Avaliação de uma Arquitetura FPGA com Múltiplos Processadores para Sistemas de Processamento de Imagens Digitais
Evaluation of a Multiple Processor FPGA Architecture for Digital Image Processing Systems

Lucas Martins Mendes
Departamento de Metal-Mecânica
Instituto Federal de Santa Catarina (IFSC)
Campus Florianópolis - Brasi
lucas.m11@aluno.ifsc.edu.br

Maurício Edgar Stivanello
Departamento de Metal-Mecânica
Instituto Federal de Santa Catarina (IFSC)
Campus Florianópolis - Brasi

Manoel Kolling Dutra
Departamento de Metal-Mecânica
Instituto Federal de Santa Catarina (IFSC)
Campus Florianópolis - Brasi

ABSTRACT

Uma questão muito importante no desenvolvimento de um sistema de visão computacional é a escolha da plataforma computacional a ser utilizada. Existem várias alternativas baseadas em diferentes tecnologias, como os tradicionalmente utilizados processadores de propósito geral, processadores de sinais digitais e os dispositivos reconfiguráveis. A utilização de FPGA, onde os algoritmos são implementados em hardware, apresenta uma série de vantagens em função do desempenho. Porém, os algoritmos devem ser implementados em linguagens específicas, o que dificulta a reutilização de bibliotecas e código legado implementado em outras plataformas. No presente trabalho é apresentado o estudo e validação de uma arquitetura que emprega múltiplos processadores NIOS sintetizados em FPGA no processamento de imagens digitais. Os resultados demonstram a viabilidade da utilização de uma abordagem empregando múltiplos processadores NIOS para os casos onde a implementação dos algoritmos diretamente em hardware não seja adequada.

ABSTRACT

A very important issue in the development of a computer vision system is the choice of which computing platform to use. There are several alternatives based on different technologies, such as the commonly used general-purpose processors, digital signal processors, and FPGA. The use of FPGA, where algorithms are implemented in hardware, has several performance advantages. However, the algorithms must be re-implemented in specific languages and using different concepts and resources, making it difficult to reuse libraries and legacy code traditionally implemented on other platforms. This paper presents the study and validation of an architecture that employs multiple Soft Processors on FPGA for digital image processing. The results demonstrate the feasibility of using this approach for cases where the implementation of algorithms directly in hardware is not adequate or feasible.

CCS Concepts

•**Computing methodologies** → *Image processing;* •**Hardware** → Reconfigurable logic and FPGAs;

Keywords

Sistema de visão computacional; FPGA; processadores NIOS; imagens digitais
Computer vision system; FPGA; NIOS processors; digital images

1. INTRODUÇÃO

Na indústria de transformação, é essencial detectar os defeitos nos estágios iniciais do processo. Na produção de tecidos, por exemplo, uma série de defeitos podem surgir nos processos de fiação e tecelagem [13]. No corte de tábuas de madeira, defeitos como furos, trincas e nós são frequentes [12]. Na indústria de envase ou de enlatados, defeitos nos rótulos, lacres ou mesmo a presença de contaminantes podem ocorrer [11]. A detecção prévia deste tipo de defeito permite a atuação direta na linha de produção, evitando que um material defeituoso seja repassado aos processos de maior valor agregado, reduzindo assim os refugos e aumentando o rendimento.

A demanda por inspeção de qualidade em 100% dos produtos manufaturados tem exigido a automatização de etapas de inspeção em processos de produção nas diferentes indústrias. Uma ferramenta muito utilizada na inspeção de produtos é a visão computacional, que emprega imagens e algoritmos de processamento de imagens digitais na análise de características dos produtos a fim de detectar defeitos de forma automatizada [10] [14]. Porém, inspeções cada vez

mais complexas, com requisitos de tempo e de sincronização cada vez maiores, tornam a implementação destes sistemas um desafio tecnológico, e exigem que tanto os componentes de software como de hardware sejam criteriosamente selecionados e adaptados.

No desenvolvimento de um sistema de inspeção que atenda aos requisitos descritos, uma questão muito importante é a escolha da plataforma computacional a ser utilizada. Existem várias alternativas baseadas em diferentes tecnologias, como os tradicionalmente utilizados processadores de propósito geral (GPP - General Purpose Processor), processadores de sinais digitais (DSP - Digital Signal Processor) e os dispositivos reconfiguráveis (FPGA – Field Programmable Gate Arrays).

