

Universidade Tecnológica Federal do Paraná – UTFPR  
Departamento Acadêmico de Eletrônica – DAELN  
Disciplina: EL68A – Lógica programável  
Semestre: 2019/1  
Prof: Ricardo Pedroni

## **Relatório de implementação do jogo Space Invaders em VHDL**

Alunos:

Alison M. Fernandes, Brendon L. Pasquim, Isis M. Koehler, Jhoan M. I. Galvan, Jose R. L. da Silva, Leonaldo B. da S. Junior, Raissa Y. Y. Rodrigues, Victor H. S. dos Reis e Victoria B. Kung.  
jul.2019

---

### **1 Introdução**

O objetivo geral deste trabalho é o desenvolvimento em software e em hardware de um jogo tipo "Space Invaders". Foram dados uma série de requisitos para este trabalho, e algumas especificações extras foram estabelecidas pela equipe.

Este documento contém os requisitos de projeto estabelecidos entre o professor e a equipe na Seção 2. Na Seção 3 são descritas as responsabilidades de cada membro da equipe no projeto e o planejamento utilizado. Na Seção 4 são apresentadas as especificações técnicas do projeto. Na Seção 5 foram documentadas as principais etapas do processo de implementação do jogo. Na Seção 6 são apresentados os resultados obtidos durante o projeto e por último, na Seção 7 são feitas as considerações finais.

### **2 Requisitos de projeto**

Nesta seção são listadas todas as especificações do projeto definidas pelo professor ("Cliente") em conjunto com a equipe ("Empresa").

Foi passado pelo Cliente as regras do jogo listadas:

- O jogo tem 2 jogadores;
- Cada jogador controla uma nave, que fica na parte inferior do campo, e pode-se locomover apenas para os lados e atirar;
- As naves podem atirar um único tiro por vez (acerto ou saída do campo);
- Cada nave é independente e não interfere na outra;
- O jogo ocorre em fases;
- Cada fase começa com uma "onda" de naves inimigas;
- Os inimigos se movem horizontalmente até chegar na extremidade do campo, onde todos andam uma fileira verticalmente para baixo e então continuam o movimento lateral;
- A quantidade de inimigos e a velocidade que andam são proporcionais à fase atual;
- Cada jogador tem inicialmente 3 vidas;
- Os inimigos têm uma chance de 10% (pode variar com a fase) de soltar um tiro quando se movem;

- O tiro do inimigo se move verticalmente para baixo e pode atingir a nave do jogador;
- Se o tiro inimigo acertar o jogador, aquele jogador perde uma vida;
- Se o player chegar em 0 vidas, ele some do jogo e fica apenas o outro player;
- Se as naves inimigas chegarem na última fileira, o jogo acaba, independente do número de vidas de cada jogador.

Também foi passado pelo Cliente, os seguintes requisitos de hardware:

- O circuito principal deve estar numa FPGA;
- O display deve ser VGA ;
- O controle de entrada é feito por um controle criado por impressão 3D;
- Deve-se tocar um som para momentos significativos do jogo;
- O jogo deve ser em cores;
- Uma fonte de luz externa que indicará o status do jogo (e piscará quando o jogador for atingido).

## **2.1 Definições estabelecidas pela equipe**

Considerando as especificações passadas pelo Cliente, foram detalhados todos os seguimentos do jogo.

Como vencer o jogo?

R: Um jogador vencerá o jogo quando o outro jogador ficar sem vidas (0 vidas). Enquanto os dois jogadores tiverem vidas, o jogo continua e as fases aumentam sua dificuldade.

Quantas fases existem?

R: Existirão no máximo oito fases.

Quando o jogador perde vidas?

R: O jogador perderá vidas caso ele tome um tiro inimigo. Quando essa situação acontecer, um LED no controle do jogador irá piscar.

Como o jogador saberá sua quantidade de vidas?

