

**MC613 - Laboratório de Circuitos Digitais**

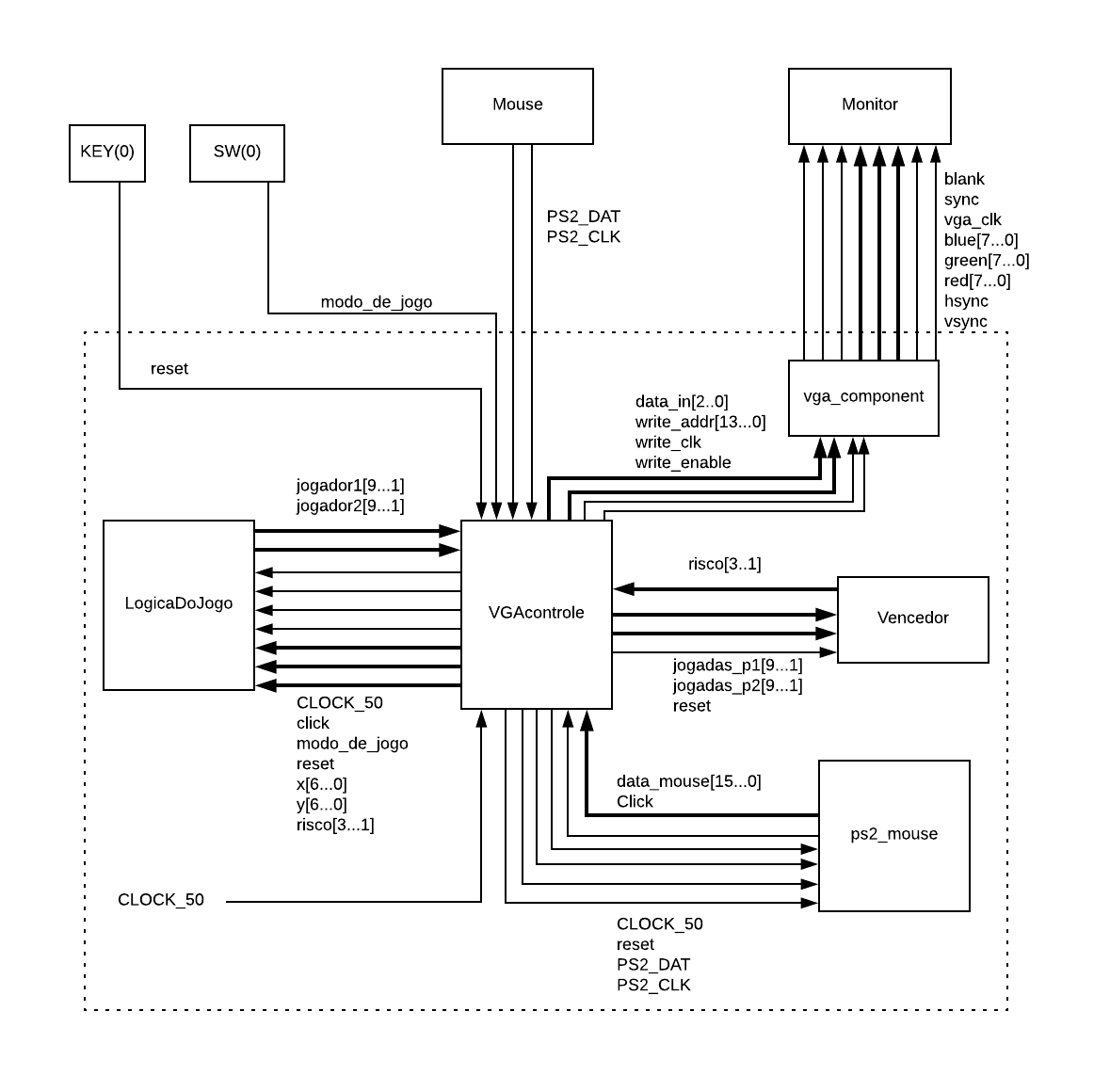
**1º Semestre de 2018**

**JOGO DA VELHA**

**Introdução**

Neste projeto, implementamos duas interfaces, Mouse(PS2) e Monitor (VGA). Nós também nos familiarizamos com projeção de um circuito lógico combinacional. Seguindo os conceitos básicos de um Jogo da Velha, oferecemos aos usuários a capacidade de realizar sua jogada, podendo jogar contra outro usuário ou contra o computador apenas setando o modo de jogo e de redefinir a placa a qualquer momento durante e a partida. No geral, o objetivo deste programa é aperfeiçoar nossas habilidades em circuitos lógicos e codificação VHDL, bem como fornecer aos usuários uma plataforma fácil e limpa de se jogar uma partida clássica de Jogo da Velha.

