Sistema Integrado para o Processamento do Filtro de Difusão Anisotrópica em FPGA

System-on-Chip for the Processing of the Anisotropic Diffusion Filter in FPGA

Felipe Viel[1], Guilherme Frederico Weidle Jr.[2], Cesar Albenes Zeferino[1]

[1] University of Vale do Itajaí

Itajai, SC, Brazil 88302-202

[2] Intelbras S/A

São José, SC, Brazil 88104-800

felipeviel@edu.univali.br, guilherme.weidle@intelbras.com.br, zeferino@univali.br

Abstract. *The anisotropic diffusion filter is a noise filtering technique used in the preprocessing of digital images. This filter uses complex operators and has a high computational cost so that coprocessors can be used to speed up its execution. In this work, we present the integration of an anisotropic diffusion filter coprocessor implemented in a SoC FPGA. The implemented system includes an ARM processor responsible for the communication interface with a personal computer. A parallelism exploration was performed in the coprocessor implementation, as well a comparison with the performance of a software implementation running on an ARM processor and in the personal computer. Results demonstrate that the FDA coprocessor achieves the smaller processing time than other processors, even though the coprocessor operates at a much lower clock frequency.*

Resumo. *O filtro de difusão anisotrópica é uma técnica de filtragem de ruídos utilizada no pré-processamento de imagens digitais. Esse filtro utiliza operadores complexos e tem um alto custo computacional de forma que coprocessadores podem ser utilizados para acelerar a sua execução. Nesse contexto, neste trabalho foi realizada a integração de um coprocessador do filtro de difusão anisotrópica implementado em um sistema integrado em FPGA. O sistema inclui ainda um coprocessador ARM com interface de comunicação para um computador pessoal. Foi realizada a exploração de paralelismo na implementação do coprocessador, bem como a comparação com o desempenho de implementações por software executadas no ARM e no computador pessoal. Os resultados demonstraram que o coprocessador consegue obter tempos de processamento menores que os dos demais processadores, mesmo trabalhando a frequências de operação bem inferiores.*

I. INTRODUÇÃO

O processamento digital de sinais (DSP – Digital Signal Processing) é a área que estuda sinais no tempo discreto (em função de variáveis discretas) e os sistemas utilizados para processá-los por meio de técnicas e hardware digitais [1]. O Processamento Digital de Imagens (PDI) é uma área de estudo dentro da área de DSP.

Segundo [2], um sistema de PDI realiza o processamento de imagens digitalizadas e é aplicado sobre imagens capturadas em diferentes espectros de ondas eletromagnéticas. Esses espectros incluem tanto as bandas visíveis ao olho humano, quanto as que não são perceptíveis. A área de aplicação de um sistema de PDI define quais faixas de espectro de onda eletromagnética serão utilizadas e, consequentemente, define quais técnicas devem ser aplicadas [2].

O PDI é executado em três etapas: pré-processamento, processamento e pós-processamento. A etapa de pré-processamento realiza a filtragem da imagem, enquanto a etapa de processamento é responsável por manipular a imagem e capturar informações de interesse da aplicação. Já a etapa de pós-processamento é responsável por tomar decisões baseada nas informações adquiridas no processamento [2].

O FDA (Filtro de Difusão Anisotrópica) foi proposto por [3] e é um filtro importante e amplamente usado como etapa de pré-processamento. Vários autores apresentam variações do FDA. Em [4], é descrita uma variação do FDA para a remoção de ruídos e destacamento de borda por meio do esmaecimento da imagem em escala de cinza. Suas áreas de aplicação incluem controle de qualidade em ambiente industrial, monitoramento de ambientes para fins de segurança e avaliação de imagens radiológicas.

Dada a sua importância e seu custo computacional elevado devido às operações e aos processos utilizados, alguns trabalhos buscam acelerar o algoritmo por meio da sua implementação em hardware.

Em [5], é apresentado um coprocessador de FDA implementado em FPGA (Field Programmable Gate Array) que utiliza uma máscara de 3x3 para cálculo dos novos pixels. No trabalho, o autor buscou minimizar a latência de processamento por meio da replicação de blocos de operação, ao preço de um alto custo de silício.

Em [6], foi realizada uma implementação alternativa do coprocessador de [5] com o objetivo de reduzir o seu custo de silício por meio do compartilhamento de operadores e da sequencialização das operações. Embora o desempenho desta implementação seja inferior, conforme apontado pelo autor, seu menor custo permite que seja obtida uma aceleração por meio da replicação do bloco de processamento mais crítico ou ainda pelo uso de múltiplas instâncias do processador. Uma análise teórica feita pelo autor estima uma aceleração de quase três vezes em relação ao coprocessador de [5].

O presente trabalho baseia-se na implementação apresentada em [6] e buscou preencher algumas de suas lacunas (indicadas como trabalhos futuros). A primeira delas diz respeito à integração do coprocessador a um sistema integrado em FPGA com interface de comunicação com um computador pessoal (PC – Personal Computer). Já a segunda refere-se à análise experimental do desempenho do coprocessador em hardware, bem como a exploração do paralelismo como estratégia para acelerar seu desempenho.

O restante deste artigo está organizado em seis seções. A Seção II descreve o filtro de difusão anisotrópico, enquanto a Seção III discute os trabalhos relacionados. A Seção IV apresenta o coprocessador FDA utilizado no trabalho. Já a Seção V descreve a infraestrutura de computação desenvolvida no trabalho, enquanto a Seção VI apresenta e discute os resultados experimentais. Por fim, na Seção VII, são apresentadas as conclusões do trabalho.

II. FILTRO DE DIFUSÃO ANISOTRÓPICO

O FDA é um algoritmo que aplica uma convolução no pixel da imagem, suavizando os pixels de regiões internas e inibindo esse processo nos pixels de borda. A convolução possui um núcleo adaptativo responsável por realizar uma suavização seletiva em imagens [4]. Proposto inicialmente por [3] como um modelo isotrópico, o FDA tem o objetivo de inibir os problemas de borramento e localização. Ele é usado amplamente na etapa de pré-processamento, removendo ruídos e melhorando o processo de detecção de bordas. A ideia principal do filtro é realizar um processo orientado por propriedades locais da imagem, no qual pixels de uma região são usados para modificar os demais pixels dessa região, como, por exemplo, a atenuação de ruído.

