OTIMIZAÇÕES EM REDES DE TRANSISTORES USANDO UMA ESTRATÉGIA DE COMPARTILHAMENTO DE ARÉSTAS EM UMA ESTRUTURA DE GRAFO

POSSANI, Vinicius N.[1]; DOMINGUES JÚNIOR, Julio S.[1]; AGOSTINI, Luciano V.[1]; MARQUES, Felipe S.[1]; DA ROSA JR., Leomar S.[1]

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

1 INTRODUÇÃO

A indústria de microeletrônica tem apresentado grandes avanços nos últimos anos, sem dúvida, projetos de circuitos digitais se tornaram uma tarefa extremamente complexa. Neste contexto surgem as ferramentas de CAD (Computer Aided Design), as quais têm dado suporte a tais projetos, contribuindo para que os desenvolvedores manipulem cada vez mais transistores e reduzam o ciclo de desenvolvimento do projeto.

Este artigo apresenta uma técnica de compartilhamento de arestas em uma estrutura de grafo para gerar redes de transistores otimizadas. A abordagem proposta parte de uma Expressão Booleana e a transforma em um grafo onde cada aresta do grafo é associada um transistor da rede. Após mapear a Expressão Booleana para o grafo, aplicam-se otimizações através de compartilhamentos de arestas, que leva a redução do número de transistores necessários para representar a expressão recebida como entrada. Atualmente, abordagens baseadas em grafos estão disponíveis na literatura e são utilizadas como alternativas aos métodos tradicionais, como a técnica de fatoração, para geração de redes de transistores (ZHU, 1993) (KAGARIS, 2007).

Os métodos baseados em fatoração consistem em manipular uma Expressão Booleana com o objetivo de reduzir o número de literais necessários para representá-la (BRAYTON, 1987). Posteriormente, a expressão otimizada é mapeada para uma rede de transistores. Métodos baseados nessa abordagem são capazes de gerar somente redes série-paralelo, o que pode não ser a melhor solução. Já as alternativas que aplicam otimizações baseadas em grafos podem gerar redes *"Wheatstone brige"*, que são arranjos de transistores capazes de representar uma função lógica com uma quantidade reduzida de transistor, superando as redes obtidas a partir de expressões otimizadas com fatoração.

2 METODOLOGIA

O método de compartilhamento de arestas proposto recebe como entrada uma SOP (Soma de Produtos). Inicialmente os produtos são separados e em seguida são mapeados para o grafo de forma que cada literal presente na SOP faça correspondência com uma aresta do grafo. Como exemplo, considere a Exp. 1 que representa uma função lógica *'XOR'* de quatro entradas. O resultado da etapa de montagem do grafo pode ser visto na Fig. 1.a, onde os literais de cada produto estão conectados em série através das arestas e vértices que compõem o grafo.

!A*!B*!C*D + !A*!B*C*!D + !A*B*!C*!D + !A*B*C*D +
A*!B*!C*!D + A*!B*C*D + A*B*!C*D + A*B*C*!D (Exp.1)

Na sequência, partindo da extremidade indicada pelo vértice 0 do grafo, todos os caminhos são percorridos com o objetivo de identificar arestas idênticas (arestas que representam o mesmo literal). Se essa condição for encontrada no grafo, então se inicia o processo de compartilhamento de arestas. Este procedimento consiste em manter somente uma das arestas iguais, eliminando as arestas redundantes e mesclando os vértices que estão conectados a estas arestas. Nas figuras abaixo, os vértices que serão mesclados estão destacados por circunferências sem preenchimento. O processo de compartilhamento é exemplificado na Fig. 1.b onde a aresta '!A' foi compartilhada e os vértices 1, 5, 8 e 11 foram mesclados. Agora, a aresta 'A' será compartilhada gerando o grafo da Fig. 1.c. Assim, os vértices 8 e 17, um de cada vez, são considerados os novos pontos de partida do processo de otimização. Neste momento o algoritmo busca por arestas iguais partindo desses dois vértices em direção ao vértice 4. Dessa forma, a aresta '!B' conectada ao vértice 8 será compartilhada e na sequência isso ocorre para a aresta 'B'. Posteriormente este mesmo processo é aplicado para as arestas '!B' e 'B' conectadas ao vértice 17. Esta etapa é representada pela Fig. 1.d.

