UMA ARQUITETURA RISC PARA COMPUTADORES SIMD EM GaAs

José Eduardo Moreira
Laboratório de Sistemas Integráveis
Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, 158 - Trav. 3
CEP 05508 - Sao Paulo - SP

RESUMO

Uma arquitetura simples o suficiente para ser implementada em poucos CIs de GaAs e ainda assim bastante poderosa para tratamento de matrizes é descrita. A linguagem APL e o computador Illiac IV são tomados como base para a arquitetura. A implementação eletrônica é levada em consideração.

ABSTRACT

An architecture is described which is simple enough for implementing with a few GaAs ICs, yet very powerful for matrix processing. The APL language and the Illiac IV computer are taken as the basis for the architecture. The electronic implementation is taken into consideration.

1. OBJETIVO

Desenvolve-se atualmente no LSI - USP estudos para o projeto de um supercomputador, representando um passo a mais na evolução da tecnologia de computadores no Brasil. Este trabalho representa parte dos estudos que visam definir a arquitetura da UCP deste supercomputador; devendo-se ainda desenvolver muitos estudos sobre essa arquitetura e outras, tanto alternativas quanto complementares.

O objetivo deste artigo é apresentar uma arquitetura de UCP para um computador multiprocessador, adequada para o processamento de estruturas do tipo matriz, com grandes aplicações nas áreas com intenso processamento numérico (processamento de imagens, solução de sistemas de equações, análise de elementos finitos, entre outras), e conveniente para implementação através de uma tecnologia de circuitos integrados com portas muito rápidas e escala de integração relativamente pequena, como é a atual tecnologia de GaAs.

2. INTRODUÇÃO

A performance de um computador de arquitetura paralela depende basicamente dos seguintes fatores:
1) A capacidade de processamento de cada um de seus processadores;
2) O grau de paralelismo intrínseco do programa sendo executado;
3) O "overhead" associado com o gerenciamento de múltiplos processadores;

O fator 1 pode ser atacado através de melhorias tecnológicas (fabricação de circuitos integrados mais rápidos e mais complexos) e de avanços na arquitetura de monoprocessadores (maior número de registradores, maior grau de paralelismo interno, etc.).

O fator 2 deve ser considerado muito profundamente, pois é fácil mostrar que a existência de pequenos trechos não sujeitos a paralelização em um programa compromete profundamente o desempenho geral do sistema.

Suponha que a execução de um certo programa seja composta por S instruções não sujeitas a paralelização e por P instruções sujeitas a

um grau de paralelismo arbitrário. Neste caso o tempo de execução de um programa num sistema composto por apenas um processador será :

$$(S + P) * t \qquad (1)$$

onde t é o tempo necessário para a execução de uma instrução.

Se esse mesmo programa for executado numa màquina paralela composta por N processadores iguais a do sistema uniprocessador, o tempo de execução do programa será :

$$(S + P/N) * t \qquad (2)$$

e o ganho no tempo de execução do sistema multiprocessador em relação ao sistema uniprocessador será dado por :

$$G = (S + P) / (S + P/N) \qquad (3)$$

Definindo agora Q como sendo a razão entre o numero de instruções não paralelizàveis e instruções paralelizàveis, ou seja :

$$Q = S/P,$$

obtem-se que o limite de G quando N tende a infinito é dado por :

$$\lim(N \to \inf)G = (S+P)/S = 1 + 1/Q \qquad (4)$$

O resultado acima demostra a importância de se reduzir ao màximo o valor de Q. Essa redução pode ser obtida através de linguagens de programação com alto grau de paralelismo intrinseco, na resolução de problemas também com alto grau de paralelismo intrinseco, num computador cuja arquitetura suporte de maneira eficiente a linguagem de programação.

Finalmente o fator 3 pode ser abordado através de arquiteturas de multiprocessadores que resultem num baixo "overhead". Dentre os pontos que causam esse "overhead", e que portanto devem ser evitados, pode-se citar :
1) os mecanismos de sincronização de processos concorrentes;
2) o compartilhamento de recursos, principalmente memòria.

3. A PROPOSTA DE UMA ARQUITETURA

3.1. Uma Linguagem de Programação Apropriada.

Tratamento de estruturas de dados do tipo matriz são problemas naturalmente sujeitos a paralelização, basta notar que a multiplicação de duas matrizes NxN pode ser executada por atè N3 processadores em paralelo. Uma arquitetura que suporte tratamento eficiente de matrizes è bastante importante. Jà foi ressaltada a importância de uma linguagem de programação para a especificação de um programa de modo que este possa ser executado de forma paralela. Uma linguagem de programação voltada ao processamento de matrizes tambèm deve ser utilizada para orientar a arquitetura do sistema multiprocessador.

