PRIMATA: Uma Arquitetura Maciçamente Paralela para Processamento de Imagens e Matrizes

Maria Fernanda Eppinghaus Belford Roxo[1]

Adriano Joaquim de Oliveira Cruz[2]

Otto Carlos Muniz Bandeira Duarte[3]

Roberto Castelo Branco Jorge[4]

Resumo

Este trabalho apresenta uma visão geral do projeto PRIMATA (Processador de Imagens e Matrizes), descrevendo a arquitetura dos seus dois principais módulos: a Unidade de Controle e o Elemento Processador. Também são apresentados alguns resultados de desempenho, comparando-os com os de outras arquiteutras SIMD.

Abstract

This work presents an overview of the PRIMATA project, describing the architecture of the two main modules, Control Unit and Processing Element. It also presents some performance results, comparing them to the other SIMD architectures.

[1]M.C.; Pesquisadora NCE/UFRJ; E-mail: fernanda@nce.ufrj.br, Endereço: NCE/UFRJ, Cx. Postal 2324, Rio de Janeiro, RJ, CEP 20001-860

[2]Ph.D.; Pesquisador NCE/UFRJ; E-mail: adriano@nce.ufrj.br

[3]Dr. Eng.; Professor COPPE/UFRJ; E-mail: otto@coe.ufrj.br

[4]M.C.; Pesquisador IPD/CTEx; E-mail: s3castel@imerj.bitnet, Endereço: IPD/CTEx, Av. das Américas, 28705, Rio de Janeiro, RJ, CEP 23020-470

1- Introdução

O PRIMATA (Processador de Imagens e Matrizes) é uma arquitetura maciçamente paralela de alto desempenho e baixo custo orientada para o processamento de imagens e matrizes em geral. Dentro deste contexto é utilizada uma arquitetura SIMD (*Single Instruction Stream Multiple Data Stream*), onde um grande número de Elementos Processadores (EPs) atuam sob o controle de uma única Unidade de Controle (UC). Ou seja, todos os EPs executam a mesma instrução simultaneamente, porém cada um sobre os seus próprios dados. Este tipo de arquitetura tem como principais vantagens o baixo custo individual de cada EP, a fácil migração dos programas seqüenciais existentes e a perfeita adaptação às aplicações em vista.

Em arquiteturas SIMD, o desempenho depende basicamente do produto do número de EPs pela velocidade individual de cada EP. Sendo assim, o desempenho individual não é crítico. O importante é manter um compromisso entre o custo e o desempenho do EP, a fim de viabilizar o fator "número de EPs".

A primeira geração de arquiteturas SIMD - como o DAP [Red73], o MPP [Bat80], a Connection Machine [Hil85] e mais recentemente o BLITZEN [Dav88] - se restringiu a processadores de 1 *bit* para permitir a construção de um enorme número de EPs, tirando deste fator o seu poder computacional.

As arquiteturas mais recentes - como o MasPar MP-1 [Bla90] e o DAP CP/8 [AMT90] - mostram uma tendência à utilização de processadores de mais de 1 *bit*. Esta tendência é um reflexo do avanço da tecnologia utilizada na implementação dos EPs. Mesmo desconsiderando-se este avanço tecnológico, uma análise mais detalhada da relação entre o custo e o desempenho dos EPs de 1 *bit* é capaz de mostrar que arquiteturas de 4 *bits* podem manter ou até melhorar esta relação.

O PRIMATA segue esta tendência, optando por um processador de 4 *bits*. Vale ressaltar que a arquitetura do Elemento Processador do PRIMATA é o resultado de um estudo [RCD93][Rox94] que analisou o custo e o desempenho de várias propostas de processadores, inclusive alguns processadores de 1 *bit*.

Este trabalho apresenta a arquitetura do PRIMATA, discute a sua implementação e tece uma comparação entre o seu desempenho e o de outras arquiteturas comerciais.

2- Arquitetura do PRIMATA

Como o próprio nome sugere, o PRIMATA destina-se, principalmente, ao processamento de matrizes e sinais bidimensionais, como imagens. Por isto, optou-se pelo desenvolvimento de uma arquitetura maciçamente paralela tipo SIMD, com um arranjo bidimensional de até 256x256 EPs.

Mostrado na Figura 1, o PRIMATA é composto, basicamente, dos seguintes módulos:

a) Matriz de Elementos Processadores (MEP);
b) Unidade de Controle (UC);
c) Unidade de Formatação de Dados (UFD);
d) Memórias *dual-port* para Dados e para Programas (MD e MP); e
e) Interface para o computador hospedeiro.

A MEP reúne um conjunto de até 65.536 EPs dispostos na forma de uma matriz bidimensional. Cada EP pode transmitir ou receber dados de qualquer um dos seus oito vizinhos (N,NE,E,SE,S,SO,O,NO) e possui, associado a ele, uma memória de 64k x 4 *bits*. A MEP também pode assumir a forma de um cilindro horizontal ou vertical, assim como a de um toróide, fazendo, por exemplo, a aresta norte ser conectada à aresta sul e a aresta oeste ser conectada à aresta leste.

