення.

Є наперед визначені атрибути.

Є атрибути користувача, визначені користувачем як особливі ознаки конкретного типу об'єкта.

Атрибути бувають різні:

тип, значення, сигнал, функція, діапазон.

Атрибут об'єкта записується як:

```
\ім'я_об'єкта\'\ім'я_атрибута\
```

Атрибути для скалярного типу T

T'left – найлівіше значення множини елементів типу T.

T'right – найправіше значення множини типу T.

T'high – найбільше значення в множині типу T.

T'low –найменше значення в множині типу T.

T'image(X) – функція строкового представлення виразу X типа T.

T'value(X) – функція значення типу T від строкового представлення X.

T'pos(X) – функція номеру позиції елемента X в множині типу T.

T'val(X) –функція значення елементу типу T, який стоїть в множині в позиції X.

Атрибути для скалярного типу T

Приклади атрибутів:

```
type st is (one,two,three);
st'right = three,
st'pos(three) = 2,
st'val(1) = two.
positive'low =  1,
positive'high =2147483647.
constant ii:integer:=integer'value("1_000");
constant si:string:= integer'image(330) ;
```

Атрибути для регулярного типу A

A'left[(N)] – ліве значення індекса по N-й розмірності.
A'right[(N)] - праве значення індекса по N-й розмірності.
A'high[(N)] - найбільший індекс по N-й розмірності.
A'low[(N)] - наименший індекс по N-й розмірності.
A'range[(N)] – діапазон індексів по N-й розмірності.
A'reverse_range[(N)] – обернений діапазон індексів
A'length[(N)] – кількість різних індексів по N-й розмірності.
A'ascending[(N)] - функція, дорівнює true, якщо діапазон індексів по N-й розмірності - зростаючий.

Атрибути для регулярного типу

Приклади атрибутів:

```
type s1 is array(2 downto 0) of integer;
Variable v1:s1;
v1'left = 2,  v1'right = 0,  v1'high = 2, v1'low = 0,
v1'range = 2 downto 0,
v1'reverse_range = 0 to 2,
v1'length = 3.
type s2 is array(2 downto 1, 0 to 3) of integer;
Variable v2:s2;
v2'left(1) = 2,  v2'right(2) = 3,
v2'high(1) = 2, v2'low(2) = 0,
v2'range(2) = 0 to 3, v2'reverse_range(1) = 1 to 2,
v2'length(2) = 4.
```

Атрибути сигналів S

S'stable[(T)] – сигнал = true, якщо за час T не було подій у сигнала S.

S'transaction – сигнал типу bit, змінює значення на протилежне в циклах моделювання, в яких було присвоювання нового значення сигналу S.

S'event – сигнал = true, якщо була подія в сигналі S в даному циклі моделювання.

S'active – сигнал = true, якщо сигналу S присвоєно нове значення в даному циклі моделювання.

S'last_value – сигнал такого же типу, що і S, який вміщує значення S перед останньою подією в ньому.

Атрибути сигналів

Приклади атрибутів при моделюванні триггерів:

```
process(CLK)
begin
   if CLK='1' and CLK'event then   -- D- тригер
      q1<=a1;
   end if;
   if not CLK'stable then          -- D- триггер
      q2<=a2;
   end if;
end process;
```

Атрибути сигналів

Приклади атрибутів при моделюванні триггерів:

```
process(CLK)
begin
    if CLK'event and CLK'last_value='0' then
        q3<=a3; -- D- тригер
   end if;
   if CLK'active                          -- D- тригер
       q4<=a4;
   end if;
  q5<=CLK'transaction;                    -- T- триггер
end process;
```

Атрибути користувача

атрибути призначені для додавання об'єктам додаткових властивостей, які не спроможні дати вбудовані типи і атрибути.

Це, наприклад:

способ кодування станів автомата,

вказівка компілятору по керуванню оптимізацією,

про розміщення блоків, їхнє виконання,

про призначення портів номерам виводів,

начальний стан схем пам'яті і т.і.

