COBENGE 2014
ENGENHARIA: Múltiplos saberes e atuações
16 a 19 de setembro | Juiz de Fora - MG

APLICAÇÃO DE PROCESSAMENTO DIGITAL DE IMAGEM COMO AUXÍLIO AO ENSINO DE LÓGICA RECONFIGURÁVEL

Allan Palmerio Vieira – allanpvieira@hotmail.com
Universidade Tecnológica Federal do Paraná (UTFPR)
Av. Sete de Setembro, 3165.
CEP 80230-901

Carlos Raimundo Erig Lima – erig@utfpr.edu.br
Universidade Tecnológica Federal do Paraná (UTFPR)
Av. Sete de Setembro, 3165.
CEP 80230-901

Resumo: *Este trabalho tem como objetivo propor o ensinamento de conceitos de lógica reconfigurável a partir de uma aplicação de processamento de imagens. Tal aplicação utiliza conceitos de paralelismo, comunicação serial, VHDL e diagrama de blocos .*

Palavras-chave: *FPGA, Lógica Reconfigurável, VHDL, Processamento paralelo*

1. INTRODUÇÃO

A Universidade Tecnológica Federal do Paraná (UTFPR) fornece ao público brasileiro e a comunidade internacional vários cursos de engenharia. Hoje, são ofertados cursos de Engenharia Elétrica, Engenharia Mecânica, Engenharia de Controle e Automação e Engenharia da Computação. Nestes cursos há a demanda pelo ensino de sistemas digitais como elementos centrais na formação de competências esperadas nos egressos das engenharias. Muitos dos experimentos propostos para exploração de todos os conhecimentos apresentados são abordados na forma de projeto baseados em plataformas de desenvolvimento baseadas em FPGA (*Field Programmable Gate Array*), utilizando diferentes ferramentas de auxílio ao projeto. Exemplos de disciplinas que usam esta abordagem são: Circuitos Digitais, Microcontroladores, Arquitetura e Organização de Computadores, Controle 2 e Lógica Reconfigurável, entre outras.

Como ensino de FPGA e lógica reconfigurável envolve vários conceitos complexos, muitas vezes de difícil aprendizado, uma abordagem adequada para o ensino é partir de projetos de aplicações que envolvam os principais conceitos das disciplinas. Ao interagir com um projeto funcional da disciplina, o aluno tem mais possibilidade de entender onde está inserido cada conceito e tem espaço para modificá-lo e melhorá-lo, na medida em que novas informações vão sendo apresentadas.

As FPGA são uma poderosa ferramenta que são bastante utilizadas para aplicações de processamento de imagens. Neste trabalho é apresentada uma aplicação baseada em FPGA

que realiza a operação de convolução de uma máscara sobre uma imagem. Essa aplicação demanda o domínio de muitas sub-blocos interessantes, tal como interface de comunicação serial, interface de vídeo, manipulação matemática, projeto de sistemas usando código VHDL *(Very High Speed Integrated Circuits Hardware Description Language)* e projeto de sistemas representados usando diagramas esquemáticos. São discutidos os principais conceitos do projeto na medida em que vão sendo explorados os pontos de oportunidade de aprendizado para o estudante.

2. RECURSOS UTILIZADOS

2.1.Placa Cyclone II

Quando um aluno está aprendendo a usar alguma plataforma de *hardware*, o melhor cenário é aquele em que ele tenha acesso ao recurso real ao invés de ater-se somente a simuladores. Uma placa de fácil manipulação por parte dos alunos é a Cyclone II FPGA Starter Board, da Altera, apresentada na Figura 1. Ela possui várias interfaces de comunicação e vários componentes que permitem a criação de vários tipos aplicações. Dentre esses componentes pode-se citar: suporte a cartão SD, saída VGA, LEDs verdes e vermelhos, Display 7 segmentos, entre outros.