Processadores de propósito geral empregados em computadores convencionais têm sido utilizados na implementação de diversos sistemas de visão. Porém, não são soluções bem dimensionadas visto que o software associado ao sistema de inspeção executa sobre um sistema operacional de propósito geral que traz consigo uma série de serviços e processos não utilizados para tal tipo de aplicação. Por sua vez, a tecnologia FPGA permite implementar sistemas dedicados muito bem dimensionados, onde apenas o fluxo de processamento relacionado ao processamento de imagens e controle são executados. Adicionalmente, tais plataformas são particularmente interessantes no desenvolvimento de sistemas no contexto da Indústria 4.0 devido aos diversos recursos e interfaces disponíveis, que permitem atender aos requisitos de alta conectividade entre dispositivos e sistemas.

Entretanto, a implementação em FPGA é consideravelmente mais especializada e custosa visto que todo o código legado, documentação e bibliotecas de sistemas de visão computacional existentes para plataformas convencionais não podem ser diretamente utilizados. Uma alternativa para utilizar-se de todos os recursos legados de outras plataformas é a sintetização de processadores NIOS sobre a malha FPGA. Tal recurso permite combinar blocos de processamento totalmente implementados em FPGA com rotinas implementadas em linguagens convencionais, como C ou C++. Esta abordagem torna-se ainda mais interessante devido ao fato de que é possível sintetizar múltiplos processadores em um mesmo projeto.

Apesar de que o emprego de recurso de paralelismo para aplicações de processamento de imagens não seja novidade, a grande maioria dos trabalhos encontrados apresenta soluções utilizando processadores de propósito geral, GPU ou FPGA [7]. Não foram encontrados trabalhos na literatura que proponham uma arquitetura baseada em FPGA que se utilize de múltiplos processadores NIOS.

No presente trabalho propõe-se o estudo e validação de uma arquitetura que empregue múltiplos processadores NIOS sintetizados em FPGA no processamento de imagens digitais. Na Seção 2 é apresentado o fluxo básico de processamento de um sistema de processamento de imagens de propósito geral. Na Seção 3 são apresentados conceitos fundamentais sobre a tecnologia FPGA e o processador NIOS. Na Seção 4 é apresentada a arquitetura proposta para o estudo pretendido. Na Seção 5 são apresentados os resultados experimentais. Na Seção 6 são discutidas as conclusões e perspectivas para trabalhos futuros.

2. FLUXO DE PROCESSAMENTO BÁSICO DE UM SISTEMA DE PROCESSAMENTO DE IMAGENS PARA INSPEÇÃO AUTOMATIZADA

Na Figura 1 é apresentado o fluxo básico de um sistema de processamento de imagens para inspeção automatizada. Tal sequência de etapas pode ser generalizado para diferentes aplicações, onde apenas as técnicas de processamento de imagens empregadas em cada etapa variam.

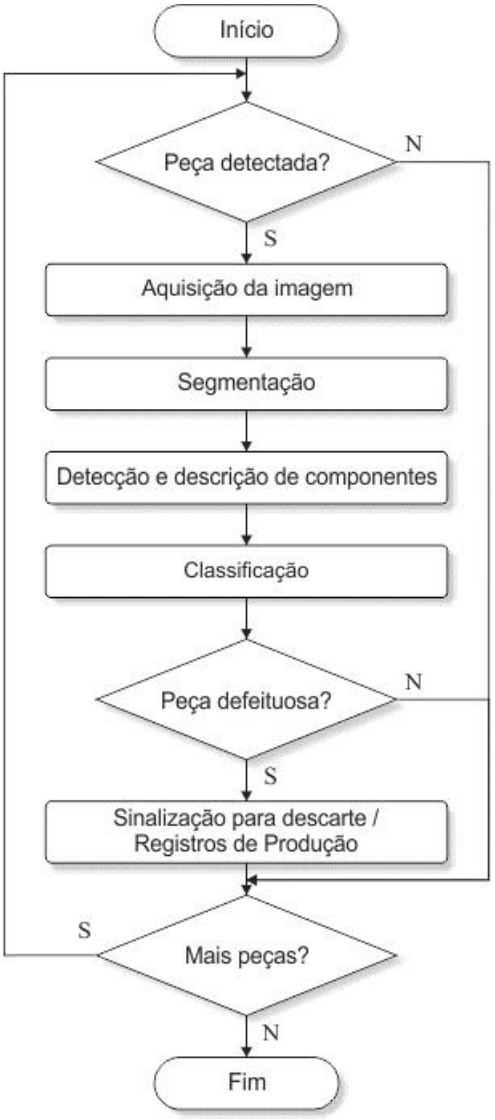


Figure 1: Fluxo de processamento de para sistemas de inspeção automatizada baseado em imagens

No fluxo de processamento exemplificado, quanto uma dada peça a ser inspecionada é apresentada ao sistema, é realizada a aquisição de uma imagem. Em seguida, realiza-se um processamento denominado segmentação, onde o objetivo é separar os pixels pertencentes aos objetos de interesse dos objetos pertencentes ao fundo.

