Dispositivos Lógicos Programáveis

Conteúdo do dia:

- - Comandos Sequenciais Básicos
- Estrutura IF ELSE;
- Estrutura CASE WHEN;

Comando *PROCESS*

Como vimos anteriormente, é um *comando concorrente* que permite criar uma área de execução de *comandos sequenciais*.

Uma descrição pode conter vários *process*, sendo todos executados de maneira concorrente.

Lista de Sensibilidade do comando *PROCESS*

É formada por sinais que, quando alterados, fazem com que o comando *process* seja executado.

Ao término da execução, o processo fica suspenso até a ocorrência de uma nova alteração de valor de um dos sinais relacionados na lista.

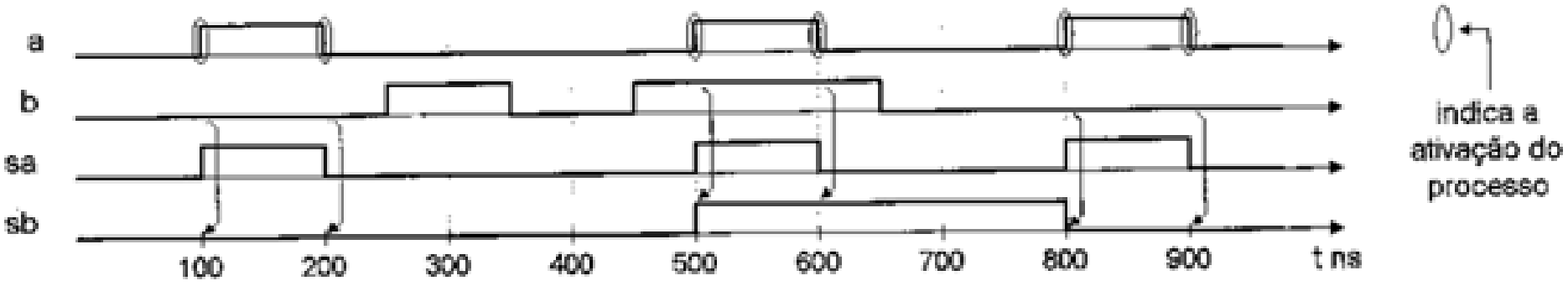
Lista de Sensibilidade do comando *PROCESS*

Exemplo de lista de sensibilidade:

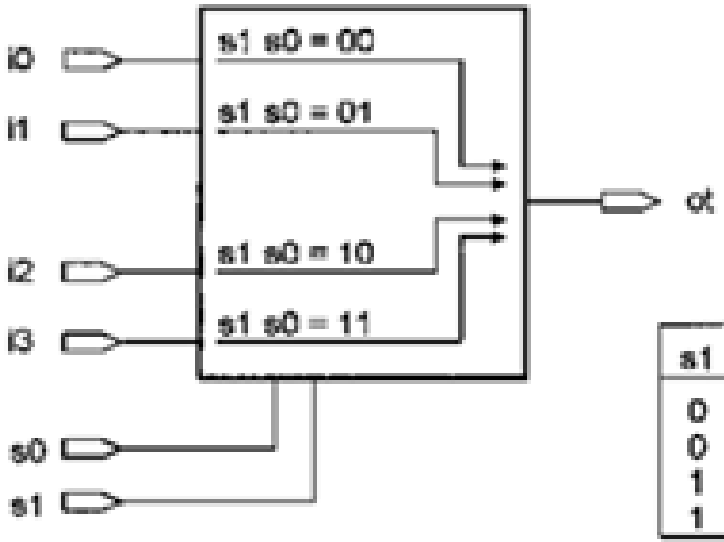
```
 1 ENTITY sens_tes IS
 2   PORT (a,  b   :  IN  BIT;
 3         sa, sb  :  OUT BIT);
 4 END sens_tes;
 5
 6 ARCHITECTURE teste OF sens_tes IS
 7 BEGIN
 8   abc: PROCESS (a) -- executado na alteracao do valor de 'a'
 9   BEGIN
10     sa <= a;
11     sb <= b;
12   END PROCESS abc;
13 END teste;
```

