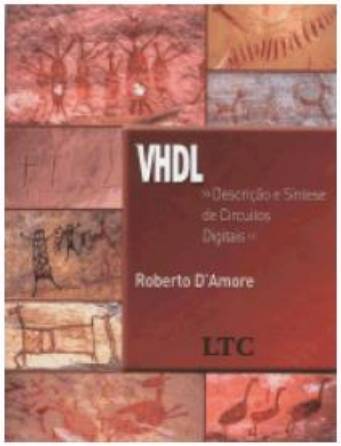
SEL5752/SEL0632 – Linguagens de Descrição de Hardware
Aula 4 – Construções Sequenciais

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

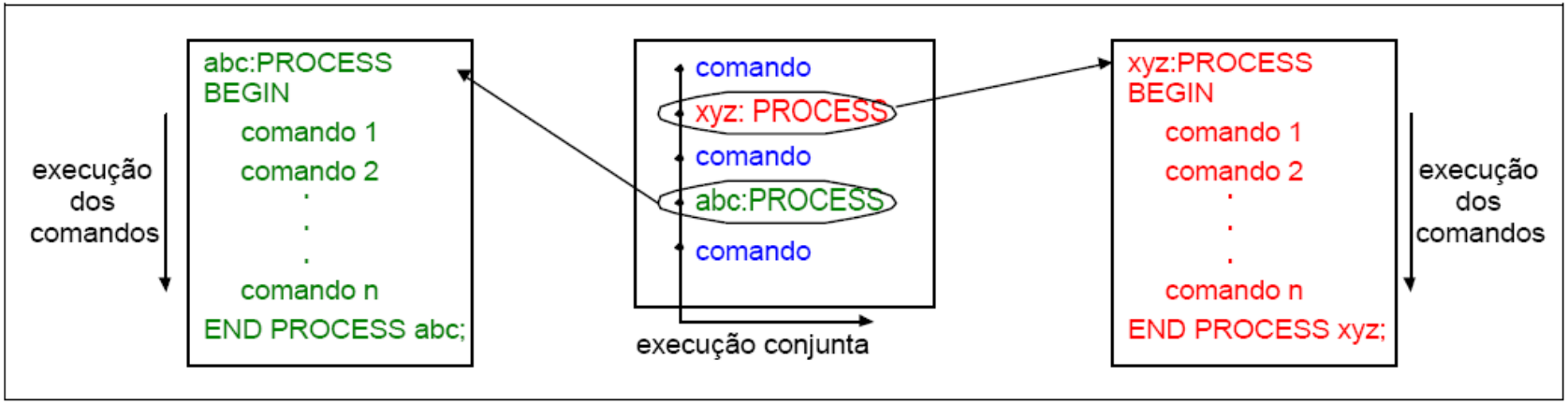
Comandos seqüenciais básicos

Tópicos

- **Lista de sensibilidade em processos**
- **Atribuição de valor para um sinal - região seqüencial**
- **Construção IF ELSE**
- **Construção CASE WHEN**
- **Comando WAIT**
- **Cuidados na descrição**
  - Comparações entre construções WHEN ELSE e IF ELSE
  - Construções IF ELSIF ELSE e CASE WHEN aninhadas
  - O emprego da construção IF ELSE e CASE WHEN

Comandos seqüenciais

- **Comandos seqüenciais podem ocorrer em:**
  - processos
  - subprogramas
- **PROCESS**
  - é um comando concorrente
  - delimita uma região contendo código seqüencial
- **Lembrando**: uma descrição é composta de comandos concorrentes
  - todos são executadas concorrentemente



Lista de sensibilidade em processos

- **Após a comando PROCESS**:
  - possível declarar a lista de sensibilidade
- **Lista de sensibilidade**:
  - define quais sinais causam a execução do processo
- **Execução do processo ocorre se**:
  - um sinal da lista tem valor alterado
- **Iniciada a execução**:
  - comandos são avaliadas na seqüência

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

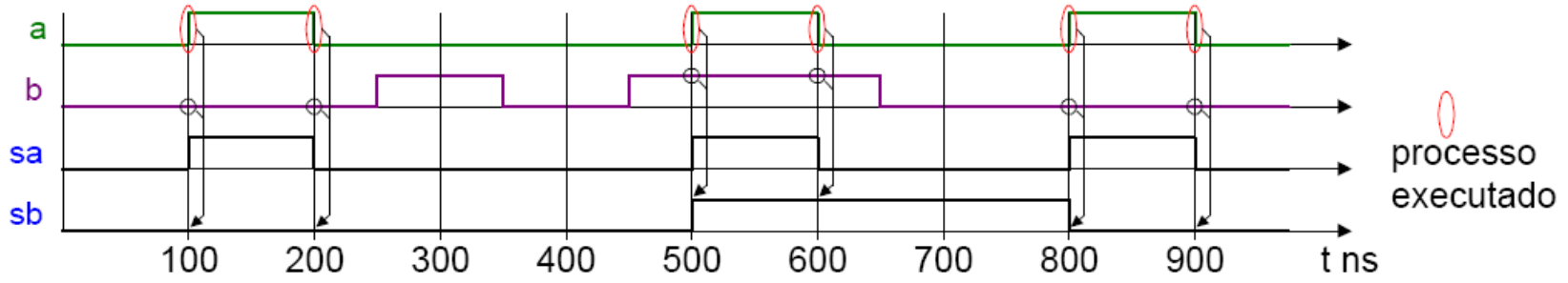
Exemplo: Lista de sensibilidade

- **Valores dos sinais a e b transferidos os sinais sa e sb**
- **Nesta descrição**: unicamente sinal a na lista de sensibilidade
- **Conseqüência**:
  - alteração do valor no sinal a:
    execução do processo → valores de a e b transferidos para sa e sb
  - alteração do valor no sinal b:
    processo não é executado → valores de a e b mantidos

```
 1 ENTITY sens_tes IS
 2   PORT (a, b :  IN  BIT;
 3         sa, sb  :  OUT BIT);
 4 END sens_tes;
 5
 6 ARCHITECTURE teste OF sens_tes IS
 7 BEGIN
 8   abc: PROCESS (a) -- executado na alteracao do valor de "a"
 9   BEGIN
10     sa <= a;
11     sb <= b;
12   END PROCESS abc;
13 END teste;
```