Атрибути користувача

Завдання атрибута складається з об'яви і специфікації

\об'ява атрибута\::=**atribute** \ідентифікатор\ : \ тип\

\специфікація атрибута\::=**attribute** \идентифікатор\ **of**
\ім'я об'єкта\ | **others** | **all** :
\клас об'єкта\ **is** \вираз\

\клас об'єкта\::= **entity** | **architecture** | **configuration** |
package | **procedure** | **function** | **type** |
subtype | **constant** | **signal** | **variable** | **file** |
component | **label** | **literal** | **units** | **group**

Атрибути користувача

Приклад: атрибут вказує спосіб кодування станів автомата:

```
type \стан\ is ( \сброс\,\початок\,\робота\,\кінець\);
attribute enum_encoding : string;
attribute enum_encoding of \стан\ : type is
                                    "000 001 010 100" ;
```

Приклад: атрибут задає стан постійної пам'яті:

```
 attribute init:  string;
  attribute init of U_ROM: label is X"AB56";
U_ROM:ROM16X1   port map(DO,A0,A1,A2,A3);
```

Псевдоніми

Псевдонім - це інше ім'я об'єкта.

Псевдоніми в VHDL допомогають представити програму в більш зручному для читання і моделювання вигляді

\об'ява псевдоніма\::= **alias** \ідентифікатор\ |
символьний літерал\| \символ оператора\
[:\підтип\] **is** \ім'я\ [\сигнатура\];

[\сигнатура\] вказує для якої з перезавантажених функцій зробити псевдонім,
а саме – типи вхідних і вихідних параметрів.

Псевдоніми

Приклад: псевдонім допомагає звертатись до поля кода операції команди як до окремого сигналу, не об'являючи цей сигнал :

```
alias \код операції\: bit_vector(7 downto 0) is
                                \команда\(31 downto 24);
```

Приклад: псевдонім константи числа пі:

```
alias  PI is IEEE.math_real.MATH_PI ;
```

Приклад: псевдонім типу:

```
alias  SV is std_logic_vector;
```

Псевдоніми

Приклад: псевдонім функції перетворення цілого у вектор з пакета IEEE.STD_LOGIC_arith є набагато коротшим:

```
alias  TO_V is CONV_STD_LOGIC_VECTOR
                    [integer, integer return std_logic_vector];
```

Тут в дужках – сигнатура.
Приклад заміни функції з пакета IEEE.numeric_bit символом

```
alias "<" is shift_left [SIGNED, NATURAL return SIGNED];
A < 3     тепер те саме що   A shift_left 3
```

Псевдоніми підтримуються компіляторами – синтезаторами, але не в повному обсязі.
Як правило, не підтримуються псевдоніми процедур і функцій.

Мітки в програмі

Мітка - ідентифікатор, унікальний в границях даної програмної одиниці :

```
Label: /будь-який оператор або декларація/
```

Мітки обов’язкові для операторів:

вставки компонента,

блока,

generate,

циклів, якщо цикли вкладені і треба вказати точку, куди вийти з цикла по **next** або **exit**.

Мітки в програмі

Мітки бажані :

-якщо оператор займає багато рядків, то на його початку і в кінці ставлять мітку, щоб легше відрізняти його.

-якщо в одній частині проекта задекларовані підпрограми, константи, які слід викликати з іншого місця, то їх відмічають міткою і знаходять для виклику по опису use.

-якщо при моделюванні бажано знайти в ієрархії проекта потрібний процес, то по його мітці це зробити значно легше.

-якщо необхідно знайти відповідність операторів в VHDL-моделі і блоків в синтезованій логічній схемі, так як ці блоки наслідують імена міток операторів.

-ім'я мітки, може нести інформацію про призначення відміченого оператора або об'єкта і тому зпрощує читання програми.