ACELERADOR MODULAR PARA ROTEAMENTO

J.S.AUDE, E.B.M.O.PAIVA, E.P.L.AUDE, E.P.L.FILHO, M.F.MARTINS, S.B.PINTO
Núcleo de Computação Eletrônica da UFRJ
Cidade Universitária - Ilha do Fundão
Caixa Postal 2324, 20001 - Rio de Janeiro, RJ

RESUMO

Este artigo descreve uma arquitetura modular dedicada à implementação do algoritmo proposto por Lee para roteamento automático de conexões elétricas em circuitos eletrônicos. O algoritmo de Lee é descrito brevemente e as arquiteturas já propostas para sua implementação em hardware são discutidas. A arquitetura proposta no artigo é baseada em duas estruturas pipeline e num esquema especial de organização da memória que armazena informação sobre a superfície a ser roteada. As modificações do algoritmo de Lee implementadas na arquitetura são discutidas e o funcionamento do hardware é descrito, mostrando-se os algoritmos implementados para execução das diferentes fases do algoritmo de Lee. A arquitetura encontra-se em fase final de detalhamento e resultados de simulação indicam que seu desempenho na execução do algoritmo de Lee será pelo menos 40 vezes melhor do que o de mainframes.

ABSTRACT

This paper describes a modular hardware router which implements Lee's algorithm. This algorithm is briefly described and the architectures which have already been proposed for its implementation are discussed. The architecture which is proposed in this paper is based on two pipeline structures and on a special organization of the memory which stores information on the layout surface. Both the modifications which have been introduced in Lee's algorithm in the architecture design and the algorithms which are implemented by the hardware for the execution of the different phases of Lee's algorithm are described. At the moment, a detailed design of the architecture is being carried out. Simulation results have shown that the architecture performance will be at least 40 times better than that presented by mainframes in the execution of Lee's algorithm.

Agradecimentos: Os autores agradecem ao CNPq o auxílio dado para a execução do projeto descrito neste artigo.

1. INTRODUÇÃO

Devido à crescente complexidade dos circuitos eletrônicos que vem sendo projetados em único circuito integrado, a demanda por maior eficiência no processamento das ferramentas de CAD tem aumentado substancialmente nos últimos anos. Em consequência, diversas arquiteturas dedicadas tem sido propostas com o objetivo de ser obter um melhor desempenho na execução de algoritmos de CAD. A maioria destas arquiteturas é baseada na exploração do potencial paralelismo existente nestes algoritmos. As ferramentas de CAD que tem sido mais frequentemente analisadas para implementação em hardware são os simuladores [PFIS86], os verificadores de regras geométricas [SEIL82], os alocadores [KRAV86] e os roteadores [RYAN87].

O objetivo do trabalho em desenvolvimento no Núcleo de Computação Eletrônica da UFRJ (NCE/UFRJ) é a construção de uma máquina dedicada capaz de processar eficientemente o algoritmo de roteamento proposto por Lee [LEE61]. A tarefa de um roteador consiste em encontrar e traçar caminhos que interliguem os terminais de componentes que pertençam a um mesmo sinal elétrico. Estes caminhos podem ser traçados em uma ou mais camadas de materiais disponíveis para roteamento dos sinais. O algoritmo de Lee é um dos algoritmos mais utilizados na implementação de roteadores. Este algoritmo garante encontrar um caminho entre dois pontos se tal caminho existe e garante que o caminho encontrado é de tamanho mínimo.

No algoritmo de Lee, a superfície a ser roteada é representada por uma matriz de células quadradas. Em princípio, apenas segmentos horizontais ou verticais são geradas pelo algoritmo no traçado dos caminhos. Para cada conexão a ser feita, a execução do algoritmo de Lee passa por três etapas. A primeira delas, a "fase de expansão", é responsável por descobrir se há um caminho interligando os dois pontos. O procedimento utilizado nesta fase consiste na definição, a cada iteração, de uma nova fronteira de células a partir da expansão das células da fronteira atual nas direções norte, sul, leste e oeste. Cada célula da nova fronteira é marcada com uma "seta" que aponta na direção da célula cuja expansão originou sua marcação. Inicialmente, a fronteira de células é composta de uma única célula, a célula que corresponde ao ponto de origem do caminho a ser traçado. Na iteração "n" da fase de expansão, uma nova fronteira com no

máximo 4n células é formada. A fase de expansão termina quando a célula que representa o ponto destino é atingida e, portanto, um caminho foi encontrado, ou quando não há mais células na fronteira para serem expandidas, significando que não existe caminho entre os dois pontos. Para um caminho de comprimento "ℓ", expresso em número de células atravessadas, o tempo gasto na fase de expansão é proporcional a ℓ**2.

A segunda fase do algoritmo de Lee é a "fase de retraço". Seu objetivo é traçar o caminho entre os dois pontos encontrado na fase de expansão. O procedimento adotado consiste em, simplesmente, seguir as setas marcadas, partindo da célula que representa o ponto destino até atingir a célula que representa o ponto de origem do caminho. O tempo gasto na execução desta fase é porporcional ao comprimento "ℓ" do caminho.

