UM MODELO EM REDE DE PETRI DE UMA ARQUITETURA PARA A SIMULAÇÃO DE CIRCUITOS

Berenice Camargo Damasceno[*], Mauro Marton[*], Norian Marranghello[**] e Furio Damiani[*]

[*]Faculdade de Engenharia Elétrica - Unicamp
E-Mail: Furio@dsif.fee.unicamp.br

[**]DCCE/IBILCE/UNESP-Campus Rio Preto
E-Mail: Norian@nimitz.ibilce.unesp.br

Resumo: O projeto de circuitos **VLSI** envolve o uso de várias ferramentas de *software,* dentre as quais destacam-se os simuladores de circuitos de alto desempenho. Neste artigo, descreve-se o modelamento de uma arquitetura para a simulação de circuitos, usando-se redes de Petri clássicas. A arquitetura modelada, chamada **ABACUS**, é capaz de simular eficientemente grandes circuitos, garantindo ao mesmo tempo altas velocidades de simulação e um alto nível de detalhamento do comportamento de circuitos **VLSI**. A análise da rede de Petri que modela o comportamento da arquitetura **ABACUS** demonstra que o sistema tem boas propriedades e é, portanto, factível.

Abstract: Among several software tools the design of VLSI circuits requires the use of high performance circuit simulators. In this paper we describe the modelling of the ABACUS circuit simulator architecture using Petri nets. The modelled architecture, that was named ABACUS, can simulate large circuits ensuring both high simulation speeds and high level of acuracy in predicting VLSI circuits behavior. The analysis of the Petri net used to model ABACUS' architecture shows that the system has a very good behavior and that its implementation can be recomended.

1 - INTRODUÇÃO

O contínuo aumento na densidade de CIs tornou necessário o uso de simuladores de circuitos na fase de projeto de circuitos VLSI. Os simuladores de circuito existentes podem prever o comportamento elétrico real de um circuito com bastante detalhes. Contudo, eles ainda não são suficientemente rápidos para simular grandes circuitos VLSI de maneira eficiente, isto é, de modo rápido e barato.

O desempenho dos simuladores de circuitos pode ser melhorado bastante com a mistura de técnicas de *hardware* e *software*. Recentemente, foi proposta a arquitetura ABACUS (Marranghello, 1992), cujo objetivo principal é a simulação eficiente de circuitos grandes, incorporando idéias dentre as mais avançadas de ambas as técnicas. Esta abordagem à simulação de circuitos evita a solução dos imensos sistemas de equações resultantes de circuitos VLSI, com um *hardware* especial que incorpora modelos de elementos de circuitos e sobre o qual os circuitos podem ser mapeados e simulados. A arquitetura ABACUS foi concebida e está sendo desenvolvida para garantir altas velocidades de simulação (cerca de mil vezes a velocidade dos simuladores padrão), mantendo a previsão do comportamento de circuitos VLSI num nível de detalhes comparável ao dos simuladores padrão.

Redes de Petri têm sido usadas no modelamento de sistemas concorrentes (Peterson, 1981), como as arquiteturas a fluxo de dados (Smigelski *et alii,* 1985). Um modelo em redes de Petri para simular a arquitetura ABACUS foi desenvolvido e analisado usando-se o programa SIPRO (Arantes, 1988).

Neste trabalho, apresenta-se o modelo obtido e analisam-se os principais resultados da simulação da arquitetura ABACUS, através de uma rede de Petri clássica, com o programa SIPRO.

Na seção 2, tem-se uma descrição da arquitetura ABACUS. A seguir, na seção 3, faz-se uma revisão dos principais conceitos sobre redes de Petri. A seção 4, contém uma descrição do modelo e uma discussão sobre as principais marcações obtidas na simulação.

* Artigo submetido em 02/08/93
1ª revisão em 05/05/94; 2ª revisão em 31/05/95;
Aceito sob recomendação do Ed.Cons. Prof.Dr Fernando A. Gomide.

2 - A ARQUITETURA ABACUS:

A arquitetura ABACUS (figura 1) é um sistema computacional voltado para a simulação de circuitos. Esta arquitetura possui um processador mais complexo, semelhante ao transputer T800 da INMOS Ltd., que serve como hospedeiro, ou gerente, de um arranjo composto por um grande número de processadores mais simples, cada um dos quais dedicados à tarefa exclusiva de simular os modelos dos componentes do circuito.