A linguagem APL [1,2] foi projetada para trabalhar com matrizes, e jà foi usada como base da arquitetura de vàrios computadores [2] (o melhor exemplo è o ETA-10 [3]). Programas escritos em APL são sujeitos a boa paralelização pois contèm grande porcentagem de instruções intrinsecamente paralelas (inversão, multiplicação, redução e ordenação de matrizes entre muitas outras). Pode-se conseguir um grande aumento na velocidade de execução de um programa em APL simplesmente paralelizando a execução de cada instrução, sem ter que se preocupar com a paralelização entre instruções (obviamente tal técnica tambèm pode ser usada).

3.2. A Estrutura do Computador.

Como o objetivo è uma arquitetura que execute eficientemente operações com matrizes e tenha pouco "overhead" de controle, uma arquitetura do tipo SIMD (Single Instruction Multiple Data) [4] è bem apropriada para o sistema multiprocessador desejado, oferecendo as seguintes vantagens:
1) não hà necessidade de sincronização, pois hà apenas uma unidade de controle e portanto

uma única instrução em execução a cada instante;
2) ao invés de ter que implementar N processadores completos, com unidade de controle e fluxo de dados, pode-se implementar apenas 1 unidade de controle e N fluxo de dados;
3) analogamente ao item 2, em vez de ter que implementar N memórias de código e N memórias de dados basta 1 memória de código e N memórias de dados.

Arquiteturas do tipo MIMD (Multiple Instruction Multiple Data) [4] são mais genéricas que arquiteturas do tipo SIMD; contudo, como o objetivo é o processamento de matrizes, arquiteturas do tipo SIMD são suficientes, e potencialmente mais poderosas.

3.3. A Eletrônica do Computador.

Embora a não necessidade de sincronização entre processos seja uma característica de arquiteturas SIMD, a sincronização dos sinais entre os vários circuitos é uma característica do projeto eletrônico. A operação de todo o computador de forma síncrona (tanto a nível de instruções como a nível de ciclo de máquina) é uma característica que resulta num maior desempenho do computador, pois não se perde tempo sincronizando sinais vindos de diferentes módulos do sistema.

A operação de todo o sistema sob um único sinal de sincronismo ("clock"), exige cuidados no projeto, pois deve-se lembrar que a velocidade de propagação da informação num circuito impresso pode ser tão baixa quanto 10 cm/ns [5]. Tem-se então que, para o projeto de um sistema composto por grande número de processadores, deve-se fazer uma análise eletromagnética cuidadosa do circuito, calculando a propagação de cada sinal. Por outro lado, o projeto de um sistema com número reduzido de processadores pode ser feito de maneira bem mais simples.

Sistemas com menos processadores também estão menos sujeitos as influências da fórmula (4), bem como do limite de ganho de velocidade de N/lnN [4], dado por alguns autores (basicamente, sistemas com menos processadores apresentam um maior fator de utilização).

3.4. Uma Arquitetura Básica.

Obviamente um sistema com menos processadores deve incluir processadores bastante poderosos, o que nos leva a alternativa de implementação desses processadores em GaAs.

Circuitos de GaAs são intrinsicamente mais rápidos que circuitos de silício, mas apresentam como desvantagem uma menor escala de integração (e um custo maior também). Atualmente são disponíveis em GaAs circuitos "full custom" de aproximadamente 40000 transistores, alguns fabricantes oferecem "gate arrays" na faixa de 2000-3000 gates, e um fabricante dispõe de "gate arrays" de 6000 gates.

Uma das implementações mais conhecidas, e também uma das mais poderosas, de uma arquitetura SIMD com pequeno número de processadores é o Illiac IV [4,6,7], que pode ser tomado como referência.

A UCP do Illiac IV é composta de uma unidade de controle (UC), 64 elementos de processamento (EP), cada qual com sua memória local (MEP). Cada EP se comunica com seus 4 vizinhos (norte, sul, leste, oeste).

Cada EP do Illiac IV trabalha com palavras de 64 bits e é composto por cerca de 100000 dispositivos. A primeira providência a ser tomada para reduzir a complexidade da arquitetura proposta é adotar uma palavra de 32 bits.

A implementação de uma arquitetura como essa em GaAs sugere o uso de 1 CI para a implementação da UC (embora possam ser usados mais, pois a UC deve representar a menor parte do custo, o uso de 1 CI se justifica pela simplicidade) e 1 CI para cada um dos EPs. Já a memória local, em se mantendo o mesmo tamanho da memória local do Illiac IV, deveria ser de 2k palavras, ou (assumindo palavras de 32 bits) 8 kbytes.

Embora sejam disponíveis memórias SRAM em GaAs de 16 kbits, parece mais realista supor memórias de 4 kbits, pois há mais fabricantes (veja também o argumento a seguir), e o uso de 16 pastilhas para cada MEP.