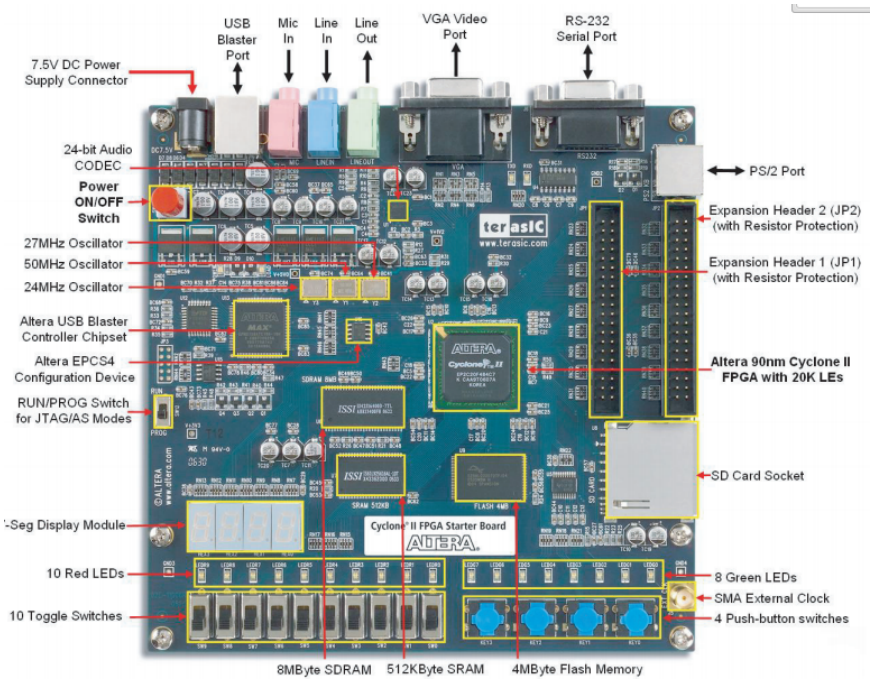

Figura 1 - Placa Cyclone II Starter Board

O uso dessa placa em particular não é ponto necessário para implementação do projeto em questão, porém algumas adaptações ao projeto original serão necessárias principalmente pelo fato de cada placa FPGA ter um padrão de pinagem distinto. Além disso, essa placa é

recomendada para estudantes pelo fato de possuir muitos componentes agregados a mesma. Isto reduz a demanda pela comunicação com outros periféricos.

2.2. QUARTUS II

A utilização de uma ferramenta de auxílio ao projeto facilita muito a implementação de qualquer sistema. No caso de projetos de FPGA, a IDE QUARTUS II é uma ferramenta completa para implementação e testes de projetos, sendo utilizada de forma generalizada pelos cursos de Engenharia da UTFPR (ALTERA, 2014).

A plataforma possui várias funcionalidades que podem ser exploradas para o aprendizado dos conceitos. Através dela é possível de se projetar um sistema através de diagramas de blocos, máquinas de estados, códigos Verilog e VHDL e até mesmo de modo híbrido utilizando-se muitas ou todas estas formas de projeto.

Além de uma vasta quantidade de funcionalidades que auxiliam na elaboração do projeto, existem também funcionalidades para execução e *debug* deste projeto. A ferramenta de desenvolvimento gera um arquivo de simulação que permite o teste do projeto ou, caso necessário, apenas algumas partes deste. A plataforma permite ainda que se grave o projeto na própria FPGA. A partir da configuração de arquivo de mapeamento, o projetista/aluno consegue ainda ter controle total sobre os pinos ativados na placa a ser testada. A Figura 2 apresenta uma tela da interface gráfica do Quartus II representando uma entrada de projeto usando representação em esquemático.

