VHDL

Introdução

Paulo C. Centoducatte
ducatte@ic.unicamp.br

março de 2005

VHDL - Comandos seqüenciais

- Chamada de procedimentos (também concorrente)
- Assertion (também concorrente)
- Assinalamento de sinal (também concorrente)
- Assinalamento de variáveis
- WAIT
- IF, CASE
- NEXT
- EXIT, RETURN, LOOP
- NULL

VHDL - Loop

```
loop
  comando_1
  comando_2;
end loop;
```

Exemplo:

```
begin
     count <= count_value;
     loop
       wait until clk = '1';
       count_value := (count_value + 1) mod 16;
       count <= count_value;
     end loop;
```

VHDL - Next e Exit

```
loop
  comando_1
  next when condição;
  comando_2;
end loop;
```

```
begin
  count <= count_value;
  loop
     loop
       wait until clk = '1' or reset = '1';
       exit when reset = '1';
       count_value := (count_value + 1) mod 16;
       count <= count_value;
     end loop;
     count_value := 0; -- at this point, reset = '1'
     count <= count_value;
     wait until reset = '0';
  end loop;
```

VHDL - While loops

```
entity cos is
  port ( theta : in real;  result : out real );
end entity cos;


…..
sum := 1.0;
term := 1.0;
n := 0;
while abs(term) > abs (sum / 1.0E6) loop
      n := n + 2;
      term := (-term) * theta**2 / real(((n-1) * n));
      sum := sum + term;
end loop;
result <= sum;


…
```

VHDL – For Loops

```
entity cos is
  port ( theta : in real;  result : out real );
end entity cos;


…..
sum := 1.0;
    term := 1.0;
    for n in 1 to 9 loop
      term := (-term) * theta**2 / real(((2*n-1) * 2*n));
      sum := sum + term;
    end loop;
    result <= sum;
```

VHDL – For Loops

```
variable a, b : integer;
  begin
    a := 10;
    for a in 0 to 7 loop
      b := a;
    end loop;
    -- a = 10, and b = 7
   wait;
```

VHDL - Assertion

[Label:] assert exp_booleana
[report expressão [severity expressão];

Type severity_level is (note, warning, error, failure)

```
assert valor <= max_valor;          --      “Assertion violation”

assert valor <= max_valor           --      Severity = erro
   report “valor maior que o permitido";

assert valor <= max_valor
        report “valor maior que o permitido“
        severity note;
```

VHDL - Assertion

```
entity SR_flipflop is
  port ( S, R : in bit;  Q : out bit );
end entity SR_flipflop;

architecture checking of SR_flipflop is
begin
set_reset : process (S, R) is
    begin
      assert   S = '1'  nand  R = '1';
      if S = '1' then
        Q <= '1';
      end if;
      if R = '1' then
        Q <= '0';
      end if;
    end process set_reset;
end architecture checking;
```

VHDL – Atributos de tipos escalar

- Atributos comuns a todos os tipos escalares:

T'left - primeiro (mais a esquerda) valor em T
T'right - último (mais a direita) valor em T
T'low - menor valor em T
T'high - maior valor em T
T'ascending - TRUE se o range de T é crescente
T'image(x) - string representando o valor de x
T'value(s) - o valor em T que é representado por s

VHDL - Atributos de tipos escalar


```
type resistance is range 0 to 1E9
    units
      ohm;
      kohm = 1000 ohm;
      Mohm = 1000 kohm;
    end units resistance;

  type set_index_range is range 21 downto 11;
  type logic_level is (unknown, low, undriven, high);
```

```
set_index_range'left = 21;
set_index_range'right = 11;
set_index_range'low = 11;
set_index_range'high = 21;
set_index_range'ascending = false;
set_index_range'image(14) = "14";
set_index_range'value("20") = 20;
```

```
resistance'left = 0 ohm;
resistance'right = 1E9 ohm;
resistance'low = 0 ohm;
resistance'high = 1E9 ohm;
resistance'ascending = true;
resistance'image(2 kohm) = "2000 ohm";
resistance'value("5 Mohm") = 5000000 ohm;
```

VHDL - Atributos de tipos escalar

- Atributos relativos aos tipos escalares discreto e físico:

T'pos(x) - número da posição de x em T
T'val(n) - valor em T na posição n
T'succ(x) - valor em T na posição maior que x
T'pred(x) - valor em T na posição menor que x
T'leftof(x) - valor em T na posição a esquerda de x
T'rightof(x) - valor em T na posição a direita de x

VHDL - Atributos de tipos escalar


```
type resistance is range 0 to 1E9
    units
      ohm;
      kohm = 1000 ohm;
      Mohm = 1000 kohm;
    end units resistance;

 type set_index_range is range 21 downto 11;
 type logic_level is (unknown, low, undriven, high);
```

```
logic_level'left = unknown;
logic_level'right = high;
logic_level'low = unknown;
logic_level'high = high;
logic_level'ascending = true;
logic_level'image(undriven) = "undriven";
logic_level'value("Low") = low;
logic_level'pos(unknown) = 0;
logic_level'val(3) = high;
logic_level'succ(unknown) = low;
logic_level'pred(undriven) = low;
```