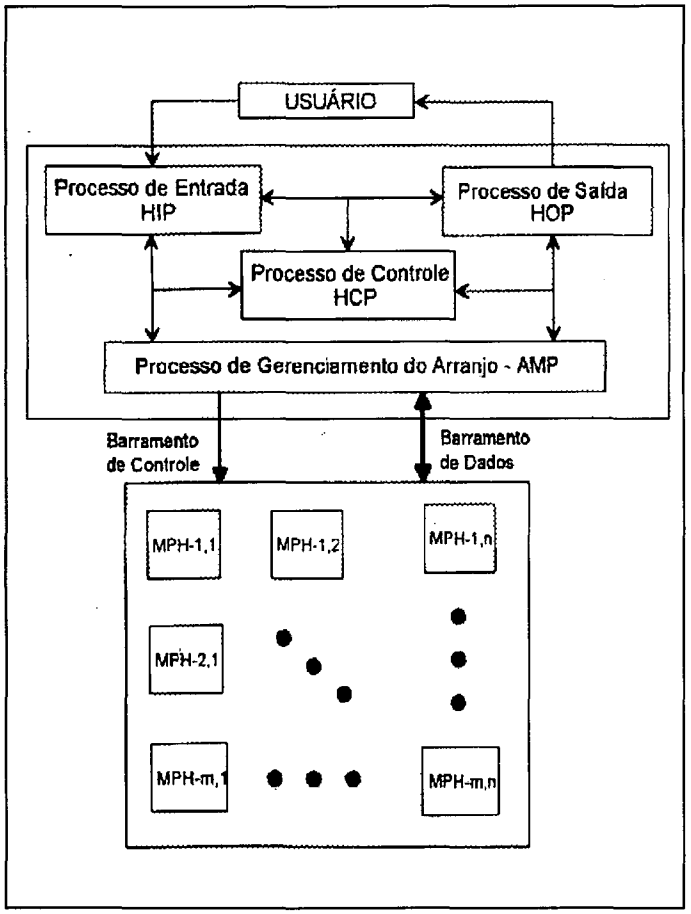

Figura 1 - Diagrama em Blocos da Arquitetura ABACUS

O arranjo é composto por um conjunto de processadores especiais (PE's), os quais trabalham assincronamente, de acordo com um princípio de operação de fluxo de dados. A interconexão entre os PE's do arranjo é reconfigurável dinamicamente pelo hospedeiro, para refletir a topologia do circuito sendo simulado. Cada PE do arranjo emula o comportamento do circuito real. O mapeamento de um circuito nos PE's é exemplificado pela figura 2. Naquela figura queremos mapear um filtro RC passa-altas num arranjo com 4 processadores. O procedimento é simples, basta encontrar três processadores desocupados que possam ser ligados entre si. Na figura em questão aos processadores <1, 1>, <1, 2> e <2, 2> podem ser alocados os elementos FTI, CAP e RES, respectivamente a fonte, o capacitor e o resistor. O mapeamento é concretizado, fechando-se as chaves que ligam os processadores correspondentes, como destacado na figura.

Antes do início da simulação, todos os PE's estão num estado desconhecido. Quando o arranjo é inicializado pelo gerente, cada PE acessa suas portas de entrada na busca por dados. Na primeira tentativa, apenas aqueles PE's que estiverem emulando as entradas primárias do circuito, encontrarão dados de entrada

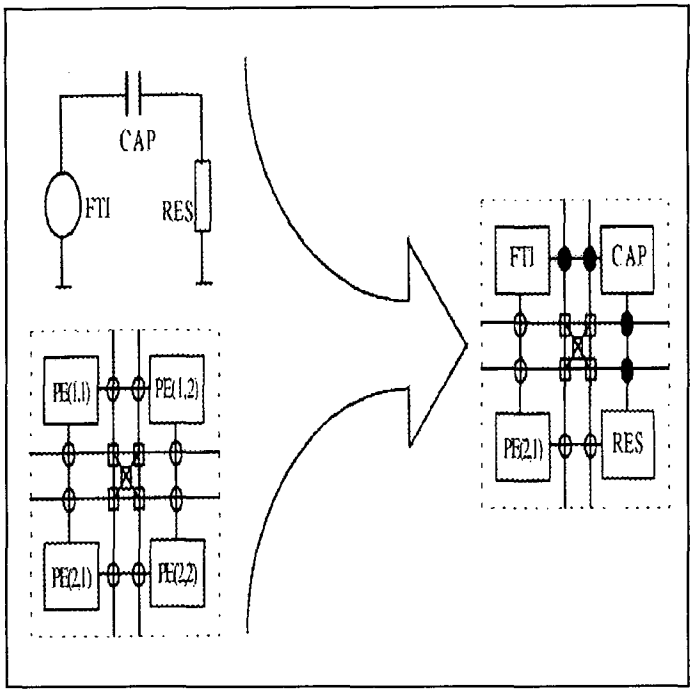

Figura 2 - Exemplo de Mapeamento.

reais. Após computarem suas respostas, estas serão colocadas nas suas portas de saída de modo que os processadores que estiverem ligados a estas portas possam encontrá-los e calcular suas respostas. Enquanto estes processadores computam suas análises, aqueles iniciam um segundo ciclo de cálculo, buscando novos dados nas entradas, processando-os e transferindo os resultados para as saídas.

