Laboratório 05

Instruções:

- Quando for demonstrar seu trabalho, tome nota do número da placa utilizada. O número da placa será utilizado para atribuir a nota ao grupo.
- A última página deste documento contém um checklist com todos os arquivos que fazem parte da entrega.
- Os nomes dos arquivos devem ser seguidos, e isso faz parte da avaliação.
- A entrega deverá estar em único arquivo .ZIP, com o nome **T_Lab05_RA.zip**, **T** é a turma, e **RA** é o RA do componente do grupo que fará a entrega. Por exemplo, B_Lab05_123456.zip é a entrega do grupo do aluno com o RA 123456, na turma B.
- Não divida ou agrupe em pastas os arquivos dentro do .ZIP.
- A entrega deve ser feita pelo Google Forms (https://forms.gle/pUcMCjhBm7faUMWWA). Você deve estar autenticado com uma conta do Google - pode ser uma conta pessoal ou da DAC.
- Apenas um integrante do grupo precisa fazer a entrega.
- Preste especial atenção aos nomes das entidades e sinais (entradas e saídas) descritos nos laboratórios. Isso também faz parte da avaliação.
- Se mais do que um arquivo for recebido para a mesma entrega, o último recebido será considerado. Utilize o mesmo RA do aluno entregando.
- Faça o download do arquivo **lab05_material_v2023.1.zip**. Esse arquivo já contém as descrições de *entity* necessárias para implementar os circuitos. Utilize elas, e não as altere.

Parte I - Barrel Shifter

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

I.1. Implemente em VHDL um *barrel shifter* de 4 bits a partir da tabela verdade acima. Simule seu funcionamento.

ENTREGA: Arquivo **barrel_shifter.vhd** e um screenshot dos resultados da simulação em **barrel_shifter.png**.

I.2. 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.

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)	$y_3\,y_2\,y_1\,y_0$

Parte II - Somador CLA

O somador *carry look-ahead* (CLA) é uma alternativa ao somador *ripple-carry*, explorado no laboratório anterior.

II.1. Implemente em VHDL um somador *carry look-ahead* (CLA) de 4 *bits.* Verifique sua implementação simulando**.**

ENTREGA: Arquivo **cla_4bits.vhd** e um screenshot dos resultados da simulação em **cla_4bits.png**.

II.2. 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.

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

II.3. Implemente em VHDL um somador *carry look-ahead* (CLA puro) 8 *bits*. Para isto, você terá que estender as equações de CLA de 4 bits (dadas em aula) para 8 bits.

ENTREGA: Arquivo **cla_8bits.vhd** e um screenshot dos resultados da simulação em **cla_8bits.png**.

- ENTREGA -

Entregue um único arquivo comprimido em formato **ZIP** de nome **T_Lab05_RA.zip**, onde **RA** é o RA do aluno entregando e **T** é a turma, contendo:

- Arquivos **barrel_shifter.vhd** e **barrel_shifter.png** do item **I.1**.
- Arquivos **cla_4bits.vhd** e **cla_4bits.png** do item **II.1**.
- Arquivos **cla_8bits.vhd** e **cla_8bits.png** do item **II.3**.