

Desenvolvimento de um Jogo de Corrida em FPGA

Italo S. Santos

João Carlos N. Bittencourt

Anfranserai M. Dias

Universidade Estadual de Feira de Santana

Abstract

This paper describes a racing game implementation using re-configurable circuits in FPGA. The model uses as output interface the VGA monitors and manual controls using push-buttons. That guarantees a better debugging of developed circuits in FPGA, increasing its potentiality and interactivity with users. The differential of this project can be defined by the use of an integrated circuit of low cost, easy maintenance and programmed in Verilog hardware description language. This project is divided in two parts, first will be presented an approach about the content of the circuit and finally described the main parts of the developed model.

Keywords:: FPGA, VGA Interface, Verilog

Author's Contact:

italo.ecomp@gmail.com
joaocarlos@ecomp.uefs.br
anfranserai@ecomp.uefs.br

1 Introdução

O avanço tecnológico no desenvolvimento de microprocessadores cresceu de forma exponencial nas últimas décadas, tornando-os cada vez menores, e embarcando um número maior de transistores. Este ritmo de crescimento pode ser atestado a partir da Lei de Moore [Schaller 1997] e verificado pela inserção periódica no mercado de novos Circuitos Integrados (CIs) com grande capacidade de processamento. Esta lei é devida a Gordon E. Moore, que em 1965 observou que a densidade de componentes em circuitos integrados dobrava a intervalos regulares, inferindo que este comportamento perduraria por muito tempo ainda. O intervalo medido por Moore para que a densidade média de CIs dobrasse foi de 18 meses, o que ainda hoje permanece uma taxa estável [Calazans 1998].

Esta elevada taxa de crescimento resulta em um curto espaço de tempo para que um novo projeto chegue ao mercado. Aliado a isto, têm-se o aumento de complexidade dos projetos devido à alta densidade de componentes nos CIs, *System-on-Chip* (SoC), mostrando a necessidade de novas metodologias que auxiliem na construção. Uma alternativa para o desenvolvimento destes CIs, está no processo de criação de *IP-cores* baseado no RUP (*Rational Unified Process*) com prototipação em dispositivos FPGA (*Field Programmable Gate Array*).

A utilização de FPGAs para o desenvolvimento de *IP-cores* é uma alternativa para projeto de circuitos integrados, pois permite ao desenvolvedor utilizar paradigmas de programação em alto nível, podendo testar e modificar o projeto antes da fase de integração do CI [Deschamps et al. 2006]. Os FPGAs permitem o uso de linguagens de descrição de hardware (*HDL - Hardware Description Language*), como *Verilog* e *VHDL*, para criação dos circuitos e as ligações entre eles, além das linguagens de verificação de funcionamento como *SystemC* e *SystemVerilog*.

Neste contexto, este trabalho apresenta o processo de desenvolvimento do protótipo de um jogo de corrida de carros, que opera em modo gráfico. Nele o jogador poderá controlar o carro utilizando botões para alterar a velocidade e visualizar o jogo em um monitor de padrão VGA. O cenário é composto por uma pista com obstáculos, como outros carros, simulando um ambiente de corrida. Na tela são exibidas informações sobre a distância percorrida e as “vidas” que restam ao jogador. Este modelo foi desenvolvido em linguagem Verilog e sintetizado em um dispositivo FPGA.

A abordagem utilizada neste trabalho envolve a descrição de todas as etapas de desenvolvimento, da concepção inicial até a etapa atual do projeto. Apresenta-se ainda uma breve descrição dos conceitos necessários para o desenvolvimento dos circuitos. Diante disso, serão mostrados os principais módulos do circuito, destacando suas funcionalidades. Por fim serão relatados os resultados apresentados até a presente etapa de desenvolvimento do projeto, bem como as perspectivas futuras.

