Processador de Vizinhança para Filtragem Morfológica

Ilka Marinho Barros *Roberto de Alencar Lotufo*
Neucimar Jerônimo Leite

Relatório Técnico DCC–95-11

Agosto de 1995

Processador de Vizinhança para Filtragem Morfológica

Ilka Marinho Barros Roberto de Alencar Lotufo*
Neucimar Jerônimo Leite†

11 de agosto de 1995

Sumário

This paper describes a systolic architecture that implements mathematical morphological filters (non-linear filters). This architecture called SAMM (Systolic Architecture for Mathematical Morphology) operates on both binary and gray-level images with a planar structuring element. It presents two modularity levels. The first level is related to numerical and binary operations being executed through the same hardware. The other modularity level is obtained by exploring the systolic characteristics of dilation and erosion operations. A comparasion with other processors presenting good performance and interesting architectures is also given in this work.

1 Introdução

O processamento de imagem, em geral, e o processamento de baixo nível em particular justificam o desenvolvimento de hardware devido à grande quantidade de dados envolvida. A morfologia matemática é uma ferramenta bastante útil no processamento baixo nível de imagens a qual, sendo implementada em hardware, viabiliza sua aplicação de maneira mais eficiente. As operações morfológicas básicas são a dilatação e a erosão que combinadas definem operações mais complexas. A implementação da dilatação e da erosão em hardware dedicado possibilita a máxima exploração do paralelismo presente nestas operações, resultando, assim, no aceleramento do processamento.

Este artigo apresenta uma arquitetura para a implementação das operações morfológicas de vizinhaça de dilatação e erosão denominada SAMM (Systolic Architecture for Mathematical Morphology). Esta arquitetura, que realiza no mesmo hardware operações sobre imagens binárias e numéricas é modular, sistólica e adequada à implementação VLSI.

O crescente desenvolvimento da Morfologia Matemática nos últimos anos tem levado vários grupos de pesquisa a implementar suas operações básicas em hardware [1], [2]. Escolheu-se para comparação com esta arquitetura alguns processadores recentemente desenvolvidos que apresentam bom desempenho e características arquiteturais interessantes.

*UNICAMP - Departamento de Engenharia da Computação e Automação Industrial Caixa Postal 6101 13081 Campinas, SP, Brasil - ilka,lotufo@dca.unicamp.br

†UNICAMP - Departamento de Ciência da Computação Caixa Postal 6101 13081 Campinas, SP, Brasil - neucimar@dcc.unicamp.br

A seção 2 apresenta as equações que deram origem ao hardware, e a implementação desta arquitetura. Na seção 3 são apresentadas outras arquiteturas que implementam as mesmas operações seguidas de comparações estruturais. Finalmente, a seção 4 apresenta uma breve conclusão do trabalho.

2 A Implementação

A partir de equações matemáticas que definem as operações de dilatação e erosão, chegou-se às seguintes formulações para a implementação da arquitetura [3].

A dilatação de f por B é a função $\delta_B(f)$ em K^E

$$\delta_B(f)(x) = max\{min\{f(y+x), M(-y)\} : y \in W \; e \; y+x \; \in E\} \tag{1}$$

A erosão de f por B é a função $\varepsilon_B(t)$ em K^E

$$\varepsilon_B(f)(x) = min\{min\{f(y+x), M(y)\} : y \in W \; e \; y+x \in E\} \tag{2}$$

onde:

B é um subconjunto de Z^2 chamado conjunto estruturante.

W é um retângulo de Z^2 onde é definida a função M em W^K, denominada matriz estruturante, de dimensões $m \times n$ (linhas por colunas).

$M(x)$ são os elementos da matriz estruturante supondo que a sua origem está no elemento central. $M(x)$ assume os valores 0 e k (planar).

E é um retângulo de Z^2 e K um intervalo $[0, k]$ de Z, com $k > 0$, ou seja, o domínio da imagem de entrada.

$f(x)$ é a imagem a ser operada.

x e y são elementos de Z^2.

É importante notar que as definições acima se aplicam tanto para imagens binárias como para imagens numéricas. Na dilatação, é considerada a matriz transposta $M(-y)$.

2.1 Arquitetura SAMM

A arquitetura SAMM (Systolic Architecture for Mathematical Morphology) é a implementação em hardware das equações 1 e 2. Estas operações morfológicas foram decompostas em sub-operações, sendo realizadas por blocos de circuitos (elementos morfológicos) que se encadeiam para realizar a operação total. Os dados são introduzidos na estrutura, e o resultado é gerado gradativa e acumulativamente durante a passagem destes através dos blocos. De acordo com a equação (1), a operação de dilatação é fragmentada em vários mínimos e o máximo entre eles, sendo que o máximo é calculado acumulativamente a medida que os mínimos são gerados. O correspondente ocorre para a operação de erosão, formada pelo mínimo dos mínimos. A figura 1, mostra o elemento morfológico utilizado tanto para a dilatação como para a erosão, pela seleção da função MÍNIMO/MÁXIMO.

