MC613B - Relatório - Projeto Damas

Guilherme Alcarde Gallo
Marcus Felipe Botacin

15 de Abril de 2013

Conteúdo

Capítulo 1

Considerações Gerais

Um jogo de damas, em 2D, na resolução 160x120, implementado em linguagem VHDL na *FPGA ALTERA CYCLONE II EP2C20F484C7*, através do QUARTUS 9.1 SP1/SP2. Utilizou-se como entrada um Mouse com barramento PS/2 através do componente mouse_ctrl e, como saída, o monitor VGA, via componente VGAcon. Realizou-se ainda, na placa, a implementação de uma memória RAM, uma memória ROM e uma máquina de estados que viabiliza o controle do jogo. O jogo é controlado por 3 clocks, sendo a tela atualizada na frequencia de 27 MHZ, assim como a lógica do jogo, e o mouse lido a uma taxa de 24 MHZ; finalmente, as estatísticas, como o relógio, são atualizadas na frequencia de 50MHZ, visando à verificação de um funcionamento mais preciso com este clock.

1.1 Objetivos do Jogo

Texto adaptado da Wikipedia:

"O jogo de damas pratica-se entre dois jogadores, num tabuleiro quadrado, de 64 casas alternadamente claras e escuras, dispondo de 12 pedras brancas e 12 pretas.

O objetivo é capturar ou imobilizar as peças do adversário. O jogador que o conseguir ganha a partida.

O tabuleiro deve ser colocado de modo que a casa angular à esquerda de cada parceiro seja escura.

No início da partida, as pedras devem ser colocadas no tabuleiro sobre as casas escuras, da seguinte forma: nas três primeiras filas horizontais, as pedras brancas; e, nas três últimas, as pedras pretas.

A pedra movimenta-se em diagonal, sobre as casas escuras, para a frente, e uma casa de cada vez. A pedra pode capturar a peça do adversário movendo-se para frente.

Quando na casa contígua a uma pedra houver uma peça adversária, com uma casa imediata vaga, na mesma diagonal, a pedra toma-la-á passando

para a citada casa vaga." Veja mais aqui

1.2 Funcionalidades

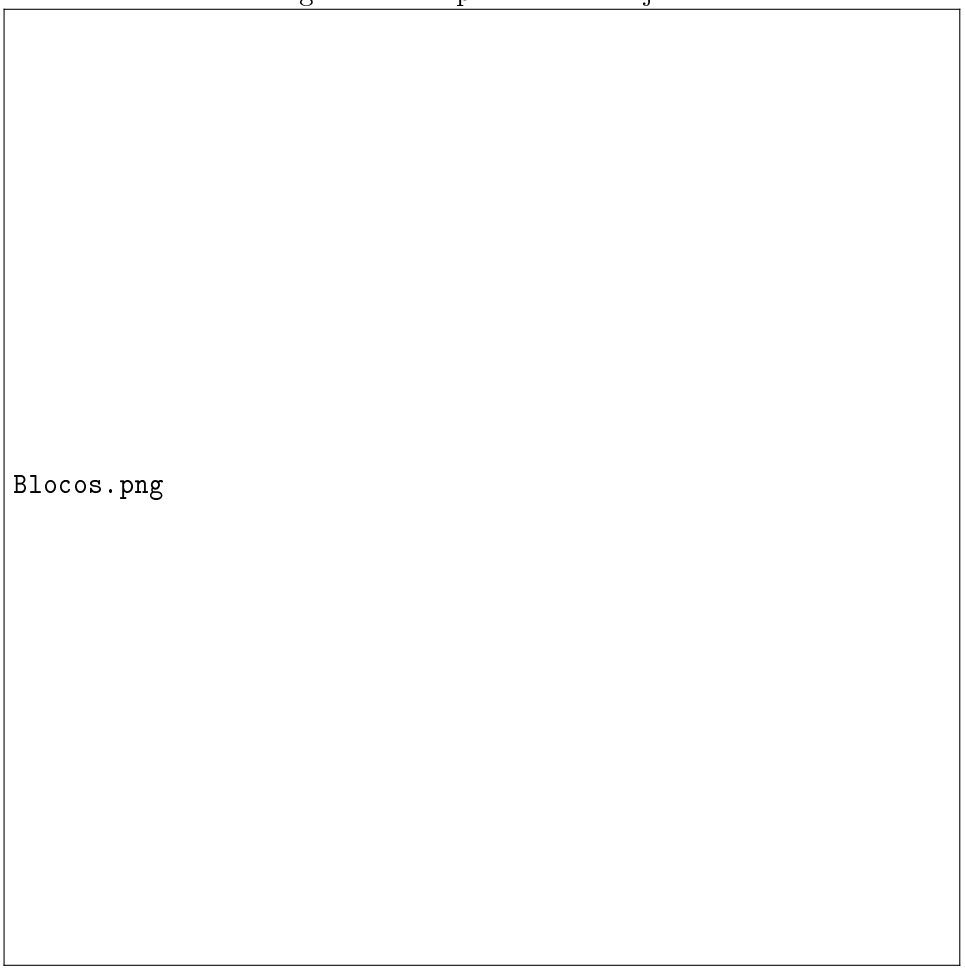
O jogo encontra-se funcional nos seguintes aspectos:

- Movimentação Simples
- Movimentação para comer peças
- Alternância de turnos
- Validação de jogadas
- Cancelamento de seleção de jogada
- Contador de peças
- Relógio de tempo de jogo
- Representação gráfica

1.3 Especificações

Seguimos como referência para implementação do jogo, o diagrama apresentado em aula. Este é composto pelo TOP LEVEL

Figura 1.1: Top Level do Projeto



Damas que instancia 3 classes de componentes:

1. Entrada (inputs do jogo) **??**
2. Engine (funcionamento do jogo) **??**
3. Saída (output) **??**

1.4 Observações

O jogo encontra-se pendente nos seguintes aspectos:

- Peça Damas

  Dado a necessidade de validação alternativa de jogadas para esta peça, esta funcionalidade não se encontra plenamente funcional, no entanto sua facilidade de implementação posterior deve ser destacada visto que basta adicionar um bit na identificação de cada tipo de bloco, prevendo este caso. [ver seção **??** – representação dos bits]

- Comer Para Trás e Comer Múltiplas Peças

  Assim como no item anterior, a validação extra de movimento é necessaria para tal caso. Essa função não foi implementada completamente por falta de tempo para testes, no entanto, dado o funcionamento do jogo nos demais casos, pode-se constatar sua viabilidade técnica de implementação. No que se refere a comer múltiplas peças, tal função está desabilitada, em virtude da função comer para trás não estar funcional. Ainda que múltiplas comidas de peças à frente pudessem funcionar, acreditamos que introduzir este recurso desta maneira poderia prejudicar os jogadores em algum momento da partida.

- Funções não previstas, mas com possibilidade de implementação futura

  - Soprar peças
    Acreditamos ser simples a implementação deste recurso ja que um relógio ja se encontra presente no jogo.
  - Fim do jogo por tempo
    Tal qual o item anterior, pelo relógio estar presente, é possível definir um tempo máximo de partida e assim encerrar o jogo.
  - Vídeo de Abertura
    Algo supérfluo a funcionalidade do jogo, mas o intuito era deixá-lo mais apresentável e menos enjoativo, já que a tela quase não muda no decorrer do mesmo.
  - Visualizar possibilidades de movimentos
    Seria possível utilizando a mesma lógica para validação de movimentos, destacar as partes do tabuleiro viáveis de movimentação. Destacamos, para esta implementação, o cuidado necessário com o tempo de barramento aberto, pois pode intereferir na VGA. [ver seção **??** – barramento].

Capítulo 2

Descrição dos Componentes

Parte I

Entrada

2.1 Especificação

A entrada do jogo de dama é dada pelo mouse. Para realizá-la utilizamos a implementação do Protocolo PS/2 fornecida através dos componentes io_base e mouse_ctrl. Nosso componente instancia tais componentes dentro dele para receber os sinais do mouse, e pode repassá-los aos TOP LEVEL tanto diretamente, quanto após um processamento prévio, descrito abaixo.