Exemplo 2: Lista de sensibilidade

- **Simulação da entidade**




```
 1 ENTITY sens_tes IS
 2   PORT (a, b :  IN  BIT;
 3         sa, sb  :  OUT BIT);
 4 END sens_tes;
 5
 6 ARCHITECTURE teste OF sens_tes IS
 7 BEGIN
 8   abc: PROCESS (a) -- executado na alteracao do valor de "a"
 9   BEGIN
10     sa <= a;
11     sb <= b;
12   END PROCESS abc;
13 END teste;
```

Atribuição de valor para um sinal - região seqüencial

- **Atribuição de valor para um sinal pode ocorrer**:
  - regiões de código concorrente
  - regiões de código seqüencial

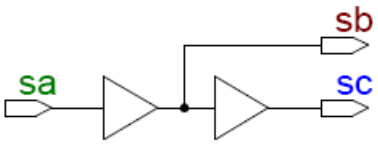
- **Região de código concorrente**
  - atribuição é executada como um comando concorrente:
    - na ocorrência de uma alteração de valor de um sinal da expressão

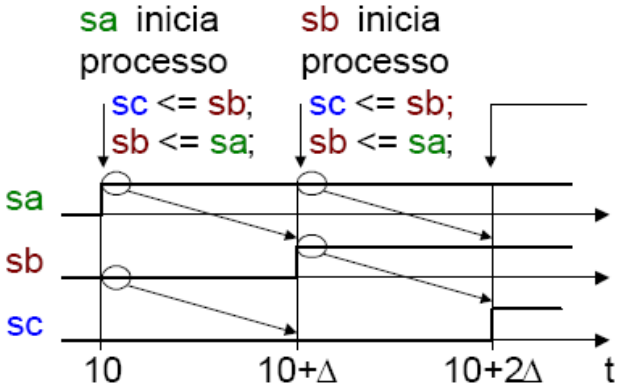
- **Região de código seqüencial** (um processo por exemplo)
  - a atribuição sempre ocorre quando o comando é executado

Atribuição de valor para um sinal - região seqüencial

- **Exemplo**:
  - atribuição do sinal ocorre Δ após a execução do comando
  - iteração ocorre na suspensão do processo
  - duas iterações ocorrem neste exemplo

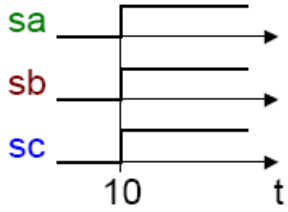
```
 6 ARCHITECTURE teste OF sig_tes0 IS
 7 BEGIN
 8   PROCESS (sa, sb)
 9   BEGIN
10    sc <= sb;
11    sb <= sa;
12   END PROCESS;
13 END teste;
```



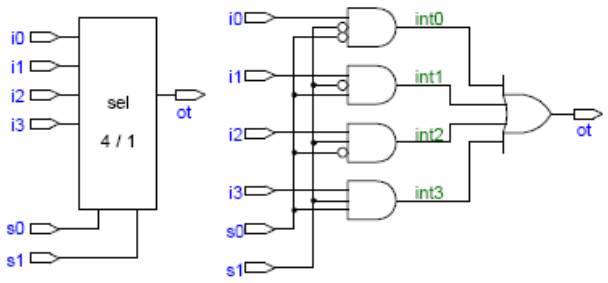


fim das iterações:
sc não consta na lista de sensibilidade

como Δ=0



Atribuição de valor para um sinal
- região de código seqüencial



- **Exemplo** - circuito de seleção:

  - importante: nesta descrição,
    int0 int1 int2 int3 devem ser
    colocados na lista de sensibilidade

```
 1 ENTITY mux_p0 IS
 2   PORT (i0, i1, i2, i3    :  IN  BIT;  -- entradas
 3         s0, s1            :  IN  BIT;  -- selecao
 4         ot                :  OUT BIT); -- saida
 5 END mux_p0;
 6
 7 ARCHITECTURE teste OF mux_p0 IS
 8   SIGNAL int0, int1, int2, int3 : BIT;
 9 BEGIN
10   PROCESS (i0, i1, i2, i3, s0, s1, int0, int1, int2, int3)
11   BEGIN
12     ot <= int0 OR int1 OR int2 OR int3;
13     int0 <= i0 AND NOT s1 AND NOT s0;
14     int1 <= i1 AND NOT s1 AND     s0;
15     int2 <= i2 AND     s1 AND NOT s0;
16     int3 <= i3 AND     s1 AND     s0;
17   END PROCESS;
18 END teste;
```

Construção IF ELSE

(similar: construção WHEN ELSE)

