Projeto 2- PSI (Documento 1)

1. Instruções Básicas

O Projeto-2 corresponde à implementação do sistema Snake completo. Ele deverá estar totalmente funcional, testado e. eventualmente, demonstrado em placa e monitor de vídeo.

Neste momento, é de fundamental importância que (a) aluno(a) tenha claro o quadro completo do projeto Snake e, para isto, deve familiarizar-se com a microarquitetura do sistema, descrita na apostila descritiva da funcionalidade e estrutura do Snake, fornecida já na aula 1.

Até a finalização do projeto pelo(a) aluno(a), cada uma dos seguintes conjunto de tarefas deverão estar devidamente concluídas:

Parte A: Revisar alguns blocos do Snake, completamente funcionais e já fornecidos em aulas anteriores: counter (aula 3); alu com ripple_carry_adder (aula 5); comparator e ofc (aula 6), fifo e mem (aula 9). Já é assumido que estão corretos, não havendo necessidade de re-simulá-los; basta relembrar o seu funcionamento.

Parte B: Consolidar a implementação e simulação dos módulos que ficaram sob a sua responsabilidade dos alunos: reg_bank (aula 2), fsm_main (aula 3) e num_gen (com lfsr geração de números aleatórios) (aula 4 e 7). Devem ser terminados, integrados individualmente e garantir o seu funcionamento a partir de testbenches.

Parte C: Estudar e simular os módulos ainda não trabalhados em aulas anteriores: button_handler, fsm_init, fsm_step. Devem ser testados individualmente através de testbenches a serem construídos pelos alunos. Em particular, o módulo fsmd_food, já trabalhado na aula 6, deverá ser revisto (com os mesmos estados e transições, porém com um número maior de saídas).

Parte D: Integração de todos as partes e o projeto topo completo deverá funcionar com o testbench padrão fornecido pelos professores.

Parte E: Síntese no Leonardo Spectrum e no Quartus, com dados comparativos .

Parte F: Programação do projeto no FPGA e demonstração de seu funcionamento com uma tela de computador (VGA). Todas os módulos para a gerenciamento dos dados na placa estarão prontas e serão fornecidas aos alunos.

Apesar de haver orientações para cada etapa nas aulas 12 a 14, os módulos para as Partes C e D serão disponibilizados o quanto antes e os alunos devem seguir a sequência de atividades o mais rápido possível.

O projeto poderá ser feito individualmente ou em duplas, havendo diferenças de cobrança de resultados nos dois casos:

a) individual- obrigatórios:

- a realização de A a E (a execução da parte E não é obrigatória, mas se realizada, dará créditos extras).
- apresentação dos testbenches da parte B

- certificação da funcionalidade no item D
- relatório geral

b) em dupla- obrigatórios (deixar clara a divisão de trabalho):

- a realização de A a F
- apresentação dos testbenches da parte B
- apresentação dos testbenches da parte C
- certificação da funcionalidade no item D
- relatório geral
- demosntração/apresentação do circuito funcional.

2. Parte B

2.1. reg bank

Usar o arquivo VHDL (multiplexador de lógica combinacional) *reg_bank_simplificado_1.vhd*, da aula 2, e estendê-lo para funcionar como na Figura 1, denominando-o como *reg_bank.vhd*. Funcionará como ilustrado também na figura 3 do texto ***snake.pdf***.

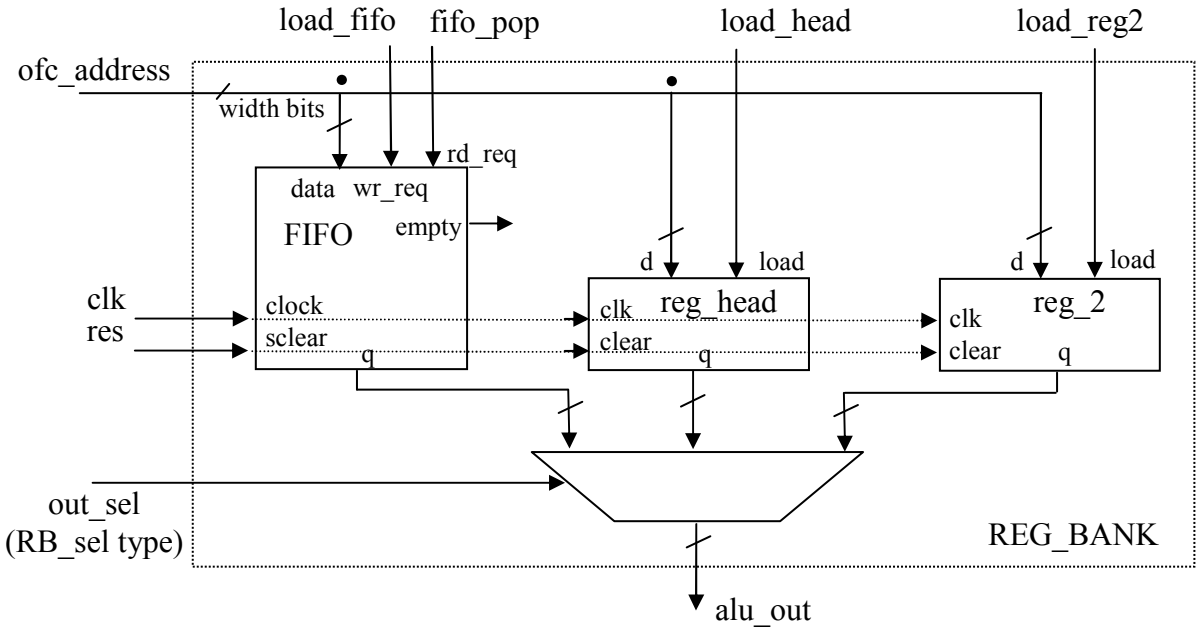

Figura 1. reg_bank - banco de registradores

O reg_bank deverá ser composto pelos módulos reg e fifo. O módulo reg foi fornecido junto ao material da aula 6 (componente do datatpah) enquanto o fifo é aquele gerado como Megafunction na aula 9.

