APERFEIÇOAMENTO DO MÉTODO CLAUSE-COLUMN TABLE PARA A GERAÇÃO EFICIENTE DE IMPLICANTES PRIMOS

IMPROVEMENT OF CLAUSE-COLUMN TABLE METHOD FOR EFFICIENT GENERATION OF PRIME IMPLICANTS

Caroline Domingues Porto do Nascimento Barbieri[1], Jean Vitor de Paulo[2], Alexandre César Rodrigues da Silva[3]

Abstract *– Efficient generation of prime implicants is an important factor in the coverage phase of minterms in boolean functions minimization methods. This paper presents an improved version of the method called Clause-Column Table, used to generate prime implicants, which also facilitates learning in the Boolean function simplification process. The original and improved algorithms were implemented in C language and the results confirmed that the improved version is better at primes implicant's generation. To obtain the minimum solution the coverage problem is formulated as an integer linear programming problem of 0 and 1, in which the objective function and the restrictions are composed by the generated prime implicants.*

Index Terms – Words: Boolean function. Prime implicants. Minimization.

INTRODUÇÃO

A utilização de circuitos digitais tem sido amplamente disseminada com o avanço da tecnologia e principalmente dos dispositivos eletrônicos digitais.

A convergência deste avanço resultou em um grande desafio. É preciso lidar com circuitos digitais cada vez mais complexos. Além disso, é necessário fazer com que o tempo e o custo de implementação sejam sempre decrescentes. A complexidade está no fato de que cada componente utilizado para implementar o circuito, ocupa um determinado espaço físico para produzi-lo. Assim, diversos pesquisadores têm recorrido à otimização lógica, que é uma forma de obter circuitos minimizados e com a mesma funcionalidade.

Encontram-se na literatura diversos métodos de minimização de funções booleanas, considerados exatos, que geram todos os implicantes primos para então obter uma solução ótima, sendo que a maioria foi desenvolvida baseada na forma tabular de representação de funções booleanas [1, 2, 3, 4].

A minimização exata requer muitos recursos computacionais, visto que a complexidade do algoritmo aumenta exponencialmente conforme o número de variáveis de entrada [5]. A fim de contornar tal situação, pesquisadores recorrem a métodos heurísticos, que geram soluções aproximadamente mínimas num tempo de execução aceitável, pois não há necessidade de buscá-las exaustivamente até encontrar uma solução ótima global [5, 6, 7, 8, 9, 10].

É neste cenário de minimização de funções booleanas que o presente trabalho se insere, propondo uma melhoria no método desenvolvido em [11], adicionando ao algoritmo original o teorema da adjacência e um novo critério de parada. Estas modificações evitaram a geração de termos nulos e iterações desnecessárias que ocorriam no algoritmo original.

Os algoritmos foram implementados em linguagem C e os resultados comprovaram que a versão aprimorada é mais eficiente que a original.

MÉTODO DE MINIMIZAÇÃO CLAUSE-COLUMN TABLE APRIMORADO

O método Clause-Column Table utiliza-se de uma técnica tabular e iterativa que permite a geração de implicantes primos de uma determinada função booleana especificada na forma canônica ou não, na forma normal de soma de produto ou de produto de soma. O resultado pode ser rapidamente obtido mesmo no caso de funções com muitos termos e variáveis.

Com a finalidade de sanar algumas deficiências encontradas no método Clause-Column Table, utilizado para a geração de implicantes primos, desenvolveu-se uma nova versão denominada de Clause-Column Table Aprimorado. Neste novo algoritmo foram feitas algumas modificações necessárias para evitar a geração de termos nulos e diminuir a quantidade de iterações utilizadas na geração dos implicantes primos. Os termos denominados de nulos originam soluções infactíveis, porém são gerados pelo método original como se fossem implicantes primos.