- **Execução condicional de um ou mais comandos seqüenciais**
- **Teste**: definido por uma lista de condições
- **Primeira condição verdadeira**: define os comandos executados
- **Condição de teste**: qualquer expressão que retorne BOOLEAN
- **Início da construção**: comando **IF**
- **Cláusulas ELSIF e ELSE**: opcionais
- **Formato da construção**:

```
IF   condicao_1 THEN
  comando_sequencial;
  comando_sequencial;
ELSIF condicao_2 THEN          -- clausula ELSIF opcional
  comando_sequencial;
  comando_sequencial;
ELSIF condicao_3 THEN
  comando_sequencial;
ELSE                          -- clausula ELSE opcional
   comando_sequencial;
END IF;
```

Construção IF ELSE

- **Possível aninhar várias construções IF ELSE**
  - IF é também um comando seqüencial

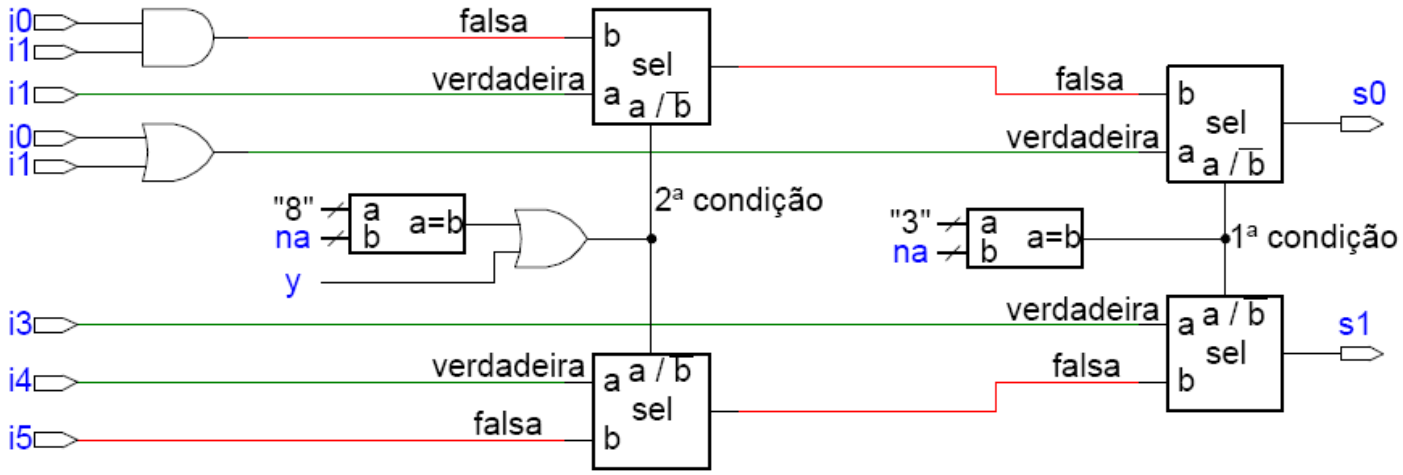
```
IF condicao_1 THEN
  IF condicao_2 THEN
    comando_sequencial;
  ELSE
    comando_sequencial;
  END IF;
ELSE
  IF condicao_3 THEN
    comando_sequencial;
  ELSE
    comando_sequencial;
  END IF;
END IF;
```

Construção IF ELSE

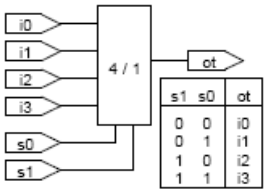
- **Exemplo**:

```
IF na =3 THEN
  s0 <= i0 OR i1;
  s1 <= i3;
ELSIF na =8 OR y ='1' THEN
  s0 <= i1;
  s1 <= i4;
ELSE
    s0 <= i0 AND i1;
    s1 <= i5;
END IF;
```

- **Circuito equivalente**:



Exemplo: - circuito de seleção - IF ELSE



- **Comando seqüencial**
- **Nota**: s1 s2 agrupados no sinal **sel**
- **Lista de sensibilidade**: sinais i0 i1 i2 i3 sel

  - remoção de destes sinais: conseqüência?

```
 1 ENTITY mux_4a IS
 2   PORT (i0, i1, i2, i3  :  IN  BIT;  -- entradas
 3         s0, s1          :  IN  BIT;  -- selecao
 4         ot              :  OUT BIT); -- saida
 5 END mux_4a;
 6
 7 ARCHITECTURE teste OF mux_4a IS
 8   SIGNAL sel : BIT_VECTOR(1 DOWNTO 0);
 9 BEGIN
10   sel <= s1 & s0;
11   abc: PROCESS (i0, i1, i2, i3, sel) --sinal "sel" inserido na lista
12   BEGIN
13     IF    sel = "00" THEN  ot <= i0;
14     ELSIF sel = "01" THEN  ot <= i1;
15     ELSIF sel = "10" THEN  ot <= i2;
16     ELSE                   ot <= i3;
17     END IF;
18   END PROCESS abc;
19 END teste;
```

Construção CASE WHEN

(similar: construção WITH SELECT)

- **Execução condicional de um ou mais comandos seqüenciais**
- **A execução dos comandos**: controlada pelo valor de uma expressão
- **Todas condições da expressão de escolha devem ser consideradas**
  - não existe uma prioridade como na construção IF ELSE
- **Opções podem ser agrupadas**: - caracter | equivale a "ou"
  - TO e DOWNTO delimitam faixas de opções
- **Opções restantes**: palavra reservada OTHERS
- **Formato da construção**:

