UNIVERSIDADE FEDERAL DE MINAS GERAIS

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Organização de Computadores II
Trabalho Prático 2 – Integração Banco de Registradores e ALU

Artur Henrique Marzano Gonzaga
Caio Felipe Zanatelli
Matheus Henrique de Souza
Tiago de Rezende Alves

Professor: Omar Paranaiba Vilela Neto
Monitor: Laysson Oliveira Luz

Belo Horizonte
15 de outubro de 2017

Sumário

1 Introdução

Este trabalho tem como objetivo principal a criação de um sistema de *harware* composto por um banco de registradores e uma unidade lógico-aritmética (ALU), capaz de realizar um conjunto bem definido de instruções. Para tanto, foi utilizada a linguagem de descrição de *harware* Verilog e o pacote de desenvolvimento DE2-115, da Altera. Por fim, as especificações de cada unidade funcional, bem como as decisões de projeto relativas a cada uma delas, serão abordadas nas seções seguintes.

2 Unidades Funcionais

Esta seção tem como objetivo apresentar cada unidade funcional desenvolvida, abordando suas especificações e decisões tomadas na fase de projeto. A Figura 1 ilustra o circuito resultante obtido.

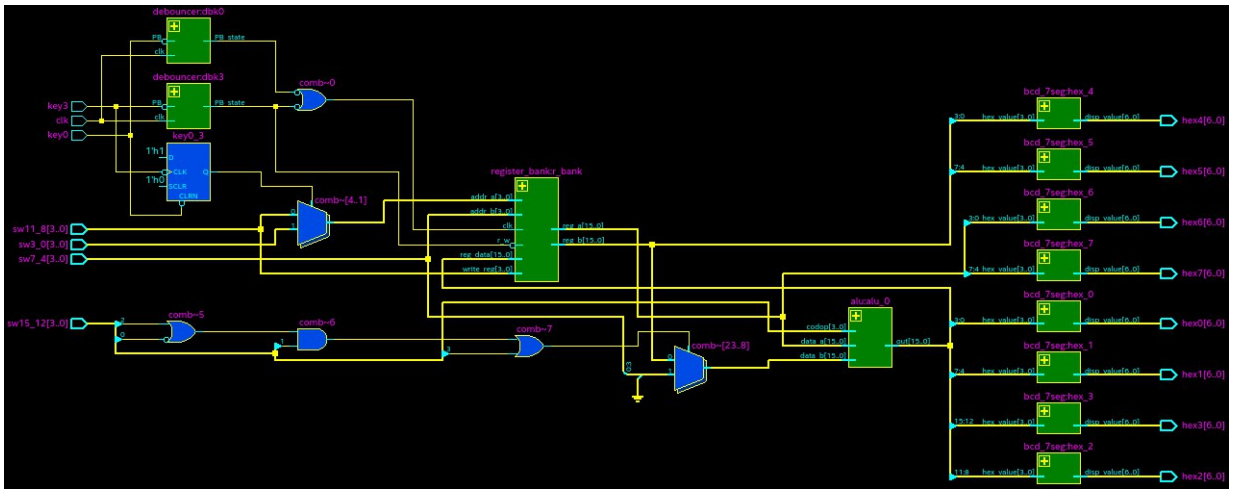

Figura 1 – Diagrama esquemático do Circuito.

2.1 Banco de Registradores

O banco de registradores é a primeira unidade funcional que compõe este projeto. Sua finalidade é proporcionar o acesso rápido, por parte do processador, a valores endereçados pelo banco, uma vez que o acesso à memória é bem mais lento. Vale ressaltar, neste ponto, que, no contexto deste trabalho, o banco de registradores é composto por 16 registradores, com cada um contendo 16 *bits*. Com isso, tem-se que o endereçamento é realizado através de 4 *bits*, como abordado na sequência.

2.1.1 Sinais de Entrada e Saída

De modo a garantir o funcionamento correto do módulo, é necessária a utilização de alguns sinais de entrada e saída. Os sinais em questão são ilustrados pela Tabela 1 e pela Tabela 2, respectivamente.

Tabela 1 – Sinais de entrada utilizados pela Banco de Registradores

Sinal	Tamanho (bits)	Tipo	Descrição
clk	1	wire	Clock da unidade.
addr_a	4	wire	Endereço do registrador a ser lido.
addr_b	4	wire	Endereço do registrador a ser lido.
reg_data	16	wire	Dado a ser escrito em um registrador.
write_reg	4	wire	Endereço do registrador que será escrito.
r_w	1	wire	Flag de leitura e escrita. 0 indica leitura, 1 escrita.

