Laboratório 6

Instruções:

Antes de iniciar o laboratório faça o download do arquivo 'lab06_material_v2018.1.zip' no moodle, ele contém arquivos de apoio para a execução dos exercícios, incluindo modelos para arquivos da entrega. Caso algum arquivo da entrega não possua um modelo, o formato é livre, desde que respeitados os nomes dos arquivos.

Os enunciados dos exercícios descrevem os nomes e tipos dos arquivos, entidades e conexões. Quaisquer entregas que não seguirem o padrão fornecido serão **desconsideradas**.

Deixe os arquivos que serão gravados na placa para demonstração em projetos separados e já pré-compilados, de forma que não seja necessária a recompilação do projeto no momento da demonstração. Assim, basta abrir o projeto correspondente e programar a placa, economizando o tempo da aula dedicado às avaliações.

Os arquivos que estiverem carregados no Moodle no momento do prazo final de entrega serão considerados como finais, mesmo que estejam marcados como rascunho. Portanto, não é necessário e nem recomendável marcar o envio como final, para que seja possível alterar os arquivos da submissão antes do prazo.

1. A entidade *latches_ffs* <latches_ffs.vhd> instancia 6 dos elementos de memória dados em aula:

- Latch SR com NANDs (entidade *latch_sr_nand*);
- Latch SR Chaveado (entidade *latch_sr_gated*);
- Latch D Chaveado (entidade *latch_d_gated*);
- Flip Flop tipo D (entidade *ff_d*);
- Flip Flop JK (entidade *ff_jk*);
- Flip Flop tipo T (entidade *ff_t*).

As entradas e saídas de cada componente foram conectadas conforme descrito no arquivo VHDL. Aos Flip Flops, foram adicionados sinais de *preset* e *clear*, que podem ser síncronos ou assíncronos, ambos ativos quando em nível lógico alto. Foi determinado que, caso ambos os sinais *preset* e *clear* estejam ativos, o *clear* prevalece.

Uma simulação da entidade *latches_ffs* gerou a forma de onda abaixo:

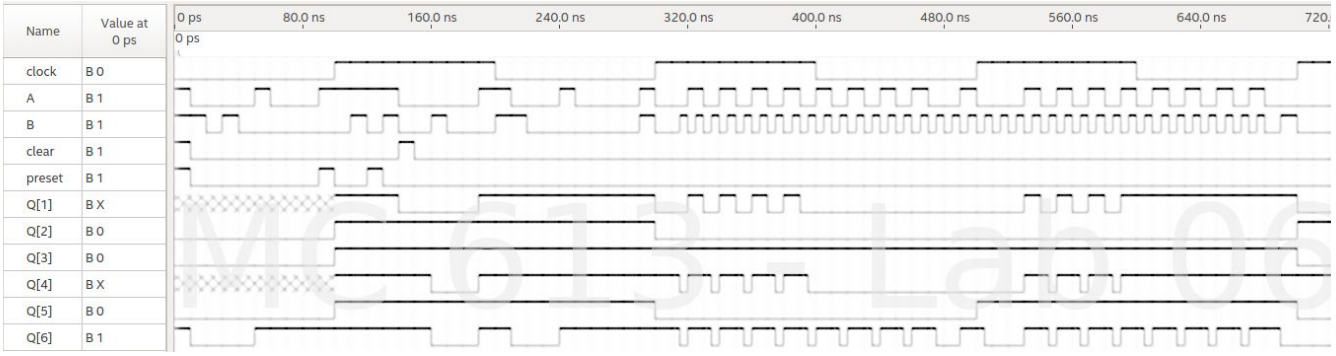

Figura 1: Forma de onda para a entidade *latches_ffs* (veja latches_ffs_base.png)

a) Identifique, de acordo com a forma de onda e as conexões dos componentes, qual saída Q corresponde à saída esperada de cada um deles. Altere o arquivo que descreve *latches_ffs* e inclua, nas linhas indicadas por "*CHANGE HERE*", os índices da saída identificada. Essa é a **única** alteração permitida no arquivo. **Entregue o arquivo latches_ffs.vhd resultante.**

b) Implemente cada um dos seis componentes em VHDL. Os componentes devem ser entregues em arquivos VHDL separados. Cada entidade **deve** respeitar o nome e o mapeamento de portas indicado no arquivo *latches_ffs.vhd*, e o nome de cada arquivo **deve** ser idêntico ao nome da entidade seguido de *.vhd*. **Entregue os seis arquivos .vhd.**