A UC [Jor94] tem por funções buscar as instruções do programa, decodificá-las e enviar as correspondentes microinstruções dos EPs para a MEP. Além das microinstruções que envia para a MEP, a UC também é responsável pela geração dos endereços do banco de memórias associado a ela e pela execução das chamadas "instruções escalares".

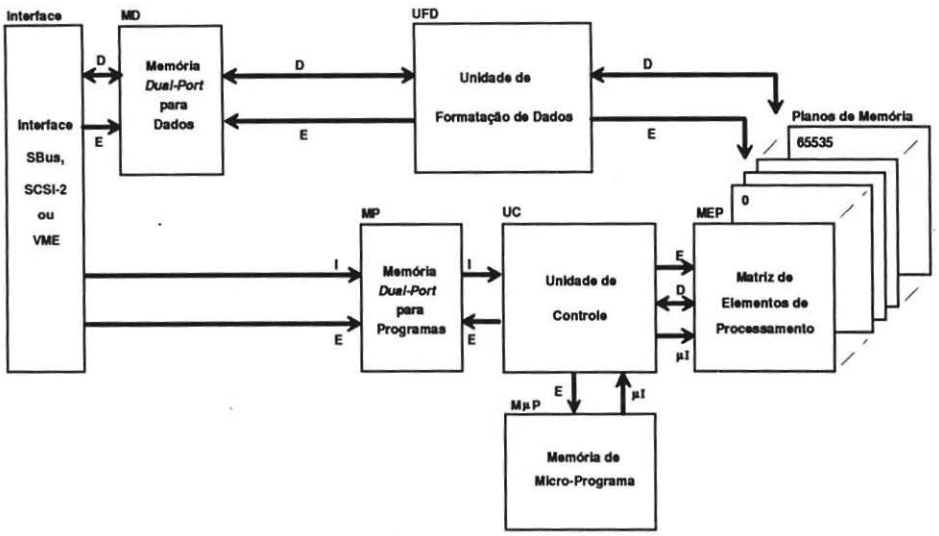

Figura 1: Diagrama em Bloco do PRIMATA.

As instruções escalares são aquelas que não afetam os dados armazenados na memória dos EPs, como, por exemplo, a contagem de um *loop*. Muitas delas determinam a seqüência das chamadas instruções vetoriais e, como o número de desvios e contagens de *loops* é grande, mesmo numa arquitetura SIMD, a UC deve ter capacidade de processá-los eficientemente para não comprometer o desempenho global do sistema.

Visando dar maior flexibilidade a todo o sistema, em todas as implementações consideradas, manteve-se sempre na UC a característica da microprogramabilidade. Assim, o conjunto de instruções pode variar segundo as necessidades do usuário e incluir, inclusive, rotinas usadas freqüentemente

A UFD tem por finalidade coordenar a transferência de dados bidirecional entre o computador hospedeiro e o banco de memórias associado à MEP. Ela também é encarregada, como o prório nome sugere, de converter os dados da forma serial-palavra para paralelo-*nibble*, e vice-versa.

Para isto, ela assume o controle do barramento de endereços das memórias dos EPs sempre que a UC não estiver enviando endereços e houver necessidade de se transferir dados de/para o hospedeiro. Ela pode fazer, por exemplo, operações de transferência no mesmo instante em que os EPs realizam operações com os dados armazenados nos seus registradores internos.

Os dados para o banco de memórias e os programas para a UC são enviados pelo hospedeiro para memórias *dual-port*, acessadas também pela UFD e pela UC, respectivamente. A comunicação com o hospedeiro é feita através de uma interface que implemente algum padrão largamente aceito, como SBus, VME ou SCSI-2. Desta forma, é possível garantir que um grande variedade de computadores poderão ser utilizados como hospedeiros do PRIMATA.

3- Unidade de Controle

A UC é dividida em três seções, a saber: seção de seqüenciamento de instruções, seção de seqüenciamento de microinstruções e seção de endereçamento da memória dos EPs. As duas primeiras

recebem as microordens de uma única MμP, que também as envia para a seção de endereçamento da memória dos EPs e para o próprio arranjo de EPs.

A busca da próxima instrução é feita a partir da determinação do endereço da mesma. O endereço da próxima instrução pode ser proveniente de três fontes possíveis: do topo de uma pilha (P1) de 16 posições de 16 *bits*, do campo de valor imediato do Registrador de Instrução (RI), ou, da forma mais usual, pelo incremento do endereço atual (PC).

A seleção é feita por intermédio de um multiplexador (M1) 3x1x16, dando origem às 16 linhas AP_0-AP_{15} ligadas ao barramento de endereços da MP. O dado lido, isto é, a instrução, é armazenada no RI.

Uma vez armazenada no RI, a instrução tem os seus 10 *bits* mais significativos (de um total de 32), correspondentes ao *op-code*, enviados para o seqüenciador de microinstruções.

