DINCON 2017
CONFERÊNCIA BRASILEIRA DE DINÂMICA, CONTROLE E APLICAÇÕES
30 de outubro a 01 de novembro de 2017 – São José do Rio Preto/SP

Emprego do método de Quine-McCluskey Estendido para gerar circuito com estruturas *ESOP* (XOR e XNOR)

Aline de Paula Sanches[1]
Departamento de Engenharia Elétrica, UNESP, Ilha Solteira, SP
Alexandre Araujo Amaral de Almeida[2]
Departamento de Engenharia Elétrica, UNESP, Ilha Solteira, SP
Edson Donizete de Carvalho[3]
Departamento de Matemática, UNESP, Ilha Solteira, SP
Alexandre César Rodrigues da Silva[4]
Departamento de Engenharia Elétrica, UNESP, Ilha Solteira, SP

Resumo Neste trabalho apresenta-se a implementação da primeira fase do método Quine-McCluskey Estendido que utiliza-se de estruturas AND-XOR-XNOR, com o objetivo de comparar os resultados obtidos com a primeira fase do método de Quine-McCluskey que utiliza-se de estruturas AND-OR, para a geração de implicantes primos. A fase de cobertura dos mintermos foi formulada como um problema de programação linear inteira 0 e 1. Na comparação da eficiência dos métodos foram analisados os custos, consumo de memória e o tempo de execução. Com os resultados obtidos pode-se concluir que, para a maioria dos casos executados, o Quine-McCluskey Estendido gera uma solução de menor custo. No entanto, com relação ao desempenho computacional (tempo de execução e memória), o método implementado apresentou-se menos eficiente se comparado ao Quine-McCluskey.

Palavras-chave. Geração de implicantes primos, Método de Quine-McCluskey, Expressões AND-XOR-XNOR.

1 Introdução

Em decorrência dos crescentes desafios em se projetar circuitos integrados (CIs) contendo cada vez mais dispositivos eletrônicos e o advento das novas tecnologias de fabricação

[1]alinepaulasanches@gmail.com
[2]aamaralalmeida@gmail.com
[3]edson@mat.feis.unesp.br
[4]acrsilva@dee.feis.unesp.br

como, por exemplo, *quantum-dot cellular automata*(QCA), DNA, *single electron tunneling*, *tunneling-phase logic* [8] dentre outros, a busca por métodos de minimização de funções booleanas tornou-se incessante.

Segundo Sasao [5], na maioria das vezes, a implementação da expressão AND-XOR requer menos termos produtos se comparado com a implementação da expressão AND-OR. Uma das vantagens dos circuitos com expressão AND-XOR se refere à testabilidade, ou seja, este tipo de estrutura é mais simples para ser testada [6].

Um método descrito na literatura que despertou interesse, devido as suas características, foi o Método de Quine-McCluskey Estendido [7]. Da mesma forma que o Método de Quine-McCluskey [4], trata-se de um método tabular composto por duas fases. A primeira fase é denominada de geração dos implicantes primos e a segunda fase é denominada de cobertura dos mintermos.

Neste trabalho implementou-se, em programa de computador, a primeira fase do método de Quine-McCluskey Estendido com o objetivo de compará-lo com os resultados obtidos pela primeira fase do método de Quine-McCluskey. A fase de cobertura dos mintermos foi tratada como um problema de programação linear inteira 0 e 1 [2]. Para efeito de comparação foram avaliados os custos dos circuitos mínimos gerados, a quantidade de memória utilizada e o tempo de execução.

2 Método de Quine-McCluskey Estendido

O método de Quine-McCluskey Estendido esta baseado no algoritmo empregado no método de Quine-McCluskey. Entretanto, além das portas OR e AND, gera-se expressões com portas XOR e XNOR no processo de minimização. A utilização de portas lógicas XOR e XNOR possibilita a redução no custo, na maioria das vezes [7].

Na representação do método de Quine-McCluskey, os mintermos e *don't care states* são representados na forma binária e as portas lógicas utilizadas são AND e OR. No método de Quine-McCluskey Estendido utiliza-se operador XOR e XNOR, cada dígito binário é representado na forma de potência, em que 1 é representado pela potência não barrada e 0 pela potência barrada. Como exemplo, considere o mintermo 5, sendo 0101 (representado na forma binária convencional) e $\overline{8}.4.\overline{2}.1$ (representado na forma de potência).

Como o objetivo do método é minimizar funções booleanas, ou seja, reduzir o custo do circuito, é necessário definir o critério de custo. De acordo com o apresentado por Silva [1], alguns critérios de custo que podem ser utilizados são:

- Menor quantidade de variáveis na forma complementada ou não;
- Menor quantidade de variáveis na forma soma de produtos ou produto de somas;
- Menor quantidade de termos em uma expressão.

