A Operação de Convolução de Imagens em uma Arquitetura Matricial

Gerson G. H. Cavalheiro[1]
César A. F. De Rose[2]
Philippe O. A. Navaux[3]

Universidade Federal do Rio Grande do Sul
Instituto de Informática
Pós-Graduação em Ciência da Computação
Caixa Postal 15064
CEP 91501-970 - Porto Alegre, RS, Brasil
Tel.: (051)336-8399 Fax: (051)336-5576

Resumo

A aplicação de paralelismo na área de processamento de imagens se apresenta como uma alternativa para viabilizar o tratamento de imagens em tempo real. O paralelismo de dados encontrado em arquiteturas matriciais facilita o mapeamento deste tipo de problema pois cada elemento processador se encarrega de um parte da imagem a ser processada. A placa GAPP (*Geometric Arithmetic Parallel Processor*) é uma arquitetura matricial com topologia de interconexão *mesh* composta de 144 (12x12) processadores. Neste trabalho é analisado o desempenho da placa matricial GAPP em operações de convolução de imagens. Os resultados obtidos são confrontados com os resultados apresentados em [WON90]. Também são discutidas medidas para a obtenção de um melhor desempenho com a utilização da placa matricial GAPP neste tipo de aplicação.

Abstract

The aplication of parallelism in the field of image processing is an alternative to implement real time image processing. The data parallelism found in array processors simplify the mapping of this kind of problem with each processor element working on part of the image. The GAPP board (Geometric Arithmetic Parallel Processor) is a mesh architecture with 144 processors interconnected as a 12x12 bidimensional array. This work analyzes the performance of the GAPP board in image convolution. The results are compared to the results presented in [WON90]. Ways to achieve a better performance of the GAPP array in this kind of aplication are also discussed.

---

[1]Bacharel em Informática (PUC/RS,1990); Mestrando do CPGCC/UFRGS; Processamento Paralelo, Arquitetura de Computadores, Linguagens Orientadas a Objetos; E-mail: gersonc@inf.ufrgs.br

[2]Bacharel em Informática (PUC/RS,1991); Mestrando do CPGCC/UFRGS; Arquitetura de Computadores, Processamento Paralelo, Sistemas Distribuídos; E-mail: derose@inf.ufrgs.br

[3]Professor UFRGS/CPGCC; Dr. Eng. em Informática (Instituto Nacional Politécnico de Grenoble, França,1979); Arquitetura de Computadores, Processamento Paralelo, Avaliação de Desempenho; E-mail: navaux@inf.ufrgs.br

1 INTRODUÇÃO

Arquiteturas matriciais vem sendo pesquisadas como uma opção para a construção de máquinas massivamente paralelas. Suas principais características são a regularidade, facilidade de expansão e mapeamento de uma grande gama de problemas de forma eficiente [HWA84].

Uma classe de problemas que se adapta bem a este tipo de arquitetura é o processamento de imagens. Com o aumento progressivo do tamanho das imagens a serem processadas, muitas vezes em tempo real, como no caso de satélites, o uso de uma arquitetura convencional consome muito tempo de processamento e memória. Uma alternativa eficiente consiste na utilização de paralelismo. O paralelismo de dados das arquiteturas matriciais facilita o mapeamento deste tipo de problema, pois cada elemento processador (EP) se encarrega de uma parte ou até mesmo de apenas um ponto da imagem a ser processada.

A placa de processamento matricial com circuitos GAPP (*Geometric Arithmetic Parallel Processor*) [NCR86] pertence a este grupo de máquinas e foi projetada pela NCR com o objetivo de fornecer um ambiente experimental de desenvolvimento SIMD (*Single Instruction Multiple Data*) [FLY72] de baixo custo. O CPGCC/UFRGS[4] adquiriu uma placa NCR-GAPP visando dar suporte a pesquisa na área de arquiteturas matriciais.

Seu hardware é composto por uma matriz de 144 processadores de um bit, dotados, cada um, de 128 bits de memória local. O alto grau de paralelismo que pode ser alcançado pela placa advém do fato destes processadores executarem simultaneamente uma mesma instrução sobre seus dados locais ou sobre os dados recebidos de seus vizinhos.

Junto a placa, é fornecido pela NCR um ambiente integrado de desenvolvimento, denominado GAPSYS [NCR86], composto por um compilador e um depurador para a linguagem GAL (GAPP *Algoritmic Language*) [NCR88].

Os objetivos deste trabalho são (*i*) analisar o desempenho de algoritmos paralelos para a convolução de imagens na arquitetura matricial GAPP e (*ii*) propor medidas para a obtenção de um melhor desempenho com a utilização da placa matricial GAPP neste tipo de aplicação.