2.1.1 Modo de Mapeamento

Optamos por mapear diretamente a posição do mouse nas casas do tabuleiro através de um HASH simples, para isso dividimos os registradores de deslocamento X e Y do mouse entre as 64 casas do tabuleiro. A posição esquerda superior, que consiste do menor numero em complemento de 2 deslocamento X e Y, é considerada 0 e a posição 63 a maior,respectivamente. A definição de uma casa consiste em: $TAMANHO_{MAX\ REGISTRADOR}/CASAS_{LINHA\ OU\ COLUNA}$ ou seja, $128/8 = 16$ movimentos do mouse.

2.2 Função

A função do módulo de entrada é decodificar a entrada do mouse, seu posicionamento e cliques, para a engine.

Parte II

Engine

2.3 Especificação

Recebe a Matriz do Tabuleiro e a entrada do Mouse. Envia os dados interpretados para a Saída.

2.4 Função

A função da Engine é tratar da lógica do jogo em geral. Desde movimentos com o mouse para selecionar até o fim do jogo. Além disso é responsável também pelas estatísticas, como: tempo de jogos e número de peças remanescentes.

2.5 Ideia Geral

2.5.1 A Máquina de Estados

A máquina de estados controla todo o funcionamento do jogo, desde a validação dos movimentos até abertura dos barramentos.

Ela consiste de 3 estados básicos:

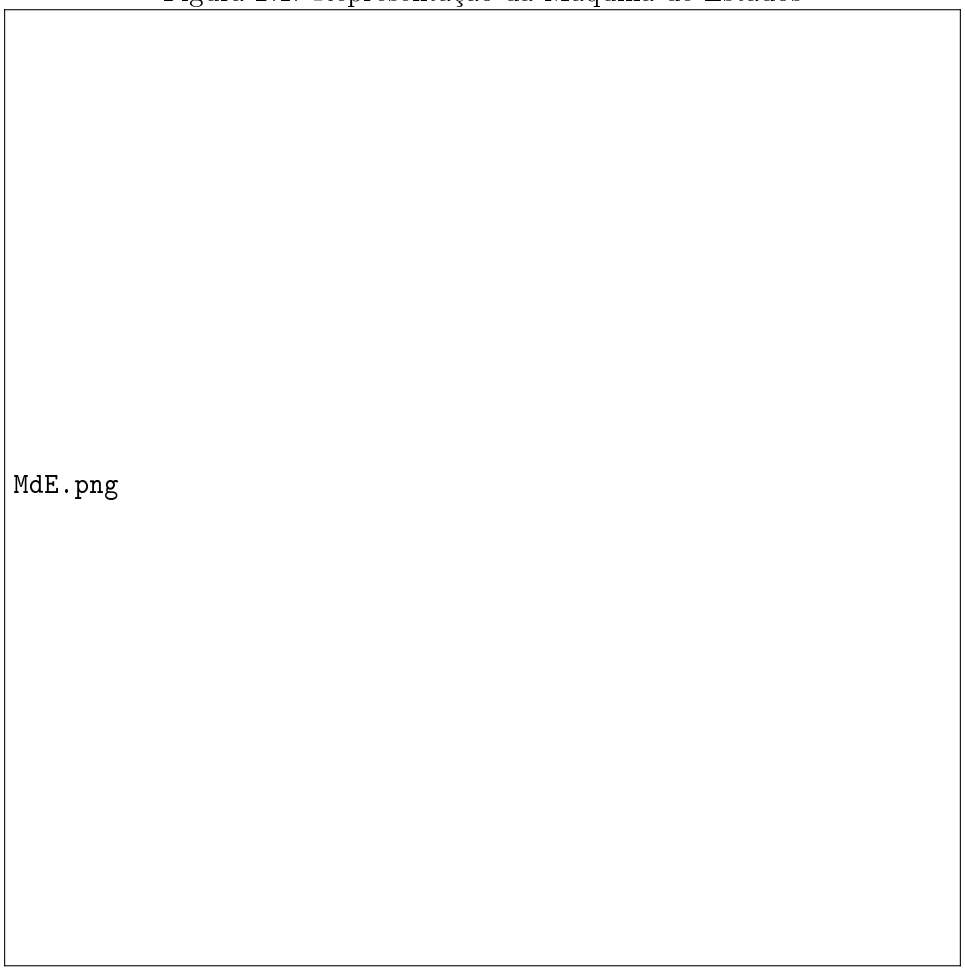
- À espera de um clique
- Já tendo sido clicada 1 vez
- Após segundo clique

