Arquitetura Paralela e Parametrizada para Convolução de Imagens

Alexandre Marques Amaral, Milene Barbosa Carvalho, Carlos Augusto Paiva da Silva Martins
Laboratório de Sistemas Digitais e Computacionais (LSDC)
Pontifícia Universidade Católica de Minas Gerais
alexmarques@ieee.org, milene@ieee.org, capsm@pucminas.br

Resumo

O baixo desempenho e baixa flexibilidade de soluções de convolução de imagens são problemas comuns. Para solucionar estes problemas, neste artigo, apresentamos uma arquitetura parametrizada e paralela para convolução de imagens. Para verificação, codificamos em VHDL diferentes variações da arquitetura para implementação em FPGA. Resultados de ganho de desempenho e de flexibilidade são comparados com arquiteturas acadêmicas e comerciais.

1. Introdução

Nos últimos anos, imagens digitais têm sido utilizadas em muitas aplicações. Esta utilização justifica o grande desenvolvimento e evolução de sistemas de Processamento Digital de Imagens (PDI). Dentre as operações de PDI, a convolução se destaca por ser utilizada na filtragem espacial de imagens [1].

A convolução de imagens possui uma grande quantidade de sub-operações a serem executadas para a obtenção de um único ponto (*pixel*) da imagem resultante. Então, a convolução de imagens possui uma carga de processamento muito elevada. Porém, algumas sub-operações são independentes entre si, caracterizando um paralelismo implícito na operação.

À medida que as soluções que implementam convolução de imagens tornam-se mais otimizadas, as aplicações demandam tempos de resposta ainda menores. Esta diminuição dos tempos requeridos e a falta de desempenho de algumas soluções constituem um problema de desempenho. Considerando as arquiteturas tradicionais de alto desempenho, como multiprocessadores, multicomputadores, GPUs (*Graphic Processor Units*) e ASICs (*Application Specific Integrated Circuit*), observamos que muitas não possuem compromisso entre desempenho e flexibilidade que atenda às aplicações. Estas características são problemas dessas arquiteturas.

Analisando os problemas apresentados, desenvolvemos esta pesquisa com o objetivo de projetar, desenvolver e verificar uma arquitetura paralela e parametrizada, dedicada para convolução de imagens, com alto desempenho e alta flexibilidade.

2. Trabalhos Correlatos

Nesta seção, apresentamos os trabalhos encontrados mais relacionados com a nossa arquitetura [1][2][3].

Em [1] são apresentadas três arquiteturas básicas que realizam convolução, em *pipeline*, independentes das dimensões da máscara. No entanto, estas arquiteturas possuem poucos valores possíveis para os coeficientes da máscara. Estas arquiteturas são independentes de largura de banda da interface de comunicação entre o circuito e o DSP, onde os *pixels* são armazenados em um banco de registradores.

Em [2] é apresentada uma arquitetura de um circuito de convolução de imagens digitais para processamento em tempo real, utilizando três níveis hierárquicos de memória. Em um nível mais alto existe uma memória cache que armazena os *pixels* subseqüentes que serão processados. Em um nível intermediário são armazenados os *pixels* que serão processados nas próximas iterações. Finalmente o nível mais baixo, armazena a imagem completa. A existência desses três níveis aumenta o custo da implementação.

Em [3] é apresentada uma arquitetura sistólica para execução de operações de PDI baseadas em operadores na forma de janelas, como o caso da convolução de imagens. O núcleo da arquitetura é composto por elementos de processamento organizados em forma de matriz 2D sistólica. A arquitetura utiliza reaproveitamento de dados, aumentando a complexidade de implementação.

3. Arquitetura de Convolução de Imagens

A operação de convolução discreta de imagens bidimensionais é calculada através da equação 1:

$$y(m,n) = \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} h(i,j) \times x(m-i, n-j) \qquad (1)$$

A convolução de imagens pode ser implementada através da varredura de uma máscara sobre a área de uma imagem. Na equação (1): "y" é a imagem filtrada de saída; "y(m,n)" é o *pixel* referente às posições "m" e "n" da imagem filtrada de dimensões "M" e "N"; "h" é a máscara de convolução; "x" é a imagem original de entrada; "H" e "W" são as dimensões da máscara.

As operações realizadas em cada iteração da máscara são independentes, assim como as operações entre iterações diferentes. Portanto, a convolução possui um alto grau de paralelismo.