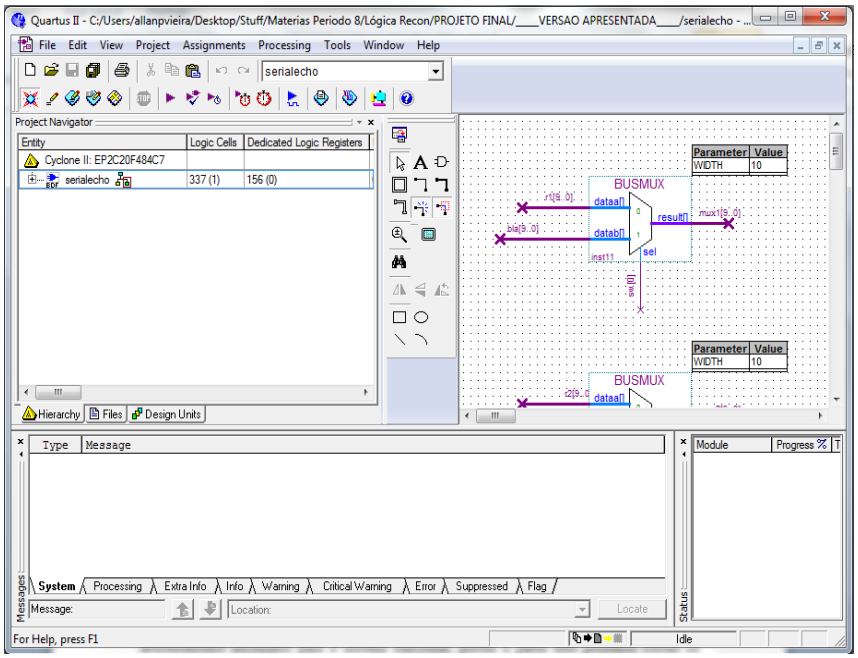

Figura 2 - Interface gráfica do IDE Quartus II

2.1.MATLAB

O projeto aqui detalhado envia e recebe dados através da interface serial de um comptador. No caso, os dados representam pixels de uma imagem. O MATLAB (http://www.mathworks.com/products/matlab/) é uma ferramenta bastante útil para converter qualquer imagem para o formato esperado pelo programa da FPGA. De igual maneira realiza a operação inversa: quando os dados são processados e retransmitidos pela FPGA, o MATALB é capaz de interpretar tais dados na forma de imagem – a qual pode ser visualizada dentro do próprio programa. Além das características citadas, o MATLAB consegue ainda enviar e receber dados via interface serial. Esta ferramente não é absolutamente necessária para o funcionamento do projeto e outras poderiam substituí-la, porém suas funcionalidades são bastante úteis para facilitar o teste do sistema projetado.

3. COMUNICAÇÃO SERIAL

3.1.Projetos de Terceiros

O objetivo de uma aplicação real de processamento de imagens é proporcionar ao aluno um melhor entendimento sobre realização de projetos de lógica reconfigurável. A realização da comunicação serial entre a placa FPGA e o computador poderia por si só ser um projeto desafiador para o aluno iniciante, por isso nesse projeto a parte da comunicação serial é realizada através da importação de blocos de projeto de terceiros.

A utilização de recursos de outros projetos é uma ferramenta importante na elaboração de projetos, pois muitas vezes diferentes aplicações contêm várias partes que são idênticas ou muito parecidas, de tal forma que a reimplementação da mesma não agrega nenhum valor ao projeto proposto. Tal processo de reutilização de blocos de projetos deve ser uma funcionalidade disponível no ambiente de projeto, o qual é imensamente facilitado pela IDE QUARTUS II.

Na internet é possível se encontrar repositórios contendo os mais variados tipos de projetos – processamento de imagens, jogos, manipulação de sons, projetos envolvendo malhas de controle, etc. Um exemplo desse tipo de repositório é o site *fpga4fun.com*, onde podemos encontrar uma parte do bloco de serial necessário este projeto. No site *sparxeng.com* podemos encontrar um pequeno tutorial sobre como criar um eco de serial, que após o recebimento de 10 bytes, reenvia-os em ordem invertida. Esse projeto é usado como base da comunicação serial no projeto proposto, ilustrado na figura 3 (FOHL, 2013).

