IMPLEMENTAÇÃO DE UM VÍDEO GAME EM FPGA COMO ATIVIDADE DIDÁTICA NO CURSO DE ENGENHARIA ELÉTRICA

Paulo Alexandre Martin[1]; Vanderlei Cunha Parro[2]
[12] Escola de Engenharia Mauá, Engenharia Elétrica
Praça Mauá, 01
09580-900 – São Caetano do Sul – SP

pauloalexandre@maua.br[1], vparro@maua.br[2]

Resumo: *Este artigo apresenta o uso de lógica programável através de FPGAs para o ensino de eletrônica digital. Como atividade didática é proposta a implementação de uma arquitetura que gera um jogo interativo semelhante a um vídeo game em FPGA. Todos os passos para a construção desta arquitetura são apresentados. Em cada passo o aluno desenvolve a criatividade para a construção de blocos digitais, onde as linguagens de descrição de hardware VHDL e AHDL são utilizadas como ferramentas para a síntese de cada bloco. No final os alunos testam os algoritmos do jogo implementado e propõem modificações para a melhoria da resolução gráfica e melhorias nos algoritmos do jogo.*

Palavras-chave: *FPGAs, Vídeo game, Ensino de eletrônica digital.*

1. INTRODUÇÃO

Uma FPGA (*field-programmable gate array*) é um dispositivo lógico programável que permite a implementação de circuitos digitais de grandes proporções. Uma FPGA possui centenas de elementos lógicos e blocos dedicados para a implementação de memórias RAM e memórias ROM (Embedded Array Block). Os elementos lógicos são formados por flip-flops e circuitos combinatórios configuráveis permitindo a implementação de contadores, registradores, circuitos combinatórios e máquinas de estado. Semelhante ao funcionamento de inúmeras PLDs (*Programmable Logic Devices*) interligadas, uma FPGA consegue implementar qualquer circuito digital desde um simples contador até um complexo microprocessador (supondo que o número de elementos lógicos seja suficiente para isto). Graças ao alto grau de integração de componentes que existe atualmente, as FPGAs permitem a implementação de circuitos digitais complexos e sofisticados nos dando um alto poder de processamento. A FPGA utilizada para a implementação da atividade didática proposta é a EPF10K70RC240-2 da família FLEX10K de fabricação ALTERA (www.altera.com). Para maiores informações a respeito das FPGAs da família FLEX10, veja FLEX10K Data Sheet, disponível em: <http://www.altera.com/products/devices/dev-index.jsp>. A “Figura 1” mostra o diagrama de blocos de um FPGA da família FLEX10K.

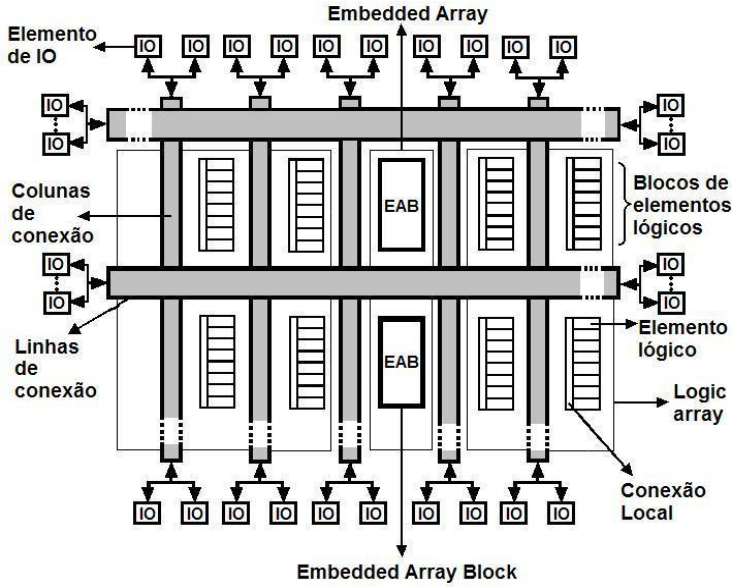

Figura 1 - Diagrama de blocos de uma FPGA da família FLEX10K.

Os elementos lógicos podem ser interligados entre si através das linhas e colunas de conexão formando sistemas digitais mais complexos. Os elementos de IO permitem integração da FPGA com o mundo externo.

Para a implementação do jogo interativo, foi utilizado o kit de desenvolvimento UP2 que possui uma interface VGA que permite que a própria FPGA do kit controle um simples monitor. O kit também possui uma interface PS2 permitindo a conexão de um mouse ou um teclado de forma que os mesmos possam ser usados como joystick. A FPGA utilizada possui todas as características para a implementação de um vídeo game. Os blocos dedicados para a implementação de memórias ROM e memórias RAM são utilizados para a geração de figuras e cenários na tela do jogo. Máquinas de estado implementadas através da linguagem AHDL (*Altera Hardware Description Language*) realizam os algoritmos para a movimentação de figuras.

Nesta atividade didática, o aluno aplica o conhecimento sobre eletrônica digital adquirido em anos anteriores para a implementação do vídeo game proposto. Este projeto consiste em construir um hardware dedicado com memórias, máquinas de estado, contadores, registradores e circuitos combinatórios que integrados conseguem implementar um jogo em FPGA. O principal objetivo desta atividade didática é desenvolver no aluno, a criatividade e capacidade para projetar hardware. Em MARTIN (2007) uma FPGA também é utilizada como ferramenta didática para o ensino de eletrônica digital. Uma implementação de circuitos aritméticos como atividade didática em FPGA é apresentada em CAPPUCCINO *et all* (1999).