Tabela 2 – Sinais de saída utilizados pela Banco de Registradores

Sinal	Tamanho (bits)	Tipo	Descrição
reg_a	16	reg	Resultado da operação.
reg_b	16	wire	Flag indicativa de resultados negativos.

2.2 Unidade Lógico Aritmética (ALU)

A segunda unidade funcional que compõe este projeto é a unidade lógico-aritmética (ALU), a qual opera sobre palavras de 16 *bits*. Assim, de forma a proporcionar uma melhor compreensão acerca do funcionamento deste módulo, a Figura 2 ilustra o formato definido para as instruções e, em seguida, a Tabela 3 apresenta o conjunto de instruções suportadas.

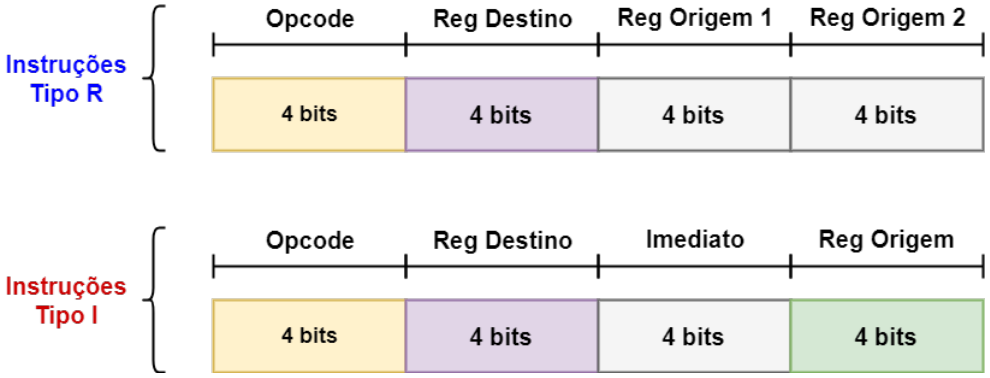

Figura 2 – Formato das instruções.

Tabela 3 – Conjunto de instruções suportadas pela ALU

Op Code	**Instrução**	**Mnemônico**	**Operação**
0	Add	Add $s4, $s3, $s2	$s4 = $s3 + $s2
1	Sub	Sub $s4, $s3, $s2	$s4 = $s3 – $s2
2	Slti	Slti $s4, Imm, $s2	*if* ($s2 > Imm) ? $s4 = 1 : $s4 = 0
3	And	And $s4, $s3, $s2	$s4 = $s3 $s2
4	Or	Or $s4, $s3, $s2	$s4 = $s3 | $s2
5	Xor	Xor $s4, $s3, $s2	$s4 = $s3 ^ $s2
6	Andi	Andi $s4, Imm, $s2	$s4 = $s2 & Imm
7	Ori	Ori $s4, Imm, $s2	$s4 = $s2 | Imm
8	Xori	Xori $s4, Imm, $s2	$s4 = $s2 ^ Imm
9	Addi	Addi $s4, Imm, $s2	$s4 = $s2 + Imm
10	Subi	Subi $s4, Imm, $s2	$s4 = $s2 – Imm

2.2.1 Sinais de Entrada e Saída

Para que as operações definidas pela Tabela 3 possam ser realizadas, faz-se necessária a utilização de alguns sinais de entrada e saída pelo módulo. Esses sinais são apresentados pela Tabela 4 e pela Tabela 5, respectivamente.

Tabela 4 – Sinais de entrada utilizados pela ALU

Sinal	**Tamanho (bits)**	**Tipo**	**Descrição**
clk	1	wire	clock da unidade.
codop	4	wire	Código de operação.
data_a	16	wire	Operando 1.
data_b	16	wire	Operando 2.

Tabela 5 – Sinais de saída utilizados pela ALU

Sinal	**Tamanho (bits)**	**Tipo**	**Descrição**
out	16	reg	Resultado da operação.
neg	1	wire	Flag indicativa de resultados negativos.
zero	1	wire	Flag indicativa de resultados nulos.
overflow	1	wire	Flag indicativa de overflow.

2.3 Debouncer

Um dos problemas encontrados na realização deste trabalho está relacionado ao *debouncing* (Figura 3) produzido pelos *push-buttons*, mesmo que o manual indique que esse problema é tratado por meio de um circuito *Schmitt trigger*. Devido a este fenômeno, o comportamento do sinal não se mantinha estável nas transições, produzindo falsas bordas. Com isso, ao pressionar o *key0* ou *key3*, usados no trabalho, o circuito se comportava de maneira inadequada, acionando outros módulos, como a ALU e o banco de registradores, mais de uma vez ou em momentos inapropriados, produzindo assim resultados errôneos. Com o intuito de corrigir esse comportamento inerente a toda chave mecânica, foi adicionado mais um módulo ao projeto – *debouncer.v*.

