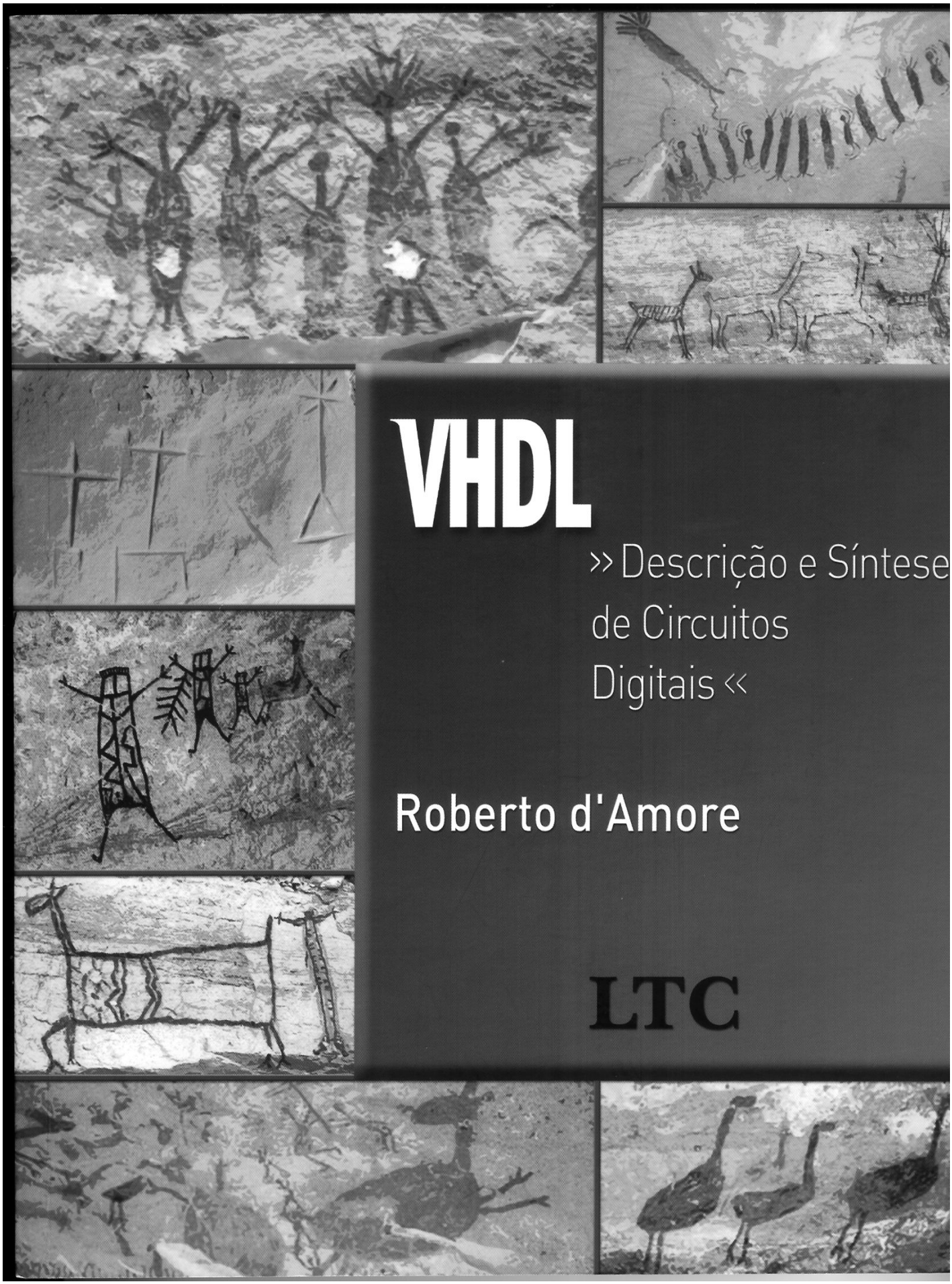
Lista de Exercícios – SEL0632

Capítulo 09



- Constantes deferidas, ou *deferred constants*, têm o seu valor definido no corpo de um pacote. Na declaração do pacote, apenas o tipo e o nome da constante são incluídos.
- Sinais globais podem ser definidos em pacotes como um meio de comunicação entre entidades.