R: O controle de cada jogador terá três LEDs, cada um indicando uma vida. No início do jogo, onde cada jogador terá três vidas, os três LEDs estarão acesos. Caso o jogador leve um tiro, por exemplo, o primeiro LED irá piscar, e em seguida permanecerá apagado. Os outros dois LEDs permanecerão acesos, indicando a esse jogador que lhe restam duas vidas.

Quais os comandos do controle?

R: O controle terá quatro botões e três LEDs. Os botões serão para movimentação (esquerda e direita), para atirar e para dar início ou reiniciar o jogo.

## **3 Responsáveis por área**

Em reunião com toda a equipe, o projeto foi dividido em blocos (vide Figura 1) e foram determinados responsáveis por cada bloco do projeto.

A organização ficou da seguinte maneira:

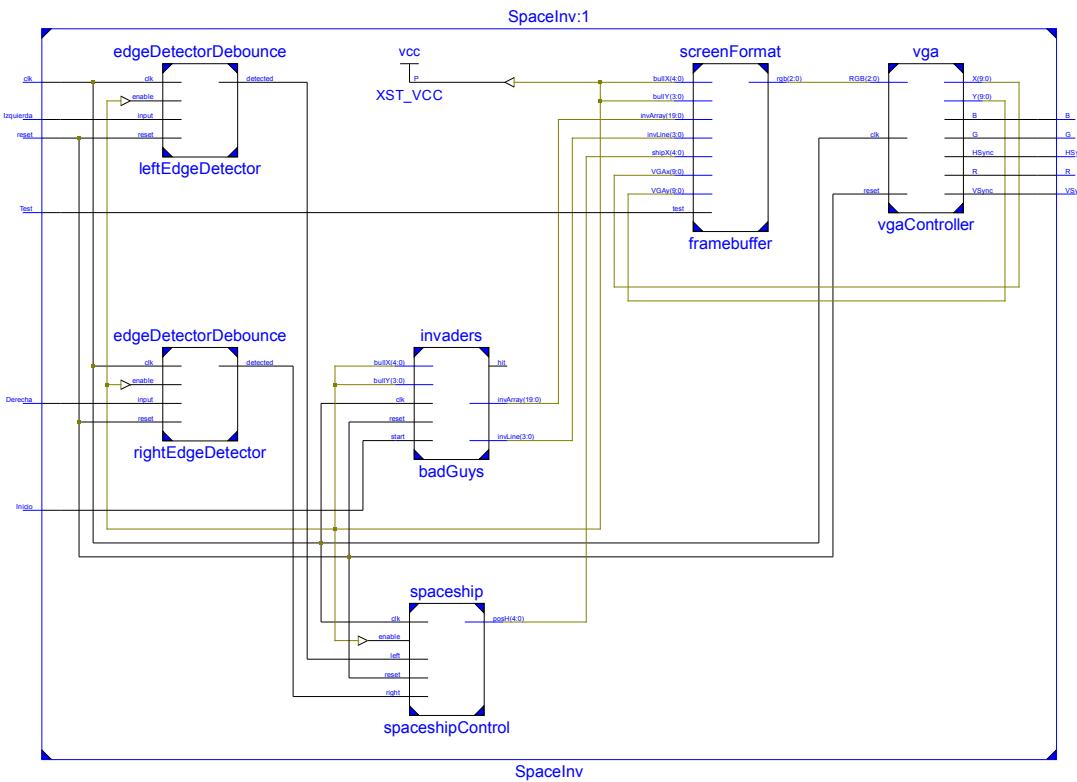


Figura 1: Diagrama de blocos do jogo

Raissa Yamasaki ficou responsável pela confecção dos controles (impressão 3D e eletrônica), pela documentação do projeto e pela gestão da equipe.

Leonaldo e Isis ficaram responsáveis pelos blocos Screen format e VGA.

Victor Salles, Alison Fernandes e Jhoan Iñiguez ficaram responsáveis bloco Invaders e todos os blocos dentro dele.