2 Fundamentação

2.1 Visão Geral de um Dispositivo FPGA

O FPGA é um dispositivo lógico composto por um arranjo bidimensional de células lógicas e chaves programáveis genéricas. Nesta configuração, uma célula lógica pode ser configurada (programada) para executar uma função e as chaves são utilizadas para realizar a conexão entre as células [Chu 2008]. Essa configuração permite ao programador desenvolver sua lógica para obter funções a partir de uma ferramenta de *design*, a gravação de seu programa em um dispositivo FPGA, obtendo um modelo de circuito modificável. Um diagrama simplificado da arquitetura interna de um dispositivo FPGA pode ser vista na Figura 1.

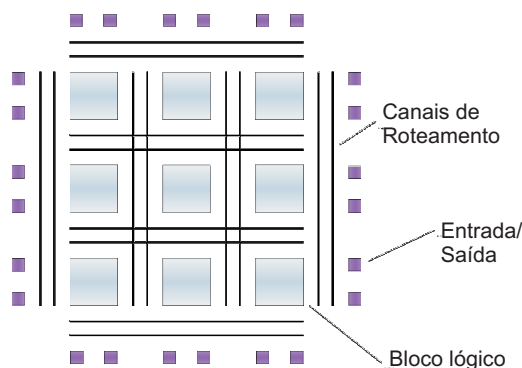


Figura 1: Diagrama simplificado da arquitetura de um FPGA

2.2 IP-Cores

Um *core* pode ser definido como a descrição de um sistema digital. Quanto à disponibilização, um *core* pode ser classificado como licenciável quando o fornecedor disponibiliza um projeto e um conjunto de ferramentas, conferindo a tarefa de projeto do sistema e sua fabricação ao cliente. Além disso um *core* pode ser definido como proprietário. Neste caso, a preocupação principal do fornecedor é proteger sua propriedade intelectual. Quanto à flexibilidade, um *core* pode ser classificado em três categorias: *hard*, *firm* e *soft*. A Tabela 1 mostra esta classificação [Palma et al. 2001].

A partir de uma análise da Tabela 1, pode-se classificar o *IP-core* descrito neste trabalho como do tipo *soft core*, uma vez que os sistemas se apresentam abertos a novas implementações e sua síntese pode ser transcrita para diversas tecnologias de lógica programável, sem alteração do seu funcionamento. Deste modo, a estrutura elaborada está apta a receber modificações de parâmetros de forma simples e independente.

2.3 Interface VGA

Em monitores do tipo CRT (*Cathode Ray Tube*) a imagem é formada pela emissão de elétrons através de um “canhão de elétrons”

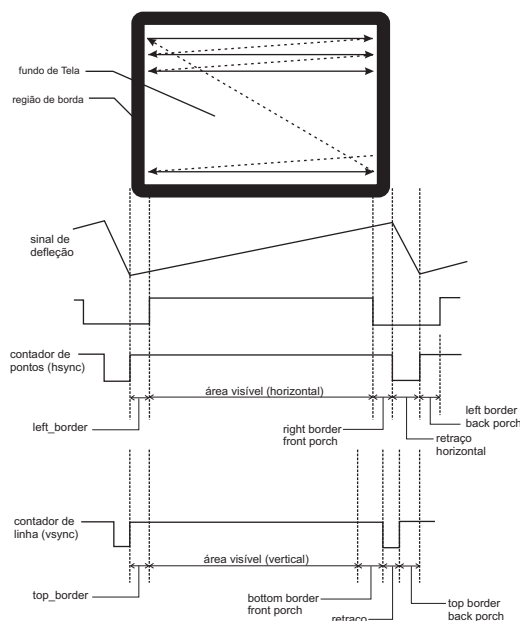
Tabela 1: Classificações de IP-cores.

