Testbench para o Snake_Hw

Para comprovar o funcionamento da entity topo Snake_hw, um testbench é fornecido para os alunos. No **Windows Explorer**, na pasta **Rede=> NEWSERVERLAB => psi3451 => Aula_13_projeto_2_3**, uma pasta Teste poderá ser encontrado contendo o arquivo topo do testbench junto com os seus submódulos (em arquivos correspondentes). A estrutura do testbench topo, *snake_testbench* é composto de 3 submódulos: snake_hw, *snake_stimuli* e *map_monitor*. O primeiro deve ser instanciado pelo aluno com o seu projeto; o segundo emula a ação do jogador pressionando os botões; por fim, o último é uma forma de verificar como os valores da memória devem ser varridas e passadas ao video (útil para descobrir o mau funcionamento do hardware, porém não serve para depuração)

Em relação ao testbench, o(a) aluno(a) deverá observar com atenção as ações definidas em snake_testbench, pois estas definirão a sequência de passos do jogo e toda a interação que o "jogador" deverá ter com o jogo. É oportuno relembrar que caso o "jogador" não pressione nenhum botão, o jogo deveria prosseguir independentemente disto, avançando no sentido definido pelo último avanço da cobra.

Os módulos de teste são os seguintes (acompanhe com a Figura 1):

a) Snake_testbench – é o arquivo topo instanciando os três módulos já mencionados acima. A figura indica as conexões existentes.

b) Snake_stimuli – módulo onde os valores de entrada são gerados no tempo. O *sys_direction* (de 4 bits) é o único sinal gerado, indicando o botão que está sendo acionado pelo usuário (será transferido para o módulo *button_handler* dentro do *snake_hw*. Pode conter os valores (0001- right; 0010- left; 0100-up; 1000-down; 0000- botão sem aperto).

c) Map_monitor – Antes, é importante entender a funcionalidade do módulo controlador de vídeo. O controlador de VGA, a ser instalado junto ao módulo snake_hw para operar dentro de um FPGA. O módulo, de tempos em tempo, solicitará ao snake_hardware, através do porto de endereço b de memória, todos os valores do tabuleiro (para apresentação no monitor de vídeo). O controlador de vídeo faz a solicitação de leitura da memória sempre que o sinal no seu porto print_rdy estiver ativo. Este sinal significa que a FSM do snake_hw está no estado IDLE, logo após algum STEP (em que alguma ação foi tomada).

O submódulo address_counter encarrega-se de realizar a contagem de 0 a 63 (tamanho do tabuleiro) e ativa o sinal count_done sempre que a contagem máxima é alcançada, o que permite ao Map_monitor terminar, por aquele passo, a requisição de dados da memória (pelo porto b).

Com as considerações anteriores, o módulo contendo Map_monitor contém dois objetivos:

- permitir a avaliação do snake_hardware como um emulador do controlador de VGA entregando os dados armazenados da memória.

- acompanhar a evolução do jogo, uma vez, que pode-se ter o mapa do tabuleiro a cada passo do jogo- para isto, o tabuleiro é desenhado em um arquivo *.txt, como indicado na Figura 1. Da mesma forma, com o porto game_over ativado, o Map_monitor saberá que ocorreu o fim de jogo, e imprimirá este resultado no arquivo txt.

O testbench pode ser utilizado nas duas formas já vistas durante o decorrer da disciplina:

1) Diretamente no Modelsim, na forma vista nas aulas iniciais sobre testbench (a entity topo é o *snake_testbench*);

2) Pelo Quartus, na forma vista na aula 8 (a entity do projeto do Quartus é o *snake_hw*)- se o(a) aluno(a) seguiu a recomendação de compilação pelo Quartus, bastará adicionar o testbench;

Para visualização no Wave, o(a) aluno(a) dever utilizar o script dado, de nome run_sim.txt. Este script pode ser usado dentro das duas formas acima:

1) Diretamente no Modelsim: siga as instruções dadas nas aulas 2 e 3 através do comando "do run_sim_1.do"

2) Pelo Quartus: ao se ajustar os arquivos de testbench, selecione também a opção de se usar script para a simulação e selecione o arquivo run_sim_1.do.

ATENÇÃO: Fica por conta do(a) aluno(a) ajustar a simulação de forma a mostrar que o jogo possa evolui corretamente em diversas condições.

