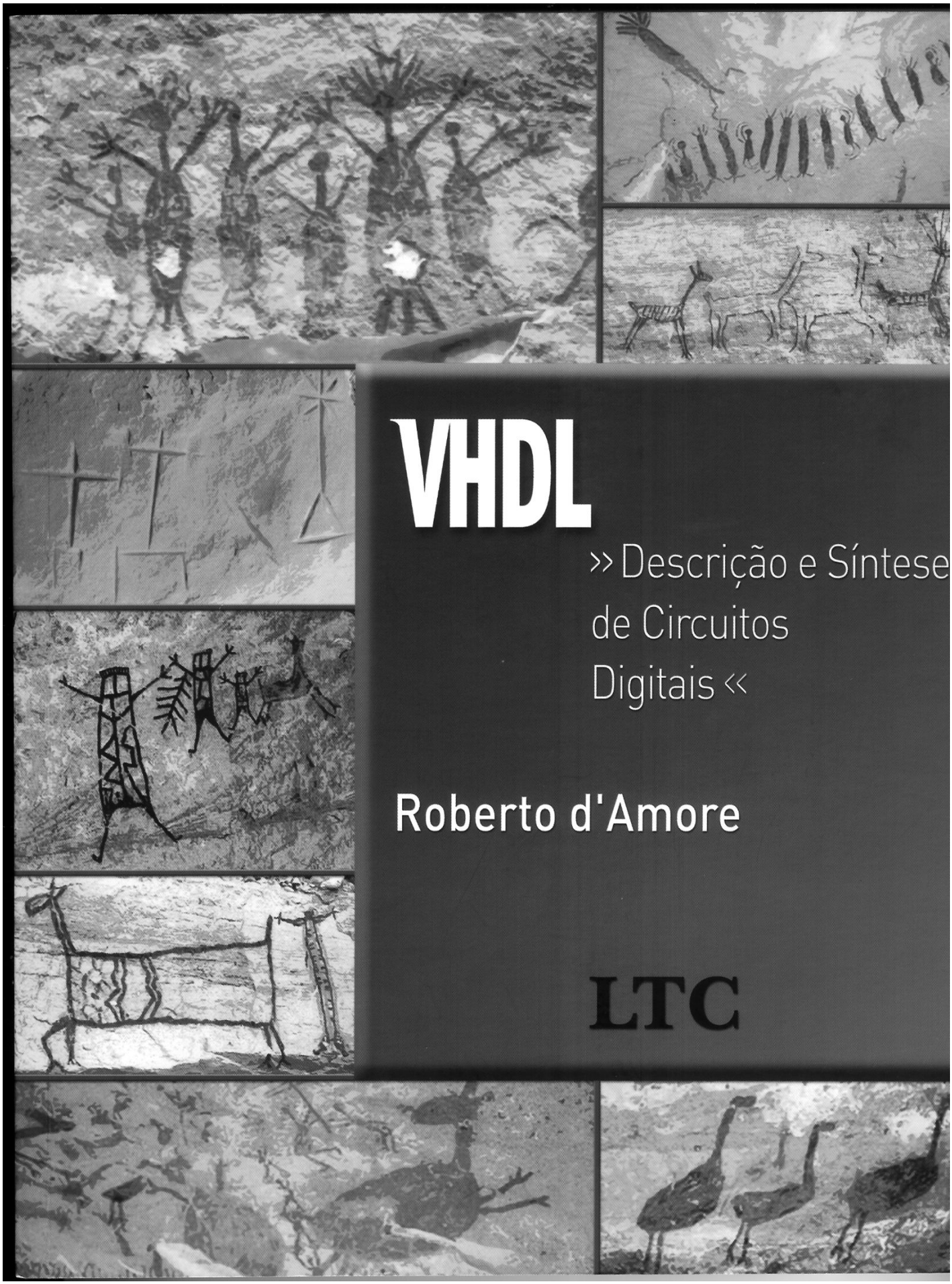
Lista de Exercícios – SEL0632

Capítulo 11



- O tipo " unsigned " representa um número sem sinal, e tipo " signed " representa um número com sinal, formato complemento de dois.
- O padrão define os pacotes denominados " numeric_bit " e " numeric_std ".
- No pacote " numeric_bit " são empregados os tipos " bit " e " bit_vector ", e no pacote " numeric_std " são empregados os tipos " std_logic " e " std_logic_vector ".
- Somente um pacote pode ser empregado por vez, porque ambos denominam os tipos numéricos do mesmo modo: " unsigned " e " signed ".
- A função " std_match " permite comparar valores do tipo " std_logic " considerando o significado desses valores.