De forma análoga ao seqüenciador de instruções, o seqüenciador de microinstruções também pode determinar o valor do próximo endereço a ser lido a partir da seleção de diversas origens. Porém, ao invés de três, para esta seção são quatro as possíveis fontes.

Além de poder gerar o próximo endereço a partir do incremento do atual (MPC), e da leitura do valor do topo de uma pilha (P2), esta configuração prevê que o próximo endereço seja proveniente, também, do campo imediato do Registrador de MicroInstruções (RμI) ou do campo de *op-code* do RI.

Um multiplexador (M3) 4x1x16 seleciona uma destas fontes, conforme microordens enviadas para suas linhas de controle. No caso da execução de uma nova instrução chegada ao RI, este *mux* seleciona a entrada correspondente ao campo de *op-code* do RI, que aponta para o endereço onde começa a seqüência de microinstruções correspondentes a esta instrução.

De um modo geral, os endereços apontados pelo *op-code* contém, apenas, uma microinstrução de desvio para o endereço onde começa, de fato, a seqüência das microinstruções. Assim, é possível a UC poder decodificar e executar até 1024 instruções ou rotinas diferentes, armazenadas na MμP.

Além de endereços de microinstruções, a pilha deste seqüenciador pode também armazenar dados (de 8 *bits*), provenientes de um contador. Este recurso é muito útil em casos de *loops* simultâneos, em diversos níveis.

A seção de endereçamento das memórias dos EPs, por sua vez, possui um banco de 16 registradores de 4 *bits* (R0-R15), além de 4 registradores de 16 *bits* (E0-E3), responsáveis por formar os endereços para as memórias dos EPs. Com esta combinação, é possível endereçar até 4 operandos diferentes, suficientes para a maioria das operações, inclusive a divisão.

O Banco de Registradores R pode receber dados provenienentes de 4 fontes: parte do campo imediato do RI, saída do circuito somador (S1), valor do Registrador Auxiliar ou parte do campo imediato do RμI. Esta seleção é feita por intermédio de microordens que habilitam uma das fontes a ocupar o espaço o barramento (de 4 *bits*).

Também derivados do banco de registradores R são os dados contidos no contador C1 e os selecionados pelo *mux* de endereçamento relativo. No primeiro caso, os 3 *bits* de dados do contador C1 (S0-S2) alimentam as linhas de controle do banco de registradores A dos EPs. No segundo caso, os 4 *bits* provenientes do *mux* informam, se selecionados pelo *mux*, os 4 *bits* menos significativos do endereço dos dados armazenados nas memórias dos EPs.

A implementaçao da UC será feita usando um FPGA (*Field Programmable Gate Array*) modelo XC4006, da Xilinx, de modo a ocupar pequeno espaço em placa, além de garantir bom desempenho e flexibilidade para alterações futuras. Outras opções de implementação (como EPLDs e CIs MSI) foram consideradas. Porém, face à necessidade da existência das pilhas e bancos de registradores, elas foram descartadas, pois necessitariam de vários circuitos integrados, aumentando a complexidade da placa e reduzindo a velocidade global da UC.