9.7 Exercícios

9.7.1 Desenvolva um pacote contendo subprogramas descrevendo as operações realizadas pelos blocos apresentados na Figura 9.7.1. O bloco " x " contém um somador completo e um circuito de seleção controlado pelo sinal " qi ". Na condição " qi=1 " o resultado da soma é transferido para a saída " ri+1 ", na condição " qi=0 " é transferido o sinal " ri ". O bloco " y " contém apenas o circuito somador. Todos os sinais são do tipo " bit ". O local de armazenamento do pacote compilado deve ser numa nova biblioteca criada especialmente para esse pacote, denominada " dv_celulas ".

Figura 9.7.1 Células empregadas nos Exercícios 9.7.1, 9.7.2 e 9.7.3.

9.7.2 O algoritmo "divisão com restauração", para dois números na base dez, pode ser compreendido pela seqüência descrita na Figura 9.7.2. O processo consiste em subtrair o resto parcial até que o resultado atinja um valor negativo. Quando o valor negativo for encontrado, significa que a operação de subtração foi realizada uma vez além do necessário, e o valor parcial anterior deve ser restaurado.

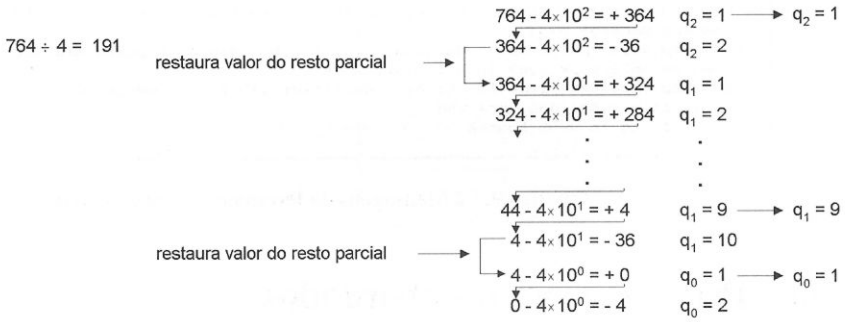


Figura 9.7.2 Ilustração do algoritmo de divisão com restauração — base 10.

Na base dois, o dígito do cociente pode assumir o valor zero ou um. Caso a operação de subtração resulte num valor maior ou igual a zero, o dígito do cociente é igual a " 1 ", e o resto parcial corresponde ao resultado da operação de subtração. Caso a subtração resulte num valor negativo, o dígito do cociente é igual a " 0 ", e o resto parcial anterior é restaurado transformando-se no novo resto parcial (ver exemplo da Figura 9.7.3).

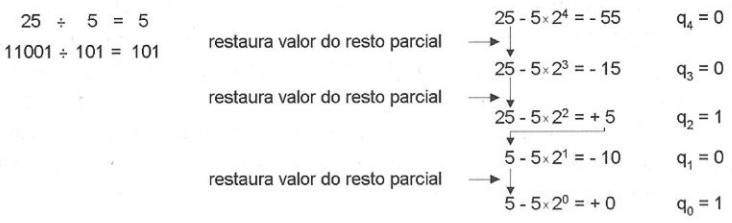


Figura 9.7.3 Algoritmo de divisão com restauração — base 2.

A divisão com restauração pode ser implementada por células que contenham um subtrator completo e um circuito de seleção, denominadas "subtrator controlado". A saída da célula é interligada ao circuito de seleção, o qual transfere o resultado parcial da subtração ou o dígito do minuendo, conforme um sinal de controle externo. Esse controle corresponde ao último bit da cadeia de subtratores, e sinaliza se o resultado da operação de subtração foi negativo ou não.

