UNO RAID: UM VIDEO GAME EM VHDL

ARTHUR PICCOLI[1]; ÍTALO NOLASCO RAMOS[2]; RAFAEL IANKOWSKI SOARES[3]; LEOMAR SOARES DA ROSA JUNIOR[4]

[1]Universidade Federal de Pelotas – apiccoli@inf.ufpel.edu.br
[2]Universidade Federal de Pelotas – inramos@inf.ufpel.edu.br
[3]Universidade Federal de Pelotas – rafael.soares@inf.ufpel.edu.br
[4]Universidade Federal de Pelotas – leomarjr@inf.ufpel.edu.br

1. INTRODUÇÃO

Uno Raid é um jogo com uma jogabilidade simples e intuitiva que segue o exemplo de projetos anteriores que mostraram ser possível emular um Atari 2600 em uma placa FPGA (FLACH, 2012) e baseado nos jogos River Raid desenvolvido pela Activision em 1982 para o console Atari 2600, e RoadBlasters lançado para diversas plataformas em 1987 pela Atari Games. Foi completamente desenvolvido em VHDL para ser executado no kit de desenvolvimento Altera DE2 Cyclone II. Sendo um jogo *single player*, consiste em pilotar um carro através de uma estrada com vários outros carros, dividida em três pistas, marcando pontos ao ultrapassá-los, aumentando gradualmente a velocidade do jogo e, assim, o nível de dificuldade fornecido ao jogador.

2. METODOLOGIA

Quartus II é o *software* de projeto de dispositivos lógicos programáveis usado para desenvolver o *hardware* e programar o dispositivo FPGA de acordo com a pinagem disponível no manual do usuário da placa Altera DE2. Como a saída do projeto é puramente visual, a porta VGA foi usada para apresentar o jogo usando uma resolução de 1280 por 1024 pixels com uma taxa de atualização de 60 Hz. Para obter a frequência necessária para o *clock* do pixel, um arquivo de design VHDL (.vhd) foi gerado usando o plug-in ALTPLL que eleva um *clock* padrão da placa de 27 MHz a 108 MHz por meio de uma malha de captura de fase (PLL). Conforme indicado no relatório de compilação, o jogo utilizou um total de 3236 elementos lógicos, 198 registros, 41 pinos e 1 PLL.

Para se ter controle dos elementos no jogo, o mesmo foi desenvolvido utilizando uma máquina de estados finitos (GOLSON, 1994) de cinco estados que é atualizada a cada quadro, ou seja, em uma frequência de 60 Hz, que é mostrada na Figura 1. No primeiro estado a pontuação do jogo é zerada e somente o carro do jogador é mostrado na pista central da estrada, com as linhas e o gramado movendo-se lentamente na vertical para criar a ilusão de movimento conforme os quadros avançam. Os objetos mencionados são deslocados 4 pixels para baixo a cada quadro, enquanto o estado espera que o quarto botão seja pressionado para iniciar o jogo, mudando assim para o próximo estado da máquina.

O segundo estado introduz o primeiro adversário na estrada, movendo-se a 4 pixels por quadro, enquanto o gramado e as linhas começam a se mover a 8 pixels por quadro, criando um efeito de velocidade maior no carro controlado pelo jogador; quando a parte superior do carro do adversário alcança a linha 384 da parte visível da tela, o jogo avança mudando para o terceiro estado da máquina.