**Diagrama de Blocos**



**Descrição Funcional**

1. **Mouse**

O Mouse é o dispositivo de entrada para o jogo, em que o jogador mantém o ponteiro do mouse no quadrante desejado informando a posição(PS2\_DAT) e clique(P2S\_CLK) para realizar uma jogada.

1. **ps2\_mouse**

O ps2\_mouse irá receber informações que o Mouse passou para o VGAcontrole (PS2\_DAT, PS2\_CLK). Ele então converte esses dados e devolve um vetor com posição do ponteiro(data\_mouse) e uma variável informando se houve um clique(Click).

1. **LogicaDoJogo**

Esse bloco é responsável pelo funcionamento do jogo. Utilizando uma máquina de estado de Moore (Figura 1), que define os turnos da cada jogada, além de um estado inicial responsável por iniciar/reiniciar o jogo e um estado vitória, que só ocorre caso um dos jogadores vença a partida.

O jogo começa no estado início, que pode ser chamado a qualquer momento caso reset seja ‘0’, reiniciando o jogo. Em seguida, é chamado o estado P1, , responsável pela jogada do primeiro jogador. A logicaDoJogo recebe de VGAcontrole as informações do mouse, sua posição(x[6...0], y[6...0]) e se houve um clique (click). Quando há um clique, é verificado se a posição do clique se refere a algum dos noves quadrantes. Se sim é verificado se a posição está livre, caso afirmativo novamente, uma posição do vetor jogador1[9…1], relacionado ao quadrante selecionado é setado como ‘1’.

Se a modo\_de\_jogo estiver setado como ‘0’, a máquina vai para o estado P2, que funciona de forma análoga ao estado P1, e salva sua jogadas em jogador2[9…1]. No entanto, se modo\_de\_jogo, estiver setado como ‘1’, o próximo estado chamado é o CPU, que a partir de um algoritmo, realiza sua jogada de de forma a sempre buscar sua vitória ou de impedir a vitória do oponente. A CPU realiza sua jogadas com as seguintes prioridades:

1. Se é possível vencer, ela vencerá;
2. Se o adversário estiver em um condição de vitória, ou seja, apenas um jogada para vencer, ela jogará neste quadrante para bloquear;
3. Se o quadrante central estiver disponível, a máquina o escolherá;
4. Caso o jogador tenha jogado em um canto, a máquina jogará no canto oposto;
5. Se um canto estiver disponível, a máquina o ocupará;
6. A máquina jogará em um canto.

O modo CPU também salva suas jogadas em jogador2[9…1]. Após ter realizado qualquer jogadas, a máquina volta para P1, seja qual for o modo de jogo.

Por fim, o estado Vitoria pode ser chamado a qualquer momento, desde que seja detectado que um jogador venceu o jogo. Esse estado apenas impede que sejam feitas novas jogadas. A máquina entra nesse estado caso risco[3...0] seja diferente de “0000”, o que indica que houve um vencedor.

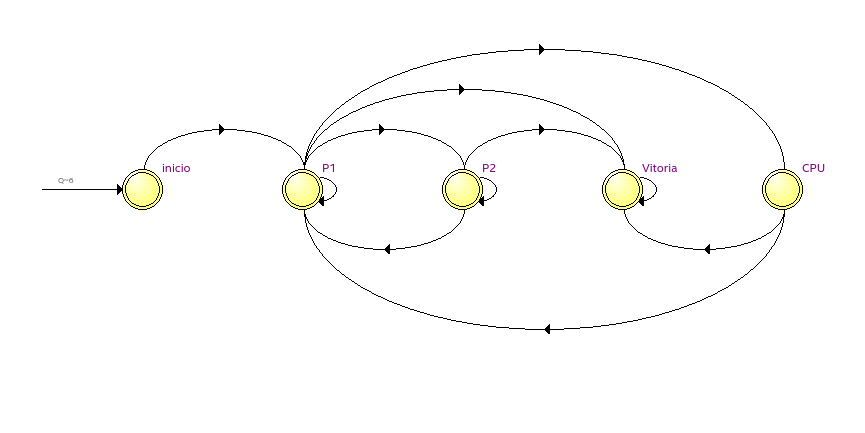
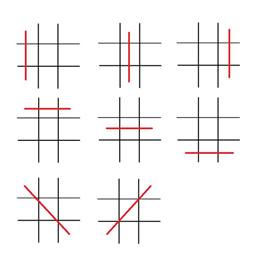


Figura 1:Máquina de estados que controla os turno de cada jogada.