Neste trabalho adotou-se como critério de custo a menor quantidade de variáveis na forma soma de produto ou produto de somas.

2.1 Descrição do Algoritmo de Quine-McCluskey Estendido

Os passos do algoritmo são descritos a seguir:

1. Representar os mintermos e *don't care states* em uma coluna na forma de potência;

2. Separar os mintermos em caixas de acordo com o número de 1's (as potências não barradas indicam a quantidade de 1's que possuí cada elemento);

3. Execute os seguintes passos:

   a. Comparar todos os elementos da própria caixa com os elementos das duas caixas subsequentes. Os elementos devem se diferenciar em uma ou duas variáveis não complementadas. Marcar com * os elementos combinados. Dois elementos são combinados se forem coincidentes ou diferirem em um ou dois bits. Substituir as posições que diferem pela soma das potências dos bits correspondentes. Se dois termos diferem em apenas 1 bit, a posição correspondente deve ser substituída por X;

   b. Ao executar o passo 3a, analisar também os elementos que foram combinados e aplicar as operações apresentadas na Tabela 1, salientando que a quantidade de potências não barradas de um termo para o outro indicará o tipo de operação a ser realizada;

Tabela 1: Tipos de operadores utilizados na combinação das caixas da função.

Diferença	**Operadores**	**Caixa**
0	Exclusive-OR	Caixa 1 - Caixa 1
1	AND	Caixa 1 - Caixa 2
2	NOT Exclusive-OR	Caixa 1 - Caixa 3

4. Repetir o passo 3 com as novas caixas geradas, até que nenhuma combinação seja possível;

5. Listar os implicantes primos. Os implicantes primos são aqueles que não foram marcados por *.

Considere como exemplo a aplicação do método de Quine-McCluskey Estendido na função booleana $F_1(A, B, C) = \sum m(1, 3, 4, 6, 7)$. Na aplicação do algoritmo no caso sob estudo gera-se a Tabela 2 e a Tabela 3.

Como na Tabela 3, alguns elementos não foram combinados, portanto não foram marcados, logo o conjunto de implicantes primos da função $F_1(A, B, C) = \sum m(1, 3, 4, 6, 7)$, são: $\overline{6}.\overline{6}.1 + 4.\overline{3}.\overline{3} + X.2.1 + 4.2.X + 5.X.5$ que corresponde à expressão: $(\overline{A \oplus B})C + A(\overline{B \oplus C}) + BC + AB + (A \oplus C)$.

A partir dos implicantes primos obtidos na primeira fase do método de Quine-McCluskey Estendido, deve-se realizar a fase de cobertura, para obter uma solução mínima. Formula-se então, a cobertura dos mintermos como sendo um problema de programação linear inteira 0 e 1.

Dessa forma, as variáveis correspondem os implicantes primos e as restrições são desigualdades do tipo"≥ 1", cujo lado esquerdo é a soma dos implicantes primos que cobrem cada mintermo [2].

Tabela 2: Combinação entre as caixas da função F_1.

Caixa	Implicante
1	$\overline{4}.\overline{2}.1$ (1)*
	$4.\overline{2}.\overline{1}$ (4)*
2	$\overline{4}.2.1$ (3)*
	$4.2.\overline{1}$ (6)*
3	4.2.1 (7)*

Combinações
$5.\overline{2}.5$ (1,4)
$\overline{4}.X.1$ (1,3)
$\overline{6}.\overline{6}.1$ (1,7)
$4.X.\overline{1}$ (4,6)
$4.\overline{3}.\overline{3}$ (4,7)
5.2.5 (3,6)
$X.2.1$ (3,7)
$4.2.X$ (6,7)

Tabela 3: Combinação das caixas geradas da função F_1.

Caixa	Implicante
1	$5.\overline{2}.5$ (1,4)*
	$\overline{4}.X.1$ (1,3)*
	$\overline{6}.\overline{6}.1$ (1,7)
	$4.X.\overline{1}$ (4,6)*
	$4.\overline{3}.\overline{3}$ (4,7)
2	5.2.5 (3,6)*
	$X.2.1$ (3,7)
	$4.2.X$ (6,7)

Combinações
$5.X.5$ (1,3,4,6)

Sendo $a = (\overline{A \oplus B})C$, $b = A(\overline{B \oplus C})$, $c = BC$, $d = AB$ e $e = (A \oplus C)$, tem-se o seguinte problema de programação linear:

$$\text{Min } 5a + 5b + 3c + 3d + 3e$$
$$\text{Sujeito às restrições}$$
$$\begin{cases} a + e \geq 1 \\ b + e \geq 1 \\ c + e \geq 1 \\ d + e \geq 1 \\ a + b + c + d \geq 1 \end{cases}$$
$$a, b, c, d, e \in \{0, 1\}$$

Resolvendo-se o problema matemático formulado obtém-se o conjunto solução, podendo ser $c = BC$ e $e = A \oplus C$ ou $d = AB$ e $e = A \oplus C$. Para verificar a diferença dos custos entre os dois métodos, obteve-se a solução da mesma função pelo método de

Quine-McCluskey. A solução encontrada foi $\overline{A}C + A\overline{C} + BC$ ou $\overline{A}C + A\overline{C} + AB$. Pode-se observar que os implicantes primos, $\overline{A}C + A\overline{C}$, obtidos no método de Quine-McCluskey correspondem a um único agrupamento representada pela expressão $A \oplus C$.

