NSP Kernel Finder - A Methodology to Find Non-Series-Parallel Arrangements

[1]Vinicius Neves Possani, [2]Vinicius Callegaro, [2]Andre Inácio Reis,
[2]Renato Peres Ribas, [1]Felipe Souza Marques, [1]Leomar Soares da Rosa Jr.
{vnpossani, felipem, leomarjr}@inf.ufpel.edu.br
{vcallegaro, andreis, rpribas}@inf.ufrgs.br

[1]Universidade Federal de Pelotas – UFPel
Centro de desenvolvimento Tecnológico – CDTec
Grupo de Arquiteturas e Circuitos Integrados – GACI

[2]PPGC - UFRGS, Porto Alegre - RS, Brazil

Abstract

This paper presents a graph-based methodology to determine if an ISOP may be implemented in NSP arrangement. The experiments demonstrate that the proposed method deliver exact answers for unate logic functions. Furthermore, the method was able to determine exact answers in 82.69% of the cases when considering binate logic functions. The experiments were performed over the set of four inputs P-class logic function and the topology of our results was compared to the Moore catalog. In another experiment, using a synthetic benchmark, the results demonstrate that the proposed method was able to find NSP Kernels, delivering good clue to generate optimal transistors networks or, at least, the closest possible solution.

1. Introduction

In VLSI design, the total number of transistors necessary to implement a logic cell and the number of transistors associated in series are directly related to the power consumption, area and delay of the circuit [1]. In this sense, CAD tools have held designers to manipulate more transistors allowing achieving optimized cell libraries. Nowadays, in the literature there are some methods to optimize and generate transistors networks. The most traditional are based on algebraic and Boolean factorization [2-4], where a Boolean expression is manipulated to reduce the number of literals necessaries to represent the function. In a Boolean expression, every instance of a Boolean variable is called literal, and a product of literals is formally called cube. Afterward, the factored expression is translated in a transistor network using only series-parallel arrangements.

Other existent methods are based in graph optimizations, where a Boolean expression is translated in a graph that is optimized by edges sharing [5, 8] or an optimized graph is graduating composed from the initial Boolean expression [6]. These techniques are able to deliver better results when comparing to the factorization methods. This is possible because Wheatstone bridges arrangements (non-series-parallel - NSP) may be found during the graph optimization process. This kind of arrangement can implements a logic function with a reduced number of transistors, because it haves a large sharing between the cubes of the function, overcoming the series-parallel arrangements [7].

The existent techniques to generate optimized transistors networks usually using greed strategies and deliver satisfactory results. However, to the best-known of the authors, no one of these methods can achieves good results for a set of simple functions when thinking in a literal count. This set of functions respects the lower bound of transistors stack, being able to be used in current CMOS technology. Since several available techniques cannot deliver exact results, one question becomes obvious: is there some way to discover if a transistor network that represents a given logic function may be implemented in a NSP arrangement? This paper proposes a graph-based method able to identify if a Boolean expression can be or not implemented using a NSP arrangement.

2. NSP Kernel Finder

The proposed method starts from an ISOP and builds a graph where the cubes of the function are the vertices and the edges exist if the vertices have common literals. If the obtained graph is an isomorphic sub-graph to a Wheatstone bridge arrangement and each cube have all literals shared through the edges, then this function can be implemented using NSP arrangements.

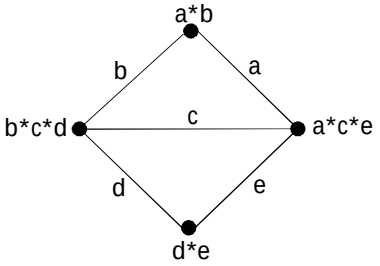
2.1. Standard Algorithm

For $n = cubes(f)$, select four cubes in combinations C_n^4. Afterward, for each combination the algorithm builds a graph as is explained following. We define a graph $G = (V, E)$ of a function H which is given as a sum of products with exactly four cubes. The vertices in $V = \{v_1, v_2, v_3, v_4\}$ represent different cubes in H, and $|V|$ is the number of vertices in the set V. An edge $e = (v_i, v_j)$ in E exists if and only if at least one literal appears in both v_i, v_j. The operation $(v_i \cap v_j)$ represents common literals in both v_i, v_j vertices. So, an edge e formally exists if and only if:

$$(v_i \cap v_j) \neq \emptyset \quad (1)$$

We define the label of e by the following: $label(e) = (lit(v_i) \cap lit(v_j))$ where $lit(v_i)$ are the literals in the vertex v_i. Let E_{v1} be the set of edges that are connected to v_1. Let $\#edges = |E_{v1}|$. The function could be implemented in NSP if: $\cup(i = 1; \#edges) = v1$ for all $v \in V$. For instance considerate the fig. 1.a that shows the NSP Kernel obtained from the Equation 2.