Os resultados obtidos são confrontados com uma análise de desempenho do circuito matricial GAPP feita em [WON90], de forma a verificar se estas considerações se comprovam para os algoritmos implementados.

Uma visão geral sobre algoritmos de convolução e de suas aplicações no processamento de imagens é dada na seção 2. Na seção 3 é apresentada a arquitetura da placa matricial GAPP juntamente com uma comparação com outras arquiteturas matriciais. Na seção 4 são analisados os aspectos referentes as implementações paralelas que foram utilizadas para avaliar o desempenho da placa nas operações de convolução.

2 A OPERAÇÃO DE CONVOLUÇÃO

O tratamento gráfico de imagens (processamento gráfico) é realizado com a finalidade de modificar ou analisar cenas digitalizadas em computador, trabalhando com as cores dos seus pontos. Os algoritmos para processamento de imagens podem ser classificados em [DAW87]:

[4]Curso de Pós-Graduação em Ciências da Computação – Universidade Federal do Rio Grande do Sul

- processamento pontual: a cor final de um ponto é determinada unicamente em função da cor original do ponto;
- processamento de área: a cor final de um ponto é calculada em função das cores dos pontos vizinhos;
- processamento geométrico: o algoritmo possui a capacidade de alterar a posição dos pontos da imagem; e,
- processamento de frames: a alteração da cor de um ponto de uma imagem depende da comparação de duas ou mais cenas.

2.1 Processamento de área

O processamento de área [MAS89], comumente chamado de filtragem ou convolução, consiste em um processo de atenuação (ou acentuação) de intensidade de uma cena, ou de detecção de padrões, ou ainda de remoção de ruídos espúrios da imagem.

Para a convolução ser aplicada, é necessário quantificar e mapear as intensidades dos pontos do espaço Euclidiano de uma cena na memória do computador sob a forma de uma matriz.

Sobre os pontos da matriz da imagem original (M) são aplicados cálculos ponderados levando em consideração uma matriz de convolução (C), resultando em uma matriz de pontos "convoluida" (R). Abaixo mostramos um exemplo de aplicação de convolução:

Seja a matriz M uma matriz de $3x3$ de pontos de uma imagem a ser trabalhada e C uma matriz de convolução tal que a matriz C seja dada por:

$$C = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

A matriz R com o resultado da aplicação da convolução é obtida pela expressão:

$$\begin{aligned} R_{(x,y)} = {} & M_{(x-1,y-1)} + M_{(x,y-1)} + M_{(x+1,y-1)} + \\ & M_{(x-1,y)} + M_{(x,y)} + M_{(x+1,y)} + \\ & M_{(x-1,y+1)} + M_{(x,y+1)} + M_{(x+1,y+1)} \ / \ 16 \end{aligned}$$

ou em uma representação simplificada:

$$R_{(x,y)} = C[M_{(x,y)}] \ / \ 16$$

onde:

- x e y: coordenadas do ponto a ser processado; e,
- 16: soma dos elementos da matriz C.

A aplicação da matriz C do exemplo acima implica no processo de filtragem de ruídos. Tal filtro não é capaz de remover o ruído completamente, sendo necessário aplicar filtros de melhor qualidade. Porém aplicar repetidas vezes o processo de convolução (sete ou oito vezes), de forma iterativa, com a matriz C produz resultados satisfatórios [PAZ88].

Costuma-se dividir os filtros em **passa–baixas** e **passa–altas**. Filtros passa–baixas atenuam as altas frequências, suavizando a cena e passa–altas atenuam as baixas frequências, dando uma maior definição das transições de regiões de uma imagem.

Diversas matrizes de convolução passa–baixas e passa–altas foram propostas, mas segundo [DAW87], há muito campo para pesquisa nesta área.

3 A ARQUITETURA GAPP

As arquiteturas matriciais, classificadas como **SIMD** (*Single Instruction stream, Multiple Data stream*) por [FLY72] e implementadas como mostra [CAR89], consistem em um conjunto de elementos de processamento dispostos em uma malha de interconexão uniforme. Problemas típicos para serem solucionados nestas arquiteturas são manipulação de matrizes, transformação rápida de Fourier (FFT), processamento de imagens e outros onde seja verificada a independência de dados.

A placa de processamento GAPP pertence a este grupo de máquinas. Seu projeto, pela NCR, teve como objetivo fornecer um ambiente experimental de desenvolvimento SIMD.

3.1 A placa NCR

A placa NCR GAPP é composta de 2 circuitos GAPP **NCR45CG72**, possuindo ao total 144 EPs, interconectados em uma malha 12x12 EPs. A matriz foi configurada como um *wraparound* cilíndrico Norte a Sul e Leste a Oeste. Para as operações de controle é necessário um hospedeiro. Para tanto, seu hardware foi projetado segundo o padrão IBM-PC, o qual serve como hospedeiro, uma vez estando conectada ao barramento de entrada e saída (E/S).