Ao invés de implementar canais de comunicação entre cada EP, é mais eficiente e mais genérico (embora um pouco mais complicado) permitir que cada MEP seja acessada por seu

EP e cada um dos 4 EPs vizinhos a este. Uma opção para essa implementação seria usar CIs de memória (especialmente projetados) com múltiplos portos de acesso. Este quesito complica um pouco a pastilha de memória, o que é mais um motivo para se considerar apenas 4 kbits por CI de memória. Uma outra opção de implementação é usar um CI para memória e outro associado para implementar as várias portas de acesso, mas isso força um aumento de CIs ou uma redução da memória total. Como essa última opção pode ser implementada utilizando-se CIs de memória comercialmente disponíveis, e o CI que implementa as várias portas de acesso é bastante simples (veja item 3.7), adota-se ela como opção principal.

Deve-se notar que cada EP não tem uma unidade de controle. O espaço que seria ocupado pela unidade de controle pode ser usado para maior funcionalidade (principalmente suporte para ponto flutuante). Esse ganho de espaço da unidade de controle é muito mais valioso em GaAs do que em silício (MOS).

3.5. Adições à Arquitetura Básica.

Alguns pontos fracos da arquitetura do Illiac IV devem ser lembrados e corrigidos :

1) a não presença de um conjunto de registradores de propósito geral nos EPs. 16 registradores (usado no MIPS [5] e no Clipper [8]) parece ser um bom número; deve-se ter também um conjunto de instruções orientado a registrador (típico de processadores RISC) com instruções "load/store" para acesso a MEP;

2) a não presença de uma grande memória partilhada por todos os EPs. Uma memória de centenas de megabytes pode ser facilmente implementada com os chips DRAM atuais de 1Mbit. Essa memória partilhada se comunicaria diretamente com as MEPs através de um duto de comunicação externa que acessa todas elas;

3) um sistema de E/S pouco poderoso (em capacidade de armazenamento) em comparação com a UCP; o serviço de E/S pode ser realizado por um segundo computador especialmente projetado (ou pelo menos dedicado) para isto.

3.6. O Sistema de Interconexão.

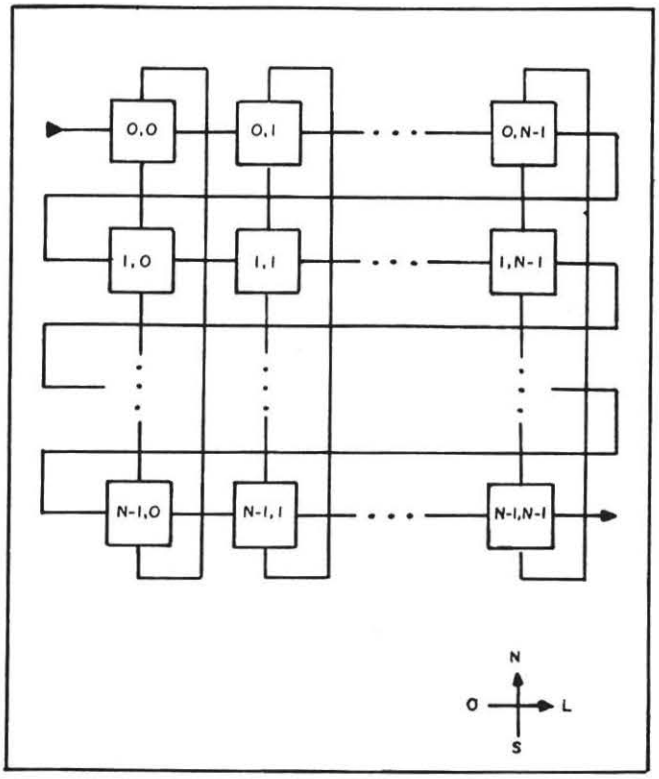


Figura 1. Sistema de interconexão entre NPs. Os diversos NPs do sistema estão conectados numa topologia de "mesh", como no Illiac IV. Todos os NPs são conectados a um duto de comunicação externa (não representado na figura).

Chamaremos de núcleo de processamento (NP) ao conjunto formado por um EP, sua MEP e o circuito para interconexão com outros NPs. Os diversos NPs do sistema são organizados numa matriz NxN e conectados entre si numa topologia de "mesh" (figura 1). Um NP é identificado por dois índices, i e j, que representam o número da linha e da coluna, respectivamente, do NP na matriz (0 <= i,j <= N-1). Um NP de índices i,j se conecta a 4 outros NP, por cada uma das suas saídas:

```
1)pelo norte: com (i-1) mod N,j;
2)pelo sul: com (i+1) mod N,j;
3)pelo oeste: se j=0
                então com (i-1) mod N,N-1
                senão com i,j-1;
4)pelo leste: se j=(N-1)
                então com (i+1) mod N,0
                senão com i,j+1.
```

Todos os NPs estão conectados a um duto para comunicação ao meio externo. Esse duto tem acesso direto as membrias de cada NP, e não necessariamente tem 32 bits de largura, um duto mais largo pode acessar vários NPs simultaneamente, apresentando uma maior banda de comunicação.