2. DIAGRAMA DE BLOCOS DO VÍDEO GAME

O vídeo game implementado pode ser dividido em três blocos principais. O primeiro destes blocos é o gerador de sincronismo VGA que tem como principal função gerar os sinais de sincronismos e cor que controlam o monitor. O segundo bloco é responsável por enviar os dados a serem exibidos no monitor. Estes dados compõem o cenário e os personagens do jogo. Além disto este segundo bloco implementa os algoritmos do jogo de modo que haja uma maior iteração entre o jogo e o usuário e também dá um certo nível de inteligência ao jogo. O terceiro bloco implementa uma interface entre o hardware e o usuário como por exemplo a

leitura das coordenada de um mouse ou a leitura de um teclado de computador. A “Figura 2” apresenta o diagrama de blocos do vídeo game implementado.

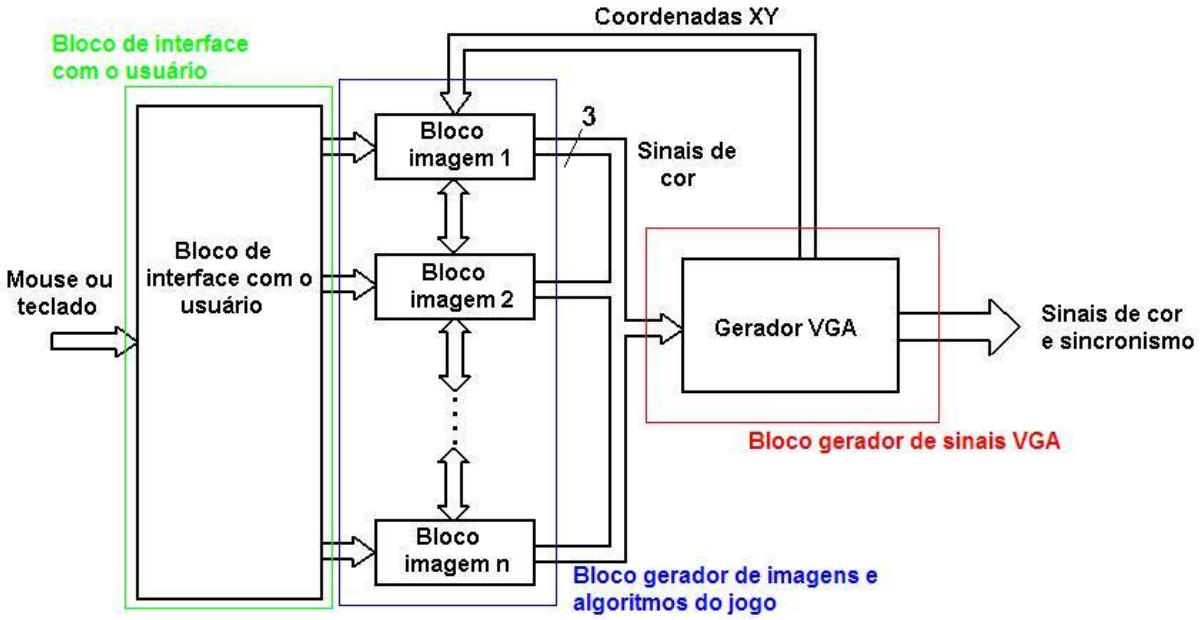

Figura 2 - Diagrama de blocos do vídeo game.

No diagrama de blocos da “Figura 2”, percebe-se que cada bloco possui uma função específica. O projeto de uma arquitetura dedicada que gera um jogo semelhante a um vídeo game torna-se complicado quando tentamos projetar a arquitetura como um todo. Dividindo o projeto em blocos e testando cada bloco individualmente o projeto torna-se mais simples e a localização de erros no projeto torna-se mais fácil. Esta metodologia de “dividir para conquistar” é passada para os alunos como metodologia de projeto de hardware. Cada bloco que compõe a arquitetura foi projetado, simulado e testado individualmente. Quando todos os blocos estavam funcionando, os mesmos foram integrados formando o jogo interativo em FPGA.

3. BLOCO GERADOR DE SINAIS VGA

Este bloco é responsável por gerar os sinais de cor e sincronismo VGA. O mesmo possui três entradas para os sinais de cor (vermelho, verde e azul) que vem do bloco gerador de imagens e algoritmos do jogo. Ao receber estes sinais de cor, o bloco gerador de sinais VGA envia estes sinais de cor junto com os sinais de sincronismo para o monitor. Este bloco utiliza um clock de 25 MHz e permite gerar imagens de resolução máxima de 640 colunas por 480 linhas sendo que cada ponto na tela pode assumir oito cores. Os sinais de sincronismo vertical e horizontal são sinais digitais enquanto que os sinais de cores que vão para o monitor podem ser sinais analógicos. Para cada nível de sinal de cor vermelho temos uma tonalidade diferente de vermelho. Portanto se para cada sinal de cor (vermelho, verde e azul) utilizarmos um conversor DA de 8 bits, podemos gerar 256^3 cores diferentes. Como o kit UP2 não possui conversores DA e utiliza somente diodos para adaptar o nível lógico da FPGA ao nível do monitor, pode-se gerar apenas oito cores. O sincronismo VGA é feito de acordo com o mostrado na “Figura 3”.

