Segundo Exercício-Programa (EP 2)

MAC329 – Álgebra Booleana e Aplicações

Marcelo S. Reis [1]
Junior Barrera [1]
Gabriela E.F. Sanada [1]

[1] Instituto de Matemática e Estatística, Universidade de São Paulo.
msreis@ime.usp.br

11 de abril de 2013

1 Introdução

A partir deste Exercício-Programa (EP), implementaremos em um simulador lógico todos os componentes necessários para o funcionamento de um computador HIPO, máquina virtual decrita no EP anterior [1]. O simulador lógico a ser empregado para o desenvolvimento dos componentes será o Logisim [2].

Logisim é um programa que permite o desenho e a simulação de circuitos lógicos (e.g. Booleanos). Esse programa possui uma interface gráfica amigável e foi desenvolvido em Java (portanto, funciona em diversos sistemas operacionais). Logisim é disponibilizado sob a licença GNU-GPL e pode ser baixado na página oficial do projeto:

`http://ozark.hendrix.edu/~burch/logisim/index.html`

1.1 Objetivos

O principal objetivo deste EP é o desenvolvimento de uma Unidade Lógica e Aritmética (ALU – *Arithmetic and Logic Unit*), que futuramente será utilizada na simulação de um computador HIPO.

Para este EP, a ALU deverá fazer as seguintes operações aritméticas sobre números inteiros: soma, subtração, multiplicação e divisão. Além disso, implementaremos verificações de ocorrência de transbordamento (*overflow*).

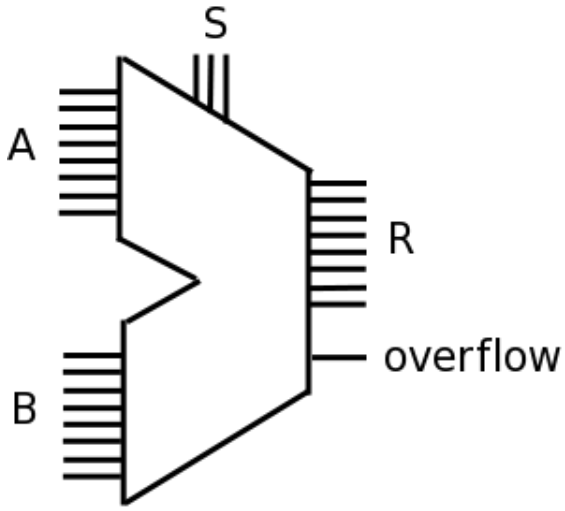

Figura 1: diagrama da ALU deste EP, no qual são exibidas as entradas e as saídas da mesma.

2 Descrição da ALU

A ALU deverá ter 8 bits e operar com complemento de 2 (o que implica que os valores possíveis de entrada e de saída estão contidos no intervalo $[-128, 127]$). Um diagrama contendo os principais componentes dessa ALU é exibido na figura 1.

A implementação da ALU no Logisim deverá ser feita apenas com portas lógicas (*gates*), sem utilizar os recursos já prontos de aritmética (*Adder*, *Subtractor*, etc.). Será permitido o uso do multiplexador já pronto (*Multiplexer*), embora a sua utilização não seja obrigatória. Para o cálculo da multiplicação e da divisão, também é facultativa a utilização de registradores (para implementar um acumulador), de *Clock* e de *Shifters*.

2.1 Entrada

Para modificar os valores de entrada da ALU utilizaremos pinos de entrada (O item *Pin* na biblioteca *Wiring*). A ALU deverá ter os seguintes pinos de entrada:

- 8 pinos, nomeados $a_7 \dots a_0$, que representam a entrada A. a_7 e a_0 são, respectivamente, o bit mais e menos significativo.
- 8 pinos, nomeados $b_7 \dots b_0$, que representam a entrada B.
- 3 pinos, nomeados $s_2 \dots s_0$, que representam S, o seletor de operação.

A seleção de operação deverá ser feita da maneira mostrada na tabela 1.

S	Operação	Tipo de operação
000	$A + B$	aritmética
001	$A - B$	aritmética
010	$A \times B$	aritmética
011	quociente da divisão de A por B	aritmética
100	resto da divisão de A por B	aritmética

Tabela 1: descrição das operações que a ALU deve realizar, de acordo com valores de S.

2.2 Saída

A ALU deverá ter os seguintes pinos de saída:

- 8 pinos, nomeados $r_7 \dots r_0$, que representam a saída R.
- 1 pino, nomeado $overflow$, que representa a ocorrência de transbordamento.

O pino de saída que indica *overflow* deverá ter valor 1 caso o mesmo ocorra, e 0 caso contrário.

3 Entrega do EP e observações

3.1 Observações importantes

- Este EP deverá ser feito em grupos de 4 pessoas, de preferência mantendo a mesma equipe do EP anterior; em casos excepcionais e com autorização do professor / monitor, pode-se aceitar grupos com menos integrantes.
- O EP será corrigido de forma automatizada; portanto, é muito importante que o código siga **exatamente** os formatos de entrada e de saída.
- Minimize o número de portas lógicas de sua ALU; para isso, utilize mapa de Karnaugh e/ou o algoritmo de Quine-McCluskey.
- Documente adequadamente o seu circuito, através de comentários em um arquivo texto README que deverá acompanhar o código-fonte.
- Recomenda-se que as dúvidas sejam discutidas no fórum da disciplina no Moodle.

3.2 Data de entrega

Este EP deverá ser entregue até o dia **27 de abril (sábado)**, através do fórum da disciplina no Moodle. Após esse dia, o sistema não permitirá a entrega deste EP.

Deverá ser entregue um zip ou tar.gz, contendo o arquivo `.circ` do código da ALU descrita na seção 2, além de um arquivo README contendo o comentários que acharem

pertinentes para o entendimento da implementação. Apenas um membro do grupo deverá subir o arquivo; portanto, lembrem-se de colocar no arquivo README o nome completo e o número USP de todos os integrantes do grupo.

Referências

[1] Valdemar W. Setzer. The HIPO computer – a tool for teaching basic computer principles through machine language. `http://www.ime.usp.br/~vwsetzer/hipo/hipo-descr.html`, 2000.

[2] Carl Burch. Logisim – a graphical tool for designing and simulating logic circuits. `http://ozark.hendrix.edu/~burch/logisim/index.html`, 2001.