O FDA é um algoritmo que realiza um processamento intenso, decorrente dos cálculos matemáticos envolvidos sob cada pixel da imagem. Esse processamento elevado se torna proporcional ao tamanho da imagem, sendo potencializado pelo número de iterações (rodadas) em que o algoritmo é usado para se atingir o resultado buscado.

O FDA é calculado como uma máscara de convolução aplicada sobre cada ponto da imagem em um processo iterativo. Os parâmetros que configuram a execução do algoritmo são: o número de iterações (i) e a intensidade da difusão (λ). O esmaecimento da imagem é proporcional ao valor de λ. Assim, quando se busca um maior esmaecimento da imagem, aumenta-se a o valor de λ [7].

Duas matrizes são utilizadas para calcular o novo valor de cada pixel: M (matriz que contém os valores dos pixels vizinhos ao pixel calculado) e W (*kernel*, contendo pesos relativos à aplicação). Uma operação de convolução é realizada entre M e W para se obter o novo valor do pixel central, mostrado em (1).

$$P = M * W \tag{1}$$

sendo que:

$$M = \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & P & p_5 \\ p_6 & p_7 & p_8 \end{bmatrix} \tag{2}$$

$$W = \begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & wC & w_5 \\ w_6 & w_7 & w_8 \end{bmatrix} \tag{3}$$

A matriz W é o tensor de difusão e seus coeficientes são calculados para cada pixel e a cada iteração com o uso de (4), assim atualizando os valores do kernel no tempo de execução [4][7].

$$w(\lambda,\tau,\alpha,\beta)_{\mathrm{i}} = \frac{1-e^{-8*\tau*e^{-\left(\frac{\left|\sqrt[5]{\alpha-\beta}\right|}{\lambda}\right)^{5}}}}{8} \qquad (4)$$

Em (4), α representa o pixel central que terá o valor atualizado $f(x, y)$, β é um dos oito pixels vizinhos ao pixel central e τ, segundo [6], é definido como o tamanho do passo de tempo [4]. Essa última variável decorre da necessidade de se emular um processo físico de transporte de massa por um processo iterativo de estabilização em que τ é a variação de tempo t utilizada para difusão. Ela é definida no intervalo de 0 a 0,5, como:

$$\tau = \Delta t\,(t \mid 0 < t < 0.5) \qquad (5)$$

O coeficiente central wC de W é dado por (6):

$$wC = \sum_{i=1}^{8} (1 - W_{\mathrm{i}}) \qquad (6)$$

Considerando o mapeamento de pontos da imagem em um plano cartesiano, o tensor de difusão W é calculado usando (7) para o pixel que terá o valor atualizado.

$$W = \begin{vmatrix} (w_1(\lambda,\tau,x-1,y+1)) & (w_2(\lambda,\tau,x,y+1)) & (w_3(\lambda,\tau,x+1,y+1)) \\ (w_4(\lambda,\tau,x-1,y)) & (wC) & (w_5(\lambda,\tau,x+1,y)) \\ (w_6(\lambda,\tau,x-1,y-1)) & (w_7(\lambda,\tau,x,y-1)) & (w_8(\lambda,\tau,x+1,y-1)) \end{vmatrix} \qquad (7)$$

III. TRABALHOS RELACIONADOS

Nesta seção, são apresentados trabalhos que implementam o filtro de difusão anisotrópica em FPGA. Os trabalhos abordam o ganho de desempenho ao se acelerar o processamento em hardware comparando com implementações em software, por meio de abordagens em linguagem C e MATLAB.

Em [8], os autores utilizaram FPGA para paralelizar o processamento de imagens nos filtros FDA e Sobel. O FDA realiza até cem iterações e processa imagens 2D de 256x256 pixels em escala de cinza. Foi verificado em testes que três instâncias do FDA operando a 100 MHz apresentam melhor resultado que uma operando a 300 MHz.

Em [9], os autores implementaram em FPGA um FDA para destacar objetos em imagens capturadas por próteses de retina. A versão implementada é a de [3]. O FDA foi implementado utilizando dois buffers com dois módulos Sobel simples modificados. Nos testes, foram usadas imagens 2D em escala de cinza, variando de 16x16 a 512x512 pixels. O FDA foi descrito em Verilog. A implementação em FPGA, operando a 40 MHz, teve uma aceleração superior à 10 vezes ao algoritmo em MATLAB, variando com o tamanho da imagem.

Em [10], os autores implementaram um FDA modificado em FPGA para melhorar a imagem de impressão digital. Técnicas de exploração de paralelismo espacial

e temporal, representação em ponto fixo e normalização da imagem foram usadas para ganho de performance. As imagens utilizadas nos testes foram em 2D, em escala de cinza e variando de 288x348 a 640x480 pixels. Foi obtida uma aceleração de 308 à 340 vezes em relação ao filtro em MATLAB e de 28 a 30 vezes em relação ao filtro em C.

Em [11], os autores apresentaram uma implementação em FPGA do FDA com o processo de convolução modificado. A convolução foi substituída por uma subtração, economizando nove multiplicações e oito somas por iteração. Foi utilizada representação em ponto fixo para aumento de desempenho e diminuição da complexidade. Nos testes, foram usadas imagens 2D, em escala de cinza e variando de 150x150 a 1024x1024 pixels. A ferramenta Xilinx DSP blockset foi usada para implementar o filtro. Foi obtida uma aceleração de 3 a 91 vezes comparado ao filtro em software, dependendo do tamanho da imagem.

Em [12], os autores implementaram um FDA para processamento de imagens 2D e 3D em escala de cinza. Na implementação em FGPA, os autores utilizaram as ferramentas Xilinx System Generator e MATLAB Simulink para gerar as funções de difusão em hardware, que são executadas em paralelo. A implementação em FPGA foi integrada com o software MATLAB para controle. Foram utilizadas imagens variando de 128x128 a 512x512 pixels. A ênfase do trabalho estava no processo de remoção de ruído e, assim, as bordas das imagens foram preservadas de forma muito eficiente

A Tabela I apresenta uma síntese dos trabalhos relacionados e posiciona este trabalho em relação a eles. A comparação leva em conta o tipo de imagem usado, as abordagens adotadas para comparação, o uso de paralelismo e a realização de uma integração com comunicação com PC ou processador embarcado.