A tabela 1 apresenta o diagrama tempo-espaco transitório da erosão de uma imagem $f(x)$ por uma matriz estruturante 1x3 (W), implementada através da estrutura da figura 2. A matriz estruturante determina o tamanho e o formato da estrutura de processamento, estando cada um dos seus elementos armazenado em um elemento morfológico (registrador). O cálculo do pixel $\varepsilon_B(f)(j, i)$ é dado pela equação 3.

$$\varepsilon_B(f)(j,i) = min(min(f(j,i-1), M(0,-1)), k,$$
$$min(f(j,i), M(0,0)), min(f(j,i+1), M(0,1))) \quad (3)$$

Esta arquitetura caracteriza-se como um sistema sistólico por apresentar comunicação local e síncrona, ser composta por elementos simples e regulares, e apresentar taxa de comunicação constante com o meio externo [4].

A complexidade desta arquitetura é diretamente proporcional ao tamanho da matriz estruturante (W) e independe do tamanho da imagem. Para operar uma imagem com N colunas por uma matriz estruturante 3×1, temos a estrutura mostrada na figura 3 e seu funcionamento no diagrama tempo-espaço na tabela 2. Assim, considerando:

Matriz Estruturante =

$$\begin{bmatrix} M[-1,0] \\ M[0,0] \\ M[1,0] \end{bmatrix}$$

Matriz da Imagem =

$$\begin{bmatrix} & & \vdots & & \\ \cdots & f(j-1,i-1) & f(j-1,i) & f(j-1,i+1) & \cdots \\ \cdots & f(j,i-1) & f(j,i) & f(j,i+1) & \cdots \\ \cdots & f(j+1,i-1) & f(j+1,i) & f(j+1,i+1) & \cdots \\ & & \vdots & & \end{bmatrix},$$

e como unidimensionais os índices da matriz imagem, tem-se a matriz a seguir:

Matriz da Imagem =

$$\begin{bmatrix} f(0) & \cdots & \cdots & \cdots & f(N-1) \\ & & \vdots & & \\ \cdots & f(k-N-1) & f(k-N) & f(k-N+1) & \cdots \\ \cdots & f(k-1) & f(k) & f(k+1) & \cdots \\ \cdots & f(k+N-1) & f(k+N) & f(k+N+1) & \cdots \\ & & \vdots & & \end{bmatrix}$$

A equação 4 enfoca a formação do pixel $\varepsilon_B(f)(k)$.

$$\varepsilon_B(f)(k) = min(min(f(k-N), M(-1,0)), k,$$
$$min(f(k), M(0,0)), min(f(k+N), M(1,0))) \quad (4)$$

A matriz estruturante é sempre posicionada no processador com uma rotação 180 graus em relação ao seu eixo horizontal, tornando-se isto visível neste caso. Assim, no cálculo de um pixel, é obtido, na primeira linha da estrutura, o resultado parcial referente à operação dos pixels de entrada com relação à linha inferior da matriz estruturante original. As configurações com $m \geq 2$ (onde m é o número de linhas da matriz estruturante) possuem elementos de atraso entre uma linha e outra da estrutura para armazenar o pixel até o instante da sua utilização pela próxima linha.

Devido a forma de propagação do pixel, o qual é entregue simultaneamente a todos os elementos de processamento da linha da estrutura e ao registrador de deslocamento da linha seguinte, o tempo de armazenamento do pixel no registrador de deslocamento na arquitetura SAMM é de $N + n$ períodos (onde N e n são o número de colunas da imagem e da matriz estruturante, respectivamente. Como o resultado parcial de uma linha leva n períodos para ser formado e estar disponível na saída, são necessários no registrador de deslocamento, além de N estágios para armazenar a linha de entrada, mais n estágios para tornar disponível o pixel para o próximo processamento.

2.1.1 Modos Numérico e Binário de operação do processador

A partição do hardware para uso e aproveitamento integral nas operações sobre imagens numéricas e binárias, implementada em [2], é também utilizada na arquitetura SAMM. Isto é alcançado através da decomposição dos circuitos de cálculo de máximo e mínimo para uma imagem numérica em "fatias" binárias, formando-se os elementos morfológicos binários mostrados na figura 4.

Na operação modo numérico, $k = 255$ (8 bits), é necessário que os elementos morfológicos binários trabalhem interligadamente. Para operar com uma matriz estruturante 3x3, por exemplo, tem-se a estrutura da figura 6. Aproveitando-se o mesmo hardware para o modo binário, encadeia-se as oito células básicas. Como cada célula implementa uma operação morfológica completa sobre uma imagem binária, é possível a realização de até oito operações morfológicas diferentes e sucessivas (figura 5). Neste último caso os elementos morfológicos correspondentes trabalham independentemente.

3 Comparação com outras arquiteturas

3.1 Arquitetura "Bit-Plane"

O processador apresentado em [2] foi desenvolvido especialmente para a realização de operações de Morfologia Matemática em PLCA ("Progammable Logic Cell Array"). É uma arquitetura programável, construída a partir de células básicas binárias, que permite uma configuração ótima do hardware no tratamento de imagens binárias e numéricas. Cada célula é um processador de arquitetura direta (figura 7), onde os pixels a serem operados em cada instante estão presentes na memória interna e são simultaneamente acessados por um bloco de lógica binária que os processa. Na arquitetura direta há, portanto, uma distinção na célula básica entre memória local dos pixels e parte de processamento, não apresentando modularidade neste nível. Como cada célula realiza uma operação morfológica binária completa, várias delas podem ser encadeadas, realizando vários processamentos em pipeline sobre a imagem binária de entrada, do mesmo modo que a partição implementada em SAMM. Quando a imagem a ser processada é numérica, as células trabalham paralelamente, cada uma sobre um "bit-plane" da imagem, sendo portanto, uma estrutura "bit-slice".