Na sequência, realizam-se a detecção e descrição dos agrupamentos de pixels resultantes da segmentação. Os agrupa-

mentos de pontos que representam os objetos são então classificados em função dos requisitos do domínio de aplicação, e sinais de descarte ou rotinas de registro são executados com base nos resultados obtidos.

3. A TECNOLOGIA FPGA, O PROCESSADOR NIOS E SUAS APLICAÇÕES NO PROCESSAMENTO DE IMAGENS DIGITAIS

FPGA é um dispositivo que possui componentes lógicos reconfiguráveis, como portas lógicas, somadores e blocos de memória. Estes elementos podem ser interconectados a fim de formar circuitos para uma dada aplicação, isto permite prototipagem rápida de circuitos digitais complexos, reduzindo custo e tempo de desenvolvimento. É altamente flexível, em contraste com os circuitos integrados de aplicação específica (ASICs - Application Specific Integrated Circuits), devido à sua programabilidade e infra-estrutura [15] [6]. Uma tendência tem sido incluir outros dispositivos digitais e analógicos especializados no chip para permitir projetos de sistema-em-um-chip (SoC - System-on-a-chip), como processadores de sinais digitais (DSP), núcleos de processador, gerenciadores de clock, e memória [9].

Os dispositivos FPGA têm se mostrado bastante atrativos para aplicações que exijam processamento em tempo real, baixo consumo de energia e tamanho reduzidos. O desenvolvimento em tais dispositivos é possível através da utilização de diferentes kits disponíveis no mercado. Na Figura 2 é apresentado o kit de desenvolvimento DE1-SoC da Intel. Tais kits trazem de maneira integrada ao dispositivo FPGA uma série de periféricos, memórias e interfaces de comunicação. Além dos kits de desenvolvimento, também estão disponíveis diferentes acessórios opcionais, como a câmera D8M apresentada na mesma imagem, e que pode ser utilizada no desenvolvimento de projetos de visão computacional e processamento de imagens.

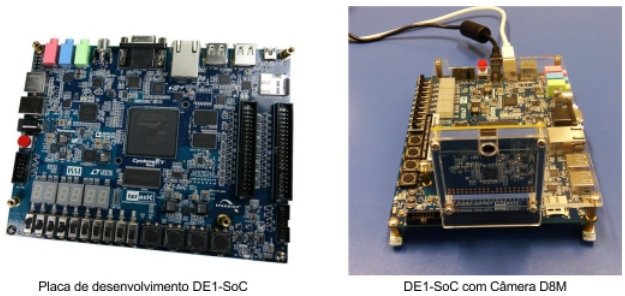


Figure 2: Placa de desenvolvimento DE1-SoC e câmera D8M

FPGAs representam tecnologia de computação reconfigurável, que é de certa forma muito adequada para algoritmos de processamento de imagens. Especificamente para aplicações envolvendo processamento de imagens, sucesso tem sido alcançado na implementação de algoritmos específicos como filtragem [2], extração de características [4], segmentação e localização de objetos [8]. Os algoritmos implementados nestes sistemas utilizam em muitos casos a abordagem de processamento de dados através da técnica de pipeline, onde os algoritmos são convertidos para implementações de hardware através de linguagens específicas como Verilog ou VHDL, e onde o fluxo de pixels é propagado através de diferentes elementos de computação [5]. Nessa abordagem, pode-se efetuar computações de forma paralela, obtendo resultados as vezes virtualmente "instantâneos", mas, a implementação destes algoritmos é lenta e custosa, por ser uma arquitetura completamente diferente.

Entretanto, diversos algoritmos de visão computacional, como é o caso da segmentação de objetos baseado no crescimento de regiões, não obedecem a estrutura pré-determinada e sequencial de acesso e processamento dos pixels conveniente para tal abordagem de implementação. Para estes casos, desta forma, o uso de GPPs pode ser mais conveniente. Na direção de se utilizar das vantagens de cada uma das arquiteturas, foi introduzido a possibilidade da implementação de processadores via componentes lógicos no FPGA.