Brendon Pasquim, José Reinaldo Lopes e Victoria Kung ficaram responsáveis pelos blocos Player 1, Player 2 e pela integração dos blocos.

#### 4 Especificações Técnicas

Para facilitar a implementação deste jogo, foi utilizado um código disponível no GitHub como base, dos autores David Estévez Fernández e Sergio Vilches Expósito (<https://github.com/David-Estevez/spaceinvaders>).

#### 5 Principais etapas da implementação

##### 5.1 VGA

O bloco vgaController é responsável por gerar os sinais de controle para a tela. O padrão VGA apresenta 3 sinais analógicos para representação das cores (R, G e B) e dois sinais de sincronismo (Vsync e Hsync). Para a implementação do controlador foi utilizada uma frequência de 25MHz porém como a placa possui um clock de 50Mhz foi necessário dividir essa frequência no início do bloco VgaController, que gera uma resolução de 640 x 480 pixels a uma taxa de atualização de 60 Hz. Para formar a imagem desejada na tela,

o vgaController envia para o barramento RGB os valores presentes nas variáveis R,G e B. As variáveis x e y indicam a posição do pixel, que recebem valores da posição atual horizontal e vertical de cada pixel.

O bloco de screenFormat é responsável por manipular uma série de sinais binários dos "invaders", "ships" e "bullets" em figuras (sprites) para mostrar na tela. Nesse bloco, as variáveis x e y serão macropixels que recebem os últimos bits das variáveis vgax e vgay do bloco vgaController, para conversão de pixels em macropixels na tela, sendo 20 macropixels horizontais e 15 verticais. A variável specialScreen determina qual tela será mostrada, podendo ser: jogando, vitória da partida, vitória do jogo e derrota.

## 5.2 Controles

Os dois controles do jogo foram feitos por uma impressora 3D, pelo método FDM (*Fused Deposition Modeling*). O modelo 3D foi adquirido do repositório de peças 3D ThingVerse, depois da peça escolhida, o modelo é exportado em STL. Esse arquivo STL é interpretado pelo software CAM Repetier, que prepara o modelo para fabricação dividindo-o em camadas. O modelo escolhido é representado na Figura 2.

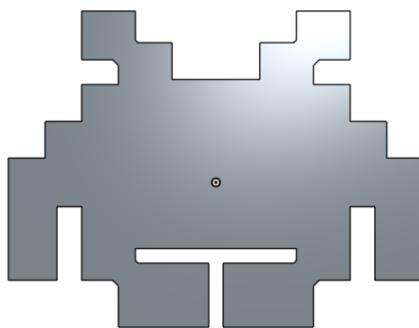


Figura 2: Imagem frontal do modelo do controle. Fonte: <https://www.thingiverse.com/thing:2718867>

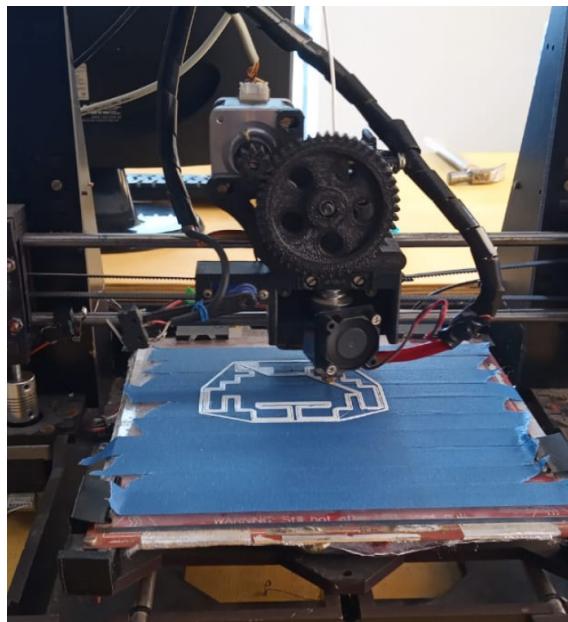
A impressora utilizada é de um dos membros do time, é um modelo Graber I3. O material utilizado foi um filamento PLS da cor branca, pela resistência do material.