	<i>Hard Core</i>	<i>Firm Core</i>	<i>Soft Core</i>
Rigidez	A organização é predefinida.	Combinação de código fonte e netlist dependente de tecnologia.	Apresenta um código fonte comportamental independente da tecnologia.
Modelagem	Modelado como um elemento de biblioteca.	Combinação de blocos sintetizáveis fixos. Permite compartilhar recursos com outros cores.	Sintetizável com diversas tecnologias.
Flexibilidade	Não pode ser modificado pelo projetista.	A personalização de funções específicas é dependente da tecnologia.	O projeto pode ser modificado e independe da tecnologia.
Previsibilidade	Temporização é garantida.	Caminhos críticos com temporizações fixas.	A temporização não é garantida, podendo não atender os requisitos do projeto.
Proteção de Propriedade Intelectual	Alta. A descrição normalmente corresponde a um <i>layout</i> .	Média.	Muito pequena. Código fonte aberto.

sob uma superfície coberta por uma camada de material fosforescente, gerando um ponto na tela (*pixel*). Em cada etapa é desenhado um novo ponto na tela até que se realize uma varredura completa na tela. A intensidade do brilho do ponto exibido é controlada pelo nível de voltagem introduzido pelo sinal de vídeo. O controle do caminho percorrido pelo feixe de elétrons é feito através de placas defletoras, horizontal e vertical, que produzem um campo magnético determinando o caminho que o elétron deve percorrer. Estas placas são responsáveis, portanto, pela identificação dos sinais de sincronismo.

Um controlador de vídeo é responsável por gerar os sinais de sincronismo e as saídas de dados para cada *pixel* de forma serial. O sincronismo horizontal (*hsync*) especifica o tempo requerido para percorrer uma linha, o vertical (*vsync*) controla o tempo necessário para percorrer toda a tela [Chu 2008]. Sendo assim, um circuito digital de controle dos sinais de sincronismos requer que sejam definidas, de forma objetiva, a disposição da parte visível da tela no monitor.

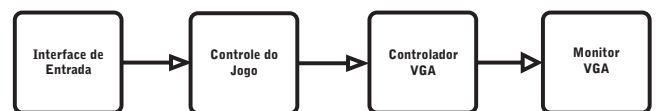
Monitores CRT, durante o sincronismo horizontal, apresentam uma pequena borda preta ao redor da imagem, definidas a partir dos valores de *back porch* (borda da esquerda) e *front porch* (borda da direita). Durante o sincronismo vertical também são verificadas regiões onde o sinal de vídeo deve ser desativado, regiões chamadas de *bottom border* e *top border* [Chu 2008]. Estes dados, além de todos os elementos que compõem uma varredura horizontal e vertical, são descritos na Figura 2. Nela é apresentado um diagrama dos ciclos de sincronismo horizontal e vertical para dispositivos CRT com resolução de 640x480 *pixels*, baseado no padrão VGA.

**Figura 2:** Diagrama de tempo de uma varredura horizontal e vertical

Analisando os sinais de sincronismo de um controlador VGA, existe ainda, além dos sinais de sincronismo *hsync* e *vsync*, o retraço horizontal e vertical. Assim como nas bordas, durante o retraço o sinal de vídeo deve ser desativado. No retraço horizontal, o sinal de vídeo deve ser desativado na região onde o feixe de elétrons retorna para a borda esquerda. No retraço vertical, o sinal deve ser desligado na região em que o feixe retorna para o topo da tela.

3 Desenvolvimento do Jogo

O desenvolvimento do projeto, ilustrado na Figura 3, baseou-se na criação do controlador VGA capaz de gerar sinais para realizar os sincronismos do vídeo através do uso de contadores. Este também serviu de suporte para o controle do posicionamento das imagens exibidas no monitor. O *core* é composto ainda pelo controle de jogo, responsável por responder aos sinais de entrada e processá-los, transformando-os em dados como velocidade e posição.

**Figura 3:** Diagrama de blocos do projeto

A implementação gráfica dos elementos do jogo é composta por *sprites* em duas dimensões, criados para representar imagens que se repetem, como carros e obstáculos. A seguir serão descritos os módulos desenvolvidos para o protótipo do jogo, incluindo os circuitos de controle, como o responsável por manipular os elementos na tela.

