International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

Exact Lower Bound for the Number of Switches in Series to Implement a Combinational Logic

*Mr.P.Kiran **Mr.G.Sakru

*Associate. Professor Dept of E.C.E, Balaji Institute of Technology & Science

**M.Tech Dept of E.C.E, Balaji Institute of Technology & Science

Abstract —

This paper presents a new methodology to generate efficient transistor networks. Transistor-level optimization consists in an effective possibility to increase design quality when generating CMOS logic gates to be inserted in standard cell libraries. Starting from an input ISOP, the proposed method is able to deliver series-parallel and non-series-parallel arrangements with reduced transistor count. The experiments performed over the set of 4-input P-class Booleans functions have demonstrated the efficiency of the proposed approach.

Keywords— Logic synthesis, transistor networks, EDA, CMOS**.**

I. INTRODUCTION

In current VLSI design, the total number of transistors necessary to implement a logic gate is strongly related to the signal delay propagation, power consumption and area of integrated circuits (ICs) [1-4]. Transistor netlists are of special interest when designing standard cell libraries [5] or custom gates for improving a design [6]. To increase design quality in full-custom methodology, a handcraft generation of transistor netlists for each functional block may be performed. However, this is an extremely time-consuming task for larger ICs, making the adoption of such strategy prohibitive. Thus, it becomes crucial to have available efficient algorithms to

automatically generate optimized transistor arrangements.

In the last decades, several methods to generate and optimize transistor networks have been proposed. The most traditional solutions are based on algebraic and Boolean factorization [7-9]. In a Boolean expression, every instance of a variable is called *literal*, and a product of literals is formally called *cube*. The factorization process manipulates a Boolean expression in order to reduce the number of literals necessary to represent a Boolean function. Afterwards, the factored expression is directly translated to a transistor (switch) network. In this case, only series and parallel (SP) arrangements are obtained, related respectively to AND and OR operations present in Boolean expression.

Alternative methods to generate transistor networks are based on graph optimizations, where a Boolean expression is translated to a graph. This graph can be optimized by edges sharing [10-12] or can be gradually composed from an input expression [13]. In some cases, these techniques are able to deliver better results than factorization based methods if non-series-parallel (NSP) arrangements are able to be found during the graph manipulation process. Such optimization obtained exploiting NSP topologies is due to the large sharing between the paths that represent cubes of a function, so reducing the total switch count and overcoming SP arrangements [10,12,13].

This paper proposes a new graph-based method able to generate optimized transistor networks. Our approach presents a structural algorithm based on SP arrangements to avoid unnecessary computation during the generation of transistor networks. Different from the approach presented in [12], this new method delivers the networks not only applying transistor sharing, but also considering topological information during the generation process. Moreover, this paper presents a methodology based on SP kernels different from previous method described in [14], in which the NSP Kernel concept was introduced.

The remaining of this paper is organized as follows. Section II introduces the synthesis methodology to generate optimized switches

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

networks. In Section III some experimental results and comparisons are presented. Finally, conclusions are outlined in Section IV.

II. SYNTHESIS METHODOLOGY

The proposed method starts from an irredundant sum-of-products (ISOP), and tries to combine cubes to build SP kernels.

A SP kernel structure is illustrated in Fig. 1(a). The synthesis methodology is divided in two steps. The first step aims to build the SP kernels. The second one tries to merge the kernels found in order to deliver an optimized switch network. Thus, these steps are run in the following sequence:

(A) SP Kernel Finder.

(B) Kernel Composition.

Depending on the Boolean function (expressed through an ISOP), the routines *(A)* cannot find any kernel. Therefore, the switch network is generated during the step *(B)*, applying edges sharing technique presented in [12].