Lista de Sensibilidade do comando *PROCESS*

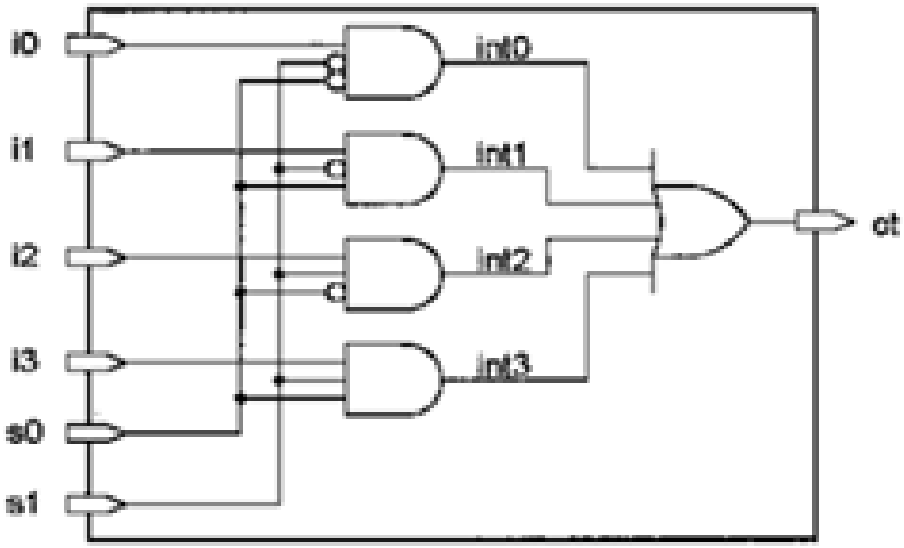
Exemplo de lista de sensibilidade:



Exercício 1 – Descreve este circuito utilizando o comando PROCESS. Considere todos os sinais de entrada na lista de sensibilidades.



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



Exercício 1 – Descreve este circuito utilizando o comando PROCESS. Considere todos os sinais de entrada na lista de sensibilidades.

Resposta:

```
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)
11   BEGIN
12             int0 <= i0 AND NOT s1 AND NOT s0;
13             int1 <= i1 AND NOT s1 AND     s0;
14             int2 <= i2 AND     s1 AND NOT s0;
15             int3 <= i3 AND     s1 AND     s0;
16                 ot <= int0 OR int1 OR int2 OR int3;
17   END PROCESS;
18 END teste;
```

Notem a diferença entre a descrição utilizando a linguagem concorrente e a linguagem sequencial.

```
ENTITY mux00 IS
  PORT (i0, i1, i2, i3  :  IN   BIT;  --entradas
        s0, s1          :  IN   BIT;  --seleção
        ot              :  OUT  BIT); --saida
END mux00;
ARCHITECTURE nivel_logico OF mux00 IS
  SIGNAL  int0, int1, int2, int3: BIT; --sinais
  BEGIN
       ot   <= int0 OR int1 OR int2 OR int3;
       int0 <= i0 AND NOT s1 AND NOT s0;
       int1 <= i1 AND NOT s1 AND     s0;
       int2 <= i2 AND     s1 AND NOT s0;
       int3 <= i3 AND     s1 AND     s0;
END nivel_logico;
```

```
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)
11   BEGIN
12             int0 <= i0 AND NOT s1 AND NOT s0;
13             int1 <= i1 AND NOT s1 AND     s0;
14             int2 <= i2 AND     s1 AND NOT s0;
15             int3 <= i3 AND     s1 AND     s0;
16                  ot <= int0 OR int1 OR int2 OR int3;
17   END PROCESS;
18 END teste;
```

Comando *IF ELSE*

A construção "IF ELSE" permite a execução condicional de um ou mais comandos sequenciais, de acordo com uma lista de condições.

As condições são avaliadas na ordem de apresentação do código.

Comando *IF ELSE*

O comando IF inicia a lista de condições e pode ser seguido por cláusulas ELSIF contendo, também, condições a serem verificadas.

Caso nenhuma condição verdadeira for encontrada, e existir uma cláusula ELSE, o conjunto de comandos que a segue será executado.

As cláusulas ELSIF e ELSE são opcionais, e o número de cláusulas ELSIF não é limitado.

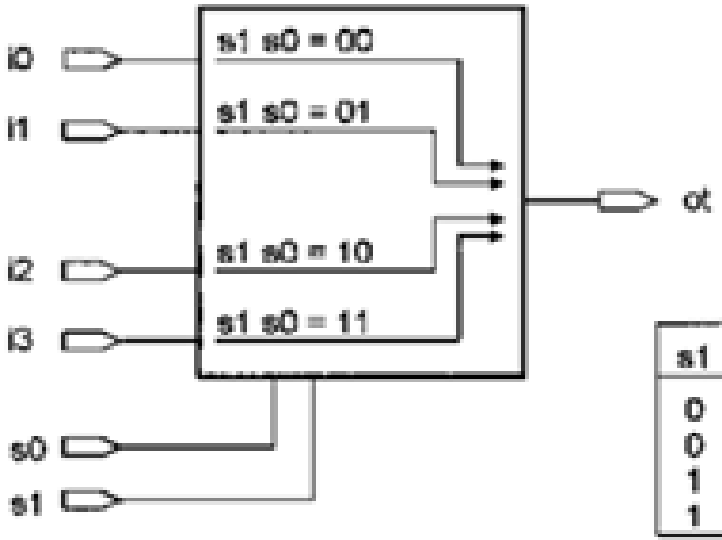
Comando *IF ELSE*

```
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;
```