11.6 Exercícios

11.6.1 Proponha uma descrição para verificar a operação das funções " shift_right ", " shift_left ", " rotate_right ", " rotate_left ", " sri ", " sli ", " ror " e " rol ".

11.6.2 Crie uma descrição de teste das funções " resize ", e comprove as operações de aumento e redução do tamanho de vetores do tipo " signed " e " unsigned ".

11.6.3 Verifique as operações das funções de conversão para tipos " natural ", " integer ", " unsigned " e " signed ".

11.6.4 Considerando o código do Quadro 11.6.1, comente e compare o resultado que deve ser esperado para os sinais da linha 6 e linha 7.

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3  USE ieee.numeric_std.ALL;
4
5  ENTITY std1076k IS
6    PORT (H_igual_ni,     H_igual_1,     H_igual_H     : OUT BOOLEAN;
7          H_std_match_ni, H_std_match_1, H_std_match_H : OUT BOOLEAN);
8  END std1076k;
9
10 ARCHITECTURE teste OF std1076k IS
11    CONSTANT valor_H     : STD_LOGIC := H;
12 BEGIN
13  H_igual_ni <= TRUE WHEN valor_H = - ELSE FALSE;
14  H_igual_1  <= TRUE WHEN valor_H = 1 ELSE FALSE;
15  H_igual_H  <= TRUE WHEN valor_H = H ELSE FALSE;
16
17  H_std_match_ni <= TRUE WHEN std_match(valor_H, -) ELSE FALSE;
18  H_std_match_1  <= TRUE WHEN std_match(valor_H, 1) ELSE FALSE;
19  H_std_match_H  <= TRUE WHEN std_match(valor_H, H) ELSE FALSE;
20  END teste;
```

Quadro 11.6.1 Exercício 11.6.4.

11.6.5 Apresente um código para um procedimento que implemente um contador com inicialização assíncrona e sensível à borda de subida. Os parâmetros formais do procedimento são os seguintes: " ck " (relógio), " rst " (inicialização), " max_cnt " (constante que define o valor máximo de contagem), e " q_uns " (valor da contagem). O número de bits do contador é determinado pelo tamanho do vetor " q_uns ".

Insira o procedimento dentro de um pacote, e faça o teste a partir de uma outra entidade que realiza uma chamada ao contador (ver Figura 11.6.1). Empregue os seguintes tipos nos parâmetros formais: " ck " e " rst ", tipo " std_logic ", " max_cnt " tipo " natural ", e " q_uns " tipo " unsigned ".

Sintetize o código para verificar se a ferramenta aceita a descrição proposta.

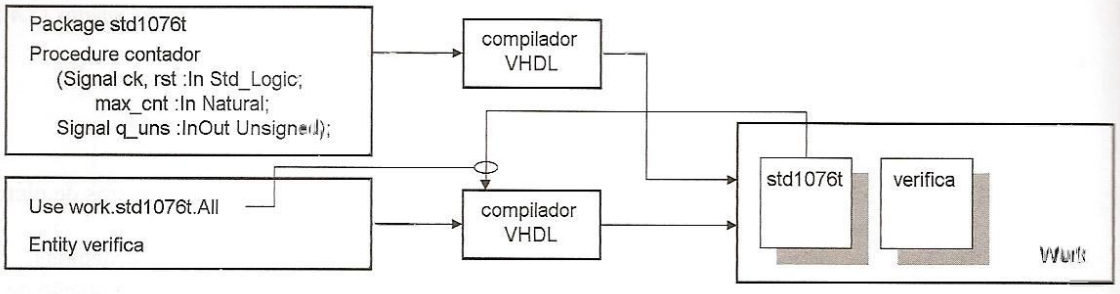


Figura 11.6.1 Exercício 11.6.5.

11.6.6 Considere uma unidade lógica aritmética com oito funções, conforme ilustrado na Figura 11.6.2. A operação executada entre os dados " rs " e " ss " é definida pelo sinal " opr ", a saída " ov " indica um transbordo numa operação de soma ou subtração, a saída " zr " indica que o resultado da operação é nulo, e entrada " ci " corresponde a um dado do tipo "vem um".

Apresente uma descrição para esse circuito, empregando para os sinais " rs " e " ss " tipo " signed " com elementos do tipo " bit ", e para os sinais restantes, tipo " bit ". O número de bits dos sinais " rs ", " ss " e " ts " deve ser definido através de uma cláusula " GENERIC " no início do código.

opr	operação	ov	zr
000	ts ← rs + ss	↑↓	↑↓
001	ts ← rs + ss + ci	↑↓	↑↓
010	ts ← rs - ss	↑↓	↑↓
011	ts ← rs - ss - ci	↑↓	↑↓
100	ts ← rs E ss	?	?
101	ts ← rs OU ss	?	?
110	ts ← rs OU-EXCLUSIVO ss	?	?
111	ts ← NAO rs	?	?

? = indeterminado

↑↓ = afetado pela operação

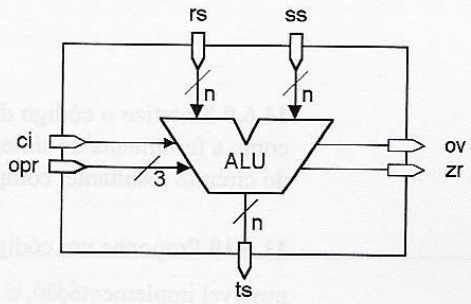


Figura 11.6.2 Exercício 11.6.6.

11.6.7 O código proposto no Quadro 11.6.2 realiza a operação " $a \times 2^n$ " com sinais do tipo inteiro. Altere o código de modo a empregar tipos " unsigned ", observando o número de bits necessário para a gama de valores que cada sinal pode assumir. Como a função " x^y " não é definida nos pacotes do padrão 1076.3, é necessário substituí-la por uma outra operação. Sugestão: Analise como a operação " 2^n " pode ser realizada através de um deslocamento (ver Figura 11.6.2).

Analise, também, o número de bits necessário para realizar as operações.