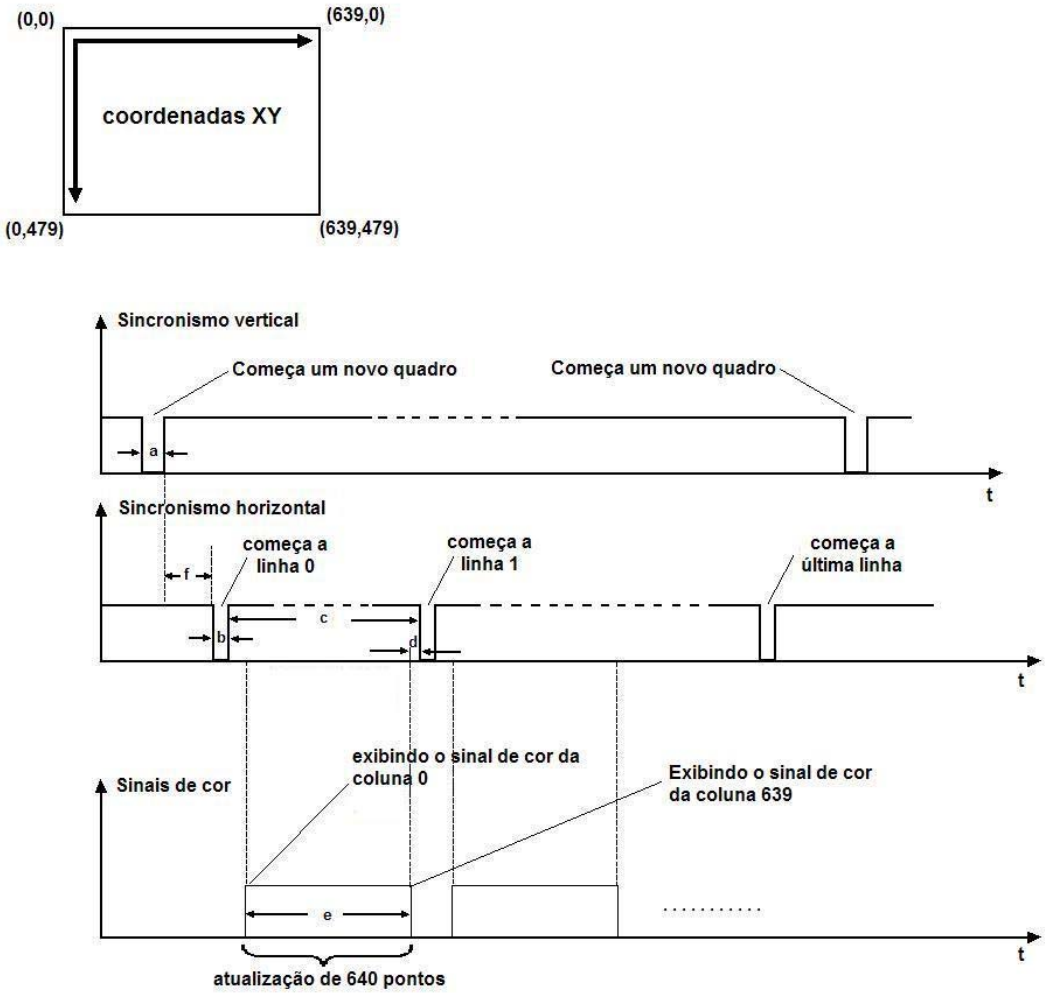

Figura 3 – Coordenadas do monitor VGA e carta de tempo dos sinais de cor e sincronismo.

A “Tabela 1” mostra os parâmetros dos sinais de cor e sincronismo.

Tabela 1 – Parâmetros dos sinais de cor e sincronismo.

Parâmetro	Valor
a	64 μs
b	3,77 μs
c	28 μs
d	0,94 μs
e	25,17 μs
f	1,02 ms

Conforme apresentado na “Figura 3”, os sinais de sincronismo vertical e horizontal são sinais digitais e um pulso de nível lógico baixo informa ao monitor que um novo quadro se inicia (se for o sinal de sincronismo vertical) ou que uma nova linha de inicia (se for o sinal de sincronismo horizontal). Após o pulso de nível lógico baixo no sincronismo horizontal, os sinais de cor da respectiva linha são enviados através de 3 bits denominados RGB (Red, Green e Blue). Como neste caso utilizou-se somente 3 bits para informar os sinais RGB e o kit não possui conversores DA, pode-se apenas gerar oito cores no monitor.

A ferramenta utilizada para a construção do bloco gerador de sinais VGA foi a linguagem de descrição de hardware VHDL. Esta linguagem de descrição de hardware é baseada na linguagem PASCAL. Através desta linguagem, pode-se projetar e sintetizar hardwares através de uma linguagem de programação ao invés de um diagrama esquemático. Quando pronto, o compilador do fabricante da FPGA compila o código digitado e permite a gravação da FPGA. O código apresentado na "Figura 4" implementa o bloco gerador de sinais VGA.

```
ENTITY VGA IS
PORT(CLK,R_IN,G_IN,B_IN: IN BIT;
      VS,HS,R_OUT,G_OUT,B_OUT: OUT BIT;
      X,Y: OUT INTEGER RANGE 0 TO 1023);
END VGA;

ARCHITECTURE VGA_BEHAVIOR OF VGA IS
BEGIN
PROCESS(CLK)
VARIABLE X_COUNT: INTEGER RANGE 0 TO 1023;
VARIABLE Y_COUNT: INTEGER RANGE 0 TO 1023;
BEGIN
IF (CLK'EVENT) AND (CLK='1') THEN
        X_COUNT:=X_COUNT+1;
        IF (X_COUNT>=659) AND (X_COUNT<=755) THEN
                HS<='0';
        ELSIF (X_COUNT=799) THEN
                X_COUNT:=0;
                Y_COUNT:=Y_COUNT+1;
                IF (Y_COUNT=525) THEN
                        Y_COUNT:=0;
                END IF;
        ELSE
                HS<='1';
        END IF;
        IF (Y_COUNT>=493) AND (Y_COUNT<=494) THEN
                VS<='0';
        ELSE
                VS<='1';
        END IF;
        IF (Y_COUNT<=479) AND (X_COUNT<=639) THEN
                R_OUT<=R_IN;
                G_OUT<=G_IN;
                B_OUT<=B_IN;
                X<=X_COUNT;
                Y<=Y_COUNT;
        ELSE
                R_OUT<='0';
                G_OUT<='0';
                B_OUT<='0';
                X<=1023;
                Y<=1023;
        END IF;
END IF;
END PROCESS;
END VGA_BEHAVIOR;
```

Figura 4 – Código em VHDL que implementa o bloco gerador de sinais VGA.

As referencias AMORE (2005), PERRY (1999) e NAVABI (1998) apresentam uma ótima introdução da linguagem VHDL e exemplos didáticos.

