EXTRAÇÃO DE EXPRESSÕES BOOLEANAS A PARTIR DE REDES DE TRANSISTORES REPRESENTADAS POR GRAFOS

DOMINGUES JÚNIOR, Julio S.[1]; DE SOUZA, Renato S.[1]; POSSANI, Vinicius N.[1] MARQUES, Felipe de S.[1]; DA ROSA JR, Leomar S.[1]

[1]Universidade Federal de Pelotas, Centro de Desenvolvimento Tecnológico – CDTec.
{jsdomingues, rsdsouza, vnpossani, felipem, leomarjr}@inf.ufpel.edu.br

1 INTRODUÇÃO

Atualmente, a maior parte do mercado em microeletrônica é composto principalmente de circuitos *Very Large Scale Integration* (VLSI). Estes circuitos eletrônicos possuem em média milhões de transistores em um único chip. Devido à alta complexidade do projeto deste tipo de circuito, ferramentas que apóiam o desenvolvimento desempenham um papel importante. Algumas destas ferramentas trabalham ao nível da síntese lógica, minimizando expressões lógicas através de procedimentos de fatoração (BRAYTON,1987), a fim de obter uma expressão lógica com menor número de literais. Esta forma fatorada pode ser usada para representar redes de transistores da tecnologia CMOS. Neste caso, cada literal de uma equação é equivalente a um transistor. Portanto, quando existe redução do número de literais, a rede resultante possui menor número de transistores. Isso leva a economia de área e mais eficiência no consumo de energia.

Em um trabalho anterior, propusemos um método para gerar redes de transistores otimizadas a partir de soma de produtos (SOP). Este método utiliza técnicas de compartilhamento de arestas para reduzir o número de transistores necessários para implementar uma dada função Booleana. O método transforma cada rede de transistores em um grafo. A implementação desta metodologia resultou na ferramenta *Soptimizer*.

O *Soptimizer* funciona a nível gráfico (grafo), e não é capaz de gerar uma expressão Booleana que simboliza o grafo otimizado. A fim de preencher esta lacuna no projeto, este trabalho apresenta um algoritmo capaz de extrair a expressão que representa esse grafo.

Para melhor compreensão deste trabalho, alguns conceitos relacionados são apresentados a seguir: Soma de produtos (SOP) são formas canônicas para representar funções Booleanas. Fatoração é uma técnica de otimização que pode ser aplicada para expressões Booleanas (ex:SOPs), visando a minimização de alguns critérios (BRAYTON,1987) (ZHU,1993). Normalmente, tem como objetivo a redução do número de literais em uma determinada expressão. Um grafo é um par ordenado $G = (V, E)$ que compreende um conjunto V de vértices ou nós, juntamente com um conjunto E de arestas ou linhas, que são dois elementos subconjuntos de V. Diferentes tipos de estruturas de dados são usadas para representar grafos em sistemas computacionais. A matriz de adjacência é uma delas. Além da representação da função Booleana, grafos podem também ser usados para representar redes de transistores. Neste caso, cada aresta representa um transistor e cada vértice é um ponto de conexão entre os transistores.

2 METODOLOGIA (MATERIAL E MÉTODOS)

Esta seção descreve a solução proposta para a extração da expressão Booleana a partir do grafo que representa a rede de transistores. O método proposto foi integrado com a ferramenta *Soptimizer,* a qual mapeia a função Booleana para

um grafo e então aplica otimizações reduzindo o número de arestas sem alterar a função lógica. Como resultado, a ferramenta *Soptimizer* apresenta um grafo minimizado. A motivação para realização deste trabalho reside no fato de que sendo possível realizar a extração da equação que representa este grafo otimizado, torna-se possível efetuar a integração da ferramenta *Soptimizer* com outras ferramentas e procedimentos de validação.

