Luiz Eduardo Cartolano - RA: 183012
Yago Barbosa - RA: 188727

Grupo 14 - Jogo da Velha

Trabalho apresentado no curso de MC613 - Turma A
pelos alunos Luiz Eduardo e Yago Barbosa,
emails: l183012@g.unicamp.br e y188727@g.unicamp.br

Orientadores: Prof. Dr. Sandro Rigo
Isaías Felzmann

18 de Junho de 2018

Conteúdo

1 TEORIA

Nesta seção trataremos de alguns conceitos teóricos que foram abordados no trabalho e que exigiram conceitos além do que foi visto em classe.

1.1 Trabalho com diversas entidades

O trabalho com múltiplas entidades, que deviam trabalhar de forma sincronizada, foi, sem sombra de dúvidas, o maior desafio do projeto. Para obter exito na execução da tarefa foi preciso entender melhor o funcionamento do *clock*.

Notamos que a melhor maneira de sincronizar os processos realizados pelo *mouse, vga e programa* era a realização das ações a medida que os os *clocks* do sistema ocorriam.

1.2 Comunicação com dispositivos de entrada e saída

O projeto escolhido, foi o *jogo da velha*, de modo que, parte crucial do projeto era a interação com dispositivos externos a placa, como o *mouse* e o *monitor*. Para isso, foi preciso que aprendessemos a manusear ambas as interações, aprimorando os conhecimentos básicos que haviam sido adquiridos nos laboratórios.

Para o *mouse*, buscamos entender melhor o código fornecido para o laboratório 07 e também o materia disponível em [1]. A partir de ambos foi possível estruturar nossa entidade que nos fornecia, com base no deslocamento nos eixos x *e* y e em um ponto inicial, a posição do cursor na tela.

Enquanto que para o *monitor*, baseamos nosso trabalho no entendimento de como deveríamos fazer para *"desenhar"* neste. Chegamos à conclusão de que para tal, era necessário preencher o conjunto de píxels no formato desejado.

1.3 Modo de jogo automático

Por fim, outro grande desafio do projeto que nos fez desprender uma grande quantidade de estudo foi o desenvolvimento de uma *"inteligência artificial"* para o Jogo da Velha. A fim de buscar a melhor implementação possível, dento das limitações e dificuldades impostas pelo *VHDL*, foi preciso estudar mais a respeito de algoritmos gulosos [2] e sobre os melhores tipos de algoritmos para se ganhar no jogo da velha. Estes serão melhor discutidos na póxima seção.

2 DESCRIÇÃO DO SISTEMA

2.1 Estratégia de Implementação

O projeto foi implementado seguindo algumas regras básicas do jogo da velha [3], além delas, optou-se por algumas modificações, a fim de facilitar a implementação, como a troca dos tradicionais "xis"e "bola"pelo uso das cores vermelha e verde (para demarcar a jogada). E também adotou-se que a cor vermelha sempre inicie o jogo.

Outra estratégia interessante de ser comentada é a respeito da maneira a qual implementamos o projeto, que julgamos ter facilitado bastante o trabalho. No início, concentramos nossos esforços em conseguir realizar a leitura da posição do *mouse*, feito isso, trabalhamos em conseguir mostrá-lo no monitor, vale ressaltar, que até que o jogo estivesse totalmente funcional trabalhamos com uma tela 4:3. Uma vez que conseguíamos observar o cursor do *mouse* na tela, resolvemos trabalhar na validação do clique, ou seja, em como iríamos garantir que nenhum usuário pudesse clicar em áreas não marcáveis (fora dos *grids* ou dos botões implementados). E então, trabalhamos na implementação da tela inicial do jogo, que era algo bem simples, dois botões para escolher o modo de jogo (contra outro usuário ou contra o "computador") e a cor do jogador (vermelha ou verde), além de um para iniciar o jogo. Depois, iniciamos o desenvolvimento da segunda tela, que consistia no funcionamento de ambos os modos de jogo e na determinação do vencedor. Ao fim desses passos, possuíamos um jogo funcional, mas feio e nada intuitivo, logo era hora de expandir de uma tela 4:3 para uma de maiores proporções, na qual seria possível desenhar uma interface gráfica a que foi feita.

Agora faremos uma explicação mais detalhada dos blocos mostrados na Figura 1.

2.2 Bloco Mouse

- **Descrição:** O bloco tem como entrada dados recebidos na entrada PS2 da placa(de I/O), além da posição x e y do mouse advindos de *Atualiza Posição*, com relação a origem adotada no mapeamento de pixels do monitor. E também um sinal do bloco *Valida Clique* informando se o clique recebido foi válido. Sua saída é a posição atual do mouse (para o Monitor), dados iniciais da partida e a jogada do turno (para a UC). Sua função é gerenciar as ações a serem tomadas pelo clique do mouse, além de gerenciar o funcionamento dos blocos *Valida Clique e Atualiza Posição* para coletar e processar os dados do mouse.

