µProcessador 2 ULA

Tarefa: Construir uma ULA

- Será utilizada no projeto do processador, portanto escolha operações sensatas
- Quatro operações no mínimo, incluindo obrigatoriamente adição e alguma forma de realizar subtração (seja via uma instrução de subtração ou uma instrução de inversão de sinal); pode fazer operações lógicas bit-a-bit, decremento, divisão, comparação,...
- O professor vai atribuir uma operação especial para cada uma das equipes
- Contas de no mínimo 4 bits, feitas com portas lógicas (não pode usar o componente ULA já pronto, nem somadores já prontos do Quartus II etc.); pode incluir flags (sinalizadores, como o "Zero" visto no circuito em aula) se quiser
- *Por favor* dê nomes claros aos sinais (p.ex., *op*, *selec, selec_op* ou *SelecionaOperacao* e não "slo")
- Dica: se você quiser, pode utilizar um componente com um MUX pronto; dê uma olhada em BUSMUX, MUX e LPM_MUX. Idem para seletor/DEMUX

Entrega Eletrônica (em teste beta)

- ~~Data recomendada: 6a feira da semana que vem até 23h59 (dá tempo de reenviar arquivos)~~
- ~~***Deadline:*** daqui a duas semanas, até 6a feira às 13h00~~
- Deadline: 13h00 da 6a feira da 1a semana de aula de 2014
- Enviar para o e-mail de trabalhos (não verei apresentações em aula)
- Formato: criar o arquivo texto de configuração, criar um .qar e enviar ambos em anexo
  - Os nomes dos arquivos e pastas ***não podem ter espaços ou acentos!*** (Renomeie e recompile se precisar). Use só A..Z, a..z, 0..9 e _
  - Crie um arquivo compactado .qar do projeto dentro do Quartus, usando o menu Project => Archive Project
  - Crie um arquivo de texto simples (bloco de notas, não Word) chamado "projeto.txt" com as definições de operações, projeto e arquivos. Abaixo, uma reprodução do meu arquivo:

```
NOME_PROJETO = lab2          # nome do projeto (é o arquivo .qpf)
SINAL_SEL_OP = sel_op        # nome do pino que seleciona a operação
SINAL_ENTR1 = a              # nome dos pinos da entrada 1 da ULA
SINAL_ENTR2 = b              # nome dos pinos da entrada 2 da ULA
SINAL_RESULT = saida         # nome dos pinos da saída da ULA
SINAL_ZERO = isEqual         # pino de saída da comparação [OPCIONAL]
LARGURA_ULA = 4              # com quantos bits se faz as contas
NUM_OPERACOES = 4            # quantas operações a ULA faz
OP0 = soma                   # lista das operações da ULA, em ordem
OP1 = subtracao              # de acordo com a seleção
OP2 = incremento             # ex: se os bits da seleção de operação forem
OP3 = xor                    # 01, a ULA faz subtração
```

  - Favor tentar obedecer à risca o formato acima e os nomes que estão em maiúsculas. Como estamos em *beta testing,* os nomes das operações podem variar um pouco, mas vamos tentar combinar

Testes Sugeridos

- Não serão avaliados, mas podem ajudar no seu debugging (vou fazer meus próprios testes)
- Agrupe as entradas, sem visualização dos bits individuais e retirando sinais de debug
- Faça visualização em decimal sinalizado para operações aritméticas (subtração, comparação) e em *hexadecimal* para operações lógicas (XOR, AND), se houver
- Pode fazer um arquivo .vwf para cada operação ou fazer um .vwf único pra todas
- Um exemplo, com valores diversificados para teste de soma de 4 bits:

	tempo => ...													
Entr1	0	0	0	1	1	-1	-8	5	4	4	-5	0	16x randômico	-2
Entr2	0	1	-1	0	-1	0	-8	1	3	-2	3	contagem -8 a +7	2	16x randômico
Result	0	1	-1	1	0	-1	0	6	7	2	-2	...	...	...

Questões sobre Testes (mostrar no papel o ★)

Para as questões abaixo, assuma que até 512 combinações de entrada são simuladas em um tempo razoável pelo Quartus e podem ser inspecionadas diretamente por um ser humano. Mais do que isso, o processo torna-se lento e cansativo, embora ainda possível.

