

FORMULAÇÃO DO PROBLEMA DE COBERTURA DE FUNÇÕES BOOLEANAS ATRAVÉS DO MÉTODO DE EXPANSÃO DE SHANNON

Alexandre César Rodrigues da Silva¹, Mariângela de Carvalho Bovolato² e Fábio Ricardo Pontes Emer³

Resumo — Este artigo descreve um método para a geração de implicantes primos a partir dos quais pode-se formular o problema de cobertura de funções booleanas como um problema de PLI. O Teorema de Expansão de Shannon foi estruturado em Diagrama de Decisão Binária modificado para atender às peculiaridades do problema e implementado em linguagem C. Dessa forma, pode-se formular o problema de cobertura como um problema de programação matemática em que a função objetivo a ser minimizada, é a soma ponderada dos implicantes primos gerados e as restrições correspondem à desigualdade do tipo \geq , onde o lado esquerdo representa a soma dos implicantes primos que cobre o mintermo e o lado direito é representado pela unidade. Estudos realizados mostraram que o algoritmo implementado gerou uma menor quantidade de implicantes primos quando comparado com o tradicional método de Quine-McCluskey. Isto resulta em uma formulação menos complexa facilitando a obtenção da solução mínima.

Palavras Chave — Minimização de Funções Booleanas, Cobertura dos mintermos, Ferramenta de Síntese, Otimização

INTRODUÇÃO

A simplificação de funções booleanas é parte essencial na redução dos custos da realização de circuitos lógicos e aritméticos binários.

Muitos dos algoritmos de minimização foram baseados nas idéias de Quine [2] e McCluskey [3] que deram origem ao Método de Quine-McCluskey [4] que requer a geração de todos os implicantes primos e, então, a seleção de uma cobertura mínima, que constitui a solução mínima do problema. Apesar do procedimento de Quine-McCluskey produzir uma solução mínima exata, a complexidade computacional do algoritmo, que aumenta exponencialmente com o número de mintermos, torna-o bastante limitado. O número de implicantes primos de uma função com n variáveis pode ser tão grande quanto $3^n/n$ [5], o que torna o método bastante dispendioso em tempo e memória.

Em virtude disso, pesquisadores têm recorrido ao emprego de heurísticas, gerando uma solução inicial e, então, melhorando-a iterativamente. Os Programas Mini [8], Presto [9] e Espresso-Exact [7] são exemplos de

programas que se utilizam de técnicas heurísticas, sendo que esse último foi, por muitos anos, considerado o padrão em ferramentas para a otimização lógica a dois níveis.

Atualmente, com o avanço das estações de trabalho renovou-se o interesse pela otimização lógica exata e surgiram vários algoritmos para a geração de implicantes primos bem como para a obtenção da solução ótima. Algoritmos que utilizam DDBO (diagrama de decisão binário ordenado) para a geração de implicantes primos de forma eficaz foram amplamente empregados nos trabalhos de Meinel [11]-[12], Long [13], Thornton [14], Dreschler [15], Sauerhoff [16] e Bryant [17].

Em trabalhos recentes como o de Fallah [10], buscou-se a otimização exata aplicando-se técnica de programação linear inteira, particularmente de relaxação de programação linear (caso particular de PLI), para solucionar o problema de cobertura.

Dentro deste contexto é que este trabalho de pesquisa se insere, buscando através do método de Scheinman [6] a geração eficiente dos implicantes primos da função, simplificando, assim, a formulação do problema de cobertura, que poderá ser resolvida através de métodos matemáticos clássicos ou métodos desenvolvidos especificamente para este fim.

O PROBLEMA DE COBERTURA SOB A ÓPTICA DA PROGRAMAÇÃO INTEIRA 0 E 1

A obtenção de cobertura de uma função booleana com o menor número de termos produtos pode ser resolvida em dois passos. A geração de todos os implicantes primos (1ª fase do método de Quine-McCluskey) e a seleção de um conjunto mínimo de implicantes primos que cobre os mintermos da função (2ª fase do método de Quine-McCluskey), problema considerado não polinomial completo (N-P completo).