4. BLOCO GERADOR DE IMAGENS E DE ALGORITMOS DO JOGO

Este bloco é responsável pelo armazenamento das imagens que compõem o cenário do jogo e os algoritmos do jogo. Conforme apresentado no diagrama de blocos da "Figura 2", os blocos geradores de imagens recebem as coordenadas da tela que estão sendo atualizadas no momento. Como o gerador de sincronismo VGA gera os sinais para o monitor, este mesmo bloco sabe qual ponto em uma determinada coordenada esta sendo atualizado no exato momento. Se o jogo deve desenhar um quadrado de 200 pontos de lado nas coordenadas (50,100), o bloco gerador de imagens deve comparar as coordenadas (vide "Figura 3" onde são mostradas as coordenadas do monitor) que vem do bloco gerador de VGA com as constantes, 50, 250, 100 e 300. Se a coordenada X estiver entre 50 e 250 e a coordenada Y estiver entre 100 e 300, então o bloco gerador de imagens deve enviar os sinais de cores para o gerador de sinais VGA. O código da "Figura 5" mostra o código em VHDL que gera o bloco que implementa o desenho de um quadrado nas coordenadas (50,100).

```
ENTITY  SQUARE IS
PORT(X: IN INTEGER RANGE 0 TO 640;
    Y: IN INTEGER RANGE 0 TO 480;
    B: OUT BIT);
END SQUARE;

ARCHITECTURE SQUARE_BEHAVIOR OF SQUARE IS
BEGIN
PROCESS(X,Y)
BEGIN
IF (X>=50) AND (X<=250) AND (Y>=100) AND (Y<=300) THEN
        B<='1';
ELSE
        B<='0';
END IF;
END PROCESS;
END SQUARE_BEHAVIOR;
```

Figura 5 – Código em VHDL que gera um quadrado.

Como resultado, temos um quadrado fixo nas coordenadas (50,100) conforme é apresentado na "Figura 6".

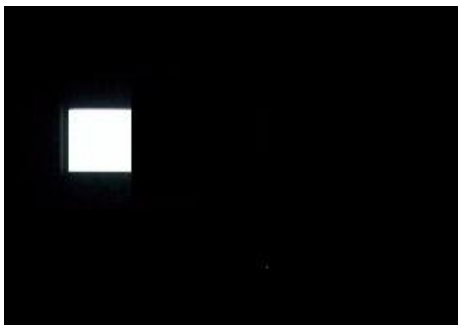

Figura 6 – Quadrado gerado pelo código em VHDL da "Figura 5".

Modificando o código da “Figura 6”, pode-se criar um bloco gerador de imagens que implementa um quadrado que pode ser movimentado. Este código é apresentado na “Figura 7”.

```
ENTITY SQUARE2 IS
PORT(X,X_POS: IN INTEGER RANGE 0 TO 640;
   Y,Y_POS: IN INTEGER RANGE 0 TO 480;
   B: OUT BIT);
END SQUARE2;

ARCHITECTURE SQUARE2_BEHAVIOR OF SQUARE2 IS
BEGIN
PROCESS(X,X_POS,Y,Y_POS)
VARIABLE XAUX: INTEGER RANGE 0 TO 640;
VARIABLE YAUX: INTEGER RANGE 0 TO 480;
BEGIN
XAUX:=X_POS+100;
YAUX:=Y_POS+100;
IF (X>=X_POS) AND (X<=XAUX) AND (Y>=Y_POS) AND (Y<=YAUX) THEN
        B<='1';
ELSE
        B<='0';
END IF;
END PROCESS;
END SQUARE2_BEHAVIOR;
```

Figura 7 – Código que implementa um quadrado que pode ser movimentado na tela.

Isto é feito comparando uma entrada do bloco na estrutura IF conforme pode ser visto na “Figura 7”. Se forem conectados contadores a essas entradas (X_POS e Y_POS), o quadro irá se movimentar na tela. A velocidade de movimento e a direção do movimento podem ser controladas pela freqüência do clock dos contadores e pela criação de uma entrada nos contadores externos ao bloco que controla se o contador deve contar no sentido crescente ou no sentido decrescente. Outra maneira de criar figuras é através de memórias ROM implementadas diretamente na FPGA. Através de memórias ROM pode-se criar imagens mais sofisticadas e com maior resolução. Neste método de implementação, uma memória ROM irá armazenar a imagem. Os bits de endereçamento da memória são separados em linha e coluna e a saída da memória irá informar a cor do ponto para um determinado valor de linha e coluna (endereçamento da memória). Para esta implementação, um bloco com funcionamento semelhante ao da “Figura 7” é usado para endereçar a memória e um bit de habilitação permite a passagem da informação da cor para o bloco gerador se sinais VGA conforme é apresentado na “Figura 8”.

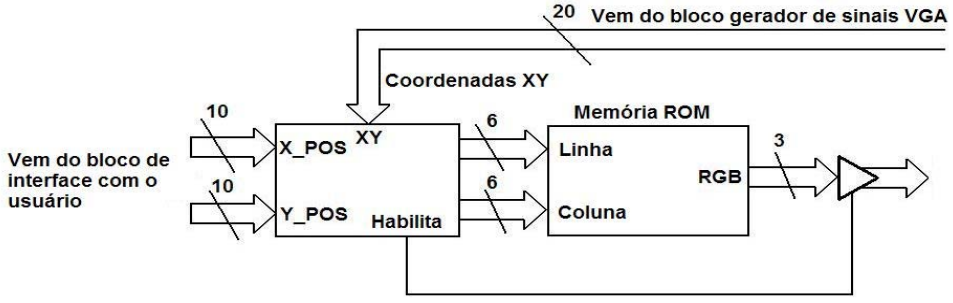

Figura 8 – Diagrama de blocos para a geração de imagens através da memória ROM.

A “Figura 9” mostra o resultado obtido com esta metodologia de projeto dos blocos geradores de imagens.