Tabela 1 - Síntese e comparação dos trabalhos relacionados

Trabalhos	Tipo Imagem	Comparação	Uso de Paralelismo	Integração
[8]	2D e escala de cinza	Instâncias do FDA	Instâncias do FDA	Não
[9]	2D e escala de cinza	MATLAB	Blocos internos	Não
[10]	2D e escala de cinza	MATLAB e C	Paralelismo espacial e temporal	Não
[11]	2D e escala de cinza	Software em C	Blocos internos	Não
[12]	2D, 3D e escala de cinza	MATLAB	Paralelismo espacial	MATLAB
Este Trabalho	2D e escala de cinza	Software em C	Blocos internos e instâncias	PC e ARM

Como em todos os trabalhos analisados, este trabalho desenvolve um sistema para processamento de imagens 2D, em escala de cinza, e faz comparação com implementações de referência em software. Todos os trabalhos utilizam paralelismo, havendo replicação de blocos internos, de operadores e de instâncias do processador dedicado. Não foram identificados, em nenhum trabalho, a integração e o uso do FDA como coprocessador por um sistema embarcado completo. Também não foram observadas comparações do desempenho com arquiteturas de processadores. Outro aspecto que deve ser destacado é que, enquanto os trabalhos analisados utilizam representação de números reais em ponto fixo, o coprocessador usado neste trabalho [6] utiliza representação em ponto flutuante de acordo com o padrão IEEE 754 de precisão simples (32 bits). Isso confere maior qualidade à imagem, equivalente à obtida por um processador de propósito geral usado em PCs.

IV. COPROCESSADOR FDA

O coprocessador FDA é um núcleo composto de uma unidade de controle e uma unidade de processamento (caminho de dados), associados à lógica de interfaceamento (*wrapper*). A unidade de controle gerencia a operação do caminho de dados, o qual é formado por componentes que implementam as equações do filtro descritas na seção anterior. O *wrapper* possui registradores para armazenar os parâmetros do filtro (i e λ), as dimensões da imagem, estruturas de controle que indicam o início e fim do processamento e uma memória para armazenamento da imagem a ser manipulada.

Os componentes de processamento que formam o caminho de dados do FDA são o *Calc_qX*, *Calc_qC* e *Calc_newC*, os quais compartilham duas unidades de aritmética em ponto flutuante (FPU – Floating Point Unit). Esses componentes também são organizados em um bloco de controle e um caminho de dados. Inicialmente, o componente *Calc_qX* calcula o coeficiente de difusabilidade de cada um dos oito pixels vizinhos do pixel central utilizando (4). Após isso, o componente *Calc_qC* aplica (6) para calcular o valor do coeficiente de difusibilidade do pixel central. Por fim, o componente *Calq_newC* calcula o novo valor do pixel central utilizando (1), o qual é armazenado em um *buffer* temporário. A Fig. 1, apresenta o diagrama de blocos simplificado do coprocessador FDA descrito em [6].

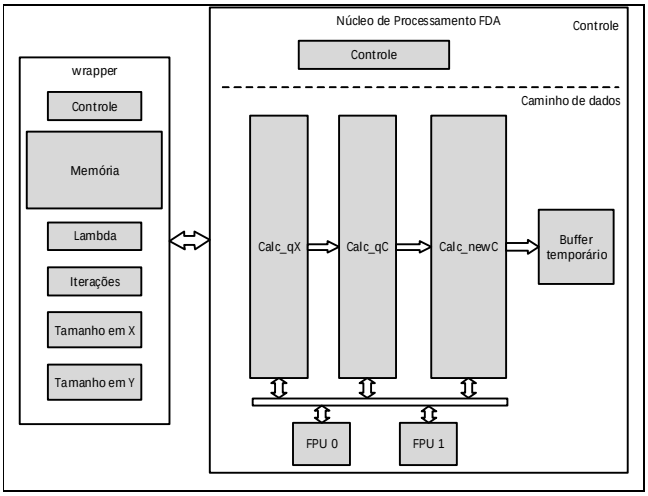


Fig. 1.Diagrama de blocos do coprocessador FDA [6]

Dos três blocos principais do caminho de dados do coprocessador de FDA, o mais crítico é o *Calc_qX*, pois é utilizado oito vezes para calcular o coeficiente de difusibilidade dos pixels vizinhos.

Neste trabalho, é realizada a avaliação do desempenho do coprocessador FDA descrito em [6] em um sistema integrado implementado em FPGA e a exploração de paralelismo com a processador e também com o uso de instâncias adicionais do bloco *Calc_qX*. Para fins de referência, o coprocessador original será referenciado como Single FDA.

V. INFRAESTRUTURA DE COMPUTAÇÃO

A infraestrutura de computação desenvolvida neste trabalho se divide entre software de controle executando em um PC e um sistema integrado em FPGA incluindo, principalmente, um processador ARM e o coprocessador FDA. Esse sistema foi prototipado no Kit de Desenvolvimento DE1-SoC da Terasic.

A. Software de controle no PC

Para realizar a transferência de imagens entre o PC e o Kit de Desenvolvimento DE1-SoC, optou-se pela comunicação via interface de rede Ethernet com o uso do protocolo SSH, nativo em distribuições Linux, dada à maior largura de banda em relação às demais alternativas disponíveis no kit. No computador pessoal, foi utilizada a aplicação PuTTY, juntamente com as aplicações *stand-alone* PSCP e PLINK. O PuTTY é um cliente que implementa os serviços do protocolo SSH, entre outros. Já o PSCP e o PLINK implementam, respectivamente, os serviços do protocolo SSH de envio e recebimento de arquivos e execução de comandos no destino. Essas duas aplicações foram utilizadas como base para uma aplicação de interface com usuário. A Fig. 2 ilustra a comunicação entre o PC e o kit DE1-SoC.

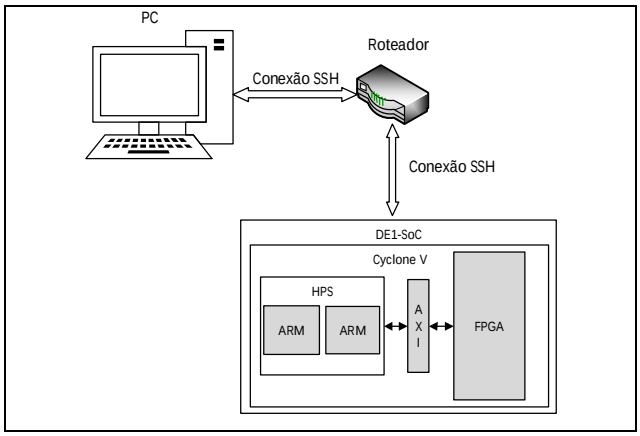


Fig. 2. Diagrama de blocos exemplificando conexão SSH entre PC e DE1-SoC

Para enviar as imagens a serem processadas na DE1-SoC, foi desenvolvida uma aplicação com interface gráfica para o sistema operacional Windows 10 de 64 bits. A aplicação foi escrita em C++ e com o uso do *framework* Qt para construção da interface, a qual é apresentada na Fig. 3.