OBSERVAÇÕES. Há alguns aspectos que o(a) aluno(a) deverá controlar no seu testbench:

a) o arquivo de script do Wave, run_sim_1.do, não tem correlação com os arquivos VHDL do *testbench*, podendo ser posicionado em qualquer pasta; é necessário, no Quartus, durante o ajuste dos testbenches, apontar para a posição correta.

b) observe que o esquema de escrita no arquivo *.txt é implementado na entidade *map_monitor*, porém o nome do arquivo é dado no campo generics, portanto pode ser definido na entidade topo, *snake_testbench*.

c) o arquivo snake_stimuli fornecido é apenas uma amostra. De acordo com a sua particular geração de números aleatórios (na verdade, pseudoaleatório, portanto a simulação ´"determinística" e repetitiva), deve-se alterar a sequência de eventos de entrada para observar (pelo *.txt) a evolução do jogo nas suas várias possibilidades.

d) Da mesma forma, use os sinais da janela do Wave para observar a evolução do jogo, e, em particular, a geração dos números aleatórios.

d) Para os que pretendem olhar o jogo funcionando na placa/monitor de vídeo, antes de programar o circuito no FPGA, é recomendável realizar a simulação em gate-level com timing. Em gate level, é necessário modificar os nomes de algumas entradas como feito na Aula 10, uma vez que alguns tipos enumerados serão transformados em bits.

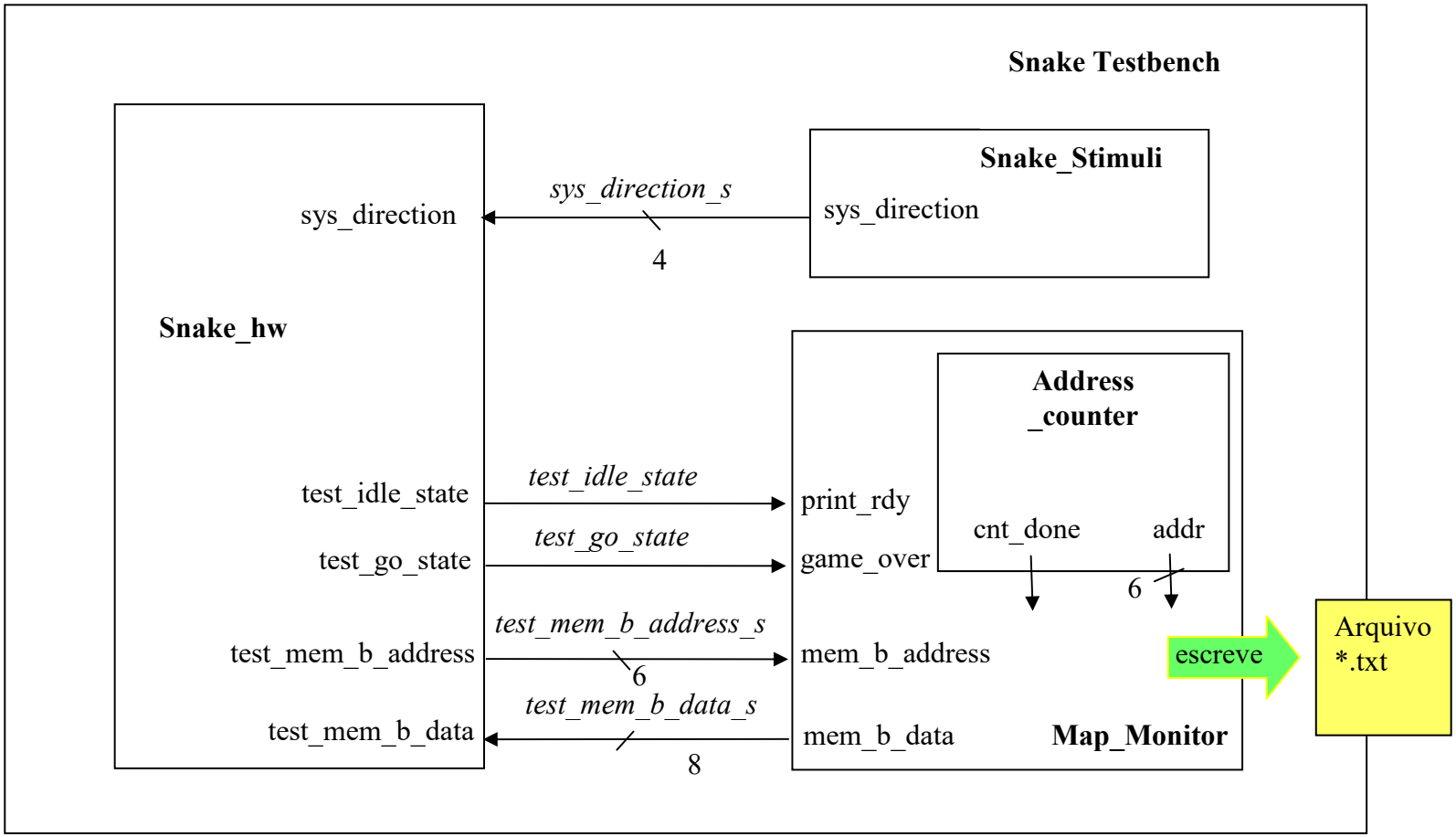

Figura 1. Esquema do módulo topo do testbench