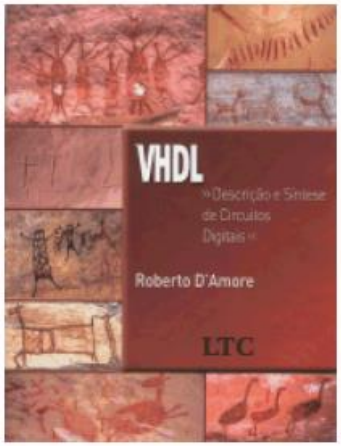
SEL0384/SEL0606 – Laboratório de Sistemas Digitais
Aula 4 – Construções Concorrentes

Prof. Dr. Maximiliam Luppe

Livro adotado:

VHDL - Descrição e Síntese de Circuitos Digitais
Roberto d'Amore

ISBN 85-216-1452-7
Editora LTC www.ltceditora.com.br



Para informações adicionais consulte: **www.ele.ita.br/~damore/vhdl**

Introdução

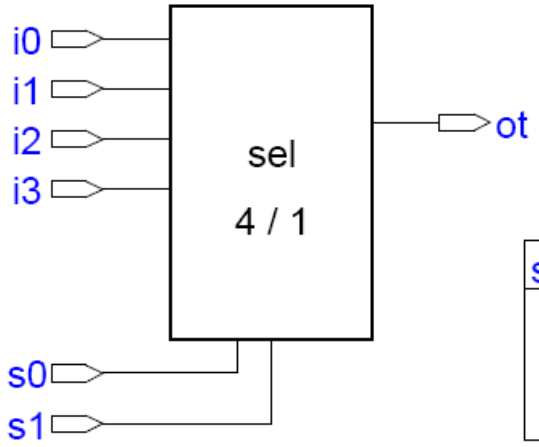
Tópicos

- Histórico
- Aspectos gerais da linguagem
- Síntese de circuitos

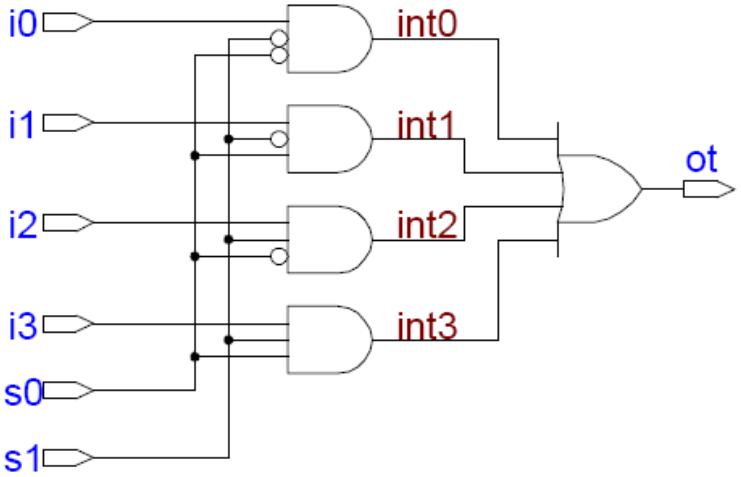
- Entidade de projeto
- Classes de objetos: constante, variável e sinal
- Tipos
- Operadores
- Construção concorrente WHEN ELSE
- Construção concorrente WITH SELECT
- Processos e lista de sensibilidade
- Construção seqüencial IF ELSE
- Construção seqüencial CASE WHEN
- Circuitos síncronos

Descrição de um circuito de seleção

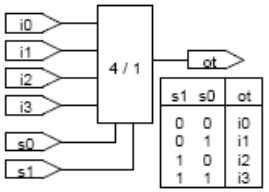
- **entradas:** i0, i1, i2 e i3
- **saída:** ot
- **controle da seleção:** s0 e s1



s1	s0	ot
0	0	i0
0	1	i1
1	0	i2
1	1	i3



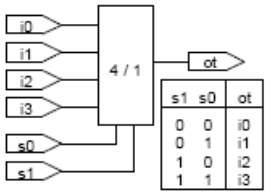
Exemplo: descrição do circuito de seleção



- **descrição emprega uma única expressão**
- **observar:** uso de parêntesis → AND e OR igual precedência

```
 1 ENTITY mux_0 IS
 2   PORT (i0, i1, i2, i3    :  IN  BIT;  -- entradas
 3         s0, s1            :  IN  BIT;  -- selecao
 4         ot                :  OUT BIT); -- saida
 5 END mux_0;
 6
 7 ARCHITECTURE nivel_logico OF mux_0 IS
 8 BEGIN
 9   ot <= (i0 AND NOT s1 AND NOT s0) OR
10         (i1 AND NOT s1 AND     s0) OR
11         (i2 AND     s1 AND NOT s0) OR
12         (i3 AND     s1 AND     s0);
13 END nivel_logico;
```