Para reduzir o número de iterações adicionou-se a propriedade algébrica $AB + A\overline{B} = A$ pertencente à álgebra

[1] Caroline Domingues Porto do Nascimento Barbieri – PhD Student - Sao Paulo State University – Department of Electrical Engineering - carolbarbieri1@gmail.com
[2] Jean Vitor de Paulo - PhD Student - Sao Paulo State University – Department of Electrical Engineering - jeanvitordepaulo@gmail.com
[3] Alexandre Cesar Rodrigues da Silva - Professor – Sao Paulo State University – Department of Electrical Engineering - acrsilva@dee.feis.unesp.br
DOI 10.14684/ICECE.9.2015.132-136
© 2015 COPEC

booleana. Ao aplicá-la nas colunas referentes à tabela inicial formada pelos maxtermos da função booleana vários termos são eliminados.

O método original gera na fase inicial a tabela ilustrada na Figura 1, em que as colunas rotuladas por M_1, M_2, M_3 e M_4 representam maxtermos da função. Nota-se que as colunas M_1 e M_4, podem ser simplificadas algebricamente de acordo com o teorema da adjacência. O mesmo ocorre para as colunas M_2 e M_3. O método original não realiza tal simplificação.

M1	M2	M3	M4
$\bar{A}$	A	A	$\bar{A}$
$\bar{B}$	$\bar{B}$	B	$\bar{B}$
$\bar{C}$	C	$\bar{C}$	C

↓

M1	M2
$\bar{A}$	A
$\bar{B}$	C

FIGURA 1
Simplificação algébrica aplicada nas colunas da tabela inicial do algoritmo.

Além do emprego do teorema da adjacência incluiu-se no algoritmo original um novo critério de parada com a finalidade de evitar a geração de termos nulos: se restar apenas uma coluna a partir da segunda iteração o algoritmo deve ser interrompido.

ALGORITMO PARA A GERAÇÃO DE IMPLICANTES PRIMOS ATRAVÉS DO MÉTODO CLAUSE-COLUMN TABLE APRIMORADO

A contribuição deste artigo consiste na definição de um critério de parada e o teorema da adjacência inexistente no algoritmo original conforme já salientado na seção anterior.

O algoritmo Clause-Column Table Aprimorado é composto pelos seguintes passos:

1º Passo: Forme uma tabela em que cada coluna é composta por um maxtermo da função. Se a função booleana inicial estiver na forma de produto de soma, execute o próximo passo, senão obtenha o seu dual.

2º Passo: Execute as seguintes ações iterativamente até que todas as colunas sejam excluídas, caso contrário o 3º passo deve ser executado:

a) Se algum critério de parada do 6º passo é atendido, interrompa o algoritmo, caso contrário continue;
b) Se houver colunas dominadas, elimine-as;
c) Se existir colunas redundantes elimine aleatoriamente uma delas;
d) Simplifique as colunas restantes de acordo com o teorema de adjacência (apenas uma vez em cada iteração);
e) Se houver literal essencial, selecione-o. Se houver mais de um literal essencial selecione um deles aleatoriamente;
f) Verifique novamente critério de parada 6 –c;
g) Elimine todas as colunas que contém literais essenciais;
h) Elimine o complemento do literal essencial selecionado das colunas restantes.

3º Passo: Se houver um literal essencial selecionado no passo anterior execute o 5º passo, caso contrário execute o 4º passo;

4º Passo: Verifique na tabela reduzida a quantidade de incidência de cada um dos literais na forma complementada ou não;

a) Se houver literais com a mesma incidência selecione um deles aleatoriamente;
b) Se o literal com a maior incidência estiver contido em todas as colunas ele deve ser considerado implicante primo da iteração atual;
c) Execute o 6º passo.