Pelo modelo 3D escolhido não ser um controle, tiveram que ser feitas furações após os controles prontos. Foram adicionados para os botões chaves táteis, com lógica convencional (tensão em 3,3V quando apertado), e LEDs alto brilho para a simbolização das vidas. Para conexão dos controles com a placa, foi utilizado um cabo flat.

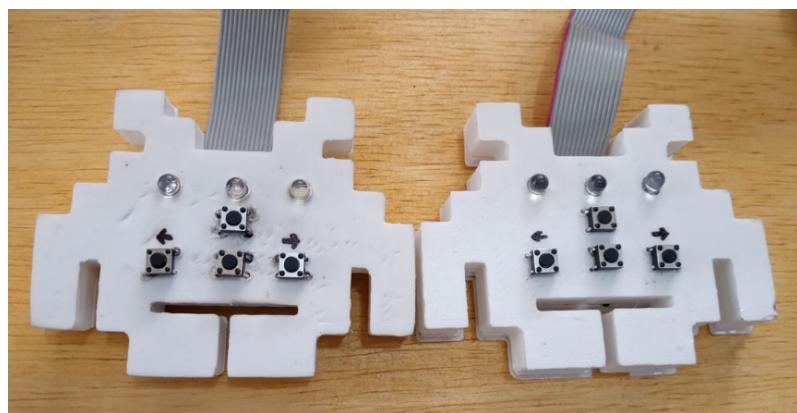
Após toda a circuitaria dos controles finalizadas, o resultado final dos controles finalizados pode ser visto nas Figuras 4 e 5.

## 5.3 Invaders

Tendo como base o código disponibilizado no GitHub, a primeira adaptação realizada para atender aos requisitos do projeto foi alterar o comportamento das ondas inimigas. A primeira mudança implementada foi fazer com que o número de inimigos por onda aumentasse conforme a fase. Além disso, foram criados diferentes tipos de inimigos, diferenciados por cores, os quais os verdes são eliminados com um tiro, amarelos com dois e vermelhos com três. Os inimigos mais fortes, que aparecem nos níveis mais avançados, são mais resistentes aos tiros, ou seja, precisam ser atingidos mais de uma vez para serem eliminados.



*Figura 3: Imagem do processo de impressão 3D.*



*Figura 4: Imagem frontal dos dois controles finalizados.*



*Figura 5: Imagem de trás do controle, mostrando a fixação da tampa com parafusos.*

Outra modificação feita ainda no início foi em relação a velocidade com que os *invaders* se locomovem e avançam em direção ao jogador. Conforme os jogadores progridem de nível, os inimigos pulam mais linhas de uma só vez. Essas mudanças foram feitas no

arquivo *invaders* do projeto, alterando o incremento das variáveis contidas dentro de um *case* que controla o número de linhas puladas e velocidade.

Mais adiante, foram implementados recursos mais expressivos que necessitaram alterações mais profundas na lógica do código base, além da adição de novos algoritmos. Uma dessas funcionalidades foi a criação do projétil para os *invaders* e o seu comportamento, ambos dentro do bloco *invaders*.

Nas regras do jogo, se o projétil inimigo atingir a nave do jogador, este perde uma vida. Essa condição foi cumprida adicionando-se os sinais das coordenadas do projétil inimigo dentro do bloco *player* e comparando-as com a posição da nave. Se as posições forem as mesmas, quer dizer que a nave foi atingida e o jogador perde uma vida. O bloco *player* também fica encarregado de dizer ao bloco *invader* se a nave foi atingida ou não. Foram criadas, além disso, três novas entradas no bloco *screenFormat*, para descrever a posição da munição do *invader* e outro sinal para conferir se ela está ativa.