As regras tradicionais de cobertura consistem na busca de linhas dominantes, colunas dominantes e na remoção de linhas essenciais. Essas regras são aplicadas até que todos os mintermos estejam cobertos ou até que não possam mais ser aplicadas.

Em trabalhos como o de Silva [5] o problema de cobertura foi considerado como um problema restritamente inteiro. Dessa forma, toda cobertura mínima de uma função booleana é solução de um problema de programação linear inteira 0 e 1, ou seja, toda cobertura

¹ Alexandre César Rodrigues da Silva, Universidade Estadual Paulista, Av. Brasil, Centro, nº 56, CEP 15385-000, Ilha Solteira, SP, Brasil, acrsilva@dee.feis.unesp.br

² Mariângela de Carvalho Bovolato, Universidade Estadual Paulista, Av. Brasil, Centro, nº 56, CEP 15385-000, Ilha Solteira, SP, Brasil, mange@dee.feis.unesp.br

³ Fábio Ricardo Pontes Emer, Universidade Estadual Paulista, Av. Brasil, Centro, nº 56, CEP 15385-000, Ilha Solteira, SP, Brasil .

mínima de uma função booleana é solução do seguinte problema de programação linear inteira:

- Critério de custo: O custo de uma cobertura é igual a soma aritmética ponderada de todos os implicantes primos da função (cubos);
- Restrições: Para cada mintermo da função, a soma de todos os implicantes primos que cobrem o mintermo tem que ser maior ou igual a 1.

Dessa forma, as restrições são inequações do tipo “ \geq ” cujo lado esquerdo é a soma dos implicantes primos (cubos) que cobrem o mintermo. Tem-se tantas restrições quanto forem os mintermos da função. Adotou-se como critério de custo a quantidade de variáveis em cada termo produto (mintermo ou implicante) e a quantidade de termos produtos.

Como exemplo do problema de cobertura dos mintermos formulado como um problema de programação linear inteira 0 e 1 considera-se a função $F_1(ABC) = \sum m(0,1,5) + d(2,7)$, cujo Mapa de Karnaugh está apresentado na Figura 1.

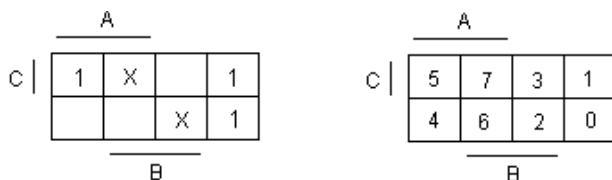


FIGURA. 1

MAPA DE KARNAUGH PARA A FUNÇÃO $F_1(ABC) = \sum m(0,1,5) + d(2,7)$.

Pela aplicação da primeira fase do Método de Quine-McCluskey são gerados os implicantes primos 00X, X01, 0X0 e 1X1, cujos custos são iguais a 3. Note que utilizou-se a notação 1X1 ou invés de AC por ser mais fácil de trabalhar em programa de computador.

A cobertura da função F_1 , formulada como um problema de programação linear inteira 0 e 1 e expressa como:

$$\text{MIN } 3 \cdot (00X + X01 + 0X0 + 1X1)$$

Sujeito às restrições:

$$00X + 0X0 \geq 1$$

$$00X + X01 \geq 1$$

$$X01 + 1X1 \geq 1$$

$$00X, X01, 0X0, 1X1 \in \{0, 1\}$$

Na primeira restrição tem-se a soma dos implicantes primos que cobrem o mintermo 0. Na segunda restrição a soma dos implicantes primos que cobrem o mintermo 1 e na terceira restrição os que cobrem o mintermo 5. Pode-se verificar que todos os mintermos estão contidos na cobertura da função e correspondem às restrições no problema de cobertura. A solução desse problema matemático é dado por $F_1(ABC) = 0X0 + X01$, cujo custo é igual a 6.

