CBPF - CENTRO BRASILEIRO DE PESQUISAS FÍSICAS
Rio de Janeiro

Notas Técnicas

CBPF-NT-005/14
setembro 2014

Desenvolvimento de ferramentas via FPGA para leitura e processamento de imagens em tempo real

Filipe S. da Silva, Herman P. Lima Júnior, Márcio P. de Albuquerque

Ministério da
Ciência, Tecnologia
e Inovação

dx.doi.org/10.7437/NT2236-7640/2014.02.005
Notas Técnicas, v. 4, n. 2, p. 23–33, 2014

Desenvolvimento de ferramentas via FPGA para leitura e processamento de imagens em tempo real

Development of tools based on FPGA for image readout and processing in real time

Filipe S. da Silva*
Centro Brasileiro de Pesquisas Físicas,
Rua Dr. Xavier Sigaud 150,
Rio de Janeiro, RJ - 22290-180, Brasil
National Instruments Brasil

Herman P. Lima Júnior† e Márcio P. de Albuquerque‡
Centro Brasileiro de Pesquisas Físicas,
Rua Dr. Xavier Sigaud 150,
Rio de Janeiro, RJ - 22290-180, Brasil

Resumo: As FPGAs devido a seu processamento paralelo e confiabilidade são uma escolha interessante para processamento de imagens em situações onde é necessário processamento em tempo real com altas taxas de execução dos algoritmos. O desenvolvimento deste tipo de abordagem cria a necessidade de se desenvolver mecanismos para transmissão das imagens entre computadores ou câmeras e os FPGAs. Assim, neste trabalho, a partir de uma plataforma da Altera, e da linguagem de programação gráfica LabVIEW, foi elaborado um sistema capaz de realizar o gerenciamento e transferência de um conjunto de imagens entre um computador e memórias embarcadas em um FPGA, através do barramento *PCI Express*. O sistema foi avaliado quanto a integridade e velocidade de transmissão dos dados, onde obteve-se uma taxa de 48,91 *MBytes/segundo* o que resulta em 1643 imagens/segundo transmitidas para o FPGA. Conforme apresentado, estes resultados obtidos devem ser analisados considerando as características de todos os elementos, *hardware e software*, que constituem o sistema de transmissão de imagens. O desenvolvimento e entendimento das características de tempo e confiabilidade deste processo de transmissão constituem um primeiro passo para a aplicação de algoritmos de processamento de imagens embarcadas em FPGA.

Palavras-chave: Processamento de imagens; tempo real; FPGA; PCIe

Abstract: FPGAs due to their reliability and parallel processing are an interesting choice for image processing in situations where real-time processing with high rates of execution of algorithms is necessary. The development of this approach creates the need to develop mechanisms for transmission of images between cameras or computers and FPGAs. In this paper, with Altera Platform, and the graphical system design language LabVIEW, a system that can control, manage and transfer images between a computer and embedded memories, through the PCIe bus, was developed. The system was evaluated for completeness and speed of data transmission, where was obtained a rate of 48.91 MBytes /second resulting in 1643 images/second transmitted to the FPGA. As presented, these results should be analyzed considering the characteristics of all elements, hardware and software, that constitute the system of transmitting images. The development and understanding of time and reliability characteristics of the image transmission process is a first step for the application of image processing algorithms embedded in FPGA.

Keywords: Image processing; real-time; FPGA; PCIe

1. INTRODUÇÃO

O processamento de imagens digitais é hoje uma importante ferramenta de análise quantitativa em várias áreas científicas. Entretanto, o desenvolvimento de técnicas de processamento de imagens pode ainda ser relativamente complexo e frequentemente custoso computacionalmente. Muitas destas técnicas podem ser desenvolvidas em dispositivos eletrônicos, como os FPGAs [1] (*Field-Programmable Gate Array)* os DSPs [2] (*Digital Signal Processor)* ou ainda por meio dos processadores multicore. O processamento paralelo dos processadores multicore é um candidato muito forte para alcançar um alto desempenho, principalmente devido à quantidade de bibliotecas computacionais existentes [3]. No entanto, como o paralelismo é muito dependente do problema, é normalmente difícil desenvolver e analisar códigos computacionais em ambiente multicore. Por outro lado, a programação em dispositivos FPGA é uma alternativa bastante interessante, e é normalmente limitada pela complexidade dos ambientes de *software* e *hardware* envolvidos. O interesse deste trabalho é explorar este ambiente de desenvolvimento em FPGA, caracterizando seu potencial de apli-

*Electronic address: filipe.silva@ni.com;fisacchi@hotmail.com
†Electronic address: hlima@cbpf.br
‡Electronic address: mpa@cbpf.br

cação para processar uma quantidade elevada de imagens por segundo.

O processamento de uma imagem está relacionado diretamente ao processamento da informação nela presente. Em ambientes na qual está técnica é usada como instrumento científico, ele está na base do sistema de medida do fenômeno físico que se apresenta sob a forma de imagem (sinal bi ou tridimensional). As imagens são capturadas por câmeras ou sensores diversos e a informação passa por uma sequência de transformações com a finalidade da realização de uma medida.

As tarefas que compõem o processamento e análise de imagens podem ser agrupadas em diferentes etapas. Este agrupamento é algumas vezes referenciado como pirâmide de processamento [4], conforme visto na Figura 1. Na base da pirâmide está aquisição e o pré-processamento com objetivo de melhorar a qualidade das imagens, realçando as características relevantes (nível dos *pixels*). No nível imediatamente acima se encontra a fase de segmentação, com objetivo de separar a imagem em suas partes constituintes. Cabe destacar que esta etapa é bastante subjetiva, pois é a passagem da representação da imagem na forma de um sinal (2D ou 3D) para os símbolos (informações) nela presente. A Classificação é a etapa usada para identificar objetos ou parte destes. Por fim, na etapa de reconhecimento é obtida uma descrição ou alguma outra interpretação das imagens analisadas. Em instrumentação cientifica é possível realizar medidas em todas as fases apresentadas.

Alguns processos de controle ou análise baseados em imagens requerem taxa de aquisição bastante controlada (em alguns casos com taxas muito rápidas).

O comportamento em algumas situações pode exigir um funcionamento com características de tempo real, ou seja, a resposta para um determinado evento deve ser dada dentro de um limite de tempo.

Figura 1: Pirâmide com etapas do processamento de imagens

O objetivo de sistemas computacionais em tempo real é cumprir uma exigência de tempo específico para cada tarefa. A propriedade mais importante de um sistema em tempo real não é sua capacidade de executar tarefas rápidas (que é uma característica relativa) e sim sua previsibilidade, ou seja, sua funcionalidade e comportamento em um intervalo de tempo devem ser tão previsíveis quanto necessário para satisfazer as especificações do sistema principal. Um sistema computacional rápido pode ser útil para atender especificações de tempos rigorosos, mas a computação rápida por si só não garante a previsibilidade, necessária em sistemas de tempo real [5].

A partir da perspectiva do processamento de imagens, um sistema de tempo real será aquele que captura regularmente as imagens, analisa, obtém informações e então utiliza estes dados para alguma ação de controle. Todos estes passos devem ocorrer dentro de um intervalo de tempo bem definido [1].

Como exemplo de sistema de processamento em tempo real, pode ser citada a navegação autônoma de robôs através de imagens de câmeras em ambientes dinâmicos [6].

Conforme mencionado anteriormente, a necessidade de resposta para eventos dentro de um intervalo limite de tempo, muitas vezes na ordem de milissegundos ou menos, faz com que a abordagem tradicional de processamento de imagens por meio de processamento serial dos algoritmos não seja indicada, já que nestes casos o tempo pode ser crítico.