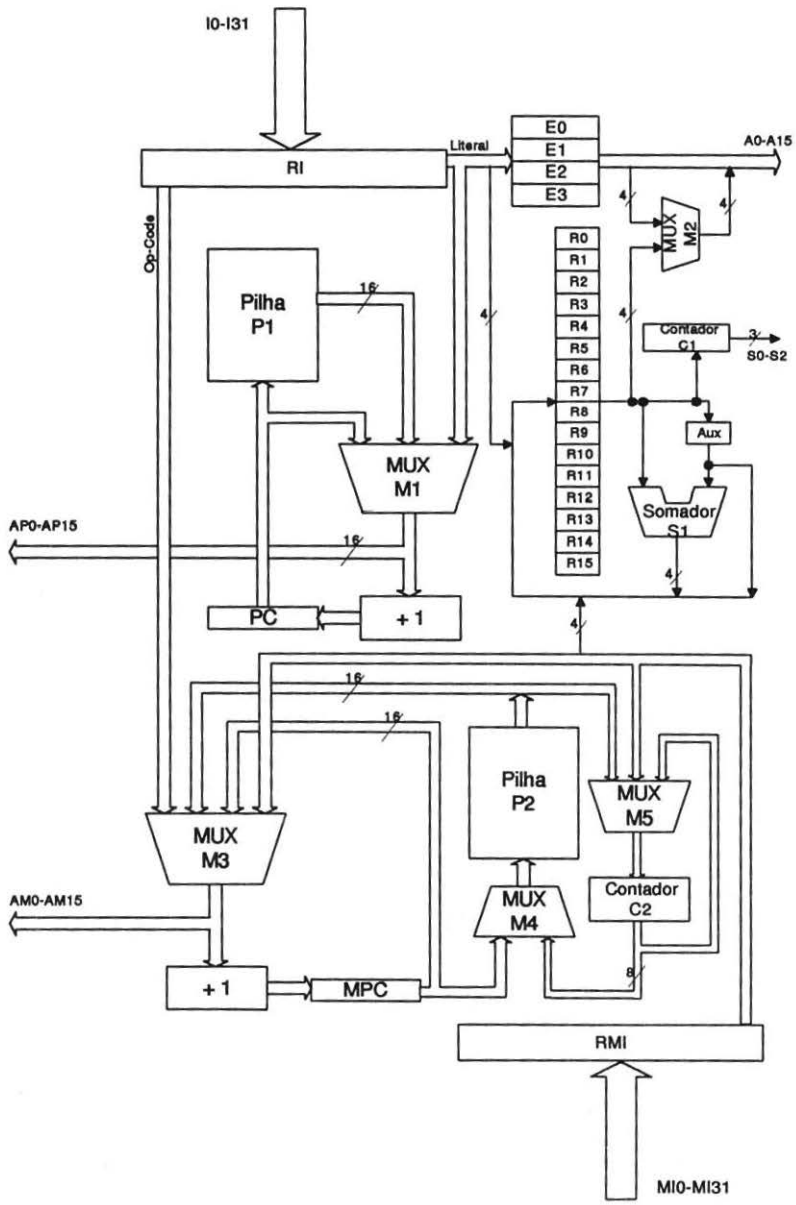

Figura 2: Arquitetura da Unidade de Controle.

4- Matriz de Elementos Processadores

A organização espacial dos processadores do PRIMATA tem os EPs dispostos em forma de matriz. A definição da comunicação de dados entre EPs levou em consideração que nem todos os processadores estarão no mesmo *chip* e nem todos os processadores estarão na mesma placa. Ou seja, é fundamental analisar o peso das ligações entre placas e, principalmente, entre *chips*. Um aumento das ligações entre *chips* implica num maior número de pinos e, conseqüentemente, num aumento da área do *chip*.

O mecanismo de troca de dados entre EPs encontrado na maioria das arquiteturas SIMD é a comunicação *bit* a *bit* entre cada EP e seus quatro vizinhos mais próximos: norte (N), sul (S), este (E) e oeste (O). Para o encapsulamento de n^2 EPs dispostos numa matriz n x n, este mecanismo acarreta na utilização de 4n pinos só para a comunicação entre EPs. Por exemplo, no encapsulamento de 16 EPs, 16 pinos seriam usados para a comunicação de dados. A possibilidade de utilizar canais de 4 *bits* ao invés de canais de 1 *bit* é descartada, pois isto quadruplicaria o número de pinos gastos para a comunicação de dados. Para o mesmo exemplo anterior, seriam gastos 64 pinos, ao invés de 16, o que aumentaria absurdamente o custo do *chip*.

Ao invés disto optou-se pela utilização de uma rede de interconexões em X de 1 *bit*, similar a do Blitzen. Cada EP está associado a 4 linhas direcionadas aos seus vizinhos diagonais: NE, SE, SO e NO (Figura 3). Na interseção das linhas que se cruzam há uma conexão física. Isto permite que cada EP se comunique com seus 8 vizinhos mais próximos: norte (N), nordeste (NE), este (E), sudeste (SE), sul (S), sudoeste (SO), oeste (O) e noroeste (NO). A combinação das direções de envio e recebimento definem o fluxo da comunicação de dados. Por exemplo, para estabelecer um fluxo de dados na direção este (E), basta que cada EP envie dados na direção nordeste (NE) e receba dados da direção noroeste (NO). Este mecanismo de comunicação tem o mesmo custo que a comunicação de 1 *bit* com os quatro vizinhos, porém dobra o número de vizinhos a quem se está ligado diretamente.

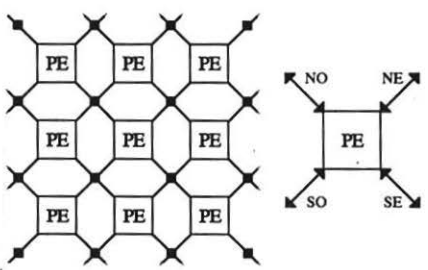

Figura 3: Rede de interconexões entre EPs.

A comunicação de dados entre a matriz de processadores e a Unidade de Controle é feita através de um árvore *sum-OR*. Por este mecanismo, cada EP envia o conteúdo do seu barramento principal de 1 *bit* (b1) para a UC. Todos os *bits* enviados passam por um árvore de portas OU-inclusivo, gerando um único *bit* de sinalização para a UC. O *bit* recebido permite à Unidade de Controle tomar decisões baseadas em resultados obtidos por um ou mais EPs. Uma aplicação deste mecanismo é o tratamento de erros e exceções durante o processamento. Por exemplo, numa operação de divisão pode-se identificar facilmente se houve uma exceção do tipo divisão por zero em algum dos processadores. Uma outra aplicação é utilizar os resultados de uma operação na definição de um teste de parada de um laço. Ou seja, só encerrar a execução do laço quando todos os processadores tiverem atingido um determinado objetivo.