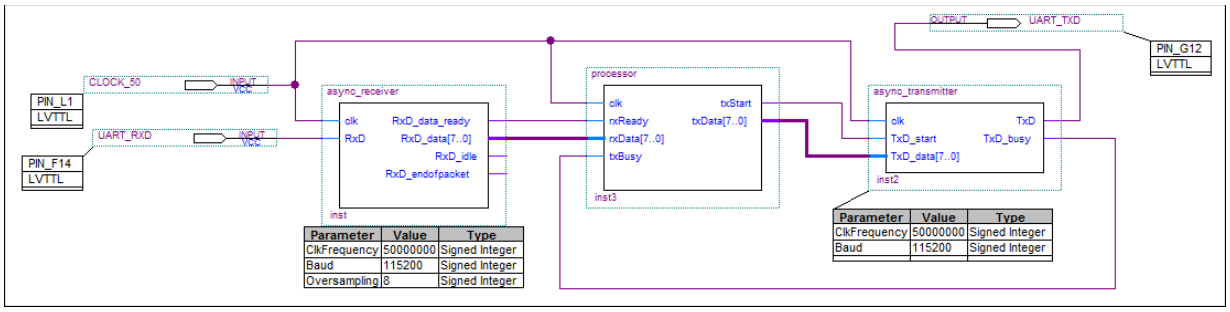

Figura 3 - Diagrama geral do Projeto de eco de serial importado ao projeto

3.1.Programas para Teste

Os blocos para comunicação serial do exemplo mencionado podem ser testados com alguns programas que realizam tal tarefa tais como o *TeraTerm*, o *RealmTerm*, ou mesmo o *MATLAB*. Para efeito de teste nesta etapa, os programas *TeraTerm* e *RealmTerm* são mais recomendados, eles acessam facilmente a serial por uma interface gráfica e tudo o que é recebido pela porta de conexão é mostrado na tela. Contudo, para efeito de manipulação de dados, útil para testar o projeto uma vez que esteja finalizado, o MATLAB é percebido como mais interessante, conforme explicado na seção *2.3*.

3.1.Adaptação para o Projeto

Mais importante do que importar projetos de terceiros é realizar a adaptação dos mesmos para a aplicação desejada. No caso da aplicação proposta, o eco de serial não é a parte relevante, muito menos a inversão dos bytes recebidos. O que se deve usar como recurso para o projeto é a forma como os blocos lidam com os dados recebidos pela serial.

Diferentemente da aplicação explicada na seção *3.1*, que envia o byte recebido pela porta serial diretamente para o bloco *processor*, a aplicação proposta deve armazenar tais bytes em uma memória que será depois acessada para realizar os cálculos de convolução. Assim sendo, as estruturas do mecanismo de comunicação de comunicação serial utilizado no projeto tem a composição explicitada na Figura 4.

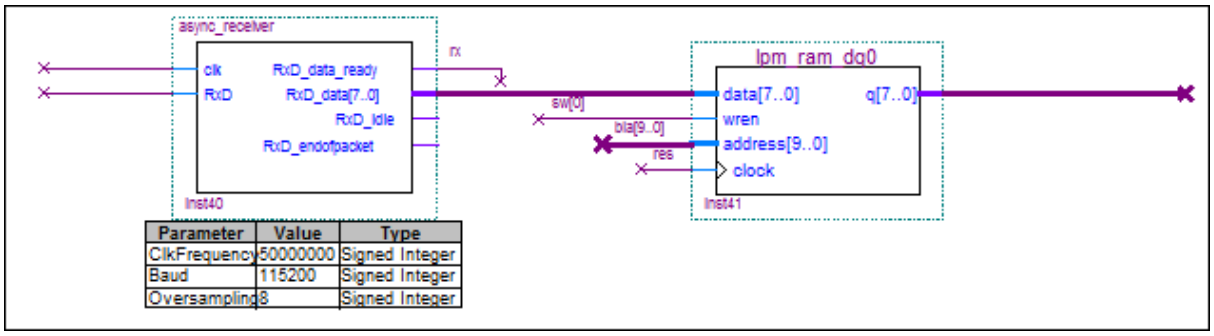

Figura 4. Diagrama em blocos de estrutura adaptada

Outra importante modificação realizada foi no bloco *processor*, que teve retirada do seu código a parte onde faz a inversão dos bytes – quando os dados do resultado da convolução forem enviados pela serial, devem estar colocados na ordem correta para que o computador possa tratar os dados recebidos de forma adequada.