A aplicação desenvolvida conta com campos para parâmetros de configuração do FDA e informações de *login* (necessários para autenticação pelo protocolo SSH) da distribuição Linux rodando na DE1-SoC. A aplicação permite o envio somente de imagens no formato PGM (Portable Graymap) para a DE1-SoC, sendo uma restrição do software executando no Kit. Inicialmente, é chamada a aplicação PSCP, que recebe como parâmetros as informações de *login*, endereço de origem, endereço de destino e o nome da imagem que se deseja processar. Em seguida, após a confirmação do envio da imagem, é então chamada a aplicação PLINK.

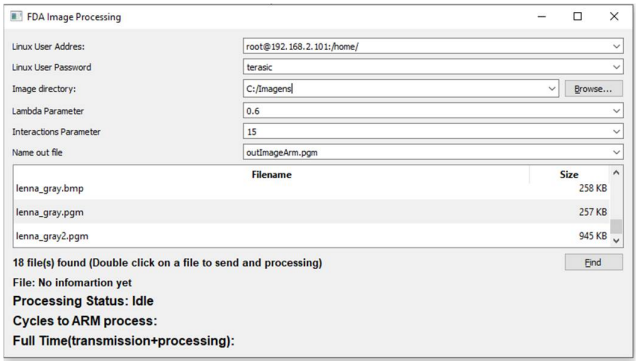


Fig. 3.Interface gráfica da aplicação desenvolvida para comunicar PC com o Kit de desenvolvimento DE1-SoC.

Essa aplicação recebe por parâmetro as informações de *login*, o endereço e o nome da aplicação na DE1-SoC que interage com o FDA em FPGA, além dos parâmetros de configuração do filtro. Quando há a confirmação do término da filtragem, a aplicação PSCP é chamada para buscar a imagem de saída. A aplicação PuTTY foi utilizada para depurar e realizar testes e configurações diretamente na DE1-SoC, a fim de deixar estáveis as aplicações que rodam no kit de desenvolvimento e garantir a correta transmissão de dados e comandos.

B. Integração software/hardware no SoC FPGA

Após a definição da interface e do protocolo de comunicação, foi desenvolvido o sistema integrado proposto. O dispositivo disponível no kit DE1-SoC (Cyclone V modelo 5CSEMA5F31C6) é organizado em duas partes: HPS (Hard Processor System) e uma região de lógica programável (o FPGA). O HPS contém dois *hard cores* ARM e diversos periféricos. O FPGA possui 32.070 ALMs (Adaptative Logic Module), 4.065.280 bits de memória embutida e 87 blocos de DSP (utilizados para a implementação de operadores aritméticos). A interconexão entre o HPS e o FPGA é feita por meio de um barramento AMBA AXI.

Para o desenvolvimento do sistema, foram utilizados o software de projeto de lógica programável Intel/Altera Quartus II 13.1 e a ferramenta Qsys (integrada ao Quartus II). Essa ferramenta foi utilizada para instanciação de componentes de interface que são ligados ao barramento AMBA AXI, gerando endereços para posterior acesso em software. Além do processador ARM, os componentes principais instanciados foram blocos de PIO (Parallel IO) e de PLL (Phase-locked Loop). Os PIOs foram utilizados para conectar e permitir acesso do coprocessador FDA pelo ARM via software, sendo

utilizado no mínimo nove para um FDA e acrescentando-se mais cinco a cada instância adicional. Já o PLL é utilizado para aumentar a frequência de operação obtida de um cristal oscilador de 50 MHz.

Ao gerar a descrição do SoC, a ferramenta Qsys cria um arquivo que é utilizado para gerar biblioteca, para linguagem C e com extensão ".h", que contém informações e endereços de acesso de cada componente conectado ao barramento. Para isso, foi utilizado o Nios II Command Shell, um terminal de execução de comandos disponível junto à ferramenta Qsys.

O *wrapper* do coprocessador FDA serve como meio para integração e comunicação com o HPS. Ele é conectado aos PIOs e ao PLL, abstraindo a interface do FDA. A Fig. 4 apresenta um diagrama de blocos que ilustra como os componentes se conectam.

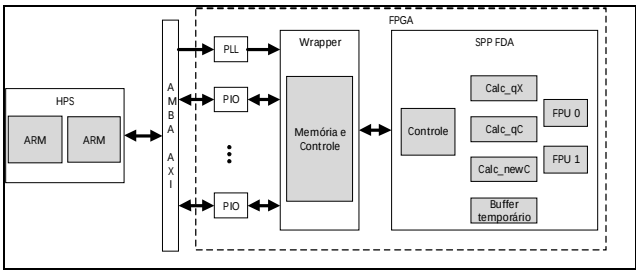


Fig. 4. Diagrama de blocos do Single FDA em hardware

Foi desenvolvida uma aplicação embarcada em linguagem C para execução sobre o sistema operacional Linux Ubuntu 16.04.1 LTS DE1 - SoC (GNU/Linux 4.5.0) de 32 bits em um dos processadores ARM do HPS. O compilador cruzado da ferramenta Intel SoC Embedded Developement Suite (EDS) 16.0 foi utilizado para gerar o arquivo executável dessa aplicação. O código da aplicação utiliza funções de biblioteca para: (*i*) obter informações e endereços dos componentes instanciados no Qsys; (*ii*) abrir e salvar imagens no formato PGM; e (*iii*) acessar o barramento AMBA AXI.

A aplicação embarcada primeiramente realiza leitura da imagem no formato PGM. Após, calcula um endereço virtual que é usado para gerar os ponteiros para acesso, leitura e escrita, de cada PIO utilizado como interface com o *wrapper*. A imagem, já com a borda virtual, e os parâmetros são então gravados no *wrapper* do coprocessador FDA, o qual então inicia o processamento. Quando detectado o término da filtragem, a imagem é então lida do *wrapper* e salva no cartão SD (Secure Digital), na DE1-SoC pelo ARM, pronta para ser recuperada pela aplicação no PC.

