Comandos concorrentes básicos

Tópicos

- **O atraso delta**
- **Atribuição de valor para um sinal**
- **Construção WHEN ELSE**
- **Construção WITH SELECT**
- **Comando BLOCK**
- **Comando PROCESS**
- **Cuidados na descrição**
  - Comparações entre WHEN ELSE e WITH SELECT
  - Criação de *latch* com WHEN ELSE e WITH SELECT

Atraso delta Δ

- **Região de código concorrente:**

  a ordem dos comandos não influencia no resultado final da simulação

- **Como é feito:**

  a avaliação dos eventos é ordenada por um atraso interno Δ

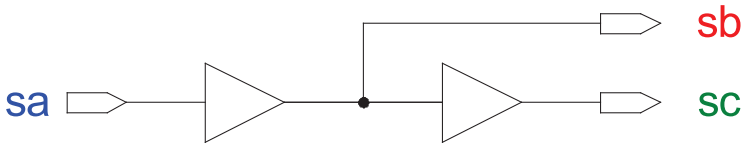
- **O atraso Δ é nulo: Δ = 0**

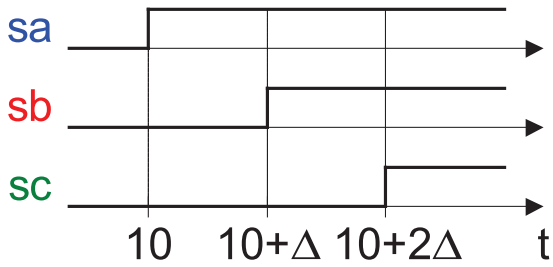
- **Exemplo:**

```
 7 BEGIN
 8   sc <= sb;    -- transferido 1 delta apos
 9   sb <= sa;    -- transferido 1 delta apos
10 END teste;
```

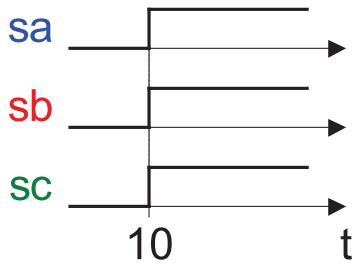
iterações

tempo	sa	sb	sc
0	0	0	0
10	1	0	0
10+Δ	1	1	0
10+2Δ	1	1	1





como Δ=0

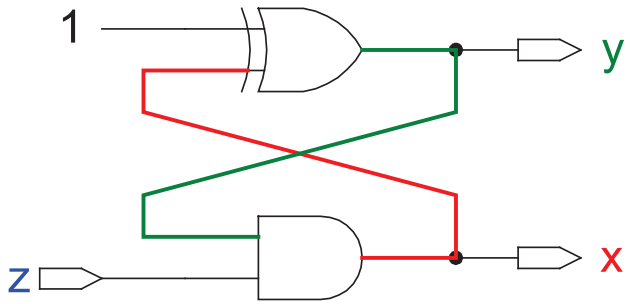


Atraso delta Δ

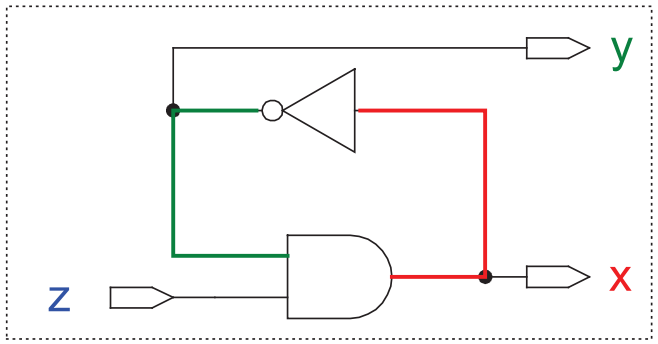
- **Considerando outro exemplo**

```
 1 ENTITY c04_delt IS
 2   PORT (z        :  IN     BIT;
 3         x,  y    :  BUFFER BIT);
 4 END c04_delt;
 5
 6 ARCHITECTURE teste OF c04_delt IS
 7 BEGIN
 8   x <=  z  AND y;
 9   y <= '1' XOR x;
10 END teste;
```