3.1 O Controlador VGA

O trabalho aqui descrito, teve origem em uma das avaliações do Estudo Integrado de Sistemas Digitais do curso de Engenharia de Computação. Este componente curricular engloba conceitos de Arquitetura de Computadores. Uma das restrições do problema era o uso do padrão VGA. Com a continuidade do projeto verificou-se que o kit FPGA disponível possui apenas 100 mil portas, um número pequeno para construir um *buffer* de memória. Devido a esta restrição, a interface gráfica do jogo é desenhada *pixel a pixel*, necessitando de pequenos blocos de memória para armazenar os *sprites*.

Basicamente, o controlador VGA é composto por contadores, a partir dos quais é controlada a exibição das imagens na tela. O diagrama do controlador está descrito na Figura 4. Estes sinais são utilizados para controlar o início e o final da área útil da tela, estabelecendo valores de *font porch* e *back porch*, bem como gerar os sinais de sincronismo vertical e horizontal necessários para o funcionamento correto do monitor de acordo com a resolução definida (640x480 *pixels*).

O projeto utilizou um sinal de *clock* de 32,768 MHz, definido pelo kit [Altera 2009], que estabelece a frequência máxima com que os

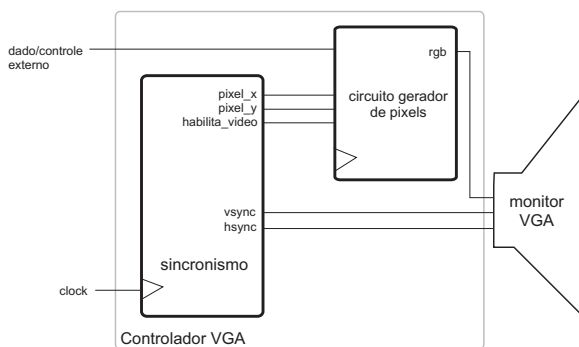


Figura 4: Diagrama do controlador VGA

pixels podem ser enviados ao monitor. Esta frequência é muito alta para os monitores disponíveis, que operam entre 60 Hz e 120 Hz, em sua maioria. Além disso, não é desenhado apenas um ponto na tela. Desde modo, foram projetados dois contadores, um para o sincronismo horizontal e outro para o vertical.

Ambos os contadores têm seus valores iniciados em zero. O contador horizontal é incrementado a cada transição de *clock* do FPGA. Inicialmente é enviado um pulso de sincronismo horizontal (*hsync*), indicando o começo de uma nova linha. Entre a faixa [0-96] do contador o vídeo é desabilitado, para garantir o processo de retraço. O contador vertical incrementa a cada ciclo horizontal completo, ou seja, quando uma linha de *pixels* é completada. Entre [0-2] do contador o vídeo é desativado, para garantir o retraço vertical. Neste intervalo é enviado um pulso de sincronismo vertical (*vsync*), representando um ciclo completo de tela.

3.2 Controlador do Jogo

Inicialmente, foi necessário definir a área do jogo dividindo a tela em regiões onde serão desenhados o carro, a pista, as faixas laterais e centrais, que serão utilizadas para prover a noção de movimento e velocidade. Cada região representa uma condição para a análise da cor a ser exibida. A Figura 5 mostra as regiões e as cores escolhidas para as mesmas.



Figura 5: Área do Jogo

Os valores dos contadores vertical e horizontal são utilizados para representar as coordenadas (x,y) do *pixel* que será impresso. Para exibir imagens na tela, foram feitas comparações com os contadores horizontais e verticais. Quando os valores (x,y) estão dentro de uma das regiões definidas na Figura 5 o módulo comparador identifica a posição e envia a cor relativa à posição do *pixel* que está sendo impresso na tela, desse modo, são exibidas as figuras que compõem os elementos gráficos do jogo.