3.7. Núcleo de Processamento.

O núcleo de processamento (NP) é formado por um elemento de processamento (EP), sua memória (MEP) e o circuito para interconexão (CPI), conectados conforme figura 2.

Cada MEP é formado por 8 CIs de memória de 1kx4 bits (ou 4kx4 bits), numa configuração de 1kx32 palavras (ou 4kx32 palavras).

Associado a cada CI de memória há um CI de conexão (figura 3). Esse CI conecta o duto de dados da memória com um dentre outros seis dutos: processador local, duto de comunicação externa, e saídas norte, sul, leste e oeste. O CI de conexão tem 32 pinos de sinais e uma complexidade aproximada de 80 gates [9]. O CPI é formado pelo conjunto de todos os CIs de inteconexão.

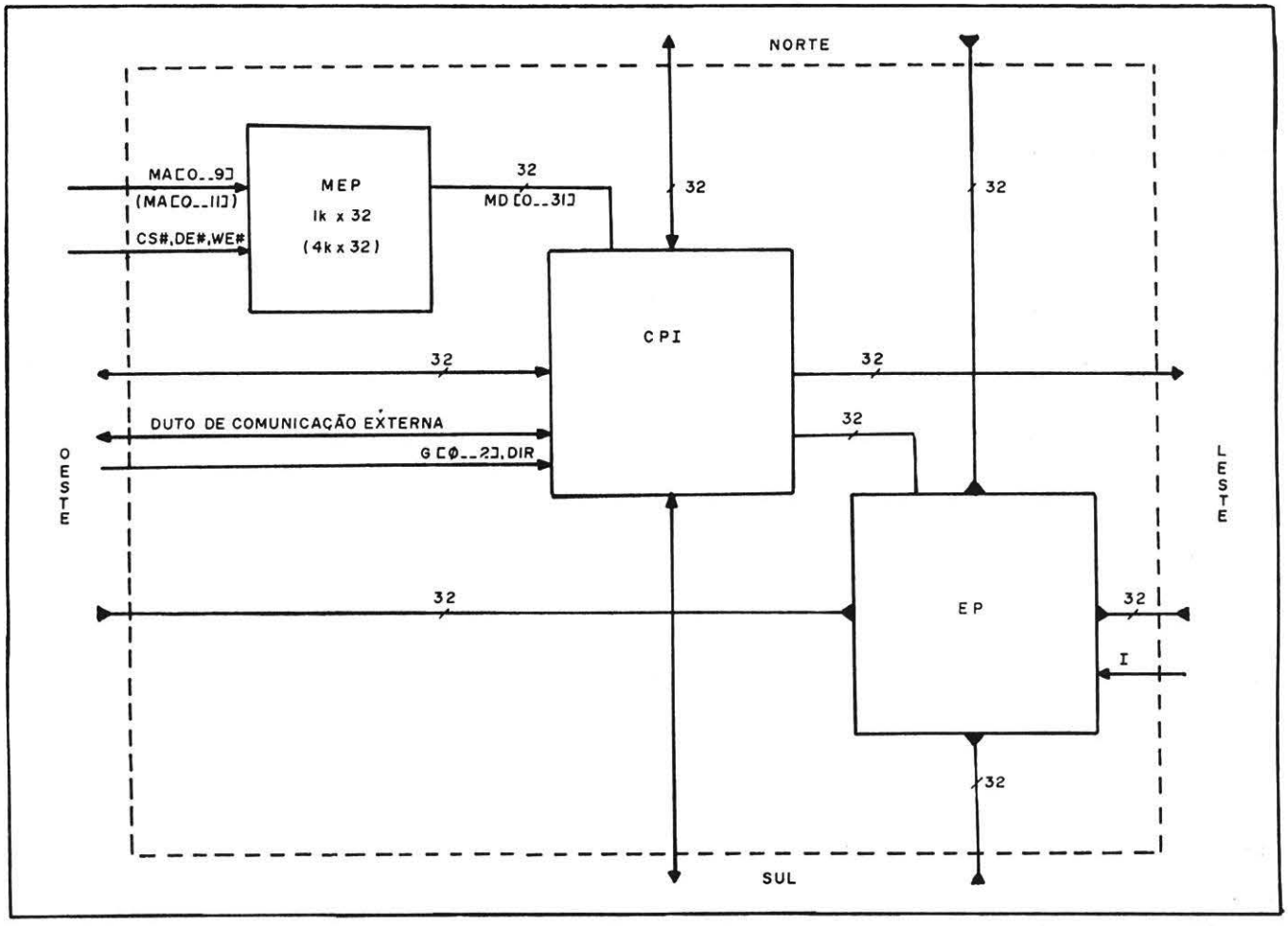


Figura 2. Estrutura de um NP. A MEP recebe os sinais de endereço e controle da UC. O CPI é controlado pelos sinais G[0..2] e DIR (provenientes também da UC) e faz a conexão do duto de dados da memória (MD[0..31]) com o EP, o duto de comunicação externa e as saídas norte, sul, leste e oeste. O EP se conecta a MEP local e a dos seus vizinhos pelas entrada norte, sul, leste e oeste; os sinais de controle para a EP vem da UC pelo duto I.