Felizmente, a estrutura das imagens digitais possibilita uma abordagem de processamento por meio de códigos de execução em paralelo, especialmente para as etapas que compõem os primeiros níveis da pirâmide de processamento [1]. Deste modo, os dispositivos de *hardware* reconfiguráveis (FPGAs) se apresentam como uma boa alternativa para o processamento de imagens, já que são genuinamente paralelos, permitindo assim sua aplicação em sistemas com necessidades de operação em tempo real.

Na literatura é possível encontrar diferentes trabalhos relacionados à utilização de FPGA para processamento de imagens em situações onde é necessária uma abordagem de tempo real. Em [7] é demonstrada a utilização de FPGA para reconhecimento facial. Já em [8] utiliza-se a abordagem em FPGA para reconhecimento de placas em uma rodovia. [9] mostra a utilização de FPGA no tratamento de imagens médicas para segmentação das áreas branca e cinza do cérebro.

A utilização de processamento baseado em *hardware* configurável é uma opção interessante para experimentos científicos onde é necessária a observação de fenômenos, aquisição e processamento de imagens. Nos exemplos mostrados em [10] [11], onde taxas de até 10000 imagens/segundo são necessárias para detecção de um fenômeno denominado *Marfe* [12], que ocorre no interior de reatores de fusão nuclear.

Assim, com o objetivo de se estudar a inserção de algoritmos de processamento de imagens em hardware configurável, e a fim de se criar ferramentas que possam ajudar a solucionar desafios tecnológicos como os anteriormente mencionados, o presente trabalho visa estabelecer a metodologia para o envio de um conjunto de imagens, já previamente obtidas, para um kit contendo FPGA, através de barramento PCIe [13]. Além disto, outro objetivo é obter uma estimativa do tempo para transferência dos dados através do sistema proposto. O entendimento destes processos é o passo inicial para aplicação de processamento de imagem embarcado em *hardware*, conforme necessidades dos exemplos citados anteriormente.

2. FERRAMENTAS DE DESENVOLVIMENTO

Este trabalho apresenta o desenvolvimento, aplicação e caracterização de uma ferramenta para um problema de

processamento de imagens, ainda nas etapas iniciais da pirâmide de processamento, a fim de se avaliar seu potencial de aplicação em sistemas de tempo real. No desenvolvimento e caracterização do processo de transferência de imagens entre um computador e a memória de um FPGA, utilizou-se componentes de *hardware* e *software*, que permitiram, além da implementação do barramento de transmissão PCIe, aplicar o pré-processamento necessário às imagens antes das transferências destas para o FPGA.

O Sistema de Acesso ao FPGA, para gerenciamento e transferência das imagens foi desenvolvido e testado em um computador Pentium com CPU 2.80GHz, 1GB de RAM e HD de 220 GBytes, rodando sistema operacional *Windows XP Professional*, versão 2002/*Service Pack 3.*

2.1. Kit Altera Cyclone IV GX

O item de *hardware* utilizado foi o kit Altera Cyclone IV GX [14]. Este possui um FPGA modelo EP4CGX150DF31, interfaces de memória e periféricos que permitem a comunicação com dispositivos externos ao kit.

Uma característica deste kit é a existência de uma porta de comunicação com barramento PCIe x4 Gen1[1], a qual permite que ele seja diretamente conectado à placa-mãe do computador de desenvolvimento.

A Figura 2 mostra o kit, e os seus recursos disponíveis.

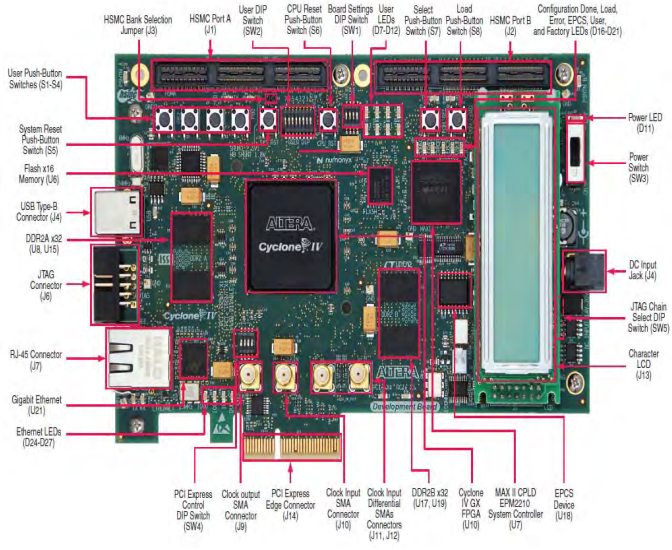

Figura 2: Kit de desenvolvimento Altera Cyclone IV G2.2 – Ferramenta Quartus II

Para o desenvolvimento de projetos com FPGA, a Altera disponibiliza a ferramenta de *software* Quartus II [15], a qual pode ser obtida gratuitamente de seu site, na versão *Web Edition.*

[1] Na especificação do barramento PCIe 1.0, denominada Gen1, a velocidade de transmissão é de 250MB/s por canal de transferência (lane).

Com esta ferramenta é possível desenvolver os projetos em linguagem descritiva de hardware (HDL), simular, compilar e através de uma ferramenta de programação, realizar a configuração do FPGA.

O ambiente de desenvolvimento do Quartus II permite a integração de arquivos provenientes de diferentes fontes para a criação da lógica a ser sintetizada no dispositivo.

É possível inserir algoritmos criados diretamente em linguagem HDL (e.g. VDHL ou Verilog) [16], ou mesmo trabalhar com blocos lógicos previamente criados que representam funções como portas lógicas e memórias.

Um recurso importante que faz parte do ambiente Quartus II é a ferramenta de integração Qsys. Esta, através de um alto nível de abstração e automatização, cria as interconexões entre blocos de funções pré-existentes (*IP blocks*), poupando o programador de realizar, através de linguagem HDL, todas as interfaces necessárias.

Um projeto dentro do ambiente Quartus II pode ser visualizado através de um arquivo bdf(*block design file*). Através deste, toda lógica é vista por meio de um diagrama em blocos, o que facilita verificação das interconexões entre os mesmos e a inserção de mais blocos funcionais. A Figura 3 mostra o ambiente de desenvolvimento Quartus II, e a visualização dos componentes lógicos através de blocos.

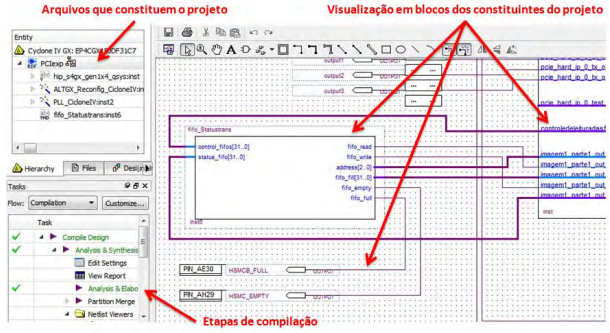

Figura 3: Visão geral do ambiente de desenvolvimento Quartus II

2.2. Ferramenta LabVIEW

Foi necessário o desenvolvimento de um sistema capaz de importar as imagens de um arquivo, realizar o pré-processamento nas mesmas e acessar o barramento PCIe de forma ordenada para a correta transferência das imagens para o kit de desenvolvimento.

Para elaboração da etapa anteriormente mencionada, utilizou-se a ferramenta de desenvolvimento LabVIEW, da empresa *National Instruments*. Com esta foi possível, de forma gráfica, elaborar o algoritmo necessário.

Uma característica desta ferramenta é a facilidade na elaboração de interfaces visuais de controle, as quais permitem aos usuários interação com os algoritmos criados, inserindo e obtendo informações dos mesmos.

O LabVIEW conta com uma biblioteca de funções denominada VISA (*Virtual Instrument Software Architecture*) desenvolvida pela *National Instruments*, que permite o acesso a dispositivos externos de instrumentação, através de interfaces GPIB, PXI, PCI,PCIe, Ethernet, USB e Serial.

