FPGA - Bulls And Cows

SISTEMAS DIGITAIS - 98G02-04
Professor: Anderson Domingues
Turma: 10
Bernardo Fraga, João Victor Terra, Pedro Oliveira, Raul Costa.

Objetivo

O objetivo desse projeto é implementar o jogo *"Bulls And Cows"* em uma placa de desenvolvimento **FPGA Nexys A7 -** utilizando a linguagem de descrição de hardware Verilog -, a qual tem capacidade de receber entradas de até 16 bits - incluindo botões e switches - e saídas de led e displays de 7 segmentos, o que possibilita a interação com o jogo e a visualização dos resultados.

Sobre o Jogo

O jogo "Bulls and Cows" é um desafio de lógica no qual um jogador tenta adivinhar um código secreto gerado pelo oponente.

As rodadas seguem da seguinte maneira: Cada jogador define um código secreto de 4 dígitos binários, sem repetições. Os jogadores então se revezam tentando adivinhar o código do oponente. A cada tentativa, o sistema informa a quantidade de "bulls" (dígitos corretos nas posições corretas) e "cows" (dígitos corretos em posições erradas). O jogo termina quando um dos jogadores acerta completamente o código adversário.

Metodologia

1. **Entradas.**

Para receber a entrada do usuário, utilizamos os 16 switches disponíveis na placa FPGA Nexys A7. Esses switches funcionam como um vetor de bits o qual irá representar os dados em binários inseridos pelo jogador. Cada tentativa ou definição de segredo será composta por 4 dígitos decimais, formado por 4 bits cada, totalizando assim, os 16 bits do vetor de switches. Os dígitos, de 0 à 9, irão passar por uma validação posterior para garantir que cada dígito esteja no intervalo de 0 à 9 e que sejam distintos entre si.

2. **Máquina de Estados.**

Para a implementação do jogo, desenvolvemos uma Máquina de Estados Finitos (FSM) responsável por controlar o fluxo da partida. A FSM é composta por seis estados principais, conforme descrito a seguir:

➢ **S1** (Segredo 1): Exibe "P1 SU" (Player 1 Set Up) no display. Recebe um código em binário de 4 dígitos, se houver dígitos iguais no código, ele retorna para **S1,** se não, mostra a mensagem e passa para o estado **S2**.

➢ **S2** (Segredo 2): Exibe "P2 SU" (Player 2 Set Up) no display. Recebe um código em binário de 4 dígitos, se houver dígitos iguais no código, ele retorna para **S2**, se não, mostra a mensagem e passa para o estado **T1**.

➢ **T1** e **T2** (Tentativas): Exibe "P1/P2 GUESS" (Player 1/2 Guess). Recebe um código em binário de 4 dígitos, se houver números iguais no código, retorna para a tentativa atual (T1 ou T2), se não, chama a função *calc_bulls_cows,* que analisa se existem dígitos corretos e/ou nas devidas posições e guarda na memória o número de "touros e vacas" - em caso de acerto completo, passa para o estado **WIN** -. Em seguida, passa para o estado **RESULT**.

➢ **RESULT** (Resultado): Exibe "xB xC" (x Bulls and x Cows) no display. Usado exclusivamente para mostrar nos displays a quantidade de touros e vacas, calculado pela função *calc_bulls_cows* e decidir qual será o próximo jogador a realizar uma nova tentativa (T1 ou T2), utilizando um simples sistema de *flags* para saber qual a próxima tentativa.

➢ **WIN** (Vitória): Estado final do jogo, quando um dos jogadores acerta completamente o segredo do adversário, esse estado mostra a mensagem "b E" no display (bullsEye). Quando o usuário aperta o botão, o jogo volta para o estado **S1**, passando assim para a próxima rodada.

3. Módulos.

Sobre os módulos do projeto, foram desenvolvidos diversos arquivos em Verilog, cada um com uma função específica na implementação do sistema:

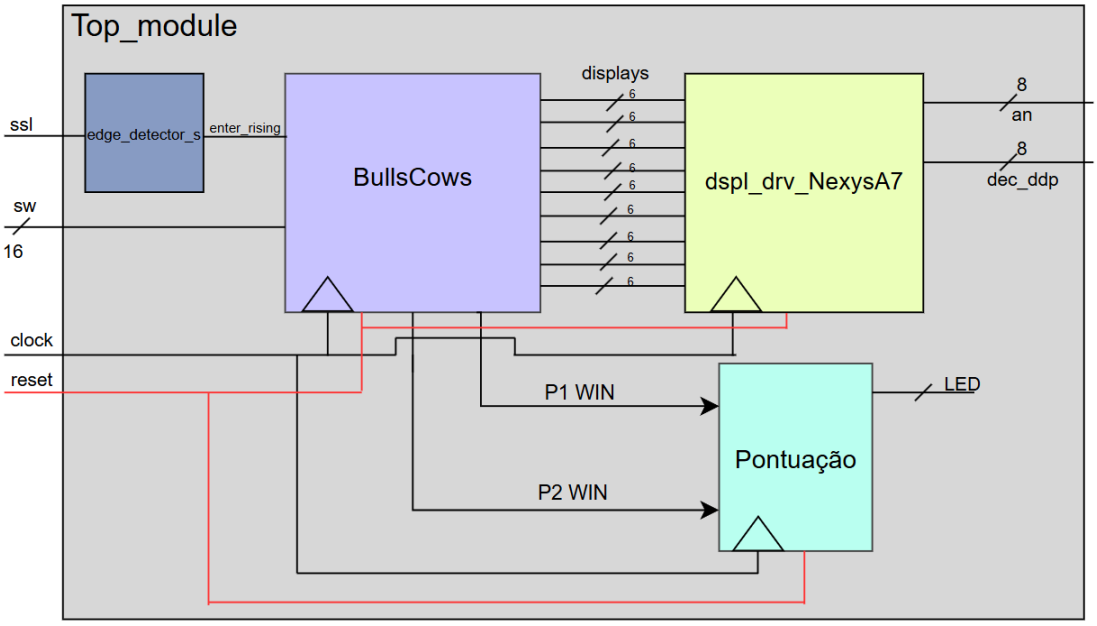
➢ **Top_module.sv**: Responsável por integrar todos os componentes do sistema, conectando a FSM, módulos de controle dos displays, lógica de pontuação e as interfaces de entrada/saída da placa.