A geração da expressão Booleana requer travessias sucessivas do grafo de um nó terminal até outro. O algoritmo executa essas travessias aplicando compactação de arestas. Em cada travessia, são identificadas arestas associadas em série. Estas arestas identificadas são compactadas para uma única aresta. Esse processo é executado até que não seja possível mais identificar arestas candidatas. Em seguida, o algoritmo percorre o grafo novamente visando a compactação de arestas associadas em paralelo. Quando não houver candidatas em paralelo a serem compactadas, o algoritmo é reiniciado à procura de compactação em série. Esse processo iterativo é executado até que não exista possibilidade de compactação tanto em série, quanto em paralelo. As etapas de compressão do grafo são apresentadas no exemplo ilustrado pela Fig.1.

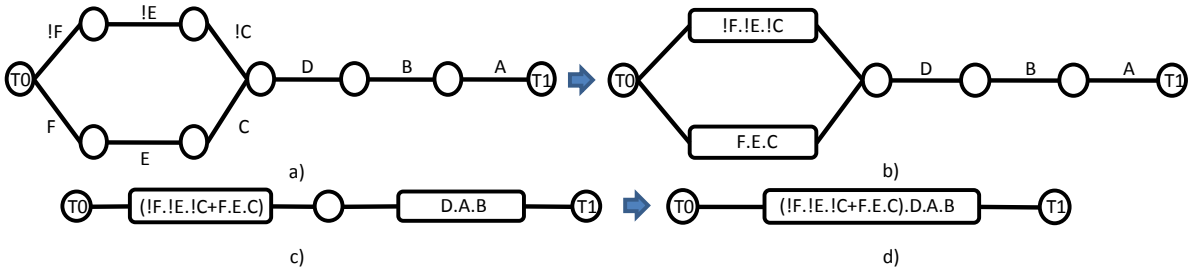

Figura1: Etapas do algoritmo de compactação do grafo.

O grafo inicial é apresentado na Fig.1.a. O processo inicia pesquisando arestas adjacentes ligadas em série em subgrafos. Neste exemplo, os primeiros grupos de arestas a serem compactadas estão ligadas ao nó terminal “T0”. No exemplo, as arestas "!F", "!E" e "!C" estão associadas em série, e podem ser compactadas para uma única aresta. Portanto, o subgrafo composto por essas arestas é substituído por uma nova aresta. De maneira semelhante, as arestas "F.E.C" e "D.B.A" são criadas. Este processo é apresentado na Fig.1.b. Depois de executar a primeira iteração da série de compactação, o algoritmo procura por associações paralelas. Na Fig.1.c, existem dois subgrafos em paralelo, então as arestas "!F.!E.!C" e "F.E.C" podem ser compactadas em uma única aresta "!F.!E.!C+ F.E.C". Neste ponto, não existem mais arestas em paralelo. O algoritmo é reiniciado, a fim de procurar novos candidatos associados em série. Assim, o grafo apresentado na Fig.1.a pode ser reduzido ao da Fig.1.d, onde o grafo final possui uma única aresta que representa a função Booleana expressa por "((!F.!E.!C+F.E.C). D.B.A)".

Um fator importante relacionado ao trabalho proposto reside no fato de que usando compactação do grafo torna-se possível identificar conexões do tipo “*bridge*”. De acordo com Zhu (ZHU,1993), o número mínimo de transistores para implementar determinadas funções Booleanas não pode ser alcançado puramente com redes do tipo série-paralelo. Redes do tipo *"bridge"* são a menor configuração de rede para um amplo conjunto de funções Booleanas.

O objetivo principal do método consiste em extrair uma expressão Booleana a partir de um grafo. Quando o grafo pode ser compactado em uma única aresta (o que significa que existe apenas associação de transistores do tipo série-

paralelo), a expressão Booleana é naturalmente alcançada. Mas, no caso de redes do tipo *"bridge",* isto não acontece. Para tratar redes *"bridge",* um algoritmo adicional torna-se necessário para permitir a correta extração das expressões. Um exemplo de compactação *"bridge"* é ilustrado pelas Fig.2.a. e Fig.2.b.