Os elementos gráficos que apresentam movimento dentro da imagem do jogo foram concebidos como *sprites*, imagens que são armazenadas em matrizes na memória do FPGA. Cada matriz armazena valores binários, sendo que cada bit indica um *pixel*, e os bits que possuem valor igual a 1 representam *pixels* que serão exibidos na tela. Alguns *sprites* são compostos por várias cores, para isso, eles são necessariamente desenvolvidos em matrizes separadas, cada uma relacionada à uma, quando a imagem é exibida na

tela mostra-se a figura composta por todas as matrizes.

Para a exibição de cada *sprite* é necessário definir parâmetros relativos à sua posição. O carro foi definido em duas matrizes de 60 linhas e 120 colunas, totalizando uma região de 720 *pixels*. A primeira matriz armazena o desenho da carenagem do carro, e a segunda guarda os desenhos das rodas e capacete do piloto.

3.2.1 Movimentação do Carro

O efeito de movimento foi desenvolvido alternando as posições das figuras. Foram criados retângulos que representam faixas na lateral e no centro da pista, e a cada um deles foram associadas variáveis contendo os vértices. O movimento é obtido pelo incremento desses vértices de acordo com a velocidade atual, deste modo, a figura é exibida em uma nova posição, até chegar ao limite da tela. Quando o limite é alcançado, o parâmetro recebe o valor inicial, e recomeça o ciclo. A velocidade é estabelecida de acordo com o valor de entrada do botão de velocidade, sendo que, a cada vez que o botão é acionado, a variável de velocidade é incrementada de um valor previamente definido. A Figura 6 mostra o diagrama do controlador do jogo.

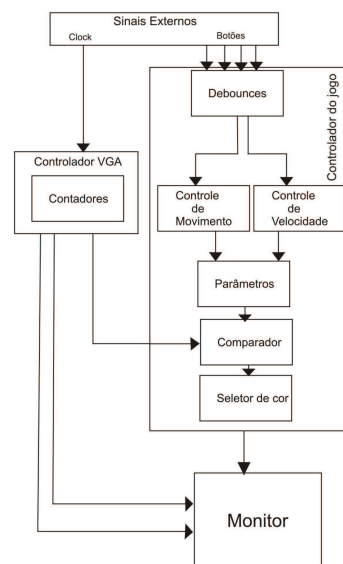


Figura 6: Detalhe do controlador do jogo

Inicialmente, o protótipo utiliza botões como interface com o usuário para a movimentação do carro. Para evitar a ação de ruídos e de repetições quando um botão se mantiver pressionado foi necessária a implementação de *debounces*, que geram sinais estáveis para o módulo do jogo. O circuito do *debouncer* foi desenvolvido como um módulo e embarcado no dispositivo FPGA. A Figura 7 ilustra a ligação dos botões de entrada.

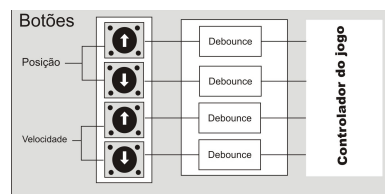


Figura 7: Diagrama da interface de entrada

O carro apresenta uma posição relativa chamada de âncora de *sprite*, que possibilita a movimentação da imagem na tela. Para a movimentação do carro é necessário alterar o valor da âncora, que é modificado realizando-se operações de incremento e decremento, de acordo com o acionamento dos botões. Ao somar o valor da posição inicial, consequentemente o carro é movido para uma posição inferior na tela. Enquanto que, subtraindo valores, o carro é movido na direção da porção superior. Para impedir que o carro

saia da pista, foram definidos parâmetros limitadores de posição, que indicam os limites verticais e horizontais para deslocamento do carro.

3.3 Resultados

A ferramenta utilizada para o desenvolvimento dos circuitos lógicos e o estabelecimento das ligações foi a *Altera Quartus II 9.0 Web Edition* e o dispositivo programado foi o *FPGA Starter Kit Altera (PL1K100A)* [Altera 2009]. A interface de entrada e saída do protótipo foi implementada em uma matriz de contatos, onde foi desenvolvido o circuito que proporciona a interface VGA para a placa FPGA. O padrão VGA utiliza conector D-Sub de quinze pinos, sendo que nem todos são utilizados, apenas os sinais de sincronismo e do RGB.