5- Elemento Processador

A Figura 4 apresenta um diagrama funcional do Elemento Processador (EP) do PRIMATA. As linhas mais largas representam barramentos ou unidades de 4 *bits*, enquanto que as linhas mais finas representam barramentos ou unidades de 1 *bit*. Cada EP contém: dois registradores de 1 *bit* (K e C), um registrador de 4 *bits* (B), um registrador de deslocamento de 4 *bits* (SH) e um banco de registradores (A). Este último é composto por oito registradores de deslocamento de 4 *bits* interligados, formando um registrador de deslocamento de 32 *bits* (Figura 5). Cada EP está ainda associado à uma memória externa de acesso randômico (M). Para auxiliar a execução das operações lógicas e aritméticas são utilizados: uma unidade lógica de 1 *bit* (LK), um operador lógico *not-zero* (NZ), uma unidade lógica de 4 *bits* (L) e um somador de 4 *bits* (S).

Cada EP possui dois barramentos principais: o barramento de 1 *bit* (b1) e o barramento de 4 *bits* (b4). Pode-se dividir a arquitetura do EP em dois blocos principais: um bloco de 1 *bit* e um bloco de 4 *bits*. O processador provê operações de 1 e 4 *bits* simultâneas. Os elementos responsáveis pela comunicação de dados entre estes dois blocos são: o registrador de deslocamento SH, o operador NZ e um fio que liga o *bit* mais significativo do barramento b4 ao barramento b1.

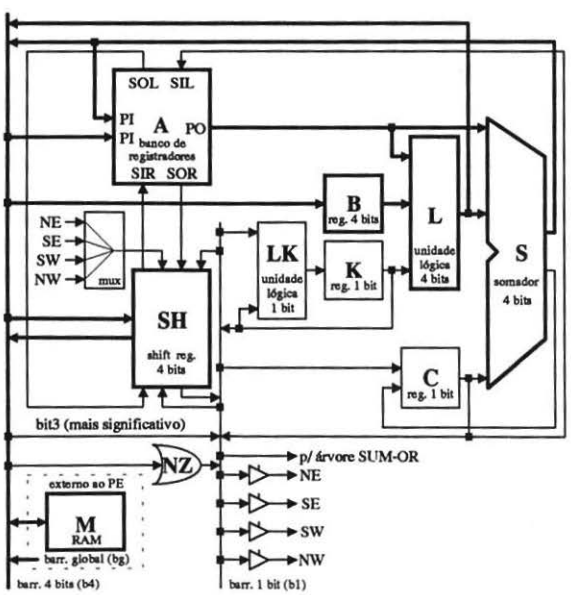

Figura 4: EP4BR - Elemento Processador de 4 *bits* com banco de registradores.

5.1- Unidade Aritmética e Banco de Registradores

O registrador C é usado para armazenar o *carry* de operações aritméticas. Ele pode ser carregado com '1' (um), '0' (zero), o valor de b1, ou a saída de *carry* do somador S. O somador S executa a soma

entre: o resultado da operação lógica executada por L, o valor armazenado em C e, opcionalmente, o valor lido do banco de registradores A.

O registrador B é utilizado para armazenar um dos operandos de uma operação lógica ou aritmética. O outro operando é armazenado no banco de registradores A. Este banco de registradores tem a função de uma memória interna, com a vantagem de permitir deslocamento de dados, o que facilita enormemente a execução de algumas operações aritméticas, como a multiplicação e a divisão, por exemplo. Apenas as operações de deslocamento (para a esquerda ou para a direita) são executadas simultaneamente em todos os registradores que compõem o banco. As operações de leitura e escrita são executadas em apenas um registrador de cada vez. Para selecionar qual registrador do banco será lido ou escrito, são utilizados 3 *bits* de controle.

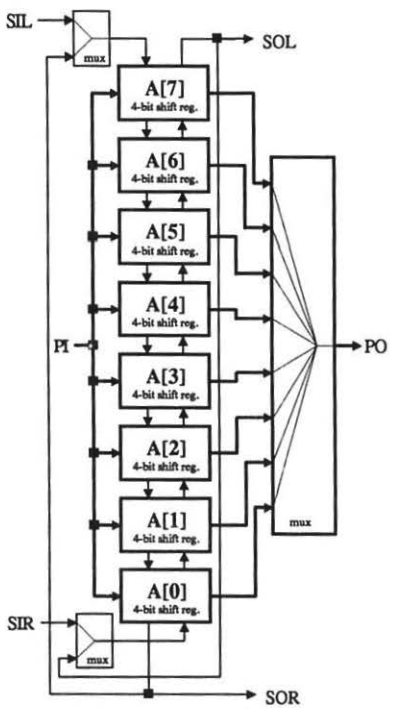

Figura 5: Banco de Registradores A.

5.2- Registrador de Máscara

O registrador K é chamado de registrador de máscara. Ele é uma característica das arquiteturas SIMD e permite um controle individual de cada processador. Através do registrador de máscara é possível implementar de forma simples uma estrutura do tipo SE ... FAÇA .... Embora esta estrutura pareça simples em arquiteturas seriais, em uma arquitetura SIMD ela não seria trivial sem o auxílio de um registrador de máscara, já que todos os processadores executam a mesma instrução. Tradicionalmente, armazena-se a

condição no registrador de máscara e este inibe diretamente a instrução condicional, que pode ser uma soma, ou uma escrita em um registrador ou na memória, por exemplo.