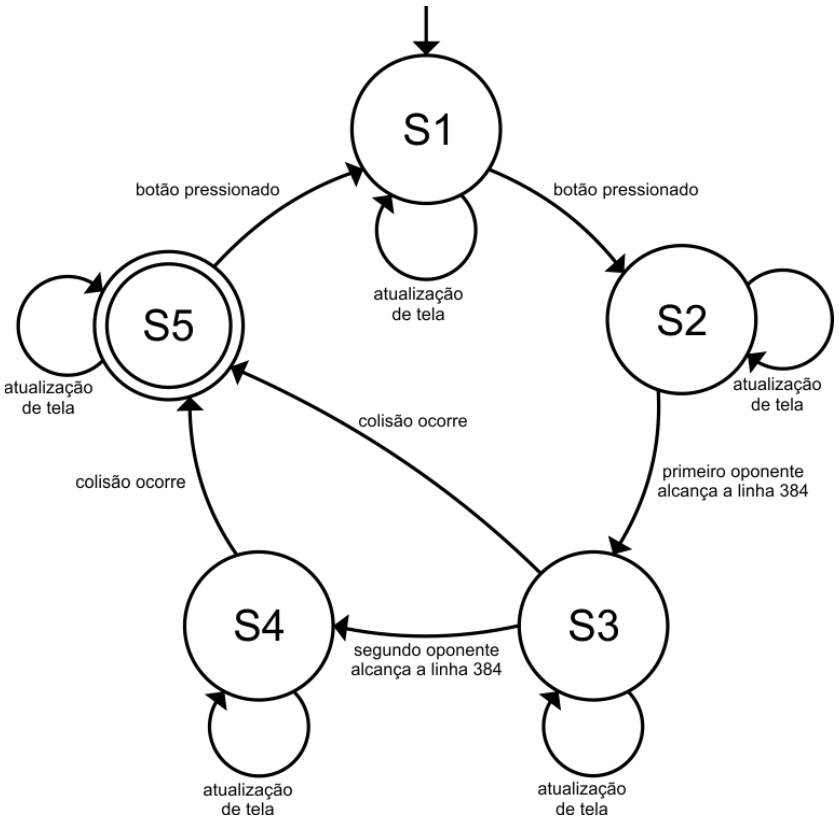

Figura 1. A máquina de estados finita que controla o jogo

No terceiro estado o segundo adversário é introduzido na tela em uma pista diferente da primeira, e quando atinge a mesma altura que a do anterior, o jogo vai para o quarto estado ou no caso de uma colisão vai para o quinto estado.

No quarto estado, o terceiro e último oponente aparece na tela deixando todos os três adversários igualmente espaçados. Assim que um carro ultrapassado deixa o escopo da tela, a pontuação do jogo é aumentada em um ponto e sua posição na estrada é alterada, fazendo com que reapareça na parte superior da tela em uma pista diferente da anterior. A velocidade na qual os carros e a estrada se movem é baseada na pontuação do jogo, aumentando em 2 o valor de deslocamento do oponente e em 4 o do gramado e das listras, quando o jogador alcança as pontuações 5, 6, 15, 16, 50 e 51. Esses aumentos subsequentes existem para criar o efeito de aceleração. O jogo permanece nesse estado até que o jogador cause uma colisão e, portanto, mude para o quinto estado.

O último estado, o quinto, congela a pontuação alcançada pelo jogador e os objetos na tela, esperando que o terceiro botão seja pressionado para retornar ao estado inicial.

O sistema de colisão funciona ativando uma *flag* quando o sinal de desenhar o carro controlado pelo jogador está ativo ao mesmo tempo que o sinal de desenhar um carro adversário, avisando a colisão entre os objetos. A posição dos oponentes é definida de acordo com a última posição em que estavam na tela, de modo que toda vez que o jogador colidir, ele mudará onde os oponentes aparecerão na tela, com base na iteração anterior, gerando uma pseudo-aleatoriedade a cada início de jogo.

A saída VGA funciona como em telas analógicas, onde cada pixel é impresso na tela através de uma varredura pixel-a-pixel (SKLIAROVA, 2005). Além dos pixels visíveis, há os pixels de *front porch* e *back porch* e os pixels de sincronização horizontais e verticais, portanto a resolução total, incluindo os elementos não visíveis, é 1688 horizontalmente e 1066 verticalmente. Para atingir a taxa de

atualização de 60 Hz, cada pixel precisaria ser atualizado em uma frequência de 107.964.480 Hz, o que é suficientemente aproximado dos 108 MHz possibilitados pelo PLL.

O jogo tem arquivos de design VHDL auxiliares que ajudam a informar como desenhar os carros e as faixas na tela. Como a única diferença entre o carro do jogador e os carros dos adversários é o acessório anexado ao topo, seus arquivos são muito semelhantes, com o do jogador tendo mais dois sinais de cor. Ambos recebem a coordenada do canto superior esquerdo do carro da parte visível da tela e verifica se o pixel atual está dentro da área que o objeto deve ocupar de 144 por 256 pixels, enviando dois sinais no caso positivo, sendo um o sinal de desenhar e outro informando qual cor deve ser impressa. Os outros objetos não possuem sinal de cor porque são mais simples. O contador da pontuação tem uma entrada específica além daquelas comuns aos outros objetos que é a responsável pelo valor que deve exibir, o qual é calculado pela máquina de estados. Os objetos das faixas móveis e do gramado funcionam como os mencionados anteriormente, mas são mais simples porque seu valor horizontal é fixo.

Todos esses sinais de controle entram em uma cadeia de funções *if* que os convertem em valores RGB que estão pinados a saída VGA. Os elementos restantes, que são fixos e não têm arquivos separados, como o asfalto, as faixas fixas e o fundo do gramado, são codificados diretamente na cadeia de funções *if*. Se todos os sinais de desenhar estiverem desligados e o pixel não fizer parte dos elementos fixos, significa que o mesmo está fora da parte visível da tela, de modo que preto puro é enviado para a saída VGA. Depois que cada pixel recebeu um valor RGB, um quadro é formado e a sequência de quadros forma a imagem do jogo, neste caso, em um monitor que a saída VGA está conectada, como visto na Figura 2.