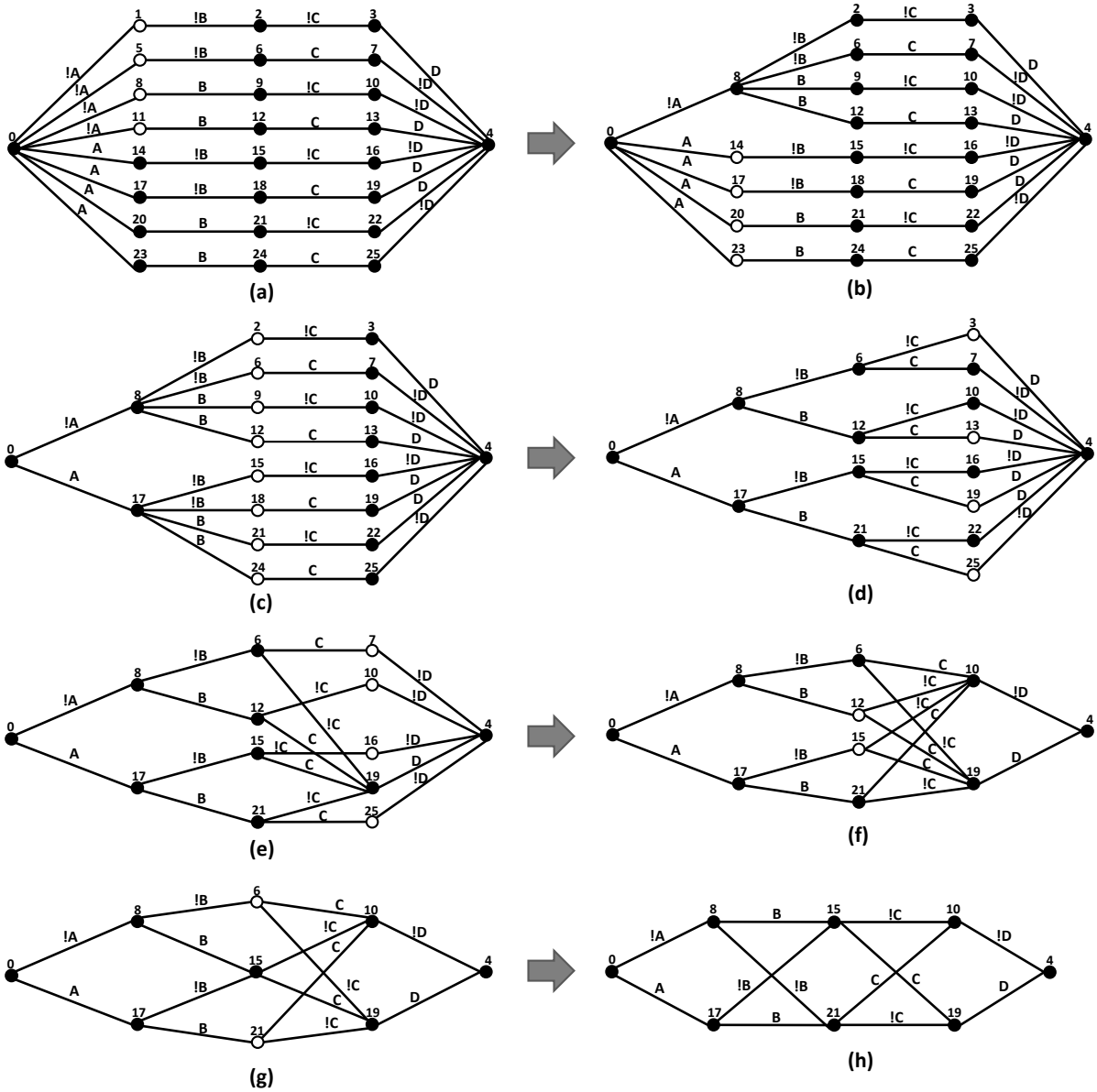

Fig. 1 – Passos do compartilhamento de arestas para a 'XOR' de quatro entradas.

Considerando o grafo ilustrado pela Fig. 1.d, partindo dos vértices 6, 12, 15 e 21, um de cada vez, em direção ao vértice 4 não é possível realizar novas otimização, pois não existem arestas iguais entre cada um desses vértices e o vértice 4. Para realizar novas otimização o algoritmo é aplicado partindo do vértice 4 em direção ao vértice 0. Assim, é possível identificar as quatro arestas 'D' e realizar o compartilhamento como mostra a Fig. 1.e. No próximo passo as arestas '!D' serão identificadas e compartilhadas como mostra a Fig. 1.f.

Neste momento, os vértices 10 e 19 são considerados, um de cada vez, como os novos pontos de partida do processo de otimização. Como pode ser visto pela Fig. 1.f, existem duas arestas '!C' conectadas ao vértice 10. Então, é possível remover a aresta '!C' conectada entre os vértices 10 e 12 e mesclar os vértices 12 e 15. Neste caso, ao mesclar os vértices 12 e 15, duas arestas 'C' estarão conectadas entre os vértices 9 e 15. Quando isto é identificado, apenas uma dessas arestas é mantida no grafo como mostra a Fig. 1.g. Este processo é aplicado novamente, mas agora para as arestas 'C' conectadas ao vértice 10, mesclando os vértices 6 e 21. Este processo irá derivar duas arestas '!C' entre os vértices 19 e 21, uma delas permanecerá no grafo e a outra será eliminada resultando no grafo final ilustrado pela Fig. 1.h. Por fim, partindo do vértice 19, não é possível realizar novos compartilhamentos fazendo com que o processo de otimização seja finalizado.