3.1.1 Estrutura funcional

Os componentes de hardware da placa encontram-se distribuídos em 6 blocos básicos. A figura 1 apresenta a disposição destes blocos na placa.

- Conjunto de 5 registradores: um registrador (de 7 bits) contendo o endereço da memória RAM para os EPs; um registrador de instrução GAPP (13 bits); um registrador de dados (8 bits); um registrador de saída de dados (8 bits) e um registrador de controle de hardware.
- Lógica de relógio permitindo que instruções sejam executadas nos EPs ou no CTLB (ver adiante) ou ainda em ambos.
- *Corner Turner Line Buffer* - CTLB, módulo responsável por formatar os dados de entrada do GAPP e reformatar sua saída. Este módulo consiste em 2 circuitos **NCR45CG72** adicionais.

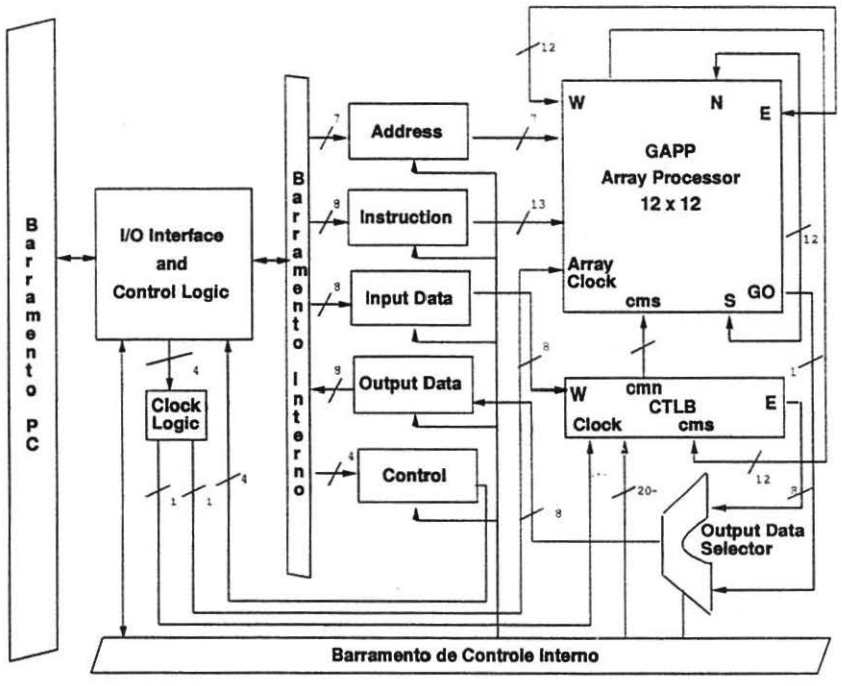

Figura 1: Esquema da placa GAPP

- Interface de E/S e lógica de controle, permitindo a comunicação entre o PC e a placa NCR GAPP através de 8 registradores mapeados em E/S. Os dados recebidos do hospdeiro são manipulados e transmitidos para os registradores e para a lógica de controle.
- Módulo de processamento, consistindo em dois circuitos **NCR45CG72** compondo a malha de $12x12$ EPs.
- Seletor de saída, através do qual, a saída disponível apresenta o sinal **GO** obtido da aplicação da função NOR sobre o sinal **GO** de todos 144 EPs do módulo de processamento ou a partir dos dados presentes no CTLB.

3.2 Programação da placa

A comunicação da placa GAPP com o computador hospedeiro se dá através de 8 registradores mapeados na memória de entrada e saída. Através destes registradores, denominados P0, P1, ..., P7, se dá a programação da placa.

O registrador P4 é o único que fornece resultados de saída da placa, sendo os demais utilizados apenas para escrita.

Os registradores P0, P1 e P2 são utilizados para o envio da microinstrução. Junto à microinstrução é enviado o endereço de memória interna dos EPs cujos dados devem ser

manipulados pela instrução (7 bits mais significativos de P2).

O registrador P3 é utilizado para o envio de dados de 8 bits para o CTLB da placa GAPP.

A escrita nos registradores P5, P6 e P7 seleciona se a instrução enviada deve ser executada pelo CTLB (P7), pelo módulo de processamento principal (P6) ou por ambos (P5). Esta seleção é feita através da escrita de um valor qualquer em um destes registradores.