Recentemente, com o crescimento das capacidades dos FPGA's, tornou-se possível a sintetização de microprocessadores ou Soft Processors. Soft Processors são microprocessadores que podem ser completamente implementados utilizando os diferentes dispositivos semicondutores contendo lógica programável encontrados nos FPGA's. Através de tais microprocessadores pode-se reaproveitar o legado de algoritmos e ferramentas desenvolvidos para processadores tradicionais. Fabricantes como a Xilinx e a Altera (Intel) tem inserido no mercado diversas plataformas utilizando esta solução [8] [3].

O processador NIOS II da Altera é um processador RISC (Reduced Instruction Set Computer) com uma arquitetura Harvard, onde há a possibilidade de uma máquina digital armazenar seus programas no mesmo espaço de memória que os dados, podendo assim manipular tais programas [1]. Muitos parâmetros arquitetônicos podem ser personalizados em seu tempo de design. O usuário pode decidir entre sua versão "f" (Fast), que, em troca de custar mais recursos lógicos do dispositivo, possui performance melhor ou "e" (Economy) que atua em sentido oposto. Pode-se além disso configurar vários parâmetros internos do processador como caches de instruções e dados, interfaces de depuração e acelerações de hardware para certas instruções.

4. METODOLOGIA UTILIZADA

No contexto do presente trabalho pretendeu-se realizar uma avaliação da utilização de uma arquitetura com múltiplos processadores NIOS na implementação de algoritmos de processamento de imagens digitais. Para isso, foi criado um projeto de hardware e software específico para este fim implementado sobre o kit de desenvolvimento DE2-115 da Intel. Os detalhes de tal projeto são apresentados na sequência.

4.1 Arquitetura Proposta

Para avaliar e comparar a performance de algoritmos de processamento de imagens quando paralelizados em processadores NIOS sobre um dispositivo FPGA, foram criadas 3 arquiteturas distintas. Para o propósito deste trabalho, foram elaboradas versões com 1, 2 e 4 processadores. A Figura 3 apresenta um diagrama funcional das arquiteturas propostas da perspectiva global.

Todas as arquiteturas utilizam um layout semelhante, assim como memória e periféricos, de forma a possibilitar a comparação e os ganhos reais de performance obtidos a partir da paralelização. Cada processador NIOS é encapsulado em um subsistema próprio, o que facilita a criação de designs com múltiplos processadores por permitir a repetição fácil

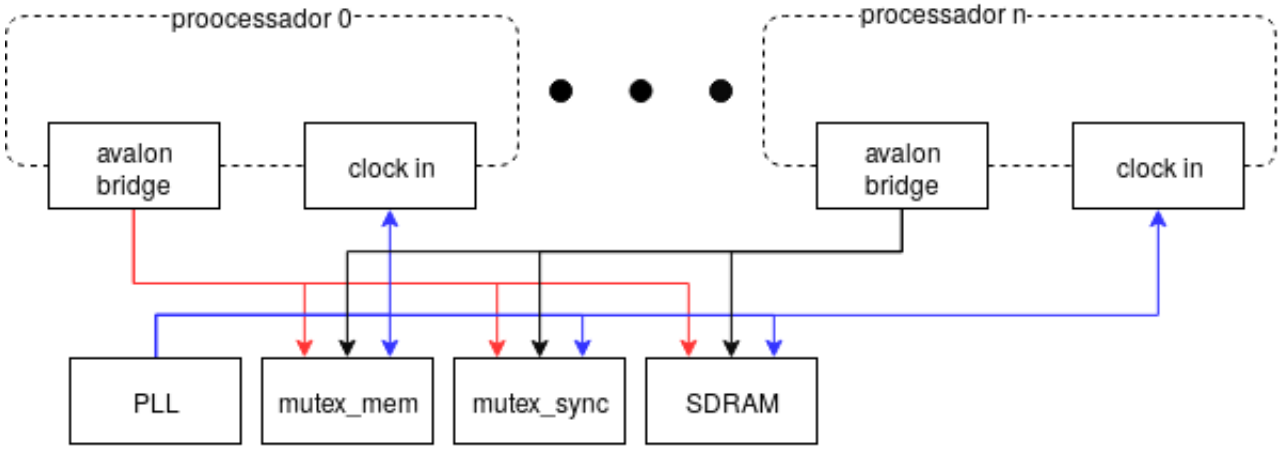


Figure 3: Diagrama das arquiteturas propostas

de um elemento. Um diagrama funcional do mesmo pode ser visto na Figura 4.