A terceira fase do algoritmo é a "fase de reinicialização". Seu papel é preparar a matriz de células que representa a superfície de roteamento para a execução da fase de expansão da próxima conexão. Basicamente, nesta fase, as células utilizadas no caminho definido pelo retraço são transformadas em células de bloqueio para as próximas conexões e as setas marcadas nas células não utilizadas são apagadas. O tempo gasto nesta fase é proporcional a p*q, onde "p" e "q" são as dimensões da matriz de células que representa a superfície a ser roteada.

A Seção 2 deste artigo discute algumas das arquiteturas já propostas para implementação do algoritmo de Lee. Na Seção 3, as modificações no algoritmo básico de Lee implementadas na arquitetura proposta neste trabalho são discutidas. Ainda nesta seção é feita uma descrição da arquitetura do acelerador. A Seção 4 discute o funcionamento do acelerador em maiores detalhes. Finalmente, a Seção 5 descreve o estágio atual do projeto, apresentando uma estimativa de desempenho da arquitetura em função de resultados obtidos com um simulador do acelerador.

2. ARQUITETURAS PROPOSTAS PARA IMPLEMENTAÇÃO DO ALGORITMO DE LEE

Como as fases de expansão e de reinicialização do algoritmo de Lee são as que mais consomem tempo de máquina, a maioria das arquiteturas propostas para implementação em hardware do algoritmo de Lee visam tornar mais eficiente o processamento destas duas fases.

A máquina "L" de Breuer e Shamsa [BREU81] foi a primeira arquitetura proposta para implementação em hardware do algoritmo de Lee. A máquina "L" é uma arquitetura matricial, onde cada processador é associado a uma única célula da matriz que descreve a superfície a ser roteada. Com esta estrutura, na fase de expansão do algoritmo de Lee, a expansão das células de cada fronteira pode ser realizada em paralelo. Este fato faz com que o tempo gasto na expansão se torne proporcional a "ℓ" e não mais a "ℓ**2" para um caminho de tamanho "ℓ". O uso de uma arquitetura matricial permite obviamente que a fase de reinicialização seja feita em paralelo para todas as células em um único passo. Infelizmente, a proposta da máquina "L" tem sua aplicação limitada na prática a problemas de roteamento em que a superfície pode ser descrita por uma matriz de células de pequenas dimensões.

Visando contornar o problema apresentado pela máquina "L", Blank, Stefik e Van Cleemput [BLAN81], propuseram a máquina SAM("Synchronous Active Memory"). A arquitetura desta máquina é também matricial, porém a tecnica de "folding" é utilizada para permitir que a cada processador sejam associadas várias células da matriz que descreve a superfície a ser roteada. Com isso, reduz-se a quantidade de circuito necessária para implementação da máquina, porém perde-se flexibilidade na exploração de paralelismo nas fases de expansão e de reinicialização do algoritmo. Um esquema semelhante foi também utilizado na máquina WRM [HONG81] em que aos elementos de uma matriz 8x8 de microprocessadores são associadas, por um mecanismo de "folding", todas as células que compõem a matriz que representa a superfície a ser roteada.

Na máquina IAP proposta por Iosupovici [IOSU86], uma estrutura SIMD também é utilizada. Nesta estrutura, a cada processador está associada uma única célula da matriz. A implementação da máquina prevê a utilização de uma matriz de chips. Cada chip contém matrizes nxn de processadores, onde n=2**k-1. A comunicação entre chips vizinhos se dá através de "k" fios bidirecionais, que permitem o acesso sequencial aos processadores situados nas bordas vizinhas dos chips. Iosupovici mostra que esta comunicação sequencial entre os chips, necessária para reduzir o número de fios entre eles, não compromete significativamente o desempenho da máquina, ou seja, o tempo gasto na fase de expansão continua sendo proporcional a "ℓ" para um caminho de comprimento "ℓ".

Na máquina ISMA [RYAN87], foi utilizada uma idéia bastante original para se implementar o algoritmo de Lee com uma arquitetura matricial envolvendo um número não muito grande de processadores. A proposta nesta arquitetura é executar o algoritmo de Lee em dois passos. No primeiro passo, considera-se a matriz que representa a superfície a ser roteada dividida em submatrizes. Cada submatriz é considerada como uma única célula que pode estar bloqueada ou não dependendo do grau de ocupação das suas células. A cada submatriz é associado um processador da máquina ISMA. Como resultado deste primeiro passo, um roteamento aproximado é definido, determinando-se as submatrizes que devem ser atravessadas pelo caminho a ser traçado. No segundo passo, o roteamento detalhado em cada uma das submatrizes utilizadas no primeiro passo é realizado em sequência. A matriz de processadores deve ter dimensões iguais às das submatrizes. A implementação do algoritmo de Lee na máquina ISMA na forma descrita não garante encontrar o menor caminho entre dois pontos.

