MAC0329 – Álgebra booleana e circuitos digitais

DCC / IME-USP — Primeiro semestre de 2017

Projeto de circuito 1 (Data para entrega: até 30/04/2017)

A **ULA** (unidade lógico-aritmética) é a parte do processador responsável pelas operações lógicas e pelas operações aritméticas. O objetivo deste projeto é criar parte do circuito de uma ULA, conforme detalhado mais adiante. Note que a ULA que especificamos aqui é apenas um modelo simples.

Para o desenvolvimento do projeto deve ser usado o software `Logisim` (http://www.cburch.com/logisim/).

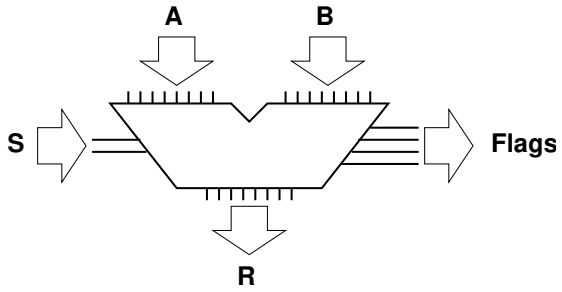
O projeto deve ser desenvolvido em grupos com 3 membros. Recomenda-se que os membros do grupo trabalhem juntos, principalmente na parte de planejamento e discussão. O trabalho pode ser dividido entre os membros, mas todos devem ter ciência sobre os detalhes do projeto.

A entrega do projeto será via PACA. Devem ser entregues o arquivo `.circ`, criado com o `Logisim`, e contendo o circuito da ULA, mais um documento (pode ser `txt` simples) contendo o nome dos membros do grupo e uma explicação sucinta e clara de como está organizado o circuito. Os dois arquivos devem ser empacotados em um único arquivo (`.zip` ou `.tar.gz` ou `tgz`).

Postem suas dúvidas ou descobertas no Fórum da disciplina.

Detalhamento

Vamos considerar palavras de 8 *bits*. Vamos também supor que todos os números estão na notação complemento de dois. Assim, os números que podem ser representados em uma palavra variam de -128 a 127. O esquema da ULA a ser implementada é mostrado na figura a seguir.



As entradas A e B na parte superior ($A = a_7\, a_6\, \ldots\, a_1\, a_0$ e $B = b_7\, b_6\, \ldots\, b_1\, b_0$) correspondem aos dois números a serem operados, enquanto a saída R na parte inferior ($R = r_7\, r_6\, \ldots\, r_1\, r_0$) corresponde ao resultado da operação (quando for o caso). Na lateral esquerda temos os pinos seletores S ($S = s_1\, s_0$), sendo s_0 o de cima e s_1 o de baixo, que servem para indicar a operação a ser executada, e na lateral direita temos alguns *flags* de saída.

Os seletores devem funcionar da seguinte forma:

$s_1\, s_0$	Operação a ser executada
00	Adição $(A + B)$
01	Subtração $(A - B)$
1*	comparação de A com 0

(na qual * indica qualquer um dos dois valores, 0 ou 1).

Já as *flags* de saída consistem de 4 *bits*, $o_3\, o_2\, o_1\, o_0$, sendo o_o a primeira e o_3 a última de cima para baixo. As *flags* indicam os seguintes estados:

$$o_0 = 1 \iff overflow$$
$$o_1 = 1 \iff A < 0$$
$$o_2 = 1 \iff A = 0$$
$$o_3 = 1 \iff A > 0$$

Note que para uma operação específica nem todas as saídas fazem sentido (por exemplo, o resultado em R e a saída em o_0 – que indica se houve ou não *overflow* – não importam se a operação é a de comparação). Você deve projetar o circuito de forma a garantir que as saídas correspondentes à entrada especificada estejam corretas (e não precisa se preocupar com a coerência das demais saídas).

Observações e dicas:

- O `Logisim` possui alguns módulos prontos. Neste projeto, porém, é esperado que todo o circuito seja implementado sem o uso de módulos prontos do `Logisim`, com exceção do multiplexador (seletor de dados).
- Planeje a organização do circuito antes de começar a desenhá-lo no `Logisim`. Uma boa prática é organizar seu circuito em módulos (subcircuitos). Assim, à medida que o seu circuito é incrementado, tornando-se cada vez mais complexo, os subcircuitos podem ficar "encapsulados" em uma caixinha, permitimdo uma organização hierárquica.

  Além da organização hierárquica, pense também o leiaute do desenho sem si.
- Para se acostumar com o `Logisim`, além do projeto0, experimente inicialmente fazer, por exemplo, um somador de números de 2 ou 3 *bits*.
- Em certas partes do circuito pode ser conveniente utilizar *bits* de dados "largos", assim como os *splitters*. A figura a seguir ilustra um pino de entrada com 8 *bits* e o uso de *splitters* para separar (sub)grupos de *bits*.