5º Passo:

a) Todas as colunas que contiver o literal selecionado devem ser eliminadas;
b) O complemento do literal selecionado também deve ser eliminado das demais colunas;
c) Forme a tabela reduzida com o restante das colunas;
d) Se na tabela reduzida houver mais de uma coluna selecione o literal essencial, senão escolha aleatoriamente o de maior incidência e repita os passos 5-a) e 5-b).
e) Realize a operação lógica AND entre o literal selecionado e a tabela reduzida, cujos termos resultantes são implicantes primos.

6º Passo: Verifique se algum critério de parada é atendido.

a) Todas as colunas sejam excluídas;
b) Dois literais, complementados ou não, tornem-se essenciais ao mesmo tempo;
c) Um literal torne-se essencial e na iteração seguinte o seu complemento também;
d) Reste apenas uma coluna a partir da segunda iteração.

7º Passo: Uma vez atingido algum critério de parada reúna os implicantes primos de cada iteração, eliminando os termos redundantes e forme o conjunto de solução.

O algoritmo deve ser executado iterativamente até que uns dos critérios de parada sejam atendidos. Para apresenta-lo utiliza-se como exemplo a função de 4 variáveis, $F(A,B,C,D) \quad ||M(0,1,2,3,8,9,10,11)$.

1º Passo: Forme uma tabela em que cada coluna é composta por um maxtermo. Se a função booleana inicial estiver na forma de produto de soma, execute o próximo passo, senão obtenha o seu dual.

$$F(A,B,C,D) \quad ||M(0,1,2,3,8,9,10,11)$$

TABELA I
Tabela inicial do método Clause-Column Table Aprimorado.

M0	M1	M2	M3	M8	M9	M10	M11
A	A	A	A	$\bar{A}$	$\bar{A}$	$\bar{A}$	$\bar{A}$
B	B	B	B	B	B	B	B
C	C	$\bar{C}$	$\bar{C}$	C	C	$\bar{C}$	$\bar{C}$
D	$\bar{D}$	D	$\bar{D}$	D	$\bar{D}$	D	$\bar{D}$

2º Passo: Analisando a Tabela I, nota-se que o passo 2- d (Simplifique as colunas restantes de acordo com o teorema de adjacência) deve ser executado. O teorema de adjacência aplica-se nas colunas M0 e M1, M2 e M3, M8 e M9, M10 e M11. Em seguida tem-se a Tabela II que foi reduzida formando as colunas M1, M3, M9 e M10.

TABELA II
Tabela do método Clause-Column Table Aprimorado após a simplificação algébrica.

M1	M3	M9	M11
A	A	$\bar{A}$	$\bar{A}$
B	B	B	B
C	$\bar{C}$	C	$\bar{C}$

3º Passo: No passo anterior não houve nenhum literal essencial selecionado. Execute o 4º passo.

4º Passo: Na Tabela II pode-se verificar que o literal B é o que mais ocorre e está contido em todas as colunas da tabela, portanto é considerado o implicante primo da iteração atual.

5º Passo: Como o literal B está contido em todas as colunas, não existe colunas restantes para formar a tabela reduzida. Portanto, execute o 6º Passo.

6º Passo: Nenhum critério de parada é atendido. Execute o 2º passo novamente.

2ª Iteração:

2º Passo: Como o literal B foi o primeiro selecionado na iteração anterior ele é excluído permanentemente, como pode-se observar na Tabela III.

TABELA III
Tabela referente à segunda iteração do método Clause-Column Table Aprimorado.

M 1	M 3	M 9	M 11
A	A	$\bar{A}$	$\bar{A}$
C	$\bar{C}$	C	C

Nota-se na Tabela III que o teorema de adjacência aplica-se nas colunas M1 e M3, M9 e M11, formando a tabela V que reduziu-se, composta agora pelas colunas M3 e M11. Posteriormente o critério de parada 6-b é atendido (Um literal e seu complemento tornem-se essenciais ao mesmo tempo).

Tabela IV
Tabela após a simplificação algébrica.