3.3 O elemento de processamento

O elemento de processamento do GAPP é um processador de 1 bit, ver figura 2, composto pelos seguintes elementos:

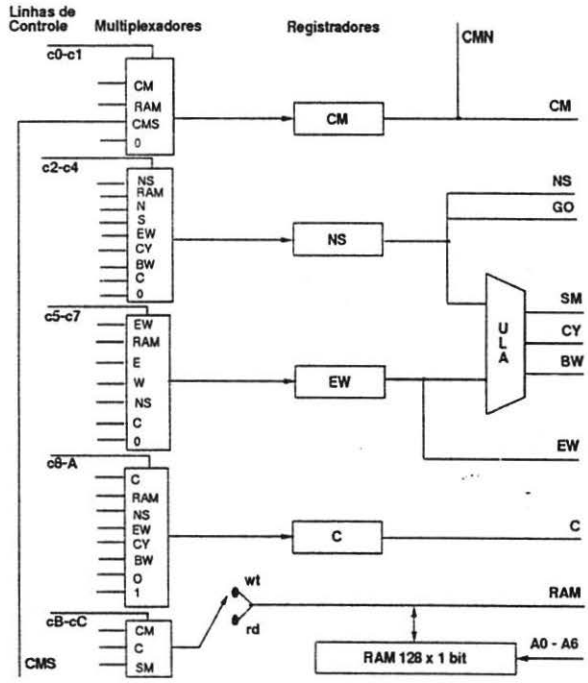

Figura 2: Elemento de processamento GAPP

- 1 unidade lógica e aritmética (**ULA**) somadora/subtratora;
- 4 registradores de 1 bit (**CM**, **NS**, **EW**, C);
- 128 bits de memória RAM local;
- 13 linhas de controle de operação (microcódigo);
- 7 linhas de endereçamento de memória;

- 5 multiplexadores;
- 7 sinais internos (**CM**, **NS**, **SM**, **CY**, **BW**, **EW**, **C**);
- 4 linhas de saída (**CMN**, **NS**, **EW**, **GO**); e,
- 5 linhas de entrada (**CMS**, **N**, **S**, **E**, **W**).

3.4 O GAPP frente a outras arquiteturas

No trabalho de Wong e Lua [WON90] encontramos uma comparação do desempenho da placa GAPP, calculado em função de programas escritos em GAL e executados no GAPP PC *Development Kit* com o desempenho de três outras arquiteturas SIMD (CLIP4, ICL DAP e MPP), apresentados por Gerritsen [GER83]. A tabela 1 sumariza as caracteríticas destas arquiteturas.

Ao analisar um conjunto de operações básicas no GAPP e nas arquiteturas SIMD, [WON90] mostra que, de modo geral, é necessário um maior número de instruções para o GAPP do que o necessário nos outros sistemas. Em contra-partida, para uma determinada área, o GAPP é capaz de fornecer até 9 vezes mais resultados que as demais.

Um dado que comprova esta informação é um *benchmark* apresentado: o **Abingdon Cross**, que realiza uma operação de convolução. Estes resultados encontram-se reproduzidos na tabela 2. O índice "fator de qualidade" apresentado refere-se a razão de **T** para N, onde **T** é o tempo de execução do *benchmark* em um *array* **N**x**N**.

O desempenho da placa GAPP também foi comparado por [WON90] com o desempenho de uma arquitetura MIMD (Transputer) e uma arquitetura *pipeline* (TITMS32020).

Tabela 1: Arquiteturas SIMD comparadas por Wong e Lua

	GAPP	CLIP4	ICL DAP	MPP
Array	12x12	96x96	64x64	128x128
Memória	128 bits	32 bits	4096 bits	1024 bits
ULA	1 bit adição AND - OR - NOT XOR - XNOR	1 bit adição AND - OR	1 bit adição AND - NOT	1 bit adição 16 funções lógicas
malha	mesh (4)	mesh (4) hexagonal (6)	mesh (4)	mesh (4)
registradores	4	4	3	6
EPs por *chip*	72	8	4	8

Tabela 2: Abingdon Cross benchmark

	GAPP	CLIP4	ICL DAP	MPP
Fator de Qualidade	$1.0x10^4$	$7.3x10^3$	$7.0x10^4$	$5.0x10^4$

O Transputer consiste em um circuito com 4 *links* de comunicação bi-direcionais de 10 Mbits por segundo, memória de, pelo menos, 2 Mbits e um *clock* interno de 20 MHz nos

modelo T414-20. A ligação de Transputer através de seus *links* permite a construção de grandes sistemas MIMD.

Na sua comparação com a placa GAPP, [WON90] utilizou uma configuração $3x3$ Transputers, observando que seu desempenho foi bastante superior a da placa GAPP, porém consumindo uma área consideravelmente maior: 2 circuitos GAPP possui área equivalente a um circuito Transputer. Outro problema está no aumento da complexidade do processo de carga de dados de um problema com características SIMD em nodos MIMD.