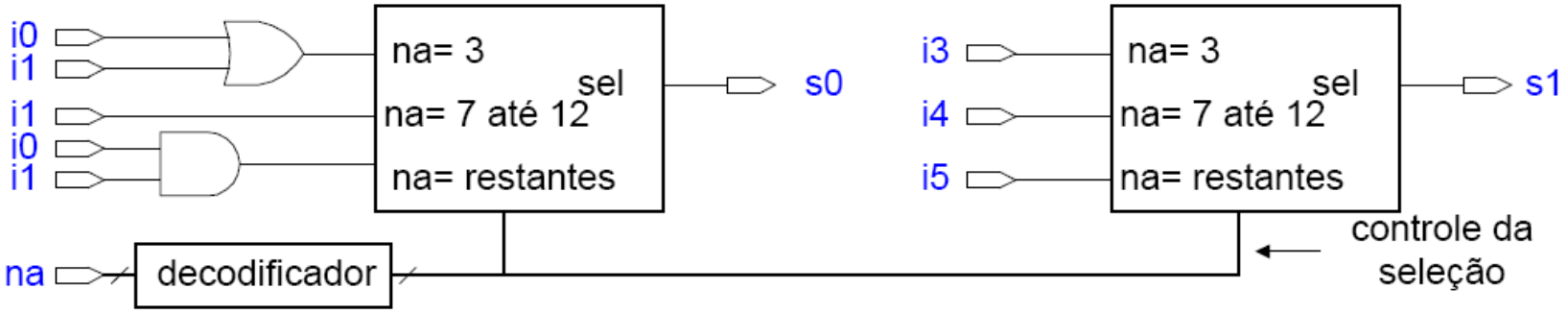
```
CASE expressao_escolha IS                                    -- expressao_escolha =

  WHEN condicao_1                => comando_b; comando_c;          -- condicao_1

  WHEN condicao_2 | condicao_3   => comando_d;        -- condicao_2 ou condicao_3

  WHEN condicao_4 TO condicao_9  => comando_d;       -- condicao_4 ate condicao_9

  WHEN OTHERS                    => comando_e; comando_f;  -- condicoes restantes
END CASE;
```

Construção CASE WHEN

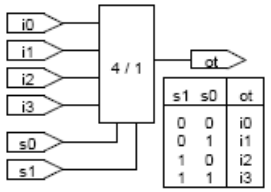
- **Exemplo**:

```
CASE na IS
  WHEN 3        =>  s0 <= i0 OR i1;    s1 <= i3;
  WHEN 7 TO 12  =>  s0 <= i1;          s1 <= i4;
  WHEN OTHERS   =>  s0 <= i0 AND i1;   s1 <= i5;
END CASE;
```

- **Circuito equivalente**:



Exemplo: - circuito de seleção - CASE WHEN



- **Comando seqüencial**
- **Nota**: s1 s2 agrupados no sinal sel

```
 1 ENTITY mux_5a IS
 2   PORT (i0, i1, i2, i3  :  IN  BIT;
 3         s1, s0          :  IN  BIT;
 4         ot              :  OUT BIT);
 5 END mux_5a;
 6
 7 ARCHITECTURE teste OF mux_5a IS
 8   SIGNAL sel: BIT_VECTOR(1 DOWNTO 0);
 9 BEGIN
10   sel <= s1 & s0;
11   abc: PROCESS (i0, i1, i2, i3, sel) --sinal "sel" inserido na lista
12   BEGIN
13     CASE sel IS
14        WHEN "00"   => ot <= i0;
15        WHEN "01"   => ot <= i1;
16        WHEN "10"   => ot <= i2;
17        WHEN OTHERS => ot <= i3;
18     END CASE;
19   END PROCESS abc;
20 END teste;
```

Comando WAIT

- **WAIT**: suspende a execução de um processo
- **Opções**: 3 formas ou uma combinação delas

- **WAIT ON**: equivalente a uma lista de sensibilidade
  processo suspenso até a mudança de valor sinais relacionados
- **WAIT UNTIL**: processo suspenso aguarda condição booleana
- **WAIT FOR**: processo suspenso por período de tempo

```
WAIT ON lista_de_sensibilidade;
WAIT UNTIL condicao_booleana;
WAIT FOR expressao_de_tempo;

WAIT ON lista_sensibilidade UNTIL condicao_booleana FOR expressao_de_tempo;
```

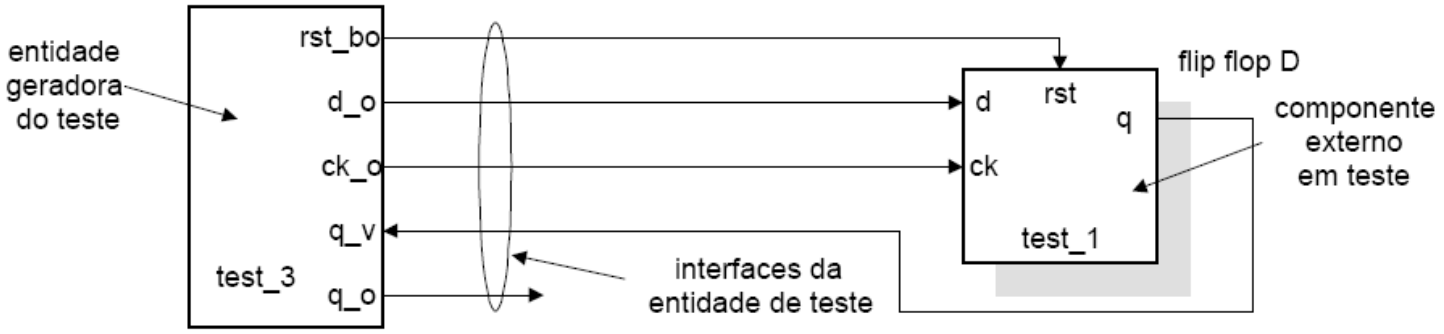
Comando WAIT - exemplos

- **1° caso:** processo suspenso
  - aguarda mudança de valor em sinal_a ou sinal_b
- **2° caso**: processo suspenso
  - aguarda a condição sinal_c =1 verdadeira
- **3° caso**: processo suspenso
  - aguarda um período de tempo de 100 ns
- **4° caso**: processo suspenso
  - aguarda a condição sinal_d =1 por um período de no máximo 50 ns

```
WAIT ON sinal_a, sinal_b;                -- 1o caso
WAIT UNTIL sinal_c = '1';                -- 2o caso
WAIT FOR 100 ns;                         -- 3o caso
WAIT UNTIL sinal_d = '0' FOR 50 ns;      -- 4o caso
```