Quando o grafo de uma rede *"brigde"* é identificado, se faz necessário aplicar uma iteração extra a fim de gerar a expressão Booleana que representa esse grafo. Este procedimento final é feito usando uma matriz de adjacência para representar o grafo compactado. A expressão pode ser extraída aplicando um algoritmo baseado em busca por profundidade sobre a matriz de adjacência. A pesquisa começa em um nó terminal até o outro extremo. As Fig.2.c e Fig.2.d mostram as etapas da busca sendo executadas sobre a matriz de adjacência que representa o grafo compactado da Fig.2.b.

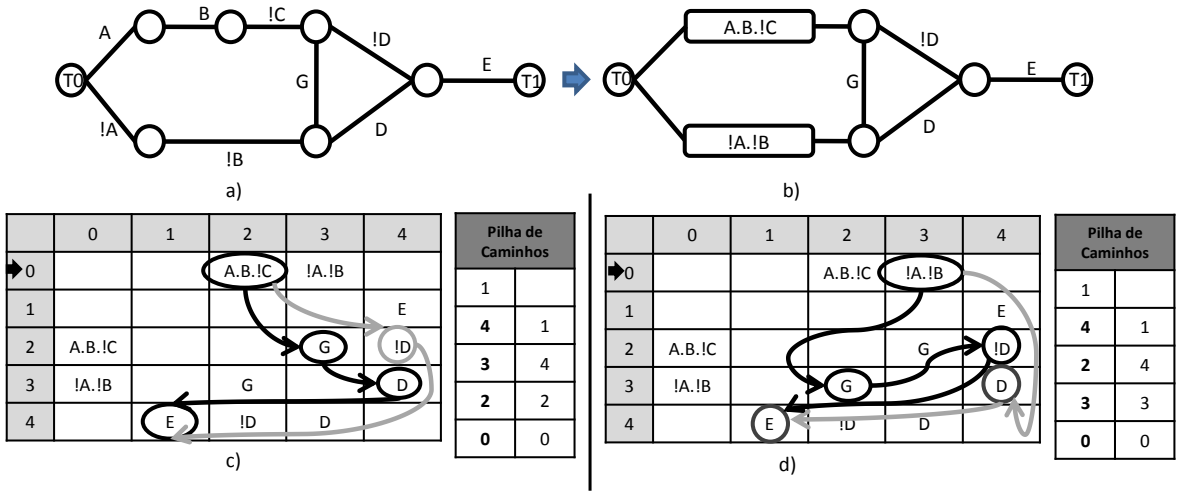

Figura 2: Associação tipo "*bridge*"(a,b) e execução do método utilizando matrizes (c,d).

A busca em profundidade inicia-se no nó terminal "T0", ou seja, sobre o índice zero da matriz de adjacência. Usamos uma pilha para manter o caminho percorrido. Esta pilha armazena os índices da matriz de adjacência que representam os nós do grafo. A partir da linha zero, o algoritmo procura o primeiro literal que representa uma aresta entre os nodos do grafo. Neste exemplo, a coluna "2", que representa a aresta "A.B.!C", é a primeira escolha. Portanto, o índice "2" é adicionado à pilha. Através de uma rotina recursiva, a linha "2" é o próximo ponto para pesquisa. Nesta linha, a primeira coluna que indica uma conexão é a coluna "0". No entanto, esta linha já foi visitada anteriormente. Desta forma, a próxima aresta a ser percorrida é "G". Portanto, a próxima linha é o índice "3". Este processo é repetido através dos vértices interligados, e acaba quando o outro vértice terminal "T1" é atingido. Nos exemplos da Fig.2.c e Fig.2.d, o índice "4" representa esse nó terminal. Ao final, haverá uma pilha que representa um caminho entre ambos os terminais.