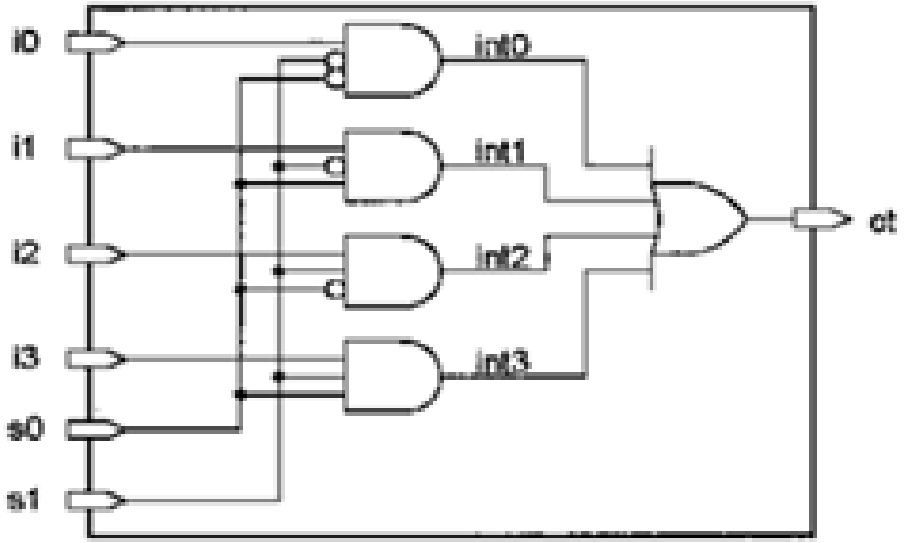
Comando *IF ELSE*

```
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;
```

Exercício 2 – Descreve este circuito utilizando os comandos PROCESS e o comando IF ELSE. Considere todos os sinais de entrada na lista de sensibilidades.



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



Exercício 2 – Descreve este circuito utilizando os comandos PROCESS e o comando IF ELSE. Considere todos os sinais de entrada na lista de sensibilidades.

Resposta:

```
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;
```

Comando *CASE WHEN*

Permite a execução condicional de um ou mais comandos sequenciais, conforme o valor de uma expressão.

Cada condição define um conjunto de comandos sequenciais a ser executado. Todas as condições possuem a mesma prioridade.

Comando *CASE WHEN*

É possível agrupar as condições através do delimitador "|".

As palavras reservadas TO e DOWNTO podem ser empregadas para delimitar uma faixa de condições.

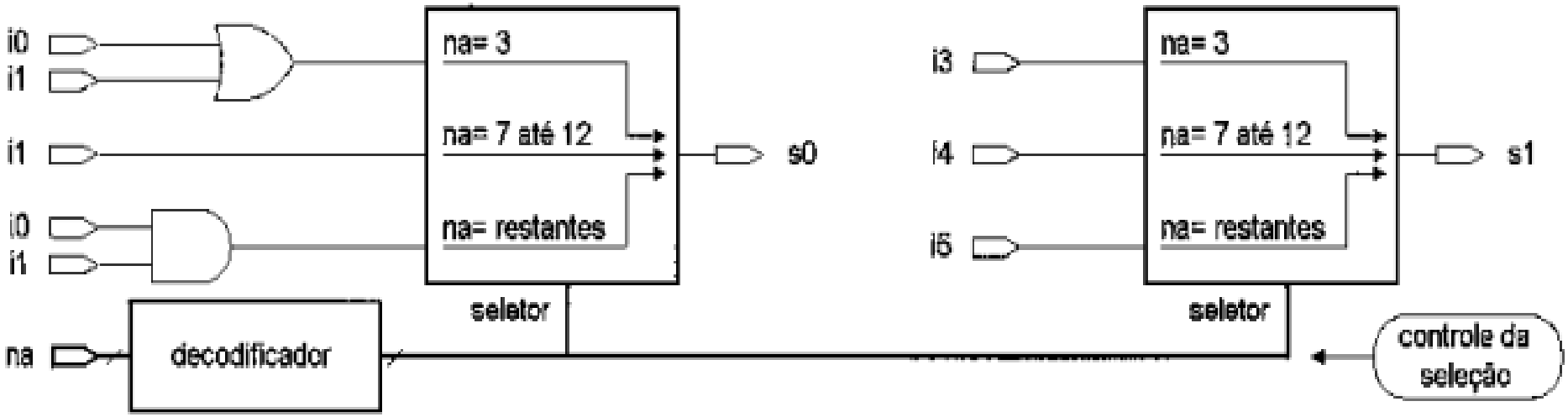
A palavra OTHERS pode ser empregada na última condição para agrupar as condições não relacionais na lista.