3.2 O Processador NPP

O processador NPP (Non-linear Pipeline Processor), desenvolvido para a implementação de filtros não lineares, é um processador sistólico. Sua arquitetura consiste em dois tipos de unidades básicas de cálculo (unidade de dilatação e unidade de erosão - figura 8), concebidas a partir das equações que definem as operações de erosão e de dilatação numéricas não planares [1].

Unidades de dilatação (erosão) são compostas basicamente por quatro registradores de deslocamento de um est'agio de b-bits, somador (subtrator) e comparador e quando encadeadas implementam as respectivas operações. Estas unidades podem ser transformadas em unidades binárias, bastando para isto substituir o somador (subtrator) por uma porta AND (OR) e o comparador por uma porta OR (AND). A comunicação entre unidades é local, simples, regular e síncrona. O fluxo dos dados através da estrutura é também síncrono e regular.

3.3 Decomposição Threshold

Outra possibilidade na implementação de operações morfológicas é a aplicação da decomposição "threshold" nas operações morfológicas numéricas [5], [6]. Através desta propriedade é possível decompor sinais numéricos (b-bits) em múltiplos sinais binários (b) os quais são tratados paralelamente. Apesar de sua lógica combinacional diferente, esta arquitetura é semelhante àquela da figura 7.

3.4 Comparações

O princípio de partição binária/numérica considerada na arquitetura SAMM é o mesmo utilizado na arquitetura "bit-plane" [2]. Em ambas, o hardware que implementa a operação numérica é fragmentado em "fatias" binárias, operando interligadamente ou independentemente sobre imagens numéricas ou binárias, respectivamente. O dado numérico não necessita sofrer nenhum processamento anterior ou posterior ao seu tratamento, bastando replicar o número de células básicas conforme o tamanho do pixel e entregar um dos seus bits a uma delas. Neste sentido, aproveita-se integralmente o hardware disponível no modo numérico para o binário através do encadeamento de várias operações binárias.

Fazendo uso da decomposição threshold, uma vez decomposto o pixel numérico em módulos binários, estes são tratados paralela e independentemente. Somente após o final do processamento os dados interagem, constituindo novamente o pixel numérico (resultado). Neste caso, portanto, há necessidade de um pré-processamento do pixel numérico para particioná-lo e um pós-processamento para sua recomposição. O hardware que implementa esta propriedade não é modular, sendo necessárias grandes modificações quando da variação do tamanho do pixel a ser processado, o que gera uma estrutura bastante irregular.

Na arquitetura SAMM há modularidade em dois níveis. O primeiro está relacionado com a estrutura interna da célula básica, composta por elementos morfológicos binários - módulos - encadeados, que são nada mais que partes da memória e do processador. O outro nível de modularidade se refere à estrutura de mais alto nível da arquitetura, a qual é composta por células básicas que se combinam diferenciadamente conforme o modo de operação (numérico ou binário). A arquitetura "bit-plane" não apresenta o primeiro tipo de modularidade, sendo sua célula básica uma implementação da arquitetura direta. Neste caso há distinção entre unidade de processamento e memória de pixels, ou seja, a célula básica "bit-plane" não é composta por módulos como a arquitetura SAMM. Esta modularidade torna a arquitetura SAMM bem adaptada à uma implementação VLSI.

Estruturalmente, a arquitetura SAMM é semelhante à arquitetura NPP [1], apresentando vantagens importantes. Ambas são arquiteturas modulares (neste caso, tratando-se da modularidade a nível interno da célula básica) e flexíveis. Seu tamanho é diretamente proporcional ao do elemento estruturante e independente do tamanho da imagem

a ser processada, podendo formar estruturas de operação unidimensionais e bidimensionais com a facilidade proporcionada por um sistema modular. Em ambas, o processador e a memória encontram-se presentes em cada elemento morfológico (SAMM) e unidade básica (NPP), ou seja, não são arquiteturas diretas como "bit-plane".

No sincronismo dos dados, a unidade básica da arquitetura NPP necessita de mais "flip-flops" que o elemento morfológico de SAMM, devido ao modo de propagação dos pixels vindos da memória. Em NPP os pixels são entregues à próxima unidade básica somente após passarem por dois atrasos, ficando temporariamente armazenados no seu interior. Isto possibilita, entretanto, a realização paralela, sobre dados distintos, das duas operações implementadas pela unidade básica, alcançando assim velocidade dobrada com relação à SAMM. Na arquitetura SAMM, o atraso unitário ("flip-flop") presente em cada elemento morfológico, garante o armazenamento temporário do resultado parcial após o cálculo dos dois circuitos (dois mínimos - erosão - ou um mínimo e um máximo - dilatação), para o sincronismo do fluxo de dados. Apesar de uma velocidade de processamento menor (equações 5, 6), a arquitetura SAMM apresenta uma menor latência de resultados. Considerando:

$T_{M/m}$ - tempo de realização do máximo/mínimo
T_a - tempo de realização da adição
T_{df} - atraso do "flip-flop"
T_{em} - perído mínimo do processador
tem-se para SAMM:

$$T_{em} = 2T_{M/m} + T_{df} \tag{5}$$

e para NPP:

$$T_{em} = T_a + T_{df} \tag{6}$$

Outro ponto distinto refere-se à formação de estruturas que operam com uma matriz estruturante com número de colunas maior que 1. Em NPP é necessário, ao final de cada linha da estrutura, uma lógica adicional para combinar os resultados parciais gerados em cada linha e obter o resultado final. Em SAMM ocorre a passagem do resultado parcial de uma linha superior para uma entrada no início da linha seguinte, que vai sendo agregado, ao longo da estrutura, a outros resultados parciais. Isto se dá sucessivamente entre uma linha e outra, sendo o resultado final gerado gradativa e acumulativamente.