VHDL - Atributos de tipos array

- **Atributos**
  - A'left(N)
  - A'right(N)
  - A'low(N)
  - A'high(N)
  - A'range(N)
  - A'reverse_range(N)
  - A'length(N)
  - A'ascending(N)

- **Exemplo**

  type A is array (1 to 4 , 31 downto 0) of boolean;

  - A'left(1) = 1
  - A'right(2) = 0
  - A'low(1) = 1
  - A'high(2) = 31
  - A'range(1) = is 1 to 4
  - A'reverse_range(2) = is 0 to 31
  - A'length(2) = 32
  - A'ascending(1) = true
  - A'ascending(2) = false

VHDL – Atributos de Sinais

S´delayed(T) - mesmo valor que S porém atrasado de T
S´stable(T) - booleano: true se não houve evento em S
S´quit(T) - booleano: true se não houve transação em S
S´transaction - bit: alterna entre ´0´ e ´1´ a cada transação em S
S´event - booleano: true se houve um evento em S
S´active - booleano: true se houve uma transação em S
S´last_event - intervalo de tempo desde o último evento em S
S´last_active - intervalo de tempo desde a última transação em S
S´last_value - valor de S antes do último evento

VHDL – Uso de Atributos

- Detecção de borda de relógio

  clk'EVENT and clk = '1'

- Determinação de largura de pulso

  Sinal'LAST_EVENT >= 5 ns

- Teste de Hold e Setup

  clk'LAST_VALUE = '0' AND clk = '1' AND dado'STABLE(min_setup_time)

VHDL - Componente

- Declaração
- Instanciação
- Instanciação condicional
- Modelamento Estrutural

VHDL - Componente

- Declaração de um componente

component identifier [**is**]
  [**generic** (generic_interface_list);]
  [**port** (port_interface_list);]
end component [identifier];

OBS.: Similar a ENTIDADE

VHDL - Componente

- Exemplo

```
component flip-flop is
    generic (Tprop, Tsetup, Thold : delay);
    port (clk: in bit; clr : in bit; d : in bit; q : out bit);
end component flip_flop;
```

VHDL - Componente

- Instanciação
  - Declaração de componente define o tipo do módulo
  - Instaciação de componenente define seu uso em um projeto

instatiation_label:

[**component**] componente_name

[**generic map** (generic_association_list)]

[**port map** (port_association_list)];

VHDL – Componente (exemplo)

- Exemplo:

```
entity reg4 is
  port ( clk, clr : in bit;  d : in bit_vector(0 to 3);
         q : out bit_vector(0 to 3) );
end entity reg4;
-------------------------------------------------------------------
architecture struct of reg4 is
  component flipflop is
    generic ( Tprop, Tsetup, Thold : delay_length );
    port ( clk : in bit;  clr : in bit;  d : in bit;
                q   : out bit );
  end component flipflop;
```

VHDL – Componente (exemplo)

```
begin
  bit0 : component flipflop
      generic map ( Tprop => 2 ns, Tsetup => 2 ns, Thold => 1 ns )
      port map ( clk => clk, clr => clr, d => d(0), q => q(0) );
  bit1 : component flipflop
      generic map ( Tprop => 2 ns, Tsetup => 2 ns, Thold => 1 ns )
      port map ( clk => clk, clr => clr, d => d(1), q => q(1) );
 bit2 : component flipflop
      generic map ( Tprop => 2 ns, Tsetup => 2 ns, Thold => 1 ns )
      port map ( clk => clk, clr => clr, d => d(2), q => q(2) );
  bit3 : component flipflop
      generic map ( Tprop => 2 ns, Tsetup => 2 ns, Thold => 1 ns )
      port map ( clk => clk, clr => clr, d => d(3), q => q(3) );
end architecture struct;
```

VHDL - Configuração

```
configuration identifier of entity_name is
      for architeture_name
          { for component_specification
                    binding_indication;
             end for;}
       end for;
end [ configuration] [ identifier];
```

VHDL - Componente

- Configuração

```
FOR nome_da_instancia|others|all: nome_componente  -- component_specification
   USE ENTITY especificação_da_entidade;    -- binding_indication
END FOR;

FOR inst51: xor_gate
  USE ENTITY lib_projeto.xor(arq_rtl);
END FOR;

FOR bit0,bit1: flipflop
  use entity work.edge_triggered_Dff(basic);
end for;

FOR others: flipflop
  use entity work.edge_triggered_Dff(rtl);
end for;
```

VHDL – Exemplo

```
Architecture rtl of top is
  Component and2
    port(a, b: in std_logic;
         c: out std_logic);
  End component;
  Component latchD
    port(d, clk   : in std_ulogic;
         q, notq : out std_logic);
  End component;

  For all : and2 use entity work.and2(rtl);
  For all : latchD use entity work.latchD(rtl);

  signal Q, NOTQ : std_ulogic := '1';
```

```
Begin
  inst_latchD: latchD
      port map(d1,clk, Q,NOTQ);

  inst_and2_a: and2
      port map(d1,Q,S1);

 inst_and2_b: and2
      port map(d2,NOTQ,S3);
End rtl;
```

OBS.: d1, d2 e clk são sinais de entrada e S1, S2 e S3 são sinais de saída