Conforme apresentado, ainda que de forma resumida, o problema de simplificar funções booleanas com n variáveis, pode ser modelado como um problema de programação linear inteira 0 e 1. Trata-se de um problema de porte muito grande. É evidente que todas as características especiais deste problema devem ser exploradas com o intuito de solucioná-lo. O Método Clássico de Quine-McCluskey pode ser visto como um método intuitivo de resolução deste problema.

A fase de geração dos implicantes primos pode ser entendida, na óptica da programação matemática, como a formulação do problema, isto é, na obtenção explícita das restrições e na resolução das equações triviais. A fase de cobertura dos mintermos corresponde, na programação matemática, à solução do problema de minimização propriamente dita. Apresenta-se na próxima seção o Teorema de Expansão de Shannon, empregado na geração de implicantes primos.

Teorema de Expansão de Shannon

A geração de implicantes primos pelo Método de Expansão de Shannon é uma técnica iterativa, que permite obter implicantes primos de uma dada função booleana através de simples operações sobre o conjunto de mintermos e/ou estados irrelevantes que descreve a função. Tal método pode ser empregado em funções com um grande número de variáveis. Todos, ou quase todos os implicantes primos que não estariam no conjunto solução são eliminados, simplificando, desse modo, a busca pela solução mínima.

O teorema diz que qualquer função booleana $f(x_1, x_2, \dots, x_n)$ pode ser expressa por: $f(x_1, x_2, \dots, x_n) = x_1 \cdot f(1, x_2, \dots, x_n) + x_1' \cdot f(0, x_2, \dots, x_n)$.

Este teorema pode ser provado por indução perfeita, ou seja, fazendo x_1 assumir o valor “1” tem-se, consequentemente, x_1' assumindo valor lógico “0”. Dessa forma, tem-se a expressão: $f(x_1, x_2, \dots, x_n) = 1 \cdot f(1, x_2, \dots, x_n)$. De modo similar, substituindo-se $x_1 = “0”$ e $x_1' = “1”$ a equação também será reduzida a uma identidade, provando, desse modo, o teorema.

Se o teorema for novamente aplicado em relação à variável x_2 , em cada um dos dois termos obtidos em relação à variável x_1 , obtém-se a seguinte expressão: $f(x_1, x_2, \dots, x_n) = x_1 x_2 \cdot f(1, 1, x_3, \dots, x_n) + x_1 x_2' \cdot f(1, 0, x_3, \dots, x_n) + x_1' x_2 \cdot f(0, 1, x_3, \dots, x_n) + x_1' x_2' \cdot f(0, 0, x_3, \dots, x_n)$.

A expressão da função $f(x_1, x_2, \dots, x_n)$ sobre o restante das variáveis produz uma forma normal disjuntiva (soma de produto). De maneira similar, a aplicação repetida do dual do Teorema de Expansão sobre as mesmas variáveis produz a forma normal conjuntiva (produto de soma).

Em termos práticos, a maneira mais rápida e simples para se obter a forma soma de produto canônica de uma função booleana é resumida a seguir:

- Examine cada termo. Se é um mintermo, mantenha-o, e continue a examinar os outros termos;

- Em cada termo produto que não é um mintermo, verifique qual a variável que está faltando. Para cada x_i que está faltando multiplique o produto por $(x_i + x_i')$;

- Multiplique todos os produtos e elimine os termos redundantes.

Do exposto pode-se concluir que a forma soma de produto canônica de qualquer função booleana é única. Essa afirmação pode ser provada assumindo que existem duas diferentes formas de soma de produto que representam uma função booleana F . Desde que se assume que as formas são diferentes, elas devem diferir em pelo menos um dos mintermos, isto é, deve existir no mínimo um conjunto de valores para as variáveis x_1, x_2, \dots, x_n , de modo que uma das formas canônicas resulta em $f(x_1, x_2, \dots, x_n) = 0$, enquanto a outra forma produz $f(x_1, x_2, \dots, x_n) = 1$, resultado este que contradiz a suposição de que ambas as formas canônicas representam a mesma função.