M3	M11
A	$\bar{A}$

7º Passo: Uma vez atingido o critério de parada, deve-se reunir todos os termos obtidos no 5º passo, pois representam implicantes primos.

Diante do apresentado nota-se através da Figura 2 que o método Clause-Column Table Aprimorado gerou somente o implicante primo essencial B, ao contrário do método original que minimizando a mesma função gerou o conjunto de implicantes primos B+ACD+AC cujos termos ACD + AC são nulos e, portanto invalidam o resultado, como é ilustrado na Figura 3. Constata-se então que a simplificação algébrica, além do novo critério de parada, evitou a geração de termos nulos e reduziu o número de iterações.

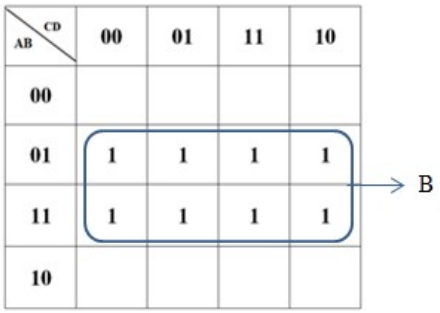

FIGURA 2
Cobertura dos mintermos gerada pelo método Clause-Column Table Aprimorado.

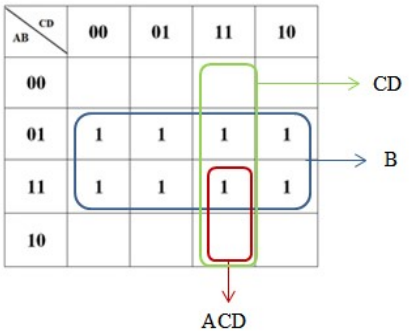

FIGURA 3
Cobertura dos mintermos gerada pelo método Clause-Column Table.

Após a geração dos implicantes primos deve-se realizar a fase de cobertura para encontrar os implicantes primos essenciais que realizam a cobertura mínima da função. Esta cobertura foi formulada com sendo um problema de programação linear inteira 0 e 1.

COBERTURA DE FUNÇÕES BOOLEANAS ATRAVÉS DE PROGRAMAÇÃO INTEIRA 0 E 1.

A obtenção de uma cobertura mínima de uma função booleana com o menor número de termos produtos pode ser resolvida em duas etapas. A primeira consiste em gerar todos os implicantes primos da função e a segunda consiste em selecionar um conjunto mínimo de implicantes, que é denominado de problema de cobertura, que pode ser expresso como um problema de programação linear restritamente inteiro (SILVA, 1993).

A partir dos implicantes primos obtidos formula-se a cobertura da função como sendo um problema de programação linear inteira 0 e 1. As variáveis do problema são os implicantes primos e as restrições são inequações de "≥ 1", em que o lado esquerdo é a soma dos implicantes primos que cobrem cada mintermo.

Com tal característica, o número de implicantes primos é igual o de variáveis do problema e o número de mintermos é igual o de restrições.

Minimizando a função booleana F(A, B, C) = $\sum m\,1, 2, 3, 4, 5, 6$, através do método Clause-Column Table Aprimorado, gerou-se os implicantes primos: $\bar{A}\bar{B} + \bar{A}\bar{C} + \bar{A}B + B\bar{C} + \bar{A}C + \bar{B}C$.

Nota-se na figura 4 que todos os mintermos estão inclusos na cobertura da função e correspondem às restrições no problema de cobertura, cuja solução é dada por $\bar{A}\bar{B} + B\bar{C} + \bar{A}C$ e o custo é igual a 9.

O critério de custo empregado é a soma aritmética ponderada de todos os implicantes da função. Cabe salientar que o custo inicial era 18, posteriormente a formulação e solução do problema de cobertura o custo foi reduzido pela metade.

