

# Projeto Aplicativo: Jogo SNAKE no processador MIPS em FPGA

André Almeida                      Bruno Tengan                      Luan Araújo  
 Lucas Lacerda                      Matheus Crestani                      Marcus Lamar\*

Universidade de Brasília, Dep. Ciência da Computação, Brasil



Figura 1: Background padrão a direita e alterado para tela de abertura do jogo Snake em FPGA a esquerda.

## RESUMO

Este trabalho apresenta o projeto final da matéria de Organização e Arquitetura de Computadores ministrada pelo professor Marcus V. Lamar, Universidade de Brasília, no segundo semestre de 2015. O seu intuito é a documentação do projeto requerido na disciplina, que consiste na reformulação do jogo clássico Snake em Assembly MIPS para ser executado no processador MIPS pipeline de 50MHz implementado no programa Quartus-II da Altera, e sendo executado na placa DE2-70. O artigo primeiramente mostra a metodologia utilizada pela equipe na execução do projeto, os desafios enfrentados durante o desenvolvimento do jogo, assim como as decisões tomadas para solucionar ou contornar tais situações adversas. Em seguida, o programa executado será explanado em todos os aspectos de interesse julgados pelo grupo como essenciais. Finalmente, os resultados são analisados e apresentados em forma de vídeo por meio do Youtube.

**Palavras-chave:** jogos, processador mips, fpga.

## 1 INTRODUÇÃO

Este documento apresenta o projeto final da disciplina Organização e Arquitetura de Computadores ministrada pelo professor Marcus V. Lamar no segundo semestre de 2015. O projeto requerido na disciplina consiste na reformulação do clássico jogo Snake [1]. Nesta disciplina, é estudada a arquitetura do processador MIPS e suas implementações Uniciclo, Multiciclo e Pipeline [2]. Nas aulas de laboratório o processador é implementado usando kits de desenvolvimento DE2-70 da Terasic [3]. O chip FPGA da família Cyclone II existente no kit é fabricado pela empresa Altera, a qual fornece a ferramenta de síntese Quartus-II [4]. Neste projeto, o jogo Snake, limitado a 4096 words (32 bits) de instruções e 2048 words de região de dados, é programado usando Assembly MIPS através da ferramenta de simulação e montagem Mars [5]. O processador deste

trabalho é definido usando a Linguagem de Descrição de Hardware Verilog[6], com a organização em pipeline. Um dos requerimentos do projeto do segundo semestre de 2015, é a modificação do processador de forma a poder ser operado a 50MHz de frequência de clock. O presente trabalho cobrirá primeiramente a metodologia utilizada pela equipe na execução do projeto, os desafios enfrentados durante o desenvolvimento do jogo, assim como as decisões tomadas para solucionar ou contornar tais situações adversas. Em seguida, o programa executado será explanado em todos os aspectos de interesse julgados pelo grupo como essenciais. Finalmente, os resultados são analisados e apresentados em forma de vídeos por meio do Youtube. Na seção 2 será apresentada a metodologia utilizada e na seção 3 detalhes do código desenvolvido são dados. A seção 4 apresenta os resultados obtidos. A seção 5 conclui este trabalho.

## 2 METODOLOGIA, DESAFIOS ENFRENTADOS E DECISÕES DE PROJETO

Nesta seção serão explicados todo o funcionamento do programa em alto nível, os problemas encontrados e as soluções encontradas, de modo a facilitar o entendimento do programa e do desempenho obtido com a sua versão final.

### 2.1 Funcionamento

No início do programa, um menu é apresentado ao usuário para que ele escolha o modo de jogo entre as diversas opções apresentadas. Entre essas opções, estão:

\*e-mail: lamar@unb.br

### 2.1.1 Dificuldade

A dificuldade no jogo consiste na velocidade com que a cobra se movimenta. Três valores de dificuldade podem ser escolhidos: fácil (valor 1), médio (valor 2) e difícil (valor 3).

### 2.1.2 Quantidade de jogadores

Duas opções: um único jogador (single player, p1), idêntico ao jogo Snake clássico, e dois jogadores (multiplayer), descrito no item a seguir.

### 2.1.3 Modos multijogador - p2 e Tron

O modo p2 (Dual Player), onde dois jogadores competem por pontos, evitando se tocarem, seguindo todas as regras do modo clássico. Neste modo, as cobras também morrem ao irem de encontro com o corpo da cobra adversária. No modo Tron [7], as cobras crescem ininterruptamente, não há comida nem pontuação, e o objetivo do jogo é sobreviver mais tempo. As cobras ainda morrem ao irem de encontro aos limites da tela, então o jogador com a melhor estratégia para encurralar o outro jogador ganha a rodada. Ao termino do jogo, o jogador com vidas sobrando é o vencedor. O usuário interage com o menu através das teclas direcionais do teclado. A seleção do jogador fica em evidência após a seleção, e a confirmação da escolha deve ser feita por meio da tecla Enter. O jogador possui sempre três vidas, independente do modo de jogo. Após a tela de menu o jogo começa. O jogador 1 (um), que joga independente da escolha de modo, controla sua cobra por meio das teclas 'W', 'A', 'S' e 'D' no teclado. O jogador 2 (dois), que joga apenas no modo multijogador, controla sua cobra por meio das teclas 'O', 'K', 'L', e 'C'. Ao fim de todas as vidas de um dos jogadores, uma tela de fim de jogo é apresentada e a tela de menu retorna ao monitor.