ALGORITMO PARA GERAÇÃO DE IMPLICANTES PRIMOS ATRAVÉS DO MÉTODO DE EXPANSÃO DE SHANNON

O Algoritmo para a obtenção dos implicantes primos de uma função booleana utilizando-se do Teorema de Expansão de Shannon foi inicialmente estudado por Scheinman [6]. Trata-se de um método manual e iterativo e de difícil entendimento. A partir dos estudos de Scheinman deu-se ao método uma abordagem através da mais avançada técnica de programação estruturada, e nas estruturas de dados utilizadas empregou-se somente apontadores e alocação dinâmica de memória.

A estrutura de dados obtida é de extrema complexidade e de difícil programação, porém, o algoritmo desenvolvido utiliza somente duas operações aritméticas, a subtração e a potenciação, e uma operação lógica, a igualdade.

O algoritmo implementado será ilustrado, para maior clareza, utilizando-se como exemplo a função de 4 variáveis, $F_2(ABCD) = \sum m(1,5,6,8,11,12,15) + d(0,3)$. Os passos do algoritmo são os seguintes:

Passo 1: Ordene em uma coluna, de forma crescente, as representações decimais dos mintermos e dos estados irrelevantes. Os estados irrelevantes devem conter uma marca e são utilizados para obter implicantes primos de dimensões maiores. Considera-se dimensão de um implicante a quantidade de variáveis irrelevantes que eles possuem. Quanto menor a quantidade de variáveis (literais), na forma negada ou não, que um implicante possui, maior será a sua dimensão. Essa primeira coluna é considerada, no algoritmo proposto, como sendo o nó raiz. A Figura 2 apresenta o nó raiz para a função F_2 .

0 •
1
3 •
5
6
8
11
12
15

FIGURA. 2

PRIMEIRO PASSO DO ALGORITMO.

Note que os estados irrelevantes contêm uma marca.

Passo 2: Divida o conjunto de termos formado no passo anterior em dois outros grupos de colunas, um rotulado com uma literal (A , por exemplo) e o outro com a mesma literal na forma negada (A' , por exemplo). Considerou-se as variáveis ordenadas como $ABCD$, sendo que os pesos são respectivamente $2^3, 2^2, 2^1$ e 2^0 . Dessa forma, a variável A é a de maior peso (peso 8). A coluna A' contém os termos (mintermos e irrelevantes) da função original, cuja representação decimal é menor do que 8 (peso da variável A). A coluna A contém os termos da função original, cuja representação decimal é maior ou igual a 8, sendo que de cada um desses números é subtraído o valor 8, valor correspondente à variável A que está sendo expandida. A Figura 3 apresenta o nó raiz subdividido em duas outras colunas.

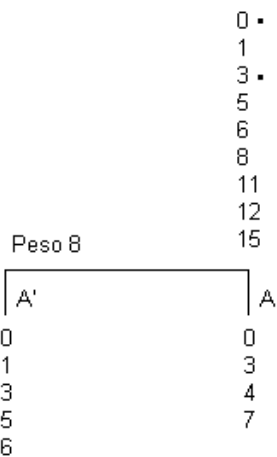


FIGURA. 3

SEGUNDO PASSO DO ALGORITMO.

Passo 3: Crie uma terceira coluna com o rótulo X . Essa coluna indicará a redundância entre A e A' . Nesta coluna, tem-se as representações decimais que são comuns tanto da coluna A como da coluna A' . Esses decimais comuns às colunas A e A' devem ser marcados nas respectivas colunas para indicar que são termos redundantes em relação à variável A . Se algum decimal contido na coluna X foi anteriormente marcado em ambas as colunas, A e A' , eles também devem ser marcados na coluna X . A Figura 4 apresenta o diagrama de decisão contendo a terceira coluna, as dos termos irrelevantes para a variável sob expansão.