- **Validação:** A validação do bloco foi feita através de testes com o mesmo na própria placa, algo que julgamos aceitável uma vez que estávamos trabalhando com algo bem visual.

- **Limitações:** O bloco em questão não possui limitações próprias, uma vez que ele é um gerenciador de outros blocos. Contudo, ele acaba transmitindo erros advindos de blocos por ele gerenciados.

2.3 Bloco Valida Clique

- **Descrição:** O bloco tem como entrada as posições x e y do mouse, recebidas do *Mouse*. E como saída um comando informando qual região da tela foi clicada, ou um comando informando que não houve clique em uma região válida. Sua função é gerenciar as ações a serem tomadas pelo clique do mouse, verificando se a região selecionada pelo usuário poderia sofrer tal ação.

- **Validação:** Novamente julgamos que a maneira mais efetiva de se fazer a validação seria através do uso do bloco na placa.

- **Limitações:** Uma vez que o projeto restringe, adequadamente, o uso do mouse, podemos dizer que ele não possui limitações.

2.4 Bloco Atualiza Posição

- **Descrição:** Sua entrada é um sinal do *Mouse* com os dados recebidos na entrada PS2. Enquanto que sua saída é a posição atual do mouse, nos eixos x e y, com relação a posição (0,0) do sistema. O bloco é bem básico e sua função é receber os dados da entrada PS2 e converte os mesmos em posições (x,y) que serão usadas pelas demais entidades.

- **Validação:** Mais uma vez, assim como os blocos a ele associados, os testes foram feitos usando a placa.

- **Limitações:** Diferente dos outros blocos que se relacionam ao uso do *mouse*, este possui limitações, uma vez que não conseguimos restringir os limites de uso dele com relação aos limites da tela, ou seja, é possível deslocá-lo para além dos limites o que faz com que o cursor se "perca".

2.5 Bloco Monitor

- **Descrição:** Para funcionar precisa da posição do mouse, oriundos do *Mouse*, e situação atual do jogo, informada pela *UC*. Sua saída é a imagem atualizada do jogo (pintura dos pixels) para a porta VGA da placa(I/O). Ou seja, sua função é manter a tela do jogo atualizada no monitor VGA.

- **Validação:** O bloco em questão era extremamente visual, uma vez que ele trabalha exclusivamente com a parte gráfica do jogo. Logo, julgamos que seria mais efetivo fazermos a validação apenas usando a placa.

- **Limitações:** O bloco não possui nenhuma limitação funcional, apenas estéticas. Visto que, a fim de facilitar a implementação do projeto, algumas letras acabarm ficando desproporcionais. Outra opção feita para facilitar a implementação nesse bloco foi a troca dos tradicionais "xis"e "bola"por quadrados nas cores verde e vermelha.

2.6 Bloco IA - Inteligência Artificial

- **Descrição:** A *IA* tem como entrada a situação do grid do jogo e um sinal informando que é a sua vez de jogar. A partir dela, emite a melhor jogada para a situação atual do jogo. Sua função é verificar a melhor jogada possível, de forma a ganhar o jogo sempre que possível ou, no pior dos casos, empatar com o adversário.

- **Validação:** A validação da *IA* apresentou falhas, visto que, não foram feitas simulações ou testadas todas as possibilidades possíveis de situações do grid.

- **Limitações:** O algoritmo usado para a inteligência artificial pode ser visto no Algoritmo 1. Ainda que ele consiga ganhar na grande maioria dos casos, existe um conjunto de jogadas que consegue derrotá-lo. Alguns algoritmos mais robustos foram encontrados, como o *minimax* e podem ser observados em [5], mas uma vez que apresentavam alta complexidade, optamos pelo algoritmo escolhido a fim de facilitar a implementação.

Algoritmo 1: IA - Jogo da Velha

```
   entrada: Grid do Jogo
   saída  : Melhor jogada possível
 1 if pode vencer then
 2 |   ganha o jogo;
 3 end
 4 else
 5 |   if adversário pode vencer then
 6 |   |   bloqueia o adversário;
 7 |   end
 8 |   else
 9 |   |   if centro esta livre then
10 |   |   |   ocupa o centro;
11 |   |   end
12 |   |   else
13 |   |   |   if adversario ocupou um canto then
14 |   |   |   |   ocupa o canto oposto;
15 |   |   |   end
16 |   |   |   else
17 |   |   |   |   if tem um canto livre then
18 |   |   |   |   |   ocupa o canto;
19 |   |   |   |   end
20 |   |   |   |   else
21 |   |   |   |   |   ocupa um espaço livre qualquer;
22 |   |   |   |   end
23 |   |   |   end
24 |   |   end
25 |   end
26 end
```

2.7 Bloco UC - Unidade de Controle

- **Descrição:** O *Mouse* informa a *UC* informações básicas do jogo e os comandos realizados pelo usuário. Enquanto a *IA* informa para a *UC* a jogada que ela deseja realizar. A *UC*, por sua vez, envia para o monitor a situação atual do grid do jogo, que deverá ser mostrada no VGA. Enquanto que para a *IA*, ela informa a situação atual do grid do jogo e o momento de jogar. Logo, podemos concluir que ela é o *core* do projeto, uma vez que tem como função controlar o fluxo de dados do projeto, acionar a atualização da tela (feita pelo *Monitor*), acionar a jogada da IA.