Ao concluir cada instante de simulação (isto é, quando todos os processadores convergiram para uma solução), os dados são transferidos para o gerente, o qual irá, oportunamente, liberar o início da simulação do instante seguinte, sempre que houver outro instante a ser simulado. Desta forma, o circuito é analisado como se tirássemos uma fotografia de seu funcionamento a cada instante simulado, significando que algumas dificuldades, como malhas de realimentação, são tratadas de maneira natural e sem maiores complicações.

Cada PE (figura 3) possui uma unidade lógica e aritmética (ULA), uma memória de escrita e leitura (MEL), uma unidade de controle elementar (UCE), uma unidade de modelos armazenados (UMA) e várias filas de entrada e saída (FES). Sobre estas unidades pode-se dizer que seu projeto deverá possibilitar a execução dos modelos dos componentes dos circuitos da forma mais eficiente possível, em *hardware*, sempre que as condicionantes do projeto permitirem a implementação do modelo desejado.
De maneira geral, pode-se representar o algorítmo de simulação usado na arquitetura ABACUS através de cinco processos principais: HCP, HIP, HOP, AMP e MPH. Os quatro primeiros processos seriam executados no processador hospedeiro e o último em cada um dos processadores especiais do arranjo (ver figura 1) (Marranghello & Damiani, 1990; Marranghello, 1992).

O processo HCP serve para controlar a operação do sistema de simulação. Este controle deve ser entendido, não como uma sincronização dos demais processos, mas como um gerenciamento assíncrono dos mesmos, ou seja, o HCP inicia

e encerra a operação do ambiente de simulação e ativa e desativa cada um dos demais processos do computador hospedeiro, sempre que necessário.

O processo HIP tem a finalidade de processar os dados de entrada fornecidos pelo usuário, organizando-os para o tratamento pelos demais processos do simulador.

O processo HOP destina-se ao fornecimento das respostas solicitadas pelo usuário, a partir das soluções geradas pelo simulador e da base de dados organizada pelo processo HIP.

O processo AMP tem por objetivo controlar o funcionamento do arranjo de processadores. Novamente, o controle não deve ser entendido como uma sincronização sistemática dos processadores, mas sim como um ponto onde, após o término de suas tarefas, os processadores do arranjo se encontram para receber outras tarefas.

O quinto processo (MPH) é aquele que roda em cada um dos PE's. Ele é que vai efetivamente executar a simulação dos componentes do circuito, com base nos dados fornecidos pelo processo AMP.

Resumindo, a arquitetura ABACUS possui vários processadores. Nas memórias de cada um destes processadores ficam armazenados os modelos dos diversos elementos de circuito, que podem ser simulados. Estes processadores são interconectados através de um conjunto de barramentos programáveis, como se interconectássemos o próprio circuito em uma *protoboard*. Em seguida, os dados primários são fornecidos ao arranjo e a simulação é iniciada. A partir do estado inicial, todos os processadores do arranjo começam a trabalhar simultaneamente, buscando dados em suas filas de entrada, processando-os e transferindo os resultados às suas filas de saída. De fato, há um retardo para que os sinais de entrada se propaguem à saída, como num circuito real. Inicialmente, apenas o primeiro nível de processadores, isto é, apenas aqueles diretamente conectados às entradas primárias do circuito, gerarão sinais diretamente derivados das condições de contorno do circuito. À medida que as iterações são processadas, estes sinais reais propagam-se pelo circuito (arranjo), até alcançarem as saídas. Este processo continua até que todo o arranjo se estabilize para um dado conjunto de sinais de entrada. Neste momento, as soluções são armazenadas, o tempo é incrementado e um novo conjunto de entradas primárias é aplicado ao circuito. O processo descrito neste parágrafo é repetido até que todos os instantes desejados sejam analisados.

O fundamental na arquitetura ABACUS é que o paralelismo é implementado na sua forma mais ampla, ou seja, apenas os instantes inicial e final do funcionamento dos processadores do arranjo são determinados pelo processador hospedeiro, mas durante a execução da simulação, os PE's funcionam concorrentemente e de forma totalmente assíncrona, tanto em relação ao hospedeiro, quanto entre si.