- **Alterando a entrada z de 0 → 1, uma condição estável nunca é atingida**



circuito proposto



circuito equivalente

iterações

tempo	z	x	y
0	0	0	1
10	1	0	1
10+Δ	1	1	1
10+2Δ	1	1	0
10+3Δ	1	0	0
10+4Δ	1	0	1
10+5Δ	1	1	1

Atribuição de valor para um sinal

- **Pode ocorrer em regiões: código sequencial código concorrente**

- **A atribuição emprega o delimitador: <=**

- **Exemplo:**

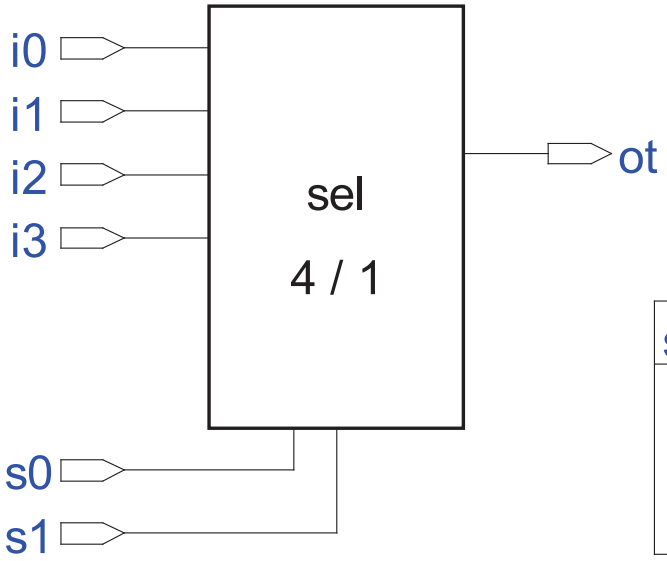
```
sinal_destino <= expressao;
```

- **A informação pode ser originada:**
  - expressão
  - sinal

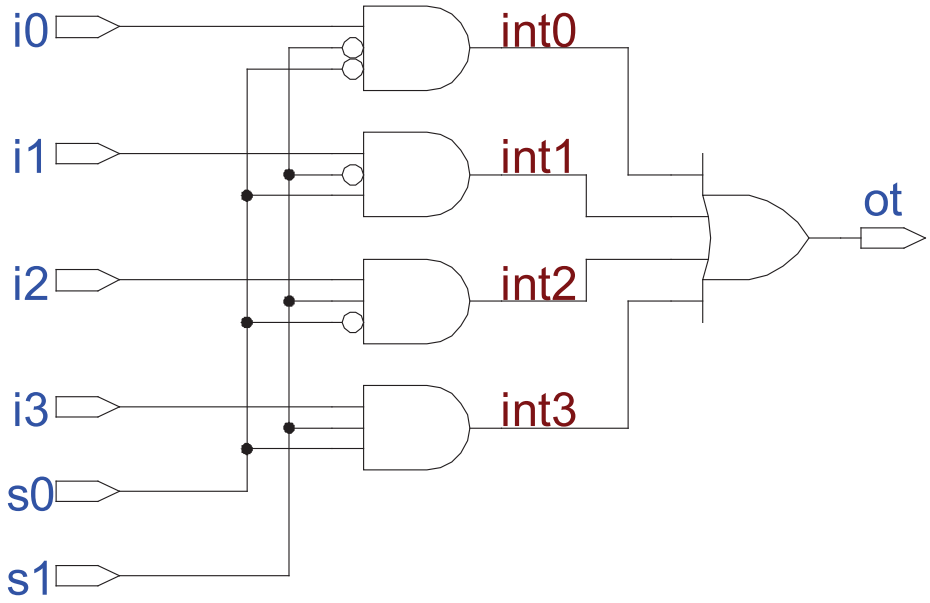
- **Nota: tipo sinal destino = tipo resultante da expressão**

Descrição de um circuito de seleção

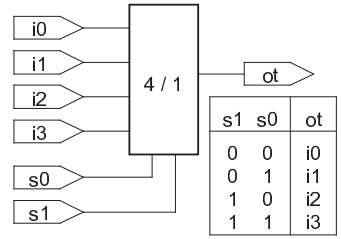
- **Entradas:** i0, i1, i2 e i3
- **Saída:** ot
- **Controle da seleção:** s0 e s1