O TITMS32020 consiste em um processador digital de sinais, próprio para aplicações em tempo real, apesar de seu *clock* de 5 MHz, por executar várias instruções por ciclo. Suas instruções são microprogramadas e possuem duas ULAs que executam em paralelo. Em sua comparação, [WON90] observou que a placa GAPP obteve um melhor desempenho do que o TITMS32020.

4 IMPLEMENTAÇÃO DE UM ALGORITMO DE CONVOLUÇÃO

4.1 Implementação de um filtro passa–baixas

O primeiro algoritmo de convolução implementado na placa GAPP foi um filtro passa–baixas [PAZ88]. Neste filtro, uma matriz C $3x3$ é aplicada sobre cada ponto da imagem, sendo

$$C = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Este algoritmo foi escolhido por se tratar de uma operação muito utilizada em processamento de imagens e também pela facilidade de implementação. Desta forma foi possível obter rapidamente uma análise inicial do desempenho da placa GAPP neste tipo de operação e identificar os principais problemas envolvidos no uso da placa matricial.

Na implementação paralela do filtro passa–baixas, cada EP da placa GAPP executa a filtragem sobre um ponto da imagem original carregado em sua memória local. Considerando-se que a placa dispõe de 144 EPs, organizados em uma matriz $12x12$, o processamento poderia teoricamente ser realizado simultaneamente sobre 144 pontos da imagem. Entretanto, como o algoritmo trabalha com valores vizinhos (norte, sul, leste, oeste), todo o processamento realizado pelos EPs de borda da arquitetura é inutil, havendo um processamento efetivo de 100 ($10x10$) pontos por vez.

O algoritmo de convolução é facilmente implementado na arquitetura GAPP. Cada EP recebe um ponto a ser processado, obtendo dos elementos vizinhos os valores dos pontos relevantes para o cálculo da convolução. A figura 3 mostra como se processa a convolução na visão de um processador. Pelo sincronismo de execução das intruções, o fluxo de comunicação mostrado se processa simultaneamente em todos os EPs.

Sendo as imagens utilizadas nesta implementação de dimensão $640x400$, com cada ponto representando um tom de cinza entre 0 e 256, o processamento integral de uma imagem exige o seu particionamento em tantas matrizes $12x12$, com sobreposição de um ponto em cada dimensão, quanto forem necessárias para filtrar todos os seus pontos. Cada uma destas

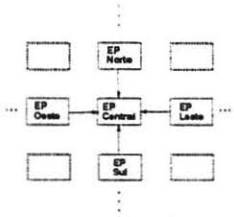

Figura 3: Implementação da convolução no GAPP

porções é carregada, processada e descarregada da arquitetura matricial a cada iteração do algoritmo, sendo só então exibida na tela da máquina hospedeira.

Para efetuar as operações de E/S com a placa GAPP foram utilizadas as rotinas otimizadas encontradas em [DER92] e não as ferramentas fornecidas com o sistema de desenvolvimento GAPSYS. Com estas rotinas, a E/S com a placa matricial é controlada totalmente pelo hospedeiro, e executada de forma mais eficiente.

O processamento a ser executado pela placa matricial em cada iteração do algoritmo foi codificado na linguagem GAL e posteriormente portado para a linguagem 'C' para ativação remota da placa a partir do hospedeiro, seguindo as técnicas descritas em [DER92]. O principio básico utilizado é a escrita das instruções a serem executadas pelos elementos processadores da matriz nos registradores de instruções da placa matricial. Estes registradores estão mapeados na área de memória de E/S do IBM-PC que funciona como hospedeiro.

4.2 Avaliação dos resultados obtidos

A tabela 3 apresenta uma comparação entre o tempo gasto por uma versão seqüencial, implementada no hospedeiro, e uma versão paralela, na placa GAPP, do filtro passa-baixas. Empregamos para fim de comparação a unidade *ticks* que representa o número de interrupções de relógio que ocorrem a cada segundo no hospedeiro. Em um IBM-PC compatível este número é de aproximadamente 18,2 vezes a cada segundo.

Tabela 3: Desempenho do filtro passa-baixas paralelo

Tipo de Processamento	Ticks
seqüencial	397
paralelo	944

O fraco desempenho da versão paralela pode ser compreendido através da tabela 4 e da figura 4(a). Observa-se que o tempo dedicado ao processamento efetivo (cálculo do valor de cada ponto da imagem) é muito pequeno, se comparado ao tempo total de processamento. Ou seja, o processamento efetivo é muito "leve" para compensar o custo de carga e descarga da placa matricial, caracterizando a operação de E/S como o gargalo do sistema.