Uma arquitetura diferente para implementação do algoritmo de Lee em hardware foi apresentada por Rutenbar et al. [RUTE84]. Nesta arquitetu

ra, matrizes 3x3 de processadores dispostas em uma estrutura de pipeline são responsáveis pela realização em paralelo da expansão de diferentes fronteiras. O número de matrizes no pipeline determina o número de fronteiras que podem ser expandidas em paralelo. O pipeline é alimentado através de uma operação de varredura sequencial das células da matriz que representa a superfície disponível para layout. Para a fase de reinicialização um único estágio do pipeline é utilizado. Esta arquitetura reduz consideravelmente o número de circuitos utilizados para implementação da máquina, porém tem uma capacidade bem mais limitada de exploração do paralelismo existente no algoritmo de Lee.

Won, Sahni e El-Ziq [WON87] propuseram a implementação em hardware do algoritmo de Lee usando uma sequência de 3 processadores em pipeline para execução da fase de expansão. Nesta arquitetura a marcação das células vizinhas de uma dada célula é feita simultaneamente e, em paralelo, com a leitura de uma nova célula da fronteira atual a ser expandida e com a adição de novas células à próxima fronteira a ser expandida. Na máquina proposta por Won et al., apenas uma camada de roteamento é considerada e cada estágio do pipeline tem seu processamento definido por rotinas em software. A arquitetura proposta neste trabalho é uma extensão da arquitetura proposta por Won et al.

3. ARQUITETURA DO ACELERADOR

O objetivo do projeto do NCE/UFRJ é implementar uma arquitetura para execução do algoritmo de Lee que seja realizável na prática com circuitos integrados SSI, MSI e LSI disponíveis no mercado e que possa funcionar como um dispositivo acelerador conectável aos microcomputadores e supermicrocomputadores nacionais. Com isso pretende-se viabilizar o desenvolvimento no país de estações de trabalho para roteamento poderosas e de baixo custo.

As arquiteturas SIMD, embora permitam a exploração ao máximo do paralelismo existente no algoritmo de Lee, demandam um grande número de circuitos e, consequentemente, requerem o uso de técnicas de integração em larga escala para sua implementação. Um outro ponto a observar é que, para implementação do algoritmo de Lee, o grau de utilização dos processadores existentes em uma arquitetura SIMD é, em geral, muito baixo. Devido a estes fatores este tipo de arquitetura mostrou-se inadequado aos objetivos do projeto.

A arquitetura a ser proposta é uma extensão da máquina desenvolvida por Won, Sahni e El-Ziq [WON87], com características modulares. O módulo básico do acelerador tem capacidade para operar com áreas de roteamento de até duas camadas e dobra o paralelismo explorado pela máquina de Won sem subutilizar o hardware adicional necessário. Esta melhoria de desempenho é alcançada com a incorporação na arquitetura da estrutura de "pipeline" utilizada no processador RPS [RUTEN84]. Um melhor desempenho também é obtido porque os estágios das estruturas de pipeline são implementados diretamente em hardware e não em software como na máquina de Won et al. Através da adição de outros módulos básicos, o acelerador se torna capaz de atacar problemas em superfícies com múltiplas camadas de roteamento e representadas por matrizes de grandes dimensões. Além disso, o acelerador passa a poder rotear em paralelo conexões confinadas a certas regiões da superfície global de roteamento.

Três alterações do algoritmo básico de Lee foram implementadas nesta arquitetura. A Seção 3.1 descreve estas alterações e a Seção 3.2 descreve a arquitetura propriamente dita.

3.1. Alterações no Algoritmo Básico de Lee.

A primeira alteração introduzida no algoritmo básico de Lee diz respeito à conexão de sinais com mais de dois pontos. Para implementação desta facilidade, a fase de reinicialização deve marcar como células destino, ao invés de células de bloqueio, as células utilizadas pertencentes a um mesmo sinal. Desta forma, para cada conexão adicional de um mesmo sinal, uma nova célula origem é definida e todas as células já utilizadas nas conexões anteriores deste mesmo sinal são consideradas células destino. Com este mecanismo, ligações em "T" podem ser produzidas pelo roteador. Quando isto é indesejável, ligações em "cadeia" são realizadas e, neste caso, a fase de reinicialização define que todas as células, com exceção da célula de origem, pertencentes à conexão anterior de um sinal são células de bloqueio para as próximas conexões. A célula de origem é definida como célula destino para a próxima conexão do sinal. Uma exceção ocorre na primeira ligação, já que a célula destino desta ligação deve permanecer como célula destino para a ligação seguinte.

A segunda alteração implementada visa habilitar o algoritmo a trabalhar com mais de uma camada para roteamento. Basicamente, a alteração introduzida no algoritmo consiste em realizar a fase de expansão considerando que a matriz de células é uma matriz tridimensional. Ao se realizar a expansão para uma célula em um dado plano, as células vizinhas situadas em planos inferiores ou superiores são também marcadas. Para viabilizar a implementação desta modificação, duas novas setas, além das quatro já existentes, são utilizadas: seta apontando para cima e seta apontando para baixo. Comumente, em problemas de roteamento em múltiplas camadas, a cada camada é associada uma direção preferencial de roteamento, horizontal ou vertical, que determina a ordem de precedência na marcação de setas em cada camada. Expansões que impliquem em mudança de camada só prevalecem quando não é possível expandir na direção preferencial da camada corrente e existe possibilidade de expansão na direção preferencial da camada para onde se está passando.

