Prática Laboratorial – Simulação de Circuitos Combinacionais com Vetores – 02/04/2019 – Roteiro de Atividades

1. Apresentação

O objetivo desta prática é capacitá-lo(a) a simular circuitos combinacionais que empreguem vetores. Você também será incentivado(a) a empregar algumas funções, além daquelas (`to_unsigned` e `std_logic_vector`) que foram apresentadas na teoria.

2. Prática

Realize todos os exercícios com os operadores aritméticos ou de comparação, isto é, não use Karnaugh.

Exercício 1: Desenvolva, sintetize e simule um circuito LT ("menor que") de dois bits (cada operando com dois bits) empregando vetores.

Exercício 2: Desenvolva, sintetize e simule um circuito somador de dois bits empregando vetores. O circuito deverá fornecer como saída um único vetor, que corresponda ao valor total da soma.

Exercício 3: Desenvolva, sintetize e simule um circuito subtrator de três bits empregando vetores. O circuito deverá fornecer como saída um único vetor, que corresponda ao valor da subtração, incluindo resultados positivos e negativos. Lembrete: VHDL opera com a notação "complemento de dois".

Exercício 4: Desenvolva, sintetize e simule um multiplicador de três bits empregando vetores. O circuito deverá fornecer como saída um único vetor, que corresponda ao valor da multiplicação.

4. Funções e operadores úteis em VHDL

Table 3.3 Type conversions between `std_logic_vector` and numeric data types

Data type of `a`	To data type	Conversion function/type casting
`unsigned, signed`	`std_logic_vector`	`std_logic_vector(a)`
`signed, std_logic_vector`	`unsigned`	`unsigned(a)`
`unsigned, std_logic_vector`	`signed`	`signed(a)`
`unsigned, signed`	`integer`	`to_integer(a)`
`natural`	`unsigned`	`to_unsigned(a, size)`
`integer`	`signed`	`to_signed(a, size)`

Table 3.1 Operators and data types of VHDL-93 and IEEE `std_logic_1164` package

Operator	Description	Data type of operands	Data type of result
`a ** b`	exponentiation	`integer`	`integer`
`a * b`	multiplication		
`a / b`	division	*integer type for constants and*	
`a + b`	addition	*array boundaries, not synthesis*	
`a - b`	subtraction		
`a & b`	concatenation	1-D array, element	1-D array
`a = b`	equal to	any	`boolean`
`a /= b`	not equal to		
`a < b`	less than	scalar or 1-D array	`boolean`
`a <= b`	less than or equal to		
`a > b`	greater than		
`a >= b`	greater than or equal to		
`not` `a`	negation	`boolean, std_logic,`	same as operand
`a` **`and`** `b`	and	`std_logic_vector`	
`a` **`or`** `b`	or		
`a` **`xor`** `b`	xor		

Table 3.2 Overloaded operators and data types in the IEEE `numeric_std` package

Overloaded operator	Description	Data type of operands	Data type of result

`a * b`
`a + b`
`a - b`
	arithmetic operation	`unsigned, natural`
		`signed, integer`
			`unsigned`
			`signed`

`a = b`
`a /= b`
`a < b`
`a <= b`
`a > b`
`a >= b`
	relational operation	`unsigned, natural`
		`signed, integer`
			`boolean`
			`boolean`