Um elemento interessante do pacote VISA é o *NI-VISA Driver Wizard* [17], ferramenta que permite a criação de *drivers* para dispositivos externos USB e PCI(e)/PXI(e).

3. DESENVOLVIMENTO

A elaboração do sistema de transferência de imagens entre computador local e o FPGA pode ser divida em três etapas. Inicialmente é necessário obter as seguintes características das imagens a serem transmitidas: tamanho em *pixel* [18], quantidade de *bytes* por *pixel* e região de interesse dentro das imagens. As características mencionadas permitem definir como as imagens serão transferidas e armazenadas no kit de desenvolvimento, além da quantidade de memória que será utilizada durante a segunda etapa que trata da programação do FPGA. Por fim, é necessário desenvolver a interface que permite carregar as imagens da memória do computador, pré-processá-las e acessar o barramento PCIe de forma ordenada para transmissão.

3.1. Imagens

Foram utilizadas imagens de distribuição livre, obtidas diretamente do banco público de imagens científicas da Fundação Oswaldo Cruz – FIOCRUZ (www.bancodeimagens.fiocruz.br*)*.

A Figura 4 apresenta um exemplo de imagem utilizada neste trabalho, suas características estão mostradas acima da imagem. Vê-se então que esta imagem tem resolução de 224 x 256 *pixels*, totalizando 57344 *pixels* com 8-bits em tons de cinza, i.e. cada pixel é formado por 1 *byte*, tendo seu valor variando entre zero e 255, onde zero representa o preto e 255 representa o branco em uma escala de cinza.

Frequentemente dentro de uma imagem, os eventos de interesse estão limitados em uma região específica, denominada região de interesse (ROI). Assim, o pré-processamento realizado em computador permite reduzir o tamanho das imagens, levando a redução no uso de memória no FPGA, e na quantidade de dados trocados entre o computador de desenvolvimento e o kit de desenvolvimento.

Para a imagem utilizada estipulou-se uma ROI conforme ilustrada na Figura 5.

Após o pré-processamento mencionado, cada uma das imagens apresenta um tamanho de 160x186 *pixels*. Ou seja, uma redução de 48,1% no tamanho original de uma imagem, que passou de 57344 Bytes para 29760 Bytes, i.e. *pixels*.

3.2. Projeto para o Kit de Desenvolvimento Altera

O projeto desenvolvido para o FPGA é constituído de uma interface PCIe programada a partir de um *IP block (IP_Compiler for PCI)* [19] que permite a transferência de imagens entre o computador e o kit de desenvolvimento e vice-versa. O projeto possui também memórias FIFO [20] responsáveis por armazenar as regiões de interesse obtidas

Figura 4: Exemplo de uma imagem a ser transferida ao FPGA

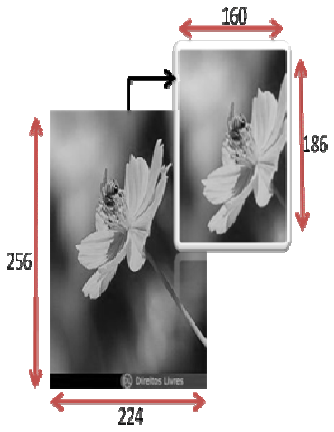

Figura 5: Exemplo da imagem cortada pela região de interesse (ROI)

das imagens a partir de seu pré-processamento realizado em computador.

Sabendo que cada uma das imagens após pré-processamento apresenta 29760 *Bytes*, e que cada uma das FIFOs inseridas na ferramenta Qsys pode possuir tamanho máximo de 8192 posições de 8 *bits*, utilizou-se uma arquitetura onde cada imagem ocupa 4 FIFOs consecutivas. De acordo com o que foi mencionado anteriormente, cada imagem dentro do FPGA ocupa uma memória de 32768 Bytes, o que representa 4% da memória total do dispositivo (810000 Bytes). A partir destes números, estipulou-se a transferência de 16 imagens, o que resulta em 64% da memória total. Deste modo, tem-se 36% de memória disponível para futuros desenvolvimentos.

Foram configuradas 64 FIFOs de entrada para recebimento das 16 imagens. Cada uma destas FIFOs é responsável por armazenar 25% da quantidade total de *pixels* de uma imagem, i.e. 7440 pixels. Cada uma destas FIFOs dentro da ferramenta Qsys recebeu um rótulo que identifica a qual imagem e qual parte desta, a FIFO está associada, por exemplo, a FIFO identificada por “Imagem1_Parte1”, armazena os primeiros 7440 pixels da primeira imagem. A Figura 6 mostra a divisão de uma imagem para armazenamento nas quatro FIFOs independentes.

É importante mencionar que esta divisão das imagens em blocos independentes, i.e. 4 FIFOS, além de ser necessária devido ao comprimento máximo das FIFOs na

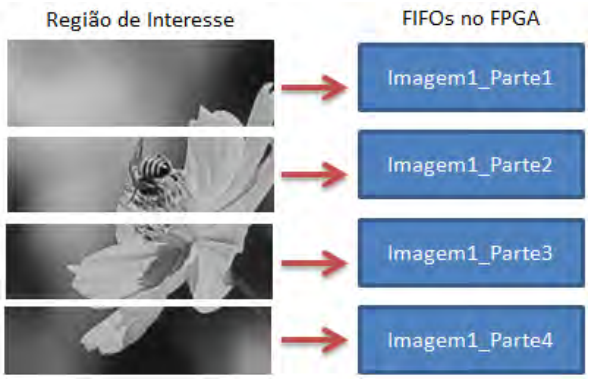

Figura 6: Divisão de uma imagem para armazenamento em 4 FIFOs independentes

ferramenta Qsys, permitirá em trabalhos futuros, explorar a técnica de paralelismo de dados [4] para o processamento de imagens em FPGA.

A Figura 7 mostra a tela de programação do *IP Block* para o barramento PCIe. Esta interface PCIe é interna ao FPGA e utiliza uma abordagem via *hard IP* [19]. Na Figura 8 é possível verificar a tela de programação utilizada para cada uma das FIFOs inseridas.

Para configuração *do IP_Compiler for PCI* é necessário inserir informações como identidade do fabricante e do dispositivo. Estas mesmas informações devem ser utilizadas para criação do *driver* de comunicação, a partir da ferramenta *NI-VISA Driver Wizard.*

Foi determinada a utilização de 3 endereços base de registro (BARs), através dos quais é possível realizar escrita e leitura no barramento PCIe.

As FIFOs foram configuradas para operar em modo *dual clock*, o que permite associar dois sinais de *clock* diferentes e criar duas portas de *status* associados à escrita e leitura dos dados.

O sinal de *clock* para controle de escrita e leitura das FIFOs foi obtido a partir do *IP_Compiler for PCI,* sendo definida a frequência de 125MHz, máximo valor possível para o kit Altera Cyclone IV GX utilizado [19].

As portas de *status*, dentre outras informações, oferecem um *flag* denominado FIFO_QuaseCheia, este informa o momento quando a memória alcança uma quantidade de elementos pré-configurada. Este indicador é interessante, pois para a transmissão das imagens, as filas de 8192 posições devem ser consideradas cheias quando alcançarem 7440 elementos.

O trabalho com a ferramenta Qsys exige que o programador execute as ligações entre *IP Blocks,* determinando características como *clock*, *reset* e interfaces de comunicação. As portas de entrada das FIFOs foram conectadas no endereço base um (BAR 1), enquanto as portas que informam o *status* de escrita, como FIFO_QuaseCheia e FIFO_Vazia foram conectados no endereço base dois (BAR 2). Já as portas de saída das FIFOS foram associadas ao endereço base três (BAR 3).

