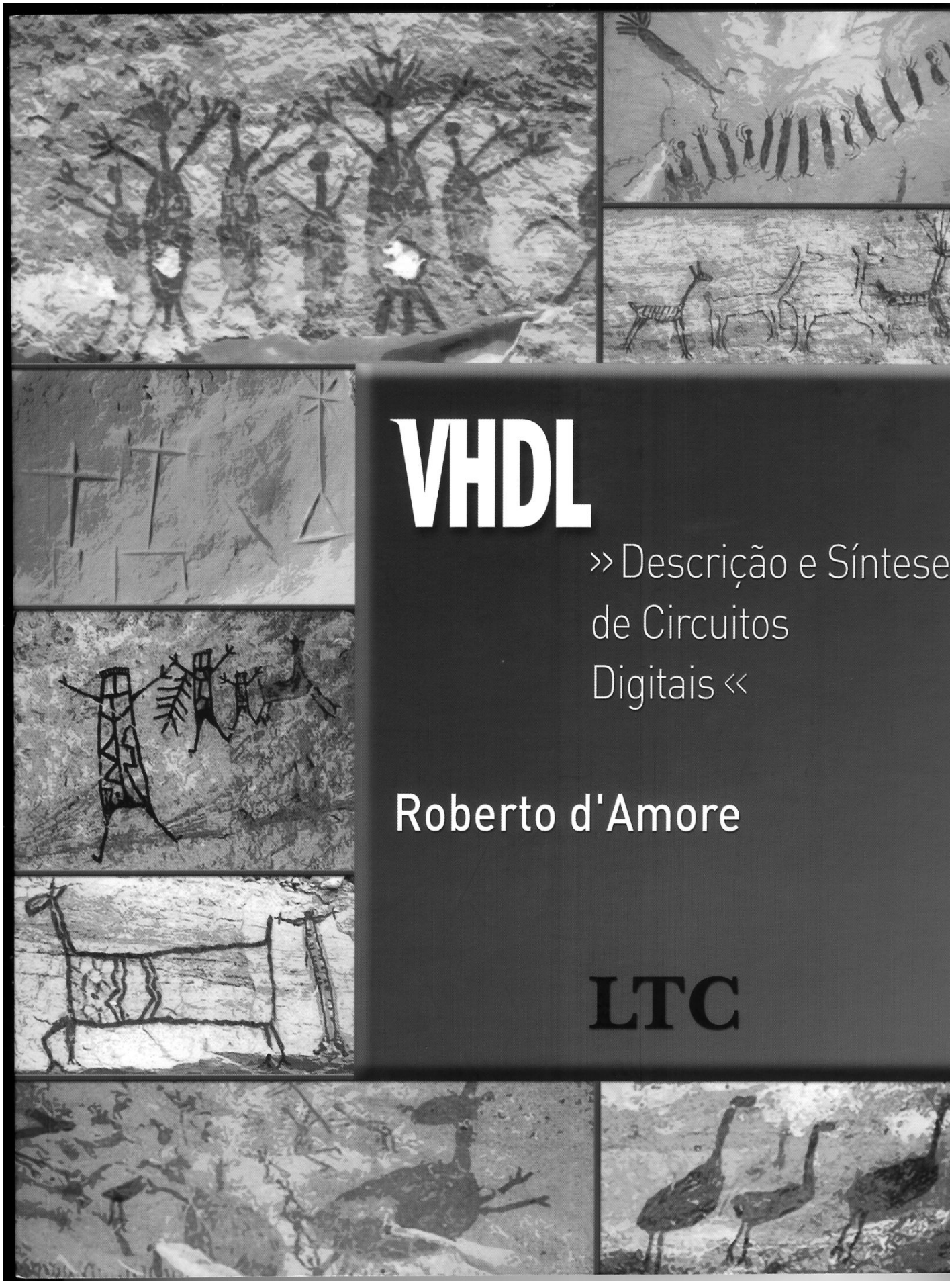
Lista de Exercícios – SEL0632

Capítulo 08



8.8 Exercícios

8.8.1 No Quadro 8.8.1 são apresentadas duas funções que realizam as operações de um somador completo. Apresente a descrição de um somador de 4 bits empregando essas duas funções, sendo a chamada das funções executada de uma região de código concorrente.

8.8.2 Altere o código da solução do Exercício 8.8.1, realizando agora a chamada da função de uma região de código seqüencial.