```
1  ENTITY ex_a1 IS
2    PORT(n : IN  INTEGER RANGE 0 TO 3;
3         a : IN  INTEGER RANGE 0 TO 31;
4         s : OUT INTEGER RANGE 0 TO 256);
5    END ex_a1;
6
7  ARCHITECTURE exemplo OF ex_a1 IS
8  BEGIN
9    s <= a * 2**n;
10 END exemplo;
```

Quadro 11.6.2 Código do Exercício 11.6.7.

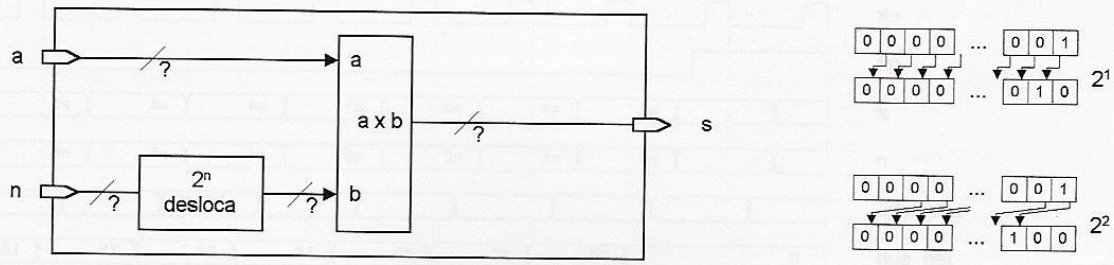


Figura 11.6.3 Sugestão para a alteração do código do Exercício 11.6.7.

11.6.8 Uma análise das operações executadas na descrição do Quadro 11.6.2 permite verificar que é possível substituir as operações executadas em " $a \times 2^n$ " por deslocamentos. Como " n " pode assumir os valores " 0 ", " 1 ", " 2 " e " 3 ", a operação " $\times 2^n$ ", na realidade, se resume em multiplicar " a " por " 1 ", " 2 ", " 4 " e " 8 ", respectivamente (ver Figura 11.6.4). Proponha uma descrição levando em consideração esse fato.

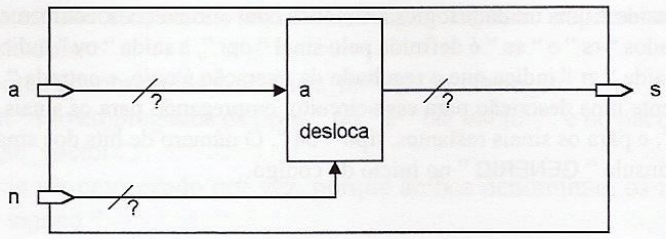


Figura 11.6.4 Nova alteração do código do Exercício 11.6.7.

11.6.9 Sintetize o código do Quadro 11.6.2 e os códigos propostos para a solução dos Exercícios 11.6.7 e 11.6.8. Analise como a ferramenta de síntese tratou a operação " 2^n " e a operação de multiplicação, para os três casos. Verifique o tamanho do circuito resultante, comparando o número de portas ou elementos equivalentes empregados.