Cada uma das FIFOs conectadas aos endereços base (BARs) da interface PCIe recebeu um endereço específico. Isto permite que o programa que é executado no computador

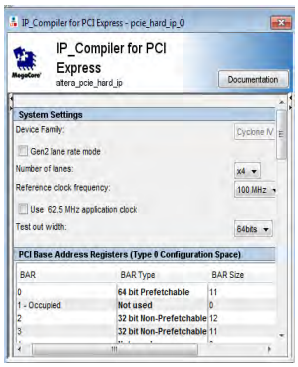

Figura 7: IP Block para configuração do barramento PCIe

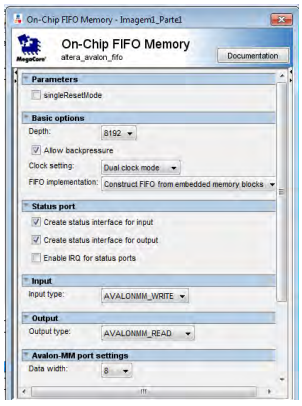

Figura 8: IP Block para configuração das memórias FIFO

local tenha capacidade de acesso independente para cada informação de entrada/saída disponibilizada pelas FIFOs. Esta tarefa de alocação manual de endereços foi realizada diretamente na ferramenta Qsys. A Figura 9 mostra a ferramenta para determinação dos endereços associados a cada BAR do barramento PCIe.

Observando a figura 9, percebe-se que o endereço 0x00 associado ao BAR1 está relacionado ao Controle_Acesso_Fifos, que foi definido como uma porta de controle para o algoritmo sintetizado no FPGA, i.e., através do acesso ao BAR1, endereço 0x00, é possível enviar até 32 bits de controle a partir do programa que roda no computador local. Assim, a partir de uma ação do usuário na interface de controle é possível disparar ações dentro do FPGA.

A ferramenta Qsys permite que informações referentes a entradas e saídas dos blocos funcionais sejam exportadas para o ambiente do Quartus II. Como exemplo, neste projeto foi realizada a exportação dos dados referentes aos *status* de leitura da primeira FIFO (Imagem1_Parte1), a fim de se associar pinos digitais disponíveis no kit de desenvolvimento, para medida com osciloscópio dos sinais relativos a informações de FIFO_Vazia e FIFO_QuaseCheia. O intervalo entre a transição destes dois sinais será uma estimativa do tempo total de transferência.

Foram adicionados blocos funcionais como PLLs e interfaces de saída digital, além de uma função personalizada programada em VHDL criada exclusivamente para configurar o parâmetro FIFO_QuaseCheia. Este parâmetro está disponível na porta *status* de leitura da primeira FIFO (Ima-

gem1_Parte1), e permite a verificação das *flags* através de interfaces digitais de saída. O controle desta função, para seleção de escrita de configuração da FIFO_QuaseCheia ou leitura dos dados de *status*, é realizado pela interface no computador local via porta de controle "Controle_Acesso_fifos". Esta porta é configurada no endereço BAR 1 do barramento PCIe. Na Figura 10 é visto o ambiente de programação do Quartus II, com o *IP Block* do PCIe obtido diretamente da ferramenta Qsys. É mostrado também o bloco personalizado para controle de configuração e leitura dos *flags* de *status* da FIFO referente à Imagem1_Parte1.

É importante mencionar que os estados relacionados às porta de escrita de todas as FIFOs são lido pelo software executado no computador local. Assim, para leitura do sinal via osciloscópio convencionou-se utilizar o registro de status relacionado à porta de leitura.

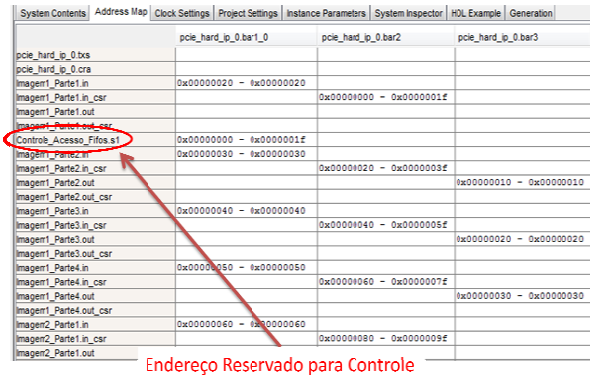

Figura 9: Mapa de endereços do barramento PCIe observados a partir da ferramenta Qsys/QuartusII

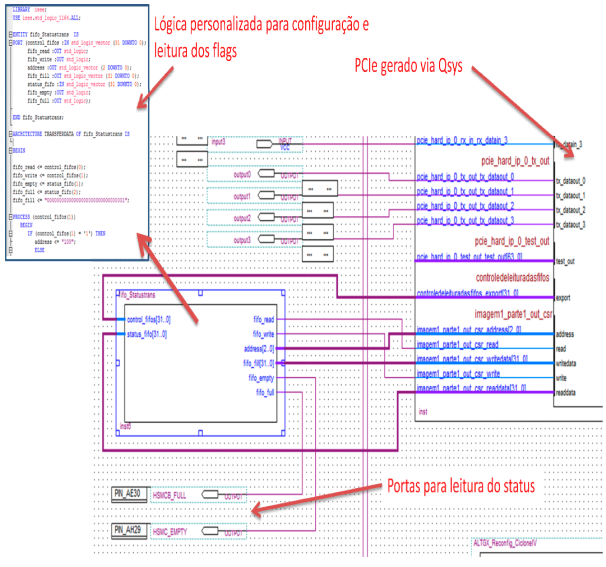

Figura 10: Parte do projeto desenvolvido no Quartus II

3.3. Interface de acesso ao FPGA

Para o acesso ao FPGA para envio das imagens, foi desenvolvido um programa em LabVIEW. Este utiliza a biblioteca VISA que permite o acesso aos registros base de dispositivos conectados a diversos barramentos, dentre eles PCIe.

Inicialmente, foi gerado o *driver* de comunicação através da ferramenta *NI-VISA Driver Wizard*. Uma vez instalado o *driver*, o FPGA já sintetizado com o protocolo PCIe passou a ser reconhecido como um instrumento conectado ao computador de desenvolvimento. A Figura 11 mostra uma imagem da tela do aplicativo "*Measurement & Automation*" da National Instruments que gerencia todos os dispositivos associados ao computador local. Nesta figura é possível observar a FPGA CycloneIV reconhecida como um instrumento conectado ao computador.

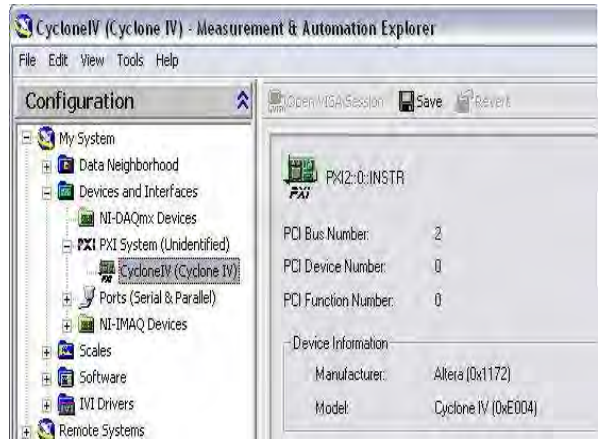

Figura 11: CycloneIV reconhecida como um instrumento

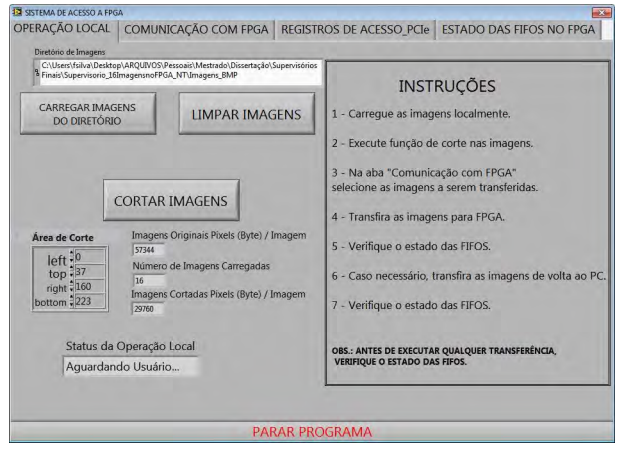

Figura 12: Interface para acesso ao FPGA

A interface para acesso ao FPGA apresenta, conforme visto na Figura 12, quatro abas com respectivas ações e informações para operação do sistema. Na seção "Operação Local", é possível carregar um conjunto de imagens do disco local e realizar operações de corte de tamanho configurável sobre as mesmas, afim de obter suas regiões de interesse. A interface também informa em Pixels (Bytes), o tamanho dos dados, antes e após o corte.