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

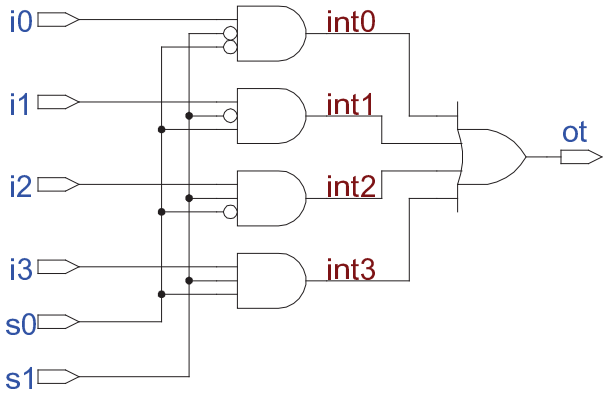


Exemplo: descrição do circuito de seleção

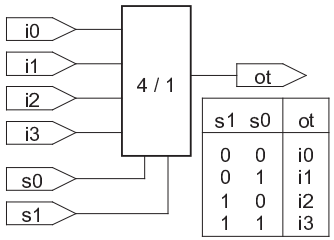


- **A descrição emprega uma única expressão**
- **Observar:** uso de parênteses → AND e OR têm precedência igual

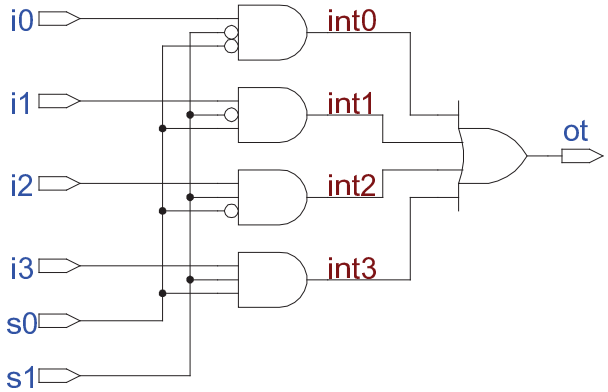
```
 1 ENTITY mux_0 IS
 2   PORT (i0, i1, i2, i3    :  IN  BIT;  -- entradas
 3         s0, s1            :  IN  BIT;  -- selecao
 4         ot                :  OUT BIT); -- saida
 5 END mux_0;
 6
 7 ARCHITECTURE nivel_logico OF mux_0 IS
 8 BEGIN
 9   ot <= (i0 AND NOT s1 AND NOT s0) OR
10         (i1 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 nas linhas 11, 12, 13 e 14




```
 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**:

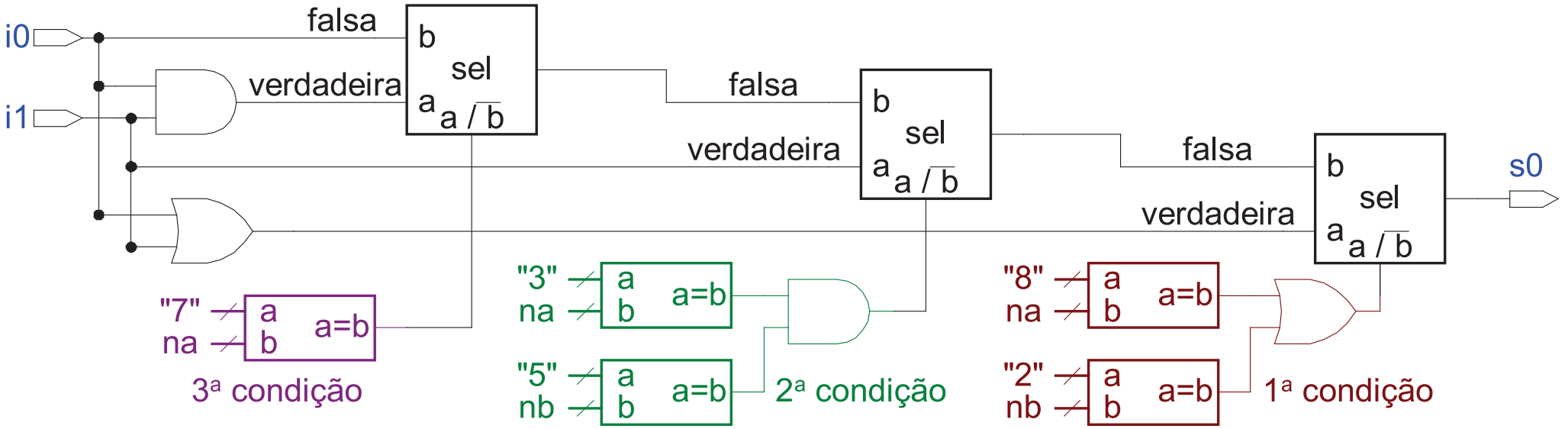
```
sinal_destino <= expressao_a  WHEN condicao_1 ELSE
                 expressao_b  WHEN condicao_2 ELSE
                 expressao_c;
```