A terceira e última alteração introduzida no algoritmo básico de Lee visa melhorar o desempenho do algoritmo, evitando que cada conexão roteada crie bloqueios que certamente dificultarão o roteamento de conexões subsequentes. O mecanismo utilizado para isso foi a introdução de

custo diferenciado para as células que compõem a matriz que representa a superfície a ser roteada. Basicamente, a idéia consiste em atribuir custo mais elevado para células vizinhas dos pinos a serem interconectados, custo médio para células sem nenhuma célula vizinha ocupada e custo mais baixo para células vizinhas de células já utilizadas por conexões anteriores. Com este esquema, busca-se evitar o bloqueio em posições adjacentes aos pinos dos componentes e dos conectores e busca-se, sempre que possível, "empilhar" conexões traçadas próximas umas das outras. Para implementação deste mecanismo de custo, tanto a fase de expansão quanto a fase de reinicialização tiveram de ser modificadas. Na fase de expansão, células na fronteira a ser expandida que possuam custo diferente de 1 não sofrem expansão. O custo delas é subtraído de 1 e, com este novo custo, elas passam a fazer parte da fronteira a ser expandida na próxima iteração do algoritmo. Com isso, o algoritmo de Lee deixa de gerar o caminho de comprimento mínimo e passa a gerar o caminho de custo mínimo. É interessante notar que o esquema de custo proposto tem um caráter dinâmico: o custo das células é redefinido à medida que o roteamento prossegue. Portanto, ao se completar uma ligação, o algoritmo deve recalcular o custo de cada célula em função dos critérios acima estabelecidos. Apenas três valores de custo estão sendo utilizados: 1, 2 e 3. A utilização ou não de custo diferenciado para as células é uma opção do usuário.

3.2. Descrição da Arquitetura.

O aspecto fundamental do projeto do acelerador de roteamento é o uso de um esquema especial de banqueamento para a organização da "memória de placa", a memória que armazena as células que compõem a matriz de representação da superfície a ser roteada. Na estrutura de banqueamento utilizada há 8 bancos distintos. Dois grupos podem ser definidos: o grupo I, contendo os bancos 0, 1, 2 e 3 e o grupo II, contendo os bancos 4, 5, 6 e 7. As células pertencentes aos bancos do grupo I tem todas as suas células vizinhas em bancos do grupo II e vice-versa. Duas vantagens importantes resultam desta estrutura de banqueamento. Em primeiro lugar, a expansão de cada célula pode ser feita em paralelo, já que as células vizinhas a serem marcadas estão em bancos diferentes que podem ser acessados concorrentemente. Em segundo lugar, durante a fase de expansão do algoritmo, pode-se sempre expandir em paralelo células pertencentes a duas fronteiras subsequentes, já que uma delas causará expansão para células do grupo I e a outra fará expansão para células do grupo II.

Visando explorar as vantagens apontadas acima, duas sequências de processadores em "pipeline" são usadas na implementação do acelerador para a fase de expansão do algoritmo. Cada sequência processa a expansão das células pertencentes a uma de duas fronteiras subsequentes. Estas sequências de processadores em "pipeline" são compostas de 3 estágios. O primeiro estágio lê a próxima célula a ser expandida da memória que armazena a fronteira em expansão. O segundo estágio realiza a expansão propriamente dita, acessando em paralelo os bancos necessários da memória de placa. Finalmente, o terceiro estágio acrescenta à memória que armazena a fronteira a ser expandida pela outra sequência de processadores as células marcadas pelo segundo estágio.

A fim de permitir que a operação realizada pelo terceiro estágio possa ser feita de forma concorrente para as células marcadas na fase de expansão, a informação é armazenada na "memória de fronteira" de forma codificada. Esta codificação indica as coordenadas da célula que deu origem à expansão e identifica as células vizinhas que devem fazer parte da próxima fronteira. Na verdade, esta memória está organizada, basicamente, em 2 conjuntos de 2 bancos. Num dos bancos, são armazenadas as células da fronteira pertencentes aos bancos 0, 1, 2 e 3 da memória de placa e no outro banco as células pertencentes aos bancos 4, 5, 6 e 7. A cada instante de tempo, uma sequência de processadores lê de um banco de um conjunto e a outra sequência de processadores escreve no mesmo banco do outro conjunto. Procedimento análogo ocorre com relação ao outro banco.

O módulo básico do acelerador compreende estas duas sequências de processadores em "pipeline" e é capaz de operar em até duas camadas de roteamento. Para que isto seja possível o número de bancos de memória de placa deve ser duplicado. Para atualização da memória de fronteira, optou-se por economia de memória e realização das operações de atualização no estágio 3 do pipeline em dois passos: em cada passo as células de uma única camada são adicionadas ao conteúdo da memória de fronteira. A organização interna do módulo básico do acelerador é mostrada esquematicamente na Figura 1.