Na seção "Comunicação com FPGA", é possível selecionar as imagens previamente carregadas e cortadas para transferência à memória do dispositivo de lógica programável, sendo possível a transferência de até 16 imagens. Informações como o *status* de comunicação com o FPGA são

vistos nesta aba. Caso necessário, é possível realizar a transferência dos dados a partir do FPGA para memória local, esta ação é tomada através do botão "Transferir para PC". O botão "Verificar FIFOS de entrada" roda uma verificação sobre o *status* de todas FIFOS, informando então na aba "Estado das FIFOS no FPGA" quantos elementos estão contidos em cada uma delas e se as mesmas encontram-se cheias ou vazias.

Na seção "Registros de Acesso_PCIe" é possível configurar os endereços para acesso às portas de entrada, portas de *status* de escrita e portas de saída de todas as FIFOs. Estes endereços devem ser os mesmos daqueles configurados no mapa de endereços da ferramenta Qsys.

A programação para realização das tarefas anteriormente mencionadas foi feita a partir de uma linguagem gráfica. Os algoritmos seguem um funcionamento baseado em máquina de estado orientada por eventos de usuário, para resposta às interações com os botões da interface [21].

A Figura 13 mostra uma das sub-rotinas do programa, responsável pela escrita individual nas FIFOs. Utiliza-se funções da biblioteca VISA que acessam individualmente os endereços base de registro (BAR) do barramento PCIe.

Uma característica importante é que o estado de inicialização do programa deve procurar pelo FPGA conectado no barramento PCIe e acessar os endereços de *status* de escrita de todas as filas, para configuração do parâmetro FIFO_QuaseCheia com um valor de 7440. Assim as FIFOs podem ser consideradas cheias quando alcançam 7440 elementos (i.e. *pixels*) escritos. A Figura 14 mostra um diagrama com as etapas de inicialização e ações disponíveis ao usuário pela interface de acesso.

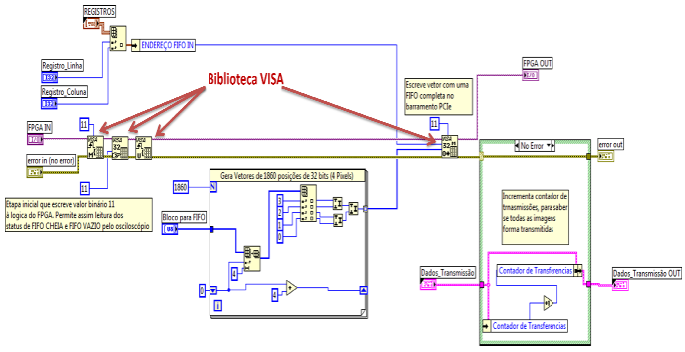

Figura 13: Programa desenvolvido em linguagem gráfica

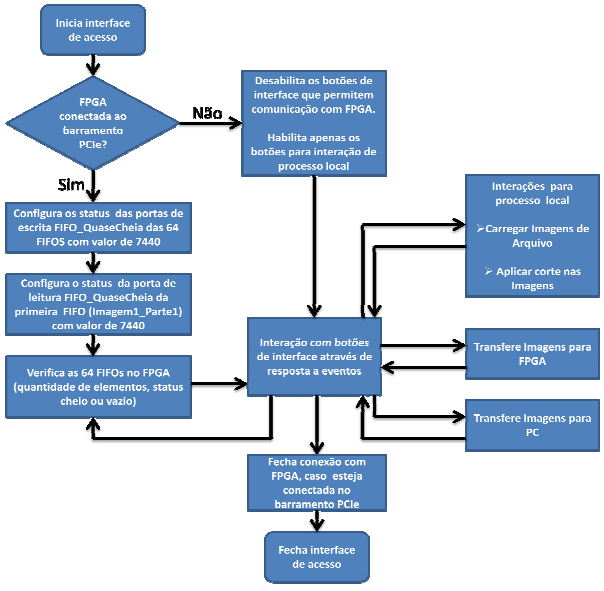

Figura 14: Etapas de inicialização e ações disponíveis ao usuário a partir da interface de acesso

4. METODOLOGIA PARA ESTIMATIVA DO TEMPO DE TRANSFERÊNCIA DOS DADOS

O *driver* VISA para comunicação com barramento PCIe através da interface de acesso, possibilita interações de escrita e leitura em espaços de memória de instrumentos externos, no caso o Kit de desenvolvimento contendo FPGA. Conforme [22], estas interações do *driver* VISA ocorrem em acessos de 32 bits no barramento PCIe. Assim, mesmo que o *payload* de dados [23] do protocolo PCIe seja configurado através da ferramenta Qsys, para o valor de 256 Bytes (máximo valor para o Kit de desenvolvimento utilizado [19]), apenas acessos independentes de 32 bits são realizados.

O *driver* VISA por se tratar de uma ferramenta de *software*, cujo código não pode ser alterado pelos usuários, impossibilita a configuração de parâmetros (endereços de escrita/leitura, quantidade de transferências a serem realizadas) de controladores de DMA (*Direct Access Memory*) [23] que por ventura sejam inseridos na FPGA conectada ao barramento PCIe. Deste modo, mesmo possuindo um controlador de DMA, a FPGA não pode operar como *Bus Master* conforme descrito em [24], inviabilizando então transferências otimizadas de dados em modo rajada (*burst*) [19] para a interface de acesso via *driver* VISA.

Sabe-se que a utilização de controladores DMA para transferência em modo rajada, acelera as transmissões de dados entre a memória do computador e do dispositivo externo conectado ao barramento PCIe e permite a exploração das bandas de transmissão do barramento PCIe [25]. No entanto, devido as características do *driver* VISA anteriormente mencionadas, este trabalho limita-se em realizar transferências individuais de pacotes de dados contendo 32bits (i.e. 4 *pixels* da imagem transferida).

A metodologia para medição do tempo de transmissão considera então que o sistema de transferência das imagens é constituído pelo *driver* de acesso VISA, barramento PCIe (contendo todo o *overhead* inserido pelo protocolo PCIe [23]) e a lógica embarcada no FPGA para armazenamento das imagens, a qual é executa com um *clock* de 125MHz, conforme mencionado na seção 3.2.

A Figura 15 mostra as camadas de *software* e *hardware* que compõem o sistema de transferência, considerando uma transmissão no sentido PC para FPGA. Percebe-se que a cada chamada do *driver*, a camada de aplicação direciona um bloco de 1860 elementos de 32 bits para serem transmitidos pelo barramento PCIe. Convencionou-se utilizar esta quantidade de elementos, pois resulta em 7440 *Bytes* (i.e. 7440 *pixels*), o que é a quantidade de um bloco individual de uma imagem. Esta quantidade de dados pode ser transferida sem a interferência da camada de aplicação, uma vez que todos

os elementos estão direcionados a uma mesmo endereço (i.e. uma FIFO) dentro do FPGA.