O projeto da arquitetura NPP não enfoca o problema da implementação de operações binárias e numéricas na mesma arquitetura, ponto importante na arquitetura SAMM. Entretanto NPP considera operações morfológicas através de matrizes estruturantes planares e não planares. Até o momento, a arquitetura SAMM considera somente as matrizes estruturantes planares, que são os casos mais usuais na prática [7].

Na implementação da decomposição threshold os resultados são obtidos paralelamente, atingindo grande velocidade de processamento. Entretanto, o hardware que a implementa é bastante complexo e não modular. Nos processadores SAMM e NPP a complexidade do hardware cresce linearmente com o número de bits. O hardware que executa uma decomposição threshold apresenta um crescimento exponencial da complexidade com relação ao número de bits (tabela 3).

Considerando:

B - número de portas de cada elemento morfológico = 48,
n, m - dimensões da matriz estruturante,
M, N - dimensões da imagem de entrada,
b - número de bits,

F - número de portas em cada "flip-flop",

N_B - número de portas dos blocos da estrutura,

N_S - número de portas dos registradores de deslocamento,

N_R - número de portas dos registradores do elemento da matriz estruturante em cada bloco básico,

N_{tot} - número total de portas,

Para a arquitetura SAMM temos a complexidade do hardware calculada por:

$$N_B = Bmnb \tag{7}$$

$$N_S = (N+n)(m-1)bF \tag{8}$$

$$N_R = 3mnb \tag{9}$$

$$N_{tot} = N_B + N_S + N_R \tag{10}$$

Para o cálculo dos dados da tabela 3, não foram incluídos o número de portas dos registradores de deslocamento nem dos registradores do elemento da matriz estruturante, a fim de estabelecer uma comparação coerente entre as arquiteturas. Os dados referentes à NPP e à decomposição threshold foram extraídos de [1] e se referem a estruturas que operam com matrizes planares e não planares. Estima-se que SAMM continue a apresentar o menor número de portas, mesmo quando implementada para operações não-planares.

4 Conclusão

SAMM é uma arquitetura sistólica que implementa, na mesma arquitetura de base, operações binárias e numéricas de morfologia matemática. Para a validação da arquitetura foi considerada a linguagem de descrição de hardware VHDL da Mentor Graphics e os resultados destas simulações comparados com os fornecidos pela caixa de ferramentas de Morfologia Matemática do Khoros[8]. Esta arquitetura incorpora características importantes de implementação tais como utilização integral do hardware em operações numéricas e binárias, alta modularidade, regularidade e baixa complexidade, que a tornam bem adaptada à implementação em VLSI.

5 Bibliografia

1. **Loui, A., Venetsanopoulos, A.N. and Smith, K.C**, *Flexible Architectures for Morphological Image Processing and Analysis*, IEEE Transaction on Circuits and Systems for Video Technology, vol. 2, n. 1, March 1992.
2. **Klein, J.C, Collange, F., Bilodeau, M.**, *A Bit Plane Architecture for an Image Analysis Processor Implemented with P.L.C.A. Gate Array*, First European Conf. on Computer Vision, Lecture Notes on Computer Science, Apr. 1990, Antibes, France.
3. **Barros, I.M., Lotufo, R.A., Leite, N.J.**, *Arquitetura VLSI de Operações de Morfologia Matemática*, Seminário Integrado de Software e Hardware, Semish-94, Caxambu, Ago., 1994.

4. **Santos, E.T.**, *Um Sistema Sistólico para Interpolação de Imagens*, Anais do VI SIBGRAPI (1993), pp. 133-138.
5. **Shih, F.Y-C, Mitchell, O.R.**, *Threshold Decomposition of Gray-Scale Morphology into Binary Morphology*, IEEE Transactions on Pattern Analysis and Machine Inteligence, vol. 11, n. 1, Jan 1989, pp. 31-42.
6. **Chang, Y., Ansari, N.**, *A Practical VLSI Realization of Morphological Operations*, SPIE Vol. 1606 Visual Communications and Image Processing'91 Image Processing, 1991.
7. **Serra, J.** *Image Analysis and Mathematical Morphology*, vol. 1, 1982.
8. **Barreira, J., Banon, G.J.F. e Lotufo, R.A.**, *A Mathematical Morphology Toolbox for the Khoros System*, Image Algebra and Morphological Image Processing V - International Symposium on Optics, Imaging and Instrumentation, San Diego, 24-29 July, 1994.