O registrador K tem um funcionamento um pouco mais sofisticado. A condição também é armazenada em K, mas é possível implementar estruturas do tipo ONDE <condição> FAÇA <expressão1> NO_RESTANTE <expressão2>. Para isto, ao invés de simplesmente inibir uma operação, ligou-se o registrador K à unidade lógica L, permitindo que, através da escolha de uma operação lógica adequada, execute-se operações diferentes de acordo com o resultado da condição armazenado em K. Por exemplo, no algoritmo de divisão sem restauração, se o resultado da última operação de soma/subtração for positivo, a próxima operação será uma subtração, caso contrário será uma soma, como no algoritmo apresentado abaixo:

```
...
para i variando de 1 até n
    se A ≥ 0, então
        A = A - B
    senão
        A = A + B
    ...
```

Se fosse usado um mecanismo de inibição tradicional, o algoritmo para uma arquitetura SIMD seria o seguinte:

```
...
para i variando de 1 até n
    K = (A ≥ 0)
    se K, então A = A - B
    K = not K
    se K, então A = A + B
    ...
```

Ou seja, seria gasto o tempo de uma soma mais o tempo de uma subtração, para que todos os processadores executassem a operação correta. Já no PRIMATA o algoritmo seria o seguinte:

```
...
para i variando de 1 até n
    K = C = (A ≥ 0)
    A = A + B xor K + C
    ...
```

Ou seja, se K=0 (A < 0) é executada a soma A=A+B. Por outro lado, se K=1 (A≥0) é executada a soma A=A+/B+1, o que equivale a subtração A=A-B. Sendo assim, gasta-se apenas o tempo de uma soma ou subtração, ou seja, metade do tempo que seria gasto se fosse usado um mecanismo de mascaramento tradicional.

6- Desempenho

Para comparar o desempenho do PRIMATA com o de outras arquiteturas comerciais, foram utilizados os algoritmos desenvolvidos para a execução de operações aritméticas sobre números inteiros e sobre números reais. Para os números inteiros foi utilizada a representação em complemento a dois. Já para os números reais foi utilizado o padrão IEEE 754 [Coo80] para operações aritméticas em ponto-flutuante. Durante o desenvolvimento dos algoritmos para as operações de ponto-flutuante, foi observado que duas modificações no padrão IEEE representavam um aumento significativo no desempenho destes algoritmos.

A primeira modificação refere-se a regra do arredondamento. Seja Z o resultado com precisão infinita de uma operação aritmética. Sejam ainda Z1 e Z2 os números representáveis na precisão do formato IEEE mais próximos de Z, tais que $Z1 \leq Z \leq Z2$. O padrão Round to Nearest manda que Z seja arredondado para o número mais próximo a ele, entre os valores Z1 e Z2. Em caso de empate, ou seja, Z está tão próximo de Z1 quanto de Z2, deve-se escolher aquele cujo *bit* menos significativo da mantissa é zero. A dificuldade encontra-se na regra em caso de empate, que introduz quatro operações lógicas no cálculo do arredondamento. Para simplificar a execução do arredondamento, presente no algoritmo de todas as operações aritméticas sobre números reais, optou-se por, em caso de empate, arredondar para o número de maior magnitude, entre Z1 e Z2. Esta pequena modificação simplifica a operação de arredondamento a um teste de 1 *bit* e um incremento condicional ao resultado deste teste.

A segunda e mais importante modificação com relação ao desempenho foi a eliminação do formato denormalizado, isto é, valores com expoente mínimo (igual a -128 no caso do formato *single*) e mantissa com parte inteira nula. Com isto foram eliminadas as operações de normalização dos operandos e denormalização do resultado. Considerando-se que cada operação de normalização ou denormalização envolve deslocamentos de até 24 *bits* na mantissa e até 24 incrementos ou decrementos do expoente, esta modificação significou um considerável aumento de desempenho. Por outro lado, a perda do formato denormalizado não é significativa, visto que o seu intervalo de representação, além de bastante limitado, apresenta ainda uma descontinuidade em relação ao intervalo de representação do formato normalizado.

Assim como o PRIMATA, a maioria das máquinas comerciais também oferece operações de ponto-flutuante utilizando o padrão IEEE com pequenas modificações. O MasPar, por exemplo, implementou exatamente as mesmas modificações que o PRIMATA.

Para a comparação de desempenho foram escolhidas as operações de soma/subtração e multiplicação de inteiros e de reais. A Tabela 1 apresenta o número de ciclos de máquina necessários à execução de cada uma destas operações. Para as operações inteiras são considerados operandos de n *bits* e para as operações reais são considerados operandos no formato *single* do padrão IEEE, ou seja, operandos de 32 *bits*.

Tabela 1: Desempenho do PRIMATA na execução de operações aritméticas.

