Processamento Distribuído de Diagramas de Decisão Binária

Ricardo Hernandes Fernandes, Francisco Assis Moreira do Nascimento

Universidade Luterana do Brasil - ULBRA
R. Miguel Tostes, 101, 477 4000, 477 1313
ricardo@inf.ulbra.tche.br, assis@ulbra.tche.br

Introdução

Os diagramas de decisão binária ordenados e reduzidos, *reduced ordered binary decision diagrams* (ROBDDs) têm se tornado a estrutura de dados predominante para representar funções booleanas na implementação de ferramentas de verificação formal, síntese lógica, dentre outras na área de projeto de sistemas digitais assistido por computador (*Computer-Aided Design/Electronic Design Automation* - CAD/EDA). Na prática, a memória requerida pelos grandes ROBDDs tipicamente se torna um fator limitante para estas ferramentas de verificação formal [STO 96]. Em alguns casos, especialmente com o uso do reordenamento dinâmico de variáveis dentro dos ROBDDs [BRY 86], o tempo de execução também passou a ser um fator importante a ser analisado. Por isso, muitas universidades estão investindo bastante em pesquisas, a fim de implementar ROBDDs com mais eficiência.

Neste trabalho, é apresentado um pacote para processamento distribuído de ROBDDs. Com este pacote, de maneira transparente para o programador, ROBDDs complexos são particionados e as operações a serem realizadas sobre eles são distribuídas pelos nodos de um multicomputador. Com esta abordagem é possível se manipular ROBDDs grandes, envolvendo muitas variáveis, mas reduzindo o tempo de processamento necessário para efetuá-los.

A seguir, serão introduzidos alguns conceitos sobre ROBDDs e como eles podem ser processados em uma máquina do tipo multicomputador. Em seguida será descrito um pacote para programação distribuída voltada para manipulação de ROBDDs. Os resultados experimentais do uso deste pacote, realizado em um multicomputador do tipo Beowulf [STE 99], para a verificação de equivalência de circuitos, possibilitou a definição de uma nova solução, teoricamente mais eficiente em termos de performance, para o particionamento e processamento distribuído dos ROBDDs. Atualmente, esta nova abordagem está sendo implementada usando a biblioteca LAM/MPI [LAM 01].

Diagramas de Decisão Binária

Diagramas de Decisão Binária (*Binary Decision Diagrams* - BDDs) são grafos orientados acíclicos com uma ou mais raízes, onde cada nodo pode ser "pai" de dois e somente dois nodos filhos. A idéia de se usar diagramas de decisão binária para representar funções boolenas foi primeiramente proposta por Lee [LEE 59] e posteriormente por Akers [AKE 78]. Mas, somente depois do trabalho de Bryant [BRY 86] é que os BDDs passaram a ser amplamente utilizados em várias áreas de CAD/EDA. Bryant desenvolveu algoritmos eficientes para a manipulação de BDDs baseando-se na ordem das variáveis. Com isto, os BDDs de Bryant foram denominados de diagramas ordenados de decisão binária (*Ordered Decision Binary Diagrams* -OBDDs). Assim, Bryant definiu OBDDs impondo restrições na formulação original destes grafos, de maneira que eles se tornassem canônicos, caracterizando, portanto, de forma única uma função booleana. Ou seja, todas as funções que forem funcionalmente equivalentes serão reduzidas ao mesmo diagrama canônico. Com o uso dos OBDDs, as funções booleanas passaram a ser representadas como árvores ordenadas, onde os nodos internos correspondem as variáveis binárias, sobre as quais a função é definida, e os nós terminais ou folhas são rotulados com os valores *0* e *1*. Desta forma, operações lógicas sobre funções booleanas podem ser implementadas como algoritmos aplicados em estruturas de dados, principalmente, focando-se em estruturas de árvores e grafos.