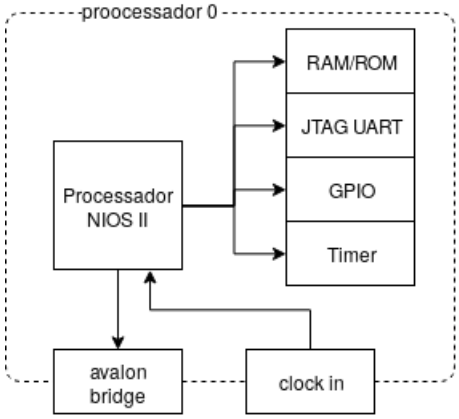


Figure 4: Subsistema do processador

Cada módulo possui os seguintes componentes, com suas respectivas funções:

- `RAM/ROM`: uma região de memória própria utilizada para conter stack+heap de cada CPU além de todas a seções do binário a ser executado como .text, .rodata, .bss, etc.;
- `Timer`: um temporizador utilizado para medidas de tempo;
- `JTAG UART`: uma interface de comunicação para depuração e transmissão de dados em geral;
- `GPIO`: portas programáveis de entrada e saída utilizadas para indicação visual do uso do *mutex mem* e também para funções de controle no processador 0, que serão explicadas posteriormente;
- `Avalon bridge`: Uma ponte avalon que intermedia o acesso a hardware externo ao subsistema atual;
- `Clock in`: uma entrada de sinal de clock, vinda do nível superior.

Como mesmo para imagens relativamente pequenas (640 X 480 pixels, escala de cinza, 8 bits de resolução) a memória disponível para cada CPU é insuficiente para carregar toda sua região de trabalho de uma vez, foi utilizado uma das interfaces de memória disponíveis no kit de desenvolvimento, uma SDRAM (Sincronous Dinamic Random Access Memory) de 64 Mbytes, para armazenar a imagem sendo processada. De forma a mediar o acesso a esta interface, além de evitar problemas de concorrência em geral, foram utilizados dispositivos de exclusão mútua (Mutex - Mutual Exclusion device), um dispositivo que permite acesso mutualmente excludente de um recurso, instanciados em hardware.

4.2 Descrição Geral da Implementação

De forma a distribuir o processamento entre as unidades, a imagem a ser processada foi dividida em regiões conforme a arquitetura avaliada, como pode ser visto na Figura 5.

De forma a atuar como região de trabalho, cada processador possui um uma matriz quadrada, alocada estaticamente, de lado aproximadamente `floor(sqrt(m))`, levando em consideração espaço para `stack+heap` e sendo `m` o total de memória disponível para um dada unidade de processamento.

Segundo o algoritmo geral adotado, cada processador carrega um bloco ou cópia de uma parte da imagem contida na SDRAM para seu espaço de trabalho local, processa o bloco ponto a ponto e por fim o escreve de volta na memória principal. O funcionamento geral de tal algoritmo é ilustrado na Figura 6.

Como a interface de memória principal é compartilhada, idealmente ela é acessada por um processador enquanto os outros estão trabalhando na imagem. Para sincronizar os processadores, foi utilizado um Mutex (*mutex sync*) para sincronizar o início e fim do processamento da imagem e um para mediar o acesso a SDRAM (*mutex mem*). Cada vez que um processador deseja acessar a memória principal, deve primeiro obter o respectivo Mutex. Quando há conflitos no acesso à memória, sempre apenas um processador obtém acesso e o outro fica bloqueado esperando a liberação do recurso desejado.

O processador "0" possui código e periféricos um pouco diferentes, principalmente o início e fim do algoritmo utilizado e algumas entradas de IO adicionais para botões e

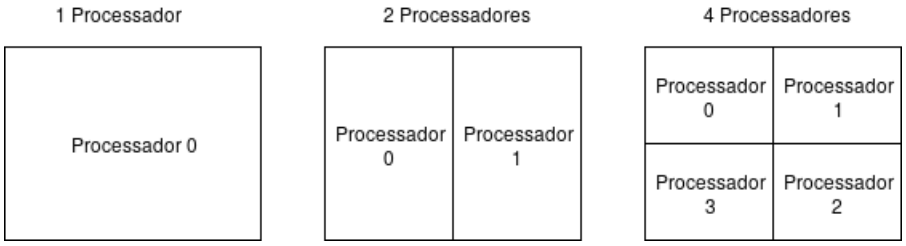


Figure 5: Separação das regiões de trabalho dos processadores

chaves utilizadas para controlar o programa, de forma que atua como "mestre", garantindo a sincronização dos processadores e preenchendo a imagem sendo processada.