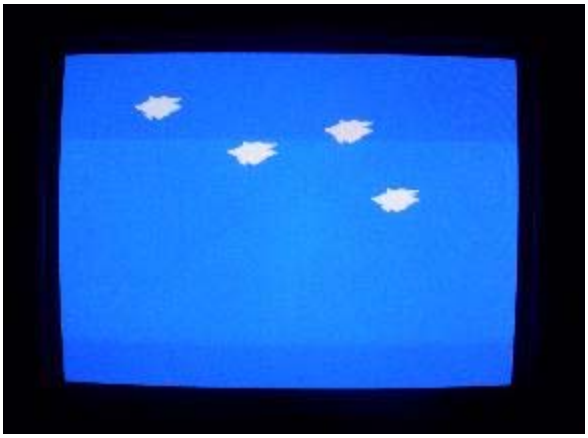

Figura 9 – Nuvens geradas através da memória ROM.

O bloco que habilita a saída da memória também permite que a mesma imagem seja gerada em mais de uma região da tela. Isto é útil, pois não é necessário que outra memória ROM seja construída economizando espaço na FPGA. Na “Figura 9” todas as nuvens são iguais pois foram geradas a partir da mesma memória ROM. O bloco que habilita a saída da memória ROM, faz esta habilitação em diferentes coordenadas, desenhando quatro nuvens na tela. Alguns blocos geradores de imagens permitem troca de informações entre si. Desta forma é possível implementar os algoritmos do jogo. Esta troca de informações é controlada por uma unidade de controle implementada através de uma máquina de estados. A referência PARHAMI (2007) apresenta uma excelente teoria a respeito de unidades de controle.

5. BLOCO DE INTERFACE COM O USUÁRIO

O bloco de interface com o usuário permite a iteração no jogo através de um mouse ou um teclado de computador conectado a uma interface PS2 no kit. Este bloco foi construído através da linguagem de descrição de hardware VHDL. Em alguns trabalhos, optou-se por utilizar o teclado de computador devido ao grande número de teclas, sendo que cada uma implementa uma determinada ação no jogo. O teclado comunica-se com o kit didático através de uma comunicação serial síncrona descrita na carta de tempos da “Figura 10”.

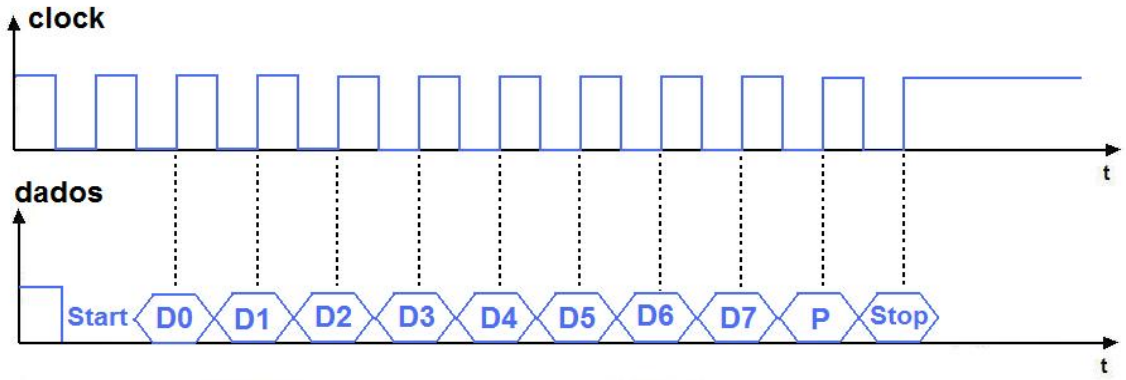

Figura 10 – Comunicação serial do teclado.

Na carta de tempos da “Figura 10”, os bits são enviados do teclado para o bloco de interface com o usuário através de uma comunicação serial síncrona. Esta comunicação é iniciada com um bit de início (Start) que sempre esta em nível lógico zero. Em seguida são enviados os oito bits de dados que indicam a tecla pressionado ou liberada (a comunicação é feita começando-se com o bit menos significativo). Após a transmissão do bit 7 (mais significativo) é enviado o bit de paridade (bit P) e em seguida o bit de parada (Stop bit). No caso do teclado o bit de parada é sempre nível lógico alto. Quando uma tecla é pressionada no teclado, o mesmo envia um determinado byte para o bloco de interface com o usuário. Quando esta mesma tecla é liberada, o teclado envia um código de 16 bits (2 bytes transmitidos em 2 comunicações seriais), sendo que este mesmo código indica que a tecla foi liberada. Isto é apresentado com mais detalhes em HAMBLEN e FURMAN (2001). O código em VHDL da “Figura 11” apresenta o bloco que recebe dados do teclado serialmente. Este bloco é uma parte do bloco de interface com o usuário.

```
ENTITY TECLADO IS
PORT (DATA,CLK: IN BIT;
    T0,T1,T2,T3,T4,T5,T6,T7,P: OUT BIT);
END TECLADO;

ARCHITECTURE TECLADO_BEHAVIOR OF TECLADO IS
BEGIN
PROCESS(CLK)
VARIABLE CONTADOR: INTEGER RANGE 0 TO 15;
BEGIN
IF (CLK'EVENT) AND (CLK='1') THEN
     CONTADOR:=CONTADOR+1;
     IF (CONTADOR=1) AND (DATA='1') THEN
               CONTADOR:=0;
     ELSIF (CONTADOR=2) THEN
               T0<=DATA;
               P<='0';
     ELSIF (CONTADOR=3) THEN
               T1<=DATA;
     ELSIF (CONTADOR=4) THEN
               T2<=DATA;
     ELSIF (CONTADOR=5) THEN
               T3<=DATA;
     ELSIF (CONTADOR=6) THEN
               T4<=DATA;
     ELSIF (CONTADOR=7) THEN
               T5<=DATA;
```

```
    ELSIF (CONTADOR=8) THEN
            T6<=DATA;
    ELSIF (CONTADOR=9) THEN
            T7<=DATA;
    ELSIF (CONTADOR=11) THEN
            CONTADOR:=0;
            P<='1';
    END IF;
END IF;
END PROCESS;
END TECLADO_BEHAVIOR;
```

Figura 11 – Código em VHDL que implementa a recepção de dados vindos do teclado.