Bryant provou que se aplicando uma série de regras de redução em um BDD usando uma determinada ordem para as variáveis, obtém-se um OBDD canônico e reduzido, ou seja, um ROBDD (OBDD reduzido, *Reduced* OBDD) capaz de definir unicamente uma função booleana e sendo de forma irredutível. Devido ao fato de que um ROBDD é uma representação canônica, muitas propriedades funcionais podem ser verificadas. Por exemplo, duas funções diferentes mas funcionalmente equivalentes resultariam em um mesmo ROBDD quando fossem ambas reduzidas. Assim sendo, é possível aplicar o método dos ROBDDs na minimização lógica de funções booleanas [RUD93] para a síntese de circuitos integrados.

No entanto, o tamanho de um diagrama de decisão binária para uma dada função depende da ordem escolhida para as variáveis. Encontrar uma ordem adequada para as variáveis no diagrama é, quase sempre, um problema crítico. Existem vários métodos heurísticos para se lidar com este problema. Mesmo assim, o fato de realizar-se determinadas operações sobre alguns ROBDDs mais complexos pode vir a ocasionar uma demanda muito grande de processamento.

Processamento Paralelo para ROBDDs

Uma possível abordagem ao problema da manipulação de ROBDDs é a do processamento paralelo para acelerar as operações entre eles. Entretanto, é importante observar que estas operações sobre grafos de decisão binária não são fáceis de serem paralelizadas. Mesmo assim, muitos grupos de pesquisa têm produzido ferramentas para programação paralela dos ROBDDs distribuídos [STO 96]. A maioria dos esforços tem se concentrado nas arquiteturas de memória compartilhada, como os multiprocessadores e as máquinas vetoriais [STO 96]. Neste trabalho, o foco de interesse está na computação paralela através das arquiteturas de memória distribuída, como é o caso dos multicomputadores. Assim, um dos principais objetivos é o de implementar operações binárias sobre diferentes ROBDDs complexos. Em se tratando de uma arquitetura paralela com memória distribuída, a programação é realizada através da utilização de trocas de mensagens de dados. Estas trocas de mensagens, contendo porções distintas de processamento, são internamente transmitidas pela rede de interconexão de um *cluster* de computadores do tipo *Beowulf* [SPE 00].

Embora, a implementação não seja fácil, percebeu-se que muitas operações realizadas sobre estes grafos podem ser feitas de forma independente. Ou seja, as operações que não apresentam dependências funcionais de controle ou de dados podem ser separadas e processadas em paralelo. Assim, as operações entre ROBDDs, efetuadas de forma remota, geram novos diagramas de decisão que são retornados ao processo principal. Este processo, por sua vez, reúne todos os diagramas já computados. Este paradigma proposto também é conhecido como computação paralela por demanda de tarefas e é muito citado na literatura. Obviamente, na área de síntese lógica as operações lidam com ROBDDs muito mais complexos do que pequenas porções de computação. Quando o processo de minimização lógica [RUD 93] é aplicado aos diagramas, a fim de se alcançar novas formas de redução do mesmos, uma das técnicas mais utilizadas é a de reordenação em tempo de execução. Os eficientes pacotes seqüenciais (por exemplo CUDD [SOM 01]), que implementam operações entre ROBDDs, fazem a reordenação das variáveis de forma dinâmica em memória. Na verdade, os algoritmos para manipulação destes ROBDDs, geralmente utilizam muita recursividade. Além disso, o processador fica bastante sobrecarregado, quando realiza atividades como: técnicas de minimização, reordenação dinâmica, composição ou combinação de diagramas, redução à forma canônica e comparação funcional de circuitos lógicos. Portanto, alguns recursos de hardware e do sistema operacional, a saber: memória principal, memória *cache*, processador, escalonadores de processos e outros são altamente requisitados quando se manipula estes diagramas em memória. Com o uso de um multicomputador, estes grafos podem ser separados e distribuídos. Deste modo, um melhor aproveitamento dos recursos é alcançado, possibilitando uma manipulação menos onerosa dos ROBDDs complexos, além de reduzir o tempo necessário para processá-los.

