Batalha Naval Implemetada em VHDL - ALTERA ED2

Adauto Ferreira de Souza Neto
FACOM - Faculdade de Computação
UFMS - Universidade Federal de Mato Grosso do Sul
Campo Grande - MS, Brasil
adauto.ec@gmail.com

Victor Kazuo Saito
FACOM - Faculdade de Computação
UFMS - Universidade Federal de Mato Grosso do Sul
Campo Grande - MS, Brasil
saitovictor36@gmail.com

***Resumo*—**

***Keywords*-Batalha Naval; VHDL; ALTERA ED0; Cyclone III; EP3C16F484C6;**

I. Introdução

Batalha naval é um jogo de tabuleiro de dois jogadores, no qual os jogadores têm de adivinhar em que quadrados estão os navios do oponente. Seu objectivo é derrubar os barcos do oponente adversário, ganha quem derrubar todos os navios adversários primeiro.

O jogo original é jogado em duas grelhas para cada jogador onde uma representa a disposição dos barcos do jogador e outra representa a do oponente. As grelhas são tipicamente quadradas, estando identificadas na horizontal por números e na vertical por letras. Em cada grelha o jogador coloca os seus navios e regista os tiros do oponente.

Antes do início do jogo, cada jogador coloca os seus navios nos quadros, alinhados horizontalmente ou verticalmente, como mostrado na figura 1. O número e o tamanho dos navios permitidos é igual para ambos jogadores eles não podem se sobrepor.

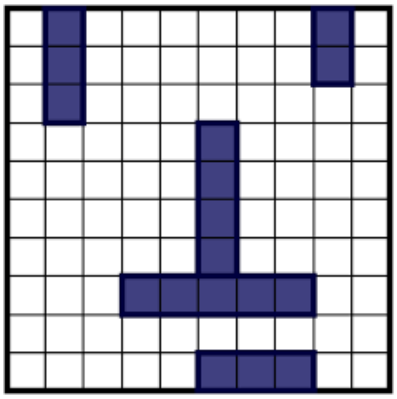


Figura 1. Grelha inicial com os navios já dispostos

Após os navios terem sido posicionados o jogo continua numa série de turnos. Em cada turno, um jogador diz um quadrado, o qual é identificado pela letra e número, na grelha do oponente, se houver um navio nesse quadrado, é colocada uma marca vermelha, senão houver é colocada um X. A figura 2 exemplifica uma grelha após as marcações feitas nas rodadas e também a disposições dos navios.

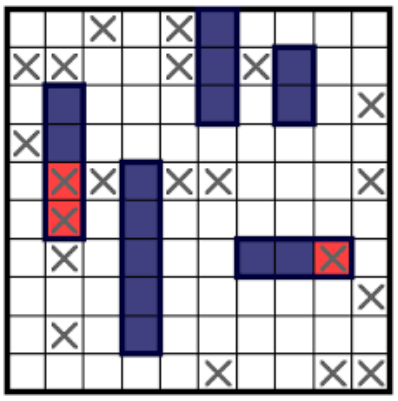


Figura 2. Grelha após o termino do jogo

A proposta do trabalho é implementar um código, em VHDL, que simulasse um jogo de batalha naval com tabuleiros 4x13 já definido previamente no código. O objetivo do jogo implementado é acertar mais embarcações que seu adversário e ao acertar todas os barcos o jogo termina. O código foi feito para ser usado no kit ALTERA DE2 Cyclone II EP2C35F672C6N.

II. Projeto e Implementação

Para este projeto usamos: um contador, um conversor de clock de 50 MHz para 1 Hz, memórias ROM, um decodificador para display de sete segmentos, um comparador.

O decodificador transforma vetores de 4 bits em vetores de 6 bits e é utilizado para inserir valores de 4 bits no display de 6 bits.

O conversor converte um clock de 50 MHz para 1 Hz, para que a contagem de tempo de cada jogador seja feita em segundos.

O contador decrementa o tempo total que cada jogador tem para realizar um movimento, o tempo de cada rodada é baseado em uma das 4 dificuldade escolhida ao iniciar o jogo. Para escolher a dificuldade, que será exibida em um display, o jogador utiliza 2 switchs.

O comparador analisa a jogada e a compara com o local na memória onde estão os navios, avisando ao jogador se a jogada teve algum efeito ou não. Em caso positivo um dos leds da ED2 se acendem em verde para o jogador 1 e vermelho para o 2.

A jogada é lida através de 16 switchs e um botão e mostrada, simultaneamente, no display. Assim que o botão

de disparo for pressionado a vez é cedida para o outro jogador.

A. Modelo

Para implementar nosso temporizador mostrado na figura 3, usando VHDL comportamental, criamos uma máquina de estados finitos, idêntica a mostrada na figura 4 que controla o funcionamento do resto do circuito através de enables e resets.