- **Validação:** Mais uma vez, acreditamos que tenha sido outro bloco no qual a validação foi negligenciada, visto que, todo o bloco foi testado apenas na placa.

- **Limitações:** A *UC* funciona perfeitamente dentro do esperado quando testado na placa, ou em situações de jogo, contudo, existe a possibilidade de que alguns problemas tenham passado despercebido devido a falta de simulações mais complexas.

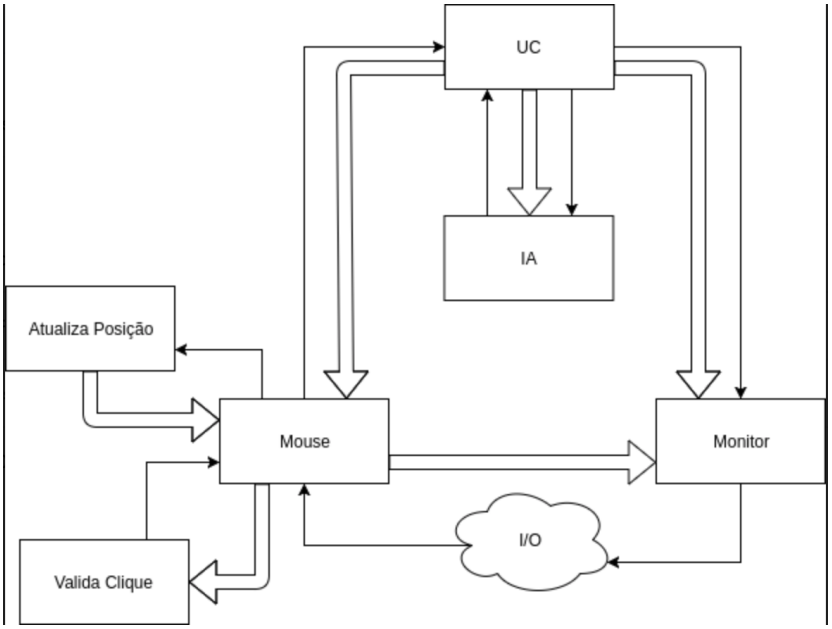

Figura 1: Diagrama de Blocos do jogo da velha implementado na placa DE1

3 CONCLUSÕES

Sem dúvidas a maior conclusão tirada pelo grupo, é a diferença existente ao se realizar o desenvolvimento de aplicações de *software* ou em *hardware*. Abandonar a noção de sequencialidade do código, e descobrir outra maneira de sincronizar as diferentes entidades que compõem o projeto. Além disso, outra grande dificuldade encontrada foi com a sintaxe do código em *VHDL*, que apresenta diferenças significativas em comparação a outras linguagens com as quais estamos mais acostumados.

Além dos problemas já citados, o grupo também encontrou grande dificuldade na validação dos módulos do projeto, diferente do que foi trabalhado nos laboratórios, os módulos eram mais abstratos e complexos, o que complicou a realização de simulações mais complexas para testar o projeto. Sendo este, sem sombra de dúvidas, o ponto de maior falha da nossa implementação.

Apesar dos problemas citados, podemos dizer que houve sucesso na implementação proposta. Visto que, o código entregue é capaz de permitir que qualquer usuário se divirta em uma partida de jogo da velha, seja ela contra outra pessoa, ou contra um modo de jogo computadorizado.

REFERÊNCIAS

[1] Disponível em: http://www.fpga4student.com/2017/12/how-to-interface-mouse-with-FPGA.html, Acesso em: 17-06-2018.

[2] Disponível em: https://www.ic.unicamp.br/ rocha/msc/complex/algoritmosGulosos-Final.pdf, Acesso em: 17-06-2018.

[3] Disponível em: http://www.bigmae.com/regras-jogo-da-velha/, Acesso em: 17-06-2018.

[4] PERTTULA Arto. Disponível em: https://pdfs.semanticscholar.org/presentation/ed96/c9abc1fa3 Acesso em: 17-06-2018.

[5] CANDIAGO Lorenzo. Disponível em: https://www.organicadigital.com/seeds/algoritmo-minimax-introducao-a-inteligencia-artificial/, Acesso em: 17-06-2018.