Portanto, o custo obtido no método de Quine-McCluskey foi 9 e o custo obtido no método de Quine-McCluskey Estendido foi 6. Pode-se notar que o método Quine-McCluskey Estendido apresentou um custo menor quando comparado ao método de Quine-McCluskey, sendo que esta redução deve-se a utilização da estrutura XOR.

3 Resultados e Discussões

Com o objetivo de avaliar a eficiência dos métodos estudados foram analisados os parâmetros custo, tempo de execução e consumo de memória.

Os algoritmos foram implementado em linguagem C, e a estrutura de dados utilizada foi lista simplesmente encadeada. Para o algoritmo de Quine-McCluskey, implementado por Franciscani [3], foi utilizada a mesma estrutura.

Na Figura 1 apresenta-se a comparação do custo do método de Quine-McCluskey Estendido e do método de Quine-McCluskey, para funções de até 8 variáveis. Pode-se observar que o custo obtido no método de Quine-McCluskey Estendido é menor que o método de Quine-McCluskey. Esta redução ocorre devido a utilização do operador XOR.

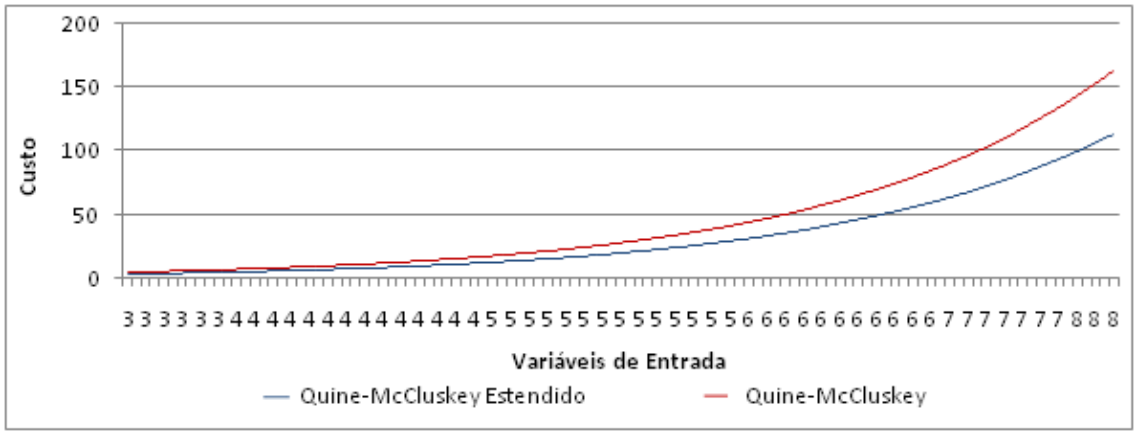

Figura 1: Gráfico de comparação do custo entre os métodos analisados.

Na Figura 2 apresenta-se a comparação do tempo de execução do método de Quine-McCluskey Estendido com o método de Quine-McCluskey. Pode-se notar que o tempo de execução crescem exponencialmente a medida que aumenta a quantidade de variáveis de entrada, e que o tempo de execução é maior quando comparado com o método de Quine-McCluskey. Este acréscimo no tempo deve-se a grande quantidade de comparações e combinações realizadas.

A quantidade de combinações realizadas no método de Quine-McCluskey Estendido é dada pela Expressão 1, em que n, é a quantidade de combinações de entrada e p são as comparações que são realizadas 2 a 2:

$$C_p^n = \frac{n!}{p!(n-p)!} \tag{1}$$

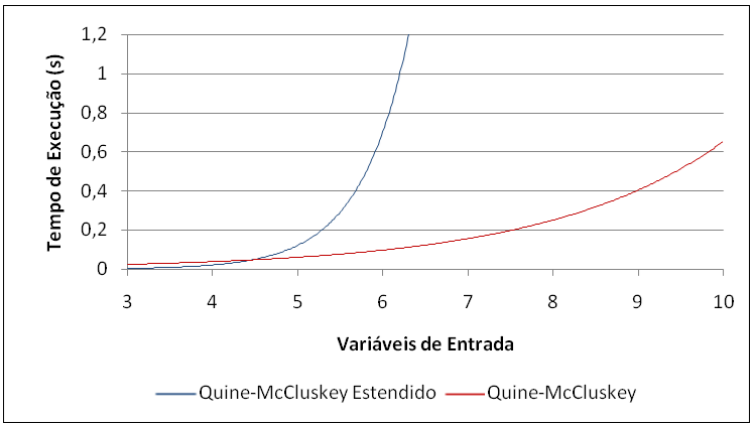

Figura 2: Gráfico de comparação do tempo de execução.