Cada estado, possui uma lógica de interna de funcionamento, que aqui chamaremos de sub-estado. O estado à espera de um clique é ativado pelo evento $MOUSE =' 1'$, quando este é feito, os subestados comandam: a leitura do barramento, a alteração dos bits necessários e a escrita dos novos bits.

Dado um segundo clique, é verificado se este foi na mesma casa do tabuleiro ou não, significando o cancelamento da ação. Em caso afirmativo, os bits anteriores são restaurados em processo análogo ao descrito no estado anterior. Em caso afirmativo, um movimento é previsto, assim, os bits são lidos da memória, e verifica-se então se é possível. Caso não seja, volta-se para o estado *1 clique*, caso seja, avança-se para o *2 cliques*.

O estado *2 cliques* é responsável por: finalizar o processo e resetar a máquina; gravar todos os bits alterados; inserir uma casa preta no tabuleiro, caso alguma peça tenha sido comida; ativar o decrementador de peças e resetar a máquina.

Figura 2.1: Representação da Máquina de Estados



2.5.2 Estatísticas

Relógio

Foram adaptados os módulos feitos no laboratório 8: o relógio, o divisor de clock e os contadores.

Contador de peças

Dois registradores de reset assíncrono, os quais tem valor inicial 12 (numero de peças) e que de acordo com o sinal JOGADOR, decrementa um deles no pulso de clock.

2.6 Observações

Por problemas de debug, atualmente o mesmo está instanciado ainda no TOP LEVEL e opera ao lado de um *process* na máquina de estados, para fazer a conferência dos dados. No entanto, dado que o problema já foi solucionado, é possível transferí-lo para o local original da hierarquia.

Parte III

Saída

2.7 Especificação

A VGA recebe muitas informações de outros módulos, devido a alta demanda de dados para o jogador ao mesmo tempo na tela. Dentre elas, são destaques:

- Número de Peças
- Tempo de Jogo
- Indicativo da vez do Jogador
- Matriz do Tabuleiro
- Endereço do ponteiro vindo da validação de Movimento

2.7.1 VGA

A interface engine->monitorVGA e topLevel->monitorVGA é tratada apenas pela VGAcon aliada ao arquivo *test.vhd*.

2.8 Função

A saída é responsável por formar a imagem que será vista pelo usuário final do jogo através da leitura da Matriz do Tabuleiro. Isso pode ser visto na figura abaixo:

2.9 Ideia Geral

O módulo em si roda em torno da VGAcon modificada. A ideia é lograr do abstração de um tabuleiro de damas ser separável em 64 blocos com poucas e limitadas opções de comportamento para os mesmos. Por exemplo: um bloco pode representar uma casa preta/branca, com/sem peça e/ou selecionada ou não.