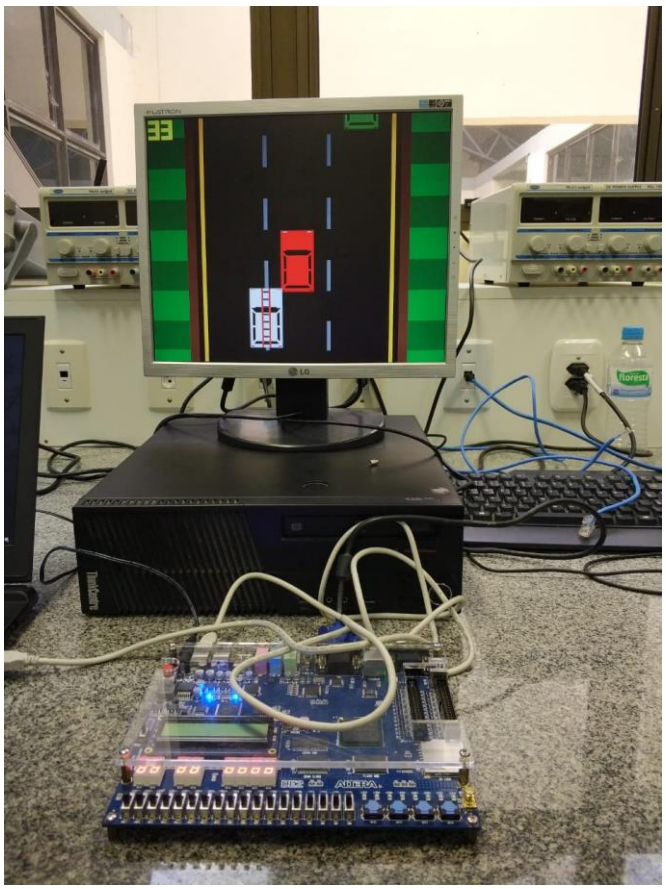

Figura 2. O jogo rodando na placa Altera DE2

3. RESULTADOS E DISCUSSÃO

Para ser simples e intuitivo, o carro do jogador é branco com um acessório no topo e pode se mover com diferentes velocidades. A primeira, e mais lenta, acompanha o jogador até a quinta ultrapassagem, quando ocorre a primeira aceleração. É possível considerar a primeira velocidade como o nível mais fácil do jogo, graças ao tempo de resposta que o jogador tem para ultrapassar os adversários. A segunda velocidade é duas vezes mais rápida que a primeira e tem como objetivo ser a dificuldade média do jogo, durando até a décima quinta ultrapassagem quando se torna três vezes mais rápida e um pouco mais desafiante. Ao atingir a quinquagésima ultrapassagem, o jogador começa a pilotar a uma velocidade quatro vezes mais rápida a qual é o nível mais desafiante do jogo, que dura até ao final.

Depois de carregar o jogo na memória interna do kit de desenvolvimento, o jogador pode começar a controlá-lo com quatro botões. Com o jogo carregado, o jogador receberá em seu monitor a imagem do carro controlável em movimento na estrada do jogo, até ele pressionar o quarto botão, quando o controle do carro vai para ele e o jogo de fato começa. O jogador pode usar o primeiro e segundo botões para mover o carro horizontalmente e assim ultrapassar os outros carros e evitar colisões.

A pontuação do jogo é incrementada a cada ultrapassagem realizada, informando o jogador através de dois dígitos localizados no canto superior esquerdo quantos carros foram ultrapassados. O jogo termina quando o jogador acaba colidindo seu carro com outro veículo.

4. CONCLUSÕES

Com uma ideia simples, Uno Raid cumpre o propósito de ser um jogo funcional em uma placa FPGA, que tem uma jogabilidade desafiadora que mantém o jogador interessado, fazendo-o tentar bater seu próprio recorde. Na parte de desenvolvimento, o jogo desafia o designer a entender como funciona a sincronização e impressão de imagens, além do uso de técnicas avançadas de VHDL e a ideia básica de como desenvolver um jogo, dividindo o projeto em objetos, mapa, máquina de estado, entre outros. O jogo também abre espaço para novos recursos, como o controle através de um teclado e efeitos sonoros para acompanhar os carros.

5. REFERÊNCIAS BIBLIOGRÁFICAS

FLACH, G.; CONCEIÇÃO, C.; JOHANN, M.; REIS, R. Revisiting Atari 2600 on an FPGA. In: **SOUTHERN CONFERENCE ON PROGRAMMABLE LOGIC**, VIII, Bento Gonçalves, 2012. Proceedings… Piscataway: IEEE, 2012.

GOLSON, S. State machine design techniques for Verilog and VHDL. **Synopsys Journal of High-Level Design**, Carlisle, v. 9, p. 1-48, 1994.

SKLIAROVA, I. Desenvolvimento de circuitos reconfiguráveis que interagem com um monitor VGA. **Electrónica e Telecomunicações**, Aveiro, v. 4, n. 5, p. 626-631, 2005.