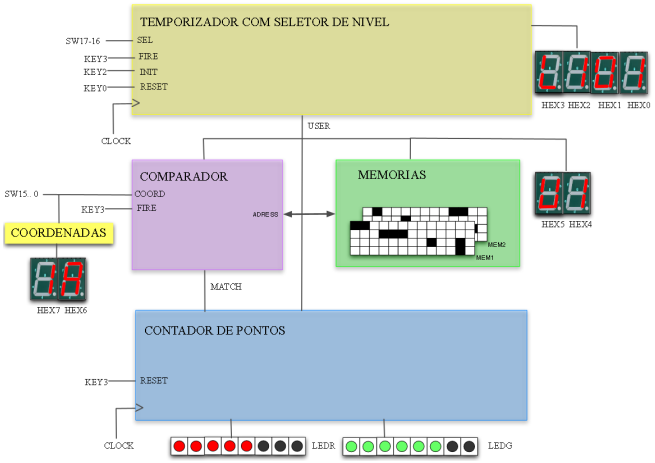


Figura 3. Diagrama de Blocos da Batalha Naval

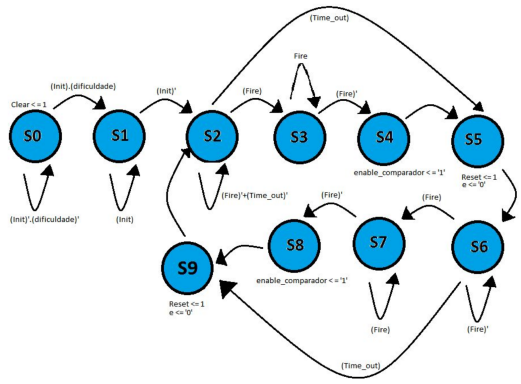


Figura 4. Maquina de Estados Finita do Projeto

O bloco Memória da figura 3 representa uma memórias onde foi colocado a matrize de 0s e 1s com os alvos.

O Temporizador com seletor de nível da figura 3 é um circuito sequencial encarregado de gerar uma contagem decrescente em segundos de dois dígitos em decimal. O tempo associado a cada rodada é definido por dois switches SW17 e SW16, permitindo 4 possíveis tempos de jogo. Além disso, o temporizador possui um reset assíncrono e botão de início de turno nos botões KEY0 e KEY3, respectivamente. Sempre que o tempo de jogo finaliza, ocorre um reset, ou é realizado um disparo com o botão KEY3, então o usuário é alternado, essa informação é dada pelo bit ”USER”.

O comparador da figura 3 representa o circuito combinacional encarregado de gerar um bit, MATCH, que determina se o usuário adivinhou o alvo do inimigo usando do switch 0 ao 15, onde MATCH assume valor de nível alto se o usuário acerta o valor, e baixo caso contrário.

O contador de pontos da figura 3 é um circuito sequencial encarregado de acumular os alvos atingidos pelos usuários (número de vezes que MATCH assume nível alto).

B. Maquina de Estados Finita

Nessa seção é apresentada a FSM montada que já foi mostrada anteriormente na figura 4.

1) S0: Estado inicial da FSM onde todas as variáveis são zeradas e então espera pela dificuldade ser escolhida e o botão INIT apertado.

2) S1: Grava a dificuldade escolhida impedindoa de ser alterada no decorrer do jogo, inicia o processo de reset do cronômetro e espera o usuário soltar o botão INIT.

3) S2: Termina de resetar o cronômetro, determina o jogador atual como Jogador 1 e passa as informações necessárias para os displays de 7 segmentos (Endereço de memória atual (Switches 15 e 14), linha de tiro (Switches 13 a 0) e tempo restante). Espera o fim do turno do jogador 1, que acaba caso ele aperte FIRE (indo para o estado S3, que contabiliza o tiro) ou caso o tempo restante chegue a 0 (indo para o estado S5 fazendo o jogador 1 “perder a vez”).

4) S3: Espera o jogador soltar o botão FIRE.

5) S4: Ativa o enable do comparador e do contador de pontos.

6) S5: Reseta as variáveis referentes ao turno do jogador 1, inicia o reset do cronômetro e desativa o enable do comparador e do contador de pontos.

7) S6: Termina de resetar o cronômetro, determina o jogador atual como Jogador 2 e passa as informações necessárias para os displays de 7 segmentos (Endereço de memória atual (Switches 15 e 14), linha de tiro (Switches 13 a 0) e tempo restante). Espera o fim do turno do jogador 2, que acaba caso ele aperte FIRE (indo para o estado S3, que contabiliza o tiro) ou caso o tempo restante chegue a 0 (indo para o estado S9 fazendo o jogador 2 “perder a vez”).