Construção WHEN ELSE

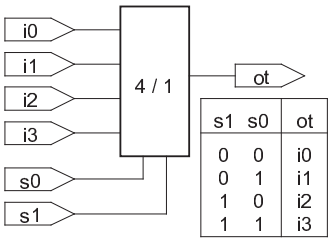
- **Exemplo:**

```
s0 <= i0 OR i1   WHEN na= 8 OR  nb= 2 ELSE    -- primeira condicao verdadeira
      i1         WHEN na= 3 AND nb= 5 ELSE    -- segunda  condicao verdadeira
      i0 AND i1  WHEN na= 7           ELSE    -- terceira condicao verdadeira
      i0;
```

- **Circuito equivalente:**



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 as condições da expressão de escolha devem ser consideradas**
  - não existe uma prioridade como na construção WHEN ELSE
- **As opções podem ser agrupadas:** o caractere | 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

- **A 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
```

- **Todas as condições da expressão de escolha devem ser determinadas na compilação**

Construção WITH SELECT

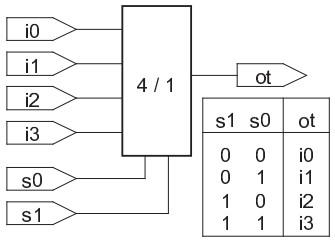
- **Exemplo de retorno da expressão de escolha:**
  - expressão de escolha: sinal s0 – retorna o tipo CHARACTER
  - exemplos de opções de escolha:
    - único valor, um valor ou outro, faixa de valores, valores restantes

```
WITH s0 SELECT    -- s0 tipo CHARACTER
    x0 <= i0 AND i1 WHEN 'a',                  -- unico valor
          i0 OR  i1 WHEN 'b'| 'c',             -- um valor ou outro
          i0 XOR i1 WHEN 'd' TO 'g',           -- faixa de valores
          i1        WHEN 'x' DOWNTO 'k',       -- faixa de valores
          i0        WHEN OTHERS;               -- valores restantes
```

- **Exemplo de retorno da expressão de escolha:**
  - expressão de escolha: b1 AND b0 – retorna o tipo BIT (tipo discreto)

```
 WITH b1 AND b0 SELECT  -- b1 e b0 tipo BIT
    x1 <= i0 WHEN '0',
          i1 WHEN '1';
```

Exemplo: 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

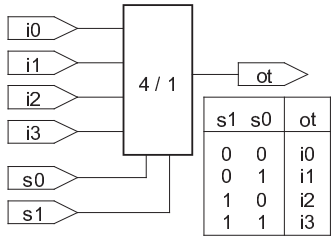
```
 1 ENTITY mux_9 IS
 2   PORT (i0, i1, i2, i3   : IN  BIT;
 3         s0, s1          : IN  BIT;
 4         ot              : OUT BIT);
 5 END mux_9;
 6
 7 ARCHITECTURE teste OF mux_9 IS
 8   SIGNAL sel : BIT_VECTOR (1 DOWNTO 0);
 9 BEGIN
10   sel <= s1 & s0;
11   WITH sel SELECT
12     ot <= i0 WHEN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