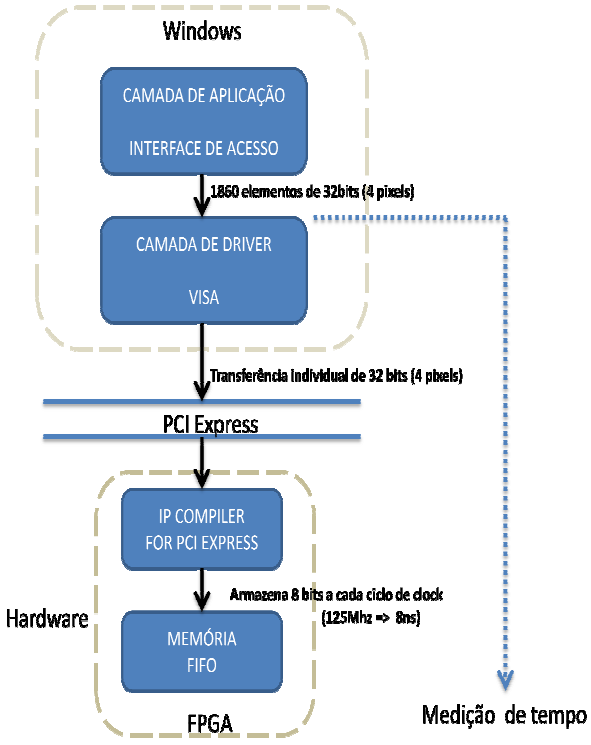

Figura 15: Camadas de software e hardware que compõem o sistema de transferência de imagens

Transferências de blocos de dados a partir da camada de aplicação são possíveis devido a função vi*MOVEOUT32* presente na biblioteca VISA. Esta apresenta o melhor desempenho em termos de velocidade quando comparado com as demais funções de acesso a registros da biblioteca VISA [26].

Quando a camada de *driver* é chamada, está realiza 1860 acessos ao barramento PCIe transferindo os pacotes de 32 bits sem interferência da camada de aplicação. Devido a este fato, a metodologia para medição de tempo desconsidera o tempo da camada de aplicação responsável por montar o bloco de 1860 elementos de 32 bits, conforme indicação da linha pontilhada na Figura 15.

Para a medição do tempo de transferência do bloco contendo os 1860 elementos de 32 bits utiliza-se os indicadores FIFO_Vazia e FIFO_QuaseCheia para a primeira FIFO (Imagem1_Parte1). O tempo de transição entre estes dois sinais indicará o tempo necessário para preencher toda uma FIFO com 7440 *pixels*.

Para realização da tarefa mencionada é preciso que na inicialização do *software* de acesso ao FPGA seja realizada a configuração do parâmetro FIFO_QuaseCheia do *status* de leitura da primeira FIFO (Imagem1_Parte1), para o valor 7440. Esta etapa pode ser vista no diagrama da Figura 14.

A Figura 16 mostra o *status* de escrita para todas as FIFOs antes da transmissão das imagens. As informações são expressas em formato de uma matriz com 64 elementos (4 x 16), onde cada um deles informa a situação para cada uma das FIFOs (i.e, cheia, vazia e a quantidade de elementos contidos nessas filas). As colunas da matriz representam a quantidade de imagens e as linhas informam as FIFOs para cada uma delas. A matriz pode ser percorrida através dos controladores de indexação localizados em seu canto superior esquerdo.

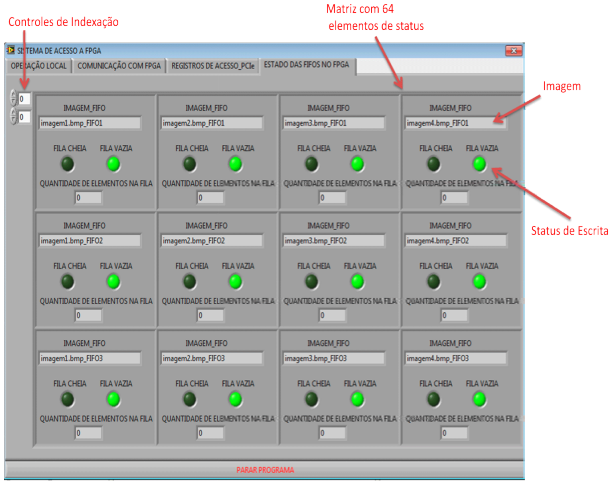

Figura 16: A imagem apresenta o Sistema de Acesso ao FPGA. É possível ver o status de 12 das 64 FIFOS antes da transmissão. Neste instante todas as 12 FIFOS estão vazias

5. RESULTADOS

A transferência dos dados para a memória do dispositivo de lógica programável foi validada a partir da utilização do Sistema de Acesso, desenvolvido neste trabalho. A confirmação da integridade dos dados recebidos e armazenados na memória do dispositivo foi possível a partir da verificação das imagens retornadas para o aplicativo (no LabVIEW). A Figura 17 apresenta as etapas de avaliação do procedimento de transferência. A Figura 18 mostra a tela de *status* após o envio das imagens. Nesta figura, é possível perceber a quantidade de elementos para cada uma das FIFOs, juntamente com seus estados atuais.

Quando o procedimento de retorno é realizado, uma matriz de dados contendo as imagens obtidas é preenchida no Sistema de Acesso. Esta matriz possui até *n* x 29760 elementos, onde *n* é a quantidade de imagens retornadas do FPGA.

Esta matriz de *pixels* pode ser então apresentada no formato de uma imagem, utilizando-se o botão "*Visualizar Imagens obtidas do FPGA*", na tela do Sistema de Acesso. A exibição da matriz de *pixels* ocorrerá em uma taxa de 1 imagem por segundo. Outra característica é que a interface informa também a matriz de *pixels* antes do envio para o FPGA, permitindo assim uma verificação dos dados antes e após a transmissão. A Figura 19 mostra a aba da interface com as matrizes anteriormente mencionadas, e a visualização dos *pixels* em formato de imagens. Durante os testes de transferência foram realizadas medições de tempo de envio de um conjunto de 7440 *pixels* (7440 *Bytes*), através de um osciloscópio Tektronix modelo TDS 2022B. Foram medidos os sinais de *flags* de FIFO_QuaseCheia e FIFO_Vazia, para

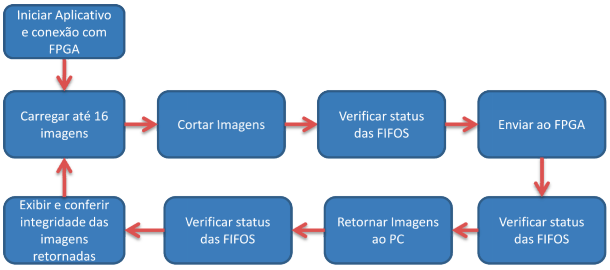

Figura 17: Esquema para teste de transferência das imagens

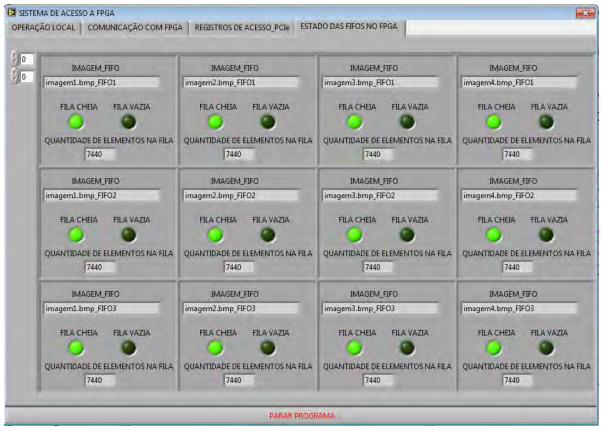

Figura 18: Verificação do status após transmissão

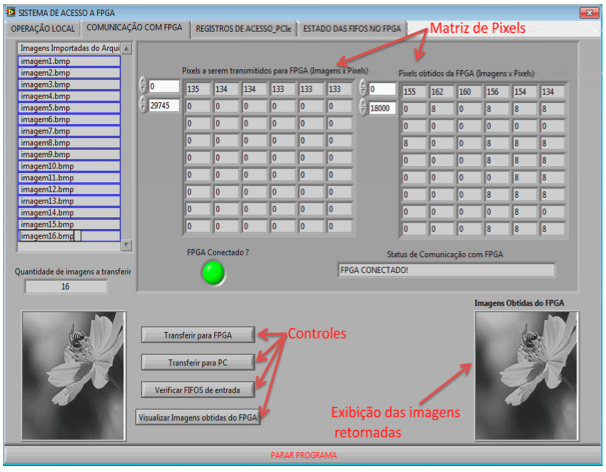

Figura 19: Sistema de acesso ao FPGA: visualização das imagens retornadas do FPG

a primeira FIFO (i.e. Imagem1_Parte1).