3.2.Configuração do bloco

É possível notar que sob os blocos de recepção e transmissão existe uma tabela com algumas variáveis e seus valores. O objetivo dessa tabela é fornecer um modo prático de se alterar o valor de importantes parâmetros da comunicação serial - sem a necessidade de se reimplementar todo o bloco. Dentre os que aparecem na tabela, o mais relevante nesse contexto é o *baud rate*. Ele pode ser definido para qualquer valor desde que seja compatível com a máquina com que se está comunicando – em outras palavras, as duas máquinas que realizam a comunicação serial devem calibrar suas portas seriais para o mesmo valor de *baud rate*. Caso o *baud rate* do computador não esteja sincronizado com o do programa em questão (incluindo a quantidade de *start* e *stop* bits), os dados enviados e recebidos pela FPGA estarão fora de ordem e não serão corretamente manipulados por nenhuma das partes.

4. IMPLEMENTAÇÃO DO PROJETO

A figura 5 mostra todos os principais componentes presentes do projeto da aplicação de convolução. Percebe-se que muitos componentes estão repetidos – o que representa todos os pontos de paralelismo do projeto.

O projeto, ainda que não seja complexo, possui alguns pontos onde o estudante deve prestar atenção. Muitas memórias e multiplexadores podem confundir o estudante, o que torna esse projeto uma boa fonte de oportunidade para o aluno entender o destino e origem de cada sinal. Os blocos utilizados são quase todos advindos do próprio programa, sem a necessidade de qualquer tipo de descrição de hardware como Verilog ou VHDL. Muitas vezes, porém, tais blocos não são tão simples de se usar de forma que a leitura da especificação de cada componente faz-se necessária.

Ciclo de Operação

O ciclo do sistema dá-se em três etapas: aquisição da imagem, processamentos dos dados e envio do resultado. Na primeira parte, o sistema recebe os 1024 bytes de dados e popula as memórias com eles. Na segunda, o sistema acessa os dados das memórias, processa-os através do bloco de convolução e guarda o resultado em outro dispositivo de memória. Na terceira, o sistema acessa a memória com os resultados gerados na etapa anterior e envia pela serial. As três etapas são controladas pelos switches *SW0*, *SW1* e *SW2*.

Conforme explicado na seção 3.3, os blocos de comunicação serial enviam para as memórias os bytes recebidos. Para que seja possível a gravação dos bytes, o bit *write enable* das memórias devem estar ativados. Tal bit é indicado pelo switch SW0, que define a etapa 1.

A etapa 2 trata do processamento dos dados, onde o componente principal é o bloco de convolução (descrito em VHDL e explicado posteriormente). Tal bloco, contudo, recebe como entrada dados de nove memórias que, apesar de conterem os mesmo dados, têm seus endereços de acesso deslocados para que o píxel adequado seja acessado.

