µProcessador 3 Banco de Registradores e ULA

O banco descrito nesta folha é de um típico processador RISC, que é o recomendado para este projeto, pois costuma ser mais simples (mas nem sempre...) e segue o esquema usado pelo livro. Mas vocês podem usar outras opções. Ao final há uma seção breve descrevendo um pouco de arquiteturas CISC.

O número de bits de largura (quantos bits cada registrador guarda) deve obviamente ser compatível com a largura da ULA e das instruções de carga de constantes. Evitem mudar isso depois.

Tarefa: Construir um Banco de Registradores e Conectá-lo à ULA

- Quatro registradores no mínimo, com no mínimo 4 bits cada
- *Atenção:* utilizem *flip-flops/registradores* sensíveis a *borda de subida*[1]
- Pode ou não utilizar um dos registradores fixo com a constante zero; se não tiver (o que é okay) será necessária futuramente uma instrução para carregar uma constante em um registrador
- Entradas (pinos):
  - Seleção de qual registrador será lido (x2, pois temos duas leituras simultâneas)
  - Barramento de dados para escrita
  - Seleção de qual registrador será escrito
  - Habilitação de escrita *(clock* de escrita)[2]
  - Pino de RESET dos registradores (zera todos eles)
- Saídas (pinos):
  - Barramento com os dados do registrador lido (x2)
  - Inclua um pino no *top level* com a saída da ULA
- Obrigatoriamente criem um bloco para a ULA[3], um bloco para o banco e usem ambos integrados no arquivo *top level* para testes (ou seja, serão *três ou mais* arquivos .bdf de esquemático)
- Obrigatoriamente façam um *reset* explícito de todos os registradores/flip-flops no início da simulação, caso contrário o Quartus não consegue determinar certos valores
- **Jamais deixe entradas de componentes em aberto!** O .vwf funciona, mas a *minha correção* não funciona porque a ferramenta é diferente. Fora que é errado fazer isso.
- *Dica:* pense no circuito antes de sair colocando componentes; é típico o aluno julgar que precisa de mais MUXes do que necessário
- Dica: você pode **visualizar os registradores** na simulação, inserindo-os diretamente no .vwf, sem pinos extras: no *Node Finder,* selecione *Filter: "Registers: Post Fitting"* e *List.* Agrupe os bits manualmente.

Sobre o Projeto do Microprocessador

O objetivo final é escrever e botar pra funcionar um programa; se me faltar criatividade, vai ser a determinação dos números primos. Se usar o crivo de Erastótenes, não precisa de divisão (!). Se fizer de outro jeito, a divisão pode ser feita na unha (com loops e somas/subtrações), bem como a multiplicação: não é necessário ter instruções nativas MUL e DIV, que provavelmente dão muito trabalho. Mas podem implementá-las, se quiserem (tem até componente pronto na biblioteca do Quartus).

1 Sensibilidade a borda é mais fácil de trabalhar do que sensibilidade a nível. A borda de subida é porque as FPGAs usualmente têm componentes que transicionam na subida e não na descida.

2 Outra alternativa, também razoável, é ter um *clock* "geral" para o banco de registradores e um bit de seleção de operação que escolhe entre escrita e leitura.

3 Dá pra desenhar o símbolo da ULA (a "bermuda de lado") no bloco: botão direito, "Edit Selected Symbol"(opcional)

Dicas:

- *As instruções e os dados não precisam ser da mesma largura:* podemos ter instruções de 12 bits e dados de 8 bits, por exemplo
- Perceba que ter poucos registradores facilita o circuito mas pode complicar a programação, e ter muitos registradores pode complicar o circuito apesar de facilitar a programação
- Teste primeiro o banco de registradores em separado e *só depois* integre com a ULA, pois a simulação não aceita sinais internos de blocos (só simula *pinos* dos blocos)