3 - REDES DE PETRI:

A teoria de redes de Petri surgiu há aproximadamente 30 anos, com a dissertação de **C. A. Petri**, objetivando formalizar uma base teórica e conceitual para a descrição, de modo exato e uniforme, do maior número possível de fenômenos relacionados às transmissões e transformações de informação (Reizig, 1985).

Nos seus primeiros dez anos de existência, aproximadamente, a divulgação desta teoria foi relativamente precária. Contudo, nos últimos 20 anos vem se tornando uma ferramenta bastante conhecida e utilizada para o modelamento de vários tipos de sistemas. Dentre estes destacam-se o *hardware* e o *software* de sistemas de computação (Peterson, 1981).

Uma rede de Petri pode ser representada por um multígrafo direcionado e bipartido (Peterson, 1981). O fato de ser um multígrafo, significa que é possível termos vários arcos entre dois nós quaisquer do grafo. Como é direcionado, os arcos têm

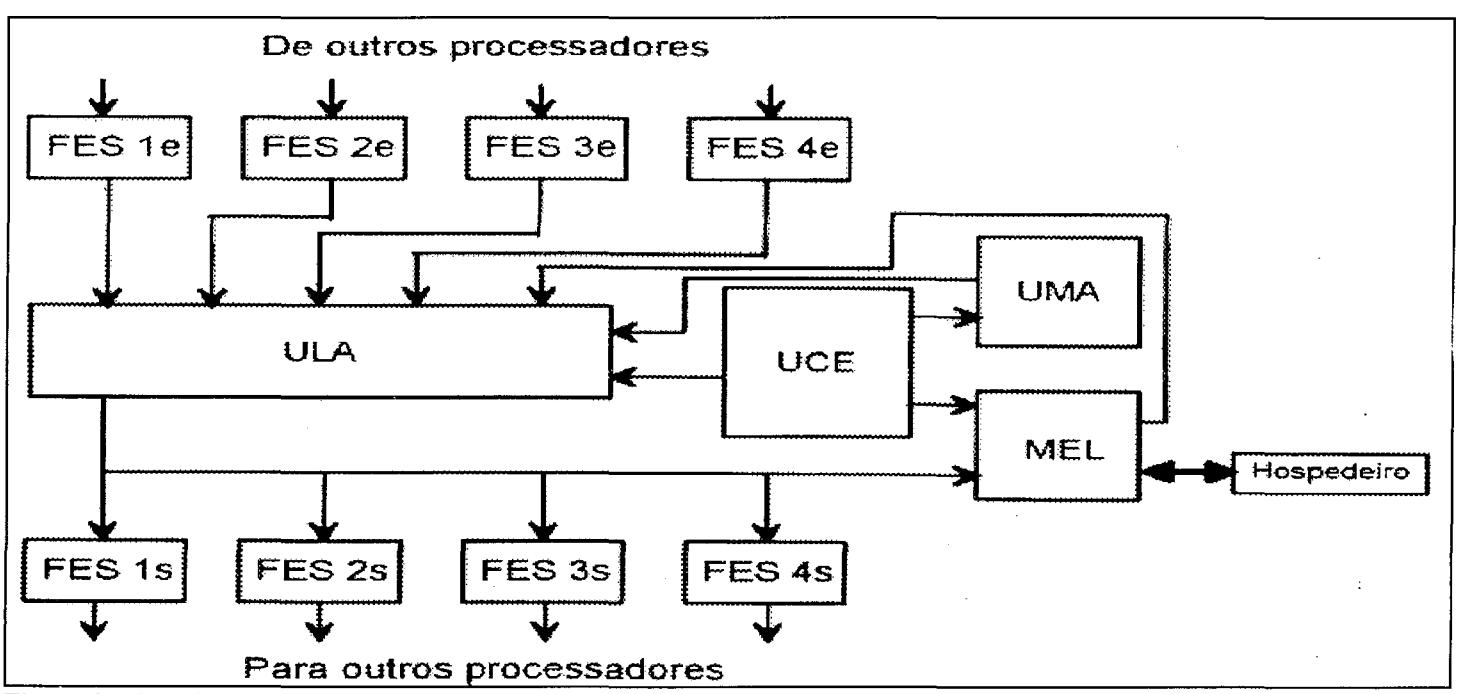

Figura 3 - Arquitetura de um processador elementar.

nós de origem e de término. As redes de Petri são bipartidas porque o grafo que as representam possuem dois tipos de nós, lugares e transições, e cada arco liga nós de tipos diferentes, isto é, lugares a transições ou transições a lugares.