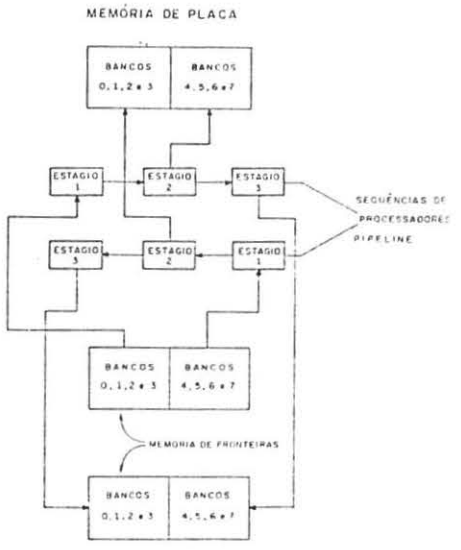

Figura 1: Arquitetura do Módulo Básico

Na memória de placa, 16 bits são utilizados para o armazenamento de cada célula. Três bits definem o status da célula, isto é, se a célula é um bloqueio, se a célula representa um ponto destino, se a célula representa um ponto de um barramento de alimentação, etc. Quatro outros bits, denominados bits de "bússola", são utilizados para caracterizar o tipo de desenho que deve ser associado àquela célula quando da impressão dos resultados. Dois bits definem o custo da célula. Três outros bits definem, na fase de expansão, a "senha" que deve ser associada àquela célula. A senha é necessária para indicar se, numa da iteração da fase de expansão, uma célula pode ser marcada ou não. A célula só pode ser marcada se sua senha indicar que ela ainda não foi marcada (senha=0) ou que sua marcação ocorreu nesta mesma iteração. Dos quatro bits restantes, três são utilizados para representar a seta e o outro, o bit de "via", para indicar se é permitido haver furo de passagem para outra camada naquela célula.

O módulo básico do acelerador é projetado para armazenar uma matriz de células dc dimensões 256 x 256 x 2. Consequentemente, como a memória de placa está dividida em 8 bancos por camada, cada banco é implementado como uma memória de 8K palavras de 16 bits.

Cada palavra da memória de fronteira deve armazenar as coordenadas x, y e z da célula expandida, os valores de custo dos seus quatro vizinhos e a senha a ser usada na expansão destes vizinhos. Valores de custo iguais a zero indicam que uma dada célula vizinha não pertence à nova fronteira. Consequentemente, 28 bits são necessários por palavra. O tamanho máximo de uma fronteira em cada camada de um módulo básico é inferior a 1K. Portanto, a memória de fronteira de cada módulo básico é implementada por 2 grupos de 2 memórias 2K x 28.

Não é apenas na fase de expansão que o acelerador pode produzir ganho de velocidade através da exploração de paralelismo. Na fase de reinicialização, os oito bancos da memória de placa são varridos concorrentemente. Neste processo de varredura o valor de senha armazenado em cada célula é zerado. Com este esquema, 16K ciclos (8K ciclos de leitura e 8K ciclos de escrita) de acesso à memória são realizados na fase reinicialização. Este número se mantém constante mesmo que diversos módulos básicos estejam sendo utilizados, já que os módulos básicos operam em paralelo.

A fase de retraço introduz um aumento de tempo muito pequeno no algoritmo de Lee e dá pouca margem à exploração de paralelismo. Consequentemente, na arquitetura em desenvolvimento, esta fase é implementada sob controle de um microprocessador. Com isso, maior flexibilidade é obtida na programação desta fase que é responsável não só por redefinir os campos de status e bússola das células utilizadas no caminho traçado bem como por atualizar os campos de custo e via das células vizinhas ao caminho. Tal flexibilidade de programação é desejável principalmente porque ela permite que diferentes políticas de atribuição de custo às células sejam utilizadas sem modificação do hardware.

Como já foi dito, os diferentes módulos do acelerador trabalham de forma concorrente na fase de reinicialização. Na fase de expansão, o trabalho concorrente dos módulos é também possível. A condição para que isso ocorra é que cada módulo esteja lidando com ligações contidas na região de 256x256 células por ele coberta. Quando um dos módulos detecta que a ligação que ele deve realizar envolve o uso de regiões diferentes da sua, este módulo solicita o controle global da máquina. Uma vez assumido o controle global, os processadores do módulo passam a ter acesso às memórias de placa residentes nos demais módulos. A expansão é realizada com o trabalho cooperativo de todos os módulos, porém, em princípio, apenas um módulo está ativo a cada instante de tempo. O módulo ativo no momento é aquele que detém o controle global da máquina e que executa a fase de expansão. Quando o módulo ativo encerra a expansão de uma fronteira, ele passa o comando global da máquina para um outro módulo que possua células a serem expandidas dentro da mesma fronteira.