Considerando:

$a = \bar{A}\bar{B}$, $b = \bar{A}\bar{C}$, $c = \bar{A}B$, $d = B\bar{C}$, $e = \bar{A}C$, $f = \bar{B}C$

têm-se o problema de programação linear:

$$\text{MIN } 3.(a + b + c + d + e + f)$$
$$a + b \geq 1$$
$$c + e \geq 1$$
$$c + d \geq 1$$
$$a + f \geq 1$$
$$e + f \geq 1$$
$$b + d \geq 1$$
$$a, b, c, d, e, f \in \{0, 1\}$$

FIGURA 4
Formulação do problema de cobertura dos mintermos.

RESULTADOS OBTIDOS

Com objetivo de avaliar a eficiência da geração de implicantes primos através do método Clause-Column Table Aprimorado foram utilizadas funções booleanas que foram comparadas com o método original de Das.

Os métodos Clause-Column Table original e Aprimorado geraram implicantes primos para função com até 60 variáveis de entrada. Entretanto, para comparar os resultados foram utilizadas funções com até 12 variáveis de entrada.

Na Figura 5 apresenta-se um gráfico comparando a quantidade de iterações utilizadas na geração de implicantes primos pelos dois métodos. Pode-se perceber que o número de iterações utilizadas pelo Clause-Column Table Aprimorado é menor do que as utilizadas pelo método original. À medida que aumenta-se o número de variáveis, o método Aprimorado torna-se significativamente mais vantajoso, pois o número de iterações é ainda mais reduzido com a inclusão do teorema da adjacência

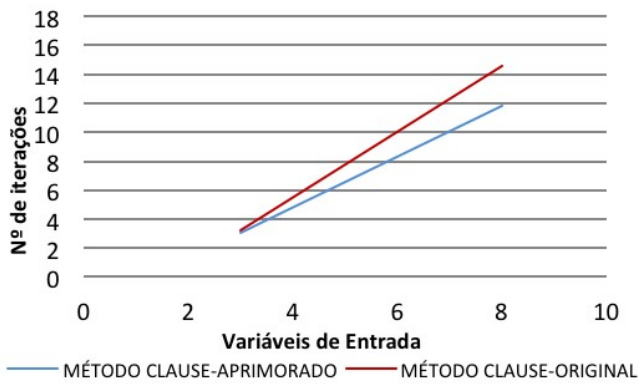

FIGURA 5
Gráfico comparativo no número de iterações utilizadas pelos métodos.

Na Figura 6 apresenta-se um gráfico, comparando o número de implicantes primos gerados pelos métodos Aprimorado e Original. É possível perceber claramente que os mesmos apresentam comportamentos semelhantes, pois a diferença está no fato de que o método aprimorado gera menos termos nulos, que são considerados implicantes primos pelo método original.

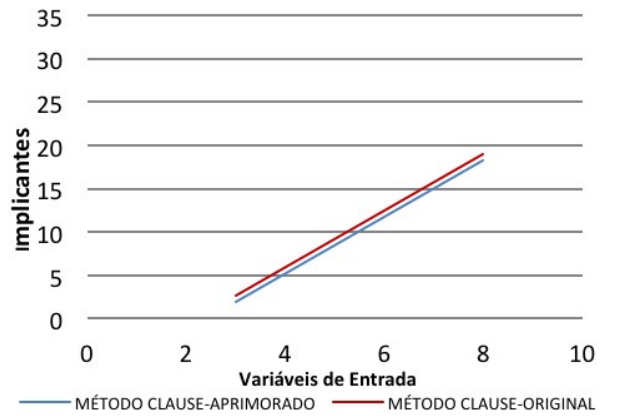

FIGURA 6
Gráfico comparativo de quantidade de implicantes gerados pelos dois métodos.

CONCLUSÃO

Este trabalho teve como principal finalidade encontrar métodos eficientes para a geração de implicantes primos no contexto de minimização de funções booleanas. Dentre os métodos encontrados na literatura, o método de Das denominado Clause-Column Table foi estudado e posteriormente aprimorado, eliminando algumas deficiências encontradas no mesmo, originando assim uma nova versão chamada de Clause-Column Table Aprimorado.