Formalmente, pode-se representar uma rede de Petri clássica por uma quíntupla:

RP = (L, T, E, S, M_0),

onde os elementos da quíntupla são definidos como um conjunto de lugares (L={l_1, l_2, ..., l_u} com u finito e u 0), um conjunto de transições (T={t_1, t_2, ..., t_v} com v finito e v 0), uma função de entrada (E), uma função de saída (S) e uma marcação inicial (M_0). Os lugares representam os possíveis estados do objeto sendo modelado. As transições representam eventos cujo acontecimento muda o estado global do objeto modelado. Os conjuntos de lugares e transições são disjuntos, ou seja, L T= . As funções de entrada e de saída relacionam lugares e transições. Desta forma, a função de entrada é um mapeamento de cada transição t_i T (0 i v) num conjunto de lugares E(t_i) L, ditos lugares de entrada da transição t_i, e a função de saída é um mapeamento de cada transição $t_{j<}$.T (o j v) num conjunto de lugares S(t_j) L, ditos lugares de saída de t_j.

Uma marcação M é uma atribuição de marcas a lugares da rede de Petri. Atribuir marcas a lugares significa dar-lhes condições de habilitarem transições. Isto significa que uma transição t_k T, (0 k v) estará habilitada, quando cada um dos lugares l_n E(t_i), (0 n u), possuir pelo menos uma marca. Uma marcação M=(m_1, m_2, ..., m_u), com u inteiro, finito e não nulo, é um vetor onde cada elemento m_i, 1 i u, representa o número de marcas disponíveis no lugar. Em outras palavras, uma marcação pode ser denotada como um vetor, de dimensão igual ao número total de lugares da rede, com cada componente representando o número de marcas que possui cada um dos lugares. A marcação inicial M_0 é a distribuição de marcas na rede de Petri no instante inicial de sua execução.

Existem vários modelos de redes de Petri (Damasceno, 1989), sendo que o modelo utilizado neste trabalho é o da rede de Petri clássica (Peterson, 1981). A rede de Petri clássica pode ser denominada como uma rede de lugares/transições (*place-transition nets*).

Qualquer transição habilitada pode disparar e o efeito deste disparo é a retirada de uma ou mais marcas de cada um dos lugares de entrada da transição e a inclusão de uma ou mais marcas em cada um de seus lugares de saída. O número de marcas retiradas dos lugares de entrada e colocadas nos lugares de saída de uma determinada transição depende da multiplicidade dos arcos entre a transição e os lugares correspondentes. Note-se que, nas redes de Petri clássicas, só é permitido um arco entre um lugar e uma transição. Portanto, apenas uma marca é retirada de cada lugar de entrada e somente uma marca é colocada em cada lugar de saída.

Ao analisar uma rede de Petri, procura-se verificar três propriedades básicas, que representam características do sistema modelado. Tais propriedades são: limitabilidade, vivacidade e reiniciabilidade, descritas a seguir (Damasceno, 1989).

Uma rede de Petri é definida como limitada a um valor inteiro *n* se para qualquer marcação M, decorrente da marcação inicial M_0, o número de marcas (m_i) em cada lugar l_i L (1 i u) for sempre menor que *n*. Uma rede de Petri limitada, com *n*=1, é dita segura. Uma rede de Petri não limitada possui um número infinito de marcações e representa um sistema físico impossível de ser implementado.

Uma rede de Petri é dita viva para uma marcação inicial M_0, se para qualquer marcação decorrente de M_0 existir uma sequência de disparos, que torne disparável qualquer outra transição da rede. Em outras palavras, uma rede de Petri viva está livre de impasses, pois, todas as suas marcações, decorrentes da inicial, levam ao disparo de alguma transição. Ao contrário, numa rede de Petri onde a propriedade de vivacidade não se verifica, existe pelo menos uma marcação a partir da qual nenhuma transição pode ser disparada, ou seja, o sistema físico estaria travado.

Finalmente, uma rede de Petri é reiniciável, para uma dada marcação inicial M_0, se, para qualquer marcação decorrente desta, existir uma sequência de disparos, que faça a rede voltar à marcação M_0.

4 - MODELAMENTO E SIMULAÇÃO:

Como acabamos de ver, redes de Petri constituem uma ferramenta matemática usada para a representação formal de sistemas a eventos discretos. Devido à sua simplicidade funcional e à sua característica gráfica elas têm sido muito usadas para a descrição de sistemas concorrentes, pois permitem uma fácil visualização dos sistemas modelados.

Em nosso caso, convertemos uma descrição informal da arquitetura ABACUS numa descrição formal usando rede de Petri clássica, tomando o máximo de cuidado ao manter as características da arquitetura e evitando ações ambíguas.

A figura 4 mostra o grafo do modelo em rede de Petri de ABACUS, onde os círculos nomeados L_{xx} indicam os lugares da rede de Petri, as barras nomeadas B_{xx} indicam as transições e as linhas tracejadas relacionam cada processo ABACUS com a porção correspondente na rede de Petri.