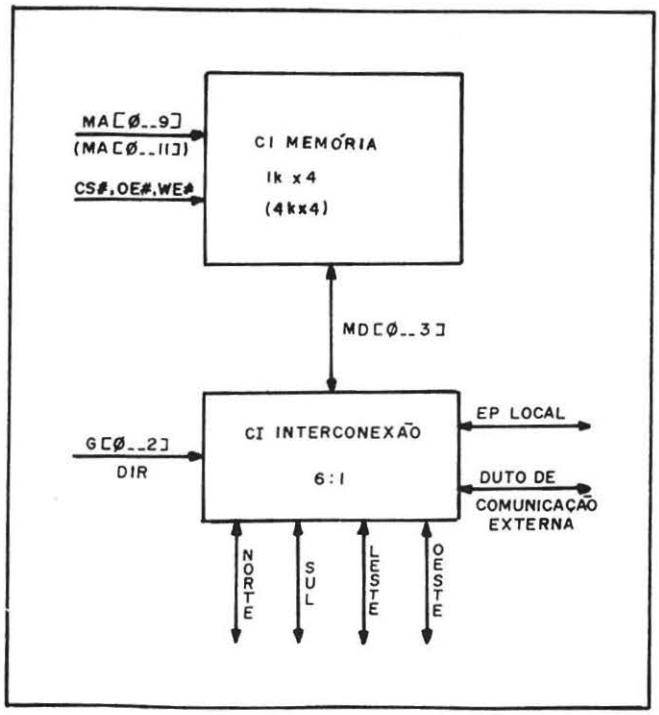


Figura 3. CI de membria e seu CI de interconexão associado. Um dos 6 dutos de dados (EP local, duto de comunicação externa, saídas norte, sul, leste e oeste) é conectado ao duto de membria (MD[0..3]), conforme o valor dos sinais G[0..2] (hà uma posição de desabilitação, também), e a direção da transferência de dados é dada pelo sinal DIR.

No duto de dados do EP são conectados o duto com a sua membria local (via CPI) e as entradas norte, sul, leste e oeste, que são as saídas das CPIs dos seus vizinhos correspondentes.

A arquitetura de um EP é mostrada na figura 4. O conjunto de registradores é capaz de realizar 2 leituras e 1 escrita simultaneamente. Entre os 2 dutos de leitura e o duto de escrita são colocadas as unidades funcionais.

O conjunto de 16 registradores de 32 bits apresenta um complexidade aproximada de 1500 gates [10]. Uma ULA capaz de realizar operações de soma e subtração, operações lógicas em campos de bits, deslocamentos, codificação de prioridade e passos de multiplicação e divisão, pode ser contruída com cerca de 5000 gates [10]. Um multiplicador de 16x16 bits necessita de 2000 gates [9], e um multiplicador 32x32 bits necessita de 8000 gates [9] aproximadamente.

Os registradores RA, RB e RW permitem que o EP trabalhe em "pipeline" de três estàgios: carga de operandos, execução de operação e escrita do resultado. Algum mecanismo de software (mais provavelmente) ou de hardware na UC evita conflitos de registradores.

O registrador de status do EP pode ser lido pela unidade de controle (UC) do sistema e também pode inibir a execução de instruções.

Utilizando-se um CI "full custom" seria possível implementar a ULA, o conjunto de registradores e o multiplicador de 16x16 bits num único CI. Para implementação num único "gate array" (6000 gates) só seria possível incluir o conjunto de registradores e uma versão ligeiramente simplificada da ULA.

A alternativa de implementação do EP em vàrios CIs possibilitaria a inclusão de um multiplicador de 32x32 bits e muitas outras unidades funcionais. Contudo, o projeto seria muito mais complexo e os CIs teriam um número muito elevado de pinos, pois cada unidade funcional teria 2 dutos de 32 bits entrando e 1 duto de 32 bits saindo.

3.8. Número de Processadores e Velocidade.

O número de NPs do computador depende muito da tecnologia de empacotamento. Uma tecnologia muito sofisticada, como usada na construção da família IBM 308X [11] permite uma densidade de até 1 CI/cm2, aproximadamente.