## 2.2 Desafios e Soluções

A conclusão deste projeto foi uma tarefa árdua realizada em aproximadamente 2 meses. Diversas situações adversas foram encontradas. As mais relevantes estão listadas a seguir:

### 2.2.1 Tratamento de Exceções

O maior problema encontrado sem dúvida foi a ausência de uma estrutura de tratamento de exceções funcional. As versões do processador MIPS V3 e do programa de tratamento de exceções 'system41.s' disponibilizados na disciplina não permitiam a execução de operações do tipo system call, devido a prováveis bugs na implementação em Verilog do processador.

### 2.2.2 Imagem de Plano de fundo

Houve um esforço para estabelecer uma imagem de background [1] durante a inicialização do processador. Para isso, dois conversores de imagem para o formato MIF (Memory Interchange Format) foram disponibilizados no site da disciplina. Porém, o programa 'bmp2mif.exe', embora convertesse corretamente para o formato MIF, sua formatação não era compatível com o processador. O programa 'png2mif.exe', por outro lado, carece de uma biblioteca pertencente ao pacote OpenCV instalada para seu correto funcionamento.

### 2.2.3 Frequência de clock do processador

Algumas instruções no pipeline não operam à frequência de 50MHz. As instruções identificadas foram lw (load word), sw (store word), lb (load byte) e sb (store byte), por exigirem acesso a memória há uma

requisição de tempo maior do que as demais instruções utilizadas em código.

### 2.2.4 Sintetizador de áudio

Sem a instrução syscall funcional, a única opção restante para acessar o sintetizador de áudio da placa DE2-70 era acessando diretamente a memória reservada ao codificador de áudio (CODEC). O sintetizador entretanto parecia estar incompleto, e a placa não gerava nada além de ruído ao setar os controladores dos canais para transmitirem as notas salvas na memória. Por conseguinte o código do jogo não utiliza nenhuma operação de chamada de sistema 'system call', fazendo com que todas entidades visuais do jogo sejam alterados diretamente na memória da interface VGA. Além disso, um procedimento para gerar um número pseudo aleatório teve que ser escrito para substituir a chamada do sistema de número 41 da arquitetura MIPS. Uma vez que o programa 'bmp2mif.exe' não gera um valor aceitável para a leitura pelo MIPS, editou-se os dados do arquivo '.mif' gerado pelo programa. Após sucessivos testes observou-se que o background é alterado apenas na primeira compilação do processador no Quartus-II, portanto, para definir o background correto, o processador deve ser recompilado. A Fig. mostra o resultado obtido. O jogo desenvolvido funciona perfeitamente com processador MIPS em pipeline operando a uma frequência de 25MHz, sendo esta utilizada. Isso foge a um requerimento do projeto, frequência de 50MHz, entretanto priorizou-se o pleno funcionamento do jogo. O sintetizador de áudio não pode ser usado na placa DE2-70. Alguns testes foram feitos no simulador Mars, onde a rotina de tratamento de exceções é funcional e alguns sons foram gerados, entretanto não foi possível implementar os sons definidos no escopo do projeto na FPGA.

## 3 CÓDIGO DESENVOLVIDO

O jogo possui um código extenso composto por aproximadamente 3424 linhas de código em Assembly MIPS, após a montagem na seção de texto apresenta 2763 instruções. Suas partes essenciais são explicadas a seguir.

### 3.1 Interface VGA

Devido à ausência de uma rotina de tratamento de exceções funcional, utilizou-se acesso direto à memória VGA. A memória de vídeo tem por endereço base 0xFF000000 com uma resolução de 320x240 pixels, estende-se portanto até o endereço 0xFF011BFF. Um sistema de coordenadas matricial foi adotado, com X representando as colunas, e Y representando as linhas. Deste modo através da Eq.1, é possível mapear toda a memória VGA e acessar todos os pixels da imagem por suas coordenadas.

$$\begin{aligned} \text{Endereço} &= 0xFF\ 000000 + 320 \times Y + X \\ 0 &\leq X \leq 319 \\ 0 &\leq Y \leq 239 \end{aligned} \quad (1)$$

### 3.2 Interface PS2

Os valores das codificações das teclas digitadas são guardados entre os endereços 0xFFFF0100 e 0xFFFF0107 da memória

usados como buffers que armazenam duas words, totalizando 8 bytes. Cada codificação de tecla ocupa 1 byte, com exceção das teclas especiais e do código que indica que a tecla foi solta. Assim cada passagem pela rotina de entrada de dados são analisados os 8 bytes do buffer através de máscaras binárias, começando da posição mais antiga, para sobrescrever sempre a última direção enviada. Para o menu foi utilizada uma rotina mais simples já que não há a necessidade de analisar todo o buffer. Essa facilidade nos permitiu usar teclas especiais tais como as setas direcionais e o 'Enter'.

