NSP KERNEL FINDER - UMA METODOLOGIA PARA GERAÇÃO DE REDES DE TRANSISTORES NÃO-SÉRIE-PARALELAS PARA A TECNOLOGIA CMOS

POSSANI, Vinicius N.[1]; SOUZA, Renato 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,
Grupo de Arquiteturas e Circuitos Integrados – GACI.
{ vnpossani, rsdsouza, agostini, felipem, leomarjr}@inf.ufpel.edu.br

1 INTRODUÇÃO

No desenvolvimento de circuitos integrados (CIs), o número total de transistores necessários para implementar cada célula lógica, bem como o número de transistores associados em série estão diretamente relacionados com o atraso de propagação do sinal, consumo de potência e área dos circuitos integrados (DA ROSA JUNIOR, 2006). Muitos dos CIs desenvolvidos atualmente são compostos por bilhões de transistores, inviabilizando que o projeto seja desenvolvido de forma manual. Assim, as ferramentas *Computer Aided Design (CAD)* têm dado suporte para que os projetistas possam gerar redes de transistores automaticamente.

Atualmente existem alguns métodos disponíveis na literatura que implementam diferentes técnicas para geração de redes de transistores otimizadas. Os mais tradicionais são baseados em fatorações algébricas e Booleanas (GOLUMBIC, 2008) (SENTOVICH, 1992) (MARTINS, 2012). Neste caso, são gerados apenas arranjos de transistores em série e paralelo (SP) devido as operações AND e OR presentes nas expressões Booleanas.

Outros métodos são baseados em otimizações através de grafos, onde, inicialmente, uma expressão Booleana é traduzida para um grafo. O grafo obtido é otimizado através de compartilhamentos de arestas (ZHU, 1993) (DA ROSA JUNIOR, 2007), ou um grafo otimizado é gradualmente composto a partir de uma expressão booleana de entrada (KAGARIS 2007). Em alguns casos, essas técnicas são capazes de gerar resultados melhores do que os métodos baseados em fatoração, por serem capazes de atingir arranjos não-série-paralelo (NSP) .

Sabe-se que nenhum desses métodos consegue atingir bons resultados (solução ótima ou próxima disso) para um conjunto de funções consideradas simples, tendo como critério o número de literais que as compõem. Tendo em vista que existem algumas lacunas a serem exploradas no processo de geração de redes de transistores, este trabalho propõe um método para descobrir se uma rede de transistores que representa uma determinada função lógica pode ou não ser implementada através de arranjos NSP.

2 METODOLOGIA

O método proposto parte de uma soma-de-produtos irredundante (ISOP) e constrói um grafo onde os cubos da expressão são representados pelos vértices e as arestas existem se os vértices possuem literais em comum. Se o grafo obtido é um grafo isomórfico a uma rede *bridge* e cada cubo compartilhou todos os seus literais através das arestas do grafo, então a ISOP pode ser implementada em NSP.

2.1 ALGORITMOS PRINCIPAL

Para *n = cubos(f)*, selecionar cubos em combinações C_n^4. Posteriormente, para cada combinação o algoritmo constrói um grafo da seguinte forma: é definido um

grafo *G = (V, E)* de uma função *H* a qual é representada por uma soma de produtos com exatamente quatro cubos. Os vértices de $V = \{v_1, v_2, v_3, v_4\}$ representam diferentes cubos de *H*, e $|V| = 4$ é o numero de vértices do conjunto *V*. Uma aresta $e = (v_i, v_j)$ em E existe se e somente se pelo menos um literal em comum apareça em v_i e v_j . A operação ($v_i \cap v_j$) representa literais em comum em ambos os vértices v_i, v_j. Então, uma aresta *e* formalmente existe se e somente se:

$$(vi \cap vj) \neq \emptyset \quad (1)$$

O rótulo de uma aresta *e* é definido da seguinte forma: $rótulo(e) = (lit(v_i) \cap lit(v_j))$ onde $lit(v_i)$ são os literais do vértice v_i. Seja Ev_1 o conjunto de arestas que estão conectados em v_1 e $|Ev_1|$ o tamanho deste conjunto. A função de entrada pode ser implementada em NSP se: $\cup(i = 1; |Ev_1|) = v_1$ para todos vértices $v \in V$. Por exemplo, considere a Fig. 1 que mostra o *Kernel* NSP obtido a partir da Equação (2).

$$f = a.b + a.c.e + d.e + b.c.d \quad (2)$$

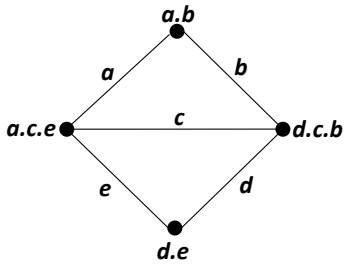

Figura 1 – *Kernel* NSP obtido a partir da Equação (2).