A seguir tem-se uma tabela com número total de CIs, dimensão (lado e semi-perímetro, supondo configuração quadrada e densidade de 1 CI/cm2), e tempo màximo de propagação de um sinal entre dois pontos (supondo o campo elètrico se propagando a 10cm/ns) de uma UCP conforme a arquitetura proposta, para diferentes números de NPs :

NP	total CIs (17xEP+1)	lado (cm)	semi-perímetro (cm)	tempo (ns)
4	69	8.3	16.6	1.66
8	137	11.7	23.4	2.34
16	273	16.5	33.0	3.30
32	545	23.3	46.6	4.66
64	1089	33.0	66.0	6.60

A coluna do tempo na tabela acima dà uma boa idèia da màxima freqência de operação da UCP. Um projeto cuidadoso serà necessàrio para operar a 200 MHz com 16 processadores e a 500 MHz com 4 processadores. Um sistema com 64 processadores poderia operar a 100 MHz.

3.9. A Unidade de Controle.

A UC contèm seu pròprio conjunto de registradores de uso geral e um registrador de habilitação, que contèm um bit associado com cada EP, habilitando ou não a sua operação.

A UC transmite, a cada ciclo, os bits de habilitação (1 para cada EP), os sinais de endereço e controle para a memòria (iguais para todas as MEPs), os sinais de seleção e de direção de interconexão (iguais para todos os CPIs) e os sinais de seleção de registradores e de operação para os elementos de processamento (iguais para todos os EPs).

A UC pode ler os registradores de status de cada EP, individualmente.

