Sistemas Reconfiguráveis – Eng. de Computação
Especificações para o primeiro projeto

2º semestre de 2022

1. Objetivo

Descrever em linguagem VHDL, comentar, simular o funcionamento e comentar os resultados da simulação de uma unidade lógica e aritmética (ALU na sigla em inglês), conforme especificado a seguir.

O trabalho deverá ser entregue em um documento padrão ABNT para trabalhos acadêmicos (Capa, folha de rosto, índice de figuras, etc, etc), em um arquivo no formato pdf, via Canvas.

2. ALU

Faz operações lógicas e aritméticas em palavras de 8 bits. Realiza 16 funções diferentes, entre operações lógicas, aritméticas, de rotação e de deslocamento. O circuito deverá ser totalmente combinacional e deverá ser descrito usando exclusivamente **código concorrente**. Todas as entradas e saídas deverão usar o tipo STD_LOGIC ou STD_LOGIC_VECTOR.

2.1. Entradas

a_in[7..0]	Entrada "a" de dados.
b_in[7..0]	Entrada "b" de dados. Usada nas operações que envolvem dois operandos.
c_in	Entrada de *carry* (usada em algumas operações aritméticas e de rotação)
op_sel[3..0]	Entrada de seleção da operação a ser realizada.

2.2. Saídas

r_out[7..0]	Saída do resultado.
c_out	Saída de *carry/borrow*. Nas operações aritméticas de soma, este sinal é o *carry out* (vai um) no bit mais significativo. Nas operações de subtração, este sinal é o *borrow out* (empréstimo). Este sinal também é usado nas operações de rotação.
z_out	Saída de zero. Sinaliza quando o resultado da operação é zero.

2.3. Operações

op_sel[3..0]	Mnemônico	Operação
0000	AND	AND lógico bit a bit: r_out = a_in AND b_in c_out = '0' z_out = '1' se o resultado for igual a zero
0001	OR	OR lógico bit a bit: r_out = a_in OR b_in c_out = '0' z_out = '1' se o resultado for igual a zero
0010	XOR	XOR lógico bit a bit: r_out = a_in XOR b_in c_out = '0' z_out = '1' se o resultado for igual a zero
0011	COM	Complemento (inverte todos os bits) r_out = NOT a_in c_in = '0' z_in = '1' se o resultado for igual a zero

0100	ADD	Soma sem *carry in*: r_out = a_in + b_in c_out = '1' se houver *carry* no bit mais significativo z_out = '1' se o resultado for igual a zero
0101	ADDC	Soma com *carry in*: r_out = a_in + b_in + c_in c_out = '1' se houver *carry* no bit mais significativo z_out = '1' se o resultado for igual a zero
0110	SUB	Subtração sem *carry in*: r_out = a_in - b_in c_out = '1' se houver *borrow* no bit mais significativo z_out = '1' se o resultado for igual a zero
0111	SUBC	Subtração com *carry in*: r_out = a_in - b_in c_out = '1' se houver *borrow* no bit mais significativo z_out = '1' se o resultado for igual a zero
1000	RL	Rotação para esquerda: r_out = a_in[6..0], a_in[7] c_out = a_in[7] z_out = '1' se o resultado for igual a zero
1001	RR	Rotação para direita: r_out = a_in[0], a_in[7..1] c_out = a_in[0] z_out = '1' se o resultado for igual a zero
1010	RLC	Rotação para esquerda através do *carry*: r = a[6..0], c_in c_out = a_in[7] z_out = '1' se o resultado for igual a zero
1011	RRC	Rotação para direita através do *carry*: r_out = c_in, a_in[7..1] c_out = a_in[0] z_out = '1' se o resultado for igual a zero
1100	SLL	Deslocamento lógico para esquerda: r_out = a_in[6..0], '0' c_out = a_in[7] z_out = '1' se o resultado for igual a zero
1101	SRL	Deslocamento lógico para direita: r_out = '0', a_in[7..1] c_out = a_in[0] z_out = '1' se o resultado for igual a zero
1110	SRA	Deslocamento aritmético para direita: r_out = a_in[7], a_in[7..1], c_out = a_in[0] z_out = '1' se o resultado for igual a zero
1111	PASS_B	*By-pass B* r_out = b_in c_out = '0' z_out = '1' se o resultado for igual a zero