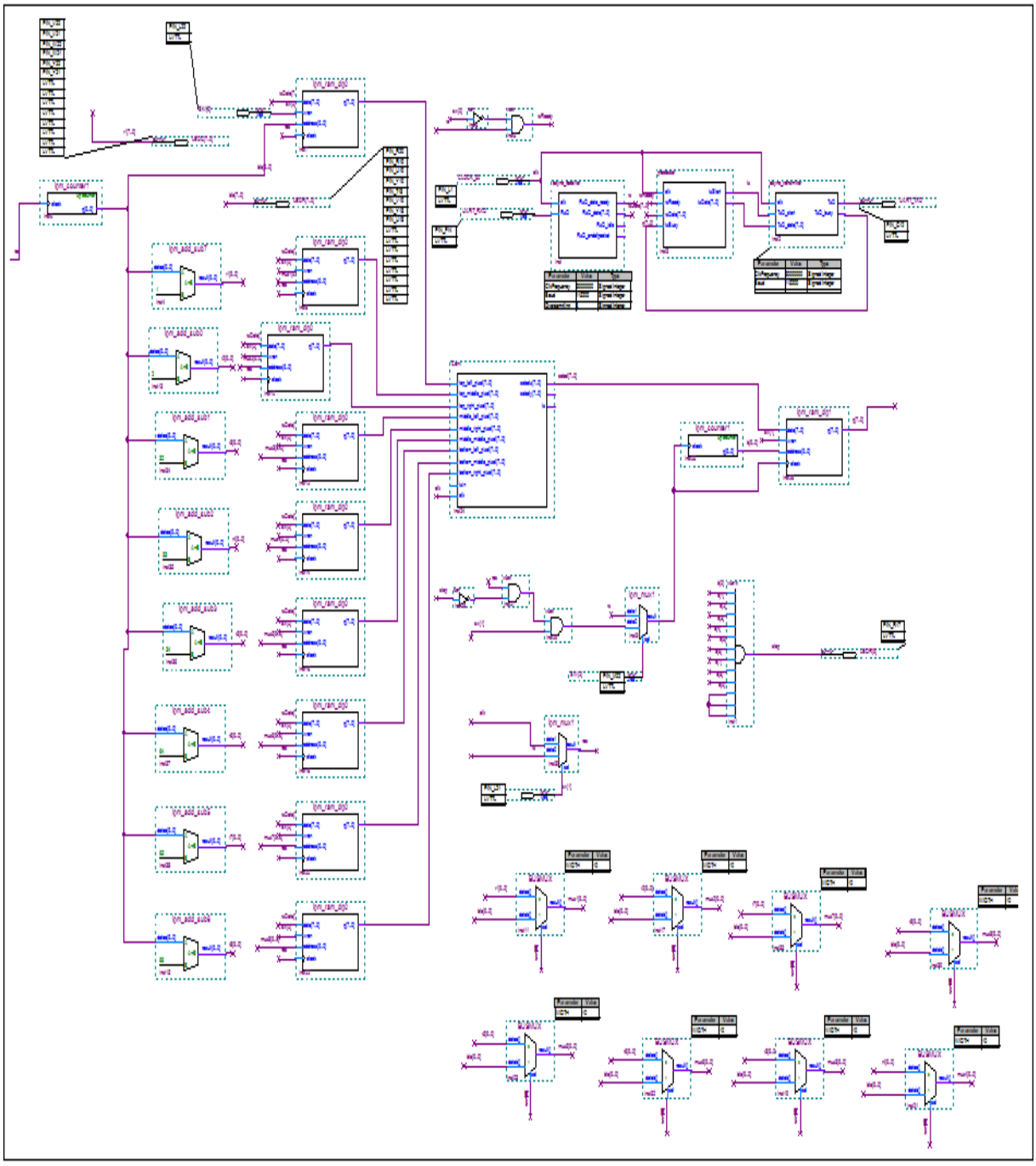

Figura 5 – Diagrama geral do sistema

O objetivo dos somadores antes de cada uma das memórias é justamente promover este deslocamento. Como as imagens são 32x32 pixels, os valores de deslocamento necessários são: 1,2,32,33,34,65,65 e 66, conforme mostrado na figura 6.

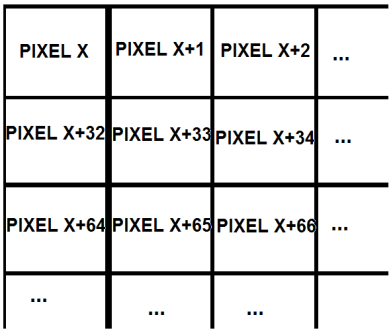

Figura 6 – modelo de deslocamento de endereço

4.1.Explorando o paralelismo com múltiplas memórias

A habilidade de realizar diversas funções em paralelo é uma das características mais relevantes de uma placa de lógica reconfigurável.

Como uma imagem de tamanho 32x32 possui um total de 1024 pixels, o processo de convolução de uma imagem requer muitos acessos à memória. Como a máscara utilizada no projeto em questão é de 3x3, ou seja, 9 pixels, para cada convolução, será necessário a consulta de 18 elementos na memória (9 da máscara mais 9 da imagem). Tendo que para cada consulta se gasta 1 ciclo de clock, 18 ciclos são necessários antes que mesmo de qualquer tipo de processamento ter começado (em um sistema com uma única memória). A situação é ainda mais crítica ao pensarmos em convoluções com máscaras maiores, por exemplo 5x5 ou 9x9. No segundo caso, 81 consultas a memória seriam necessárias antes de qualquer processamento começar.