Na Figura 3 apresenta-se uma comparação do consumo de memória. Pode-se observar que o uso de memória crescem exponencialmente a medida que aumenta a quantidade de variáveis de entrada, e que o consumo de memória é maior quando comparado com o método de Quine-McCluskey, devido o método proposto armazenar mais informações decorrente do maior número de combinações.

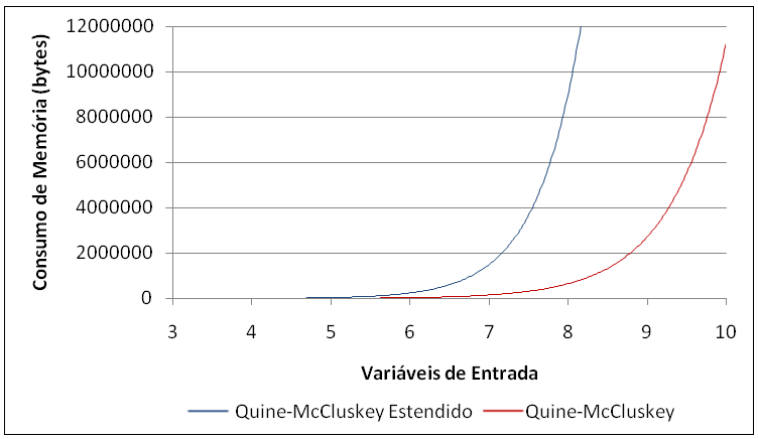

Figura 3: Gráfico de comparação do consumo de memória.

4 Conclusões

O uso de estruturas AND-XOR-XNOR possibilitou a redução no número de termos produtos quando comparado com estruturas AND-OR. Para avaliar a eficiência do

método de Quine-McCluskey Estendido, analisou-se o custo do circuito gerado, o tempo de execução e a quantidade de memória utilizada.

Com relação ao parâmetro custo, o método de Quine-McCluskey Estendido apresentou um custo menor quando comparado com o método de Quine-McCluskey, devido a utilização do operador XOR ou XNOR. Comprovou-se, desta forma, a afirmação de Sasão de que, na maioria das vezes, as implementações AND-XOR-XNOR gera circuitos com menor quantidade de termos produtos.

Em relação ao tempo de execução e consumo de memória, conclui-se que o método de Quine-McCluskey Estendido apresenta um tempo de execução e consumo de memória maior pois realiza uma maior quantidade de combinações na geração dos implicantes primos.

Agradecimentos

Os autores gostariam de agradecer ao CNPq, processo n° 309193/2015-0 e a Pós-Graduação em Engenharia Elétrica da Faculdade de Engenharia de Ilha Solteira (UNESP - Universidade Estadual Paulista).

Referências

[1] A. C. R. da Silva, Contribuição a minimização e simulação de circuitos lógicos, Dissertação de Mestrado em Engenharia Elétrica, Unesp, (1989).

[2] A. C. R. da Silva, Contribuição à síntese de circuitos digitais utilizando programação linear inteira 0 e 1, Dissertação de Doutorado em Engenharia Elétrica, Unesp, (1993).

[3] J. F. Franciscani, Consenso Iterativo: geração de implicantes primos para minimização de funções booleanas com múltiplas saídas, Dissertação de Mestrado em Engenharia Elétrica, Unesp, (2016).

[4] E. McCluskey, Minimization of boolean function, The Bell System Techinal Journal, vol. 35, 1417-1444, (1956), DOI: 10.1002/j.1538-7305.1956.tb03835.x.

[5] T. Sasao and P.Besslich, On the complexity of mod-2l sum PLA's, IEEE Transactions on Computers, vol. 39, 262-266, (1990), DOI: 10.1109/12.45212.

[6] T. Sasao, Easily testable realizations for generalized Reed-Muller expressions, IEEE Transactions on Computers, vol. 46, 709-716, (1997), DOI: 10.1109/12.600830.

[7] B.C.H. Turton, Extending Quine-McCluskey for exclusive-or logic synthesis, IEEE Transactions on Education, vol. 39, 81-85, (1996), DOI: 10.1109/13.485236.

[8] P. Wang, M.Y.Niamat, S.R.Vemuru, M. Alam and T. Killian, Synthesis of Majority/ Minority Logic Networks, IEEE Transactions on Nanotechnology, vol.14, 473-483, (2015), DOI: 10.1109/TNANO.2015.2408330.