Comando WAIT - observações

- **Não é permitido**: comando WAIT e lista de sensibilidade
- **Possível**: mais de um comando WAIT num processo
- **Síntese**:
  - WAIT ON (equivalente a uma lista de sensibilidade)
  - WAIT UNTIL (inferência de registradores - capítulo 6)
- **Aplicações das outras formas**:
  - geração de estímulos de teste

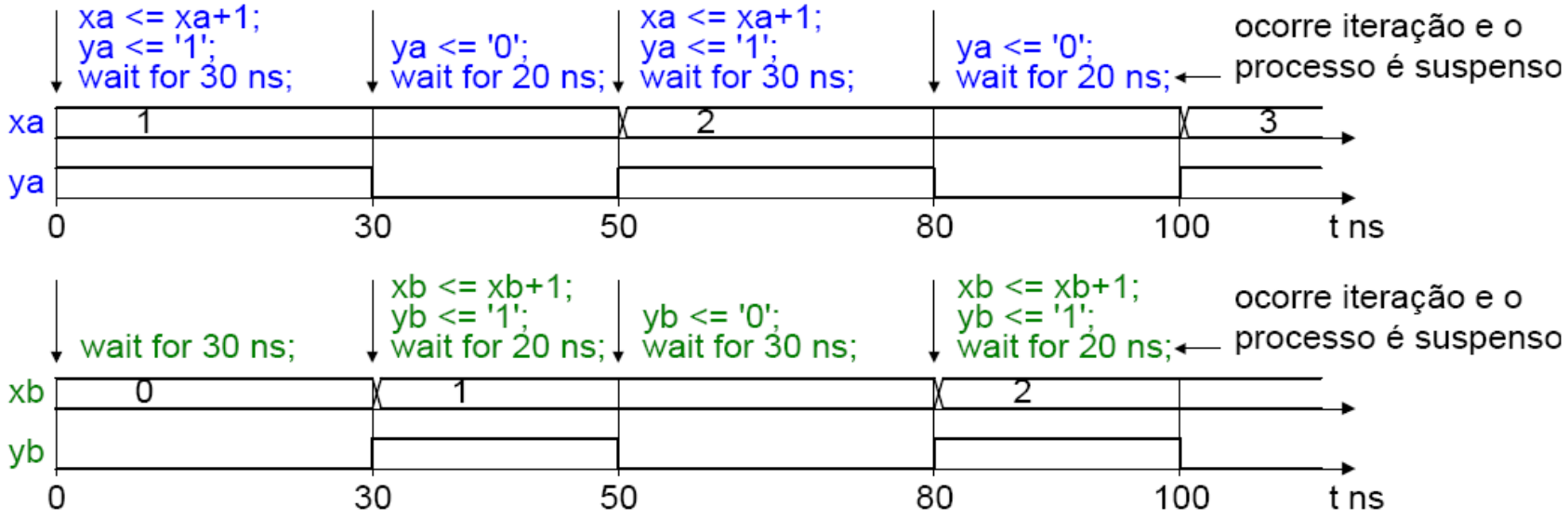


Comando WAIT - exemplo de operação

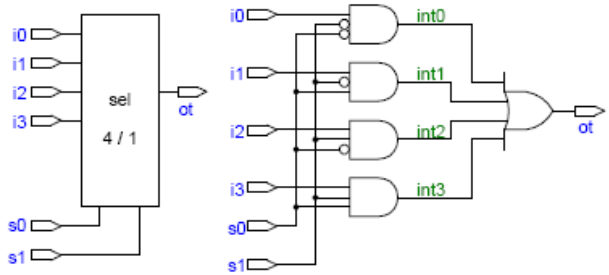
Note: não existe lista de sensibilidade processo executado continuamente

```
 8   abc: PROCESS
 9   BEGIN
10     xa <= xa +1;
11     ya <= '1';
12     WAIT FOR 30 ns;
13     ya <= '0';
14     WAIT FOR 20 ns;
15   END PROCESS abc;
```

```
16   cde: PROCESS
17   BEGIN
18     WAIT FOR 30 ns;
19     xb <= xb +1;
20     yb <= '1';
21     WAIT FOR 20 ns;
22     yb <= '0';
23   END PROCESS cde;
```