11.6.10 Proponha um código para realizar a operação "$\sum_{i=0}^{7} a_i \times 2^{n_i}$". A Figura 11.6.5 contém o diagrama de blocos de uma possível implementação, e a Figura 11.6.6 a respectiva carta de tempos. A cada ciclo de relógio, o valor resultante da operação " $a \times 2^n$ " é acumulado no registrador " reg_sub ". Essa operação é realizada por 8 ciclos. No oitavo ciclo, o resultado da saída do somador é transferido para o registrador " reg_total ", e o conteúdo de " reg_sub " é levado a zero. A entrada " rst " é empregada para inicializar a operação.

Nessa descrição empregue tipos " INTEGER " para facilitar a análise, e determine a gama de valores que cada objeto pode assumir. Isso é necessário para economizar recursos na implementação da descrição.

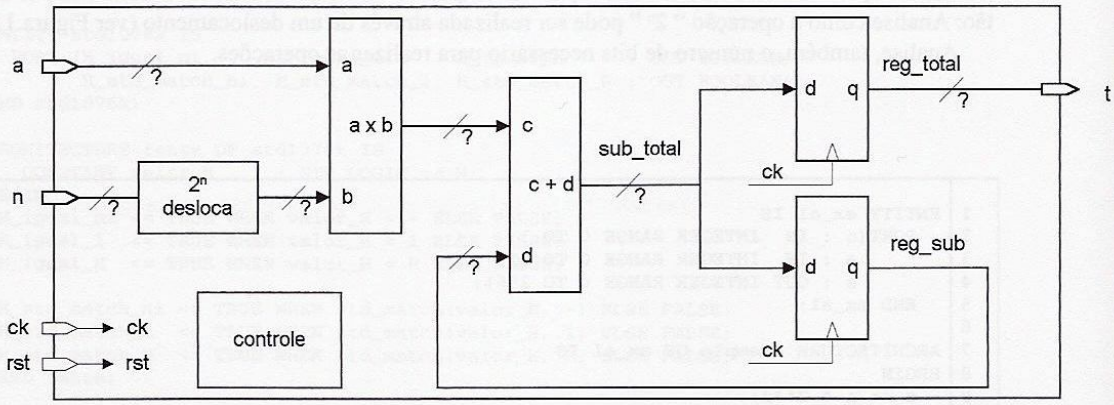


Figura 11.6.5 Diagrama de blocos para solução dos Exercícios 11.6.10 e 11.6.11.

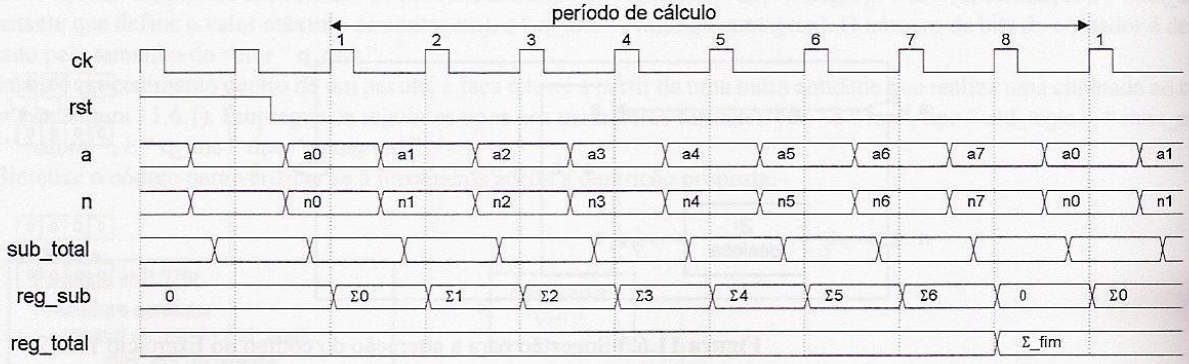


Figura 11.6.6 Carta de tempos para os Exercícios 11.6.10 e 11.6.11.

11.6.11 Altere o código proposto para o Exercício 11.6.10 de modo a empregar tipos " unsigned ". Determine o número de bits necessário para a gama de valores que cada sinal pode assumir, e complete o diagrama de blocos da Figura 11.6.5 com os valores estabelecidos.