### 3.3 Números Aleatórios

O procedimento para gerar números aleatórios visa a dispersão de frutas na tela do jogo. A implementação utilizada combina uma semente, o tempo de execução, e as entradas do usuário para gerar os números pseudo-aleatórios. A cada iteração do procedimento um valor que depende da direção que a cobra do jogador 1 (um) está seguindo são somadas as coordenadas da cauda desta. O número é normalizado para adequar-se ao padrão da área de jogo e utilizado pelo procedimento de surgimento de frutas. Esse procedimento é executado em todos os ciclos de atualização do jogo. A Fig.2 apresenta o resultado de testes de dispersão de frutas com a ferramenta bitmap do Mars.



Figura 2: Teste do procedimento de aleatoriedade simulando o desenho de frutas na tela.

### 3.4 Estrutura do Jogo

O jogo é dividido em algumas subestruturas principais, explicadas separadamente nos tópicos a seguir.

#### 3.4.1 As Cobras

No algoritmo de movimentação, adiciona-se um bloco e atualiza-se registradores com as coordenadas da cabeça e retira o bloco final da calda da cobra (exceto quando ela alcança a comida). Para cada direção seguida pela cabeça plota-se um pixel amarelo de cor ligeiramente diferente. Antes de deletar um bloco, lê-se o valor deste pixel em memória da interface VGA, com isso obtém-se a direção para atualizar na posição do fim da cobra. Essa implementação [8], [9] economiza memória e processamento quando comparado a armazenar e percorrer todas as posições do corpo da cobra. Como campo de jogo e o mapa de pixels do monitor na memória da VGA não são correspondentes uma função de mapeamento foi implementada seguindo a equação 2 a seguir.

$$\begin{aligned} \text{Endereço} &= 0 \times \text{FF} 00244 \text{ A} + 960 \times Y_p + 3 \times X_p \\ 0 &\leq X_p \leq 99 \\ 0 &\leq Y_p \leq 66 \end{aligned} \quad (2)$$

Onde  $X_p$  representa a coluna no campo de jogo e  $Y_p$  representa a linha no campo de jogo. Essa relação visa que elementos do jogo tenham uma largura mínima de 3 pixels.

#### 3.4.2 Dispersão das Frutas

A dispersão de frutas na tela utiliza o procedimento de geração de números pseudo-aleatórios para determinar a posição onde a fruta aparece. O número aleatório visa reduzir a previsibilidade do posicionamento de frutas. A área do campo de jogo corresponde à '1 1010 0010 1100' em binário, sendo os bits alternados para que haja maior variação de posição no mapa da forma apresentada na Fig.3.

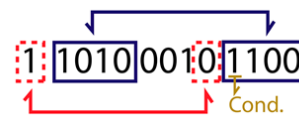


Figura 3: Adaptação do número randômico à função de plotar frutas.

O bit 3 do número em questão pode ser zerado caso o bit 12 seja 1, para garantir que a fruta não seja impressa fora do campo de jogo. O campo onde o jogo ocorre é limitado por uma área de 100x67 blocos de 9 pixels cada. Seguindo a Eq.2 de adaptação das medidas do campo para o mapa de memória da interface VGA. Para que a fruta tenha 9 pixels de área, os oito pixels ao redor do pixel resultante da Eq.2 são preenchidos da mesma cor na memória de vídeo.

#### 3.4.3 Modos de Jogo

No modo p1 (jogador único) o objetivo do jogador é acumular pontos comendo as frutas enquanto no modo p2 (dois jogadores) ganha aquele com mais pontos. Ao final do modo p2, o jogador que não perder todas as suas vidas ganha 10 pontos para cada vida que tenha. No modo Tron ganha o jogador que sobreviver.

#### 3.4.4 Menu

As setas direcionais definem o modo de jogo e a dificuldade. A tecla 'Enter' confirma a escolha do jogador.

#### 3.4.5 Placar

Um placar informa os atributos de jogo, vida (V), pontuação (P) e dificuldade (D) e delimita uma região onde o jogador poderia transitar, de 300x201. Para a apresentação de números inteiros foi utilizada a representação em display de 7 segmentos.