A matriz de contatos possui também quatro botões do tipo *push-button*, dois deles são utilizados na movimentação lateral, e os demais para o controle de velocidade. O algoritmo do jogo fica armazenado na memória interna do FPGA, onde são executadas todas as rotinas necessárias para a execução do algoritmo implementado. A Figura 8 mostra o circuito montado para a interface do jogo.

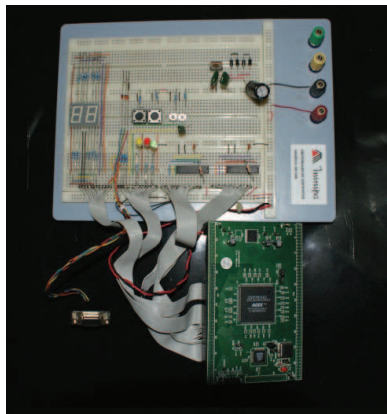


Figura 8: Circuito de interface E/S

O teste inicial do protótipo foi realizado com uma imagem estática da tela do jogo. Em seguida foi adicionado o *sprite* do carro fixo e as faixas foram colocadas em movimento. Depois dessa etapa de teste, foi implementada a interface de entrada testando se havia incremento de posição no carro e da velocidade de passagem das faixas. Por fim, foram testados os limites de posição e velocidade, executando o jogo e acionando os botões para verificar se os limites eram respeitados. A Figura 9 mostra uma foto retirada de um dos testes de funcionamento do protótipo.



Figura 9: Teste do protótipo

4 Conclusão

Este trabalho apresentou a etapa inicial de desenvolvimento de um jogo de carros em FPGA, demonstrando as características dos circuitos desenvolvidos, além da interação dos mesmos para compor

o sistema. Para a exibição das imagens na tela foi implementado um módulo controlador VGA, este gerencia todos os valores de parâmetros que dinamizam a disposição das imagens na tela.

O resultado parcial do projeto compreende o desenvolvimento da interface gráfica do jogo, do controle de velocidade e deslocamento do carro. Através dos sinais gerados pelos botões, a movimentação do carro e variação da velocidade são controladas. Ao pressionar o botão de movimento, o carro desloca-se na direção vertical, para possibilitar o desvio dos obstáculos.

A próxima fase do projeto prevê o desenvolvimento do controle de colisões do carro, a contagem da quilometragem percorrida, o gerenciamento das vidas e a colocação de controles analógicos para a velocidade e deslocamento do carro. Com o intuito de melhorar a aparência e aumentar a sensação de movimento do jogo, pretende-se inserir algumas figuras na grama (área verde da tela). A Figura 10 é um exemplo de como será uma tela da versão final deste jogo.



Figura 10: Perspectivas futuras

Referências

- ALTERA, 2009. Fpga, cpld and asic from altera. <http://www.altera.com>.
- CALAZANS, N. L. V. 1998. *Projeto Lógico Automatizado de Sistemas Digitais Sequenciais*. Master's thesis, Escola de Computação, Universidade Federal do Rio de Janeiro.
- CHU, P. P. 2008. *FPGA Prototyping by Verilog Examples*. WILEY.
- DESCHAMPS, J.-P., BIOUL, G. J. A., AND SUTTER, G. D. 2006. *Synthesis of Arithmetic Circuits - FPGA, ASIC, and Embedded Systems*. John Wiley and Sons, Inc.
- PALMA, J. C., MORAES, F., AND CALAZANS, N. 2001. *Métodos para Desenvolvimento e Distribuição de IP Cores*. Master's thesis, Faculdade de Informática - PUCRS.
- SCHALLER, R. R. 1997. Moore's law: Past, present and future. *IEEE Spectrum* 34, 6 (June), 52–59.