Outra característica implementada no jogo é que quando as vidas dos jogadores acabam, o jogo acaba.

#### 5.4 Players

O jogo possui o *player1* (nave branca) e participa tanto no modo *single game* quanto em duas pessoa, que ativa *player2* de nave vermelha. Ambos jogadores podem atirar ao mesmo tempo, porém não podem atirar novamente até que sua bala não esteja mais representada na tela. Toda a vez que um jogador for atingido pela bala de um *invader*, perde uma vida do seu total de três e quando todos os *invaders* da fase forem eliminados, uma nova fase (de um total de 8) é iniciada, mudando os tipos e disposições dos *invaders* e aumentando a velocidade do jogo.

#### 5.5 Buzzer

O *Buzzer* foi utilizado como sinalização para os *players* em duas situações específicas, de especial importância:

- Vitória de algum *player* na fase corrente
- Morte de algum *player* na fase corrente

O som feito pelo *Buzzer* é tocado entre *bips* longos na vitória e *bips* curtos na morte de algum *player*.

#### 5.6 Máquina de Estados

A máquina de estados, denominada *currentState* que foi criada para suprir as necessidades do jogo, possui quatro estados, e um quinto utilizado apenas na fase de desenvolvimento:

- *Playing*: é apresentada a partir do momento o qual é escolhida a modalidade de jogo. Enquanto os *players* jogam, esse estado é mantido e a tela atualizada constantemente, computando pontuação e perda de vida.
- *You Win* estado que indica troca de nível de dificuldade do jogo, a partir do momento em que todos os *invaders* são eliminados e partindo do princípio de que os jogadores possuem mais do que zero vida.

- *YouLose*: indica quanto um ou mais jogadores zeraram suas vidas ou os *invaders* não foram eliminados até chegar ao final da tela.
- *YouWon*: aparece apenas quando todos os *invaders* do jogo foram eliminados sem nenhum jogador perder todas suas vidas durante o jogo.
- *TestState*: estado apenas utilizado com finalidade de depuração do código, acompanhando mudança na máquina de estados.

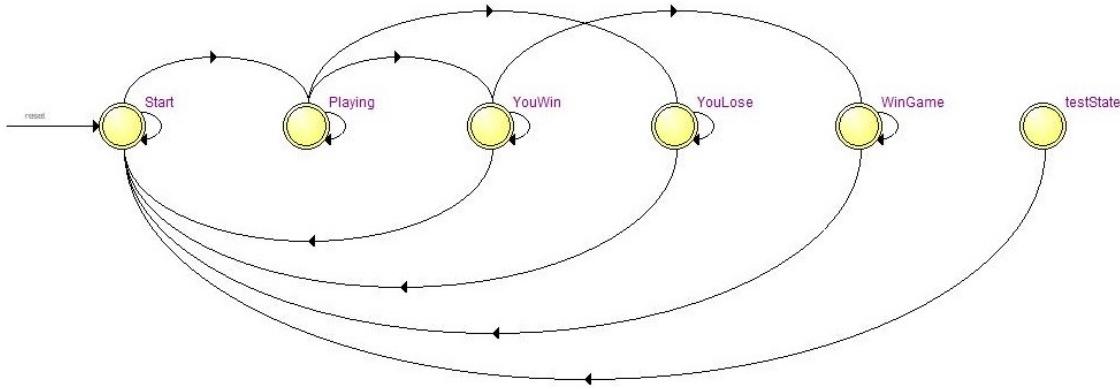


Figura 6: Diagrama da máquina de estados do jogo.