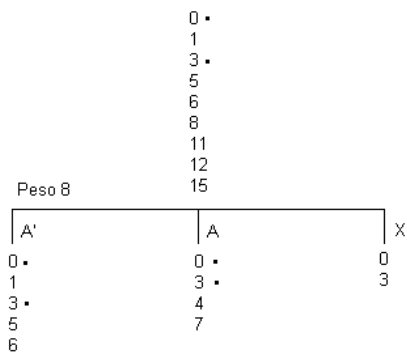


FIGURA. 4

TERCEIRO PASSO DO ALGORITMO.

Passo 4: Examine cada uma das colunas geradas. Se alguma coluna contém somente decimais marcados, toda a coluna deve ser eliminada, pois todos os seus termos estão contidos nas outras colunas;

Passo 5: Cada uma das colunas criadas no passo 3 devem ser novamente expandidas em relação à variável B, e repete-se os passos 2, 3 e 4. Se um decimal da coluna B for resultado de um decimal já marcado o seu correspondente também deverá ser marcado. No exemplo é o caso do decimal 3 da coluna B' do nó X e do decimal 1 da coluna C do nó B', sub nó de X. Quando a função é expandida em relação à variável final (a de menor peso), que no algoritmo é denominado de folhas, o resíduo deve ser 0. Os passos 2, 3, 4 e 5 devem ser repetidos para todas as variáveis até se obter resíduo igual a 0. A Figura 5 apresenta a árvore completa para a função F_2 .

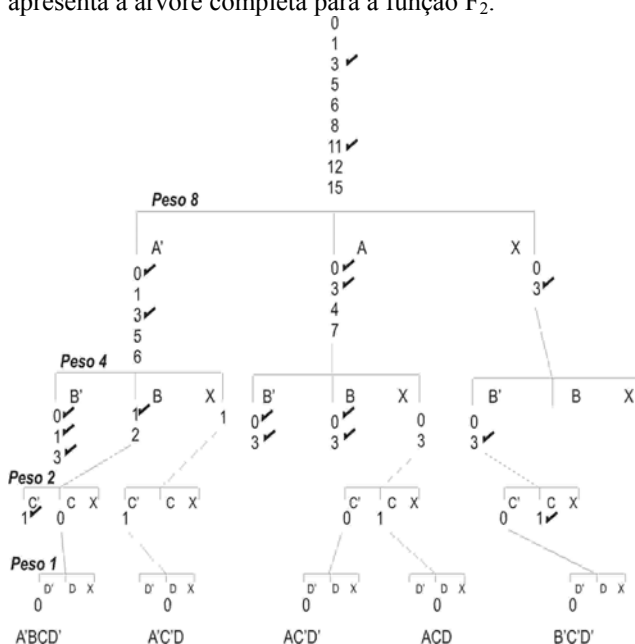


FIGURA. 5
ÁRVORE COMPLETA PARA A FUNÇÃO F_2 .

Passo 6: Obtenha os implicantes primos da função traçando os caminhos que vão das folhas (variável de menor peso) até a raiz (variável de maior peso). Os rótulos dos ramos contidos no caminho representam os implicantes primos gerados pela aplicação do algoritmo.

Pela Figura 5 pode-se notar que foram gerados os seguintes implicantes primos: 0110, 0X01, 1X00, 1X11 e X000. Um aspecto muito importante a ser salientado é que o algoritmo Expander obtém todos os implicantes primos essenciais e que alguns implicantes primos que não estariam no conjunto solução são automaticamente eliminados no processo de construção do Diagrama de Decisão.

RESULTADOS OBTIDOS

Para testar a eficiência do algoritmo Expander, implementado em linguagem C++, foram avaliadas dezenas de funções booleanas, cujos resultados foram comparados

com os obtidos com o tradicional Método de Quine-McCluskey. Escolheu-se o Método de Quine-McCluskey pois na primeira fase do método, a geração de implicantes, todos os implicantes primos são gerados. Os parâmetros utilizados para a comparação foram a quantidade de memória utilizada, o tempo de execução e a quantidade de implicantes primos gerados.

