



Sistemas Digitais

Aulas práticas de laboratório

Comandos sequenciais do VHDL

Prof. Dr. Eng. Rafael Luiz Cancian





Introdução





Introdução

- Comandos sequenciais no VHDL são aqueles usados dentro do bloco “*begin*” e “*end process*” de um comando concorrente *process*.
- Os comandos sequenciais são interpretados *como se fossem* executados em sequência, como uma linguagem de programação.
- Ao contrário dos comandos concorrentes, não há um mapeamento claro e direto entre os comandos sequenciais e o hardware gerado, o que exige do projetista muita disciplina ao usar os comandos sequenciais do VHDL.



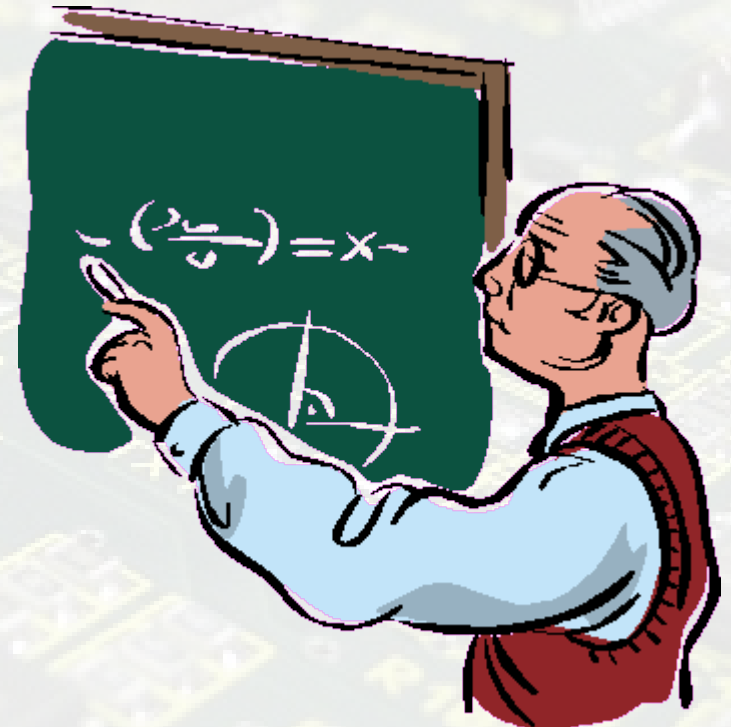
Comando Concorrente Process

- Sintaxe:

```
process(sensitivity-list)
  declarations;
begin
  sequential statement;
  sequential statement;
  ...
end process;
```
- Todos os sinais usados no process devem estar na lista de sensibilidade.
- A atribuição aos sinais (externos ao *process*) ocorre apenas quando o process termina.
- A atribuição às variáveis (locais ao process) ocorre na própria atribuição (como software).
- Atribuições “incompletas” inferem memória.



Conteúdo





Comandos Sequenciais


- Todos os comandos contidos por “*begin*” e “*end process*” de um *process* são comandos sequenciais.
- Os comandos sequenciais do VHDL são:
 - Atribuição simples a sinais;
 - Atribuição simples a variáveis;
 - if then;
 - case when;
 - for loop;
 - wait (apenas testbench).
- A descrição usando comandos sequenciais é sempre comportamental e nunca estrutural.



Atribuição Simples

- Sintaxe:

```
<signal> <= <expression_same_type>;  
<variable> := <expression_same_type>;
```


- Exemplos:

```
sigstatus <= '1';  
sigeven <= (p1 and p2) or (p3 and p4);  
sigarith_out <= a + b + c - 1;  
var1 := '1';  
varLog := (p1 and p2) or (p3 and p4);  
varInt := a + b + c - 1;
```
- A atribuição simples a sinais é o único comando sequencial que é idêntico ao comando concorrente de atribuição simples.
- Atribuição a variável ocorre imediatamente e atribuição a sinal ocorre quando o processo termina.



if then

- Sintaxe:

```
if <boolean_expr_1> then  
    <sequential_statements;>  
elsif <boolean_expr_2> then  
    <sequential_statements;>  
elsif <boolean_expr_3> then  
    <sequential_statements;>  
...  
else  
    <sequential_statements;>  
end if;
```

- Exemplos:

```
if sigStd='0' then  
    varInt := a+2;  
elsif sigInt > 1 then  
    if sigInt = 2 then  
        varInt := a-2;  
    else  
        varInt := a-10;  
    end if;  
else  
    varInt := a;  
end if;
```




Atribuição Incompleta a Sinais

- O código

```
process (a, b)
begin
  if (a=b) then
    eq <= '1';
  end if ;
end process;
```

- equivale a

```
process (a, b)
begin
  if (a=b) then
    eq <= '1';
  else
    eq <= eq;
  end if ;
end process;
```

- há inferência de um elemento de memória (um registrador), tornando o circuito sequencial.




Case When

- Sintaxe:

```
case <case_expression> is
  when <choice_1> =>
    <sequential statements>
  when <choice_2> =>
    <sequential statements>
  ...
  when <choice_n> =>
    <sequential statements>
end case;
```

- Exemplos:

```
case s is
  when "000" | "111" =>
    x <= a;
  when "010" | "101" | "011" =>
    x <= b;
  when "110" =>
    x <= c;
  when others =>
    x <= d;
end case;
```





For Loop

- **Sintaxe:**

```
for <index> in <loop_range> loop  
    <sequential statements;>  
end loop ;
```
- **Exemplos:**

```
for i in sig'range loop  
    sig(i) <= a(i) xor b(i);  
end loop ;
```



```
for i in n-1 downto 0 loop  
    sig(i) <= a(i) xor b(i);  
end loop ;
```



Wait

- Usado apenas para testbenchs (não sintetizável)

- Sintaxes:

```
wait;  
wait on <signals>;  
wait until <boolean_expression>;  
wait for <time_expression>;
```

- Exemplos:

```
process (a, b, clock) is  
begin  
    wait on a, b;  
    c <= a or b;  
    wait for 10 ns;  
    c <= a and b;  
    wait until clock='0';  
    c <= a xor b;  
    wait;  
end process;
```



Referências Bibliográficas

- Vahid, Frank. Sistemas Digitais: projeto, otimização e HDLs. Porto Alegre: Bookman, 2008. ISBN 978-85-7780-190-9
- Chu, Pomg P. RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability. Wiley-Interscience, 2006.
- Pedroni, Volnei. Circuit Design with VHDL. The MIT Press, 3th edition, 2020.