8.8.3 No Quadro 8.8.2 é apresentada a proposta de uma função para detectar o menor de dois valores. A descrição contém erros. Determine os erros e apresente o código correto.

```
1  FUNCTION soma (a : IN BIT; b : IN BIT; c : IN BIT) RETURN BIT IS
2    BEGIN
3      RETURN a XOR b XOR c;
4  END soma;
5
6  FUNCTION vai_um (a: BIT; b: BIT; c: BIT) RETURN BIT IS
7    BEGIN
8      RETURN (a AND b) OR (a AND c) OR (b AND c);
9  END vai_um;
```

Quadro 8.8.1 Funções para um somador completo.

```
1  ENTITY som_fct7 IS
2    PORT (a_i, b_i    : IN  INTEGER RANGE 0 TO 15;
3          s_o         : OUT INTEGER RANGE 0 TO 15);
4  END som_fct7;
5
6  ARCHITECTURE falha OF som_fct7 IS
7
8  FUNCTION minimo (a : INTEGER;  b : INTEGER) RETURN INTEGER IS
9    SIGNAL min  : INTEGER RANGE 0 TO 15;
10 BEGIN
11   IF (a < b)
12     THEN min <= a;
13     ELSE min <= b;
14   END IF;
15   RETURN min;
16 END minimo;
17
18 BEGIN
19   s_o <= minimo(a_i, b_i);
20 END falha;
```

Quadro 8.8.2 Função com erro.

8.8.4 Apresente o código de um somador de 4 bits empregando unidades de um somador completo de 1 bit. Cada somador completo é implementado por um procedimento contendo três entradas e duas saídas: soma e vai um. A arquitetura deve realizar a chamada dos procedimentos de uma região de código concorrente.

8.8.5 Repita o Exercício 8.8.4 de modo que a chamada do procedimento seja executada de uma região de código seqüencial.

8.8.6 Apresente o código de dois subprogramas que permitam realizar as seguintes operações de conversão: tipo vetor de bits para um tipo inteiro, e tipo inteiro para vetor de bits. Considere que o tamanho do vetor de bits e a faixa de valores para o tipo inteiro são definidos quando o subprograma é invocado.

8.8.7 Apresente a descrição de um subprograma de *flip flop* tipo D, operando conforme a Tabela 8.8.1. Os sinais de " set " e " clr " são ativos no nível baixo. A transferência do dado da entrada " d " para as saídas " q " e " q_b " é feita na borda de subida do sinal de relógio " ck ".

Tabela 8.8.1 Definição do *flip flop* para o Exercício 8.8.7.

operação	entradas				saídas	
	set	clr	ck	d	q	q_b
set - assíncrono	0	1	X	X	1	0
clear - assíncrono	1	0	X	X	0	1
não definido	0	0	X	X	1	1
transfere dado	1	1	0 → 1	0	0	1
	1	1	0 → 1	1	1	0

8.8.8 Empregando o *flip flop* tipo D desenvolvido no Exercício 8.8.7, apresente uma descrição para implementar o circuito da Figura 8.8.1. Cada *flip flop* deve corresponder a uma chamada de um subprograma. O circuito descrito na figura implementa um gerador de padrões pseudo-aleatórios por meio de um registrador de deslocamento com realimentação linear " LFSR ", ou *linear feedback shift register*. Um LFSR consiste em um registrador de deslocamento realimentado com valores gerados por portas do tipo " ou exclusivo ". No exemplo da figura, o LFSR contém quatro estágios, e gera 15 padrões pseudo-aleatórios, excetuando o padrão " 0000 ". Para iniciar a seqüência, deve ser fornecida uma "semente", ou padrão inicial, diferente de " 0000 ". A seqüência de padrões esperada é apresentada na figura.