Cada caminho construído simboliza uma sub-expressão que forma a expressão Booleana final representada pelo grafo. No exemplo das Fig.2.c e Fig.2.d, existem quatro caminhos resultantes. A união dos caminhos do grafo resulta em uma expressão Booleana que representa a rede de transistores mapeada pelo grafo. Alguns produtos em um caminho são equivalentes aos produtos de outros caminhos. Assim, técnicas associativas podem ser aplicadas para reduzir o número de literais da expressão final. Neste exemplo, o resultado deste processo é a equação "*A.B.!C.(G.D.E+!D.E)+!A.!B.(G.!D.E+D.E)".*

A biblioteca Prefuse foi usada para depurar a ferramenta *Soptimizer*. Esta biblioteca fornece um ambiente para a visualização gráfica do grafo. Um exemplo de saída do grafo é ilustrado na Fig.3.

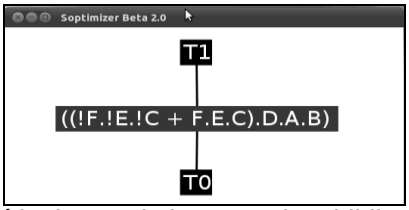

Figura 3: Saída do *Soptimizer* usando a biblioteca Prefuse.

3 RESULTADOS E DISCUSSÃO

A fim de validar as expressões extraídas pelo algoritmo proposto, executamos um conjunto de experimentos em um computador com processador Core2Duo e 4GB de RAM, rodando Ubuntu 10.10-64-bits. O experimento validou todas as 3.982 funções de 4 entradas do conjunto de funções pertencentes a classe *pClass*. Esse conjunto de funções é frequentemente utilizado como referência em processos de otimização lógica que envolvem redes de transistores. Todas as expressões foram minimizadas por meio da ferramenta *Soptimizer* e as expressões resultantes foram obtidas através do algoritmo proposto em menos de 8 minutos.

Os resultados foram avaliados e as expressões geradas pelo nosso algoritmo são equivalentes às expressões de entrada. O algoritmo proposto também pode ser utilizado para validar as redes de transistores geradas pela ferramenta *Soptimizer*. O *Soptimizer* manipula um grafo construído a partir de uma expressão de entrada. Uma vez que as expressões geradas a partir de grafos resultantes do *Soptimizer* são equivalentes a expressão original, isto significa que os métodos implementados pela ferramenta também podem ser validados pelo algoritmo proposto.

Adicionalmente, com o algoritmo proposto pudemos computar o número de transistores (número de arestas) do grafo resultante e o número de literais da expressão extraída. Como esperado, em 2.192 funções (do total 3.982), os números obtidos são os mesmos em relação ao *Soptimizer* quando o grafo resultante representa uma rede de transistores do tipo série-paralelo. No entanto, quando o grafo resultante é uma rede do tipo *"bridge"*, as expressões extraídas apresentam um número maior de literais em relação ao *Soptimizer*, pois as equações Booleanas são expressas através dos operadores AND e OR, o que denota associações série-paralelo, respectivamente.

4 CONCLUSÃO

Este artigo apresentou um método para extrair as expressões Booleanas a partir de grafos. Este método preenche uma lacuna no projeto da ferramenta *Soptimizer*. Nosso algoritmo é capaz de gerar a expressão tanto para redes de transistor do tipo série-paralelo quanto do tipo *"bridge"*. Como trabalhos futuros, pretendemos eliminar caminhos não-sensibilizados das redes de transistores geradas pelo *Soptimizer*. Dessa forma, poderemos reduzir a contagem de literais e alcançar expressões menores para as redes do tipo *"bridge"*.

5 REFERÊNCIAS

BRAYTON, R. K**. Factoring logic functions**. IBM J. Res. Dev.31,2(1987), 187- 198.

ZHU, J. et al. **On the Optimization of MOS Circuits**. IEEE Transactions on Circuits and Systems: Fundamental Theory and Applications.(1993),412-422.