A Figura 9.7.4 ilustra uma implementação de um divisor por restauração onde a entrada " a " é o dividendo, e a entrada " b " é o divisor, ambas com 4 bits. A operação de subtração na célula do subtrator controlado é implementada por um circuito somador completo, conforme descrito no Exercício 9.7.1. Desse modo, para realização da subtração é necessário complementarmos os bits correspondentes ao divisor e impormos um nível alto na entrada " ci " da primeira célula.

Apresente a descrição de um divisor com " a " contendo 5 bits e " b " contendo 4 bits. As células devem estar armazenadas no pacote desenvolvido no Exercício 9.7.1.

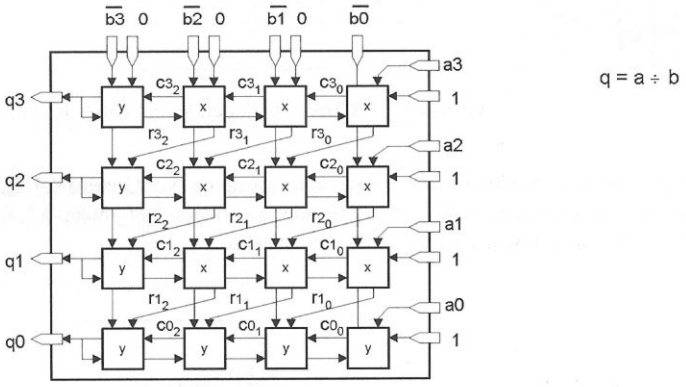


Figura 9.7.4 Divisor do tipo com restauração na forma de uma rede de células.

9.7.3 Considerando o algoritmo de divisão apresentado no Exercício 9.7.2, desenvolva uma descrição para o divisor proposto na Figura 9.7.5. Nessa nova implementação, apenas uma linha de células do tipo subtrator controlado é implementada. A divisão é executada seqüencialmente com o auxílio de dois registradores de deslocamento. A cada ciclo de relógio, um bit do cociente é armazenado, sinal " ds ", e um novo bit do dividendo é inserido na primeira célula da linha, sinal " qp4 ". O número de ciclos de relógio necessário para o processo de divisão é função do número de bits do dividendo, entrada " a ".

Insira na implementação uma unidade de controle para o divisor. Essa unidade, detectando o sinal " ld=1 ", transfere o valor da entrada " a " para o registrador de deslocamento " qp " e inicia o processo de divisão. Uma vez atingido o número de ciclos de relógio para o término da operação, o valor presente no registrador " qs " é transferido para o registrador de saída, mantendo o valor estável. Proponha um novo pacote contendo as operações dos registradores na forma de subprogramas. Tenha cuidado com a descrição dos registradores em subprogramas; lembre-se de que variáveis declaradas em funções ou procedimentos são inicializadas a cada chamada do subprograma.

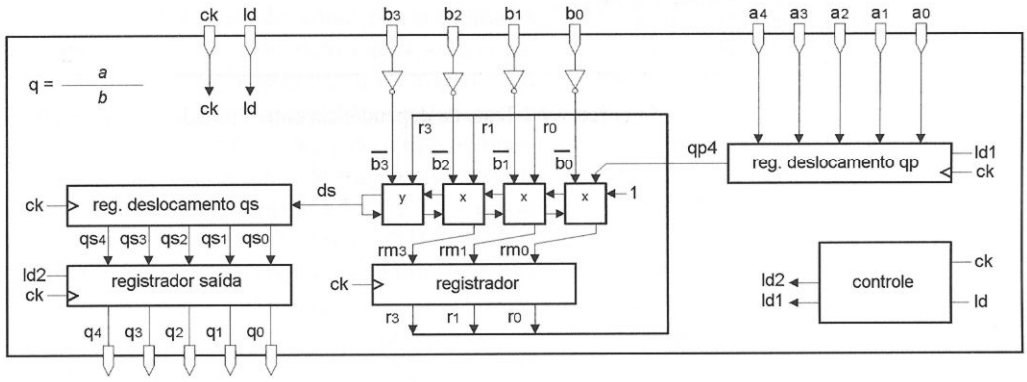


Figura 9.7.5 Divisor do tipo com restauração seqüencial.