Considerando o paralelismo implícito na operação de convolução, desenvolvemos uma arquitetura paralela e parametrizada que executa convolução de imagens com um alto desempenho e alta flexibilidade. A figura 1 apresenta o diagrama de blocos da arquitetura completa desenvolvida. Esta arquitetura possui duas memórias que armazenam as imagens de entrada e de saída. Os módulos *End 1* e *End 2*, endereçam as memórias para executar a busca e armazenamento dos *pixels* corretamente. O banco de registradores tem a função de armazenar os *pixels*, otimizando a busca deles na memória de entrada. Ele também permite que os *pixels* sejam enviados paralelamente à entrada do Núcleo, possibilitando a execução paralela da operação. O módulo habilitador controla a correta habilitação dos registradores para armazenar os *pixels* corretamente e liberá-los simultaneamente. Os multiplicadores em paralelo, a árvore de somadores, os módulos normalizador e saturador constituem o Núcleo da arquitetura e executam as sub-operações de convolução de imagens.

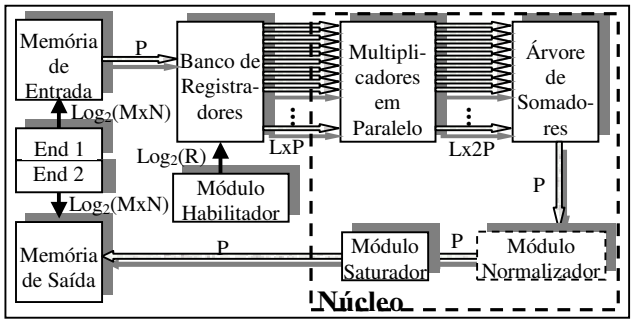

Figura 1. Arquitetura de convolução

A parametrização estrutural é uma importante característica da nossa arquitetura. Ela permite uma adaptação da arquitetura à aplicação, caracterizando grande escalabilidade e flexibilidade através da mudança dos parâmetros. Na figura 1, observamos a parametrização de algumas características da arquitetura, dentre elas: quantidade de multiplicadores em paralelo (L), largura dos dados trafegados entre os módulos (P), tamanho do banco de registradores (R) e tamanho das imagens manipuladas (MxN). Estes parâmetros estão relacionados com o tamanho da máscara e com a largura dos *pixels*.

Na figura 2, apresentamos o diagrama em blocos parametrizado do núcleo da arquitetura, que é composto pelos multiplicadores (M_n) em paralelo, onde os *pixels* de cada iteração são multiplicados, em paralelo, pelos pesos da máscara; uma árvore binária de somadores (S) efetua a soma de todos os produtos paralelamente no espaço e no tempo; um módulo normalizador (se necessário) efetua a normalização do *pixel* de saída; um módulo de saturação (se necessário), executa a saturação do *pixel* de saída.

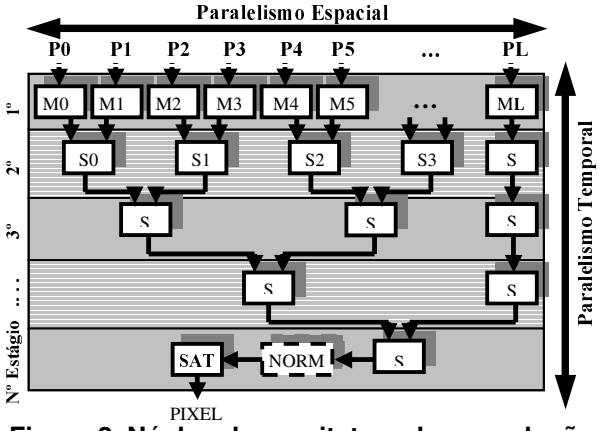

Figura 2. Núcleo da arquitetura de convolução

O Núcleo da arquitetura apresenta um paralelismo temporal - PT (*pipeline*) e espacial - PE. O PT ocorre na execução simultânea de sub-operações de estágios distintos. Este *pipeline* é sincronizado através de um sinal de *clock* e seus estágios estão representados, pela seta vertical da figura 2. O PE ocorre na realização das multiplicações e somas, de um mesmo estágio do *pipeline*, simultaneamente. O PE está representado pelas regiões horizontais da figura 2.

O número de multiplicadores em paralelo (L) é diretamente proporcional ao produto das dimensões da máscara. Este permite que todas as operações de uma iteração da máscara sejam realizadas em paralelo.

Analisando as figuras 1 e 2, observamos que a arquitetura proposta apresenta dois níveis de

paralelismo. O nível superior ocorre na execução paralela dos blocos da figura 1. O nível inferior ocorre no interior de cada bloco da figura 1, e está representado na figura 2. Neste nível, as multiplicações e adições são realizadas em paralelo.

4. Verificação da Arquitetura

Para verificação da arquitetura, codificamos, em VHDL [4], oito variações de seus parâmetros. As variações codificadas foram com máscaras 3x3, 5x5, 7x7, 8x8, 9x9, 11x11, 13x13 e 15x15 e *pixels* de 8 bits. A comunicação entre os módulos é paralela. O banco de registradores possui *H*x*N* registradores, onde *H* é o número de linhas da máscara e *N* é o número de colunas da imagem.