Esse desenvolvimento, tanto da aplicação de controle em C quanto o sistema em hardware, foi baseado em um exemplo disponibilizado por [13] como material de apoio para desenvolvedores. Após essa versão do sistema com um FDA (Single FDA), foram desenvolvidas outras três versões para medir o impacto de diferentes organizações do sistema e do FDA com exploração de paralelismo. Duas versões do sistema são apresentadas em seguida, com dois FDAs (Dual FDA), Fig. 5, e outra com quatro FDAs (Quad FDA), Fig. 6, executando em paralelo.

As versões Dual FDA e Quad FDA necessitaram de mais PIOs e de uma reestruturação da aplicação de controle em C, a qual divide a imagem em regiões processadas por cada coprocessador. A memória usada para armazenar a imagem teve seu tamanho dividido entre a quantidade de FDAs utilizados, sendo que cada FDA utilizou seu próprio *wrapper*.

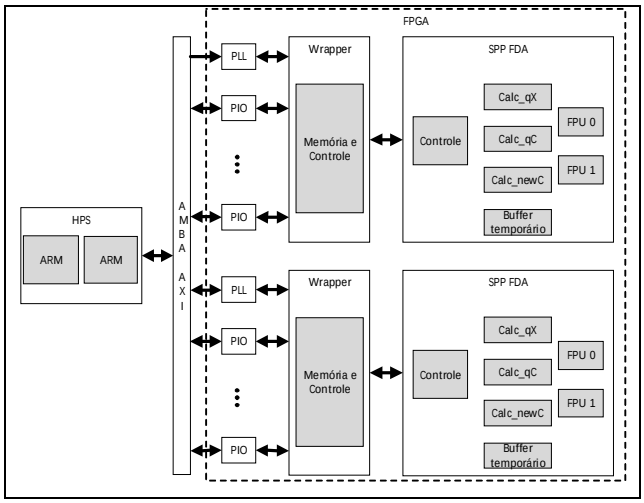


Fig. 5. Diagrama de blocos do Dual FDA.

Uma terceira versão do sistema (Fig. 7) foi desenvolvida, a Single FDA – Quad Calq_qX, na qual o bloco *Calc_qX* é replicado em quatro instâncias para acelerar o cálculo dos coeficientes de difusibilidade dos pixels vizinhos ao pixel central. Nessa versão, não foi necessário adicionar PIOs e nem modificar o software em C usado na versão Single FDA.

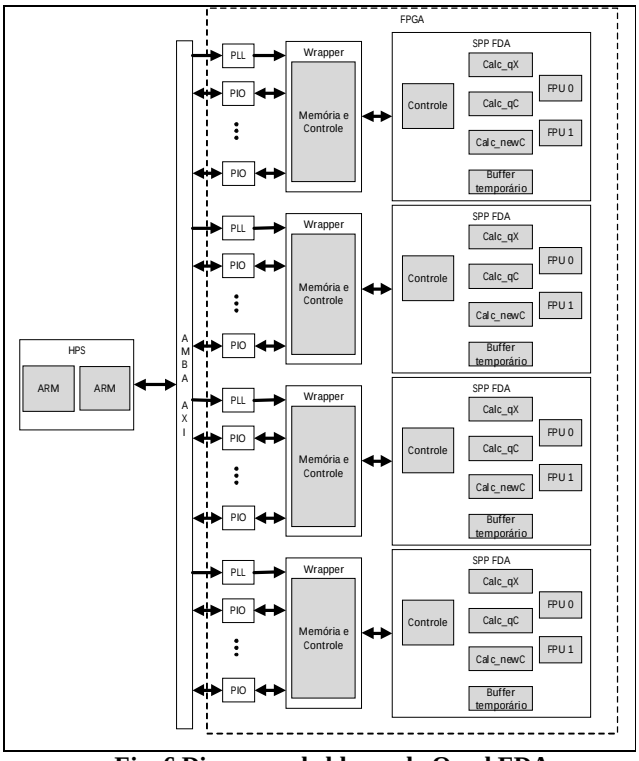


Fig. 6.Diagrama de blocos do Quad FDA

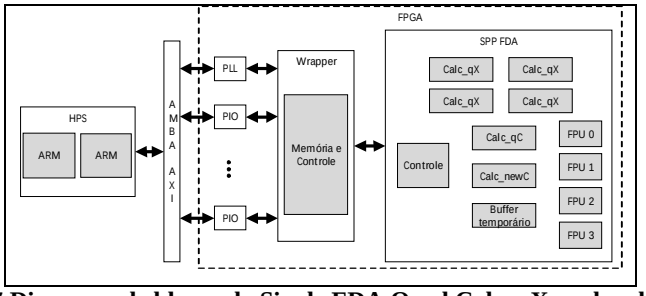


Fig. 7.Diagrama de blocos do Single FDA Quad Calc_qX em hardware

VI. RESULTADOS

Para avaliação do desempenho das diferentes implementações realizadas, foram coletados os tempos para filtragem de três imagens 2D 100x100 em tons de cinza com ruídos. A imagens escolhidas foram a Lena (Fig. 8.a), uma imagem de referência na área de PDI, e duas imagens de raio X, sendo uma imagem de um tórax humano e outra imagem de um cérebro. Nas duas imagens de raio X, foi adicionado ruído para prejudicar a sua nitidez e testar a real efetividade do FDA.

Para fins de comparação, com soluções em software, foram coletados os tempos da execução do algoritmo em um PC, no ARM e nas diferentes versões do coprocessador FDA. O PC utilizado foi um *laptop* com processador Intel Core i5 4210U de 1.7 GHz rodando o sistema operacional Microsoft Windows 10 de 64 bits. Já os processadores do

HPS são ARMs Cortex-A9 que operam a 800 MHz. Neles, é executado o sistema operacional embarcado Ubuntu 16.04 de 32 bits.

Para avaliar a aplicação no PC, foi utilizada a biblioteca *windows.h* e a função de contagem de tempo *GetProcessTime*(). Essa função retorna o tempo de execução de trechos de código de interesse com precisão de milésimos de segundo. Na DE1-SoC, por utilizar uma distribuição Linux embarcada, optou-se por usar a função *clock_gettime*() da biblioteca *time.h* nos trechos de código de interesse. Essa função retorna o tempo de execução com precisão de nanosegundos.