No sistema projetado, todas as posições de memória utilizadas para cada passo da convolução são acessadas ao mesmo tempo, de forma paralela. Dessa forma somente um ciclo de clock é usado antes do cálculo da convolução - um valor muito melhor quando comparado aos 18 ciclos necessários para o caso não-concorrente. O sistema projetado realizada tal feito através da utilização de 9 dispositivos de memórias para a função de convolução (uma outra memória ainda é instanciada para que sejam armazenados os resultados da aplicação da máscara. Cada variável na multiplicação do método acessa uma memória independentemente das outras variáveis - em um circuito não-concorrente, apenas uma variável teria acesso à memória por vez. Nenhum ônus é acrescentado ao projeto pela utilização já que trata-se apenas da utilização de recursos já presentes no dispositivo. Tendo um limite de 8Mbytes de SDRAM na placa Cyclone II utilizada, 8000 memórias de 1024 endereços poderiam ser instanciadas – obviamente tal valor seria menor ao se utilizar memória que coubessem dados de imagens com resoluções mais altas.

4.1.Convolução em VHDL

O código VHDL da aplicação apresentada não é muito complexo pelo fato de realizar a convolução de apenas um píxel por vez. Contudo, alguns conceitos devem estar bem definidos para que não haja problemas durante a execução da aplicação.

A declaração dos sinais de entrada e saída (figura 7) devem ser bem pensados para que não haja conflito de tipo ou de gama (*range*). Os dados da memória – que representam os pixel que fazem a multiplicação de máscara - por representarem um número inteiro de 8 bits, são declarados como *integer range 0 to 255*.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

entity Conv is

    port (
        top_left_pixel      : in  integer range 0 to 255;
        top_middle_pixel    : in  integer range 0 to 255;
        top_right_pixel     : in  integer range 0 to 255;
        middle_left_pixel   : in  integer range 0 to 255;
        middle_right_pixel  : in  integer range 0 to 255;
        middle_middle_pixel : in  integer range 0 to 255;
        bottom_left_pixel   : in  integer range 0 to 255;
        bottom_middle_pixel : in  integer range 0 to 255;
        bottom_right_pixel  : in  integer range 0 to 255;
        clk                 : in bit;

        sobelx              : out integer range 0 to 255;
    );
```

Figura 7 – Declaração dos sinais de entrada e saída

Outro ponto importante na elaboração de um código VHDL é quando da elaboração de um processo síncrono. Para tal, a função do bloco projetado deve estar dentro do operador condicional *if(clk'EVENT and clk='1')*.

A convolução em si é bastante simples de realizar em VHDL, seguindo basicamente a fórmula:

aux <= (0*t1 - 1*t2 + 0*t3 -1*m1 + 4*m2 - 1*m3 +0*b1 - 1*b2 + 0*b3);

O fator mais relevante nessa parte é o fato de as variáveis poderem variar somente de 0 a 255, o que muitas vezes pode prejudicar o resultado da convolução. Assim sendo, instancia-se o sinal auxiliar *aux* que tem um *range* de -4095 to 4095, conforme mostrado na Figura 8.