Dica: Lembre-se de verificar se o componente possui preset e clear síncronos ou assíncronos na forma de onda da figura 1, pois o seu componente deve implementá-los da mesma forma. Lembre-se que o clear toma precedência sobre o preset. Um componente pode utilizar outro como base, desde que ele utilize o pacote latches_ffs_pack (ver abaixo) e não sejam necessários arquivos adicionais. Lembre-se de produzir a saída desconectada dos circuitos (Qb, Q_n…). Embora não haja entrega de formas de onda de simulação, é altamente recomendável que simulações sejam feitas separadamente para cada componente.

c) Crie o pacote *latches_ffs_pack* <latches_ffs_pack.vhd> contendo os seis componentes descritos no item (b). **Entregue o arquivo latches_ffs_pack.vhd.**

d) Simule a entidade *latches_ffs* incluindo os arquivos do seu pacote (c) e dos componentes (b) no projeto. Tente reproduzir a forma de onda da Figura 1. Essa simulação é semelhante à que será feita para avaliação e pode indicar se o seu envio está correto ou não. **Entregue um screenshot dessa simulação em latches_ffs.png.**

2. Seja o registrador de n bits mostrado na Figura 2.

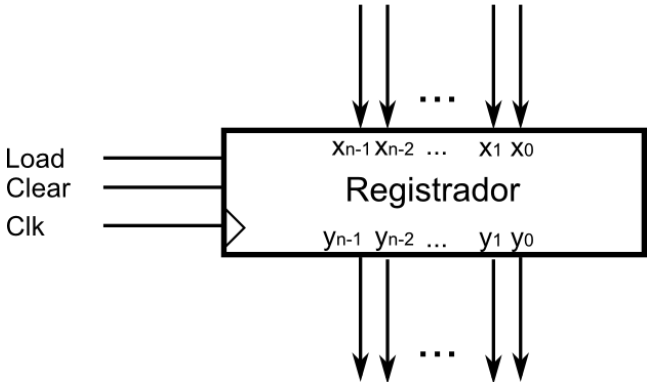

Figura 2: Registrador de n bits

a) Implemente na entidade *reg* <reg.vhd> um registrador genérico de n bits com carga (*load*) síncrona, *clear* assíncrono e leitura assíncrona. **Entregar o arquivo reg.vhd e resultados de simulação em um arquivo reg.png.**

b) Implemente, para utilizar no circuito do item (c), um decodificador 3x8 e um buffer tri-state para palavras de 4 bits. **Entregar os arquivos dec3_to_8.vhd e zbuffer.vhd.**

c) Utilizando os circuitos implementados nas partes (a) e (b) **como componentes**, implemente um banco de registradores com 8 registradores de 4 bits na entidade *register_bank* <register_bank.vhd>. Tome como base o diagrama da Figura 3. **Entregar o arquivo register_bank.vhd e resultados de simulação em um arquivo register_bank.png.**

d) Instancie o projeto do item (d) na placa, utilizando as 10 toggle switches como entrada de dados e seleção de registrador para ser escrito e lido (conforme a Figura 3). Utilize os push buttons (entradas KEY - atenção para a lógica invertida) para os sinais *Clear*, *Load* e *Clock*. Mostre o valor armazenado no *Display* de 7 segmentos hexadecimal. Verifique o funcionamento do reset assíncrono e do *Load* síncrono. **Não é necessário entregar nenhum arquivo, esta parte será avaliada na demonstração.**

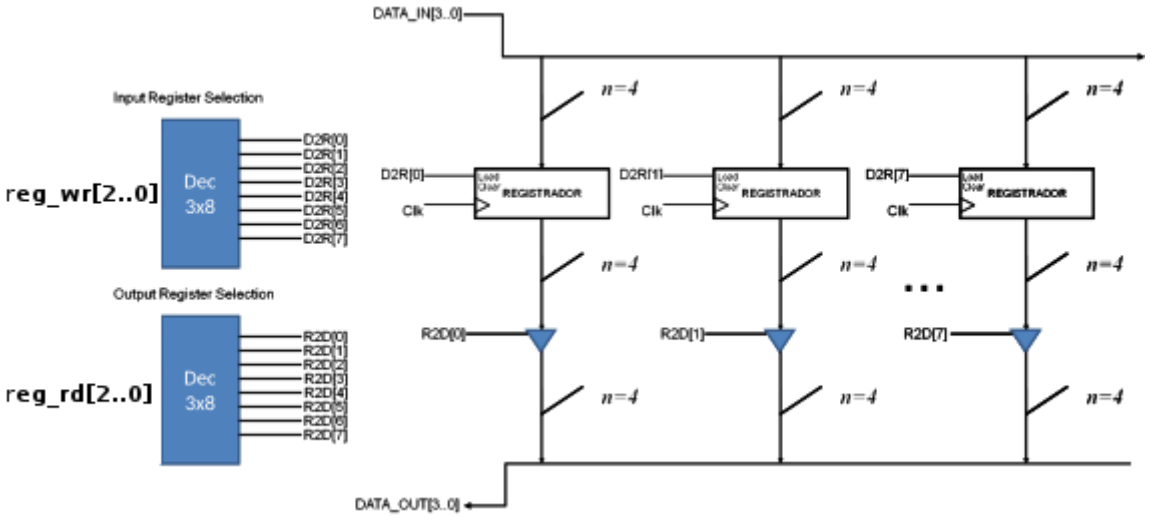

Figura 3: Banco de registradores (8 regs de 4 bits)