A Tabela II apresenta o tempo de execução do FDA para as três imagens em três iterações com λ igual a 15. A latência de processamento apresentada não inclui o tempo para transferência da imagem entre o ARM e o coprocessador FDA. Em média, são gastos 20 ms para envio da imagem de entrada e leitura da imagem de saída do FDA.

Tabela II. Latência de processamento do FDA

Processador	**Fmax (MHz)**	**Tempo Lena (ms)**
Intel Core i5	1700	78,0
ARM Cortex-A9	800	234,4
Single FDA	200	241,7
Dual FDA	200	120,8
Quad FDA	130	93,0
Single FDA (Quad Calc_qX)	200	74,4

O tempo de processamento em todas as versões do FDA é igual para três imagens pois todas possuem o mesmo número de pixels. Os coprocessadores utilizados alterariam o tempo de execução somente se a quantidade de pixels das imagens ou quantidade de iterações necessárias para se obter um resultado aceitável fossem diferentes, independentemente da versão utilizada do FDA.

A Tabela II ainda apresenta a frequência de operação de cada plataforma. A versão Quad FDA do coprocessador opera a uma frequência menor que as demais versões para a assegurar a integridade da imagem. Essa limitação se justifica pela maior complexidade do sistema e do mapeamento dos circuitos no FPGA que impactam na frequência máxima de operação.

Pelos resultados, pode-se observar que a melhor abordagem de aceleração do processamento com FPGA é na versão Single FDA Quad Calc_qX. Essa abordagem consegue ser mais rápida operando a uma frequência 8,5 vezes menor que a execução do FDA em software no PC, quando desconsiderada a latência de escrita e leitura da imagem na memória presente no *wrapper*.

A Fig. 8 mostra que o resultado da execução do FDA é o esmaecimento da imagem de entrada com a remoção do ruído da imagem original (Fig. 8.a). No caso, a Fig. 8.b mostra a saída da filtragem feita no PC.

Já a Fig. 9 mostra os resultados do FDA no processador ARM e em três versões do coprocessador. Como pode ser observado, os resultados obtidos nas diferentes implementações do coprocessador são muito similares às saídas produzidas pela execução em software. Como apontado em [6], comparando-se a filtragem realizada no PC e no hardware dedicado, a diferença média é de apenas 0,497 (dos 10.000 pixels da imagem, 4.967 pixels diferiam em apenas uma unidade.

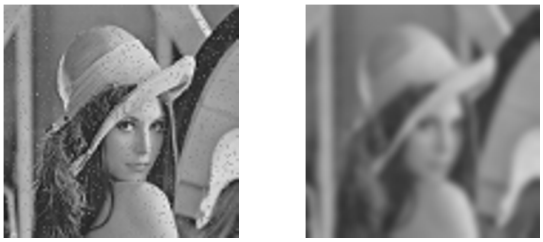


Fig. 8. Processamento do FDA: (a) Imagem de entrada ruidosa; (b) imagem de saída esmaecida

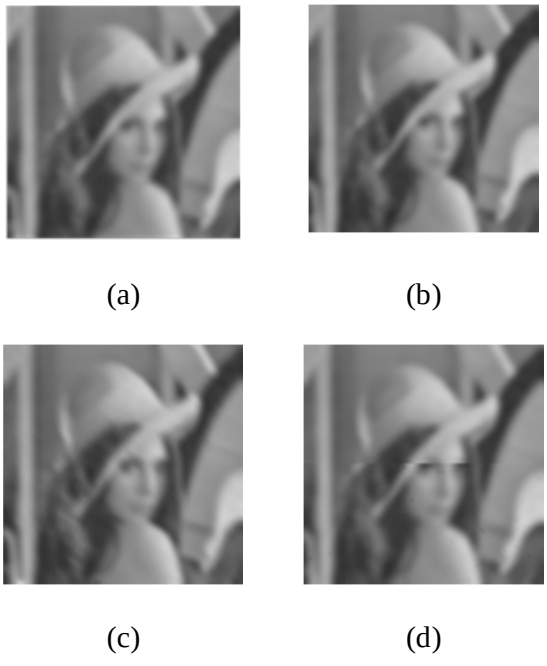


Fig. 9.Imagem de saída: (a) do ARM; (b) do Single FDA; do Single FDA Quad Calc_qX; e (d) do Dual FDA

No entanto, como pode ser observado na Fig. 9.d, a divisão da imagem 100x100 em duas imagens 100x50 distribuídas entre os dois coprocessadores (Dual FDA) realçou

a fronteira que divide essas duas imagens menores. Isso ocorre porque a implementação original do coprocessador não considerou a necessidade de compartilhamento dos pixels da fronteira. No caso, cada coprocessador processa os pixels dessa fronteira sem levar em conta os pixels vizinhos pertencentes à parte da imagem alocada ao outro coprocessador.

O realce das fronteiras provocado pelas implementações Dual e Quad FDA impacta em aplicações que necessitam destacar elementos da imagem (pontos, bordas). Na Fig. 10, por exemplo, são mostradas as saídas do Single FDA e do Quad FDA (que particiona a imagem 100x100 em quatro imagens 50x50) e o resultado da aplicação do filtro Sobel, em software, para detecção de bordas. Observa-se que a qualidade do processamento da imagem é comprometida. Na Fig. 10.d, é observado que há um realce nas fronteiras das regiões pelo filtro Sobel, resultado da divisão da imagem para processamento em múltiplos coprocessadores. Esse problema será tratado em um trabalho futuro.

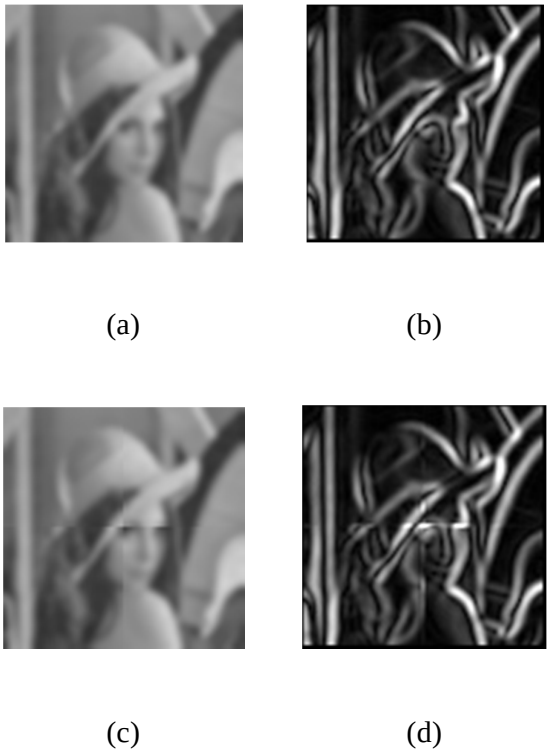
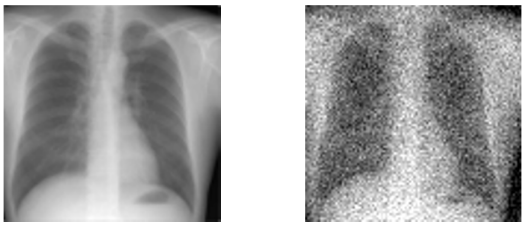


Fig. 10. Efeito do particionamento da imagem: (a) saída do Single FDA; (b) resultado da detecção de bordas com o filtro Sobel; (c) saída do Quad FDA; e (d) resultado da detecção de bordas com o filtro Sobel.