Outra característica importante, que certamente influenciou o mau desempenho da versão paralela do algoritmo, é o fato do grau de paralelismo de dados na placa matricial ser muito pequeno se comparado com o tamanho da imagem a ser processada. O número de EPs que

Tabela 4: Distribuição do tempo de processamento no algoritmo paralelo do filtro passa-baixas

Operação	Ticks	Percentual
cálculo efetivo	68	9%
gerenciamento	335	41%
E/S com a placa	411	50%

efetivamente processam em paralelo a cada iteração é de 100 o que significa apenas 0.04% da imagem total a ser processada.

Como não existe, no momento, uma placa com maior número de EPs disponível para testes no laboratório do CPGCC-UFRGS, se optou pelo aumento da carga de trabalho a ser realizada pela placa matricial como alternativa para um aumento de desempenho. Com o aumento da fatia de cálculo efetivo (figura 4(a)), fatia esta que é efetivamente executada em paralelo pelos EPs da placa matricial, e mantendo igual a fatia de E/S certamente seria obtido um desempenho melhor do algoritmo paralelo.

4.3 Implementação de um filtro mais complexo

O segundo algoritmo implementado foi um filtro ponderado. Neste filtro são utilizados também os valores dos pontos diagonais em relação ao ponto central e uma operação de divisão para o cálculo do valor em cada ponto, sendo

$$C = \begin{pmatrix} 1 & 3 & 1 \\ 3 & 5 & 3 \\ 1 & 3 & 1 \end{pmatrix} /21$$

Desta forma o cálculo feito a cada iteração do algoritmo é maior, e a taxa de E/S se mantém igual a do exemplo anterior.

O aumento de carga computacional neste filtro foi de aproximadamente 2 vezes. Como esta parte do algoritmo é feita pela placa matricial em paralelo era esperada uma melhora no desempenho da versão paralela deste filtro.

Também foi utilizado aqui o mesmo algoritmo de particionamento do exemplo anterior o que manteve igual a fatia referente ao gerenciamento na figura 4.

4.4 Conseqüências do aumento da carga de trabalho

Como já era esperado foi obtido um aumento da fatia de cálculo efetivo com o aumento da carga de trabalho do filtro ponderado. A figura 4 apresenta uma comparação entre o tempo gasto pelas etapas de ambos os filtros implementados. As fatias referentes a E/S e gerenciamento se mantiveram inalteradas como podemos ver nas tabelas 4 e 5.

O aumento de desempenho previsto no ítem anterior ocorreu, como podemos observar na tabela 6, porém não foi suficiente para superar o desempenho do algoritmo seqüencial que aumentou para 517 *ticks*. O fator de *speed-up* refere-se a a relação entre o tempo gasto pela implementação seqüencial dividido pelo tempo gasto na implementação paralela.

Para obter com a placa GAPP um desempenho melhor que a versão seqüencial executada apenas no hospedeiro, foi necessária a implementação de um algoritmo aproximadamente 8

Tabela 5: Distribuição do tempo de processamento no algoritmo paralelo do filtro ponderado

Operação	Ticks	Percentual
cálculo efetivo	259	26%
gerenciamento	335	34%
E/S com a placa	411	40%

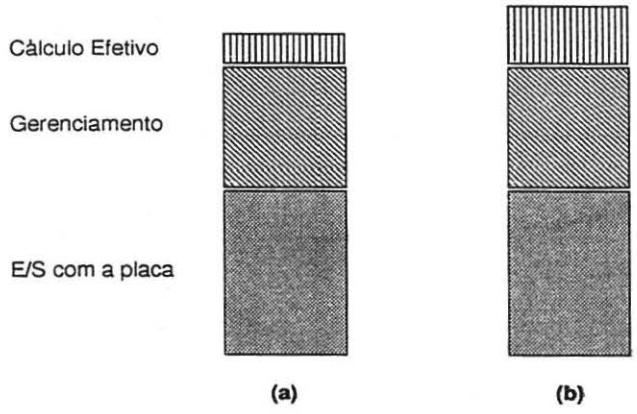

Figura 4: (a) Caracterização do processamento no filtro passa–baixas (b) Caracterização do processamento no filtro ponderado

vezes mais complexo que o filtro passa–baixas. A tabela 7 apresenta o *speed-up* obtido com os testes feitos para diferentes cargas de trabalho.

4.5 Projeção do desempenho em relação a carga de trabalho

Traçando um gráfico com os resultados das medições realizadas (figura 5), é possível identificar que a relação da carga computacional do algoritmo e seu desempenho é também linear, para as versões paralelas implementadas na placa matricial GAPP.