6. RESULTADOS OBTIDOS

As figuras a seguir mostram os resultados obtidos com o projetos realizados pelos alunos.

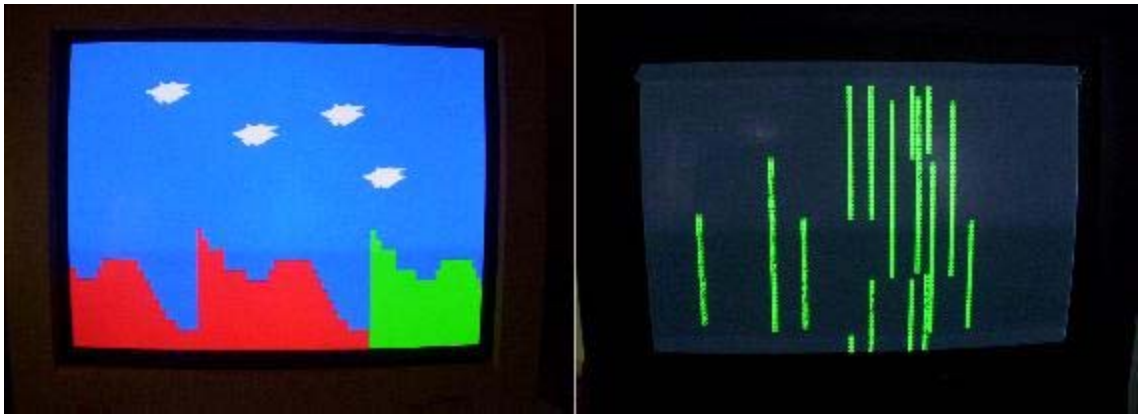

Figura 12 – À esquerda, paisagem com montanha e nuvens e à direita, texto em deslocamento.

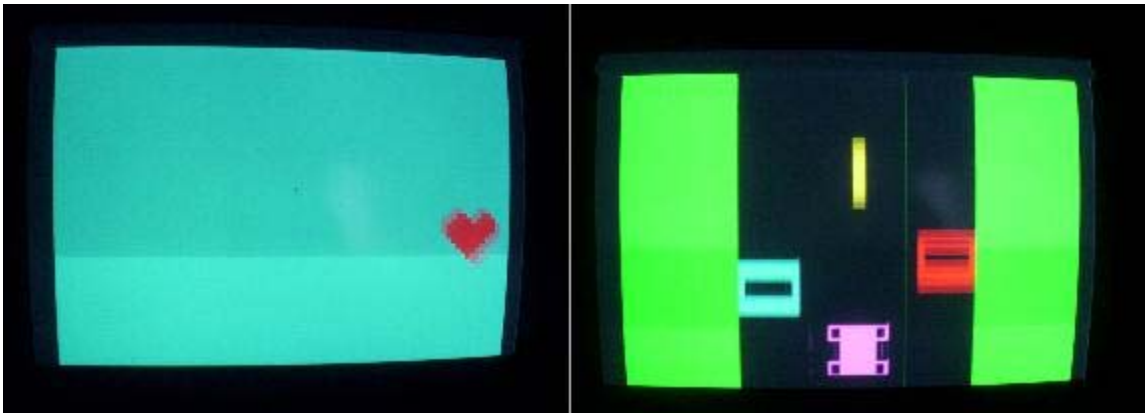

Figura 13 – À esquerda, coração em movimento e à direita, corrida de carros.

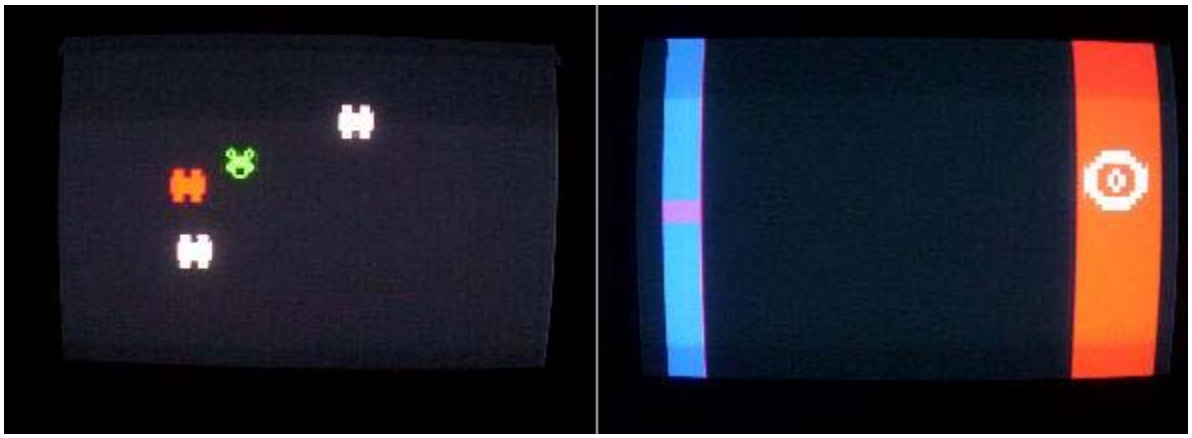

Figura 14 – À esquerda, sapo que atravessa a rua e à direita, tiro ao alvo.