1. Elabore uma sequência de testes para um somador de 3 bits
2. ★ Elabore uma sequência de testes para um subtrator de 8 bits
3. Elabore uma sequência de testes para uma ULA de 4 bits com operações de soma e XOR, incluindo um sinalizador de overflow e outro de carry.

Ainda:

- Os testes devem cobrir casos tipicamente interessantes ("será que subtraindo -1 de 0 teremos mesmo -1?") além de casos ordinários (soma 1, soma 0, soma -1 com -1...)
- Ter noção de erros típicos é bom (talvez um bit de entrada do circuito não tenha sido ligado, o que causaria uma falha numa soma 1111_2+1111_2, por exemplo)
- Sempre é bom deixar o simulador sortear alguma coisa em algum momento, afinal nossa cobertura de valores não pode ser perfeita e totalmente abrangente

Ponto Flutuante (mostrar no papel o ★)

Os números em **ponto flutuante** são representados em binário e quebrados em sinal, mantissa e expoente da seguinte forma: $3{,}1416_{10} \simeq 11{,}001001_2$ pois é $2^1+2^0+2^{-3}+2^{-6} = 2+1+0{,}125+0{,}015625 = 3{,}140625_{10}$ (perceba a aproximação).

Podemos representar π em *float* de 12 bits como $1{,}10010010_2*2^1$ (notação científica binária, similar ao formato $n*10^x$ em decimal). O primeiro 1 (o MSB) sempre fica implícito, pois ele sempre existe, então os dados são 10010010. Como o expoente é 1, temos:

sinal	expoente	mantissa
0 (positivo)	001	10010010

Como outro exemplo, o número binário 111*000100010* (mantissa destacada) significa em float decimal $-1{,}00100010_2*2^{-2} = -0{,}0100100010_2 = 0{,}25+0{,}03125+0{,}001953125 = 0{,}283203125_{10}$.

Para conversão de decimal para ponto flutuante, o método mais simples é o seguinte: primeiro convertemos a parte inteira, depois vamos multiplicando a parte fracionária por dois, tomando o primeiro dígito. Por exemplo, para π=3.141593, temos 3_2=11, inicialmente. Então:

0.141592*2=**0.**283184 retiramos o 0
0.283184*2=**0.**566368 retiramos o 0
0.566368*2=**1.**132736 retiramos o 1
0.132736*2=**0.**265472 retiramos o 0
0.265472*2=**0.**530944 retiramos o 0
0.530944*2=**1.**061888 retiramos o 1
0.061888*2=**0.**123776 retiramos o 0 e assim por diante.

Montando em ordem, finalmente temos $\pi=11.0010010_2$. Alternativamente, podemos ir calculando na mão o peso de cada bite testar a adição (ex.: o primeiro bit vale 2^{-1}=0.5, então 11.1_2=3.5 que é maior que π; portanto este bit deve ser zerado, e assim por diante).

A respeito deste formato especificado acima:

1. Converta 123,456 e -0,000456 para o formato e 010100010001 para representação decimal.
2. ★Quais os valores máximo e mínimo representáveis? Quantos dígitos temos de precisão?

3. Descreva um circuito somador e um subtrator para números em ponto flutuante.
4. Descreva um circuito multiplicador em ponto flutuante. Estime o número de clocks gastos.

Note-se que os formatos padrão (como o IEEE 754-2008) possuem algumas diferenças, entre elas a representação do expoente em excesso *(bias),* para facilitar comparações.

O Cantinho da Amnésia: a ULA, MUX...

Gaste um tempo pensando no circuito, vale a pena. Em geral as fichas caem. Sobre Muxes, consulte o caderno de Sistemas Digitais.

Se você continua não entendendo, temos duas referências: o livro-texto, nosso bom e velho Patterson-Hennessy, ou o livro do Vahid, que contém uma explicação mais detalhada (tem na biblioteca de Campo Mourão: *Frank Vahid*, "Sistemas digitais: projeto, otimização e HDLS", seções 4.9 e 4.10). É pra ter a coisa nos outros livros de digital também.