4. FUNCIONAMENTO DO ACELERADOR

Para um melhor entendimento do funcionamento do acelerador, é importante detalhar os algoritmos implementados, em hardware, para as fases de expansão e reinicialização e, em software, para a fase de retraço.

Conforme já foi mencionado, a fase de expansão é processada em paralelo por duas estruturas de pipeline com três estágios: leitura da memória de fronteira, expansao propriamente dita e escrita na memória de fronteira. O algoritmo executado pelo primeiro estágio é basicamente o seguinte:

```
Procedimento LêMemóriadeFronteira;
   Se endereço de leitura >= limite do conjunto de leitura atual
   Então Define conjunto de leitura atual como
   conjunto de escrita e o conjunto de escrita
   atual como conjunto de leitura;
   Senão Lê informação do conjunto atual de
   leitura;
      Se há vizinho com custo = 1
      Então Gera endereço [x,y,z] deste vizinho para uso do segundo estágio;
         Zera custo deste vizinho;
         Se há outro vizinho com custo = 1
         Então Armazena informação atualizada
         de volta;
         Senão Se há vizinho com custo <> 0
            Então Autalizacusto;
            Senão Incrementa endereço de leitura;
      Senão Atualizacusto;
Fim;

Procedimento Atualizacusto;
   Decrementa 1 de todos os valores de custo
   diferentes de zero;
   Incrementa o valor da senha;
```

```
   Se valor da senha > 7 Então senha := 1;
   Armazena informação atualizada no grupo de
   escrita;
   Incrementa endereço de leitura;
Fim;
```

Dois aspectos importantes no procedimento descrito devem ser destacados. O primeiro deles é que, no máximo, dois acessos à memória de fronteira são feitos na execução do procedimento. Destes acessos, um deles é de leitura e o outro é de escrita. Apenas um acesso de leitura é realizado quando a informação lida contém apenas um vizinho com custo unitário e o custo dos demais vizinhos é zero. O segundo aspecto que deve ser destacado é que a execução do procedimento Atualizacusto pode implicar em conflito no acesso ao conjunto de escrita da memória de fronteira. Este conflito surge porque o estágio 3 da estrutura de pipeline pode estar simultaneamente requerendo uma operação de escrita neste conjunto para incluir células na nova fronteira a ser expandida.

O algoritmo executado pelo segundo estágio da estrutura de pipeline, responsável pela expansão da célula lida da memória de fronteira pelo primeiro estágio, é o seguinte:

```
Procedimento de Expansão;
   Se foi lida Célula pelo primeiro estágio
   Então Calcula endereço de todos os vizinhos;
      Para cada vizinho faça, em paralelo:
         Se vizinho dentro dos limites do módu
         lo
         Então Lê vizinho na memória de placa
            Se status[vizinho] <> bloqueio e
               (senha[vizinho] = senha atual ou
               0) e seta vizinho tem prioridade
                  de menor que
                        nova seta e
               não há mudança de camada ou pode
               haver via)
            Então seta[vizinho] := nova seta;
               senha[vizinho] := nova senha;
               Atualiza vizinho na memória de
               placa;
               Guarda custo[vizinho] para uso
                  do terceiro estágio;
               Se status[vizinho] = destino ou
                  (status[vizinho] = barramento
                        de alimentação "X" e
                  qualificador do tipo de si
                  nal = "X")
               Então atingiu célula destino;
            Senão Guarda custo[vizinho] = 0 pa
                  ra uso do terceiro estágio;
Fim;
```

É importante notar que também neste procedimento são realizados no máximo dois acessos a cada banco da memória de placa: um acesso de leitura e outro de escrita.

O terceiro estágio do pipeline armazena na memória de fronteira as coordenadas da célula expandida e o custo de todos os vizinhos marcados pelo segundo estágio. Para cada módulo, este estágio deve realizar, em dois passos sequenciais, o armazenamento das informações referentes a cada uma das duas camadas. Portanto, o procedimento executado pelo terceiro estágio envolve no máximo dois acessos de escrita na memória de fronteira. O procedimento executado para cada passo é o seguinte:

```
Procedimento EscreveMemóriadeFronteira;
   Se há vizinho com custo <> 0
   Então Escreve informação {[x,y,z], custos e
   senha atual+1};
        Incrementa registro de endereço limite;
Fim;
```

O algoritmo de retraço que é implementado em software e é executado quando a célula destino é atingida pode ser descrito basicamente pelo seguinte procedimento:

```
Procedimento Retraço;
   Lê célula destino;
   Enquanto status[célula] <> início faça
      Usando seta célula , calcula [x,y,z] da
      próxima célula;
      Atualizastatus;
      Atualiza a bússola da célula, consideran
         do a bússola atual e a seta anterior,
         se existir;
      Se custo dinâmico
      Então Redefine custo dos vizinhos;
      Se há mudança de camada na célula
      Então Redefine via das células vizinhas
         de forma a impedir mudança de camada
         nestas células;
      seta anterior := seta[célula];
      Lê próxima célula;
Fim;

Procedimento Atualizastatus;
   Se última ligação de um sinal
   Então status[célula] := bloqueio porque per
   tence a caminho;
   Senão Se topologia da ligação = cadeia
      Então Se (não é primeira ligação do sinal
                e status[célula] = início)
            Então status[célula] := destino;
            Senão Se (status célula <> destino
                      ou não é primeira ligação)
                  Então status[célula] :=  blo
                     queio porque pertence a ca
                     minho;
      Senão status[célula] := destino;
Fim;
```