Comando BLOCK

- Objetivo:
  - delimitar regiões de código
- Aplicação:
  - facilitar o entendimento da descrição
  - declaração de sinais visíveis unicamente no interior do bloco
- Formato da construção:

```
nome_rotulo: BLOCK IS
   -- definicao de sinais visiveis no bloco
BEGIN
   -- comandos
   -- comandos
END BLOCK nome_rotulo;
```

Exemplo: circuito de seleção com dois blocos



- **Note**: sinal interno_def visível unicamente no bloco def

```
 7 ARCHITECTURE com_block OF mux_5 IS
 8   SIGNAL global      : BIT_VECTOR(0 TO 1);
 9 BEGIN
10   abc: BLOCK
11   BEGIN
12     ot <= i0 WHEN global = "00" ELSE
13           i1 WHEN global = "01" ELSE
14           i2 WHEN global = "10" ELSE
15           i3;
16   END BLOCK abc;
17
18   def: BLOCK
19     SIGNAL interno_def  : BIT_VECTOR(0 TO 1);  -- sinal visivel no bloco def
20   BEGIN
21     WITH sel SELECT
22       interno_def <= "00" WHEN 0,
23                      "01" WHEN 1,
24                      "10" WHEN 2,
25                      "11" WHEN OTHERS;
26     global <= interno_def;
27   END BLOCK def;
28 END com_block;
```

Comando PROCESS

- **Objetivo**: delimitar regiões de código sequencial
- **Início**: palavra reservada PROCESS
- **Lista de sensibilidade**: identifica que sinais ativam a execução do processo
- **Comandos sequenciais**: 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:**
  - a ordem das condições indica a prioridade
  - não é necessário apresentar todas as 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 as condições têm igual prioridade
  - é necessário apresentar todas as condições

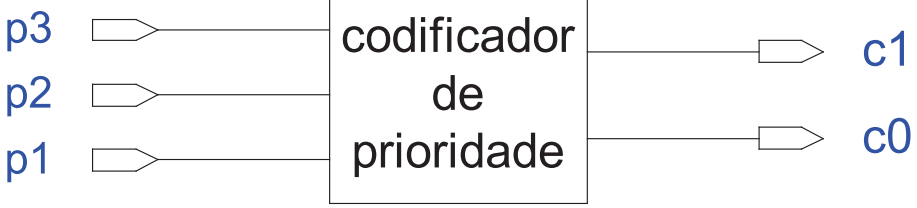
```
WITH expressao_escolha SELECT                -- expressao_escolha =
 sinal_destino <= expressao_a WHEN condicao_1,          -- condicao_1
                  expressao_b WHEN condicao_2,          -- condicao_2
                  expressao_e WHEN OTHERS;              -- condicoes restantes
```

Comparação entre as construções WHEN ELSE e WITH SELECT

- **Exemplo**: descrição de um decodificador de prioridade

  - três entradas: p3 maior prioridade p1 menor prioridade

  - saídas c1 e c0: código da entrada de maior prioridade ativa

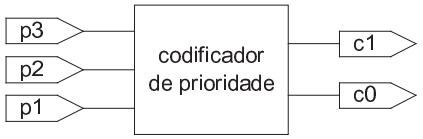


p3	p2	p1	c1	c0
1	-	-	1	1
0	1	-	1	0
0	0	1	0	1
0	0	0	0	0

- $\Rightarrow$ não importa

Comparação entre as construções WHEN ELSE e WITH SELECT

- **Exemplo**: decodificador de prioridade
  empregando a construção WHEN ELSE

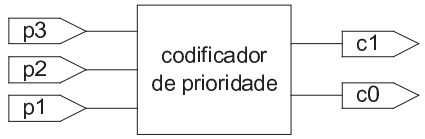


- **Neste caso**: descrição mais fácil
  - prioridade inerente da construção pode ser empregada
  - linha 8: condição de maior prioridade testada p(3)=1
  - linha 11: condição de menor prioridade

```
 1 ENTITY pr_cod1 IS
 2 PORT (p       : IN  BIT_VECTOR(3 DOWNTO 1);
 3       c       : OUT BIT_VECTOR(1 DOWNTO 0));
 4 END pr_cod1;
 5
 6 ARCHITECTURE teste OF pr_cod1 IS
 7 BEGIN                                -- p3 p1 p0     c1 c0
 8   c <= "11" WHEN p(3)='1' ELSE       --  1  -  -      1  1
 9        "10" WHEN p(2)='1' ELSE       --  0  1  -      1  0
10        "01" WHEN p(1)='1' ELSE       --  0  0  1      0  1
11        "00";                         --  0  0  0      0  0
12 END teste;
```