5. RESULTADOS E DISCUSSÕES

Para validar a arquitetura proposta na execução de algoritmos de processamento de imagens digitais foi implementado o método de limiarização global. Tal método ou técnica é muito empregado na etapa de segmentação existente no fluxo de processamento apresentado na Figura 1, sendo utilizado para separar os pixels correspondendo aos objetos de interesse dos pixels correspondentes ao plano de fundo. Na Figura 7 é apresentado um exemplo de aplicação de tal método em um sistema de inspeção de rebites. Nesse caso, a etapa de segmentação tem como objetivo identificar os agrupamentos de pontos resultantes de componentes da tampa diferentes do fundo, dentre os quais encontra-se o rebite a ser inspecionado.

Na Figura 7(a) é apresentada a imagem original. Na Figura 7(b) é apresentado o histograma de intensidades da imagem, onde é possível observar picos em torno de intensidades associadas ao fundo da tampa e aos outros componentes presentes na mesma. Desta forma, podemos selecionar convenientemente um valor de intensidade `T` que atuará como limiar, separando tais elementos da imagem. A imagem binarizada `g(x,y)` apresentada na Figura 7(c) é resultante da segmentação da imagem original `f(x,y)` para um dado valor limiar `T`. Tal imagem pode ser obtida por:

$$g(x,y) = \{255, se f(x,y) >= T;\ 0\} \quad (1)$$

Tal algoritmo foi implementado seguindo a arquitetura e o projeto descritos na Seção 4 utilizando a linguagem C. De forma a evidenciar o ganho devido a paralelização, pois o algoritmo de limiarização global é bastante simples e rápido, foi introduzido um tempo de processamento artificial em cada ponto da imagem no valor de 1s, simulando uma operação mais complexa. Isto é utilizado para demonstrar que algoritmos que demandem mais processamento para cada ponto, como os que fazem uso de núcleos de convolução, se beneficiarão da paralelização.

O principal parâmetro avaliado foi o tempo de execução do algoritmo em cada uma das arquiteturas, que se utilizam de um diferente número de processadores. Na Figura 8 é apresentado o tempo de processamento que cada uma das arquiteturas consumiu para processar uma imagem de tamanho 640 X 480.

Como pode-se observar, há uma relação positiva entre a performance do algoritmo e o número de processadores, oferecendo para o cenário avaliado um ganho de performance

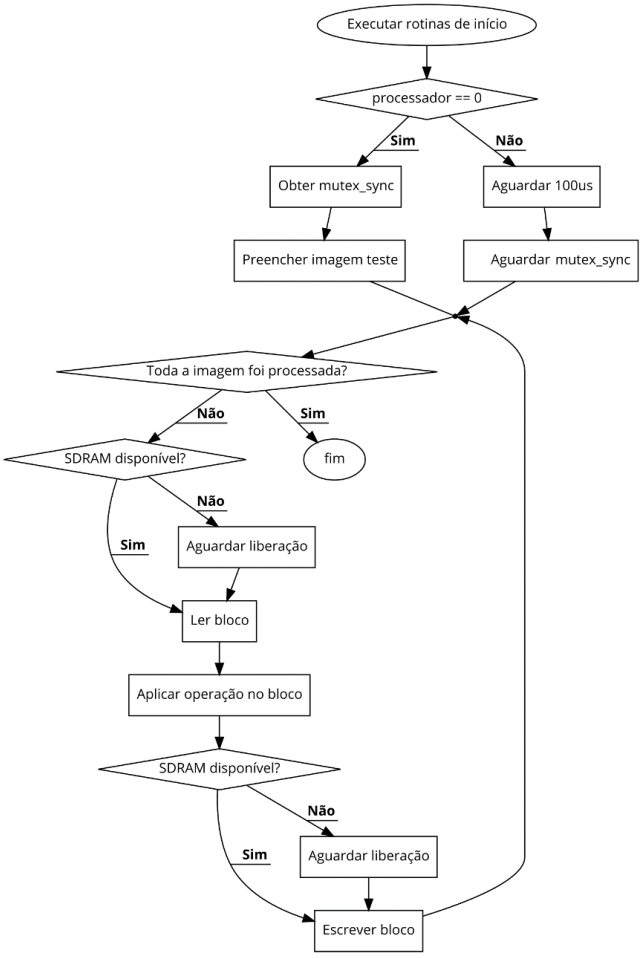


Figure 6: Fluxograma do algoritmo utilizado

próximo a `t/n`, sendo `n` o número de processadores e `t` o tempo de execução para 1 processador. O dispositivo FPGA permite a utilização de ainda mais processadores. Sabe-se, porém, observando a Lei de Amdahl, que a razão do ganho ao utilizarmos mais processadores decai, com grandes ganhos de performance de 1 para 2 e de 2 para 4, mas obtendo ganhos decrescentes com a adição de mais unidades.