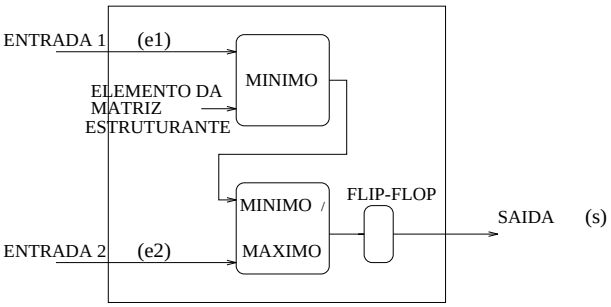

Figura 1: elemento morfológico

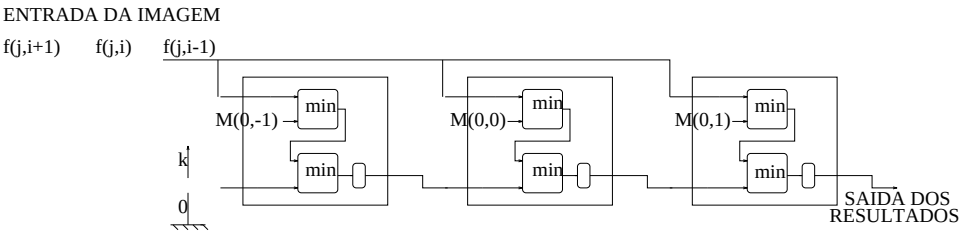

Figura 2: estrutura 1x3

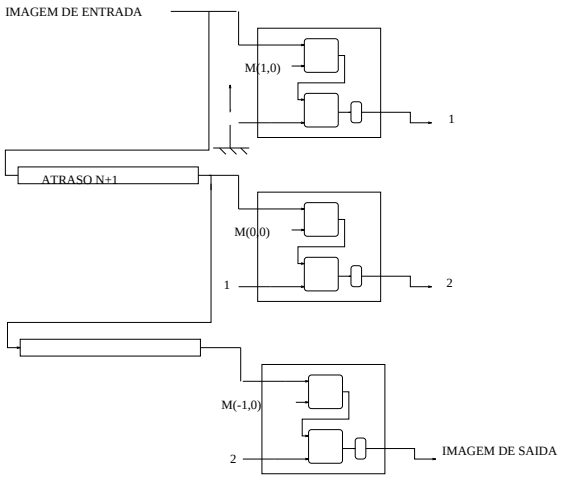

Figura 3: implementação de uma estrutura 3x1 para operações morfológicas

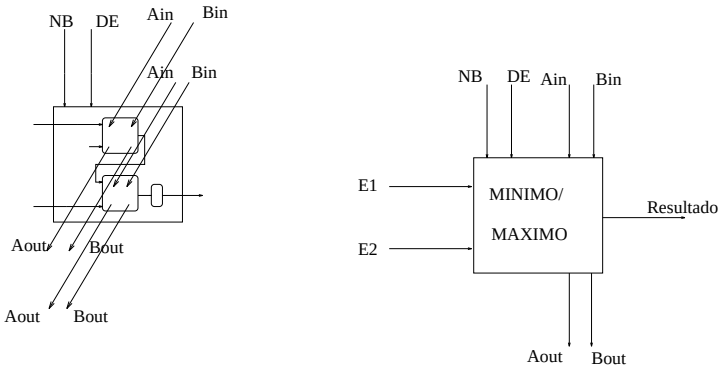

Figura 4: elemento morfológico binário e circuito de máximo/mínimo

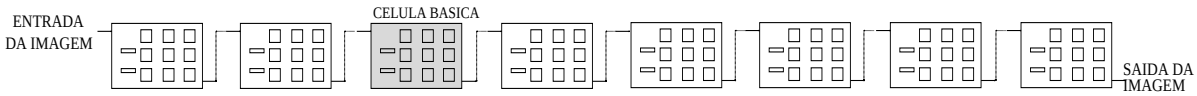

Figura 5: configuração das células no modo binário

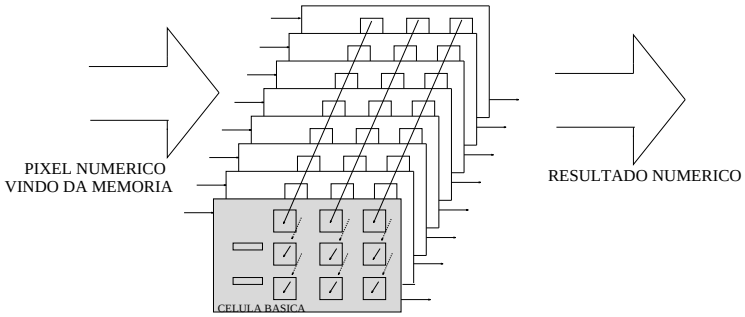

Figura 6: configuração das células no modo numérico

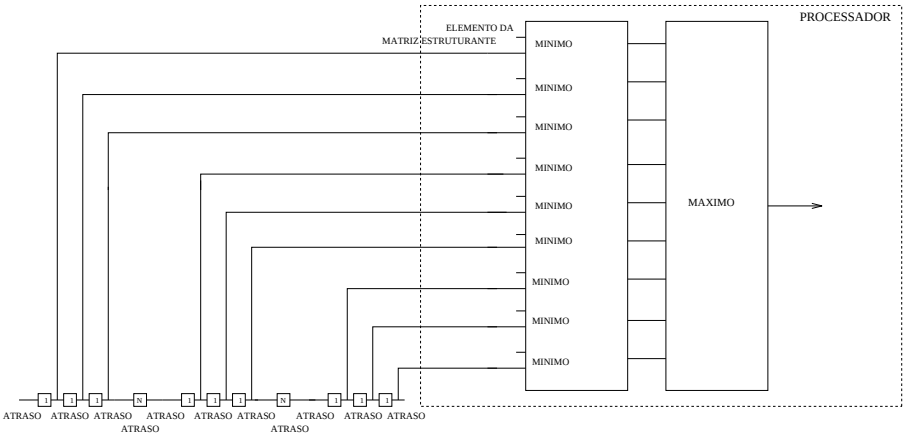

Figura 7: arquitetura direta de cada célula básica do processador "bit-plane" para operação com uma matriz estruturante 3×3

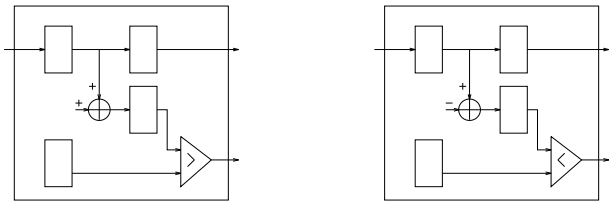

Figura 8: unidades básicas do processador NPP

TEMPOS		ELEMENTOS		
		elemento 1	elemento 2	elemento 3
T 1	entrada 1	$f(j,i-1)$	$f(j,i-1)$	$f(j,i-1)$
	entrada 2	k	X	X
	saída	$\mathbf{min(min(}f(j,i-1)\mathbf{,}M(0,-1)\mathbf{),k)}$	X	X
T 2	entrada 1	$f(j,i)$	$f(j,i)$	$f(j,i)$
	entrada 2	k	$\min(\min(f(j,i-1),M(0,-1)),\mathrm{k})$	X
	saída	$\min(\min(f(j,i),M(0,-1)),\mathrm{k})$	$\mathbf{min(min(}f(j,i-1)\mathbf{,}M(0,-1)\mathbf{),k,}$ $\mathbf{min(}f(j,i)\mathbf{,}M(0,0)\mathbf{))}$	X
T 3	entrada 1	$f(j,i+1)$	$f(j,i+1)$	$f(j,i+1)$
	entrada 2	k	$\min(\min(f(j,i),M(0,-1)),\mathrm{k})$	$\min(\min(f(j,i-1),M(0,-1)),\mathrm{k},$ $\min(f(j,i),M(0,0)),$