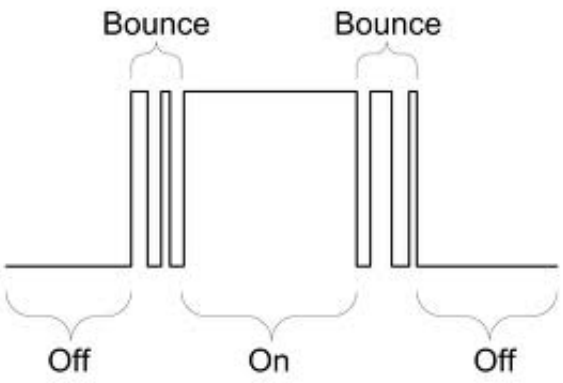

Figura 3 – Exemplo de *debouncing*.

O módulo em questão funciona monitorando o sinal em busca de mudanças no nível lógico. Ao detectar uma borda, o módulo ignora outras mudanças e após um periodo de tempo, tipicamente poucos milissegundos, ele confirma se o sinal de fato mudou de estado. Para isso, ele primeiro sincroniza o sinal com o *clock* proveniente da FPGA e, após sincronizado, utiliza um contador para criar a base de tempo para a reavaliação do sinal.

2.3.1 Sinais de Entrada e Saída

Para que o módulo implementasse suas funções corretamente, são necessários alguns sinais de entrada e saída, os quais são introduzidos pela Tabela 6 e Tabela 7, respectivamente.

Tabela 6 – Sinais de entrada utilizados pelo módulo *debouncer*

Sinal	**Tamanho (bits)**	**Tipo**	**Descrição**
clk	1	wire	Clock da unidade.
PB	1	reg	Sinal com *debouncing*.

Tabela 7 – Sinais de saída utilizados pelo módulo *debouncer*

Sinal	**Tamanho (bits)**	**Tipo**	**Descrição**
PB_state	1	wire	Sinal de saída, sem *debouncing*.

2.4 Display BDC 7 Segmentos

Por fim, o último módulo presente neste projeto tem como finalidade efetuar a decodificação de um *display* de 7 segmentos, onde são mostrados, de acordo com a especificação, os valores solicitados no formato hexadecimal. Para tanto, é recebido um valor de 4 *bits* codificado como BCD. A saída consiste de um registrador de 7 *bits*, onde a cada *bit* é atribuído o valor 0 ou 1 de acordo com o segmento que deverá ser ligado, possibilitanto assim a formação de todos os valores compreendidos entre 0x0 e 0xF.

2.4.1 Sinais de Entrada e Saída

A Tabela 8 e a Tabela 9 sintetizam os sinais utilizados por esse módulo.

Tabela 8 – Sinais de entrada utilizados pelo módulo de controle do *display* BCD de 7 segmentos

Sinal	**Tamanho (bits)**	**Tipo**	**Descrição**
hex_value	4	wire	Valor hexadecimal a ser escrito no *display*.

Tabela 9 – Sinais de saída utilizados pelo módulo de controle do *display* BCD de 7 segmentos

Sinal	**Tamanho (bits)**	**Tipo**	**Descrição**
disp_value	7	reg	Saída que contém a configuração para cada segmento do *display*.

3 Conclusão

Este trabalho apresentou o desenvolvimento de um sistema de *hardware*, o qual é constituído por um banco de registradores e por uma unidade lógico-aritmética (ALU). Para tanto, de forma a garantir a funcionalidade integral de cada módulo individual, foram realizados *testbenches* para cada um dos módulos, os quais foram simulados através do ambiente de simulação Intel Altera, utilizando os *softwares* Quartus e ModelSim. Além disso, também foram realizados testes em ambiente físico, o que foi possível através do pacote de desenvolvimento DE2-115 (Altera), possibilitando assim a prototipação do circuito desenvolvido. Neste contexto, este trabalho se mostrou importante para a consolidação dos conceitos teóricos vistos em sala de aula, pois proporcionou a realização das três fases principais para o desenvolvimento de um circuito: descrição, simulação e prototipação. Dessas fases, por fim, destaca-se a protipação, uma vez que permitiu aos alunos envolvidos a familiarização e execução de testes em ambiente real com a utilização de *hardware* reconfigurável.

4 Referências Bibliográficas

Patterson, David; Hennessy, John. Computer Organization and Design: the Hardware/Software Interface, 3th ed. Elsevier, 2005.

Tanenbaum, Andrew S.; Austin, Todd. Structured Computer Organization, 6th ed. Prentice Hall, 2012.