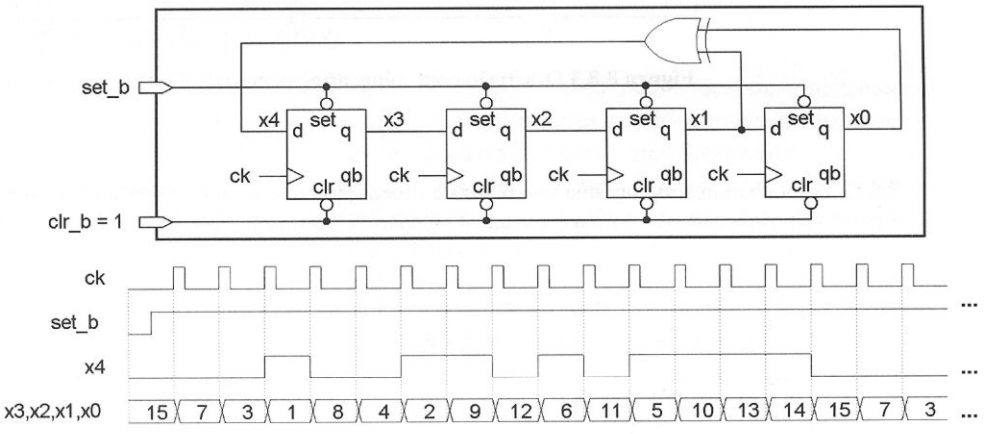
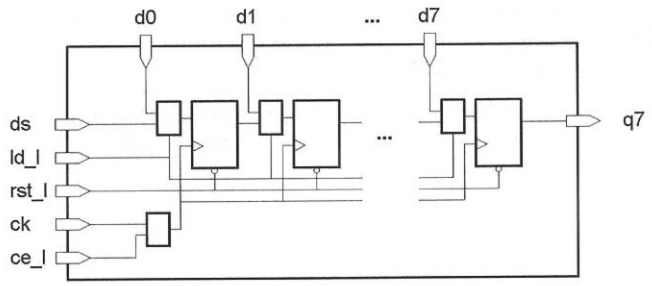


Figura 8.8.1 Registrador de deslocamento com realimentação linear LFSR (Exercício 8.8.8).

8.8.9 Desenvolva o código para um registrador de deslocamento equivalente ao circuito integrado 74166 da família TTL, conforme solicitado no Exercício 6.12.7. Nessa nova proposta, o registrador deve ser descrito na forma de um procedimento. Lembre-se de que uma variável declarada em um subprograma é inicializada a cada chamada do subprograma, e, portanto, não armazena um valor. Para teste do procedimento, insira o subprograma numa região de código concorrente de uma entidade. A Figura 8.8.2 apresenta novamente o diagrama de blocos do registrador.



rst_l	ld_l	ce_l	ck	operação
0	X	X	X	reset assíncrono
1	0	0	↑	carga paralela
1	1	0	↑	desloca
1	X	1	X	desabilitado

Figura 8.8.2 Registrador de deslocamento equivalente ao 74166 (Exercício 8.8.9).

8.8.10 Apresente um código contendo o sobrecarregamento de uma função, conforme ilustrado na Figura 8.8.3. Uma função deve realizar a soma entre dois valores do tipo inteiro, uma outra deve realizar a soma entre três operandos do tipo inteiro, e uma terceira deve realizar a soma entre operandos do tipo " BIT_VECTOR ". Nesta última, o valor da soma é acrescido de um.

8.8.11 Sintetize a descrição proposta para o Exercício 8.8.10, e comente os resultados.

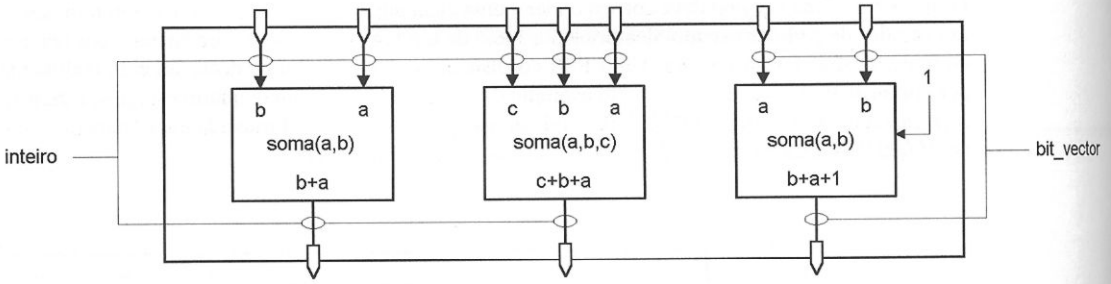


Figura 8.8.3 Descrição com sobrecarregamento da função denominada " soma ".

8.8.12 Proponha um subprograma que realiza a ordenação em um tipo " STRING ". O subprograma deve retornar com os elementos posicionados na forma crescente. O Quadro 8.8.3 ilustra o pseudocódigo de um algoritmo de ordenação simples.

```
-- dado: tipo string com limites de (1 a n)

 De i=2 Ate n Repita
   De j=1 Ate n-1 Repita
     Se dado(j) > dado(j+1)
       troca posicao entre os elementos j e j+1 de dado
     Fim Se
   Fim Repita
 Fim Repita
```

Quadro 8.8.3 Pseudocódigo para ordenação de elementos.

8.8.13 Proponha um subprograma que gera uma forma de onda quadrada com um período de 60 ns. Empregue o modo " INOUT " para o parâmetro formal, e verifique a operação do subprograma em regiões de código concorrente e seqüencial.