Exemplo: - circuito de seleção - WAIT



- **Processo sem lista de sensibilidade**
- **Execução do processo suspenso na linha 12**
  - sinais: int0 int1 int2 int3
    incluídos na lista de sensibilidade

```
 1 ENTITY mux_p1 IS
 2   PORT (i0, i1, i2, i3    :  IN  BIT;  -- entradas
 3         s0, s1            :  IN  BIT;  -- selecao
 4         ot                :  OUT BIT); -- saida
 5 END mux_p1;
 6
 7 ARCHITECTURE teste OF mux_p1 IS
 8   SIGNAL int0, int1, int2, int3 : BIT;
 9 BEGIN
10   PROCESS
11   BEGIN
12     ot <= int0 OR int1 OR int2 OR int3;
13     int0 <= i0 AND NOT s1 AND NOT s0;
14     int1 <= i1 AND NOT s1 AND     s0;
15     int2 <= i2 AND     s1 AND NOT s0;
16     int3 <= i3 AND     s1 AND     s0;
17     WAIT ON i0, i1, i2, i3, s0, s1, int0, int1, int2, int3;
18   END PROCESS;
19 END teste;
```

Cuidados na descrição: comparações WHEN ELSE e IF ELSE

- **Construção WHEN ELSE**:
  - concorrente
  - 1ª condição válida: determina expressão transferida
- **Construção IF ELSE**:
  - seqüencial
  - 1ª condição válida: determina um conjunto de comandos a ser executado

```
sinal_destino <= expressao_a  WHEN opcao_1 ELSE
                 expressao_b  WHEN opcao_2 ELSE
                 expressao_c;
```

```
IF  condicao_1 THEN
  comando_a;
ELSIF condicao_2 THEN
  comando_c;
ELSE
  comando_d;
END IF;
```

Cuidados na descrição: comparações WITH SELECT e CASE WHEN

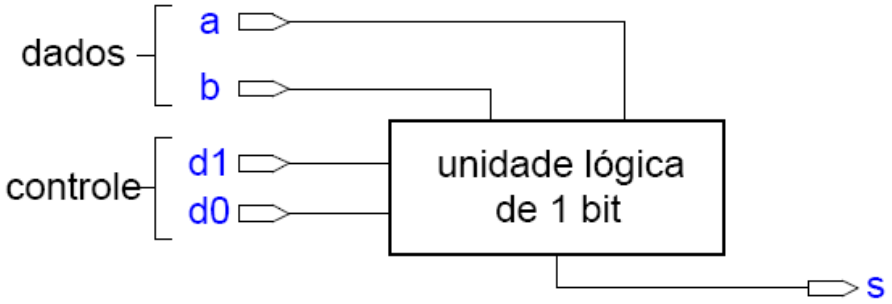
- **Construção WITH SELECT**:
  - concorrente
  - todas condições devem ser relacionadas
    - cada condição determina uma expressão transferida
- **Construção CASE WHEN**:
  - seqüencial
  - todas condições devem ser relacionadas
    - cada condição determina um conjunto de comandos a ser executado

```
WITH expressao_escolha SELECT
 sinal_destino <= expressao_a WHEN opcao_1,
                  expressao_b WHEN opcao_2,
                  expressao_c WHEN OTHERS;
```

```
CASE expressao IS
   WHEN condicao_1  => comando_a;
   WHEN OTHERS      => comando_e; comando_f;
END CASE;
```

Cuidados na descrição: IF ELSE e CASE WHEN aninhadas

- **IF ELSE e CASE WHEN são comando seqüenciais**:
  - possível aninhar vários comando
- **Este estilo pode ser evitado**: objetivo melhorar a clareza do código
- **Exemplo**:
  - descrição para uma Unidade Lógica Aritmética de 1 bit
    - operações: complemento de a
      "ou exclusivo" entre a e b
      "e" entre a e b
      "ou" entre a e b



s	d1	d0
$\overline{a}$	0	0
$a \oplus b$	0	1
a . b	1	0
a + b	1	1

- **Descrições - IF ELSE com e sem aninhamento**
  - sinais que definem a decisão d0 e d1
    - concatenação dos sinais d0 e d1 evita o aninhamento
    - código mais claro e conciso


```
 1 ENTITY t1_if1 IS
 2 PORT (d1,d0 : IN  BIT;
 3       a, b  : IN  BIT;
 4       s     : OUT BIT);
 5 END t1_if1;
 6
 7 ARCHITECTURE teste OF t1_if1 IS
 8 BEGIN
 9  PROCESS (d1,d0,a,b)
10  BEGIN
11   IF d1 ='0'  THEN
12    IF d0 ='0' THEN s <= NOT a;
13    ELSE            s <= a XOR b;
14    END IF;
15   ELSE
16    IF d0 ='0' THEN s <= a AND b;
17    ELSE            s <= a OR b;
18    END IF;
19   END IF;
20  END PROCESS;
21 END teste;
```