Testbench:

Providencie um testbench nos moldes propostos na disciplina para uma verificação segura do módulo.

A fifo deve ser testada com mais cuidado. Lembre-se que inicialmente, ela deve receber o endereço inicial da cabeça da cobra e, posteriormente, este valor é modificada

(à medida que a cobra anda). Para o teste, dois números deverão ser gerados como o endereço inicial e final a ser inserido como dado da fifo. Faça da seguinte forma

number_m= número_USP_aluno_1 mod 64.

number_n= número_USP_aluno_2 mod 64.

Sendo dois alunos, number_m é o maior e number_n, o menor.

Depois, aplique a seguinte regra:

a) se (number_m - number_n >= 30) = TRUE, OK;

b) se (number_m - number_n >= 30) = FALSE, adicione múltiplos de 10 ou subtraia múltiplos de 10 a number_m e number_n respectivamente, até que a condição se torne verdadeira (Obs. number_m < 64 e number_n >=0)

Planeje os estímulos de tal forma que a cobra caminhe da menor posição para a maior (com pelo menos uma dobra), sendo que na primeira metade do caminho a cobra caminha e encontra o alimento e na segunda apenas caminha.

2.2. fsm_main

O arquivo *fsm_main.vhd* em **X:\psi3451\aula_3\fm_1**, que corresponde ao modelo ilustrado ao final da apostila prática da aula 3 deverá ser revisado e a sua correta funcionalidade deverá ser certificada.

Providencie um testbench nos moldes propostos na disciplina para uma verificação segura do módulo.

2.3. num_gen com lfsr

Corresponde ao seu projeto de LFSR como gerador de vetores aleatórios, realizado como no Projeto-1. Deverá ser integrado como um módulo datapath_simples como ilustrado na Figura 2. Observe que a entrada ctrl_ctrl deve ser adotada, com o tipo datapath_ctrl_flags, visto na aula 6.

Providencie um testbench nos moldes propostos na disciplina para uma verificação segura do módulo.

Se realizado em dupla, escolha uma das implementações realizadas no Projeto-1.

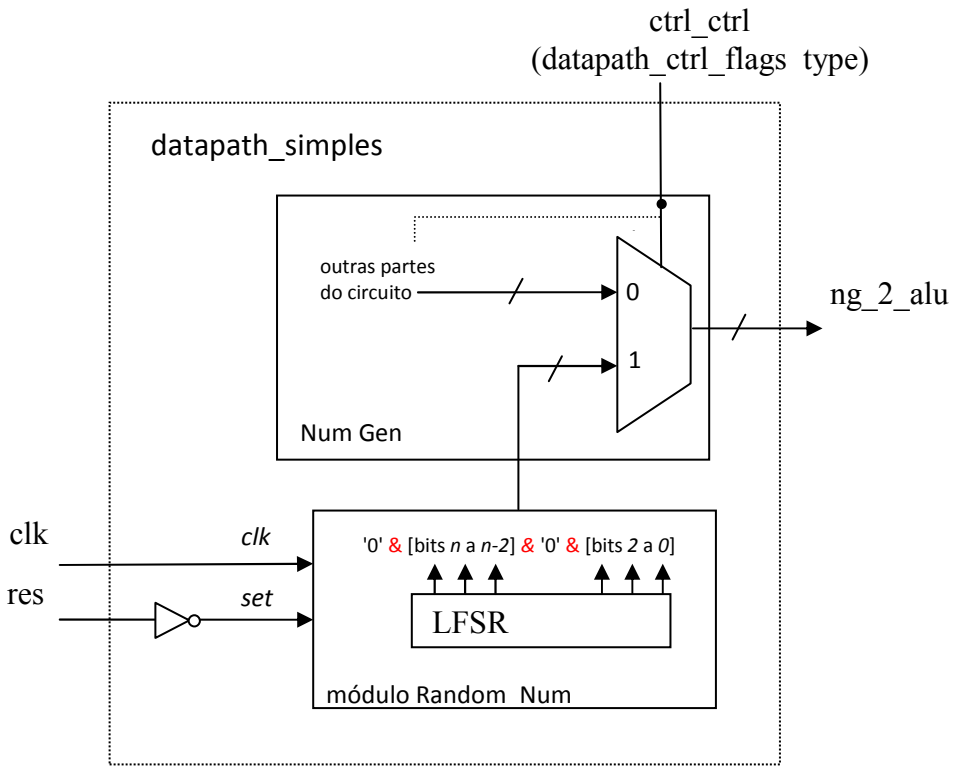

Figura 2. datapath_simples - topo similar ao datapath para teste de LFSR+num_gen