	saída	$\min(\min(f(j,i+1),M(0,-1)),\mathrm{k})$	$\min(\min(f(j,i),M(0,0)),\mathrm{k}$ $\min(f(j,i+1),M(0,1)))$	$\mathbf{min(min(}f(j,i-1)\mathbf{,}M(0,-1)\mathbf{),k,}$ $\mathbf{min(}f(j,i)\mathbf{,}M(0,0)\mathbf{),}$ $\mathbf{min(}f(j,i+1)\mathbf{,}M(0,1)\mathbf{))}$

Tabela 1: Diagrama transitório tempo-espaço da erosão para a estrutura 1x3

TEMPOS		ELEMENTOS		
		elemento 1	elemento 2	elemento 3
T_{k+N}	e1	$f(k+N)$	$f(k-1)$	$f(k-N-2)$
	e2	k	$\min($ $\min(f(k+N-1),M(1,0)),\mathrm{k})$	$\min($ $\min(f(k-3N-2),M(1,0)),\mathrm{k}$ $\min(f(k-2N-2),M(-1,0)))$
	s	$\mathbf{min(}$ $\mathbf{min(}f(k+N)\mathbf{,}M(1,0)\mathbf{),k)}$	$\min($ $\min(f(k+N-1),M(1,0)),\mathrm{k},$ $\min(f(k-1),M(0,0)))$	$\min($ $\min(f(k-3N-2),M(1,0)),\mathrm{k},$ $\min(f(k-2N-2),M(0,0)))$ $\min(f(k-N-2),M(0,0)))$
T_{k+N+1}	e1	$f(k+N+1)$	$f(k)$	$f(k-N-1)$
	e2	k	$\min($ $\min(f(k+N),M(1,0)),\mathrm{k})$	$\min($ $\min(f(k+N-1),M(1,0)),\mathrm{k}$ $\min(f(k-1),M(1,0)))$
	s	$\min($ $\min(f(k+N+1),M(1,0)),\mathrm{k})$	$\mathbf{min(}$ $\mathbf{min(}f(k+N)\mathbf{,}M(1,0)\mathbf{),k,}$ $\mathbf{min(}f(k)\mathbf{,}M(0,0)\mathbf{))}$	$\min($ $\min(f(k+N-1),M(1,0)),\mathrm{k}$ $\min(f(k-1),M(1,0)),$ $\min(f(k-N-1),M(-1,0)))$
T_{k+N+2}	e1	$f(k+N+2)$	$f(k+1)$	$f(k-N)$
	e2	k	$\min($ $\min(f(k+N+1),M(1,0)),\mathrm{k})$	$\min($ $\min(f(k+N),M(1,0)),\mathrm{k},$ $\min(f(k),M(0,0)))$
	s	$\min($ $\min(f(k+N+2),M(1,0)),\mathrm{k})$	$\min($ $\min(f(k+N+1),M(1,0)),\mathrm{k},$ $\min(f(k+1),M(0,0)))$	$\mathbf{min(}$ $\mathbf{min(}f(k+N)\mathbf{,}M(1,0)\mathbf{),k,}$ $\mathbf{min(}f(k)\mathbf{,}M(0,0)\mathbf{),}$ $\mathbf{min(}f(k-N)\mathbf{,}M(-1,0)\mathbf{))}$