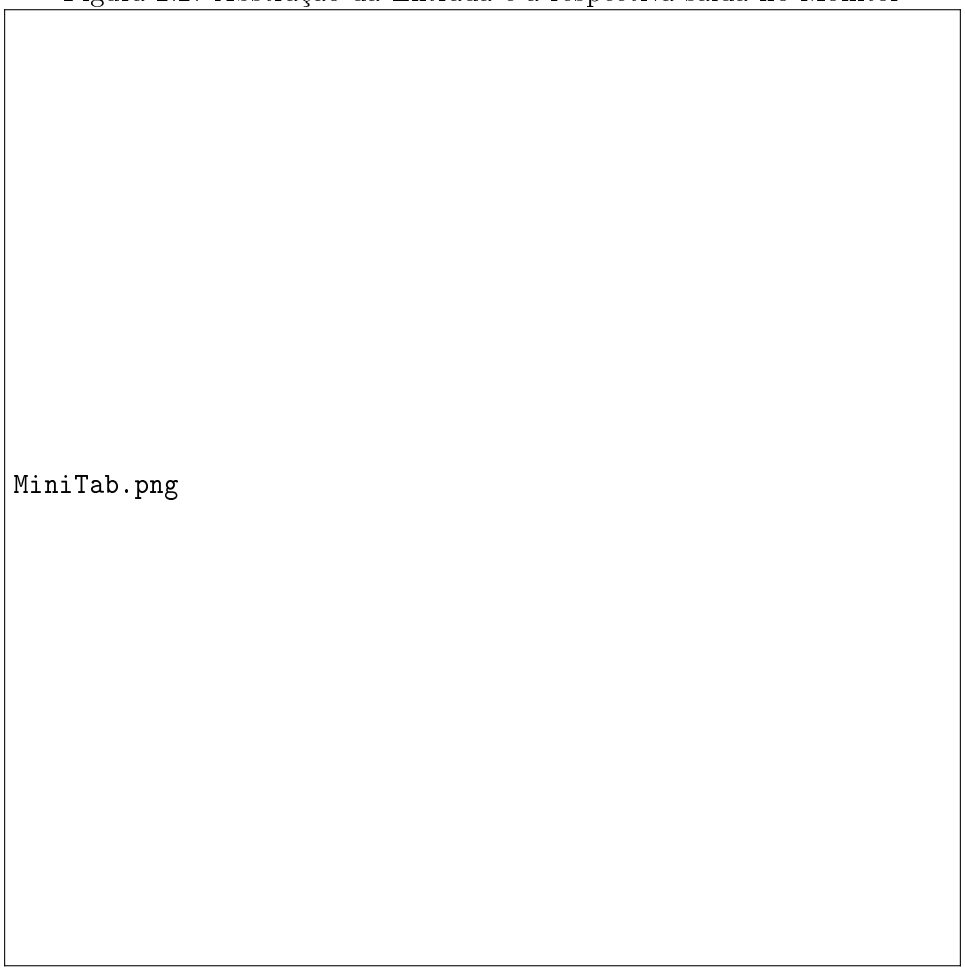
Sistema de Sprites

O sistema de gráficos por sprites consiste no aproveitamento da repetição de um mesmo bloco de imagem na tela, transformando este atributo em mapeamento para economizar memória. Para isso foi necessário modificar a VGAcon.

Modificando a VGAcon

A fim de economizar memória, gerou-se uma Memória ROM para armazenar uma figura em .mif, no qual o sistema de sprites foi utilizado.

Figura 2.2: Abstração da Entrada e a respectiva saída no Monitor



A memória interna desta foi removida e a memória ROM foi implementada em seu lugar.

Foi aproveitado o programa de *test.vhd*, o qual faz a varredura pixel por pixel numa orientação horizontal, para fazer uma função hash que lê o valor dos bits de cada elemento da matriz do tabuleiro e – simultaneamente –, de acordo com o último valor, varre um conjunto de 15 pixels horizontais que representam a figura correta da casa.

Assim a imagem correta será gerada com uma economia de 80% de memória.

Cálculo da Função HASH

Para realizar a ação de leitura simultânea Matriz-ROM é necessário fazer um cálculo baseado no endereço $read\_address$ que é o pixel que está sendo

impresso na VGA. É importante lembrar que os pixels estão sendo varridos de maneira horizontal e descendente.

Pela figura dos sprites, observa-se que cada casa tem $15px$, por isso, o cálculo será baseado neste fator de divisão.