A Figura 6 apresenta a quantidade de memória utilizada pelo Expander subtraída da quantidade de memória utilizada pelo Quine-McCluskey. O Sinal (-) foi utilizado somente para indicar que na grande maioria dos casos o Expander utilizou menos memória.

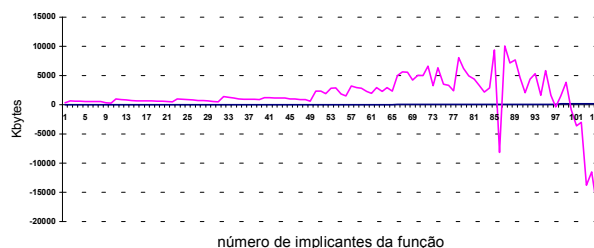


FIGURA. 6
QUANTIDADE DE MEMÓRIA UTILIZADA PELO EXPANDER E PELO QUINE-MCCLUSKEY.

A Figura 7 apresenta o tempo utilizado pelos dois algoritmos para a geração dos implicantes primos. Pode-se notar que o algoritmo Expander sempre obteve a solução em menor tempo.

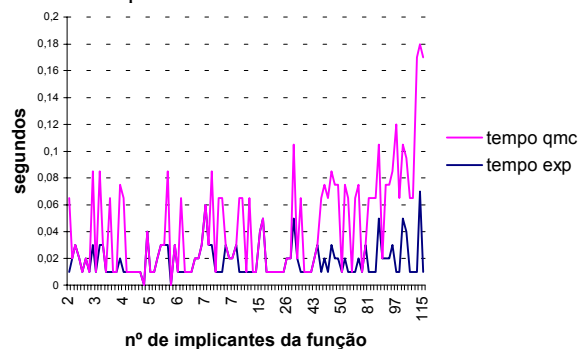


FIGURA. 7
TEMPO UTILIZADO PELO EXPANDER E PELO QUINE-MCCLUSKEY.

A Figura 8 apresenta um gráfico comparando a quantidade de implicantes primos gerados pelos dois programas.

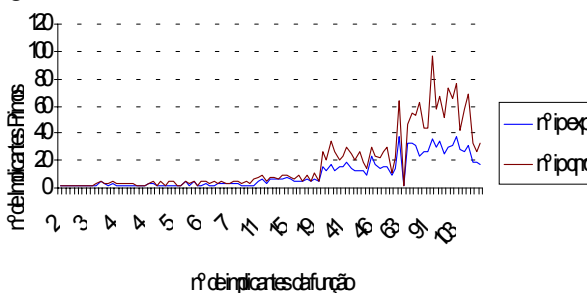


FIGURA. 8
QUANTIDADE DE IMPLICANTES PRIMOS GERADA PELO EXPANDER E PELO QUINE-MCCLUSKEY.

Pode-se perceber pela Figura 8 que o número de implicantes primos gerado pelo Expander é igual ao gerado pelo Quine-McCluskey para funções com uma pequena quantidade de mintermos. À medida que se aumenta o número de mintermos, o programa Expander supera o Quine-McCluskey, pois alguns implicantes primos são eliminados, simplificando, dessa forma, a formulação do problema de cobertura.

CONCLUSÃO

Este trabalho de pesquisa teve como principal objetivo a busca de métodos eficientes para a geração de implicantes primos de uma função booleana. De posse dos implicantes primos gerados, a cobertura dos mintermos pode ser formulada como um problema de programação linear inteira 0 e 1. Implementou-se em programa de computador, uma técnica iterativa e manual, inicialmente proposta por Scheinman, que gera um conjunto reduzido de implicantes primos, porém o suficiente para se obter a solução de custo mínimo.

O algoritmo implementado, denominado Expander, foi comparado com o método de Quine-McCluskey com o objetivo de avaliar a eficiência dos algoritmos. Utilizou-se como parâmetros o consumo de memória, o tempo de execução e a quantidade de implicantes primos gerados.