Implementamos a multiplicação através da codificação em tabelas (LUTs – *Look-up Tables*) para obtermos maior desempenho nas multiplicações. Utilizamos somadores completos com paralelismo interno, para executar as sub-operações de adição. No caso da normalização, os pesos da máscara foram normalizados para otimizar desempenho e recursos do FPGA utilizado. Os módulos multiplicadores e somadores possuem entradas para sinal de *clock* com o objetivo de sincronismo de operações no *pipeline*.

Foram realizadas simulações comportamentais e temporais (pós-*place&route*), das codificações da arquitetura. Utilizamos um filtro (máscara) passa-baixas, e simulamos a convolução de imagens 512x512, 1024x1024 e 2048x2048. A simulação comportamental apenas considera o comportamento lógico, enquanto a simulação temporal considera os tempos de atraso da implementação, possibilitando o cálculo da freqüência máxima de *clock* do circuito. Para isso, simulamos utilizando várias freqüências de *clock* distintas, para FPGAs das famílias Spartan-3 e Virtex-II [5]. Constatamos que as implementações suportam uma freqüência máxima de 125 MHz.

Como a arquitetura possui PT (*pipeline*), existe uma latência inicial de resposta. Esta latência ocorre devido aos níveis dos módulos LUTs (1,5 ciclo de *clock* cada), somadores (2,5 ciclos de *clock* cada) e saturador (2 ciclos) totalizando aproximadamente 13,5 ciclos de *clock* (110,3 ns), para o circuito com máscara 3x3 e freqüência de 125 MHz. Todos estes módulos possuem *pipeline* interno. Se os *pixels* de entrada forem disponibilizados a cada pulso de *clock*, após a latência inicial do circuito (*pipeline*), o circuito disponibiliza na saída um *pixel* por ciclo de *clock*. Nesta simulação da arquitetura, a convolução com máscara 3x3, de uma imagem 512x512 é realizada em 2,08 milisegundos.

Para verificar se o resultado de saída da simulação da arquitetura é o resultado correto, realizamos a mesma convolução no Matlab, onde foi convoluída a mesma imagem utilizada na simulação do circuito. A diferença entre as imagens, obtidas com a execução da convolução no circuito e no Matlab é nula.

5. Análise de Desempenho

Através das simulações realizadas com as diversas implementações da arquitetura proposta foi possível comparar seu desempenho com implementações de propósito geral e específico. Na tabela 1, apresentamos os resultados temporais obtidos nas simulações, além de resultados de outras implementações, obtidas em [3]. Todos os dados foram obtidos na execução de convolução de imagens 512x512.

Tabela 1. Tempos de execução da convolução.

Sistema	Arquitetura	Máscara	Tempo (ms)
Alacron's Al-860	I860 processor	8x8	66,10
TMS320C80	Multiprocessador	5x5	40,00
UWGSP5	DSP	3x3	19,00
LSI	Hardwired ASIC	8x8	13,11
CWP	Sistolica	7x7	8,35
MAP1000	Processador Midia	7x7	7,90
Blue Wave System	DSP	3x3	7,20
PDSP16488	Hardwired ASIC	8x8	6,56
Circuito 1	Arquitetura Proposta	3x3	2,08
Circuito 2	Arquitetura Proposta	5x5	2,07
Circuito 3	Arquitetura Proposta	7x7	2,05
Circuito 4	Arquitetura Proposta	8x8	2,04

Analisando a tabela 1, observamos que os circuitos que implementam nossa arquitetura possuem tempos de resposta muito menores que todas as demais implementações. Portanto, concluímos que os circuitos possuem ganhos de desempenhos em relação às demais implementações da tabela 1. Observamos que, na nossa arquitetura, para imagens de mesmo tamanho, com o aumento da máscara de convolução, o desempenho aumenta devido ao decréscimo do número de iterações da máscara sobre a imagem. Este ganho não é depreciado pelo acréscimo de mais estágios de somadores no *pipeline*.

Na figura 3, apresentamos um gráfico com o desempenho dos circuitos com a arquitetura proposta, implementados em FPGA, e outras implementações em *software*, considerando um processador de propósito geral. Os resultados dos processadores, assim como o dos nossos circuitos, foram obtidos analiticamente, considerando somente os tempos de processamento das

sub-operações da convolução para uma imagem 512x512. Consideramos o número de somas e multiplicações realizadas e o número médio de ciclos por instrução indicado pelo fabricante do processador.