Os métodos foram implementados em linguagem de programação C, através de uma técnica tabular e iterativa, que gera um conjunto reduzido de implicantes primos suficientes para obter a solução de custo mínimo.

Na comparação do método Clause-Column Table com o método Aprimorado utilizou-se como parâmetro, a quantidade de iterações e o número de implicantes primos gerados.

Pode-se constatar que o número de iterações geradas através do método aprimorado são inferiores as geradas pelo método original de Das. Constatou-se também que o método Clause-Column Table Aprimorado evitou a geração dos termos nulos na maioria dos casos analisados e diminuiu o número de implicantes primos gerados.

O problema de cobertura foi formulado como um problema de programação linear inteira 0 e 1.

Como continuação deste trabalho o método Clause-Column Table Aprimorado pode ser expandido para simplificar funções booleanas com múltiplas saídas.

REFERÊNCIAS

[1] McCLUSKEY, E. Minimization of boolean function. **The Bell System Techinal Journal**, Lugar de Publicação, v. 35, p. 1417-1444, 1956.

[2] SAFAEI, Javad; BEIGY, Hamid. Quine-McCluskey classification. In: INTERNATIONAL CONFERENCE ON COMPUTER SYSTEMS AND APPLICATIONS- AICCSA, 7., 2007. Beaumon: Ieee/acs, p. 404-411. 2007.

[3] JAIN, T. K.; KUSHWAHA, D. S.; MISRA, A. K. Optimization of the quine-mccluskey method for the minimization of the Boolean expression. In: INTERNATIONAL CONFERENCE ON AUTONOMIC AND AUTONOMOUS SYSTEMS, 4., 2008, Lugar de Realização do evento. p. 165-168.2008.

[4] RATHORE, T.S.; JAIN, A., "A systematic map method for realizing minimal logic functions of arbitrary number of variables," CIRCUITS, SYSTEMS, COMMUNICATION AND INFORMATION TECHNOLOGY APPLICATIONS (CSCITA), 2014 INTERNATIONAL CONFERENCE ON. Mumbai, Índia. p. 81-86, 4-5. Abril 2014.

[5] SILVA, Alexandre Cesar Rodrigues da. **Contribuição à síntese de circuitos digitais utilizando programação linear inteira 0 e 1.** 1993. 119 f. Tese (Doutorado)- Curso de Engenharia Elétrica, Universidade Estadual de Campinas, Campinas, 1993.

[6] LIN, Yuan; GENG, Ruiping. MLBM: machine-learning-based minimization algorithm for boolean functions. In: IEEE INTERNATIONAL SYMPOSIUM ON, 9, 2009, Seoul. Título. Seoul: Local de edição, 2009. p. 1160-1165.

[7] AMARU, Luca et al. A Novel Data-Structure Algorithms for Efficient Logic Optimization. Dac '14 proceedings of the the 51st annual design automation conference on design automation conference, San Francisco, Usa, v. 1, p.1-6, jun. 2014.

[8] BRAYTON, R et al. Logic minimization algorithms for VLSI synthesis. Higham: Kluwer Academic, 1984.

[9] HONG, S J et al. A Heuristic Approach for Logic Minimization. Ibm Journal Of Research And Development. New York, Usa, p. 443-458. set. 1974.

[10] J.HLAVICKA.; P. FISER. A Heuristc method of two-level logic synthesis. In: WORLD MULTICONFERENCE ON SYSTEMICS, CYBERNETICS AND INFORMATICS- SCI, 5., 2001, Orlando. v. 12, p 283-288. 2001.

[11] DAS, S. R.; KHABRA, N. S. Clause-column table approach for generating all the prime implicants of switching functions. IEEE Computer Society. Washington, USA,v. 21, n 11, p 1239-1246, nov. 1972.