Constatou-se que o problema de cobertura dos mintermos formulado a partir dos implicantes primos gerados pelo Expander é bem mais simples do que a formulação a partir dos implicantes primos gerados pelo Quine-McCluskey.

O Expander consumiu menor quantidade de memória e gerou os implicantes primos em menor tempo.

Dessa forma, implementou-se um algoritmo que é uma alternativa eficiente para a obtenção de implicantes primos de uma função booleana, a partir dos quais o problema de cobertura pode ser formulado como um problema de programação linear inteira 0 e 1.

AGRADECIMENTOS

Os autores agradecem à Fundunesp e o Programa de Pós-graduação em Engenharia Elétrica da Faculdade de Engenharia de Ilha Solteira – UNESP.

REFERÊNCIAS

- [1] Shannon, C. E. "The synthesis of two-terminal switching circuits", *Bell Sys. Tech. J.*, 1948.
- [2] Quine, W., "The Problem of Simplifying Truth Functions", *American Mathematical Monthly*, Vol. 59, pp. 521-531, 1952.
- [3] McCluskey, E., "Minimization of Boolean Function", *The Bell System Technical Journal*, Vol. 35, pp. 1417-1444, November, 1956.
- [4] McCluskey E. J., *Logic Design Principles*, Prentice Hall, N.J., 1986.
- [5] Silva, A.C.R., *Contribuição à Síntese de Circuitos Digitais Utilizando Programação Linear Inteira 0 e 1*, Tese de Doutorado, UNICAMP, 1993.
- [6] Scheinman, A. H., "A Method for Simplifying Boolean Functions", *The Bell System Technical Journal*, July, 1962, pp. 1337-1346.
- [7] Brayton, K. B., Hachtel, G. D., McMullen, C. T., Sangiovanni-Vicentelli A. L., "Logic Minimization Algorithms for VLSI Synthesis", *Kluwer Academic Publisher*, 1984.
- [8] Hong, S.J. Cain, R. G. and Ostapko, D.L. "MINI: A heuristic Approach for logic minimization," *IBM J of Res. And Dev.*, Vol. 18, pp. 443-458, September 1974.
- [9] Brown, D. W. "A State-Machine Synthesizer – SMS" *Proc. 18th Design Automation Conference*. pp 301-304 Nashville, June 1981
- [10] Fallah, F. Liao, S and Devadas, S. "Solving covering Problems Using LPR-Based Lower Bounds", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. Vol. 8. February 2000.
- [11] Meinel, C.; Slobodova, A. "Unifying theoretical background for some BDD-based data structures", *Formal Methods in System Design*, October, 1997, v11 i3, p. 223-237.
- [12] Meinel, C.; Slobodova A., "Speeding up variable reordering of OBDDs", *Proceedings - IEEE International Conference on Computer Design: VLSI in Computers and Processor*, 1997, p. 338-343.
- [13] Long, W.; MIN, Yinghua; Y., Shiyuan; TONG, S., "Short-time scaling of variable ordering of OBDDs", *Journal of Computer Science and Technology*, July, 1997, v12 i4, p. 366-371.
- [14] Thornton, M.A., "Modified Haar transform calculation using digital circuit output probabilities", *Proceedings of the International Conference on Information, Communications and Signal Processing, ICICS*, 1997, v1, p. 52-58.
- [15] Drechsler, R.; Becker, B.; Goeckel, N., "Genetic algorithm for variable ordering of OBDDs", *IEE Proceedings: Computers and Digital Techniques*, November, 1996, v 143 i6, p. 364-368.
- [16] Sauerhoff, M.; Wegener, I., "On the complexity of minimizing the OBDD size for incompletely specified functions", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, November, 1996, v 15, i11, p. 1435-1437.
- [17] Bryant, R., "Symbolic Boolean manipulation with Ordered Binary-Decision Diagrams", *ACM Computing Surveys*, September, 1992, v 24 i3, p. 293-318.