Comando *CASE WHEN*

```
CASE expressao_de_escolha IS                    -- expressao_de_escolha =
  WHEN condicao_1                => comando_a;                  -- condicao_1
  WHEN condicao_2                => comando_b; comando_c;       -- condicao_2
  WHEN condicao_3 |  condicao_4  => comando_d;                  -- condicao_3 ou condicao_4
  WHEN condicao_5 TO condicao_9  => comando_d;                  -- condicao_5 ate condicao_9
  WHEN OTHERS                    => comando_e; comando_f;       -- condicoes restantes
END CASE;
```

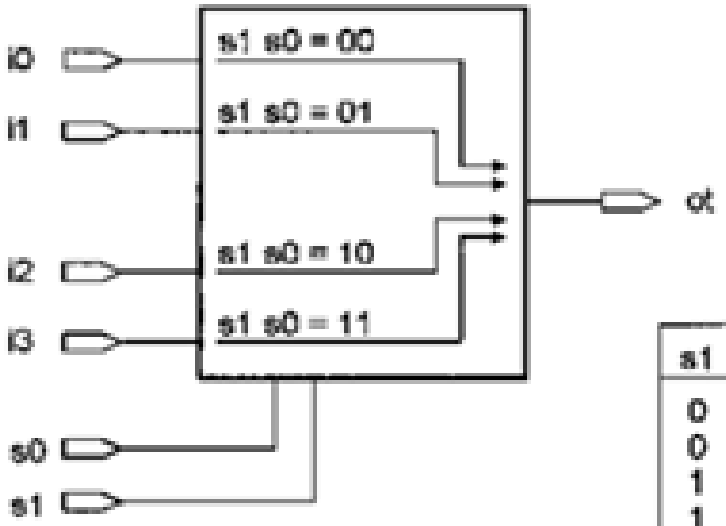
Comando *CASE WHEN*

Exemplo de aplicação:

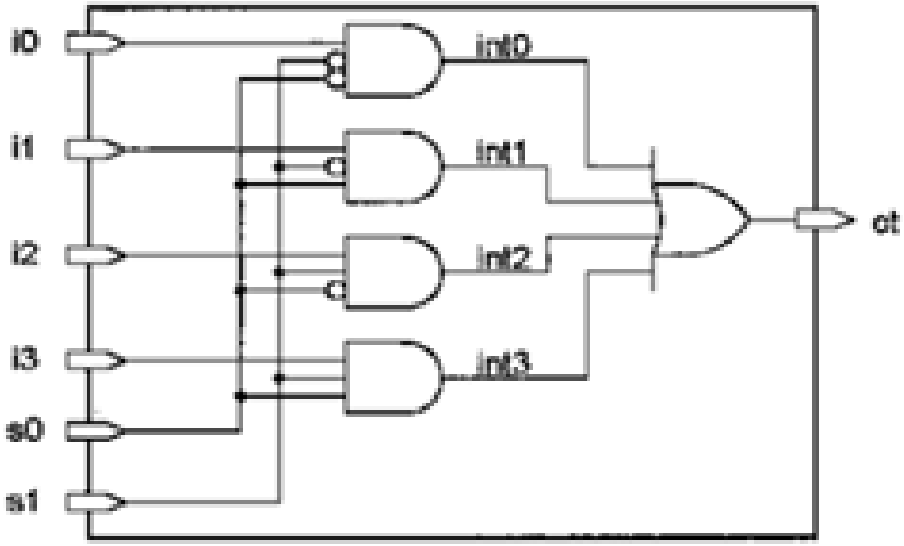



```
CASE na IS
  WHEN 3        =>  s0 <= i0 OR i1;    s1 <= i3;
  WHEN 7 TO 12  =>  s0 <= i1;          s1 <= i4;
  WHEN OTHERS   =>  s0 <= i0 AND i1;   s1 <= i5;
END CASE;
```

Exercício 3 – Descreve este circuito utilizando os comandos PROCESS e o comando CASE WHEN. Considere todos os sinais de entrada na lista de sensibilidades.



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



Exercício 3 – Descreve este circuito utilizando os comandos PROCESS e o comando CASE WHEN. Considere todos os sinais de entrada na lista de sensibilidades.

Resposta:

```
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;
```