Exemplo: - nova descrição do circuito de seleção



- **emprega 5 expressões**
- **sinais internos: int0, int1, int2 e int3**
- **observar concorrência do código:**
  - valor de ot → determinado pelas expressões linhas 11, 12, 13 e 14

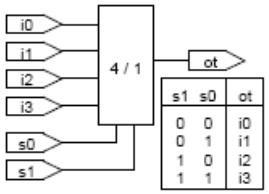
```
 1 ENTITY mux_00 IS
 2   PORT (i0, i1, i2, i3    :  IN  BIT;  -- entradas
 3         s0, s1            :  IN  BIT;  -- selecao
 4         ot                :  OUT BIT); -- saida
 5 END mux_00;
 6
 7 ARCHITECTURE teste OF mux_00 IS
 8   SIGNAL int0, int1, int2, int3 : BIT; -- sinais internos
 9 BEGIN
10   ot   <= int0 OR int1 OR int2 OR int3;
11   int0 <= i0 AND NOT s1 AND NOT s0;
12   int1 <= i1 AND NOT s1 AND     s0;
13   int2 <= i2 AND     s1 AND NOT s0;
14   int3 <= i3 AND     s1 AND     s0;
15 END teste;
```

Construção WHEN ELSE

- **Transferência condicional de um sinal**
- **Contém:** uma lista de condições e expressões
- **Primeira condição verdadeira**: define expressão transferida
- **Formato da construção**:

```
sinal_destino <= expressao_a  WHEN condicao_1 ELSE
                 expressao_b  WHEN condicao_2 ELSE
                 expressao_c;
```

Exemplo: - circuito de seleção - WHEN ELSE



- **nível de abstração mais elevado**
- **descrição mais próxima do comportamento do circuito**
- **opção de escolha:**

linhas 9 a 11: operação AND entre s0 e s1

```
 1 ENTITY mux_1 IS
 2   PORT (i0, i1, i2, i3    :  IN  BIT;
 3         s0, s1            :  IN  BIT;
 4         ot                :  OUT BIT);
 5 END mux_1;
 6
 7 ARCHITECTURE teste OF mux_1 IS
 8 BEGIN
 9   ot <= i0 WHEN s1= '0' AND s0='0' ELSE
10         i1 WHEN s1= '0' AND s0='1' ELSE
11         i2 WHEN s1= '1' AND s0='0' ELSE
12         i3;
13 END teste;
```

Construção WITH SELECT

- **Transferência condicional de um sinal**
- **Contém:** uma lista de opções
- **Todas condições da expressão de escolha devem ser consideradas**
  - não existe uma prioridade como na construção WHEN ELSE
- **Opções pode ser agrupadas:** - caracter | equivale a "ou"
  - TO e DOWNTO delimitam faixas de opções
- **Opções restantes:** palavra reservada OTHERS
- **Formato da construção:**

```
WITH expressao_escolha SELECT       -- expressao_escolha =
 sinal_dest <= expr_a WHEN condicao_1,                -- condicao_1
               expr_b WHEN condicao_2,                -- condicao_2
               expr_c WHEN condicao_3 | condicao_4,   -- condicao_3 ou condicao_4
               expr_d WHEN condicao_5 TO condicao_7,  -- condicao_5 ate condicao_7
               expr_e WHEN OTHERS;                    -- condicoes restantes
```

Construção WITH SELECT

- **Expressão de escolha deve retornar:**

  - tipo discreto

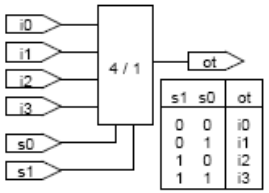
    exemplo: BIT BOOLEAN CHARACTER INTEGER

  - ou um vetor unidimensional

    exemplo: BIT_VECTOR STRING

```
WITH expressao_escolha SELECT      -- expressao_escolha =
 sinal_dest <= expr_a WHEN condicao_1,                -- condicao_1
               expr_b WHEN condicao_2,                -- condicao_2
               expr_c WHEN condicao_3 | condicao_4,   -- condicao_3 ou condicao_4
               expr_d WHEN condicao_5 TO condicao_7,  -- condicao_5 ate condicao_7
               expr_e WHEN OTHERS;                    -- condicoes restantes
```