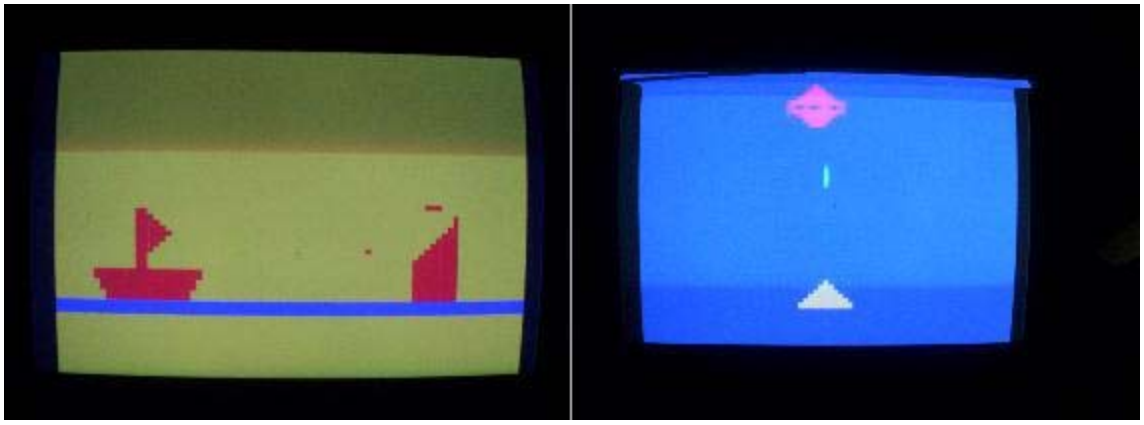

Figura 15 – À esquerda, barco e à direita, aeronave.

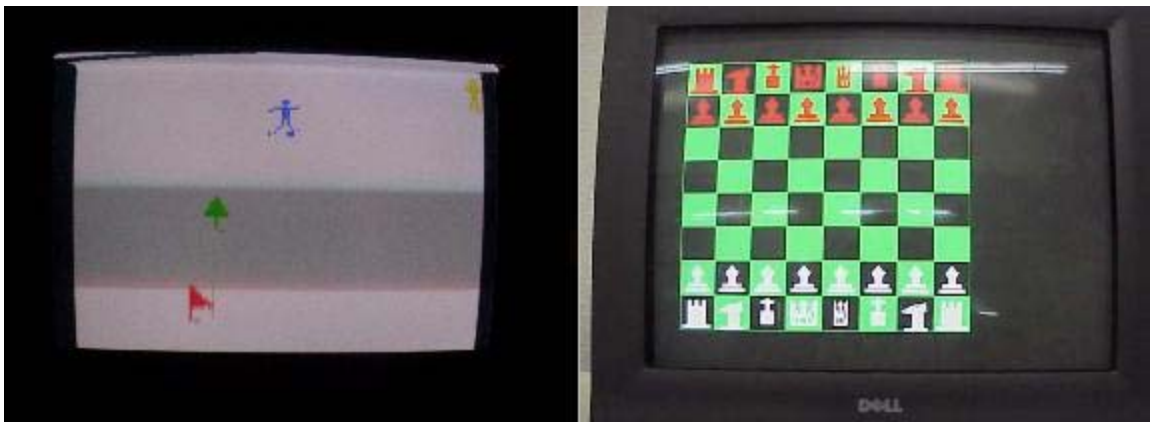

Figura 16 – À esquerda, esquiador e à direita, xadrez.

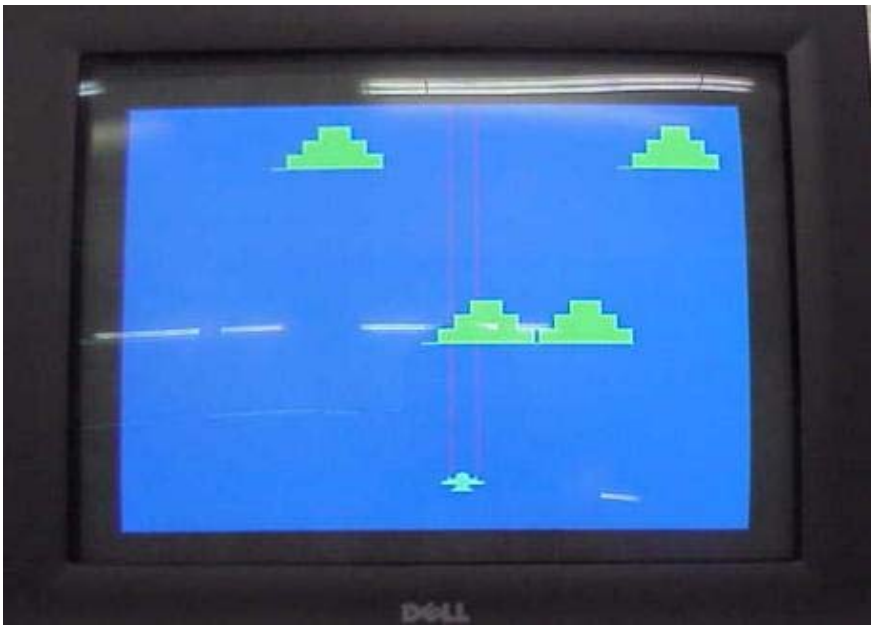

Figura 17 – Aeronave.

7. METODOLOGIA DE PROJETO E AVALIAÇÃO

Este projeto foi implementado no quarto ano de engenharia elétrica como uma atividade de laboratório. A duração total para a finalização do projeto foi de 3 meses sendo que o trabalho foi realizado em grupo com no máximo três alunos. Para a implementação deste projeto, os alunos utilizaram uma metodologia de “dividir para conquistar”. Todo o projeto foi dividido em sub-blocos, sendo que cada sub-bloco foi simulado e testado individualmente. Quando todos os blocos funcionaram corretamente, os mesmos foram integrados na arquitetura que implementa o vídeo game. Neste projeto os alunos colocaram em prática todo o conhecimento visto em circuitos digitais ao longo dos primeiros anos. Dentre estes conhecimentos podemos citar:

- **Projeto de máquinas de estado**: Alguns algoritmos no jogo são implementados através de hardwares dedicados em máquinas de estado.

- **Uso da linguagem VHDL**: Todos os blocos foram implementados através da linguagem VHDL com exceção das máquinas de estados que foram implementadas através da linguagem AHDL.