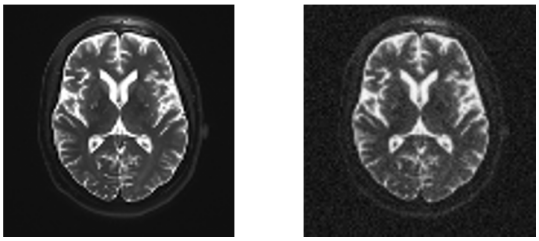
A Fig. 11 e a Fig. 12 apresentam as imagens de raio X antes e depois da aplicação do ruído. Esse ruído dificulta a observação de detalhes presentes na imagem, bem como, a detecção de borda por algoritmos como Sobel. Assim, como realizado na imagem Lena, Fig. 10, as duas imagens de raio X, com aplicação de ruído e filtrada com o FDA, também passaram pelo filtro de detecção de borda Sobel para ressaltar a importância da

remoção de ruído. Nessa comparação, a versão do FDA utilizada foi a Single FDA Quad Calq_qX. A Fig. 13 e a Fig. 14 apresentam a comparação dos resultados obtidos.



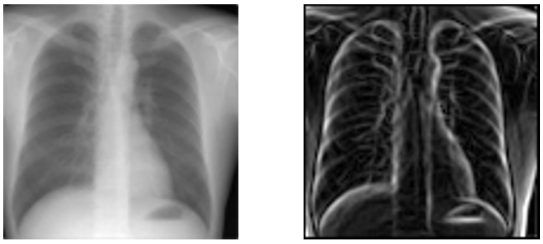
(a) (b)

Fig. 11. Aplicação de ruído na imagem do tórax: (a) imagem original; e (b) imagem pós aplicação de ruído.

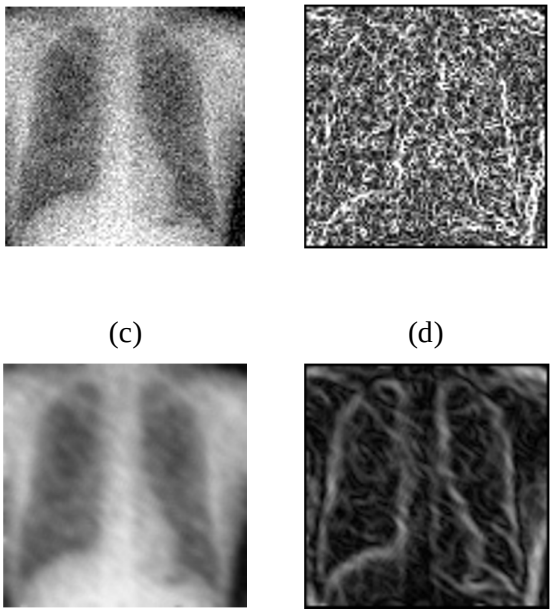


(a) (b)

Fig. 12. Aplicação de ruído na imagem do cérebro: (a) imagem original; e (b) imagem pós aplicação de ruído.



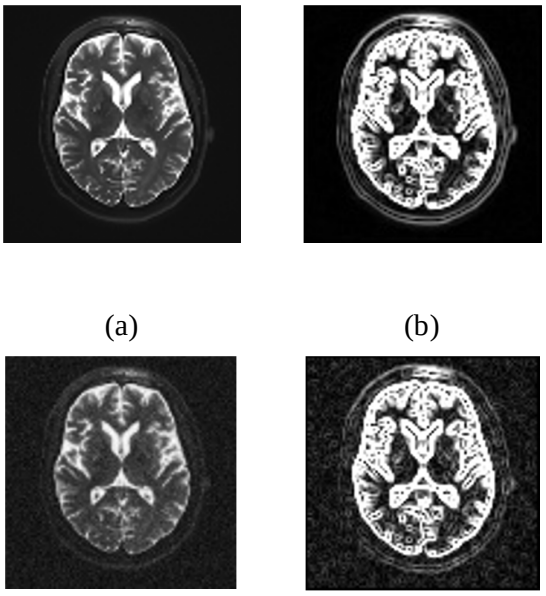
(a) (b)



(c) (d)

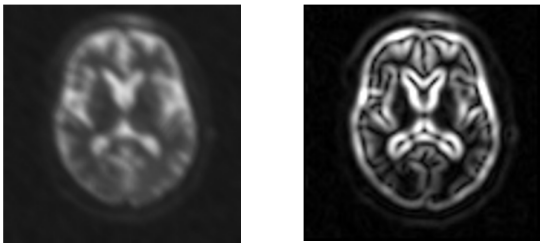
(e) (f)

Fig. 13. Efeito do FDA em imagens de raio X do torax: (a) original; (b) saída do filtro Sobel com (a); (c) com ruído; (d) saída do filtro Sobel com (c); (e) com ruído pós Single FDA Quad Calc_qx; e (f) saída do filtro Sobel com (e).



(a) (b)

(c) (d)



(e) (f)

Fig. 14. Efeito do FDA em imagens de raio x do cérebro: (a) oiginal; (b) saída do filtro Sobel com (a); (c) com ruído; (d) saída do filtro Sobel com (c); (e) com ruído pós Single FDA Quad Calc_qx; e (f) saída do filtro Sobel com (e).