Operação	Ciclos de Máquina
Soma/Subtração de Inteiros	$3n/4+1$
Multiplicação de Inteiros	p/ $n \leq 28$: $4(n/4)^2+7n/4+2$ p/ $n=32$: $4(n/4)^2+9n/4$
Soma/Subtração de Reais	219
Multiplicação de Reais	273

Para que a comparação entre as arquiteturas se concentre na capacidade do processador, independente da tecnologia utilizada na implementação do mesmo, do período do relógio ou do número de EPs do sistema; foi utilizada como base de comparação o número de ciclos de máquina necessários à execução de cada operação aritmética. Os gráficos das Figuras 6 e 7 comparam o desempenho do PRIMATA com o de outras arquiteturas comerciais na execução de operações aritméticas sobre inteiros de 8 *bits* e reais de 32 *bits* (formato *single* do padrão IEEE). O desempenho é apresentado em termos de ciclos de máquina.

Uma breve análise das duas figuras mostra que o PRIMATA ganha em desempenho de quase todas as arquiteturas, perdendo apenas para o MasPar MP-1. Porém este último tem um custo bem mais elevado do que o do PRIMATA. Ele tem mais do que 1.536 *bits* de memória local, contra apenas 42 *bits* do PRIMATA. Além disto, a sua lógica é bem mais complexa. Ele possui unidades especiais para o

processamento do expoente e para o processamento da mantissa, além de uma unidade para modificação local do endereçamento. Para se ter uma idéia, o custo estimado do PRIMATA é de cerca de 1.900 transistores por EP. Já cada EP do MasPar tem cerca de 14.000 transistores, dos quais 75% (10.500 transistores) são utilizados na implementação dos registradores internos e apenas 25% (3.500 transistores) são utilizados na implementação da lógica do EP.

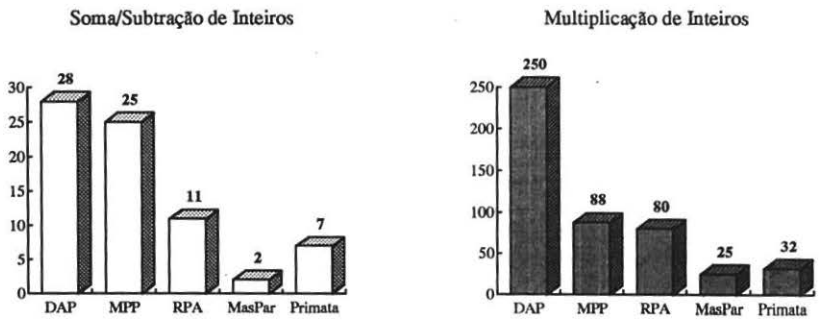

Figura 6: Desempenho em número de ciclos para operações aritméticas inteiras.

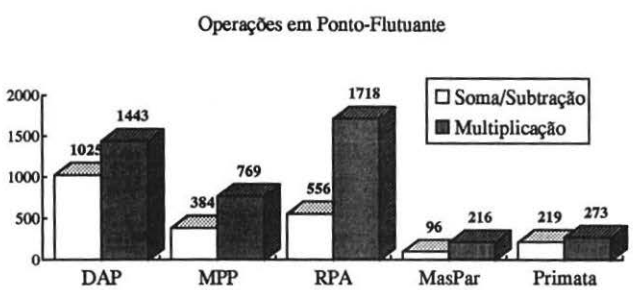

Figura 7: Desempenho em número de ciclos para operações aritméticas reais.

Fazendo-se uma análise da relação Custo x Benefício do MasPar em comparação com a do PRIMATA, percebe-se nítida vantagem deste último. A Tabela 2 apresenta uma relação Custo x Benefício dos dois processadores para as operações aritméticas analisadas. Foi considerado como custo do PRIMATA o número de transistores estimados por EP e do MasPar apenas os transistores associados à lógica do EP (3.500 transistores). Uma análise do custo total do EP do MasPar (14.000 transistores) levaria a uma vantagem ainda maior do PRIMATA, mas poder-se-ia alegar que a grande memória interna do MasPar traz como vantagem uma menor dependência de comunicação com a memória externa. Foi considerado como benefício o inverso do número de ciclos de relógio gastos para a execução de cada operação.

Para facilitar a comparação a relação Custo x Benefício foi normalizada para o PRIMATA, ou seja, esta relação foi considerada unitária para este processador em cada uma das operações. Deve-se ainda considerar que o desempenho do MasPar nas operações aritméticas inteiras foi obtido supondo que um dos operandos está num registrador especial e que o resultado também será armazenado neste mesmo registrador, o que é uma restrição significativa. Isto equivaleria a supor que, no caso do PRIMATA, um dos operandos encontra-se no banco de registradores A e o resultado também será colocado neste mesmo banco de registradores. Esta suposição acarretaria num aumento substancial da velocidade do PRIMATA, principalmente na soma/subtração de inteiros, que teria um desempenho similar ao do PRIMATA a um custo bem inferior. Ou seja, a superioridade do MasPar não justifica o seu elevado custo. Ele só se destaca na operação de soma/subtração de inteiros e apenas devido ao artifício da medição, perdendo claramente nas operações de multiplicação de tanto de inteiros quanto de ponto-flutuante.