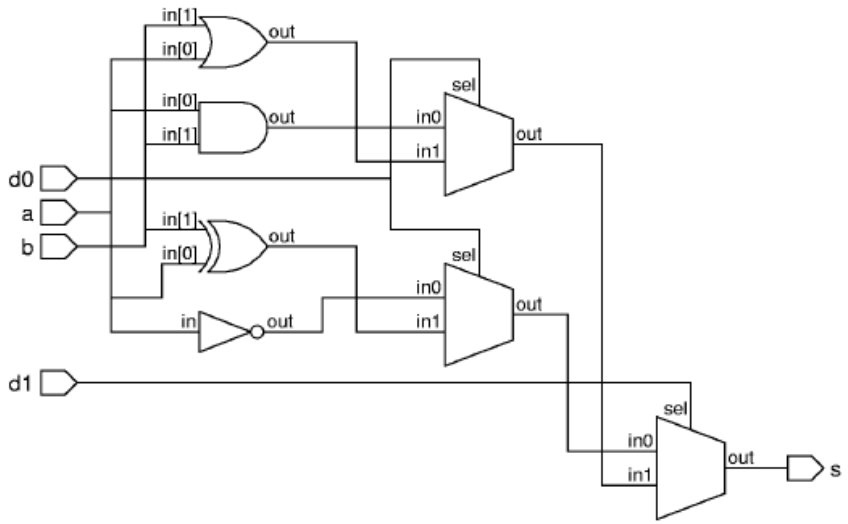
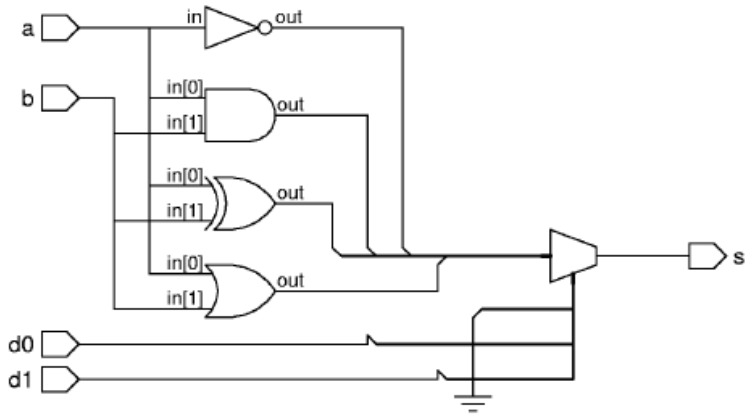
```
ENTITY t1_if2 IS
PORT (d1,d0 : IN  BIT;
      a, b  : IN  BIT;
      s     : OUT BIT);
END t1_if2;

ARCHITECTURE teste OF t1_if2 IS
  SIGNAL d : BIT_VECTOR(1 DOWNTO 0);
BEGIN
 d <= d1 & d0;
 PROCESS (d,a,b)
 BEGIN
  IF    d ="00" THEN s <= NOT a;
  ELSIF d ="01" THEN s <= a XOR b;
  ELSIF d ="10" THEN s <= a AND b;
  ELSE               s <= a OR b;
  END IF;
 END PROCESS;
END teste;
```

- **Descrições - CASE WHEN com e sem aninhamento**
  - concatenação dos sinais d1 d0 evita o aninhamento
  - novamente: código mais claro


```
 1 ENTITY t1_cas1 IS
 2 PORT (d1,d0 : IN  BIT;
 3       a, b  : IN  BIT;
 4       s     : OUT BIT);
 5 END t1_cas1;
 6
 7 ARCHITECTURE teste OF t1_cas1 IS
 8 BEGIN
 9  PROCESS (d1,d0,a,b)
10  BEGIN
11   CASE d1 IS
12    WHEN '0' =>
13     CASE d0 IS
14      WHEN '0' => s <= NOT a;
15      WHEN '1' => s <= a XOR b;
16     END CASE;
17    WHEN '1' =>
18     CASE d0 IS
19      WHEN '0' => s <= a AND b;
20      WHEN '1' => s <= a OR b;
21     END CASE;
22   END CASE;
23  END PROCESS;
24 END teste;
```

```
ENTITY t1_cas2 IS
PORT (d1,d0 : IN  BIT;
      a, b  : IN  BIT;
      s     : OUT BIT);
END t1_cas2;

ARCHITECTURE teste OF t1_cas2 IS
  SIGNAL d : BIT_VECTOR(1 DOWNTO 0);
BEGIN
 d <= d1 & d0;
 PROCESS (d,a,b)
 BEGIN
  CASE d IS
   WHEN "00" => s <= NOT a;
   WHEN "01" => s <= a XOR b;
   WHEN "10" => s <= a AND b;
   WHEN "11" => s <= a OR b;
  END CASE;
 END PROCESS;
END teste;
```

- **Resultado da síntese - nível RTL**

IF ELSE e CASE WHEN com aninhamento	IF ELSE e CASE WHEN sem aninhamento
- tende a inserir um circuito de seleção por nível de aninhamento	- decisão por um único circuito de seleção
	
t1_if1 t1_cas1	t1_if2 t1 cas2

Cuidados na descrição: emprego IF ELSE e CASE WHEN

- **Construção IF ELSE:**
  - possível uma nova expressão de escolha a cada estágio
  - excesso de liberdade pode levar a erros:
    exemplo: nem todas condições cobertas por descuido na descrição
    criação de elementos de memória desnecessários

- **Construção CASE WHEN:**
  - expressão de escolha única
  - condições de escolha não cobertas:
    mensagem de erro na etapa de compilação

- **Sempre que possível:**
  - empregar a construção CASE WHEN

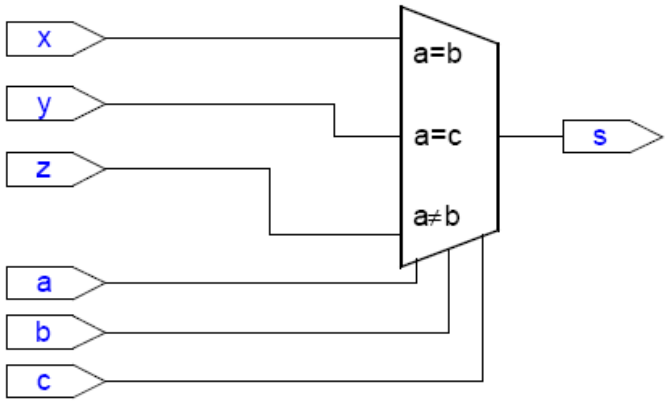
Cuidados na descrição: emprego IF ELSE e CASE WHEN

- Especificação do problema

se a = b então saída s = x

se a = c então saída s = y

se b ≠ c então saída s = z



- Possível descrição empregando IF ELSE

```
 1 ENTITY dont_if3 IS
 2 PORT (x,y,z  : IN  BIT;
 3       a,b,c  : IN  BIT;
 4       s      : OUT BIT);
 5 END dont_if3;
 6
 7 ARCHITECTURE teste OF dont_if3 IS
 8 BEGIN
 9   PROCESS (a,b,c,x,y,z)
10   BEGIN
11     IF    a=b  THEN s <= x;
12     ELSIF a=c  THEN s <= y;
13     ELSIF b/=c THEN s <= z;
14     END IF;
15   END PROCESS;
16 END teste;
```

A descrição segue a especificação do problema?