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



a)

Fig. 1 – NSP Kernel find from Equation 2.

2.2. Fake Cubes Insertion

If the standard algorithm not found NSP arrangements, the cubes are selected by combinations C_n^3 and the algorithm tries to complement the graph using a fake cube. Fake cubes are cubes that lead to redundant sub-functions. Eg: let f be the original function. Let f' be the function represented by the fake cube. The fake cube is valid only if $f' < f (f' + f = f)$. This means that the cube's function f' is covered by another original cube from the function.

We define a graph $G = (V, E)$ of a function H which is given as a sum of products form with exactly three cubes. The vertices in $V = \{v_1, v_2, v_3\}$ represent the different cubes in H, so $|V| = 3$. The conditions to insert a labeled edge in the graph are the same of the standard algorithm. In order to complete the graph G, a new vertex v_z is inserted into the set V as the following: Let E_{v_i} be the set of edges connected to the vertex v_i, and $|E_{v_i}|$ the number of edges in the set. The literals of the new vertex are defined as:

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

and formula (1) is applied to the vertex v_z in order to insert the new edges E_{vz} in E. This process is illustrated by fig. 2.a that shows a NSP Kernel with a not sensitized cube (cube that haves least one variable in both the polarities as !c*d*c) obtained from Equation 4. The fake cube also can be introduced when the algorithm performs combinations C_n^4. It is easily and naturally found by the labels of the edges. One instance this is illustrated by the fig. 2.b that shows a redundant switch ‘a’ in the NSP Kernel of the Equation 5.

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

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

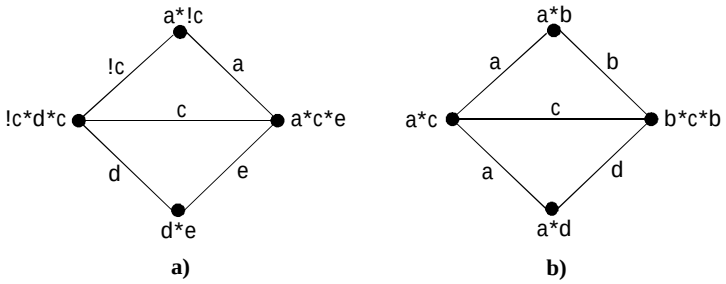


Fig. 2 – NSP Kernels with a not sensitize fake cube in a) and with a reply switch in b).

3. Experimental Results

The input of the proposed algorithm is an Irredundant Sum of Products – ISOP. According to [10], if a Boolean function is unate then only one ISOP can be expressed. A Boolean function is said to be unate if and only if all variables in the function appear in just one polarity. This statement is important because it allows the proposed algorithm to always find the exact answer when manipulating unate Boolean functions. However, when considering binate Boolean function, where variables can appear in both polarities, the algorithm occasionally cannot deliver the correct answer because sometimes an ISOP representation cannot be implemented in a Wheatstone bridge arrangement. In other words, the algorithm should be applied to another ISOP representation of the same logic function before to state if a NSP Kernel can or not be achieved.

To validate and evaluate the proposed method, a set of forty five logic functions were used to verify if the algorithm is able to find the NSP Kernels. This set is composed by only unate Boolean functions.

Tab. 1 presents the total number of switches delivered by each method available in the literature and the approximation of the optimal result according to the number of NSP arrangements found. Notice that for all functions the proposed method was able to find NSP Kernels providing good indicators for the optimal solution.

Tab. 1 – Synthetic benchmark with forty five unate NSP functions.

	Optimal	**[3]**	[4]	[1]	**[7]**	**[8]**	NSP Kernel Finder
# total of switches	304	452	417	450	459	453	-
# of NSP arrangements	45	0	0	45	10	6	45

For instance, consider Fig. 3 that shows the transistors networks obtained by each method for the Equation 5 that is part of the set of forty five functions. No one of these methods was able to generate the optimal solution, which is illustrated by the Fig. 3.a with five switches. The methods [3, 4, 7, 8] and [1] founds the solutions presents in the Fig. 3.b and Fig. 3.c respectively, with seven switches.

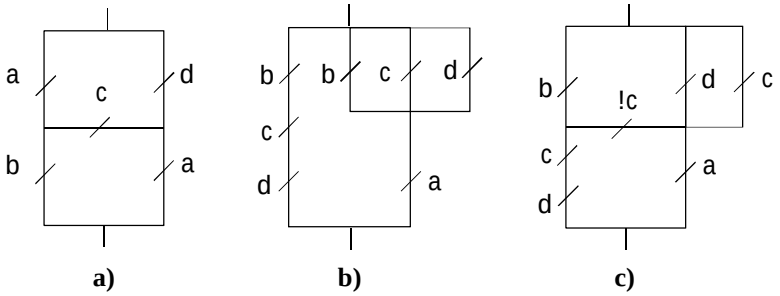


Fig. 3 – In a) optimal solution, in b) series-parallel solution and c) presents a NSP but not optimal solution.