Pacote para Processamento Distribuído de ROBDDs

Como dito anteriormente, a maioria dos pacotes desenvolvidos para processamento distribuído de ROBDDs é voltada para multiprocessadores. Um dos poucos orientados aos multicomputadores é o pacote BDDNow [MIL 98] escrito por Milvang e HU da Universidade de Copenhagen. O BDDNow está implementado no ambiente paralelo através da biblioteca PVM usando como base um *subset* do pacote seqüencial CUDD de Fábio Somenzi da Universidade do Colorado [SOM 01]. Assim, o BDDNow implementa uma camada de software em linguagem C com as funções do PVM sobre o CUDD seqüencial, transformando-o em um pacote paralelo, de modo que fique transparente. Portanto, os programadores podem manipular operações entre ROBDDs sem se preocuparem com detalhes de paralelização e comunicação, pois isto é realizado pela ligação do seus códigos binários com a biblioteca estática *libbddnow.a* depois que o pacote for compilado em ambiente UNIX.

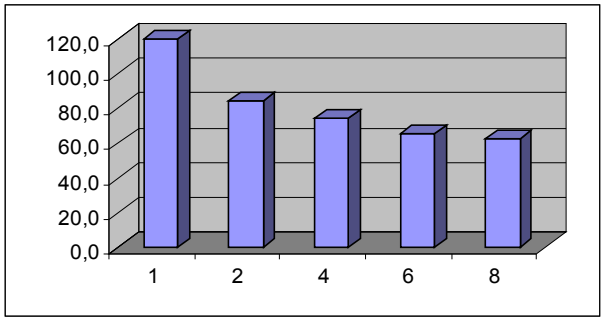
Testes e Experimentos

Foram realizados vários experimentos com o pacote BDDNow usando como um dos exemplo a verificação de equivalência entre circuitos lógicos. Os experimentos foram executados no multicomputador, construído junto ao curso de ciência da computação como parte de um projeto de pesquisa. A máquina paralela consistiu em um agrupamento do tipo Linux/Beowulf [SPE00] de 8 estações de trabalho usando o padrão *Ethernet* IEEE 802.3 para a interconexão das máquinas. Para a programação distribuída, se dispôs das bibliotecas PVM, LAM/MPI e MPICH. Os casos de testes utilizados nos experimentos foram os circuitos do *benchmark* do ISCAS'85 [ISC 85]. Quando estes arquivos descritores de circuitos lógicos foram pensados na época, a sua principal intenção era a comparação para fins de *benchmark* de circuitos relativamente pequenos. Portanto, a solução seqüencial (CUDD) tomando-se estes pequenos arquivos descritores sempre foi melhor do que a abordagem paralela (BDDNow) em nossos testes [FER 01]. Não se encontrou nenhum arquivo de circuito ISCAS' 85 que se ajustasse em 100MB ou 64MB para os testes de ROBDDs exponenciais que é o foco neste trabalho. A intenção, desde o início, era a de se testar o pacote paralelo em ROBDDs realmente muito grandes. Então, foram construídos diagramas para *n* funções diferentes g_0, ..., g_{n-1}, onde:

$$g_i = \bigwedge_{j=0}^{n-1} (a_i \vee B_{i+j \bmod n})$$

Assim, foi possível obter um BDD de tamanho exponencial, de tal forma que era possível conseguir um tamanho adequado apenas controlando a variável *n*. Para n=20, o grafo ordenado tinha 5,25 milhões de vértices e o pacote CUDD requeria 93MB de memória. O BDDNow foi testado sobre este diagrama bastante grande. Os resultados podem ser vistos a seguir, os tempos estão em segundos e são em função do número de estações de trabalho do multicomputador.