Exemplo 4: circuito de seleção - WITH SELECT



- **nível de abstração mais elevado**
- **descrição mais próxima do comportamento do circuito**
- **expressão de escolha:**

sinal sel = s1 e s0 concatenados

```
 1 ENTITY mux_9 IS
 2   PORT (i0, i1, i2, i3  : IN  BIT;
 3         s0, s1          : IN  BIT;
 4         ot              : OUT BIT);
 5 END mux_9;
 6
 7 ARCHITECTURE teste OF mux_9 IS
 8   SIGNAL sel : BIT_VECTOR (1 DOWNTO 0);
 9 BEGIN
10   sel <= s1 & s0;
11   WITH sel SELECT
12     ot <= i0 WHEN "00",
13           i1 WHEN "01",
14           i2 WHEN "10",
15           i3 WHEN "11";
16 END teste;
```

Comando PROCESS

- **Objetivo**: delimitar regiões de código seqüencial
- **Início**: palavra reservada PROCESS
- **Lista de sensibilidade**: identifica que sinais ativam a execução do processo
- **Comandos seqüenciais**: próximo capítulo

```
 abc: PROCESS (lista de sensibilidade)
      BEGIN
         comando_1;
         comando_2;
            ..
            ..
         comando_n;
END PROCESS abc;

def: PROCESS (lista de sensibilidade)
      BEGIN
         comando_1;
         comando_2;
            ..
            ..
         comando_n;
END PROCESS def;
```

Cuidados na descrição:

Comparação entre as construções WHEN ELSE e WITH SELECT

- **Construção WHEN ELSE:**
  - ordem das condições indica a prioridade
  - não é necessário apresentar todas condições

```
sinal_destino <= expressao_a  WHEN condicao_1 ELSE     -- condicao_1 = verdadeira
                 expressao_b  WHEN condicao_2 ELSE     -- condicao_2 = verdadeira
                 expressao_c;                          -- nenhuma condicao verd.
```

- **Construção WITH SELECT:**
  - todas condições têm igual prioridade
  - é necessário apresentar todas condições

```
WITH expressao_escolha SELECT                -- expressao_escolha =
 sinal_destino <= expressao_a WHEN condicao_1,              -- condicao_1
                  expressao_b WHEN condicao_2,              -- condicao_2
                  expressao_e WHEN OTHERS;                  -- condicoes restantes
```

Cuidados na descrição:

Criação de *latch* com as construções WHEN ELSE e WITH SELECT

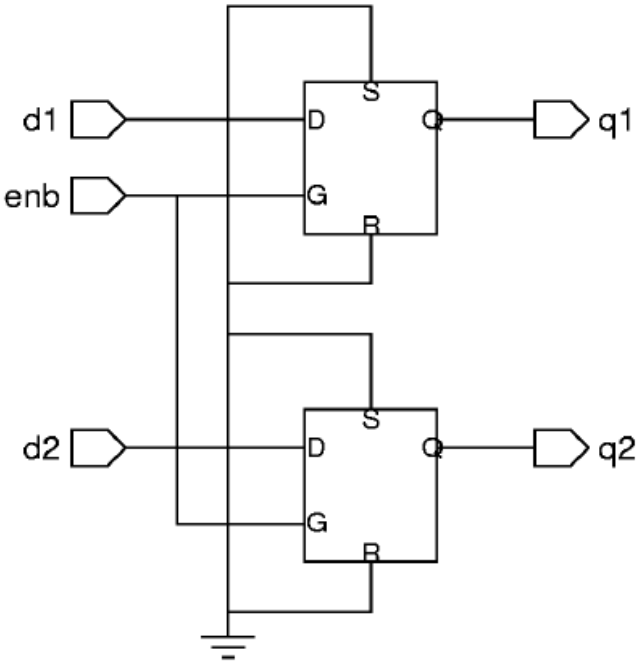
- **Caso uma expressões contenha o sinal de destino:**

```
sinal_destino <= expressao_a  WHEN opcao_1 ELSE
                 .
                 .
                 sinal_destino;
```

- **Resulta**: valor do sinal mantido em uma das condições
- **Não é mais caracterizado um circuito combinacional**
- **Necessário um elemento para armazenar o valor**

- **Exemplos e resultado da síntese (nível RTL)**

```
 1 ENTITY latch_0 IS
 2   PORT (enb, d1, d2 : IN     BIT;
 3         q1, q2      : BUFFER BIT);
 4 END latch_0;
 5
 6 ARCHITECTURE teste OF latch_0 IS
 7 q1 <= d1 WHEN enb ='1' ELSE
 8        q1;
 9
10  WITH enb SELECT
11    q2 <= d2 WHEN '1',
12          q2 WHEN '0';
13 END teste;BEGIN
```



- se enb =1 sinal q segue sinal d

  enb =0 sinal q mantido

- inferido um “lath transparente”