Primeiro, é importante saber a posição da casa referente ao pixel que está sendo impresso. Para isso, calcula-se o i e o k, os quais são – respectivamente – a abscissa e a ordenada da casa.

$$i = \frac{read\_address\%larguraTela}{larguraTela/8}\ \%8$$

$$k = \frac{read\_address}{larguraTela * 15}$$

Depois, calcula-se a linha que está sendo impressa na varredura dos pixels. Isto será importante para garantir a impressão correta do bloco na tela.

$$j = \frac{read\_address}{larguraTela}\ \%\ 15$$

Impressão do Tabuleiro O tabuleiro é um quadrado de 8 casas x 8 casas de $15px$ de lado. Por isso, terá dimensões de 120x120 px^2.

Agora a função HASH começa a ser aplicada. Assumindo que $ROMaddress$ seja a variável que aponta para o endereço da memória ROM e que $desloc$ seja o deslocamento realizado devido ao valor lido na Matriz do Tabuleiro, tem-se que:

$$ROMaddress = read\_address*160 + read\_address\%larguraTela\ -15*i + 15*desloc$$

Pode-se também calcular a posição da casa na Matriz de Tabuleiros:

$$Posicao = k * 8 + i$$

Impressão da Lateral A imagem lateral utiliza os mesmos conceitos da formação de imagem do tabuleiro, só que as dimensões são diferentes e as imagens são fixas, exceto do relógio e do contador de peças remanescentes.

2.10 Observações

A resolução da tela poderia ser aumentada para muito mais, algo em torno de 640x480. Porém a resolução 160x120 foi utilizada desde o começo. A transição para algo maior não foi completamente realizada devido a falta de tempo. Pois todos os cálculos da função HASH mudariam e tais são sensíveis: um erro de cálculo, distorce toda a imagem.

Parte IV

Outros Componentes

2.11 Especificação

2.12 Função

Utilizou-se a instanciação direta da memória tal qual o slide-aula 10. Isto evita o uso de FLIP-FLOPS na entrada e saída da RAM, podendo ser gravada e lida com apenas um pulso de CLOCK. Isto permite um rápido compartilhamento de barramento. [ver seção **??** – barramento]

2.13 Ideia Geral

- A representação das peças

  Na memória RAM foram armezanados os valores de cada posição do tabuleiro, sendo que cada valor tem sua respectiva interpretação.

  Representação dos valores dos bits da matriz:

0000	Casa Preta
0001	Casa Branca
0010	Peça Azul
0011	Peça Vermelha
0100	Seleção de 0000
0110	Seleção de 0010
0111	Seleção de 0011
1XXX	Seleção de 0XXX

  Isto está ilustrado na seguinte figura:

- Memória RAM - A Matriz do Tabuleiro:

  Foi instanciada uma Memória RAM de uma porta de leitura e outra de gravação, para acomodar uma matriz de $4bits * 64$, que representa o valor da casa para um total de 64 casas.