Os lugares da rede de Petri representam estados internos dos processos ABACUS e as transições representam as ações executadas pelos processos ABACUS. A relação completa da correspondência entre cada transição no modelo e a respectiva ação do sistema encontra-se no trabalho de Marranghello (1992).

O modelo da figura 4 foi simulado com o programa SIPRO; (Marranghello, 1992) o qual permite a análise de redes de Petri quanto às suas propriedades de limitabilidade, vivacidade e alcançabilidade. Ele também gera diagnósticos para a correção de falhas do modelo, tais como transições que não disparam. O SIPRO tem capacidade para simular até 100 transições e 100 lugares, podendo gerar até 1000 marcações.

Em nosso modelo, a marcação inicial é M_0 = {L_1, L_{20}} e representa o estado inicial da arquitetura no qual tanto o processador hospedeiro (L_1) quanto o arranjo (L_{20}) estão à

espera de um circuito a ser simulado. A submissão de um circuito para simulação, ativa a transição (b_1), iniciando o processo de simulação e levando o modelo à marcação M_1, com os processos HIP (L_2), AMP (L_3) e HOP (L_4) inicializados. A marca em L_{46} serve apenas para permitir ao modelo retornar ao estado inicial, após a conclusão dos demais processos. Representando o disparo da transição b_1 a partir da marcação M_0 por (M_0, b_1) teríamos, neste caso:

(M_0, b_1) $M_1 = \{L_2, L_3, L_4, L_{20}, L_{46}\}$.

A partir daí, os processos HIP, AMP e HOP seguem sua execução concorrente, respectivamente processando os dados de entrada, preparando-os para o processamento no arranjo e formatando a base de dados a ser apresentada na saída. Estes processos interagem, como, por exemplo, na transição b_8 onde os dados de entrada são transferidos do processo HIP ao processo HOP, ou na sequência de transições b_9, b_{10} e b_{11} onde o processo HIP consulta o processo AMP sobre o número de processadores disponíveis no arranjo, para poder subdividir o circuito adequadamente, ou ainda na sequência b_{13} e b_{14} onde estas partições do circuito são transferidas ao processo AMP para o correto mapeamento no arranjo.

Num determinado instante o modelo encontra-se na marcação $M_{40} = \{L_{16}, L_{20}, L_{21}, L_{46}, L_{47}\}$ quando, através do disparo da transição b_{15}, encerra-se o processamento dos dados de entrada. O encerramento do processo HIP pode ser representado por:

$M_{40}, b_{15})$ $M_{43} = \{L_{17}, L_{19}, L_{20}, L_{21}, L_{46}, L_{47}\}$

A seguir procede-se a transferência de dados do hospedeiro para o arranjo e dá-se início à simulação do circuito no arranjo. Esta inicialização é representada no modelo da figura 4 pela seguinte sequência de disparos:

(M_{43}, b_{17}) $M_{48} = \{L_{17}, L_{19}, L_{20}, L_{22}, L_{25}, L_{46}, L_{47}\}$

(M_{48}, b_{20}) $M_{55} = \{L_{17}, L_{19}, L_{20}, L_{22}, L_{26}, L_{30}, L_{46}, L_{47}\}$

(M_{55}, b_{18}) $M_{59} = \{L_{17}, L_{19}, L_{24}, L_{26}, L_{30}, L_{46}, L_{47}\}$

A partir daí, dá-se a simulação do circuito pelos processadores do arranjo. Note-se que as transições b_{23} e b_{29} representam a comunicação entre tais processadores. A simulação continua até que através do disparo consecutivo das transições b_{24}, b_{26}, b_{28} e b_{30} o processo AMP decida que o arranjo convergiu para a solução do problema. Neste instante temos a marcação $M_{99} = \{L_{17}, L_{19}, L_{34}, L_{38}, L_{46}, L_{47}, L_{48}\}$ da qual, pelo disparo da transição b_{34}, leva ao encerramento do processamento no arranjo. Esta situação é representada no modelo por:

(M_{99}, b_{34}) $M_{103} = \{L_{17}, L_{19}, L_{20}, L_{38}, L_{46}, L_{47}, L_{49}\}$.