O ponto de intersecção das duas retas indica a carga de trabalho necessária para que o algoritmo paralelo tenha um desempenho melhor que o algoritmo seqüencial. Para os filtros implementados neste trabalho este valor é de aproximadamente 8 vezes o custo computacional do filtro passa–baixas.

Esta carga computacional pode ser facilmente obtida com a repetida execução de um mesmo filtro em cada ponto da imagem, técnica que resulta para algums filtros em uma melhor qualidade da imagem resultante [PAZ88].

Tabela 6: Desempenho dos filtros paralelos implementados

Carga computacional	Seqüencial	Paralela	Speed-up
filtro passa–baixas	397	944	0.42
filtro ponderado	517	1005	0.51

Tabela 7: Desempenho da operação de convolução paralela com cargas de trabalho variadas

carga	Seqüencial	Paralela	speed-up
2x	518	1005	0.52
5x	999	1211	0.82
10x	1842	1552	1.19
15x	2565	1891	1.36
20x	3408	2231	1.53

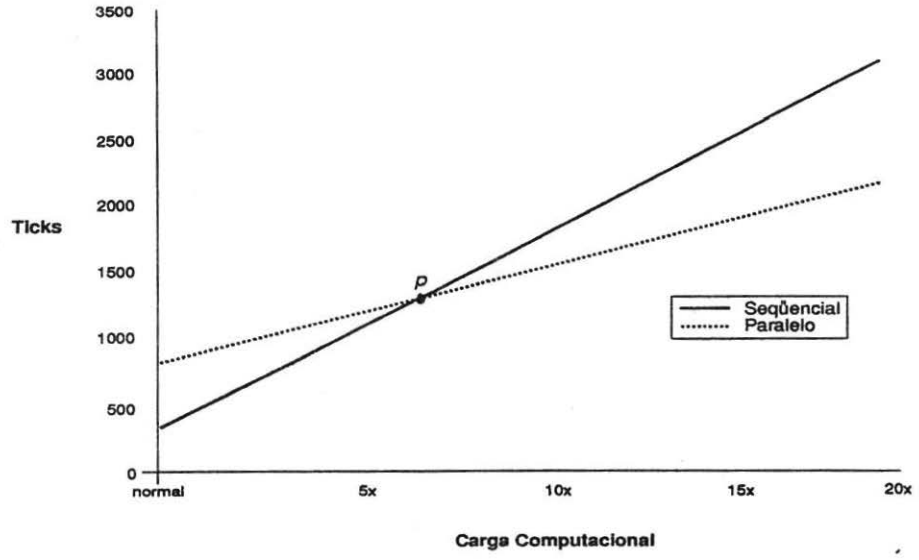

Figura 5: Comportamento do desempenho da placa GAPP em relação a carga computacional do algoritmo paralelo de convolução

De acordo com as observações feitas neste trabalho, acredita-se que o uso de uma placa com um maior número de EPs anteciparia a intersecção das retas traçadas na figura 5, diminuindo as coordenadas x e y do ponto p. Isto faria com que a necessidade de carga de trabalho diminuisse para que ocorresse um ganho em relação ao algoritmo seqüencial.

A tabela 8 apresenta o ganho estimado que um algoritmo paralelo 10 vezes mais complexo que o filtro passa-baixas apresenta com a variação no tamanho da matriz de elementos processadores.

5 CONCLUSÕES

Neste trabalho foi analisado o desempenho de algoritmos de convolução de uma imagem quando implementados de forma paralela na placa matrical GAPP. A partir dos resultados obtidos são propostas medidas para a obtenção de um melhor desempenho com a utilização

Tabela 8: Projeção dos resultados com uma placa matricial com mais EPs para uma aplicação 10 vezes mais complexa que o filtro passa-baixas

Número de Elementos	E/S	Gerenciamento	Cálculo	Total	speed-up
10x10	411	335	688	1424	1.29
46x46	411	365	32.67	778.67	2.37
118x118	411	365	4.86	750.86	2.45

da placa neste tipo de aplicação.

As medições efetuadas com a placa GAPP mostraram que o poder de processamento do *chip* NCR é bastante alto. A principal razão deste desempenho é sem dúvida o alto paralelismo de dados aliado a uma velocidade de 10Mhz. Porém, o desempenho global obtido com o uso da placa como aceleradora de operações gráficas de convolução de uma imagem para a máquina IBM-PC não foi satisfatório. Os resultados mostraram uma queda no desempenho de em média 50% em relação ao processamento seqüencial dos filtros implementados. Somente a partir de um aumento na carga de trabalho dos algoritmos de aproximadamente 8 vezes é que o algoritmo paralelo superou o seqüencial.