Na Figura 9 é apresentado o uso da malha FPGA para cada arquitetura avaliada em percentagem dos componentes lógicos utilizados e numa escala absoluta.

É possível observar que, para o FPGA utilizado, teoricamente é possível criar arquiteturas com até 20 processadores.

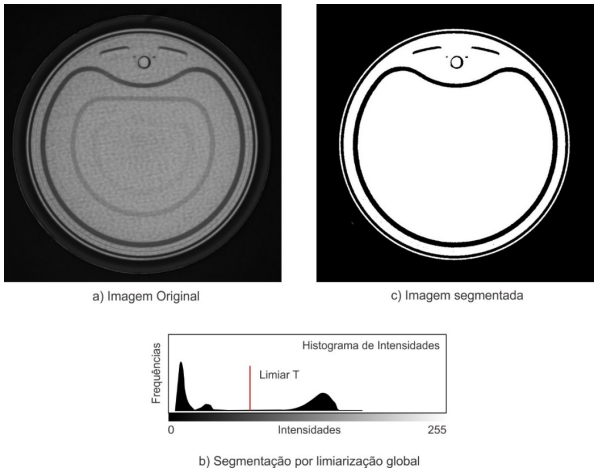


Figure 7: Exemplo de processamento de limiarização global

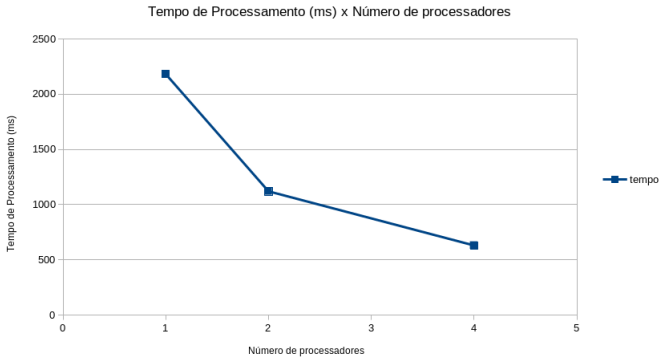


Figure 8: Tempos de processamento

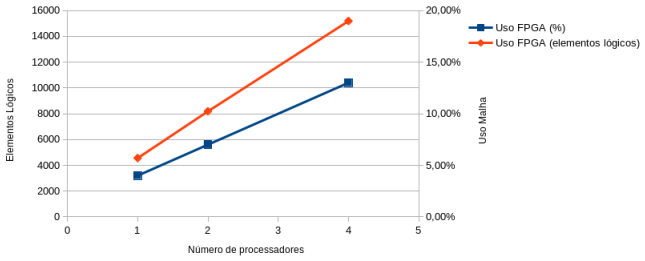
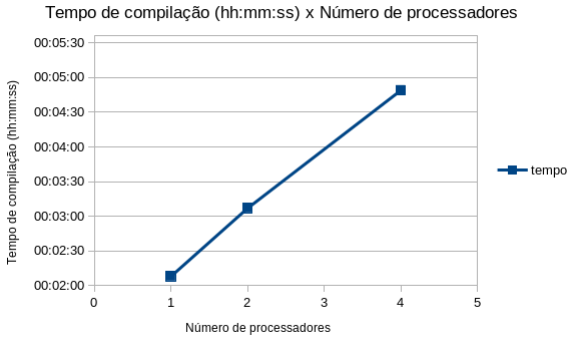


Figure 9: Uso da malha do FPGA em relação ao número de processadores

Porém, esta utilização em geral não é linear, além de que nos casos com muitos processadores começa-se a verificar problemas de fragmentação de memória. Isto advém do fato que a memória do dispositivo é dividida em blocos unitários de 9000 bits, ou seja, cada bloco de memória alocada no nível arquitetural utiliza somente blocos inteiros.

Além disso, quanto maior a utilização da malha do FPGA em geral maior o tempo de compilação, como ilustrado na Figura 5.

