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); fs_main (aula 5); alu (aula 6), reg_bank e mem (aula 8). 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: fsm_main (aula 3) e num_gen (com lfsr geração de números aleatórios) (aula 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_food, fsm_init, fsm_step. Devem ser testados individualmente através de testbenches a serem construídos pelos alunos.

Parte D: Integração de blocos importantes: num_gen com o lfsr projetado pelo(a) aluno(a), controle contendo todas as FSMs, e o datapath com alu, comparador, etc. Gerar os testbenches e simular.

Parte E: Integração de blocos para a realização as partes e o projeto topo completo deverá funcionar com o testbench padrão fornecido pelos professores.

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

Parte G: 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 11 a 14, os módulos para as Partes C, D e E 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 F (a execução da parte G não é obrigatória, mas se realizada, dará créditos extras).

- apresentação dos testbenches da parte C e D, e asua simulações
- certificação da funcionalidade no item E
- relatório geral

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

- a realização de A a G
- apresentação dos testbenches da parte B, C e D, com as simulações
- certificação da funcionalidade no item E
- relatório geral
- demonstração/apresentação do circuito funcional na placa (todas os módulos para as interfaces serão dadas para os alunos- com a certificação do item E, o funcionamento na placa deve ser automático.

2. Parte B

2.1. button handler

Usar o arquivo *button_handler.vhd* no Serverlab (pela rede) e fazer uma análise de código. O button handler é um módulo ajuste dos sinais de direção, os quais devem ser fornecidos ao controlador do Snake. Como, ilustrado na Figura 1, estes sinais, denominados *direction_sync*, são as direções consolidadas que devem ser utilizadas pelo módulo *FSM_step*, já sincronizados com o passo do jogo, dado pelo sinal *load_regs* (correspondendo aos sinal indicativo de contagem máxima no módulo *step_counter*). A Figura 1 apresenta os portos de entrada e saída do módulo.

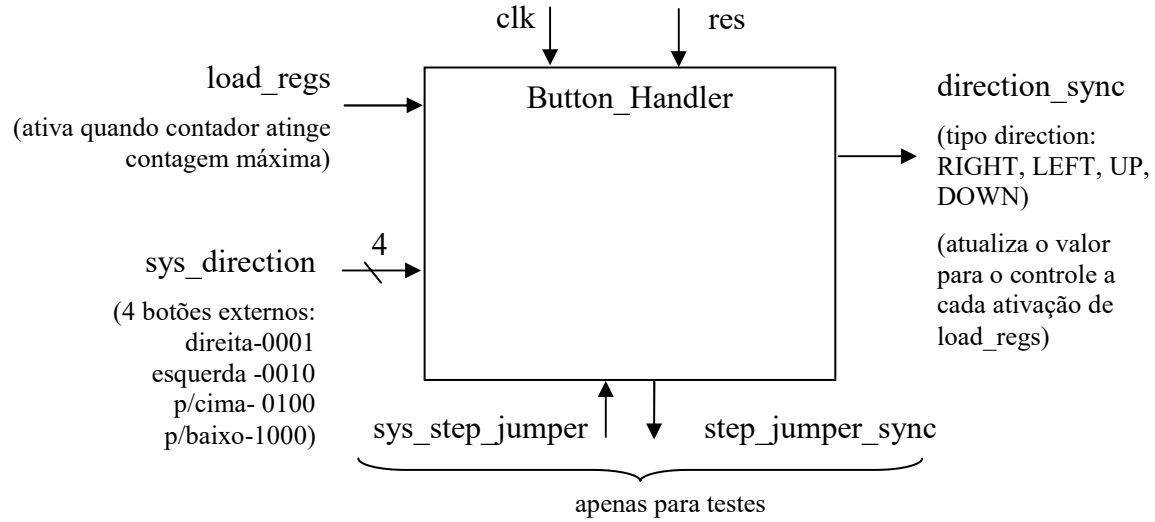

Figura 1. Digrama E/S do button handler

Vindo de quatro botões da placa, o usuário define, a qualquer momento, através do sinal *sys_direction* qual a direção que a cobra siga. O módulo *button handler*, além de sincronizar a saída, tem também a função definir a direção do novo passo, manter a mesma direção do passo anterior, em duas condições:

- o usuário não aperta o botão;

- o usuário escolhe, por distração, o sentido oposto ao do último passo, indo contra o próprio corpo, o que não é permitido.