➢ **BullsCows.sv**: Implementa a lógica principal do jogo. Contém a Máquina de Estados Finitos (FSM) com as transições de estado, verificações de validade de entrada, controle do fluxo do jogo e cálculo de "bulls" e "cows".

➢ **pontuacao.sv**: Usada para calcular a pontuação de cada jogador na partida. É responsável por controlar os LEDs abaixo dos displays, registrando a vitória de cada jogador até chegar em 7 vitórias..

➢ **edge_detector_s.sv**: Implementa a detecção de borda de subida para os botões, garantindo que uma única entrada seja registrada por clique e evitando leituras múltiplas indesejadas.

➢ **dspl_drv_NexysA7.v**: Módulo de driver dos displays de 7 segmentos da placa Nexys A7. É responsável por traduzir os dados de saída (como contagem de bulls/cows ou mensagens como "P1 S") para o formato visual dos displays.



4. **Interface com a FPGA.**

Para a interface com a FPGA Nexys A7, foi utilizado o arquivo de restrições "*Nexys-A7-100T-TP2.xdc"* no qual foram definidos os pinos para todos os periféricos utilizados no projeto. A interação do usuário com o sistema ocorre da seguinte forma:

➢ **Switches:** Utilizados para inserir o código secreto ou tentativa de adivinhação. Cada grupo de 4 bits representa um dígito decimal, totalizando os 4 dígitos necessários (16 bits no total).

➢ **Botões**: Utilizados para confirmar a entrada dos switches e avançar no fluxo do jogo. O *edge_detector_s* garante que apenas uma confirmação seja considerada por clique.

➢ **LEDs**: Utilizados para indicar a pontuação de cada jogador na partida atual.

➢ **Displays de 7 segmentos**: Responsáveis por exibir a quantidade de "*bulls"* e *"cows"* e mensagens como "P1 SU", "P2 GUESS" ou "b E" ao longo da execução.

Essa configuração permitiu uma integração completa com os recursos da FPGA e proporcionou a interatividade desejada com o usuário.

5. **Testagem e Conclusão.**

Na primeira testagem, alguns displays mostraram um "U" inesperado. Após revisar o código percebemos que a lógica de envio do controlador do display estava errada, mostrando apenas a letra padrão do display "U". Estavamos enviando um código em binário invertido em 6 bits porém o controlador do display recebe 6 bits sendo que o bit 0 é o que informa se o display está ligado ou desligado, isso é, a informação do dígito fica apenas nos bits 1 ao 5 ([4:1]).

Outro problema encontrado foi com os LEDs. Quanto um jogador ganhava o jogo, o estado WIN enviava um comando para acender um LED a mais, porém, como não existia um sistema de borda para detectar se um LED já tinha sido adicionado, ele adicionava LEDs até o limite da placa, aparecendo assim, todos LEDs acesos na primeira vitória do jogo, como não deveria acontecer.

Na segunda testagem, encontramos pequenos erros de lógica os quais permitiam ao usuário enviar um código *GUESS* com números iguais e passar para o próximo estado, diferente do que deveria acontecer (não aceitar o código e permanecer no estado atual).

Após a correção desses erros, fizemos um teste do jogo na FPGA, mostrado a seguir.

Display:

P1 SU

Primeira ação:

1001 0100 0000 0001 (9401)
ENTER

Display:

P2 SU

Segunda ação:

1000 0101 0001 0000 (8510)
ENTER

Display:

P1 GUESS

Terceira ação:

1001 0101 0000 0001 (9501)
ENTER

Display:

1B 2C

Quarta ação:

ENTER

Display:

P2 GUESS

Quinta ação:

1001 0101 0000 0001 (9501)
ENTER

Display:

3B 0C

Sexta ação:

ENTER

Display:

P1 GUESS

Sétima ação:

1000 0101 0001 0000 (8510)
ENTER

Display:

b E

Os resultados obtidos foram condizentes com o comportamento esperado, validando tanto a lógica implementada quanto a interação entre os módulos do sistema. Assim, encerramos o desenvolvimento e a testagem do projeto, garantindo a correta execução do jogo "Bulls and Cows" de forma interativa na FPGA.

Repositório:

https://github.com/VictorTerra1111/Trabalho-2-de-SD

Referências

1. DIGILENT. *Nexys A7 Reference Manual*. Disponível em: https://digilent.com/reference/programmable-logic/nexys-a7/reference-manual.

2. DOULOS. *Using SystemVerilog for FPGA Design*. Disponível em: https://www.doulos.com/knowhow/systemverilog/using-systemverilog-for-fpga-design/.

3. GITHUB. GitHub. Disponível em: https://github.com/.