8) S7: Espera o jogador soltar o botão FIRE.

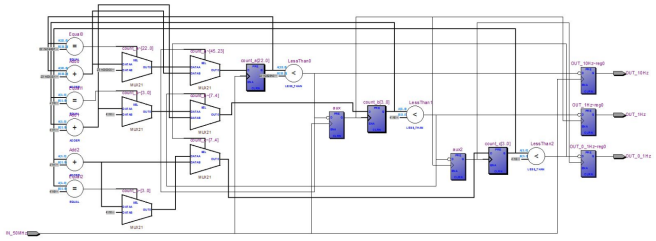
9) S8: Ativa o enable do comparador e do contador de pontos.

10) S9: Reseta as variáveis referentes ao turno do jogador 2, inicia o reset do cronômetro e desativa o enable do comparador e do contador de pontos.

C. Cronômetro

A ideia para esse cronômetro, como foi dito anteriormente, é poder informar quantos segundos o usuário possui para fazer a sua jogada.

O clock de 50 MHz , o qual foi usado como base do projeto, é muito rápido e, por si só, não seria possível indicar corretamente o tempo restante. Sendo assim, o clock de 50 MHz é ligado num conversor de clock (mostrado na figura 5) realiza uma contagem até 25 mil, incrementando o valor atual em 1 a cada ciclo de clock, e então inverte o clock de 1 Hz, recomeçando a contagem.



Esse novo clock de 1 Hz será utilizado como clock para o cronômetro que mais tarde será codificado para o display de sete segmentos. Como podemos perceber, esse método nos permite criar um clock de 1 Hz mesmo dispondo de apenas um clock de 50 MHz e outras unidades lógicas. O sinal de reset assíncrono do cronômetro nos permite recomeçar a contagem, independente do estado que estivermos. Esse processo esta representado na simulação da figura 6

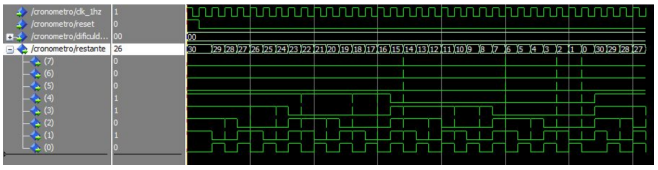


Figura 6. Simulação do Cronometro

D. Comparador

O objetivo do comparador é gerar um bit de saída para o contador de pontos chamado MATCH, que será posto em nível lógico alto caso a o tiro do jogador acerte uma posição de navio determinado na memória. Para isso, usamos um misto de VHDL comportamental e estrutural, comportamental para manter a sincronia do circuito e estrutural para ter um circuito mais simples.

Durante a partida, os usuários escolherão uma coordenada fazendo o uso das chaves para isso. Após a escolha ter sido feita, o comparador verificará se o "input" está correto, fazendo uma comparação com sua memória ROM e alterando o bit MATCH de acordo. Esse processo esta representado na simulação da figura 7.

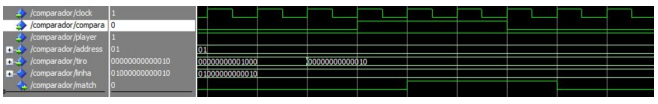


Figura 7. Simulação do Comparador de Jogadas com ROM

E. Memória

Esta memória tem como entradas: um std logic chamado player, que assume '0' para o jogador 1 e '1' para o jogador 2 e um std logic vector de 2 bits chamado address que define a linha desejada da matriz. Como única saida, contém um std logic vector de 14 bits que possui o conteúdo da linha determinada pelas entradas. O circuito desta memoria é apresentado em 8

As coordenas dos navios de cada jogador são escolhidas pelos usuários antes do início da partida. Os jogadores devem alterar o código nas matrizes, colocando 1's para os locais onde se encontram uma das coordenadas do navio e 0's nas posições com água.