Neste instante, os resultados da simulação no arranjo são transferidos para o processador hospedeiro através do disparo da transição b_{35}, ou seja:

(M_{103}, b_{35}) $M_{108} = \{L_{17}, L_{19}, L_{20}, L_{39}, L_{46}, L_{47}\}$

Agora, são três as ações possíveis: 1) iniciar um novo instante de simulação para a mesma partição, por meio da transição b_{36}; 2) transferir os resultados da simulação da partição para o HOP e iniciar a simulação de uma nova partição, pela transição b_{37}; ou 3) não havendo outra partição, enviar os resultados finais para o HOP, disparando a transição b_{38}. é importante ressaltar que o SIPRO analisa todas as alternativas possíveis, escolhendo e disparando uma transição de cada vez. Contudo, para efeitos deste exemplo, analisar-se-á apenas a última alternativa, onde tem-se:

(M_{108}, b_{38}) $M_{115} = \{L_{17}, L_{19}, L_{20}, L_{40}, L_{46}, L_{47}, L_{54}\}$

O disparo seguinte representa a formatação final dos dados pelo processo HOP, deixando-os à disposição do usuário:

(M_{115}, b_{44}) $M_{120} = \{L_{17}, L_{19}, L_{20}, L_{40}, L_{42}, L_{46}\}$

Isto representa, na prática, o fim da simulação propriamente dita. Portanto, o processo AMP pode ser concluído pelo seguinte disparo:

(M_{120}, b_{40}) $M_{122} = \{L_{17}, L_{19}, L_{20}, L_{42}, L_{44}, L_{46}, L_{50}\}$

A apresentação dos resultados da simulação ao usuário e o consequente encerramento do processo HOP são modelados pelos dois disparos a seguir:

(M_{122}, b_{41}) $M_{123} = \{L_{17}, L_{20}, L_{43}, L_{44}, L_{46}\}$ e

(M_{123}, b_{42}) $M_{124} = \{L_{17}, L_{20}, L_{44}, L_{45}, L_{46}\}$

Uma vez concluída a simulação e apresentados os resultados ao usuário, pode-se encerrar o processo de controle, HCP, e o sistema deve retornar ao estado inicial a espera de um novo circuito para ser simulado. Isto ocorre em nosso modelo, quando do seguinte disparo:

(M_{124}, b_{43}) M_0.

O programa SIPRO gerou um total de 125 marcações para o modelo recém descrito e as propriedades de limitabilidade, vivacidade e alcançabilidade foram verificadas.

A limitabilidade de nosso modelo pode ser interpretada como representando uma arquitetura livre de contenções nos barramentos e com um sistema de memória adequadamente dimensionado.

A propriedade de vivacidade garante que a rede, e consequentemente o sistema por ela modelado, está livre de bloqueios fatais e que todas as partes do sistema modelado são acessíveis.

Finalmente, a alcançabilidade de uma rede de Petri representa a capacidade do sistema modelado de retornar ao seu estado inicial após executar uma ou mais tarefas, o que, em nosso caso, pode ser interpretado como uma indicação de que o sistema modelado é convergente.