- **Analisando a especificação do problema**: levantar uma tabela das condições
- **Lembrando**: IF ELSE a seqüência define a prioridade na escolha
  - linha 11: verificada as condições 0, 1, 6 e 7 (maior prioridade)
  - linha 12: verificada as condições 2 e 5
  - linha 13: deveria verificar 3 e 4 (nunca ocorre esta opção)

Condições possíveis levantadas

	a	b	c	a=b	a=c	b≠c
0	0	0	0	V	V	F
1	0	0	1	V	F	V
2	0	1	0	F	V	V
3	0	1	1	F	F	F
4	1	0	0	F	F	F
5	1	0	1	F	V	V
6	1	1	0	V	F	V
7	1	1	1	V	V	F

V = verdadeiro
F = falso

```
 1 ENTITY dont_if3 IS
 2 PORT (x,y,z  : IN  BIT;
 3       a,b,c  : IN  BIT;
 4       s      : OUT BIT);
 5 END dont_if3;
 6
 7 ARCHITECTURE teste OF dont_if3 IS
 8 BEGIN
 9   PROCESS (a,b,c,x,y,z)
10   BEGIN
11     IF    a=b  THEN s <= x;
12     ELSIF a=c  THEN s <= y;
13     ELSIF b/=c THEN s <= z;
14     END IF;
15   END PROCESS;
16 END teste;
```

A descrição segue a especificação do problema?

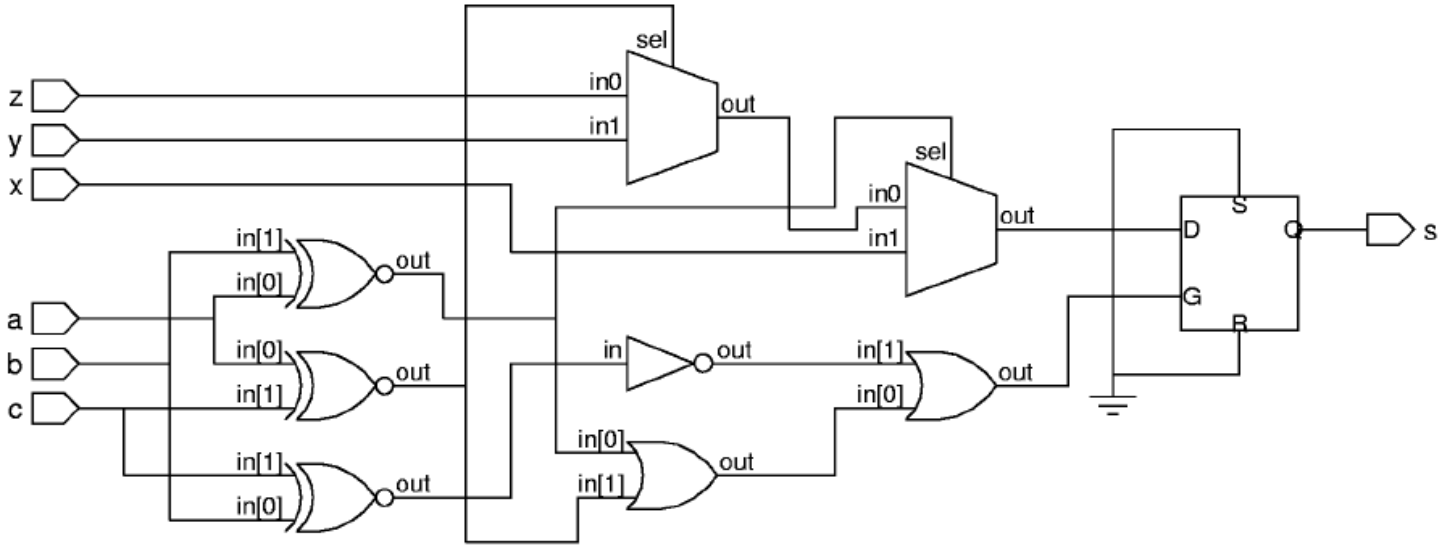
- **Não**
- **Devido a flexibilidade da construção IF ELSE**
  - problemas da especificação foram encobertos
- **Opção pela construção CASE WHEN** :
  - falha na especificação seria detectada
  - necessário verificar todas opções
- **Neste caso o problema pode ser facilmente detectado**
- **Descrições com muitas decisões podem encobrir problemas**

Síntese da descrição

- ***latch*** **tipo D inserido**
- **Mantém o valor nas condições 3 e 4** (nunca atingidas)

Condições possíveis levantadas

	a	b	c	a=b	a=c	b≠c
0	0	0	0	V	V	F
1	0	0	1	V	F	V
2	0	1	0	F	V	V
3	**0**	**1**	**1**	F	F	F
4	**1**	**0**	**0**	F	F	F
5	1	0	1	F	V	V
6	1	1	0	V	F	V
7	1	1	1	V	V	F



Leitura adicional:

- Expressão de escolha das construções WITH SELECT e CASE WHEN

- Comando NULL