Entrega Eletrônica

- ***Deadline:*** de acordo com o plano de aulas, até 6a feira às 13h00
- Enviar para o e-mail de trabalhos (não verei apresentações em aula) com três arquivos em anexo:
  - O arquivo compactado deverá ser um .qar *criado dentro do Quartus,* usando o menu Project => Archive Project
  - O arquivo de simulação .vwf, separadamente do .qar
  - O arquivo projeto.txt, como descrito abaixo
- O arquivo texto de definições ("projeto.txt") deverá estar anexado separadamente no mesmo e-mail
  - Siga os padrões anteriores; se mudar sinais da sua ULA, reflita isso no arquivo texto.
  - Se os sinais abaixo não são suficientes para a sua solução, ou são diferentes, não tem problema. Me explique o caso no corpo do email e eu me viro
  - Adicione no arquivo texto as definições de número de registradores e o nome dos sinais do banco: dados a escrever, pulso de escrita, seleção dos registradores a ler e os dois sinais de saída (se for fazer CISC, me avise e improvise um formato, eu me viro)

```
NOME_PROJETO = lab3      # atualizar!
...                      # definições anteriores
OP3 = xor                # última definição do lab uProc #2
NUM_REGS = 4             # número de registradores no banco
DADO_IN = Write_Data     # nome do sinal do dado a escrever no registrador
SEL_REG1 = Reg_Rd1       # sinal que seleciona qual o registrador 1 a ler
SEL_REG2 = Reg_Rd2       # sinal que seleciona qual o registrador 2 a ler
SEL_REGWR = Reg_Wr       # sinal que seleciona qual o registrador a escrever
RESET = Reset            # sinal de clear para todos os registradores
# abaixo há opções de sinais para gravação - nenhum é obrigatório em si
CLOCK = clock            # opção ck1: clock geral do circuito
WRITE_EN = Write_En      #            com enable para a operação de escrita
PULSO_WR = ck_wr         # opção ck2: pulso exclusivo para a escrita do dado
DATA_IN_MUX_SEL = MxSel  # alternativa de fonte de dados: caso seja usado um
                         # MUX para selecionar a fonte do dado (pode vir da
                         # ULA ou do pino DADO_IN), este é o sinal de seleção
```

Apêndice: Arquiteturas CISC com *Specific Purpose Registers*

Num processador CISC típico, todas as contas são feitas com um registrador especial chamado *Acumulador*. O outro operando da ULA vem de um registrador ou é imediato *(addi).* O resultado da operação da ULA será escrito no acumulador, ou seja, as únicas operações são obrigatoriamente *add acc,acc,reg* ou *addi acc,acc,cte*. Neste caso, *li* deve ser uma instrução de fato (com *opcode* e tudo) ao invés de uma pseudo-instrução, para carregar os outros registradores.

Alternativamente, alguns processadores não permitem operações do tipo *addi:* um registrador auxiliar deve ser carregado com a constante para a operação ser realizada, ou seja, ao invés de *addi acc,acc,1*, deve-se fazer *li b,1* seguido de *add acc,acc,b*, sendo B um registrador.

Outros processadores permitem que o resultado da ULA seja gravado diretamente em um registrador diferente do acumulador, mas não necessariamente qualquer registrador do banco.

Caso use um esquema diferente do RISC descrito no livro, por favor faça testes sensatos para a apresentação.

Questões Extras (mostrar no papel o ★)

1. Os circuitos de multiplicação são particularmente problemáticos, pois os cálculos demoram. Estime o número de clocks para uma multiplicação sinalizada de 32 bits usando um algoritmo/circuito ingênuo.
2. Uma otimização possível de um circuito de multiplicação cria inicialmente o produto de cada um dos bits do multiplicando por cada um dos bits do multiplicador (resultando em 32x32=1024 bits). A cada um deles é designado um peso, e a cada passo duas "camadas" são combinadas, resultando em O(log n) ou O($\log^2$ n) passos, sendo n=número de bits=32. Compare esta alternativa com a descrita na questão acima.
3. ★ Números representados em ***ponto fixo*** são números racionais binários em que se arbitra uma posição para o ponto decimal. Por exemplo, se tivermos 12 bits e arbitrarmos o ponto entre b4 e b5 (b0 é o LSB), o número 100010001000 significa $1000100{,}01000_2$, ou seja, $2^6+2^2+2^{-2} = 64 + 4 + 1/4 = 68{,}25$. Converta o número 123,456 para este formato de 12 bits[4] e dê o equivalente decimal de 001001010101.
4. Como ficam os circuitos de soma e subtração para números em ponto fixo? E multiplicação? Quais as vantagens de ponto fixo em relação a ponto flutuante?
5. Quantos dígitos temos de precisão decimal neste formato de ponto fixo de 12 bits?
6. Perceba que os números fracionários podem não ter representações exatas: 2/3=0,666... que é aproximado para 0,6667 em quatro casas decimais. Qual é a representação mais próxima que podemos ter de 1/5 no nosso formato de ponto fixo? Qual o erro inerente ao formato? Perceba que podemos codificar arredondando o valor para cima ou para baixo.

"Desculpa a Nubagem, mas Banco de Registradores? WTF?"

Antes de mais nada, revise seu material de Sistemas Digitais (sim, o professor da matéria deu subsídio, registradores, muxes e barramentos, embora não o circuito em si).

Como para a ULA, o Patterson-Hennessy pode quebrar um galhão, e o livro do Vahid é bem explicado (biblioteca de Campo Mourão: *Frank Vahid*, "Sistemas digitais: projeto, otimização e HDLS", seções 4.9 e 4.10). Bons livros de digital abordam o assunto.

4 Pode utilizar a descrição de conversão para números fracionários em base 2 da folha de laboratório da semana passada.