Para garantir que a rede de transistores otimizada seja uma representação equivalente a Expressão Booleana passada como entrada é preciso garantir que todos os produtos contidos na SOP estejam presentes no grafo. Além disso, também é preciso garantir que nenhum caminho inválido seja introduzido na rede. Um caminho inválido é aquele que não faz correspondência com algum dos produtos da SOP. Para isso, é preciso validar os caminhos do grafo a cada compartilhamento de arestas realizado. Então, torna-se necessário realizar a geração de todos os caminhos do grafo para executar o processo de validação.

Após gerar os caminhos, é preciso checar se cada um deles pertence à soma de produtos de entrada. Se o caminho pertence, então ele é considerado válido. Caso contrário, antes de considerá-lo inválido, é preciso verificar se o caminho é sensibilizavel ou se é não sensibilizavel. No contexto de redes de transistores um caminho é dito não sensibilizavel se ele possui um literal em ambas as polaridades como, por exemplo, 'A' e '!A'. Quando isso ocorre este caminho é considerado nulo, ou seja, ele não altera o comportamento lógico da rede, podendo, portando, ser aceito. Contudo, caso o caminho não pertença a SOP e não seja um caminho nulo, o grafo precisa ser restaurado um passo atrás ao compartilhamento que levou ao erro. Para realizar a recuperação do grafo, algumas informações a respeito das arestas e dos vértices que estão sendo removidos são armazenadas antes de realizar cada compartilhamento. Assim, através dessas informações é possível recuperar os elementos do grafo o tornando válido novamente.

Note que todos os produtos presentes na Expressão Booleana de entrada estão presentes no grafo resultante, ilustrado pela Fig. 1.h. Entretanto, devido ao compartilhamento de arestas, alguns novos caminhos foram introduzidos ao grafo. Todos estes caminhos são aceitos, pois não alteram o comportamento lógico da função que o grafo representa devido ao fato de esses caminhos serem não sensibilizaveis. Outro aspecto importante, é que o método proposto pode atingir redes *"Wheatstone bridge"* como os métodos propostos por (ZHU, 1993) e (KAGARIS, 2007). O exemplo ilustrado pela Fig. 1.h apresenta algumas estruturas do tipo *"bridge"*, o que é um benefício quando comparados a otimizações baseadas em fatoração, que pode atingir somente redes do tipo série–paralelo.

3 RESULTADOS E DISCUSSÃO

O método proposto foi implementado utilizando-se a linguagem Java. Também foi desenvolvido um módulo gráfico para visualização da rede otimizada através da biblioteca Prefuse. Além disso, foi desenvolvido um módulo para geração de *"Spice netlist"*, que são arquivos que contém a descrição da rede de transistores otimizada e podem ser aplicados diretamente em simuladores elétricos e a outras ferramentas em um fluxo comercial para o desenvolvimento de circuitos integrados.

Para descrever o método proposto utilizamos a Exp. 1 que representa uma função lógica 'XOR' de quatro entradas. A rede obtida pelo método proposto atingiu o mesmo resultado obtido por outros métodos bastante conhecidos na literatura como BDD, OpBDD, LBBDD (DA ROSA, 2008). O Resultado atingindo foi de 12 transistores, superando os resultados obtidos pelo SIS Software (SENTOVCH, 1992), que por sua vez atingiu 16 transistores. Por fim, o conjunto de funções *"p-class"* de quatro entradas foi usado para avaliação do algoritmo proposto. Este conjunto é composto por 3982 funções lógicas, onde cada uma delas foi aplicada ao método proposto e também ao SIS Software, utilizando o algoritmo *"good-factor"*. O método proposto foi capaz de gerar soluções melhores, obtendo maior redução no número total de transistores, atingindo uma redução de **2892** transistores de diferença quando comparado as resultado obtido pelo SIS. Analisando o número de transistores das **3982** funções separadamente, o método proposto obteve **ganhos em 49,37%** das funções, **empatou em 49,54%** das funções e para apenas **1,07% das funções** o SIS Software obteve resultados melhores.

4 CONCLUSÃO

Este artigo apresentou um método de compartilhamento de arestas para gerar redes de transistores otimizadas. O método foi implementado na linguagem Java e possui módulos de interface gráfica e de integração com outras ferramentas. Para descrever e avaliar o método proposto foram utilizadas as funções 'XOR' e as demais funções *"p-class"* de quatro entradas. Os resultados demonstram que o método proposto é capaz de gerar rede *"Wheatstone bridge"* levando nossa abordagem a reduzir mais transistores quando comparado ao SIS Software.

5 REFERÊNCIAS

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

Kagaris D. et al. A Methodology for Transistor-Efficient Supergate Design. **IEEE Transactions On Very Large Scale Integration (VLSI) Systems.** (2007), 488-492.

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

Da Rosa Jr, L. S. **Automatic Generation and Evaluation of Transistor Networks in Different Logic Styles.** PhD Thesis PGMicro/UFRGS, Porto Alegre, Brazil. (2008), 147 p.

Sentovich, E.; Singh, K., Lavagno; L., Moon; C., Murgai, R.; Saldanha, A., Savoj; H., Stephan, P.; Brayton, R.; and Sangiovanni-Vincentelli, A. **SIS: A system for sequential circuit synthesis.** Tech. Rep. UCB/ERL M92/41. UC Berkeley, Berkeley. (1992).