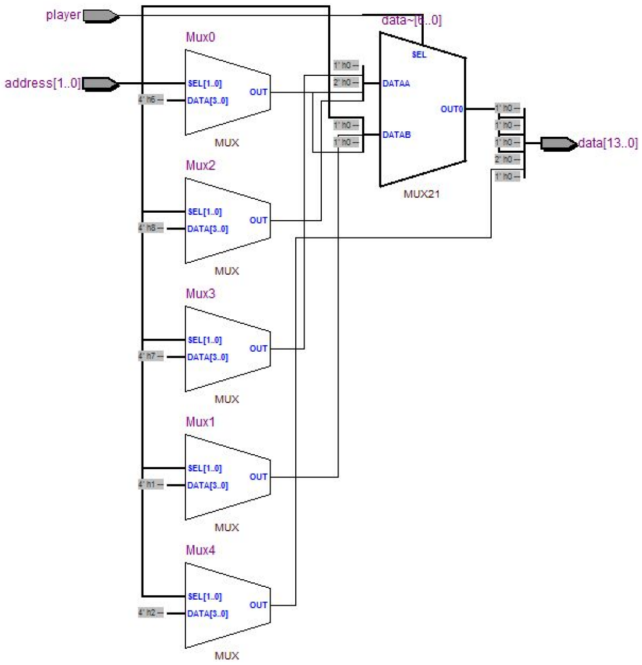


Figura 8. Circuito da ROM

F. Coordenadas

O display de sete segmentos da placa possui grande importância neste projeto. Para que pudéssemos usá-lo corretamente, fizemos um decodificador que recebe um vetor de 4 bits do sistema hexadecimal e o converte para um vetor de 7 bits que indica quais segmentos devem ser ligados ou desligados para que o número desejado apareça no display.

O conversor para sete segmentos foi feito utilizando VHDL comportamental.

G. Contador de Pontos

Para a indicar a contagem de pontos de cada jogador, foram usados 8 LEDs verdes e mais 8 LEDs vermelhos. Sempre que um jogador consegue acertar uma coordenada do navio adversário um de seus LEDs são acesos.

Para armazenar a pontuação de cada jogador, utilizando VHDL comportamental, criamos um contador de pontos que possui dois vetores de 8 bits (um para LEDR e outro para LEDG), quando o enable do comparador estiver ativo e o MATCH estiver em '1', baseado no bit que indica o jogador, um dos vetores sofrerá um Left Logical Shift, e sua posição 0 será completada com um '1'.

Outro fato que deve ser considerado é que se um jogador atirar em uma posição em que havia atirado previamente, onde havia um navio, o jogo pontuará para o usuário novamente. Isso acontece porque nossa memória suporta apenas leitura, e não escrita. Para deixar o jogo mais

real, necessitaríamos de uma memória RAM, que é mais complexa.

Simulação do contador de pontos, mostrando que um ponto só será contabilizado se o enable (variável "compara") e o match (variável "cont match") estiverem ativos ao mesmo tempo. A simulação da etapa pode ser vista na figura 9.

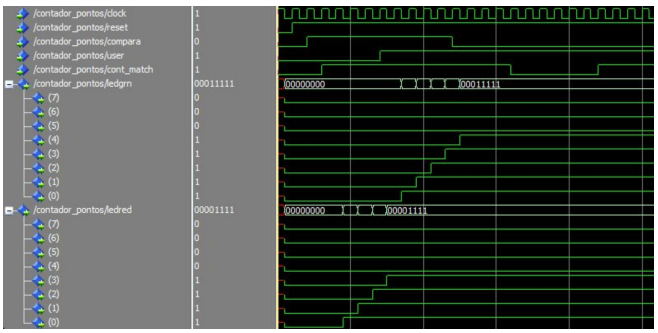


Figura 9. Simulação do Contador de Pontos

III. Resultados

Implementou-se um circuito na placa de desenvolvimento DE2 fazendo uso das estruturas e conhecimentos obtidos durante o curso. O circuito conseguiu implementar a funcionalidade de um jogo interativo similar ao batalha naval, entre dois jogadores (usuário 1 e usuário 2).

O comportamento desse jogo conseguiu seguir o modelo apresentado e discutido anteriormente.

IV. Conclusão

A maior dificuldade, sem dúvida, foi ter feito a FSM visto que ela era a parte principal do projeto e apresentar a imagem através de cabo VGA, ideia essa que foi abandona no final do projeto devido a dificuldades na leitura dos botões enquanto se controlava o VGA.

Mas analisando o projeto como um todo, não houve nenhuma unidade que apresentasse problema. Fizemos todas as simulações solicitadas, bem como os testes na placa. Em momento algum tivemos problema nos testes finais e os resultados foram todos dentro do esperado.

A maioria das unidades lógicas construídas foram associadas com a FSM, visto que ela é a base para todo o projeto. O comparador foi usado junto com a memória ROM para que ele possa validar, ou não, a jogada do usuário.

O decodificador recebe informações do usuário e da FSM e as repassa para o display de sete segmentos. O clock de 50 MHz é usado para sincronizar todo o circuito e é também usado para gerar o clock de 1 Hz. Já o clock de 1 Hz é usado no cronômetro para contar o tempo restante do usuário. Quanto ao contador de pontos, ele é sempre atualizado depois que o comparador é utilizado para verificar uma jogada.

Referências

https://riptutorial.com/vhdl/example/32047/rom Materiais da aula