Laboratório 4

Instruções:

Antes de iniciar o laboratório faça o download do arquivo 'lab04_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. Um somador binário de *n* bits é um sistema combinacional que tem duas entradas (*x* e *y*) de *n* bits que representam os operandos da operação de adição e uma saída de *n* bits que representa o resultado. Sinais adicionais de entrada e saída, chamados de *carry-in* (cin) e *carry-out* (cout) são usados para facilitar a implementação de somadores maiores.

a) Projete na entidade *full_adder* <full_adder.vhd>, com VHDL estrutural, um somador completo de um bit (com os *carries*), utilizando somente portas lógicas. Elabore uma simulação para testar o funcionamento do seu circuito. **Entregar o arquivo <full_adder.vhd> e um screenshot dos resultados da simulação em <full_adder.png>.**

b) Usando o circuito do item *a* implemente um somador *ripple-carry adder* de N bits (entidade *ripple_carry* <ripple_carry.vhd>). O circuito deve ter uma saída adicional para indicar overflow. Não utilize pacotes (packages). **Entregar o arquivo <ripple_carry.vhd> e um screenshot de resultados da simulação em <ripple_carry.png>** (O screenshot pode mostrar apenas um trecho da simulação)**.**

c) Adicione o arquivo <DE1_SoC_design_constraints.sdc> (da pasta de Material Complementar do curso) no seu projeto do item (b). Compile o projeto para N=4. Siga os passos para fazer a análise do tempo de resposta do circuito:
   1. Inicie a ferramenta *TimeQuest Timing Analyzer* no menu *Tools*;
   2. Do menu *Netlist*, selecione *Create Timing Netlist…*; aceite o valor padrão de cada campo clicando em *OK*.
   3. Novamente no menu *Netlist*, selecione *Update Timing Netlist*;
   4. No menu *Reports*, selecione *Custom Reports* e *Report Path…*;
   5. Em *Targets*, clique no botão ao lado do campo *From*;
   6. No menu drop-down *Collection*, selecione *get_ports* e clique em *List*;
   7. Selecione da lista da esquerda todas as portas de entrada do circuito e os transfira para a lista da direita; clique *OK*;
   8. Repita os passos 6 e 7 para o campo *To*, agora selecionando as portas de saída do circuito;
   9. Confirme clicando em *Report Path*.

   A ferramenta exibirá múltiplas informações sobre a propagação do sinal pelo circuito. Tome um tempo para tentar identificar componentes do circuito nos relatórios. Anote o tempo de propagação total (*Delay*). Repita o procedimento para N = 8, 16, 32 e 64. O que acontece com o tempo? Qual a explicação? **Resuma em poucas palavras sua análise em um arquivo <timings.txt>, incluindo o tempo de propagação para cada teste.**

d) Instancie o componente *ripple-carry adder* para N=4 na entidade *ripple_carry_board* <ripple_carry_board.vhd> e faça as seguintes ligações de entrada e saída: SW(7..4) para entrada do operando x e SW(3..0) para entrada do operando y; HEX4 mostra o

operando x em hexadecimal, HEX2 o operando y, e HEX0 mostra o resultado; LEDR(0) acende se houver *overflow*. **Não entregue o arquivo ripple_carry_board.vhd, ele será avaliado na demonstração.**

Dicas: Utilize seu circuito *bin2hex* (Laboratório 02) para controlar os visores de sete segmentos. Lembre-se que a representação em hexadecimal não mostrará o sinal dos números.

2. Uma unidade aritmética lógica (*arithmetic – logic* unit) ALU é um módulo capaz de realizar um conjunto de funções aritméticas e lógicas. Para isto, uma ALU tem vetores de entrada/saída de dados, bem como entradas e saídas de controle. A Figura 1 mostra uma possível especificação de uma ALU de 4 bits.

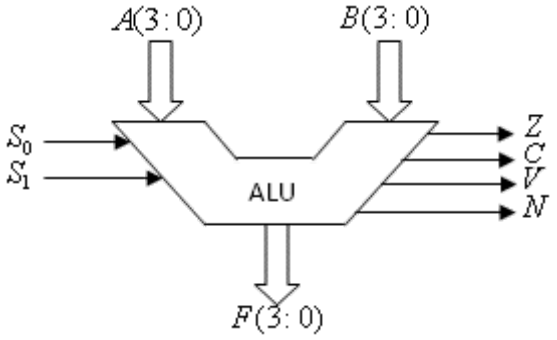

Figura 1: ALU de 4 bits

A ALU da Figura 1 tem entradas (A e B) de 4 bits, e s_0 e s_1 são sinais de controle para selecionar a operação (ver tabela 1 a seguir). A saída F de 4 bits contém o resultado da operação. Z é 1 se o resultado da operação for zero, e 0 caso contrário.

Os sinais C, V e N são 0 para operações lógicas e tem o seguinte significado para operações aritméticas. C é 1 se houver um *carry* em operações de soma, V é 1 se houver *overflow* e N é igual a 1 se o resultado for negativo. C, V e N são 0 nos restante dos casos.

S_0	S_1	Operação (F)
0	0	A+B
1	0	A-B
0	1	AND
1	1	OR

Tabela 1: Operações ALU

a) Modifique a implementação de uma das suas entidades de controle de visores de 7 segmentos (bin2dec ou bin2hex, do Laboratório 02) para tratar entradas de 4 bits em complemento de 2. Utilize o arquivo <two_comp_to_7seg.vhd>, em que a entrada bin(3..0) é o número a ser representado, a saída segs(6..0) são os segmentos de *g* a *a* (segs(0)=a, segs(6)=g) do visor e a saída neg é 1 quando o número for negativo e 0 em caso contrário. **Entregar o arquivo <two_comp_to_7seg.vhd>.**

b) Usando o somador *ripple-carry adder* da questão 1, implemente uma ALU de quatro bits (entidade alu <alu.vhd>) com as operações descritas pela Tabela 1. A ALU deve implementar as operações de soma e subtração utilizando o mesmo *ripple-carry adder*. Não utilize pacotes (packages). **Entregar o arquivo <alu.vhd> e um screenshot de resultados de simulação em <alu.png>. Inclua na sua simulação alguns casos extremos (por exemplo, overflow).**
Dica: utilize a entrada de carry-in do somador para conversão para complemento de 2.

c) Instancie o circuito da ALU para gravação na placa utilizando a entidade *alu_board* (arquivo <alu_board.vhd>). Utilize as entradas SW(9..8) como s_0 e s_1, SW(7..4) como operador A e SW(3..0) como operador B. Represente o operador A em HEX4, B em HEX2 e a saída F em HEX0, com o segmento g do visor logo à esquerda (HEX5, HEX3 e HEX1, respectivamente) representando o sinal negativo, quando necessário. Para operações lógicas mostre os valores em hexadecimal (utilizando seu circuito bin2hex do Laboratório 02), e para operações aritméticas mostre os valores em decimal com sinal. Utilize os leds LEDR(3..0) para representar Z,C,V,N, respectivamente. **Não entregue o arquivo alu_board.vhd, ele será avaliado na demonstração.**