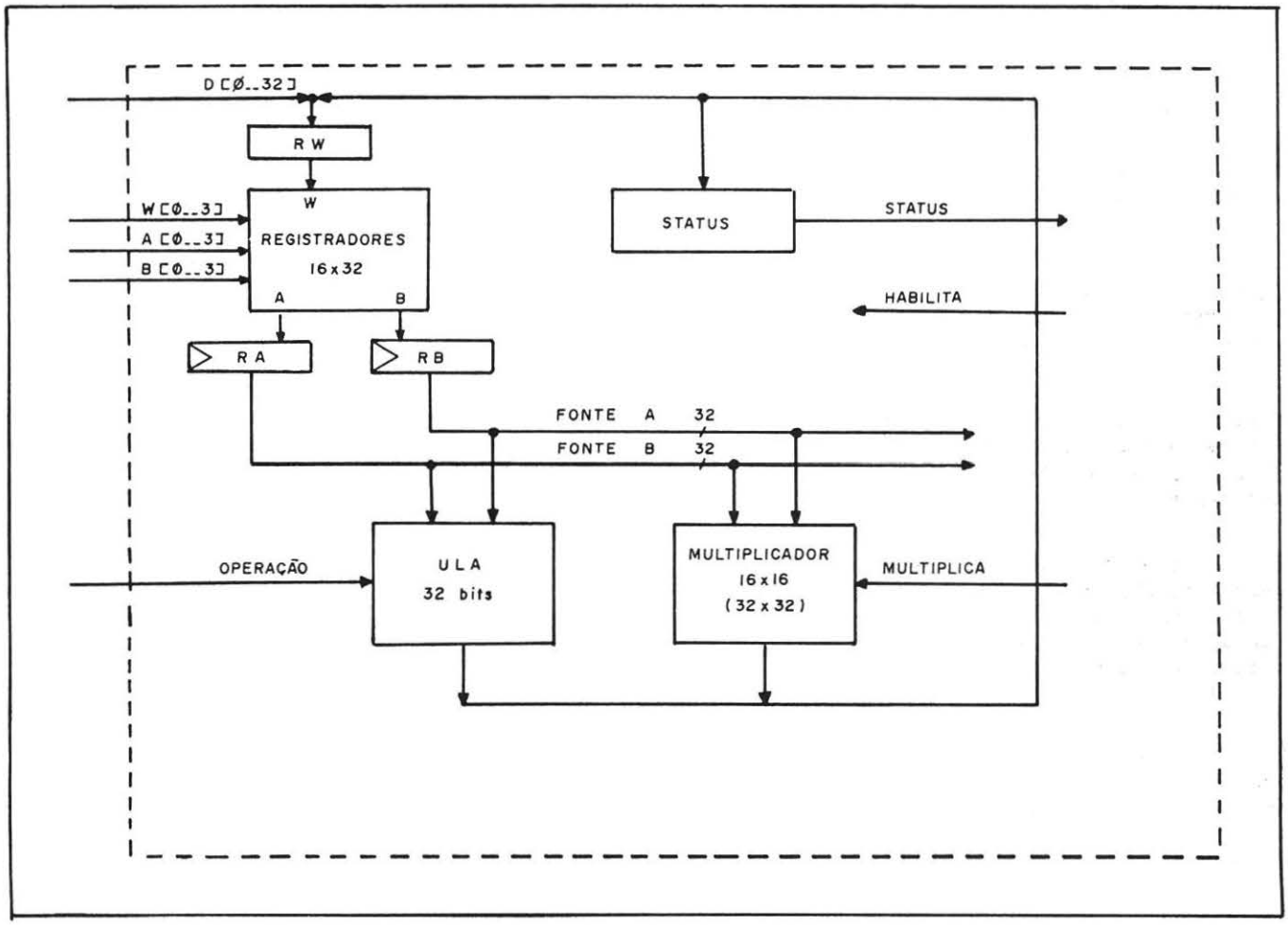


Figura 4. Organização bàsica de um EP. Os dados da MEP local e dos 4 vizinhos cheyam pelo duto D[0..31]; o registrador de status pode ser lido pela UC atravès do duto STATUS. Todos os outros sinais externos fazem parte do duto I, que controla o funcionamento do EP. Os sinais W[0..3] selecionam o registrador para escrita. Os sinais A[0..3] e B[0..3] selecionam os registradores para leitura nos portos A e B, respectivamente. HABILITA é um sinal de habilitação geral para o EP. MULTIPLICA habilita o multiplicador e OPERAÇAO è um conjunto de sinais que seleciona a operação realizada pela ULA.

A existência de uma única UC significa também a existência de apenas uma unidade de fornecimento de instruções, que pode ser cuidadosamente projetada e montada.

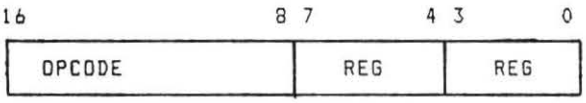
Praticamente, a única maneira de se conseguir suprir instruções a taxa de 200 MHz (ou 500 MHz) é estabelecendo-se fluxos contínuos de instruções, com alto grau de "pipeline" e portanto com grande prejuízo na quebra de sequências.

Programas em APL envolvem grande carga de processamento entre desvios, e portanto são apropriados a esse tipo de arquitetura.

3.10. O conjunto de instruções suportado.

Deve-se agora considerar o suporte para as operações básicas do APL.

As instruções para a UC devem ter formato fixo do tipo:



8 bits para o código de operação;
4 bits para o registrador origem e destino;
4 bits para o outro registrador origem ou constante.

Esse formato fixo e pequeno facilita a decodificação e a carga de instruções (2 instruções/palavra de 32 bits). Conjuntos de registradores com mais de 16 registradores podem ser suportados por um mecanismo de janelas (como no UCB RISC [5]).

Um mecanismo de continuação da instrução permite que constantes maiores que 4 bits sejam especificadas diretamente na instrução, para representação, por exemplo, de endereços de desvios.

As operações básicas do APL são (não em sintaxe de APL):
1) +, -, *, /, **mod**, **div**;
2) **and**, **or**, **not**;
3) =, <>, <, <=, >, >=;
4) **floor**, **ceil**;
5) rotação;
6) redução;
7) seleção;
8) ordenação.

As operações dos grupos 1, 2, 3, 4 e 7 são suportadas pelos NPs individualmente, podendo fazer uso de todo o paralelismo da máquina, desde que o tamanho da matriz seja maior que o número de EPs do computador; as operações do grupo 5 são suportadas pelas MEP, com a mesma utilização dos recursos da máquina que os grupos anteriores, já as operações dos gupos 6 e 8 são mais complicadas, não podendo ser resolvidas (em geral) em ciclos com total paralelismo e envolvendo muita comunicação.

Para apresentar uma aplicação faz-se uso de um "assembler" hipotético. As instruções usadas são:
MOV : movimentação de dados;
ADD : soma dois valores;
SUB : subtrai uma valor de outro;
CLR : limpa um registrador;
JMP : desvio incondicional;
CPT : compara dois valores, "trap" se iguais;

Os registradores da unidade de controle são representados por CURi (i=0..15) e os registradores dos EPs (de todos que estiverem habilitados) são representados por AURi (i=0..15).

Para movimentação de dados entre MEP e EP, o endereço da posição na MEP deve estar armazenado em algum registrador da UC, e um MOV, utlizando esse registrador como índice, é executado. Conforme o endereço contido no registrador, cada EP acessa sua MEP, ou a do vizinho norte, sul, leste ou oeste.

Como exemplo de aplicação tem-se a execução da operação +/x, que é uma redução de soma sobre o vetor x, em APL (soma de todos os elementos do vetor x). Supõe-se que o sistema é composto por 16 EPs (sem perda de generalidade) e que o vetor x tem 16xN elementos (nada impede ajustar convenientemente o tamanho do vetor, completando os elementos faltantes com 0, nesse caso).