3. Este exercício utiliza os componentes para controle de teclado *kbdex_ctrl* e *ps2_iobase* disponíveis na pasta de "*Material complementar* > *Componentes IO DE1-SoC (nova)* > *PS2 (mouse e teclado)"* do curso no Moodle. Antes de fazê-lo, leia o tutorial *"Mouse e Teclado.pdf"* na pasta *"Material complementar* > *Tutoriais IO DE1 (antiga)"*. O tutorial foi escrito para um modelo anterior da placa utilizada, portanto existem algumas pequenas diferenças para o código disponível, que foi adaptado para a placa nova.

a) Faça o download dos componentes *kbdex_ctrl*, *ps2_iobase* e *ps2_kbd_test* da pasta de material complementar. Inclua no mesmo projeto o seu componente *bin2hex* (Laboratório 02) e, utilizando *ps2_kbd_test* como entidade top-level e o arquivo de assignments padrão para a placa, grave o projeto na placa para testar com um teclado externo.

b) Utilizando a placa e o teclado conectado, compare o valor do *scan code* exibido com os disponíveis na tabela de *scan codes* disponível em *"Material complementar* > *Tutoriais IO DE1 (antiga)* > *Keyboard - Scan Codes.htm"*, considerando as seguintes teclas:
   - As 26 letras do alfabeto (A - Z);
   - Os 10 caracteres numéricos (0 - 9);
   - Os 10 números do teclado numérico (KP 0 - KP 9);
   - Shifts esquerdo e direito (L SHFT e R SHFT);
   - Caps lock (CAPS).

   Caso exista alguma diferença, crie um arquivo scancodes.txt contendo uma lista **facilmente compreensível** de teclas e o *scan codes* diferentes. Veja as orientações de entrega abaixo.

c) Implemente a entidade *kbd_alphanum* <kbd_alphanum.vhd>. O objetivo desta entidade é receber do componente *kbdex_ctrl* o(s) *scan code(s)* da(s) tecla(s) pressionada(s) e exibir, em visores de sete segmentos, o código ASCII do caractere alfanumérico pressionado, ou não mostrar nada caso não haja caractere pressionado. Se houver várias teclas pressionadas em simultâneo, exiba o valor apenas da primeira, com exceção da primeira tecla ser o SHIFT. Neste caso, exiba o valor da segunda.
   Para esta tarefa, você pode criar quantas entidades e/ou pacotes (*packages*) adicionais julgar necessário, e implementá-los em quantos arquivos VHDL achar necessário. Não utilize representações em BDF ou outras HDLs, somente os arquivos .vhd serão considerados. Não crie ou utilize *libraries* customizadas, mas você pode utilizar as *libraries* disponíveis na ferramenta.
   Dicas e simplificações:
   - Lembre-se que a tecla SHIFT tem função invertida quando CAPS LOCK está ativo;
   - Você pode assumir que SHIFT e CAPS LOCK não exercem efeito em teclas numéricas;
   - Você pode atribuir e comparar *std_logic_vectors* com valores em hexadecimal usando a notação: `x"8F"`, onde 8F é o valor hexadecimal.
   - Utilize decodificadores para converter o valor do *scan code* para ASCII.
   - Guarde o estado de SHIFT e CAPS LOCK em Flip Flops para controlar a decodificação.

d) Utilize a entidade *kbd_alphanum_board* <kbd_alphanum_board.vhd> como top-level, sem alterações, e o arquivo de assignments padrão para a placa e grave seu circuito na placa, adicionando seus arquivos VHDL do item (c) e os componentes *kbdex_ctrl* e *ps2_iobase* no projeto. Teste o funcionamento do seu circuito. **Esta configuração será avaliada na demonstração.**

Entrega:

Reúna todos os arquivos VHDL do item (c) e o arquivo scancodes.txt do item (b) (se necessário) em um único arquivo zipado kbd_alphanum.zip. Os arquivos deverão estar na raiz da estrutura do .zip, e não distribuídos em pastas.

Importante:

Não haverá avaliação parcial para envios incompletos, portanto certifique-se de que você adicionou todos os arquivos VHDL para o funcionamento correto do seu projeto. Certifique-se, também, de não alterar a declaração da entidade *kbd_alphanum.*