2.2 INSERÇÃO DE CUBO FALSO

Se o algoritmo principal não encontrar arranjos NSP, os cubos são selecionados em combinações C_n^3 e o algoritmo tenta completar o grafo com um quarto cubo falso. Um cubo falso é um cubo que gera uma sub-função redundante., Seja *f* uma dada função e seja *f'* a função que o cudo falso representa. O cubo falso é válido apenas de $f' < f\ (f'+f = f)$. Isso significa que a função do cubo *f'* está coberta por outro cubo da função *f*.

Define-se um grafo *G = (V, E)* de uma função *H* a qual é representada por uma soma de produtos com exatamente três cubos. Os vértices de $V = \{v_1, v_2, v_3\}$ representam diferentes cubos de *H*, e $|V| = 3$ é o numero de vértices do conjunto *V*. As condições para inserir arestas rotuladas no grafo são as mesmas descritas no algoritmo principal. Com o objetivo de completar o grafo *G*, um novo vértice v_z é inserido no conjunto de vértices *V* como descrito a seguir: Seja E_{vi} o conjunto de arestas conectadas ao vértice V_i, e $|E_{vi}|$ o número de arestas deste conjunto. Os literais do novo vértice são definidos como:

$$lit(v_z) = \prod_{i=1}^{|V|} \left(lit(v_i) - \bigcup_{j=1}^{|E_{v_i}|} label(e_j) \right) \quad (3)$$

A fórmula (1) é aplicada para o vértice v_z com o objetivo de inserir as novas arestas de E_{vz} em E. Esse processo é ilustrado pela Fig. 2(a) que mostra um *kernel* NSP com um cubo não sensibilizado (cubo que possui pelo menos uma variável em ambas as polaridades como *!c.d.c*) obtido a partir da Equação (4). Um cubo falso também pode ser introduzido quando o algoritmo realiza combinações C_n^4. Isso é facilmente encontrado através dos rótulos das arestas como mostra a Fig. 2(b), onde a chave '*a'* aparece replicada no *kernel* NSP obtido pela Equação (5).

$$f = !a.!d + !a.!b.c.d + !b.!c.!d \quad (4)$$

$$f = a.b + a.c + a.d + b.c.d \quad (5)$$

(a) (b)

Figura 2 – *Kernels* NSP com um cubo não sensibilizado em (a) e com uma chave replicada em (b).

3 RESULTADOS E DISCUSSÃO

Como já mencionado previamente, a entrada do algoritmo proposto é uma ISOP. De acordo com (BRAYTON, 1984), se uma função Booleana é *unate* então essa função pode ser representada através de uma única ISOP. Uma função Booleana é dita *unate* se e somente se todas as variáveis aparecem em apenas uma polaridade. Isso é importante, pois permite que o algoritmo proposto encontre a resposta exata quando manipula funções *unate*. Entretanto, considerando funções Booleanas *não-unate*, onde as variáveis podem aparecer em ambas as polaridades, o algoritmo ocasionalmente pode não encontrar uma resposta exata. Isso ocorre, pois funções *não-unate* podem ser representadas por diferentes ISOP, dessa forma, em alguns casos o algoritmo proposto pode estar manipulando uma determinada ISOP que não pode ser implementada em NSP. Porém, o algoritmo poderia ser aplicado à outra ISOP que representa a mesma função lógica para verificar se essa função pode ou não ser implementada em NSP.

Para validar e avaliar o método proposto, um conjunto de quarenta e cinco funções foi submetido ao algoritmo proposto. A Tab. 1 apresenta o número total de chaves atingido por cada método disponível na literatura e a aproximação ao resultado ótimo de acordo com o número de arranjos NSP encontrados. Nota-se que, para todas as funções, o método proposto conseguiu encontrar *kernels* NSP fornecendo boas indicações para atingir a solução ótima.

Tabela 1 – *Benchmark* sintético com 45 funções *unate*.