A Figura 20 mostra um exemplo de captura dos sinais com o osciloscópio. Nesta mesma figura o intervalo de tempo é diretamente obtido com utilização de cursores posicionados nos instantes em que sinais dos *flags* de *status* começam a mudar de nível.

Foram realizadas 100 transferências a fim de se obter uma média e desvio padrão para o tempo medido. Estes dados medidos são mostrados no Gráfico 1.

A partir dos dados coletados, o valor médio obtido para o tempo de transferência e armazenamento de um bloco

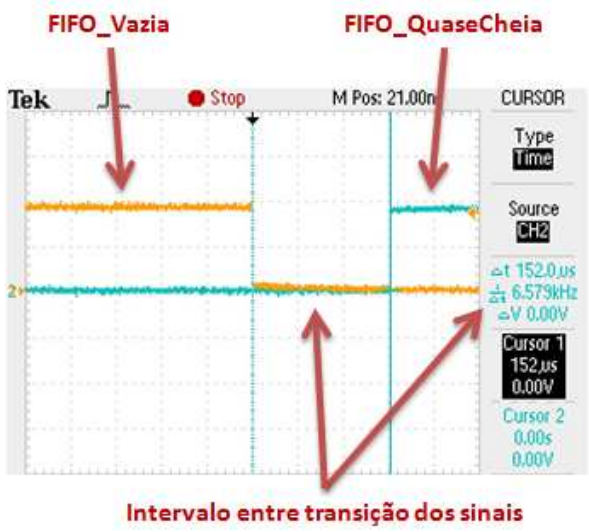

Figura 20: Medição do intervalo de tempo entre os *flags* de status

de 7440 *pixels* foi de 152,12 μs**.** Assim pode-se estimar o tempo para uma imagem completa formada por quatro blocos de 7440 *pixels* e se obter a taxa em *MBytes/s* do sistema de transferência e armazenamento de imagens. Estes dados encontram-se na Tabela 1.

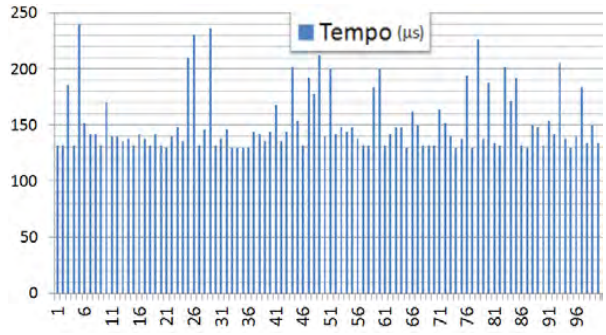

Gráfico 1 – Valores medidos para tempos de transferência de um bloco de dados com 7440 pixels (7440 bytes)

Tempo para um bloco de 7440 pixels (μs)	**152,12**
Tempo para uma imagem completa (μs)	**608,48**
Taxa de Transferência/Armazenamento (*MBytes/s*)	**48,91**

Tabela 1 – Intervalo de tempo para um bloco de 7440 pixels, para uma imagem completa (4 blocos de 7440 pixels) e a taxa de transferência e armazenamento dos dados em *MBytes/s*

O valor obtido para taxa de transferência/armazenamento de 48,91 *MBytes/s* para um bloco de 1860 elementos transmitidos é esperado quando comparado com dados de *benchmark* obtidos junto com a equipe de suporte técnico da *National Instruments*, para a transferência

de blocos de dados utilizando a função vi*MOVEOUT32*. Estes dados podem ser vistos na Tabela 2. Para correta análise deve-se ter em mente que este teste de *benchmark* é antigo, tendo sido realizado com uma plataforma computacional com *clock* da ordem de 400MHz, bem inferior ao que foi utilizado neste trabalho. O que justifica assim o maior valor encontrado.

Quantidade de elementos transmitidos	**Taxa (*MBytes/s*)**
1024	2,90
2048	5,00

Tabela 2 – Dados de *benchmark* obtidos junto à *National Instruments* para transferência de dados utilizando a função vi *MOVEOUT32*

O desvio padrão para os dados medidos é de 27,41μs, o que resulta em uma taxa de transmissão de 1643 $\pm$295 imagens/segundo.

Quando observado todo o sistema de transferência das imagens, conforme Figura 15, vê-se que o desvio padrão encontrado, na ordem de 18% do valor médio, provavelmente está associado à variação nos tempos de execução do *driver* VISA durante suas interações com o barramento PCIe. Esta variação pode ser explicada pela não utilização de DMA na transmissão dos dados, assim a cada interação do *driver* VISA para escrita de 32 bits, a CPU do computador é exigida para interface entre memória RAM e barramento PCIe, o que para sistemas não dedicados, apresenta tempos variáveis.

Além disto, a não utilização de DMA impede que o barramento PCIe que é compartilhado, seja alocado para uma transmissão contínua de todo o bloco de dados (1860 *pixels*).

Em [23] são fornecidos vários cálculos para o desempenho teórico de um barramento PCIe levando em consideração as características de *overhead* introduzidas pelo protocolo. Neste artigo alcança-se uma taxa de 200*MBytes/s/Lane* para a Gen1.0 do protocolo PCIe, quando considerado apenas o barramento de transmissão. Em [27] são apresentados dados de desempenho para o kit Cyclone IV, para transmissão via barramento PCIe entre computador e memórias embarcadas no FPGA, utilizando DMA. Neste artigo são atingidas taxas de 220*MBytes/s/Lane* para a Gen1.0 do protocolo.

Caso fosse possível atingir as taxas de transmissão mencionadas anteriormente para o presente trabalho, o *clock* interno do FPGA para armazenamento dos dados na FIFO, configurado para 125MHz, limitaria o desempenho total do sistema de transferência e armazenamento de imagens para uma taxa teórica de 4200 imagens/segundo.

6. CONCLUSÃO

Este trabalho apresenta um sistema para transferência de imagens digitais entre um computador e um FPGA através do barramento PCIe. Foram caracterizados os algoritmos necessários e as taxas de transferência máximas para utilização do mesmo em ambientes de processamento em tempo real.

Foi utilizado um kit de desenvolvimento da Altera e a linguagem de programação gráfica LabVIEW para criação de uma interface de acesso. O sistema desenvolvido permite monitorar os recursos de memória presentes no FPGA, aplicar pré-processamento (corte nas imagens) e disparar a troca de dados de imagem entre o computador e o kit utilizado.

A partir da medição do tempo de transferência de um bloco de 7440 *Bytes* (7440 *pixels*) foi estimado um intervalo de 608,48 μs para transferência de uma imagem completa formada por quatro blocos de 7440 *pixels*, o que resulta em uma taxa média de transferência e armazenamento de 1643 imagens por segundo. Deve-se destacar que estes valores podem e provavelmente serão diferentes, para diferentes plataformas computacionais utilizadas.