Tabela 2: Diagrama em regime tempo-espaço da erosão para a estrutura 3x1

ARQUITETURA	TAMANHO DA MATRIZ ESTRUT.	NUMERO DE PORTAS		
		1-bit	4-bits	8-bits
THRESHOLD	3x3		1214	326.7K
	4x4		2054	555.1K
	5x5		3134	848.9K
NPP	3x3		2205	4401
	4x4		3920	7824
	5x5		6125	12.2K
SAMM	3x3	432	1728	3456
	4x4	768	3072	6144
	5x5	1200	4800	9600

Tabela 3: Comparação da complexidade do hardware para operações de dilatação/erosão

Relatórios Técnicos – 1992

92-01 **Applications of Finite Automata Representing Large Vocabularies,** *C. L. Lucchesi, T. Kowaltowski*

92-02 **Point Set Pattern Matching in d-Dimensions,** *P. J. de Rezende, D. T. Lee*

92-03 **On the Irrelevance of Edge Orientations on the Acyclic Directed Two Disjoint Paths Problem,** *C. L. Lucchesi, M. C. M. T. Giglio*

92-04 **A Note on Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams,** *W. Jacometti*

92-05 **An (l, u)-Transversal Theorem for Bipartite Graphs,** *C. L. Lucchesi, D. H. Younger*

92-06 **Implementing Integrity Control in Active Databases,** *C. B. Medeiros, M. J. Andrade*

92-07 **New Experimental Results For Bipartite Matching,** *J. C. Setubal*

92-08 **Maintaining Integrity Constraints across Versions in a Database,** *C. B. Medeiros, G. Jomier, W. Cellary*

92-09 **On Clique-Complete Graphs,** *C. L. Lucchesi, C. P. Mello, J. L. Szwarcfiter*

92-10 **Examples of Informal but Rigorous Correctness Proofs for Tree Traversing Algorithms,** *T. Kowaltowski*

92-11 **Debugging Aids for Statechart-Based Systems,** *V. G. S. Elias, H. Liesenberg*

92-12 **Browsing and Querying in Object-Oriented Databases,** *J. L. de Oliveira, R. de O. Anido*

Relatórios Técnicos – 1993

93-01 **Transforming Statecharts into Reactive Systems,** *Antonio G. Figueiredo Filho, Hans K. E. Liesenberg*

93-02 **The Hierarchical Ring Protocol: An Efficient Scheme for Reading Replicated Data,** *Nabor das C. Mendonça, Ricardo de O. Anido*

93-03 **Matching Algorithms for Bipartite Graphs,** *Herbert A. Baier Saip, Cláudio L. Lucchesi*

93-04 **A lexBFS Algorithm for Proper Interval Graph Recognition,** *Celina M. H. de Figueiredo, João Meidanis, Célia P. de Mello*

93-05 **Sistema Gerenciador de Processamento Cooperativo,** *Ivonne. M. Carrazana, Nelson. C. Machado, Célio. C. Guimarães*

93-06 **Implementação de um Banco de Dados Relacional Dotado de uma Interface Cooperativa,** *Nascif A. Abousalh Neto, Ariadne M. B. R. Carvalho*

93-07 **Estadogramas no Desenvolvimento de Interfaces,** *Fábio N. de Lucena, Hans K. E. Liesenberg*

93-08 **Introspection and Projection in Reasoning about Other Agents,** *Jacques Wainer*

93-09 **Codificação de Seqüências de Imagens com Quantização Vetorial,** *Carlos Antonio Reinaldo Costa, Paulo Lício de Geus*

93-10 **Minimização do Consumo de Energia em um Sistema para Aquisição de Dados Controlado por Microcomputador,** *Paulo Cesar Centoducatte, Nelson Castro Machado*

93-11 **An Implementation Structure for RM-OSI/ISO Transaction Processing Application Contexts,** *Flávio Morais de Assis Silva, Edmundo Roberto Mauro Madeira*

93-12 **Boole's conditions of possible experience and reasoning under uncertainty,** *Pierre Hansen, Brigitte Jaumard, Marcus Poggi de Aragão*

93-13 **Modelling Geographic Information Systems using an Object Oriented Framework,** *Fatima Pires, Claudia Bauzer Medeiros, Ardemiris Barros Silva*

93-14 **Managing Time in Object-Oriented Databases,** *Lincoln M. Oliveira, Claudia Bauzer Medeiros*

93-15 **Using Extended Hierarchical Quorum Consensus to Control Replicated Data: from Traditional Voting to Logical Structures,** *Nabor das Chagas Mendonça, Ricardo de Oliveira Anido*

