Laboratório 5

Instruções:

Antes de iniciar o laboratório faça o download do arquivo 'lab05_material_v2018.1.zip' no moodle, ele contém todos as descrições de *entity* necessárias para implementar os circuitos. Cada interface deve ser respeitada e isso será avaliado.

Os arquivos devem ser enviados conforme a orientação de cada questão, nesse laboratório nenhuma entrega deve ser comprimida em qualquer formato, como *zip, tar, tar.gz*.

Deixe os arquivos que serão gravados na placa para demonstração em projetos separados e já pré-compilados, de forma que não seja necessária a recompilação do projeto no momento da demonstração. Assim, basta abrir o projeto correspondente e programar a placa, economizando o tempo da aula dedicado às avaliações.

1. Rotação binária é uma operação que consiste em, dado uma palavra de *n bits*, fazer um reposicionamento dos *bits* na palavra de maneira circular, mantendo a ordem. Isto é, quando o reposicionamento é diferente de um múltiplo de *n*, o *bit* mais significativo da palavra rotacionada é aquele de posição anterior, no vetor original, ao *bit* menos significativo. Ex.: $x_3\ x_2\ x_1\ x_0$, rotacionado em 2 é $x_1\ x_0\ x_3\ x_2$. Uma forma de implementar essa operação em *hardware* é através do circuito chamado *barrel shifter*. Dados uma palavra binária $W = w_3\ w_2\ w_1\ w_0$, um valor de rotacionamento $S = s_1\ s_0$, a saída $Y = y_3\ y_2\ y_1\ y_0$ será conforme a tabela verdade mostrada abaixo:

s_1	s_0	y_3	y_2	y_1	y_0
0	0	w_3	w_2	w_1	w_0
0	1	w_0	w_3	w_2	w_1
1	0	w_1	w_0	w_3	w_2
1	1	w_2	w_1	w_0	w_3

a) Implemente em VHDL um *barrel shifter* de 4 bits a partir da tabela verdade acima. Simule seu funcionamento. **Entregar o arquivo <barrel_shifter.vhd> e um screenshot dos resultados da simulação em <barrel_shifter.png>.**

b) **Execute** sua implementação na DE1-SoC, utilizando o mapeamento, sendo switches as entradas e os leds as saídas do circuito, conforme especificado na tabela abaixo. **Não entregue nenhum arquivo, ele será avaliado na demonstração.**

Porta FPGA	Porta Entidade VHDL
SW(3 downto 0)	$w_3\ w_2\ w_1\ w_0$
SW(5 downto 4)	$s_1\ s_0$
LEDR(3 downto 0)	$y_3\ y_2\ y_1\ y_0$

2. O somador *carry look-ahead* (CLA) é uma alternativa ao somador *ripple-carry*, explorado no laboratório anterior. **As análises do caminho crítico devem ser entregue em um arquivo <timings.txt>, incluindo o tempo de propagação para cada teste.**

a) Implemente em VHDL um somador *carry look-ahead* (CLA) de 4 *bits.* Verifique sua implementação simulando**.** Compare, usando o analisador de tempo, o caminho crítico e seu tempo dessa implementação com a do laboratório anterior (somador *ripple-carry* de 4 *bits*). Qual tem o menor tempo? **Entregar o arquivo <cla_4bits.vhd> e um screenshot dos resultados da simulação em <cla_4bits.png>.**

b) **Execute** sua implementação na DE1-SoC, utilize o mapeamento, sendo os switches as entradas e os leds as saídas do circuito, conforme especificado na tabela abaixo. **Não entregue nenhum arquivo, ele será avaliado na demonstração.**

Porta FPGA	Porta Entidade VHDL
SW(3 downto 0)	$x_3\, x_2\, x_1\, x_0$
SW(7 downto 4)	$y_3\, y_2\, y_1\, y_0$
SW(8)	cin
LEDR(3 downto 0)	$sum_3\, sum_2\, sum_1\, sum_0$
LEDR(4)	cout

c) Implemente em VHDL um somador de 8 bits com CLA parcial, utilizando dois somadores CLA de 4 *bits*, interconectados em cascata (carry-out do CLA menos significativo conectado ao carry-in do CLA mais significativo). Compare, usando o analisador de tempo, o caminho crítico e seu tempo dessa implementação com a do laboratório anterior (somador *ripple-carry* de 8 *bits*). Qual tem o menor tempo? **Entregar o arquivo <cla_8bits_partial.vhd> e um screenshot dos resultados da simulação em <cla_8bits_partial.png>.**

d) Implemente em VHDL um somador *carry look-ahead* (CLA puro) 8 *bits*, sem nenhuma ligação em cascata ou do tipo ripple carry. Para isto, você terá que estender as equações de CLA de 4 bits (dadas em aula) para 8 bits. Compare, usando o analisador de tempo, o caminho crítico e seu tempo dessa implementação com as implementações anteriores de 8 bits (*ripple carry*, CLA parcial). Qual tem o menor tempo? **Entregar o arquivo <cla_8bits.vhd> e um screenshot dos resultados da simulação em <cla_8bits.png>.**