O procedimento executado em hardware para a fase de reinicialização é o seguinte:

```
Procedimento Reinicialização;
   Para todo banco da memória de placa,  faça,
   em paralelo:
      endereço := 0;
      Enquanto endereço <= último endereço  do
      banco faça:
         Lê célula;
         senha[célula] := 0;
         Se última ligação
         Então Se status[célula] = destino
               Então status[célula] :=    bloqueio
                  porque pertence a caminho;
         Armazena informação da célula atualiza
               da;
Fim;
```

A análise deste procedimento mostra que cada ciclo da fase de reinicialização executa dois acessos à memória de placa: um acesso de leitura e outro de escrita.

As Figuras 2 e 3 mostram os circuitos de controle de memória de placa e da memória de fronteira, respectivamente, necessários para implementação em hardware dos procedimentos descritos.

5. ESTÁGIO ATUAL E PERSPECTIVAS FUTURAS

O projeto de hardware do acelerador se encontra, no momento em fase final de detalhamento da arquitetura. Sua implementação se dará basicamente com uso de memórias MOS estáticas, microprocessadores e alguns circuitos TTL SSI e MSI. Estima-se que cerca de 300 circuitos integrados serão utilizados na implementação do módulo básico do acelerador.

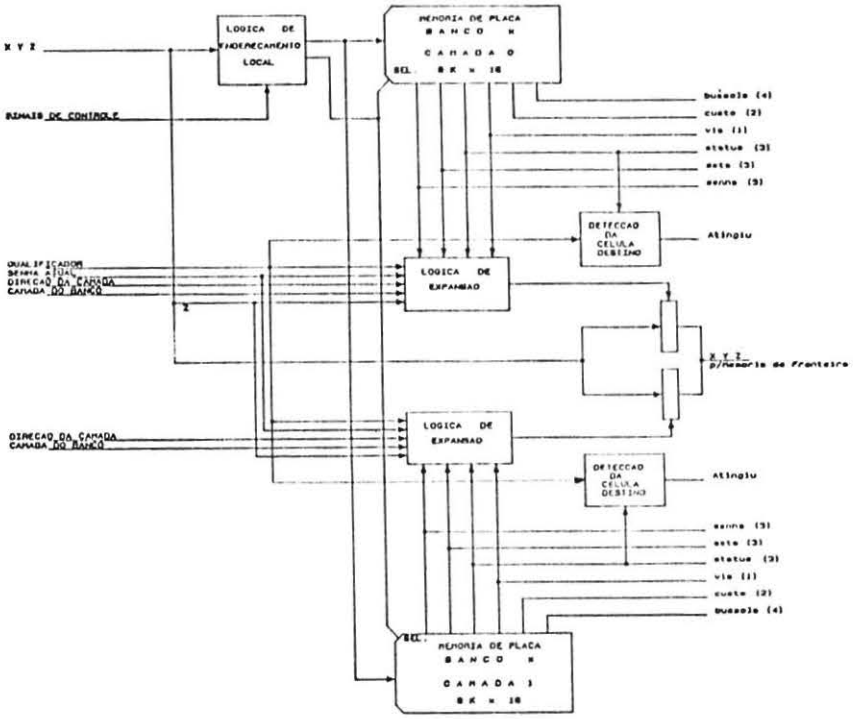

Figura 2. Lógica de Controle da Memória de Placa.

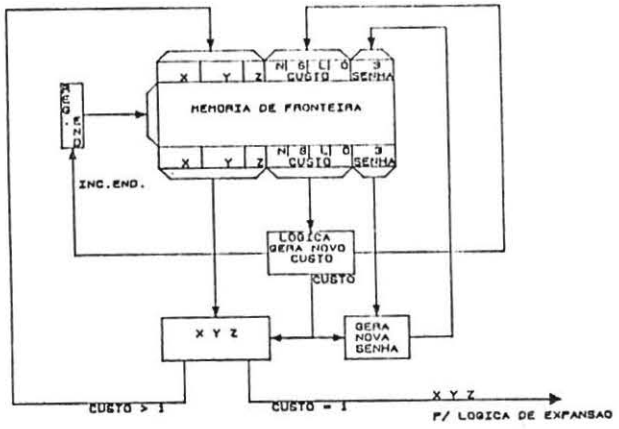

Figura 3. Controle da Memória de Fronteira.

Um simulador da arquitetura do acelerador foi desenvolvido como ferramenta para avaliação do projeto principalmente no que diz respeito a: determinação da melhor partição do acelerador em módulos básicos, avaliação do desempenho do acelerador e medida do grau de utilização dos estágios que compõem os pipelines.