A Tabela III apresenta a taxa de ocupação de recursos do FPGA pelas diferentes implementações do coprocessador. Como pode ser observado, o principal fator que limitou o número de instâncias do coprocessador foi a quantidade de blocos de DSP disponível no FPGA. Outro aspecto a ser destacado é que a versão Single FDA Quad Calc_qX, que teve o melhor desempenho, teve custo inferior à solução com quatro instâncias de FDA. Além disso, as versões Dual e Quad FDA deverão consumir mais recursos lógicos para resolver o problema decorrente do particionamento da imagem.

Tabela III. Taxas de ocupação dos recursos do FPGA

Recurso	**Coprocessador**			
	Single FDA	**Dual FDA**	**Quad FDA**	**Single FDA Quad Calq_qx**
Módulos de lógica adaptativa	25%	40%	70%	41%
Blocos de bits de memória	40%	41%	41%	41%
Blocos de DSP	24%	48%	97%	75%

VII. CONCLUSÕES

Este trabalho descreveu o projeto, a implementação e a avaliação de sistemas computacionais que integram um ou mais coprocessadores especializados na filtragem por difusão anisotrópica em FPGA com processadores ARM, permitindo também a utilização do sistema embarcado por um PC. No trabalho, foram desenvolvidas duas aplicações, uma de interface e controle para execução em PC e outra para execução no processador embarcado. A solução desenvolvida permite transferir a imagem para o dispositivo e executar o filtro seja no processador ARM, seja nos coprocessadores.

Os resultados demostraram a efetividade de toda a infraestrutura implementada. Com os resultados, observou-se a importância da exploração do paralelismo espacial na

redução da latência causada pelo processamento, tanto de instâncias do coprocessador quanto de componentes que possuem um alto custo computacional. Observou-se que, apesar do sistema implementado operar em uma frequência menor e ser menos complexo que processadores usados em PCs, consegue apresentar um resultado aceitável e realiza o processamento mais rápido.

Um aspecto a ser discutido refere-se à eficiência energética e a viabilidade de uso do coprocessador em sistemas dedicados. A solução integrada em FPGA possui um custo energético menor que uma implementação em PC, pois opera a uma frequência de relógio 8,5 vezes menor e possui tempo de processamento similar, mesmo se for considerada a latência da transferência da imagem. Logo, a solução desenvolvida tem potencial para uso em sistemas embarcados que necessitem filtrar imagens para realizar algum processamento.

Das configurações analisadas, a que apresentou melhor desempenho foi a que utilizou um único coprocessador e quatro instâncias do bloco *Calc_qX*. No entanto, o espaço de projeto a ser explorado permite inferir que outras configurações que apliquem paralelismo no coprocessador e no bloco *Calc_qX* possam obter desempenho superior.

Como trabalhos futuros, pretende-se resolver o problema do compartilhamento dos pixels de fronteira e adotar a representação em ponto fixo para reduzir o custo do circuito e aumentar o desempenho da solução. A partir disso pretende-se explorar outras configurações de paralelismo espacial, bem como investigar a possibilidade de utilizar técnicas de *pipeling*, para explorar o paralelismo temporal.

AGRADECIMENTOS

Os autores agradecem ao apoio da Capes – Coordenação de Aperfeiçoamento de Pessoal de Nível Superior.

REFERÊNCIAS

[1] P. S. R. Diniz, E. A. B. da Silva, and S. L. Netto, Processamento Digital de Sinais: Projeto e Amilise de Sistemas, Porto Alegre: Bookman, 2014.

C.R. Gonzalez and E.R. Woods, “Processamento de Imagens Digitais”,

Ed. Edgard Blücher, Brazil, 2000

C.R. Gonzalez and E.R. Woods, “Processamento de Imagens Digitais”,

Ed. Edgard Blücher, Brazil, 2000

[2] R. C. Gonzalez and R. E. Woods, Digital Image Processing (*2nd Edition*). Prentice Hall, 2002.

[3] P. Perona and J. Malik, “Scale-Space and Edge Detection Using Anisotropic Diffusion,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 12, no. 7, pp. 629-639, July 1990.

[4] J. Weickert, Anisotropic Diffusion in Image Processing. B.G. Teubner Stuttgart, 1998.

[5] A. A. Stein, “Filtro de difusão anisotrópica em FPGA,” Master’s thesis, Univali, 2015, *In Portuguese*.

[6] G. F. Weidle. “Desenvolvimento de um Processador do Filtro de Difusão Anisotrópica em FPGA”. Trabalho de Conclusão de Curso (Especialização em Sistemas Embarcados), Univali, São José, 2017.

[7] L. Coser, “Filtro de difusão anisotrópico orientado por evidência de borda,” Master’s thesis, UFSC, 2009, *In Portuguese*.

[8] W. Atabany and P. Degenaar, “Parallelism to reduce power consumption on fpga spatiotemporal image processing,” in *IEEE Int. Symp. On Circuits and Systems - ISCAS* 2008. IEEE, 2008, pp. 1476–1479.

[9] M. Al Yaman, W. Al-Atabany, A. Bystrov, and P. Degenaar, “FPGA design for dual-spectrum visual scene preparation in retinal prosthesis,” in *36th Annual Int. Conf. of the IEEE in Engineering in Medicine and Biology Society* (*EMBC*). IEEE, 2014, pp. 4691–4694.

[10] T. M. Khan, D. G. Bailey, M. A. Khan, and Y. Kong, “Efficient hardware implementation strategy for local normalization of fingerprint images,” *J. of Real-Time Image Processing*, pp. 1–13, 2016.

[11] C. Pal, A. Kotal, A. Samanta, A. Chakrabarti, and R. Ghosh. “An efficient FPGA implementation on optimized anisotropic diffusion filtering on images,” in *International Journal of Reconfigurable Computing*, 2016:1,2016.

[12] S. Saha, M. Roy, M. Tithi Dey, and A. Chakrabarti. “An efficient fpga implementation of anisotropic diffusion filter on 2- dimensional and 3-dimensional images,” in *International Conference on Computer, Electrical & Communication Engineering (ICCECE).* IEEE, 2016, pp. 1–5.

[13] Terasic, ‘DE1-SoC: My first HPS-FPGA’, 2014. [Online]. Avaible: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=205&No=83 6&PartNo=4 . [Accessed: 5-May-2017].

[14] S. Tatham, O. Dunn, B. Harris, J. Nevis. ‘PuTTY’, 2017. [Online]. Avaible: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.htm l . [Accessed:10-May-2017].

[15] Terasic, ‘DE1-SoC: User manual’, 2014. [Online]. Avaible: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=205&No=83 6&PartNo=4 . [Accessed: 5-May-2017].