A similar experiment was performed to the set of four input P-class logic functions that is composed by 3982 functions. For each function the experiment verifies if the obtained result have the same topology of the equivalent functions in Moore's catalog [9]. Tab. 2 presents the results of this analysis considering unate and binate functions separately.

Tab. 2 – NSP Kernel Finder results matching to Moore's catalog topology for P-class 4 inputs functions.

	# of functions	% of hits compared to Moore
Unate functions	198	100%
Binate functions	3784	82.69%

When the topology of our results is the same of the Moore's catalog we have a hit. A miss not means that the algorithm presents an error, but that for this input ISOP our method was not able to identify a NSP arrangement. As explained before, a binate function may be represented by several ISOP forms and some of them cannot be translated to a NSP arrangement. In these cases, to obtain a satisfactory answer it is necessaries to find another ISOP or to use a SOP composed by all prime cubes as input of the algorithm. As demonstrated in Tab. 2, for all unate functions of this set, the proposed method is able to deliveries 100% of hits when compared to Moore's catalog. When considering the binate functions the algorithm presented 82.69% of hits.

4. Conclusions and Future Work

This paper presented a graph-based method to identify if an ISOP can or cannot be implemented in NSP arrangement. The algorithm was developed in Java language and consists in calculates the intersections between the combinations of cubes that compose the input expression using a graph structure. Afterward, the algorithm checks the topology of the resultant graph. If it is topologically equivalent to a Wheatstone bridge arrangement then it can be implemented in NSP.

The experimental results demonstrate that when the input Boolean expression is unate our method found the same topology of the optimal solution in 100% of cases. These results were found for a synthetic benchmark as well as for the set of 4 input P-class logic functions. When the proposed method was applied over binate expressions the experiments demonstrates that 82.69% of cases the found topology is the same of the optimal solution presented by Moore's catalog, presenting a significant hit hate. Available methods in the literature are based in greed strategies that not always carry to the optimal solution. The synthetic experiment demonstrates that no one of these methods is able to deliver optimal solutions.

The kernels found by the proposed method provide good clue to generate optimal transistors networks or, at least, the closest possible solution. As future work we intend to design a method that uses these kernels to generate transistors networks.

5. Acknowledgment

Research partially funded by CNPq and FAPERGS Brazilian funding agencies under grant 11/2053-9 (Pronem).

References

[1] 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, 2007, Rio de Janeiro. 20th ACM Symposium on Integrated Circuits and Systems Design Proceedings. New York : ACM, 2007. p. 93-98.*

[2] Golumbic, M. C., Mintz, A., Rotics, U. (2008)." An improvement on the complexity of factoring read-once Boolean functions". *Discrete Appl. Math, Vol. 156, n. 10, 1633-1636.*

[3] Callegaro, V. ; Da Rosa Junior, L. S. ; Reis, A. I. ; Ribas, R. P. "A New Algorithm for Fast and Efficient Boolean Factoring". *In: 24th South Symposium on Microelectronics, 2009, Pelotas.*

[4] 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, 2010, Amsterdam. Proceedings of the IEEE International Conference on Computer Design, 2010. p. 229-234.*

[5] Zhu, J., Abd-El-Barr, M. (1993). "On the optimization of MOS circuits". *IEEE Transactions on Circuits and Systems: Fundamental Theory and Applications, Theory Appl., vol. 40, no. 6, pp. 412–422.*

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

[7] Da Rosa Junior, L. S. ; Schneider, F. ; Ribas, R. P. ; Reis, A. I. . "Switch Level Optimization of Digital CMOS Gate Networks". *In: 10th IEEE International Symposium on Quality Electronic Design, 2009,*

San Jose. 10th IEEE International Symposium on Quality Electronic Design Proceedings. Los Alamitos : IEEE Computer Society, 2009. p. 324-329.

[8] Possani, V. N. ; Timm, E. F. ; Agostini, L. V. ; Da Rosa Junior, L. S. . "A graph-based technique to optimize transistor networks". *In: 2nd IEEE Latin American Symposium on Circuits and Systems, 2011, Bogotá. 2nd IEEE Latin American Symposium on Circuits and Systems, 2011.*

[9] E. F. Moore, "Table of four-relay contact networks". *In: Logical Design of Electrica Circuits, by R. A. Higonnet and R. A. Grea, McGraw-Hill, 1958.*

[10] Robert King Brayton, Alberto L. Sangiovanni-Vincentelli, Curtis T. McMullen, and Gary D. Hachtel. 1984. "Logic Minimization Algorithms for VLSI Synthesis". *Kluwer Academic Publishers, Norwell, MA,USA.*