Testbench:

Providencie um testbench nos moldes propostos na disciplina para uma verificação segura do módulo. Providencie comentários pelas linhas de código do arquivo de testbench que possam esclarecer os objetivos específicos dos comandos.

Planeje os estímulos de tal forma que todas as condições de entrada dadas pelo usuário ocorra.

2.2. 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. Veja do que se trata nos pacotes.

Testbench:

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.

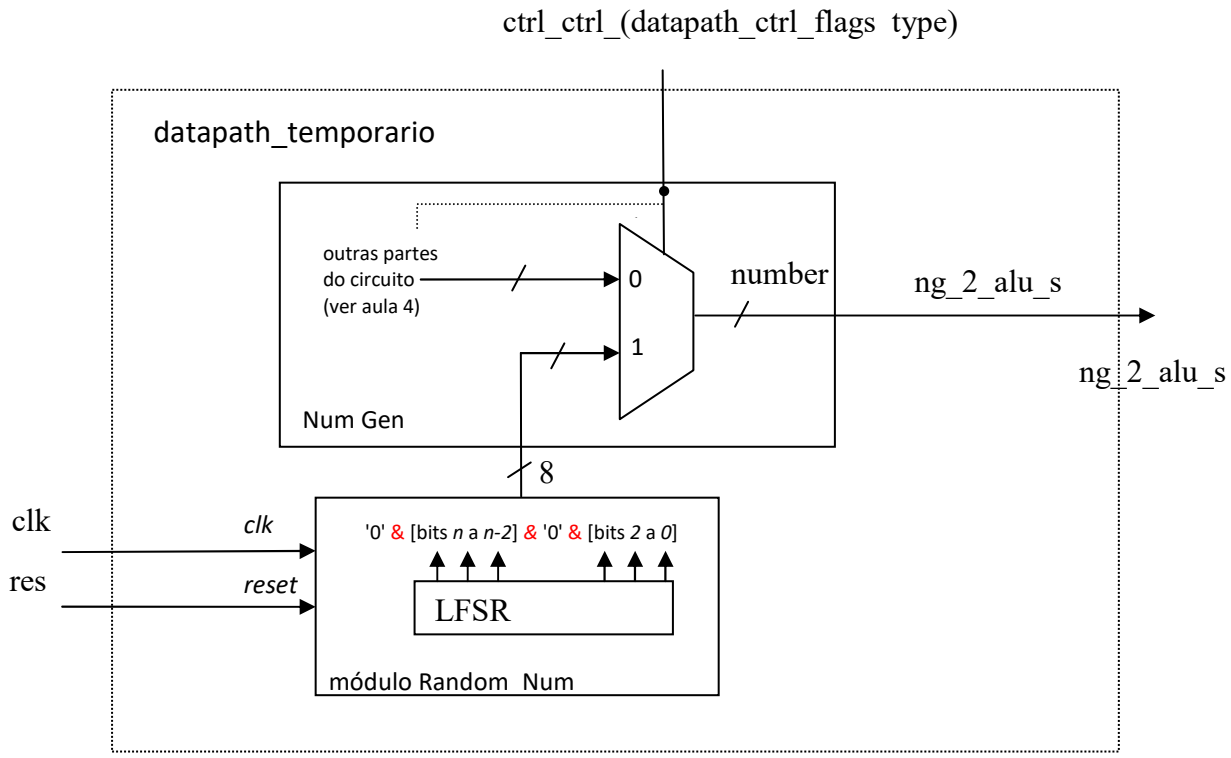

Figura 2. datapath_temporario - topo simplificado "emulando o datapath do snake

2.3. Fsm_Init

O módulo Fsm_Init, cuja representação em forma de abstract state machine, ASM, é mostrada na Figura 3. Pela ASM, os retângulos indicam o estado e as atribuições de saída e os hexágonos correspondem às decisões que associadas aos estados. corresponde à máquina de estado de iniciação do jogo.