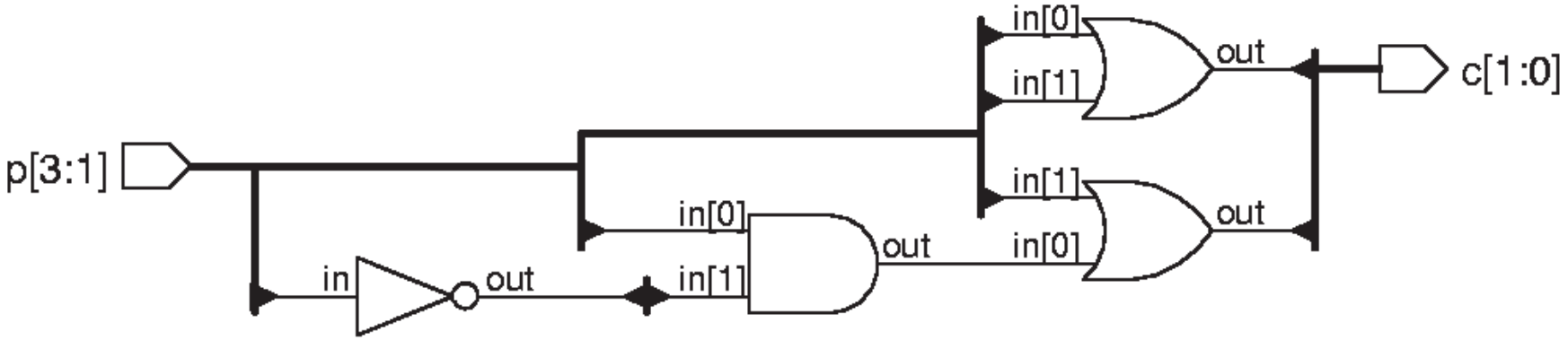
Comparação entre as construções WHEN ELSE e WITH SELECT

- **Circuito gerado com a construção WHEN ELSE**



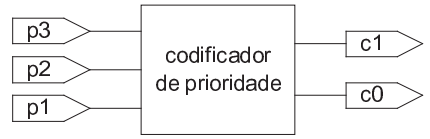
- **Nível RTL**

- **Resultado**: circuito mais compacto neste nível



Comparação entre as construções WHEN ELSE e WITH SELECT

- **Exemplo:** decodificador de prioridade
  empregando a construção WITH SELECT

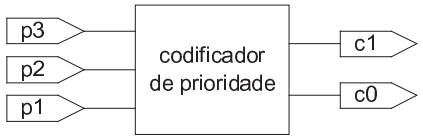


- **Neste caso**: a construção demanda um maior detalhamento
  - todas as condições devem ser especificadas
  - para cada condição de saída: relação das combinações possíveis de p

```
 1 ENTITY pr_cod2 IS
 2 PORT (p       : IN  BIT_VECTOR(3 DOWNTO 1);
 3       c       : OUT BIT_VECTOR(1 DOWNTO 0));
 4 END pr_cod2;
 5
 6 ARCHITECTURE teste OF pr_cod2 IS
 7 BEGIN
 8   WITH p SELECT                              -- p3 p1 p0   c1 c0
 9   c <= "11" WHEN "111"|"110"|"101"|"100",    --  1  -  -    1  1
10        "10" WHEN "011"|"010",                --  0  1  -    1  0
11        "01" WHEN "001",                      --  0  0  1    0  1
12        "00" WHEN "000";                      --  0  0  0    0  0
13 END teste;
```