O simulador fornece medidas do tempo gasto em cada fase do algoritmo em relação ao tempo total de processamento e da taxa de ocupação dos estágios dos pipelines. Além disso, o simulador fornece medidas da frequência com que contenções ocorrem durante o processamento. Basicamente, existem dus situações em que podem ocorrer contenções. A primeira delas é quando um módulo assume o controle global da máquina para completar uma operação de roteamento, forçando que os demais módulos tenham suas atividades suspensas. A segunda situação ocorre em problemas de roteamento em que o custo das células não é unitário, quando é possível que haja conflito no acesso à memória de fronteira.

Resultados já obtidos com o simulador, sem levar em conta o paralelismo na operação entre módulos, demonstram que, usando-se custo unitário, o roteamento de cerca de 1000 ligações de

comprimento médio igual a 200 pode ser realizado após cerca de 200 milhões de ciclos de máquinas. A cada ciclo corresponde a execução das tarefas de um estágio do pipeline na fase de expansão ou a uma operação concorrente de zeramento da senha de 8 células na fase de reinicialização. Considerando-se que as fases de expansão e de reinicialização são as que mais consomem tempo de máquina na execução do algoritmo de roteamento e que cada ciclo destas duas fases realiza basicamente dois acessos a memória e, portanto, seu tempo de duração será certamente não superior a 1 microsegundo, podemos estimar que o acelerador completará esta tarefa de roteamento em cerca de 200 segundos, ou seja, em menos de 4 minutos. Tipicamente, a solução deste problema de roteamento consome cerca de 3 horas em máquinas do tipo VAX.

O acelerador funciona integrado a um sistema de software [AUDE88] que executa as seguintes tarefas: alocação dos componentes a serem interconectados, geração automática de uma descrição adequada da superfície a ser utilizada para roteamento a partir de uma descrição dada na forma de texto, ordenação dos sinais a serem roteados segundo critério de escolha do usuário e interface gráfica para visualização dos resultados produzidos pelo roteador em hardware. A exceção do alocador, todos os outros programas já se encontram com seus protótipos iniciais funcionando. O desenvolvimento de software foi todo feito em microcomputadores nacionais de 16 bits e a linguagem empregada foi Pascal.

Estima-se que até o final de 1988 o sistema de roteamento completo estará funcionando em microcomputadores nacionais de 16 bits. Em 1989 pretende-se iniciar o desenvolvimento de circuitos integrados dedicados para implementação dos processadores do acelerador. Com isso, espera-se que o acelerador apresente um ganho ainda maior em velocidade e, obviamente, possua uma implementação em hardware mais compacta.

REFERÊNCIAS

[AUDE88] "Sistema de Geração Automática de Layout com Acelerador de Roteamento", Aude,J.S., Paiva,E.B.M.O., Aude,E.P.L.,Lopes,E.P., Martins,M.F., Pinto,S.B., Trabalho a ser publicado nos anais do VIII Congresso da SBC, 1988;

[BLAN 81] "A Parallel Bit Map Processor Architecture for DA Algorithm", Blank,T.,Stefik, M., VanCleemput, W., Proceedings of the 18th Design Automation Conference, Junho, 1981, pp. 837-845;

[BREU 81] "A Hardware Router", Breuer,M.A., Shamsa,K., Journal of Digital Systems, Vol. 4, no. 4, 1981, pp. 393-408;

[IOSU 86] "A Class of Array Architectures for Hardware Grid Routers", Iosupovici,A., IEEE Transactions on Computer-Aided Design, Vol. CAD-5, No. 2, Abril, 1986, pp. 245-255;

[LEE 61] "An Algorithm for Path Connections and its Applications", Lee,C.Y., IRE Transactions on Electronic Computers, Vol. EC-10, Setembro, 1961, pp. 346-365;

[RUTE84] "A Class of Cellular Architectures to Support Physical Design Automation", Rutenbar,R.A., Mudge,T.N., Atkins,D.E., IEEE Transactions on Compuer-Aided Design, Vol. CAD-3, No. 4, Outubro, 1984, pp. 264-278;

[KRAV86] "Multiprocessor Based Placement by Simulated Annealing", Kravitz,S.A., Rutenbar,R.A., Proceedings of the 23rd Design Automation Conference, Junho, 1986, pp. 567-573;

[PFIS86]"The IBM Yorktown Simulation Engine", Pfister,G.F., Proceedings IEEE, Junho,1986, pp. 850-860;

[RYAN87] "An ISMA Lee Router Accelerator, Ryan,T., Rogers,E., IEEE Design and Test of Computers, Outubro, 1987, pp. 38-45;

[SEIL82] "A Hardware Assisted Design Rule Check Architecture", Seiler,L., Proceedings of the 19th Design Automation Conference,Junho, 1982, pp. 232-238;

[WON 87] "A Hardware Accelerator for Maze Routing", Won,Y., Sahni,S., El-Ziq,Y., Proceedings of the 24th Design Automation Conference, Junho, 1987, pp. 800-806.