Tabela 2: Relação Custo x Benefício do MasPar e do PRIMATA normalizada.

Operação	MasPar	PRIMATA
Soma/Subtração de Inteiros	0,53	1,00
Multiplicação de Inteiros	1,44	1,00
Soma/Subtração de Reais	0,81	1,00
Multiplicação de Reais	1,45	1,00

7- Perspectivas Futuras

Um protótipo da arquitetura com 16 EPs organizados em uma matriz 4 x 4 está sendo implementado utilizando EPLDs (*Erasable Programable Logical Devices*). Estes dispositivos apresentam como vantagens serem programáveis e apagáveis, o que facilita a implementação de modificações durante a fase de desenvolvimento do projeto. Além disto eles são versáteis, possuindo um bom número de *flip-flops* e pinos de entrada e saída. Mas o principal motivo que levou à escolha destes dispositivos foi a disponibilidade não só dos próprios *chips*, como também das ferramentas de desenvolvimento e gravação dos mesmos.

A implementação dos EPs em EPLDs da linha MAX5000 da Altera [Alt91] não obteve resultados satisfatórios, principalmente devido a limitada lógica combinacional destes dispositivos. Só foi conseguido implementar um único EP em cada *chip*. Atualmente está sendo desenvolvido um EP utilizando EPLDs da linha MAX7000, que tem melhores recursos de lógica combinacional, onde se espera um resultado melhor, principalmente em termos de atraso.

Para uma versão definitiva porém, seria interessante a integração em VLSI de vários EPs em um único *chip*. Isto diminuiria o custo por EP e a área ocupada pela matriz de EPs, permitindo a construção de uma matriz de EPs de maiores dimensões.

Numa análise superficial do número de transistores necessários à implementação de cada EP, poder-se-ia pensar que é possível integrar um número muito grande de EPs em um único *chip*. Porém, além das limitações de transistores, neste caso os *chips* possuem uma limitação maior que se refere ao número de pinos. Além dos *bits* de controle, comuns a todos os EPs, cada EP é responsável pela inclusão de mais quatro pinos para a comunicação com a sua memória externa, fora os pinos gastos com a comunicação de dados entre EPs.

Uma solução para este problema é a utilização de uma memória interna maior, aproveitando a maior disponibilidade de transistores. Por esta solução, todas as operações têm operandos e resultados armazenados nesta memória. A memória externa é compartilhada por vários ou todos os EPs de um *chip*, economizando pinos de comunicação com a memória. Para trocar dados entre as memórias interna e

externa, passam a ser necessários vários ciclos, já que a memória externa tem que se comunicar com várias memórias internas. Sendo assim a memória interna deve ser dimensionada de forma a minimizar este tipo de comunicação e esta comunicação deve ser paralela a outros processamentos internos do EP.

Referências

[Alt91] Altera, *Altera Data Book*, Altera Corporation, 1991.

[AMT90] Active Memory Technology Ltd (AMT), *DAP Series Technical Overview, Introducing the DAP/CP8 range*, Sales Suport Note 7, Abril de 1990.

[Bat80] K. E. Batcher, *Design of a Massively Parallel Processor*, IEEE Transactions on Computers, pp 837-840, Setembro de 1980.

[Bla90] T. Blank, *The MasPar MP-1 Architecture*, Proceedings of the 35th IEEE Computer Society International Conference-Spring COMPCON 90, pp 20-24, San Francisco, CA, Fevereiro de 1990.

[Coo80] J. T. Coonen, *An Implementation Guide to a Proposed Standard for Floating-Point Arithmetic*, IEEE Transactions on Computers, pp 68-79, Janeiro de 1980.

[Dav88] E. W. Davis e J. H. Reif, *Architecture and Operation of the BLITZEN Processing Element*, Proc. of the 3rd International Conference on Supercomputing, Boston, MA, pp128-137, Maio de 1988.

[Hil85] W. D. Hillis, *The Connection Machine*, The MIT Press, Massachussets, 1985.

[Jor94] R. C. B. Jorge, *Uma Unidade de Controle para Arquiteturas Paralelas SIMD*, Tese de Mestrado, IME, Rio de Janeiro, RJ, Janeiro de 1994.

[RCD93] M. F. E. B. Roxo; A. J. O. Cruz; O. C. M. B. Duarte, *PRIMATA: Desenvolvimento de um Elemento Processador para uma Arquitetura Massivamente Paralela*, Anais do V Simpósio Brasileiro de Arquitetura de Computadores - Processamento de Alto Desempenho SBAC-PAD, Florianópolis, SC, Volume I, pp 150-165, Setembro de 1993

[Red79] S. F. Reddaway, *The DAP Approach*, Infotech State of the Art Report: Supercomputers Vol2, pp 311-329, 1979.

[Rox94] M. F. E. B. Roxo, *PRIMATA: Um Processador Maciçamente Paralelo*, Tese de Mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Março de 1994.