1. **Vencedor**

Este bloco é responsável por verificar se ao término de cada jogada houve um vencedor. Ele recebe dois vetor de 9 posições(jogadas\_p1[9...1], jogadas\_p2[9...1]) que indicam as jogadas de cada jogador. Se ele encontra uma combinação de jogadas que indicam uma vitória, ele passa essa informação ao VGAcontrole em um vetor de 4 posições(risco[3...0]) que indica onde houve uma vitória, sendo possível 8 possibilidades(Figura 2).

Figura 2: Locais onde é pintado na tela um risco caso um jogador vença.

1. **vga\_component**

É o responsável por conectar as informações do VGAcontrole e o Monitor. Ele envia ao Monitor as informações que devem ser pintadas na tela e os parâmetros, como o RGB e as sincronizações horizontais e verticais.

1. **KEY(0)**

Quando é pressionado, a variável reset é setada como ‘0’ e o jogo é reinicializado.

1. **SW(0)**

Define o modo de jogo, quando seu valor é ‘0’, o jogo está no modo jogador vs. jogador. Quando é setado para ‘1’, o jogo entra no modo jogador vs. CPU.

1. **VGAcontrole**

Este é o componente responsável por controlar todo o funcionamento do sistema. Ele receber as entradas KEY(0) (reset), SW(0)(modo\_de\_jogo) e as informações dos mouse (click e data\_mouse). Com essas informações de entrada, é chamado o LogicaDoJogo e o Vencedor, que realizam sua tarefas.

Um processo define as posições em que o tabuleiro, as jogadas, risco da vitória e o cursor do mouse será pintado na tela. Outro processo percorre os endereços de vídeo, quando um endereço contém o valor de uma posição definida, ela é pintado

Para o monitor é utilizando uma resolução de 128x96, que é passado para o vga\_component juntos com o endereços e as cor do que será pintado.

1. **Monitor**

O Monitor será o dispositivo de saída para o jogo. Nele é exibido graficamente o jogo, com o tabuleiro, quais jogadas foram selecionados por cada jogador e quais ainda estão vazios, além do resultado do jogo. Para isso, ele recebe as informações setadas pelo vga\_component.

**Bugs**

Em todo o sistema do jogo, há dois bugs que foram encontrados. No modo jogador vs. jogador há o “problema do clique inválido” que ocorre quando um jogador realizar um clique em alguma parte da tela que não faz parte de nenhum quadrante, isso acarreta na troca de turno, mesmo que nenhuma jogada tenha sido feita.

O segundo problema está na no modo jogador vs. CPU, em que a máquina “não respeita” a jogada anterior, ou seja, ela realiza sua jogada sem considerar a jogada anterior do jogador.

**Performance**

Tabela 1: Informações de compilação

|  |  |
| --- | --- |
| Logic utilization (in ALMs) | 579/32070 (2%) |
| Total registers | 292 |
| Total pins | 39/457 (9%) |
| Total block memory bits | 36864/4065280 (<1%) |
| Total PLLs | 1/6 (17%) |

**Conclusão**

O projeto implementou muitos conceitos vistos durante o curso, como as interfaces VGA e PS2, que são fundamentais para o projeto, para as duas, utilizamos códigos vistos nos laboratórios.

Durante todo o desenvolvimento tivemos algumas dificuldade, principalmente em implementar a parte gráfica e na correção de alguns bugs. No entanto a parte gráfica foi concluída com êxito e apenas dois bugs na lógica não foram possíveis de serem solucionados no prazo estipulado.

Portanto, consideramos o resultado final do projeto positivo, pois cumprimos o objetivo de implementar um Jogo da Velha funcional com um nível de dificuldade que nos proporcionou aprender mais a fundo a linguagem VHDL.

**Referências**

Sistema de análise de sinais: Módulo de interface VGA em FPGA. Disponível em:

<https://wiki.sj.ifsc.edu.br/wiki/images/9/94/TCCCSTST008\_2011.pdf>. Acesso em:

06/06//2018.

How to Interface PS/2 with FPGA/CPLD UDB. Dísponível em:

<https://www.pantechsolutions.net/fpga-tutorials/how-to-interface-ps-2-with-fpga-cpld-udb>.Acesso em: 08/06/2018.

# 