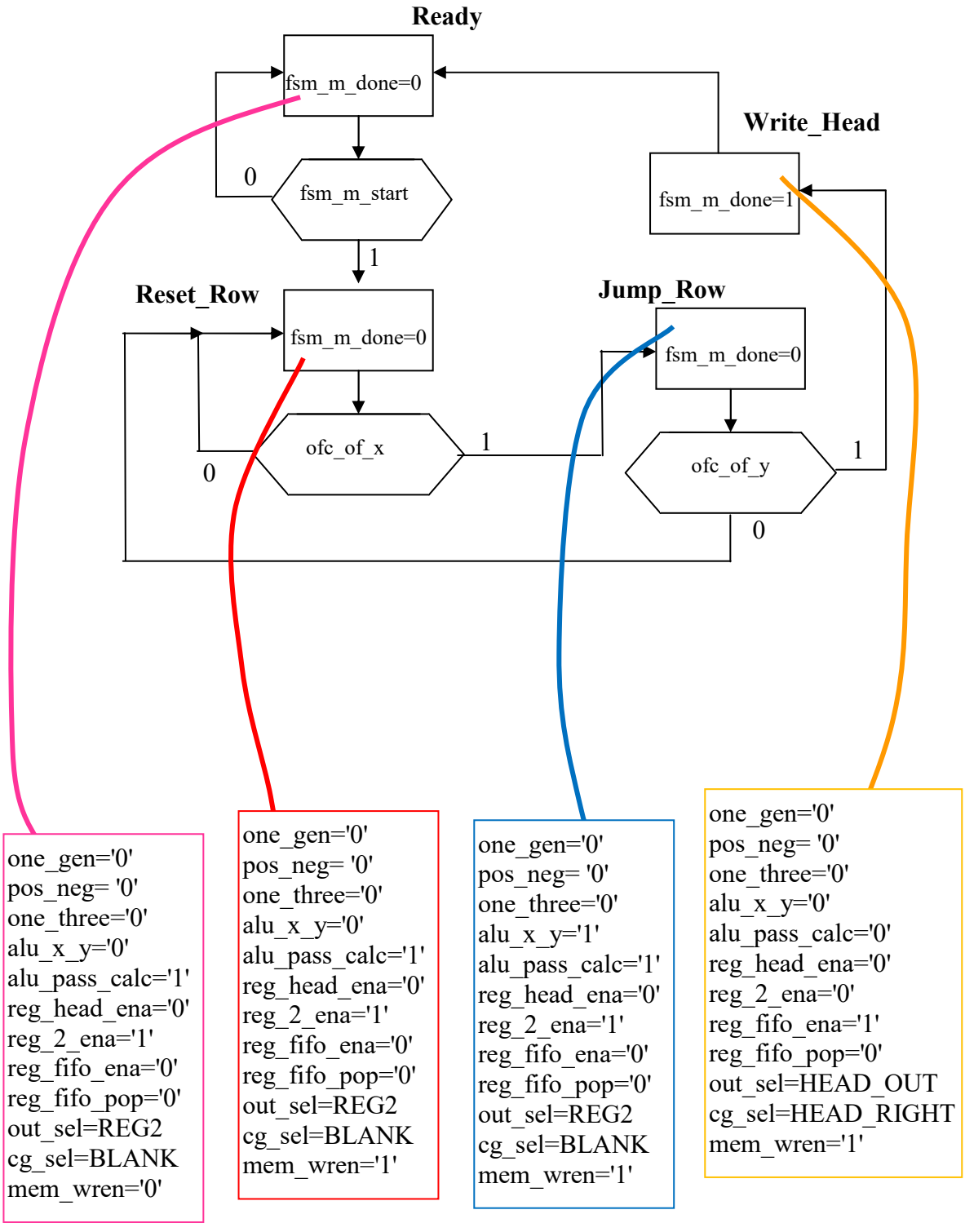

Figura 3. Diagrama da ASM (Fsm_Init)

Como a lista de atribuições dos flags de saída não cabem nos retângulos, elas estã descritas nas caixas de texto associadas.

A Fsm_Init Realiza basicamente duas ações em sequência:

- escrita do valor BLANK em todas as posições de memória, de coluna a coluna e, ao final de uma fila, saltando para a próxima;

- escrita da cabeça, caminhando para a direita (HEAD_RIGHT), na posição 0 da memória.

Deve-se observar que:

- os sinais externos, *ofc_of_x* e *ofc_of_y*, indicam a finalização da escrita em uma linha e de um quadro da memória, respectivamente;
- os sinais *reg_head_ena*, *reg_2_ena*, *reg_fifo_ena*, *reg_fifo_pop*, agem diretamente sobre o módulo *reg_bank*, módulo visto na Aula 8. Adicionalmente, o sinal *out_sel* determina qual a saída selecionada no *reg_bank*.

Testbench:

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.