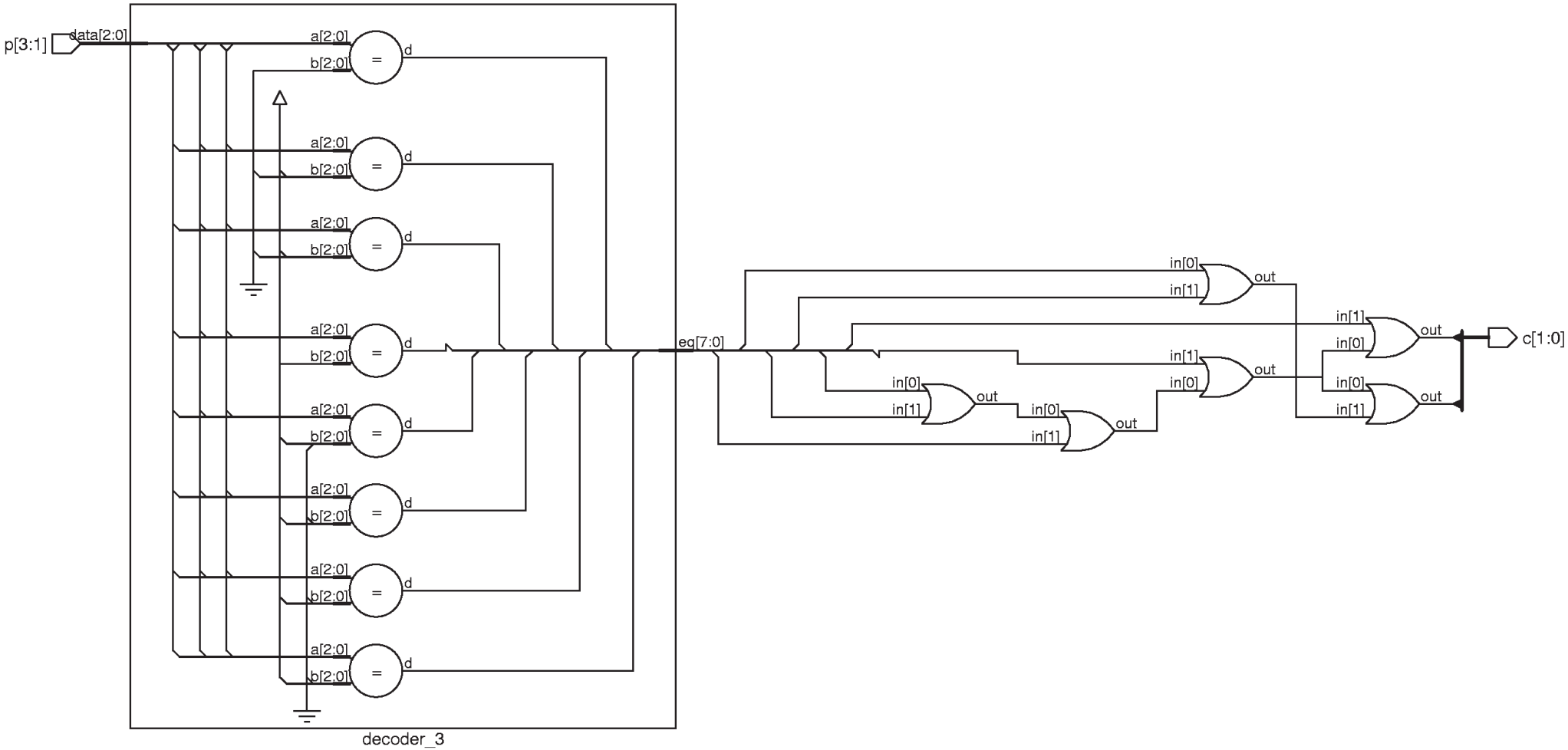
Comparação entre as construções WHEN ELSE e WITH SELECT

- **Circuito gerado com a construção WITH SELECT**



- **Nível RTL**
- **Resultado**: circuito mais complexo

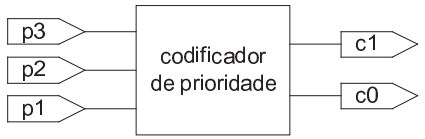
  para cada condição testada é empregado um comparador



Comparação entre as construções WHEN ELSE e WITH SELECT

- **Circuito gerado – nível porta lógica**:

  construções WHEN ELSE e WITH SELECT



- **Ambas as construções com o mesmo resultado**

- **Devido à etapa de minimização da ferramenta de síntese**