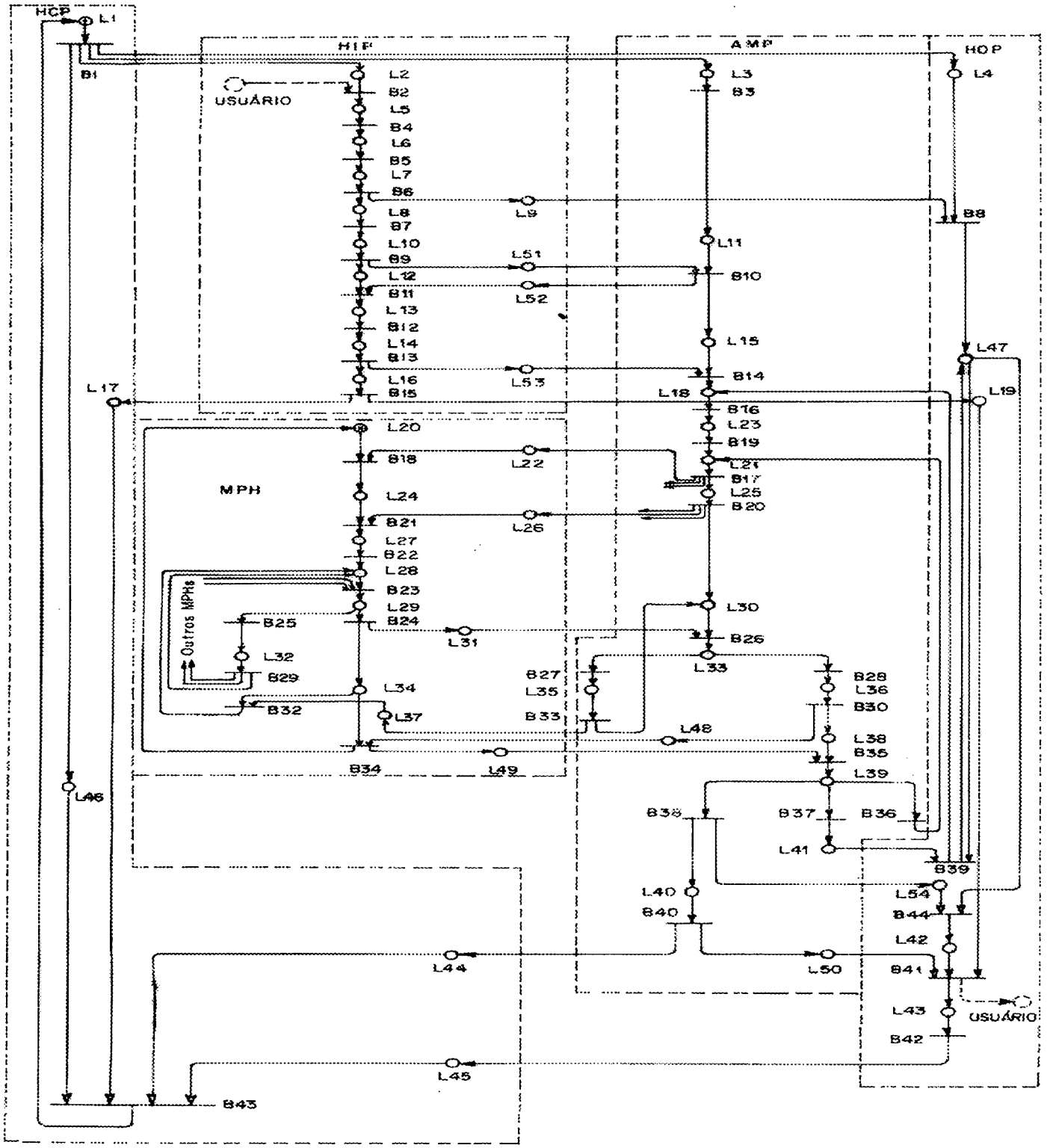

Figura 4 - Modelo em Rede de Petri da Arquitetura ABACUS

5 - CONCLUSÕES:

Em parte, este trabalho serviu para demonstrar a utilidade de redes de Petri como uma ferramenta de auxílio na análise de arquiteturas computacionais concorrentes. Contudo, o SIPRO, programa que analisa as propriedades de redes de Petri clássicas, não permitiu a representação adequada do paralelismo inerente à arquitetura ABACUS. Atualmente, estamos desenvolvendo um modelo usando rede de Petri numérica (Damasceno & Yamakami, 1991), as quais possuem condições de habilitação e disparo de transições de tipos de marcas diferentes, para representar o paralelismo real da arquitetura ABACUS. A este modelo, será introduzida temporização (Marton, 1989), de forma que se possa analisar o desempenho da arquitetura ABACUS.

AGRADECIMENTOS:

Os autores gostariam de expressar seus agradecimentos ao prof. A. Yamakami do DT/FEE/UNICAMP, pelas suas sugestões e a J. H. B. dos Santos, do DCCE/IBILCE/UNESP-SJRP, pelo auxílio na preparação deste artigo. Durante a realização deste trabalho o prof. Marranghello estava no DSIF/FEE/UNICAMP, com uma bolsa de doutoramento da FAPESP (Processo no. 90/4744-0).

REFERÊNCIAS BIBLIOGRÁFICAS:

Arantes, M. P. C. (1988). *Um Analisador Automático de redes de Petri para validação de protocolos de comunicação*. Dissertação de Mestrado, DT/FEE/UNICAMP, 169pp.

Damasceno, B. C. (1989). *Modelos baseados em extensões de rede de Petri para análise de protocolos de comunicação*. Dissertação de Mestrado, DT/FEE/UNICAMP.

Damasceno, B. C. and A. Yamakami (1991). *FMS: A tool for analysis and simulation based on Numerical Petri Nets*. 30° TIMS (Julho).

Marranghello, N. (1992). *Uma metodologia para a simulação de circuitos VLSI*, Tese de Doutoramento, DSIF/FEE/UNICAMP, 110pp.

Marranghello, N. and F. Damiani (1990). Architecture and Algorithm of a circuit simulator, *Proceedings of the SPIE,* vol. 1405, pp. 167-173.

Marton, M. (1989). *Um Analisador automático de redes de Petri temporizadas*. Dissertação de Mestrado, DT/FEE/UNICAMP, 107pp.

Peterson, J. L. (1981). *Petri net theory and modelling of systems*, Prentice-Hall Inc., 263pp.

Reizig (1985). *Petri Nets - An Introduction*. Springer Verlag, 161pp.

Smigelski, T., T. Murata, and M. Sawa (1985). A timed Petri net model and simulation of a dataflows computer. *IEEE Transactions on Computers,* pp. 56-63.