9.7.4 O objetivo do exercício é verificar a ordem na análise da compilação e os reflexos que a alteração de uma unidade pode acarretar nas unidades que a referenciam. Considerando o Quadro 9.7.1, altere a declaração da entidade " and2 " mudando, por exemplo, o nome das portas. Recompile somente a declaração e tente simular a entidade " and3 ". Com base na mensagem de erro, proceda com as alterações necessárias para que a simulação possa ser executada com sucesso.

```
1  -- arquivo dep_err0.vhd: duas entidades
2  -- VHDL-1993
3  ENTITY and2 IS
4    PORT (i0, i1: IN BIT; s : OUT BIT);
5  END and2;
6
7  ENTITY and3 IS
8    PORT (i0, i1, i2 : IN BIT; s : OUT BIT);
9  END and3;
10
11 ARCHITECTURE arq_e2 OF and2 IS
12   BEGIN s <= i0 AND i1;
13 END arq_e2;
14
15 ARCHITECTURE arq_e3 OF and3 IS
16   SIGNAL int : BIT;
17 BEGIN
18   x1: ENTITY WORK.and2(arq_e2) PORT MAP(i0,  i1, int);
19   x2: ENTITY WORK.and2(arq_e2) PORT MAP(int, i2, s);
20 END arq_e3;
```

Quadro 9.7.1 Teste da dependência entre unidades (Exercício 9.7.4).

9.7.5 O exercício é semelhante ao proposto anteriormente (ver Quadro 9.7.2). Nesse caso, altere a declaração do pacote. Re-compile somente essa declaração, e tente simular a entidade " or_multiplo ". Com base na mensagem de erro, altere o código para que a simulação possa ser executada com sucesso.

```
1  -- arquivo dep_err3.vhd
2  PACKAGE pk_sup IS    -- declaracao do pacote
3    FUNCTION or2 (SIGNAL i0, i1 : IN  BIT) RETURN BIT;
4  END pk_sup;
5
6  USE WORK.pk_sup.ALL; -- torna todos itens do pacote pk_sup visiveis para pk_inf
7  PACKAGE pk_inf IS    -- declaracao do pacote
8    FUNCTION or4 (SIGNAL i0, i1, i2, i3 : IN  BIT) RETURN BIT;
9  END pk_inf;
10
11 USE WORK.pk_inf.ALL; -- torna itens do pacote pk_inf visiveis para or_multiplo
12 ENTITY or_multiplo IS
13   PORT (a, b, c, d : IN  BIT; s_e : OUT BIT);
14 END or_multiplo;
15
16 ARCHITECTURE teste OF or_multiplo IS
17 BEGIN
18   s_e  <= or4(a, b, c, d);
19 END teste;
20
21 PACKAGE BODY pk_inf IS  -- corpo do pacote pk_inf
22   FUNCTION or4 (SIGNAL i0, i1, i2, i3 : IN  BIT) RETURN BIT IS
23   BEGIN RETURN or2(i0, i1) OR or2(i2, i3);
24   END or4;
25 END pk_inf;
26
27 PACKAGE BODY pk_sup IS  -- corpo do pacote pk_sup
28   FUNCTION or2 (SIGNAL i0, i1 : IN  BIT) RETURN BIT IS
29   BEGIN RETURN (i0 OR i1);
30   END or2;
31 END pk_sup;
```

Quadro 9.7.2 Teste da dependência entre unidades (Exercício 9.7.5).