## 6 Resultados

O projeto do jogo Space Invaders foi dividido em blocos por equipe (mencionados acima) e posteriormente foi feita a integração de todos os blocos que estavam funcionais separadamente. A integração do projeto seguiu-se de acordo as etapas listadas a seguir:

- Junção dos códigos parciais do projeto em VHDL
- Correção de bugs de código do projeto em VHDL
- Teste do projeto com um monitor com entrada VGA e os botões da placa
- Inserção e teste do controle no jogo

Durante a integração do projeto do jogo as etapas citadas não foram executadas na sequência, pois a qualquer momento pode-se voltar a uma etapa, ou avançar uma etapa, para fazer a devida correção do problema enfrentado. A utilização do código base facilitou em todo desenvolvimento do projeto, principalmente na integração, já que não precisou-se fazer muitas correções nos códigos devido a boa documentação e planejamento do jogo. Os maiores problemas enfrentados pela equipe no desenvolvimento do jogo foram:

- Falha na impressão 3D do controle
- Inserção do tiro dos invaders, com chance de 10%
- Compreensão do funcionamento da entrada VGA

O projeto do jogo Space Invaders finalizou-se com sucesso e cumprindo todos os requisitos previstos no contrato feito previamente entre o professor ("Cliente") e a equipe ("Empresa"). Abaixo estão as imagens registradas durante todo o processo de desenvolvimento do projeto.



*Figura 7: Imagem de parte do grupo fingindo estar feliz.*



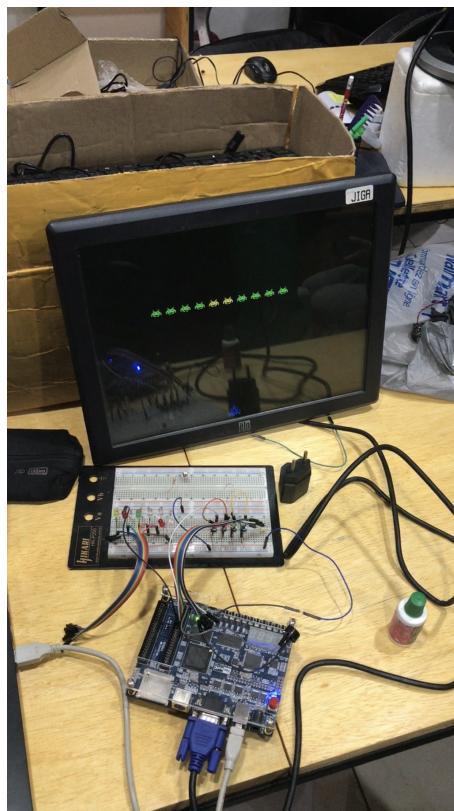
*Figura 8: Imagem do grupo se aquecendo.*

Assim, o projeto Space Invaders teve como artefatos finais:

- Código VHDL revisado e funcional
- Dois controles personalizados
- Nova documentação para o código VHDL, atualizada e mais detalhada



*Figura 9: Imagem do grupo feliz com um resultado parcial.*



*Figura 10: Imagem do circuito provisório do jogo.*

## **7 Conclusão**

O projeto Space Invaders possibilitou aprimoramento técnico bem como aprimoramento interpessoal. Para esse projeto foi necessário lidar com a gestão das pessoas da equipe e das atividades propostas.

Do ponto de vista técnico esse projeto permitiu revisar diversos conceitos abordados na disciplina de Lógica Reconfigurável, como máquina de estados, circuitos sequenciais e hierarquização de componentes em VHDL. Isto permitiu uma visão mais ampla e prática dos assuntos estudos e gerou uma sensação de "know-how" para os integrantes da equipe.

Para um projeto dessa magnitude foi necessária a divisão da equipe em subtimes, para lidar com a complexidade dos requisitos exigidos pelo cliente. Como exposto na seção 3 a equipe foi dividida em quatro e isso permitiu uma melhor utilização do tempo dos membros, bem como permitiu uma gerência de projeto mais eficiente, no qual era possível manter os módulos do projeto progredindo em paralelo e o impacto de problemas extensos eram minimizados em relação ao projeto como um todo.

Todo código e documentação podem ser vistos no repositório do GitHub:  
<https://github.com/raissayukie/SpaceInvaders>