Projeto de circuito

MAC0329 – Álgebra booleana e aplicações (DCC / IME-USP — 2023)

– Todas as etapas do projeto deverão ser feitas usando o Logisim-evolution
https://github.com/reds-heig/logisim-evolution –

Parte 2: ULA – entrega no e-disciplinas, até 14/06

A partir desta parte, os EPs poderão ser feitos em grupos de até 3 membros.

ULA é a contração do nome Unidade Lógico-Aritmética. Ela é a parte da CPU que é responsável por executar as operações aritméticas e lógicas. Nesta etapa, construiremos um **comparador** e juntaremos com o somador feito na primeira etapa para construir uma ULA.

1 Especificação do comparador

Na nossa ULA, o comparador é um circuito que recebe dois números de 8 *bits* cada na entrada, que denotamos **A** e **B**, e produz na saída os resultados das comparações.

Vamos construir um comparador que terá 6 *bits* na saída, cada um representando um *flag* para indicar o resultado de diferentes comparações, conforme detalhado a seguir:

- $c_0 = 1$ se e somente se $\mathbf{A} > \mathbf{B}$, considerando-se a interpretação sem sinal de seus valores
- $c_1 = 1$ se e somente se $\mathbf{A} < \mathbf{B}$, considerando-se a interpretação sem sinal de seus valores
- $c_2 = 1$ se e somente se $\mathbf{A} == \mathbf{B}$ (isto é, se ambas as entradas são iguais; note que aqui a interpertação sem/com sinal não é relevante)
- $c_3 = 1$ se e somente se $\mathbf{A} == \mathbf{0}$ (isto é, se o valor da primeira entrada é zero; note que aqui também a interpertação sem/com sinal não é relevante. Também não é relevante o valor de **B**),
- $c_4 = 1$ se e somente se $\mathbf{A} > \mathbf{B}$, considerando-se a interpretação com sinal (complemento de dois) de seus valores
- $c_5 = 1$ se e somente se $\mathbf{A} < \mathbf{B}$, considerando-se a interpretação com sinal (complemento de dois) de seus valores

Sugestão: seguindo o discutido em aula, podemos implementar um comparador de *bits* (com o *bit* habilitador de comparação) e usar vários desses em cascata para implementar o comparador de números de 8 *bits*. Na cascata, a ideia é fazermos as comparações do *bit* mais significativo para o menos significativo. O desafio será ajustar os valores das *flags* de saída de acordo com as interpretações. Para ajudar a pensar em uma forma de fazer esses ajustes corretamente, segue a tabela dos números de 3 *bits* e as interpretações sem e com sinal:

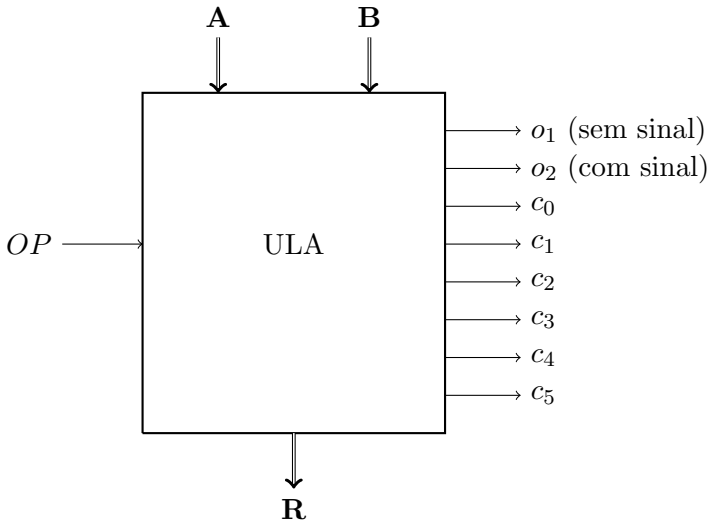
Binários	sem sinal	com sinal
0 0 0	0	0
0 0 1	1	+1
0 1 0	2	+2
0 1 1	3	+3
1 0 0	4	-4
1 0 1	5	-3
1 1 0	6	-2
1 1 1	7	-1

Será que podemos fazer um único circuito comparador (considerando-se a intepretação sem sinal) e então determinar c_4 e c_5 olhando o resultado desse circuito e o valor de mais alguns *bits* ?

2 Especificação da ULA a ser implementada

A ULA que iremos implementar será capaz de realizar apenas duas operações aritméticas (adição e subtração) e comparações de dois números, todos inteiros representados em 8 *bits*.

A ULA deve seguir o seguinte esquema quanto às suas entradas e saídas:



Entradas:

- **A** e **B** são os operandos, ambos de 8 *bits*;
- OP é um *bit* para indicar o tipo de operação: se 0, deve ser calculada a adição $\mathbf{A} + \mathbf{B}$ e se 1 deve ser calculada a subtração $\mathbf{A} - \mathbf{B}$; as comparações serão sempre calculadas;

Saídas:

- **R** é o resultado da operação aritmética (8 *bits*), como no EP1;
- o_1 e o_2 são os *flags* que indicam *overflow* na operação aritmética, para os casos de interpretação sem sinal e com sinal, respectivamente (como no EP1);
- c_0 a c_5 são os *flags* que indicam os resultados das comparações conforme especificados acima.

3 Modularização

De forma similar ao EP1, organize o circuito em módulos, cada módulo correspondendo a um subcircuito. Neste sentido, além dos circuitos e subcircuitos feitos na etapa 1, teremos o circuito e subcircuitos correspondentes à parte de comparação. A ULA deve encapsular a circuitaria que realiza as operações aritméticas e a circuitaria que efetua as comparações, e quando vista como "caixa-preta" deve obedecer a estrutura de entradas e saídas especificada acima.

Para testar a ULA, crie um circuito `main`.

4 Entrega e avaliação

Os circuitos devem ser implementados usando o software `logisim-evolution`. O circuito somador do EP1 pode ser reaproveitado integralmente.

Deve ser entregue o arquivo com a extensão `.circ`, gerado pelo `logisim-evolution`, contendo o circuito desenvolvido.

> Inclua o nome de todos os integrantes do grupo no circuito `main`. Apenas um membro do grupo deve fazer a entrega.

Serão avaliados os seguintes aspectos:

- corretude dos circuitos
- aderência à especificação
- organização (modularização) e clareza (com rotulação/comentários adequados).

Dúvidas ? Poste suas perguntas no fórum de Dúvidas/Comentários no moodle do e-disciplinas