A avaliação do tempo de compilação foi feita utilizando o Quartus 16, com compilação paralela habilitada utilizando



Tempo de compilação em função do número de processadores

todos núcleos do processador, na sua versão para Linux Ubuntu Debian 18.04.2 LTS em um computador com as características apresentadas na Tabela 1.

Table 1: Especificações do computador de desenvolvimento

Componente	Descrição
Modelo Comercial	Lenovo ThinkCentre M900
Memória	8GiB DDR4 2133MHz
Processador	Intel (R) Core (TM) i5-6500 CPU @ 3.20GHz

6. CONCLUSÕES

No presente trabalho foi apresentada a avaliação de uma arquitetura de múltiplos processadores sintetizados em um dispositivo FPGA. Sobre esta arquitetura foi realizada a validação referente à execução de algoritmos de processamento de imagens digitais.

Os resultados mostram que é possível, diferente de outros trabalhos encontrados na literatura onde os algoritmos são implementados diretamente na malha FPGA através de linguagens como Verilog, utilizar-se de implementações tradicionais legadas fazendo-se uso de Soft Processors. Ainda, fica comprovado que a perda de performance associada a esta abordagem quando comparada à abordagem clássica pode ser reduzida ao se utilizar de múltiplos processadores.

Como trabalhos futuros pretende-se avaliar uma arquitetura que contemple todo o fluxo de um sistema de processamento de imagens, como o apresentado na Figura 1, onde cada processador seja responsável por uma das etapas presentes. Também existe a possibilidade de uma implementação híbrida, que utilize alguns elementos do FPGA, como execução de certas instruções de forma paralela e lógicas customizadas, mas ainda aproveitando as implementações e códigos de legado já existentes.

7. REFERENCES

[1] R. Cofer and B. Harding. Fpga soft processor design considerations. *Programmable Logic Design Line*, 2005.

[2] R. Dobai and L. Sekanina. Image filter evolution on the xilinx zynq platform. In *2013 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2013)*, pages 164–171, June 2013.

[3] R. C. Gonzalez and R. E. Woods. *Digital image processing*. Prentice Hall, Upper Saddle River, N.J., 2008.

[4] R. Kapela, K. Gugala, P. Sniatala, A. Swietlicka, and K. Kolanowski. Embedded platform for local image descriptor based object detection. *Applied Mathematics and Computation*, 267:419–426, 2015.

[5] T. Kryjak, M. Komorkiewicz, and M. Gorgon. Real-time hardware–software embedded vision system for ITS smart camera implemented in zynq SoC. *Journal of Real-Time Image Processing*, 15(1):123–159, May 2016.

[6] Y. Li. Fpga implementation for image processing algorithms. Technical report, EEL 6562 Course Project Report, 2006.

[7] L. M. Russo, E. C. Pedrino, E. Kato, and V. O. Roda. Image convolution processing: A gpu versus fpga comparison. In *2012 VIII Southern Conference on Programmable Logic*, pages 1–6, March 2012.

[8] P. M. Santos, J. C. Ferreira, and J. S. Matos. Scalable hardware architecture for disparity map computation and object location in real-time. *Journal of Real-Time Image Processing*, 11(3):473–485, June 2016.

[9] J. Schlessman, M. Lodato, B. Ozer, and W. Wolf. Heterogeneous mpsoc architectures for embedded computer vision. In *2007 IEEE International Conference on Multimedia and Expo*, pages 1870–1873, July 2007.

[10] C. Steger, M. Ulrich, and C. Wiedemann. *Machine vision algorithms and applications*. John Wiley & Sons, 2018.

[11] M. Stivanello, P. César, and P. Gomes. Inspeçao visual industrial automatizada por análise de forma com descritores de fourier e redes neurais artificiais. In *Anais do Seminco - Seminário de Computação da Universidade Regional de Blumenau*, 2006.

[12] M. Stivanello, G. POSSAMAI, A. VALMORBIDA, M. Roloff, and S. Vargas. Visage - sistema de inspeção de produtos por visão computacional baseado em diagrama de blocos. In *12th IEEE/IAS International Conference on Industry Applications*, 2016.

[13] M. E. Stivanello, S. Vargas, M. L. Roloff, and M. R. Stemmer. Automatic detection and classification of defects in knitted fabrics. *IEEE Latin America Transactions*, 14(7):3065–3073, July 2016.

[14] R. Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[15] B. Zitova and J. Flusser. Image registration methods: a survey. *Image and vision computing*, 21(11):977–1000, 2003.