	Ótimo	**DA ROSA JUNIOR, 2006**	**SENTOVICH, 1992**	**MARTINS, 2010**	**DA ROSA JUNIOR, 2007**	**POSSANI, 2012**	**NSP Kernel Finder**
Nº de chaves	**304**	450	476	417	459	453	**342**
Nº de arranjos NSP	**45**	45	0	0	10	6	**45**

Um experimento similar foi realizado através do conjunto de funções lógicas de quatro entradas *P-class* que é composto por 3982 funções. Para cada função, o experimento avaliou se o tipo de arranjo encontrado (SP ou NSP) pelo método proposto foi o mesmo da função correspondente no catálogo desenvolvido por Moore (MOORE, 1958). Quando a estrutura (SP ou NSP) da rede obtida pelo método proposto é a mesma descrita no catálogo de Moore considera-se um acerto. Uma falha não significa que o algoritmo gerou um erro, mas significa que para a ISOP usada como entrada para o algoritmo não foi possível gerar uma arranjo com a mesma estrutura do catálogo de Moore. Isso ocorre devido ao fato de que funções *não-unate* podem ser representadas por diferentes ISOP, assim algumas dessas

ISOP podem ou não ser implementadas em NSP. Para todas as funções *unate* do conjunto *P-class* o método proposto foi capaz de atingir 100% de acertos quando comparado ao catálogo de Moore. Já para as funções *não-unate* o método foi capaz de atingir 82,69% de acertos.

4 CONCLUSÃO

Este trabalho apresentou um método baseado em grafo para identificar se uma ISOP pode ou não ser implementada através de arranjos NSP. O algoritmo consiste em calcular a intersecção entre as combinações dos cubos que compõem a expressão de entrada usando uma estrutura de grafo. Se a topologia do grafo obtido é equivalente a uma rede de transistores do tipo *bridge* então a expressão de entrada pode ser implementada através de arranjos NSP. Os resultados experimentais demonstraram que o método apresenta respostas exatas para determinar a topologia da rede em 100% dos casos para funções *unate*. Quando manipula funções *não-unate* o método é capaz de fornecer bons indicativos para gerar redes de transistores ótimas ou o mais próximo disso possível.

5 REFERÊNCIAS

DA ROSA JUNIOR, L. S.; MARQUES, F. S.; CARDOSO, T. M. G.; RIBAS, R. P.; REIS, A. I. Fast Disjoint Transistor Networks from BDDs. In: **19th ACM Symposium on Integrated Circuits and Systems Design**, Ouro Preto, 2006. P. 137-142.

GOLUMBIC M. C.; MINTZ, A.; ROTICS, U. An improvement on the complexity of factoring read-once Boolean functions. **Discrete Appl. Math**. Vol. 156, n. 10, p. 1633-1636. 2008.

SENTOVICH, E. et al. SIS: A system for sequential circuit synthesis. **Technical Report No. UCB/ERL M92/41, EECS Department**, University of California, Berkeley, 1992.

MARTINS, M. G. A.; DA ROSA JUNIOR, L. S.; RASMUSSEN, A.; RIBAS, R. P.; Reis, A. I. Boolean Factoring with Multi-Objective Goals. **In: IEEE International Conference on Computer Design***,* Amsterdam. 2010, p. 229-234.

ZHU, J.; ABD-EL-BARR, M. On the optimization of MOS circuits. **IEEE Transactions on Circuits and Systems: Fundamental Theory and Applications, Theory Applications**, vol. 40, no. 6, pp. 412–422, 1993.

DA ROSA JUNIOR, L. S.; MARQUES, F. S.; SCHNEIDER, F.; RIBAS, R. P.; REIS, A. I. A Comparative Study of CMOS Gates with Minimum Transistor Stacks. In: **20th ACM Symposium on Integrated Circuits and Systems Design**, Rio de Janeiro, 2007, p. 93-98.

KAGARIS, D.; HANIOTAKIS, T. A Methodology for Transistor-Efficient Supergate Design. **IEEE Transactions On Very Large Scale Integration (VLSI) Systems**, p. 488-492, 2007.

BRAYTON, R. K.; SANGIOVANNI-VINCENTELLI, A. L.; MCMULLEN, C. T.; HACHTEL, G. D. Logic Minimization Algorithms for VLSI Synthesis. **Kluwer Academic Publishers**, Norwell, MA,USA, 1984.

MOORE, E. F. Table of four-relay contact networks. **Logical Design of Electrica Circuits, by R. A. Higonnet and R. A. Grea, McGraw-Hill***,* 1958.