93-16 $\mathcal{LL}$ **– An Object Oriented Library Language Reference Manual,** *Tomasz Kowaltowski, Evandro Bacarin*

93-17 **Metodologias para Conversão de Esquemas em Sistemas de Bancos de Dados Heterogêneos,** *Ronaldo Lopes de Oliveira, Geovane Cayres Magalhães*

93-18 **Rule Application in GIS – a Case Study,** *Claudia Bauzer Medeiros, Geovane Cayres Magalhães*

93-19 **Modelamento, Simulação e Síntese com VHDL,** *Carlos Geraldo Krüger e Mário Lúcio Côrtes*

93-20 **Reflections on Using Statecharts to Capture Human-Computer Interface Behaviour,** *Fábio Nogueira de Lucena e Hans Liesenberg*

93-21 **Applications of Finite Automata in Debugging Natural Language Vocabularies,** *Tomasz Kowaltowski, Cláudio Leonardo Lucchesi e Jorge Stolfi*

93-22 **Minimization of Binary Automata,** *Tomasz Kowaltowski, Cláudio Leonardo Lucchesi e Jorge Stolfi*

93-23 **Rethinking the DNA Fragment Assembly Problem,** *João Meidanis*

93-24 EGOLib — **Uma Biblioteca Orientada a Objetos Gráficos,** *Eduardo Aguiar Patrocínio, Pedro Jussieu de Rezende*

93-25 **Compreensão de Algoritmos através de Ambientes Dedicados a Animação,** *Rackel Valadares Amorim, Pedro Jussieu de Rezende*

93-26 GeoLab: **An Environment for Development of Algorithms in Computational Geometry,** *Pedro Jussieu de Rezende, Welson R. Jacometti*

93-27 **A Unified Characterization of Chordal, Interval, Indifference and Other Classes of Graphs,** *João Meidanis*

93-28 **Programming Dialogue Control of User Interfaces Using Statecharts,** *Fábio Nogueira de Lucena e Hans Liesenberg*

93-29 EGOLib – **Manual de Referência,** *Eduardo Aguiar Patrocínio e Pedro Jussieu de Rezende*

Relatórios Técnicos – 1994

94-01 **A Statechart Engine to Support Implementations of Complex Behaviour,** *Fábio Nogueira de Lucena, Hans K. E. Liesenberg*

94-02 **Incorporação do Tempo em um** SGBD **Orientado a Objetos,** *Ângelo Roncalli Alencar Brayner, Claudia Bauzer Medeiros*

94-03 **O Algoritmo KMP através de Autômatos,** *Marcus Vinícius A. Andrade e Cláudio L. Lucchesi*

94-04 **On Edge-Colouring Indifference Graphs,** *Celina M. H. de Figueiredo, João Meidanis, Célia Picinin de Mello*

94-05 **Using Versions in** GIS, *Claudia Bauzer Medeiros and Geneviève Jomier*

94-06 **Times Assíncronos: Uma Nova Técnica para o** Flow Shop Problem, *Hélvio Pereira Peixoto e Pedro Sérgio de Souza*

94-07 **Interfaces Homem-Computador: Uma Primeira Introdução,** *Fábio Nogueira de Lucena e Hans K. E. Liesenberg*

94-08 **Reasoning about another agent through empathy,** *Jacques Wainer*

94-09 **A Prolog morphological analyser for Portuguese,** *Jacques Wainer, Alexandre Farcic*

94-10 **Introdução aos Estadogramas,** *Fábio N. de Lucena, Hans K. E. Liesenberg*

94-11 **Matching Covered Graphs and Subdivisions of K_4 and $\overline{C_6}$,** *Marcelo H. de Carvalho and Cláudio L. Lucchesi*

94-12 **Uma Metodologia de Especificação de Times Assíncronos,** *Hélvio Pereira Peixoto, Pedro Sérgio de Souza*

Relatórios Técnicos – 1995

95-01 **Paradigmas de algoritmos na solução de problemas de busca multidimensional,** *Pedro J. de Rezende, Renato Fileto*

95-02 **Adaptive enumeration of implicit surfaces with affine arithmetic,** *Luiz Henrique de Figueiredo, Jorge Stolfi*

95-03 **W3 no Ensino de Graduação?,** *Hans Liesenberg*

95-04 **A greedy method for edge-colouring odd maximum degree doubly chordal graphs,** *Celina M. H. de Figueiredo, João Meidanis, Célia Picinin de Mello*

95-05 **Protocols for Maintaining Consistency of Replicated Data,** *Ricardo Anido, N. C. Mendonça*

95-06 **Guaranteeing Full Fault Coverage for UIO-Based Methods,** *Ricardo Anido and Ana Cavalli*

95-07 **Xchart-Based Complex Dialogue Development,** *Fábio Nogueira de Lucena, Hans K.E. Liesenberg*

95-08 **A Direct Manipulation User Interface for Querying Geographic Databases,** *Juliano Lopes de Oliveira, Claudia Bauzer Medeiros*

95-09 **Bases for the Matching Lattice of Matching Covered Graphs,** *Cláudio L. Lucchesi, Marcelo H. Carvalho*

Departamento de Ciência da Computação — IMECC
Caixa Postal 6065
Universidade Estadual de Campinas
13081-970 – Campinas – SP
BRASIL
reltec@dcc.unicamp.br