- Acesso Inteligente à Memória

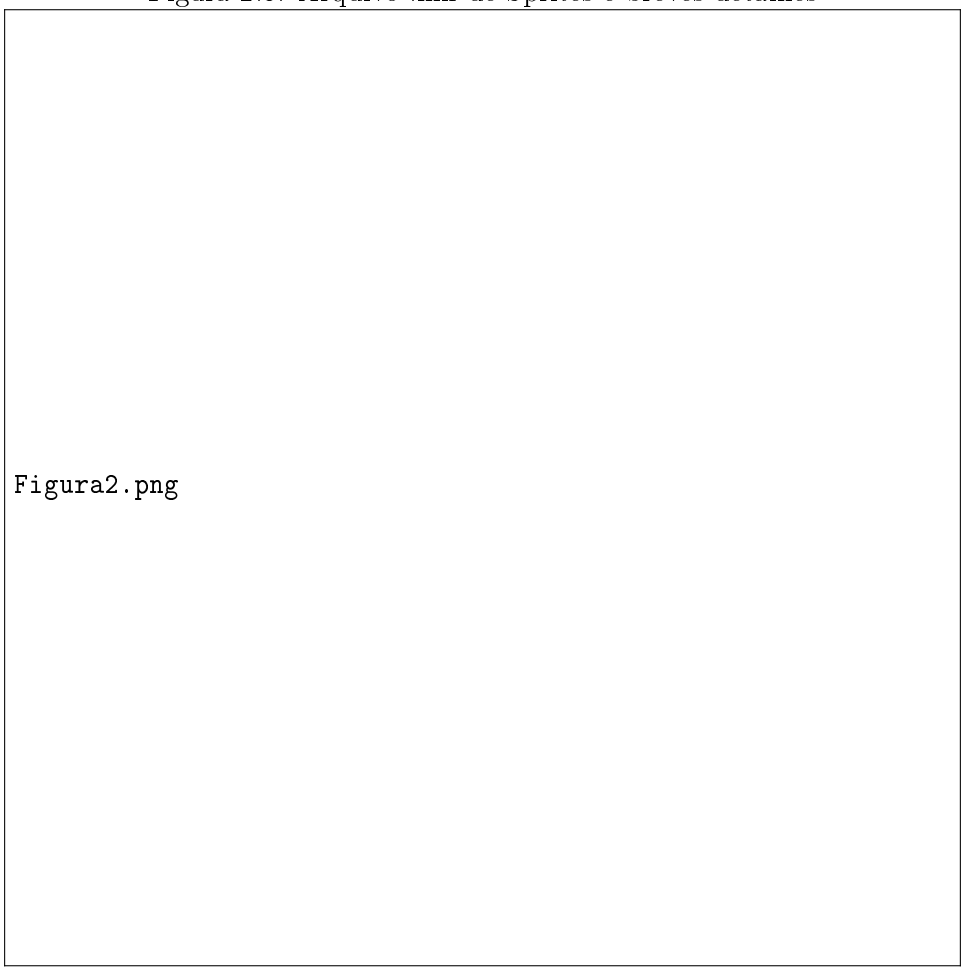
  As 64 posições do tabuleiro foram representadas num vetor 0-63, em conformidade com o processamento da posição do mouse, permitindo assim, num evento de clique, repassar o sinal de mouse diretamente como endereço da RAM, tornando o acesso mais eficiente.

- Barramento

  Tanto a VGA quanto a lógica da máquina de estados utilizam-se de dados contidos na Memória RAM, isto causa um conflito na requisição dos dados, sendo necessário então criar uma linha de dados com seleção de pedidos.

  A Memória RAM está instanciada no TOP LEVEL servindo as camadas inferiores e o controle do barramento está feito num *process* neste, mas pode ser facilmente transformado em um componente.

Figura 2.3: Arquivo .mif de Sprites e breves detalhes



A VGA foi tomada neste caso como padrão de requisições, cabendo então a lógica (muito menos frequente), interromper o fluxo de dados. Quando a máquina de estados emite um $'1'$ na saida, após sair de algum dos estados, um multiplexador altera a entrada de dados na RAM para o canal de dados da máquina de estados, voltando, automaticamente, no próximo clock, a colocar o canal da VGA como padrão. Desta forma, a varredura da VGA é muito mais frequente que o tempo que o barramento fica bloqueado, passando a impressão de persistência visual ao usuário.

Capítulo 3

Conclusão

O projeto, em sua especificação, foi consideravelmente cumprido. Muitos itens são difíceis de implementar e só alguns deles de fato foram, como a Máquina de Estados e a VGA.

Apesar de tudo, consideramos o projeto de Damas inovador na disciplina, pois cobrou um certo nível de abstração tal qual exigiu da dupla uma destreza de engenharia para: retomar os conhecimentos anteriores, mixar aos novos e implementar no projeto num certo período de tempo. E ainda não havia acesso a trabalhos anteriores para facilitar.

Em suma, foi de grande ajuda para nós. Desenvolvemos a nossa capacidade de: lidar com bugs, modularizar projetos grandes, paralelizar o trabalho e conectar tudo no final. Características que fazem parte da formação de um engenheiro de computação.