Estações	Tempo
1	119,70
2	84,62
4	74,20
6	65,00
8	62,80



Conclusões e novos Trabalhos

A implementação paralela para a manipulação de diagramas de decisão binária nem sempre é a soluções mais eficiente. No caso de circuitos relativamente pequenos, como no exemplo dos

arquivos do *benchmark* ISCAS'85, ficou comprovado que a implementação paralela traz um efeito negativo com relação ao ganho computacional.

No momento, baseado nos resultados obtidos com estes experimentos, está sendo implementada uma nova abordagem para o uso dos ROBDDs de modo a se obter um melhor desempenho. Ou seja, está sendo desenvolvido um novo pacote paralelo que ao invés de utilizar-se de chamadas ao PVM o mesmo irá executar sob o padrão MPI de troca de mensagens. Na programação do novo pacote distribuído está se utilizando linguagem C com a biblioteca LAM/MPI [LAM 01]. O novo pacote foi batizado como MPIBDDNow e com ele será possível fazer uma comparação entre as bibliotecas PVM [GEI 94] e LAM/MPI [LAM 01]. Assim, será possível comparar os resultados de *speedup*, *overhead*, comunicação e sincronização de ambas as bibliotecas sob diagramas médios e exponenciais pesquisando as suas semelhanças e diferenças [FER 01]. Além disto, alguns outros trabalhos estão sendo pensados com o propósito de unificar a área de síntese lógica com o processamento paralelo e a computação de alto desempenho.

Referências

[AKE 78] AKERS, S. **Binary decision diagrams**. IEEE Transactions on Computers, v. C-27, n. 6, 509-516, june 1978.

[BRY 86] BRYANT, R. E. **Graph-based algorithm for boolean function manipulation**. IEEE Transaction on Computers, New York, v.C-35, n.8, p.677-691, August 1986.

[FER 01] FERNANDES, R. H. **Implementação de uma aplicação distribuída usando troca de mensagens MPI em um cluster Linux/Beowulf**. Universidade Luterana do Brasil, Canoas-RS , (Trabalho de Conclusão de Curso), 2001, a ser publicado.

[GEI 94] GEIST, A. et al. **PVM: Parallel Virtual Machine**. Cambridge: MIT Press, 1994.

[ISC 85] ISCAS85 **ISCAS'85 Benchmark information**. Available from http://www.cbl.ncsu.edu/www/CBL_Docs/iscas85.html.

[LAM 01] LAM/MPI: **LAM/MPI Parallel Computing - Local Area Multicomputer**. Available from http://www.lam-mpi.org/

[LEE 59] Lee, C. **Representation of switching circuits by binary-decision programs**. Bell System Technical Journal v. 38, n.7, pp. 985-999, july 1959.

[MIL 98] MILVANG-JENSEN, K.; HU, A. J. BDDNOW: **A Parallel BDD Package**. Berlin: Springer-Verlag. (Lecture Notes in Computer Science, v.1522), pp.501-507, 1998.

[MPI 94] MPI FORUM. **The MPI message passing interface standard**. Knoxville: University of Tennessee, 1994.

[RUD 93] RUDELL, R. **Dynamic variable ordering for ordered binary decision diagrams**. International Conference on Computer-Aided Design, New York, pp. 42-47, Nov. 1993.

[SOM 01] SOMENZI, F. **CUDD: CU decision diagram package**. Available from http://vlsi.colorado.edu/~fabio.

[SPE 00] SPECTOR, D. **Building Linux clusters: Scaling Linux for scientific and enterprise applications**. New York: Oreally & Associates, 2001.

[STE 99] STERLING, T. L.; SALMON, J.; BECKER, D. J.; SAVARESE, D. F. **How to build a beowulf: A guide to the implementation and application of PC clusters**. Cambridge: MIT Press, 1999.

[STO 96] STORNETTA, T.; BREWER, F. **Implementation of an efficient BDD package**. Design Automation Conference 33, New York, pp. 641-644, june 1996.