A razão deste fraco desempenho com os filtros implementados se deve à forma que a placa efetua a troca de dados com o hospedeiro. A operação de entrada e saída na placa GAPP é feita de forma seqüencial e bloqueante, interrompendo tanto hospedeiro como placa. Como se não bastasse, esta operação ainda é muito lenta quando comparada com a velocidade de processamento do *chip* GAPP. A operação de E/S tem um custo tão grande que apenas o tempo de comunicação com a placa nos filtros implementados já supera o tempo de execução do algoritmo seqüencial. Com a operação de E/S como gargalo do sistema, a placa só consegue um bom desempenho quando o trabalho a ser efetuado pelos EPs compensa o alto custo de E/S.

Outra razão para o fraco desempenho obtido é o baixo grau de paralelismo de dados encontrado na placa. Com uma matriz de $12x12$ elementos processadores apenas 0.04% da imagem (de dimensões $640x400$) pode ser processada a cada iteração do algoritmo. Com um número maior de elementos processadores o paralelismo de dados seria maior e a placa forneceria resultados melhores, até mesmo para os algortimos mais simples, com menor carga de trabalho.

Os resultados encontrados neste trabalho confirmam as conclusões de [WON90] em relação ao desempenho do *chip* GAPP. Porém sua análise não leyou em consideração o problema de E/S, que, como foi apresentado neste trabalho, mostrou-se um ponto fraco da arquitetura. Desta forma, nas operações de convolução implementadas não foi possível obter resultados tão animadores quanto os de [WON90]. A conexão da placa GAPP com o hospedeiro compromete o desempenho do conjunto, uma vez que a operação de E/S, realizada na forma serial, constitui um gargalo para muitas aplicações, em que a quantidade de processamento não justifica a carga de dados.

É importante ressaltar que a placa GAPP não foi desenvolvida com o intuito de ser uma placa aceleradora de alto desempenho e muito menos uma máquina multiprocessadora SIMD completa. O ambiente GAPP é apenas o primeiro passo neste sentido, permitindo que o *chip* NCR matricial seja pesquisado, e que aplicações paralelas sejam desenvolvidas e analisadas neste tipo de arquitetura.

Os resultados deste trabalho mostraram que se bem aplicado, o *chip* GAPP é uma boa

opção para a construção de máquinas matriciais de grande porte, devido a boa relação do número de EPs por área, aliada a sua alta velocidade de processamento.

Agradecimentos

Ao colega Ricardo Menna Barreto pela contribuição dada para a realização deste trabalho.

Referências

[CAR89] CARÍSSIMI, Alexandre da S. **Implementação de Arquiteturas SIMD**. Porto Alegre, CPGCC da UFRGS, 1992. (Dissertação de Mestrado)

[DAW87] DAWSON, Benjamin M. Introduction to Imagem Processing Algorithms. **Byte**, v. 12, n 3, pp. 169-186, Mar. 1987.

[DER92] DE ROSE, César; CAVALHEIRO, Gerson G. H; MENNA BARRETO, Ricardo. **Rotinas de Comunicação para a Placa GAPP**. Porto Alegre: CPGCC da UFRGS, 1992. Relatório de Pesquisa.

[FLY72] FLYNN, M. J. Some computer organizations and their effectiveness. **IEEE Transactions on Computers**, New York, v. C-21, n. 9. pp. 948-160, Sept. 1972.

[GER83] GERRITSEN, F. A. A comparison of the CLIP4, DAP and MPP Processor-array implementations. **Computing Strutures for IMage Processing**. New York: Academic Press, 1983.

[HWA84] HWANG, K.; BRIGGS, F. **Computer Architecture and Parallel Processing**. New York: McGraw Hill. 1984.

[MAS89] MASCARENHAS, Nelson; VELASCO, Flávio. Processamento Digital de Imagens. Escola Brasileiro–Argentina de Informática, 4., 1989, Termas de Rio Hondo. **Procedings...** Buenos Aires: Kapelusz, 1988.

[NCR86] NCR Microeletronics. **GAPP Personal Computer Development System User's Manual**. Fort Collins:[s.n.] 1986. 99p.

[NCR88] NCR Microeletronics. **GAPP Algorithm Language User's Manual**. Fort Collins:[s.n.] 1988. 99p.

[PAZ88] PAZ, Eduardo P. CUNHA. Tarcisio N. **Iniciação ao Processamento Digital de Imagens**. Rio de Janeiro: UFRJ/NCE, 1988. 55p. (curso ministrado no *VIII* Congresso da Sociedade Brasileira de Computação - *VII* JAI - Versão Preliminar)

[WON90] WONG, W. F.; LUA, K. T. A Preliminary Evaluation of a Massively Parallel Processor: GAPP. **Microprocessing and Microprogramming**, Nort-Holand, v.29, n.1, p. 53-61, July 1990.