O primeiro passo consiste em carregar, via o duto de comunicação externa, as MEPs, cada uma com N elementos do vetor (se N for maior que o número de palavras em cada MEP, a redução pode ser feita sobre um pedaço do vetor de cada vez), ocupando as posições 0 até N-1. Carregadas as MEPs, pode-se executar a primeira parte do algoritmo, em que cada EP calcula a soma dos N elementos que estão na sua MEP:

```
;todos os EPs habilitados
      MOV   CUR0,N        ;inicializa indexador
      CLR   AUR0          ;inicializa soma
L:    SUB   CUR0,1        ;atualiza indexador
      MOV   AUR1,(CUR0)   ;carrega elemento de x
      ADD   AUR0,AUR1     ;adiciona na soma
      CPT   CUR0,0        ;"trap" se fim
      JMP   L             ;próximo elemento
```

Executada essa primeira parte do algoritmo, em que se aproveitou todo o paralelismo da máquina, o registrador R0 de cada EP contém a soma de N elementos do vetor. Basta agora somar esses 16 valores. Essa soma pode ser feita em 4 etapas, na qual cada EP soma seu resultado com o do seu vizinho, e usa a soma para a próxima etapa:

```
;preparar os endereços dos locais de troca de
;dados:
      MOV   CUR0,LOCAL  ;endereço MEP local
      MOV   CUR2,SUL    ;endereço MEP sul
      MOV   CUR3,LESTE  ;endereço MEP leste
;calcula somas de 2:
      MOV   (CUR0),AUR0 ;armazena local
      MOV   AUR1,(CUR2) ;lê do sul
      ADD   AUR0,AUR1   ;soma
;calcula somas de 4:
      MOV   (CUR0),AUR0 ;armazena local
      MOV   AUR1,(CUR3) ;lê do leste
      ADD   AUR0,AUR1   ;soma
;transporta tudo uma linha para norte
      MOV   (CUR0),AUR0 ;armazena local
      MOV   AUR1,(CUR2) ;lê do sul
      MOV   (CUR0),AUR1 ;armazena local
;calcula somas de 8:
      MOV   AUR1,(CUR2) ;lê do sul
      ADD   AUR0,AUR1   ;soma
;transporta tudo uma coluna para oeste
      MOV   (CUR0),AUR0 ;armazena local
      MOV   AUR1,(CUR3) ;lê do leste
      MOV   (CUR0),AUR1 ;armazena local
;calcula o resultado final
      MOV   AUR1,(CUR3) ;lê do leste
      ADD   AUR0,AUR1   ;soma
```

Ao final dessa segunda parte, a soma de todos os elementos do vetor encontra-se no registrador R0 de todas as EPs. Nessa segunda parte o grau de paralelismo foi de apenas 8, 4, 2 e 1 para as etapas 1, 2, 3 e 4 respectivamente, e constituem um "overhead" que não aparece em sistemas monoprocessadores. Contudo, se o número de elementos do vetor x for grande, então a maior parte do processamento é representada pela primeira parte, e o fator de aceleração do sistema em relação a um monoprocessador é de praticamente 16 (ou seja, o número de processadores), para essa operação de redução.

Para uma comparação quantitativa considera-se que cada instrução é executada em um ciclo, e portanto o número de ciclos para realizar a operação num sistema com 16 EPs é: 2+Nx5+19; e para realizar a operação num monoprocessador é: 2+16xNx5; portanto o fator de aceleração S é dado por:

$$S = (2+80xN)/(21+5xN)$$

Tabelando para alguns valores de N, obtem-se valores bastante satisfatórios já para um vetor de 80 elementos:

N	S
5	8.7
10	11.3
20	13.2
50	14.8
100	15.6

4. CONCLUSÃO

Mostra-se, neste trabalho, como é possível implementar um computador apropriado para tratamento de matrizes e com suporte eficiente para uma linguagem do tipo APL, utilizando uma arquitetura bem conhecida como a do Illiav IV e usando-se tecnologias mais modernas, como CIs de GaAs e memórias DRAM de alta capacidade para obter uma máquina poderosa, pequena e com baixos custos de projeto e fabricação.

AGRADECIMENTOS

Este estudo contou com apoio financeiro da FINEP e do CNPq.

REFERÊNCIAS

[1] Iverson, K. E., A Programming Language. John Wiley and Sons, Inc. New York NY, 1962.

[2] Tucker, A. B., Programming Languages. McGraw-Hill Book Company. New York NY, 1986.

[3] ETA Systems, ETA10 System Reference Manual. St. Paul MN, 1987.

[4] Hwang, K. and Briggs, F. A., Computer Architecture and Parallel Processing. McGraw-Hill Book Company. New York NY, 1987.

[5] Milutinovic, V. M. (editor), Computer Architecture, Concepts and Systems. North-Holland. New York NY, 1988.

[6] Siewiorek, D. P., Bell, C. G. and Newell, A., Computer Structures: Principles and Examples. McGraw-Hill Book Company. New York NY, 1983.

[7] Hayes, J. P., Computer Architecture and Organization. McGraw-Hill Book Company. New York NY, 1982.

[8] Fairchild, Clipper 32-bit Microprocessor User's Manual. Prentice-Hall, Inc. Englewood Cliffs NJ, 1987.

[9] AMD, Bipolar Microprocessor Logic and Interface Data Book. Sunnyvale CA, 1985

[10] AMD, Am29C300/29300 Data Book. Sunnyvale CA, 1988.

[11] Blodgett Jr., A. J., Microelectronic Packaging. Scientific American, Julho 1983, pp 76-86. New York NY.