O protocolo PCIe Gen 1.0 promete taxas elevadas de transmissão, da ordem de 250*MBytes/s/Lane*. No entanto, para o desenvolvimento de um sistema de transferência de dados que consiga explorar de forma efetiva as reais taxas do barramento (taxas reais são mais baixas devido ao *overhead* introduzido pelo protocolo), deve-se atentar não apenas ao protocolo de transmissão utilizado, mas a todos os constituintes que formam o canal de transmissão deste sistema [27]. Por exemplo, possibilidade ou não de se utilizar DMA através do *driver* de acesso e limitações de frequência de operação para o *hardware* que irá receber os dados a partir do barramento PCIe.

A principal melhoria a ser aplicada no sistema desenvolvido, seria a utilização de um *driver* de acesso que fornecesse suporte a troca de dados por meio de DMA ou mesmo que explorasse de forma mais efetiva o *payload* de dados do protocolo PCIe disponível para o kit de desenvolvimento. No entanto, seria necessário também trabalhar com uma maior frequência de escrita dos dados nas FIFOs do FPGA. Para este fim , poder-se-ia utilizar, por exemplo, o Kit Stratix IV GX, o qual permite frequência de *clock* de até 250MHz para controle das memórias dentro do ambiente Qsys. Além disto, variações do FPGA que compõem este kit, podem alcançar até 2592kbytes de memória embarcada, o que permitiria o armazenamento em *hardware* de até 79 imagens, mais que o triplo conseguido com o atual kit.

A partir deste ponto, estando as imagens já armazenadas em hardware, é possível desenvolver módulos de processamento de imagens específicos que explorem as características de processamento paralelo e determinístico dos FPGAs, para ambientes onde se necessite comportamento de tempo real.

[1] BAILEY, D. Design for Embedded Image Processing on FPGAs. 1st. ed. [S.l.]: John Wiley & Sons (Asia), 2011. 12 p.

[2] SMITH, S. Digital Signal Processing: A Practical Guide for Engineers and Scientists. 1st. ed. [S.l.]: Newnes, 2002.

[3] ALBUQUERQUE, M. et al. A 10000 image per second parallel algorithm for real time detection of Marfes on Jet. IEEE Trans. Plasma Sci., v. 41, n. 2, p. 341-349, Feb 2013. ISSN 0093-3813.

[4] DOWNTON, A.; CROOKES, D. Parallel architectures for image processing. Electronics & Communication Engineering Journal, v. 10, n. 3, p. 139-151, Jun 1998. ISSN 0954-0695.

[5] MCKENNEY, P. Real Time vs. Real Fast How to Choose? Proceedings of the 11th Linux Symposium. Dresden, Germany: [s.n.]. 2009.

[6] LEWIS, F.; SHUZHI, G. Autonomous Mobile Robots. 1st. ed. 2006: Taylor & Francis Group.

[7] NGUYEN, D. et al. Real-time face detection and lip feature extraction using field-programmable gate arrays. IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, v. 36, n. 4, p. 902-912, August 2006.

[8] CAO, T.; ELTON, D.; DENG, G. Fast buffering for FPGA implementation of vision-based object recognition systems. Journal of Real-Time Image Processing, v. 7, n. 3, p. 173-183, April 2012. ISSN 1861-8219.

[9] DILLINGER, P. et al. FPGA-Based Real-Time Image Segmentation for Medical Systems and Data Processing. Nuclear Science, IEEE Transactions on, v. 53, n. 4, p. 2097-2101, Aug. 2006. ISSN 0018-9499.

[10] SOUZA, M. Paralelismo computacional de processamento digital de imagens aplicado à detecção de MARFEs no JET. Centro Brasileiro de Pesquisas Físicas (Dissertação de Mestrado). Rio de Janeiro. 2013.

[11] ALBUQUERQUE, M. et al. High Speed Image Processing Algorithms for Real Time Detection of MARFEs on JET. IEEE Trans. Plasma Sci, v. 40, n. 12, p. 3485-3492, Dec. 2012. ISSN 0093-3813.

[12] LIPSCHULTZ, B. et al. Marfe: an edge plasma phenomenon. Nucl. Fusion, v. 24, n. 8, 1984. [Online], Disponível: http://www.psfc.mit.edu/library1/catalog/reports/1980/83ja/83ja033/83ja033_full.pdf.

[13] SHANLEY, T.; ANDERSON, D.; BUDRUK, R. PCI Express System Architecture. [S.l.]: Addison-Wesley, 2003.

[14] ALTERA CORPORATION. Cyclone IV GX FPGA Development Board - Reference Manual. San Jose. 2010.

[15] ALTERA CORPORATION. Quartus II Handbook Version 12.0. Sao Jose. 2012.

[16] BOTROS, N. HDL Programming Fundamentals: VHDL and Verilog (Davinci Engineering). 1st. ed. [S.l.]: Charles River Media, 2005.

[17] NATIONAL INSTRUMENTS. Using the NI-VISA Driver Wizard and NI-VISA to Develop a PXI(e)/PCI(e) Driver in Windows. Austin. 2013.

[18] GONZALEZ, R.; WOODS, R. Digital Image Processing. 3th. ed. [S.l.]: Pearson Prentice Hall, 2008.

[19] ALTERA CORPORATION. IP Compiler for PCI Express - User Guide. San Jose. 2011.

[20] ALTERA CORPORATION. Embedded Peripherals IP - User Guide. San Jose. 2011.

[21] NATIONAL INSTRUMENTS. LabVIEW CORE I e II Manual. [S.l.]. 2012.

[22] VXIPLUG & PLAY SYSTEMS ALLIANCE AND IVI FOUNDATION. VPP-4.3: The VISA Library 5.1. [S.l.]. October 2012.

[23] XILINX. Understanding Performance of PCI Express Systems - WP350 (v1.1). [S.l.]. 2008.

[24] BITTNER, R. Speedy Bus Mastering PCI Express. 22nd International Conference on Field Programmable Logic and Applications (FPL 2012). [S.l.]: [s.n.]. 2012.

[25] QIANG, W. et al. The research and implementation of interfacing based on PCI express. Electronic Measurement & Instruments, 2009. ICEMI '09. 9th International Conference on. Beijing: IEEE. 2009. p. 3-116 - 3-121.

[26] NATIONAL INSTRUMENTS. NI-VISA User Manual. [S.l.]. 2001.

[27] ALTERA CORPORATION. PCI Express High Performance Reference Design. San Jose. 2014.

Notas Técnicas é uma publicação de trabalhos técnicos relevantes, das diferentes áreas da física e afins, e áreas interdisciplinares tais como: Química, Computação, Matemática Aplicada, Biblioteconomia, Eletrônica e Mecânica entre outras.

Cópias desta publicação podem ser obtidas diretamente na página web http://notastecnicas.cbpf.br ou por correspondência ao:

Centro Brasileiro de Pesquisas Físicas
Área de Publicações
Rua Dr. Xavier Sigaud, 150 – 4º andar
22290-180 – Rio de Janeiro, RJ
Brasil
E-mail: socorro@cbpf.br/valeria@cbpf.br
http://www.biblioteca.cbpf.br/index_2.html

Notas Técnicas is a publication of relevant technical papers, from different areas of physics and related fields, and interdisciplinary areas such as Chemistry, Computer Science, Applied Mathematics, Library Science, Electronics and Mechanical Engineering among others.

Copies of these reports can be downloaded directly from the website http://notastecnicas.cbpf.br or requested by regular mail to:

Centro Brasileiro de Pesquisas Físicas
Área de Publicações
Rua Dr. Xavier Sigaud, 150 – 4º andar
22290-180 – Rio de Janeiro, RJ
Brazil
E-mail: socorro@cbpf.br/valeria@cbpf.br
http://www.biblioteca.cbpf.br/index_2.html