- **Implementação de arquiteturas dedicadas**: Hardwares dedicados são ferramentas importantes no projeto de sistemas digitais que utilizam processamento paralelo. Através desta atividade os alunos implementaram sistemas digitais dedicados que realizam operações em paralelo como por exemplo o bloco que recebe bytes do teclado operando em paralelo com os blocos geradores de imagens e algoritmos do jogo.

- **Otimização de hardware**: Outro ponto importante desta atividade foi o aprendizado de otimização do uso da FPGA, com o qual os alunos utilizam recursos como memórias ROM e redundâncias nos cenários do jogo para minimizar a ocupação da FPGA.

Na avaliação, foram dados maiores pesos às melhorias e aperfeiçoamentos realizadas no projeto, como por exemplo um hardware com um certo nível de inteligência que modifica a estratégia do jogo de acordo com as ações do usuário. Projetos mais complexos, porém com otimizações que levam a uma menor ocupação do dispositivo lógico programável obtiveram melhores notas por mostrarem melhoria com relação a ocupação do dispositivo. Isto é importante nos dias de hoje, pois como profissionais, os alunos deverão desenvolver projetos que tenham uma ótima confiabilidade, mais recursos com relação ao produto do concorrente e também economizando componentes. A nota final é a somatória de 5 avaliações:

- Interação com o usuário (1,5 pontos)
- Qualidade dos gráficos (2,0 pontos)
- Originalidade (1,0 ponto)
- Melhorias e aperfeiçoamentos (3,0 pontos)
- Minimização do uso da FPGA (2,5 pontos)

Totalizando 10 pontos que é a nota máxima do projeto. Ao enfatizar mais os dois últimos itens (melhorias, aperfeiçoamentos e minimização), os alunos procuram desenvolver um hardware com melhorias em relação aos projetos dos anos anteriores. Desta forma, os alunos desenvolvem sua criatividade no projeto de hardwares, algo muito importante dos dias de hoje, pois com a popularização das FPGAs, muitos sistemas que utilizam microprocessadores passam a utilizar FPGAs devido ao fato que as FPGAs possuem uma maior capacidade de processamento.

8. CONSIDERAÇÕES FINAIS

Esta atividade contribui de maneira significativa para o ensino de eletrônica digital em um nível mais avançado. Através de FPGAs pode-se propor trabalhos que seriam impossíveis de se implementados através de componentes discretos. Nesta atividade, desenvolveu-se a habilidade e visão de projeto de hardware dedicado, algo muito importante na engenharia atual. Muitos sistemas exigem uma quantidade de processamento muito grande onde seria impossível implementar uma solução com um microcontrolador ou um microprocessador. São nestes casos que utilizam-se FPGAs. Como exemplo disto, podemos citar o bloco gerador de sinais VGA que deve atualizar os pixels do monitor a uma taxa de 25 milhões de pixels por segundo. Uma solução com microprocesssador ficaria impossível ou muito cara, pois seria necessário um microprocessador com freqüência da ordem de gigahertz, pois seriam necessárias várias instruções em linguagem de máquina para processar o pixel e exibi-lo na tela. Com o uso de FPGAs, implementamos hardware dedicados que operam em freqüências elevadas nos dando um alto poder de processamento. Esta atividade visou ensinar projeto de hardwares dedicados de uma maneira inovadora e criativa implementando um vídeo game em FPGA. Além do ensino do projeto de hardware, esta atividade contribuiu para um maior conhecimento por parte do aluno com relação à linguagem VHDL e a interface de diversos sistemas como o monitor, o mouse e o teclado. Percebeu-se um grande aproveitamento do curso por parte dos alunos, preparando-os melhor para o mercado de trabalho onde o uso das FPGAs esta cada vez mais presente.

Agradecimentos

Especiais agradecimentos à Escola de Engenharia Mauá pelos kits didáticos para FPGAs e softwares de desenvolvimento.

REFERÊNCIAS BIBLIOGRÁFICAS

AMORE, R. **VHDL: Descrição e Síntese de Circuitos Digitais**. Rio de Janeiro: LTC, 2005.

CAPPUCCINO, G.; COCORULLO, G. ;CORSONELLO, P; PERRI, S. **Educational Design of High-Performance Arithmetic Circuits on FPGA**. IEEE Transaction on Education. v. 42, n. 4, p. 366-393. 1999.

FLEX10K Data Sheet. Disponível em: <http://www.altera.com/products/devices/dev-index.jsp>. Acesso em 07 mar. 2008.

HAMBLEN. J. O. e FURMAN, M. D. **Rapid Prototyping of Digital Systems: A Tutorial Approach**. Boston: Kluwer Academic Publishers, 2001.

MARTIN, P. A. **Uso de FPGAs para o ensino de arquitetura de microcontroladores**. In: Congresso Brasileiro de Educação em Engenharia, 2007, Curitiba.

NAVABI, Z**. VHDL: Analysis and Modeling of Digital Systems.** New York: McGraw-Hill, 1998.

PARHAMI, B. **Arquitetura de Computadores**. São Paulo: McGraw-Hill, 2007.

PERRY, D. L. **VHDL.** New York: McGraw-Hill, 1999.

IMPLEMENTATION OF A VIDEO GAME IN FPGA AS DIDACTIC ACTIVITY IN THE ELECTRICAL ENGINEERING COURSE

Abstract: *This paper presents the use of programmable logic through FPGAs to the digital eletronic teaching. As didatic activity it is proposed the implementation of an architecture that generates an iteractive game similar to a video game in a FPGA. All steps to the construction of this architecture are presented. At each step the student develop the criativity to the construction of digital blocks, where the hardware description languages VHDL and AHDL are used as tools to the synthesis of each block. At the end the students test the implemented game algorithms and propose modifications to improve the graphics resolution and to improve the game algorithms.*

Key-words: *FPGAs, video game, digital electronic teaching.*