```
architecture cv of Conv is

    signal aux : integer range -4095 to 4095;
    signal t1 : integer range 0 to 255;
    signal t2 : integer range 0 to 255;
    signal t3 : integer range 0 to 255;
    signal m1 : integer range 0 to 255;
    signal m2 : integer range 0 to 255;
    signal m3 : integer range 0 to 255;
    signal b1 : integer range 0 to 255;
    signal b2 : integer range 0 to 255;
    signal b3 : integer range 0 to 255;
```

Figura 8 – Variáveis auxiliares para realizar convolução

4.1.Resultado da aplicação

A figura 9 mostra o resultado da aplicação da convolução realiaza através da FPGA. A máscara utilizada foi o sobel horizontal, uma máscara de detecção de borda. Nota-se que o resultado é bastante coerente com o esperado para um filtro de realçamento de borda.

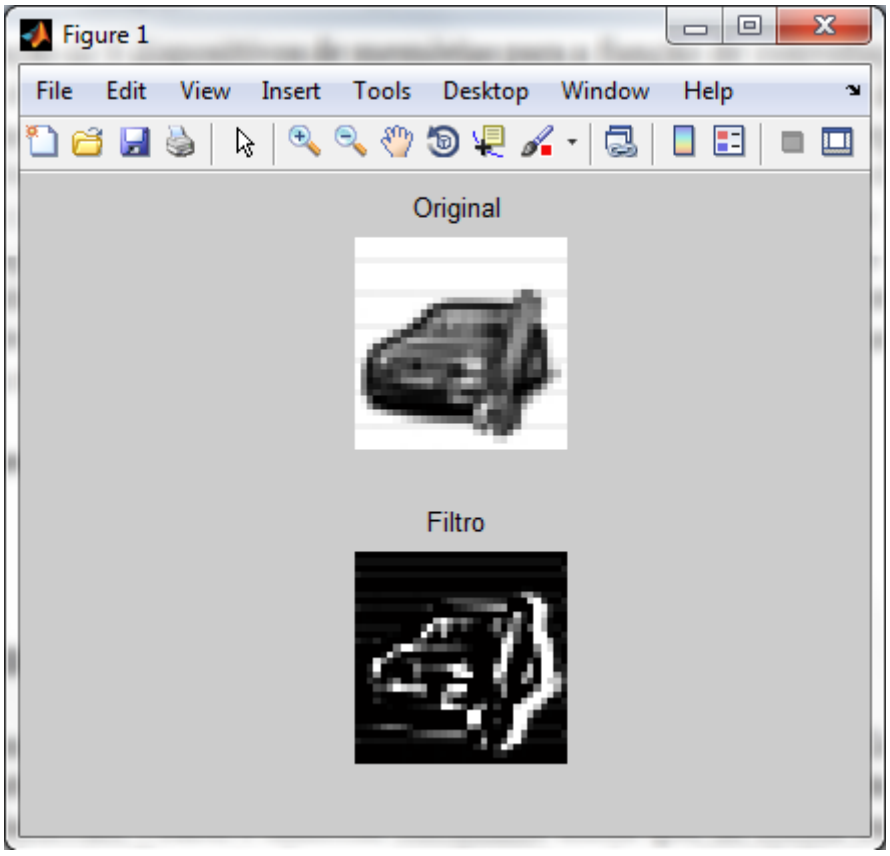

Figura 9 – Resultado da aplicação da convolução da imagem

5. CONSIDERAÇÕES FINAIS

Para promover a facilidade no aprendizado de conceitos de FPGA, a aplicação proposta demonstra o uso de muitos conceitos e dos mais variados componentes. Além disso, a linguagem de hardware VHDL é também trabalhada - ainda que de forma sucinta. O aluno deverá ser capaz de entender muito do processo de projeto de sistema de lógica reconfigurável após implementação de tal projeto. O paralelismo é explorado no projeto, podendo ainda ser bastante expandido. Seria possível, por exemplo, enviar pela porta serial o resultado da convolução a medida que cada pixel resultado fosse calculado. A aplicação não cobre todos os conceitos básicos necessários de aprendizado de FPGA, mas mostra uma aplicação com real utilizadade e que pode ser bem explorada para o ensinamento de lógica reconfigurável.

6. REFERÊNCIAS

ALTERA. **Design Tools & Services.** Disponível em < http://www.altera.com/products/software/sfw-index.jsp> Acesso em: 23 mai 2014.

Cyclone II FPGA Starter Development Board Reference Manual. Disponível em < http://www.altera.com/products/devkits/altera/kit-cyc2-2C20N.html> Acesso em: 23 mai. 2014.

FOHL, Dax. **Communicating with your Cyclone II FPGA over serial port, Part 3: Number Crunching.** Disponível em: <http://www.sparxeng.com/blog/software/communicating-with-your-cyclone-ii-fpga-over-serial-port-part-3-number-crunching > Acesso em: 23 mai. 2014.

MATLAB the language of Technical computing. Disponível em < http://www.mathworks.com/products/matlab/> Acesso em: 23 mai. 2014.

DIP APPLICATION AS CONTRIBUTION ON RECONFIGURABLE LOGIC TEACHING

Abstract: *This project proposes a teaching method of reconfigurable logic over the implementation of an image processing application. Such application contains concepts of parallelism, serial communication, VHDL and block diagram*

Key-words: *FPGA, Reconfigurable Logic, VHDL, Concurrent computing*