#### 4 RESULTADOS

Tendo os problemas sido solucionados, o programa foi finalmente implementado, completamente funcional, à exceção dos efeitos sonoros, no processador MIPS com organização pipeline à 25 MHz no kit de desenvolvimento Altera DE2-70. Algumas fotos do programa funcionando estão apresentadas na Fig.4, 5 e 6.

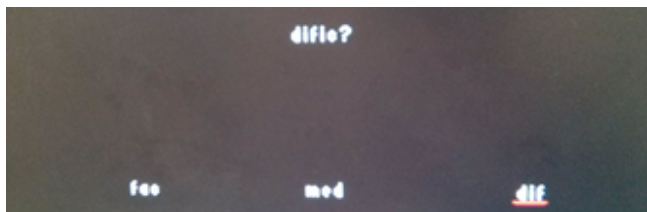


Figura 4: Menu - seleção de dificuldade.

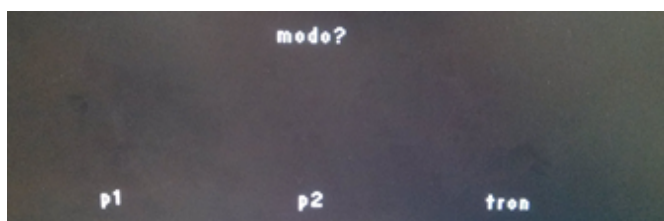


Figura 5: Menu - seleção de modo.

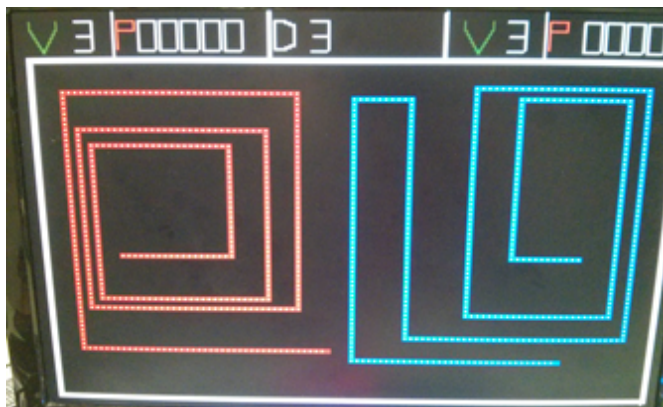


Figura 6: Modo Tron.

Mais informações podem ser encontradas no canal no Youtube <sup>123</sup>.

1 Single Player: <https://www.youtube.com/watch?v=QeUQBybHMWA>

2 Multiplayer: <https://www.youtube.com/watch?v=17mBKNhQnYk>

3 Tron: <https://www.youtube.com/watch?v=o46xtFB1oDY>

#### 5 CONCLUSÃO

A execução desse projeto passou pela análise de restrições de hardware (memória, capacidade de processamento, uso de recursos periféricos) e de software (funcionalidades de jogo). Priorizar algumas restrições levou a implementação de um jogo com diversos benefícios: diversos modos de jogo, níveis de dificuldades, com uma apresentação agradável e colorida ao usuário. Entretanto isso implicou na ausência de efeitos sonoros na placa Altera DE2-70, melhorias o processamento no hardware. Mesmo com o faltas nas restrições não foram encontrados bugs no programa, o que se deve a um controle maior do código pelo não uso de chamadas de sistemas.

#### REFERÊNCIAS

- [1] Helpfushsheep. Snake 2. [Online]. Available: <http://helpfushsheep.com/snake/>
- [2] D. A. Patterson and J. L. Hennessy, Organização e Projeto de Computadores: A Interface software/hardware 3ª edição. Elsevier, 2005.
- [3] Altera Corporation. Altera DE2-70 board. [Online]. Available: <http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=39&No=30>
- [4] —. Quartus-II - Web Edition. [Online]. Available: <http://dl.altera.com/13.0sp1/?edition=web>
- [5] K. Vollmar and P. Sanderson. MARS (MIPS assembler and runtime simulator). [Online]. Available: <http://courses.missouristate.edu/kenvollmar/mars/>
- [6] I. S. Association et al., "Ieee standard for verilog hardware description language, Design Automation Standards Committee, IEEE Std 1364TM-2005, vol. 2, 2005.
- [7] S. Ambrosio, Tron: Legacy, ser. Disney Comic (eBook). Disney Publishing Worldwide, 2012. [Online]. Available: <https://books.google.com.br/books?id=QjFk2caxoGoC>
- [8] Confect's Codex. Mips, snake, primlib. [Online]. Available: <https://dtconfect.wordpress.com/2011/11/15/mips-snake-primlib/>
- [9] —. Mips: Snake and primlib. [Online]. Available: <https://dtconfect.wordpress.com/projects/year2/mips-snake-and-primlib/>