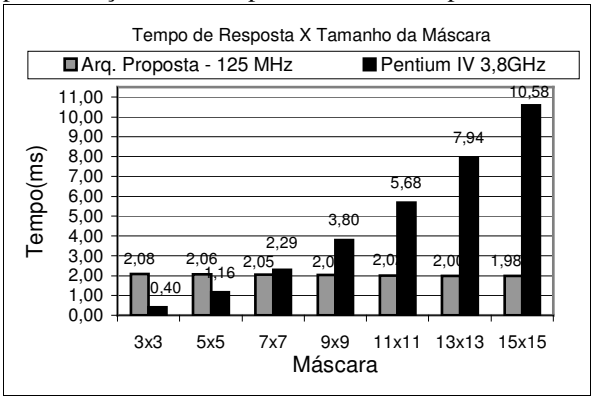

Figura 3. Tempos de execução da convolução.

Analisando a figura 3, observamos que os tempos de resposta dos circuitos que implementam a arquitetura proposta são menores, para a maioria dos tamanhos de máscara utilizados. Apesar de os tempos para máscaras 3x3 e 5x5 ficarem maiores que os tempos do processador, para máscaras maiores que 5x5, os tempos são menores. Além disso, com o aumento da máscara para uma mesma imagem, os tempos dos circuitos diminuem. Isto ocorre porque a quantidade de iterações é menor com o aumento do tamanho da máscara, pois a borda da imagem é maior. Observamos ainda que à medida que a máscara torna-se maior, a perda de desempenho do processador aumenta significativamente. Isto ocorre, pois os processadores de propósito geral executam as instruções seqüencialmente. Portanto, para implementações com máscaras maiores, mais instruções são executadas seqüencialmente. Isto aumenta bastante o tempo de execução, em relação à convolução com máscaras menores. Portanto, mesmo que o processador seja de propósito específico, e execute as instruções seqüencialmente, seu desempenho é depreciado significativamente pelo aumento da máscara. Isto não acontece com os circuitos que implementam a arquitetura proposta, devido à execução paralela no espaço e no tempo das sub-operações.

Comparando os resultados dos circuitos que implementam a arquitetura proposta com os circuitos dedicados da tabela 1 e dos trabalhos correlatos, notamos que o ganho de desempenho dos primeiros sobre os demais é muito grande. As características de paralelismo e parametrização apresentadas na arquitetura proposta justificam este ganho. Em [1] os autores apresentam resultados de taxa de um a dois *pixels* na saída do circuito por pulso de *clock*. Nosso circuito atinge uma taxa de um *pixel* na saída por pulso de *clock*. Em [2], a convolução com máscara 15x15 é processada em 74,7 quadros/s numa freqüência de *clock* de 70 MHz. Operando em 70 MHz, com máscara 15x15, nosso circuito executa em uma taxa de 282 quadros/s. Em [3], a convolução com máscara 7x7 de imagem 512x512 é processada em 8,35ms com freqüência de *clock* de 60 MHz. Em 60 MHz, com máscara de 7x7, nosso circuito executa em 4,27ms.

6. Conclusões

Considerando as análises dos resultados, concluímos que nesta pesquisa, projetamos, codificamos e implementamos uma arquitetura de convolução de imagens com alto desempenho e alta flexibilidade. Analisando os resultados apresentados, concluímos que os objetivos deste trabalho foram alcançados. Os resultados de desempenho dos circuitos implementados foram melhores que outras implementações. Concluímos que as características de paralelismo e parametrização contribuíram para estes ganhos de desempenho e para as diversas possibilidades de sua implementação, adaptando-se à aplicação.

A principal contribuição deste trabalho é a apresentação de uma arquitetura paralela e parametrizada para convolução de imagens, com alto desempenho e alta flexibilidade. Destacamos que existem possibilidades de implementação desta arquitetura com baixo custo e baixa complexidade.

Alguns dos possíveis trabalhos futuros são: adição de novas otimizações na arquitetura proposta neste artigo; projeto e implementação de um circuito de convolução de imagens reconfigurável e uma análise do desempenho de algumas otimizações arquiteturais.

Referências

[1] B. Bosi, G. Bois, Y. Savaria. Reconfigurable Pipelined 2D Convolvers for Fast Digital Signal Processing. IEEE VLSI Systems Transactions, vol. 7, no. 3, 1999, pp.299 - 308.

[2] H. Jiang, V. Öwall. FPGA Implementation of Real-time Image Convolutions with Three-Level of Memory Hierarchy, FPT, Tokyo, Japan, 2003, pp. 424- 427.

[3] C.T. Huitzil, M. A. Estrada. Real-time Image Processing with a Compact FPGA-based Systolic Architecture. Journal of Real-time Imaging, Elsevier, 2004, pp. 177-187.

[4] P.J. Ashenden, *The Designer's Guide to VHDL*, San Francisco: Morgan Kaufmann Publishers, Inc, 1996.

[5] Xilinx: The Programmable Logic Company. Em http://www.xilinx.com/. Acessado em junho de 2006.