A. SP Kernel Finder

The SP Kernel Finder algorithm proposed herein can be described as follows. For $n=|cubes(f)|$, four cubes are selected by combinations . Afterwards, the algorithm builds a graph for each combination, as explained bellow. 4 n C

We define an undirected graph $G = (V,E)$ of a function H which is given by a SOP with exactly four cubes. The vertices in $V = \{v1,v2,v3,v4\}$ represent different cubes in H, and $|V|$ is the number of vertices in the set V. An edge $e = (vi,vj)$ in E exists if and only if at least one literal appears in both vi and vj. The operation $(vi \cap vj)$ represents common literals in both vi and vj vertices. Thus, an edge e formally exists if and only if:

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

We define the label of e by using $label(e) = (lit(vi) \cap lit(vj))$, where $lit(vi)$ represents the set of literals present in vi. To ensure that the obtained graph is a valid SP kernel two rules must be checked:

Rule 1 – Let Evi be the set of edges that are connected to vi. Each cube shares all its literals if the following equation is satisfied for all v V:

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

$$\bigcup_{j=1}^{|E_{vi}|} label(e_j) = lit(v_i) \quad (2)$$

Rule 2 – The obtained graph must be an isomorphic sub-graph to the graph template illustrated in Fig. 1(a). In this work this structure is called *SP kernel*.

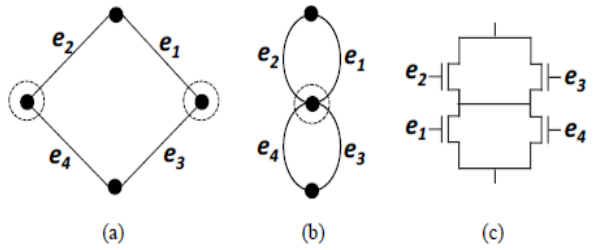

Figure 1. Vertices merging (b) and edge reordering process (c) on a S kernel (a).

After building SP kernels, the algorithm must apply some transformations over the graph to map each found kernel to a switch network. Therefore, the first step consists in merging the rounded vertices of the template shown in Fig. 1(a) in a single vertex, as shown in Fig. 1(b). Afterwards, the edges reordering routine is applied over the graph illustrated in Fig. 1(b), resulting in the switch network illustrated in Fig. 1(c). This kernel structure, illustrated in Fig. 1(a), was chosen because it leads to arrangements with a large sharing between the paths that compose the network, as shown in Fig. 1(b). It is interesting because the cubes from the input ISOP can be implemented with a reduced number of switches. This way, if this kind of arrangement may be built, our method finds it in the first step of the optimization process, avoiding unnecessary computation.

To a better understanding of such process, consider the following equation as input of the algorithm:

$$f = a.c + a.d + b.c + b.d \quad (3)$$

Due to the characteristics of this function, there is only one possible combination to select the cubes and to try to build a kernel. Thus, the algorithm found the kernel illustrated in Fig. 2(a). It can be mapped directly to the switch network illustrated in Fig. 2(b), through the vertices merging and the edges reordering routines.

Notice that the vertices merging and the edges reordering routines are necessary to implement the sharing between the paths of

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

the network that represent the cubes from Equation (3).

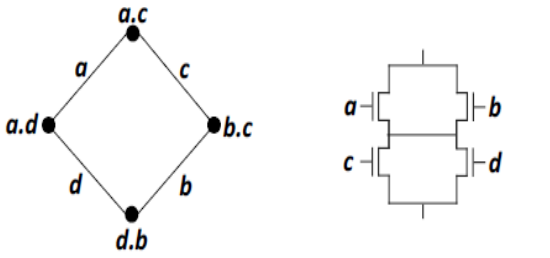

Figure 2. SP kernel (a) derived from the Equation (3) and network (b) obtained after applying vertices merging and edges reordering routines.

As demonstrated above, through this kind of SP arrangements, a set of four cubes can be implemented with a reduction of 50% in the number of literals (switches) when compared to Equation (3). This optimization rate tends to increase when multiple kernels are found and merged by the edges sharing technique applied in the step *(B)*.

B. Kernel Composition

It is important to notice that, depending on the input ISOP, multiple kernels can be found. Moreover, some cubes from the input ISOP cannot compose any kernel. Thus, five possible cases can occur when generating the transistor networks: (i) a network can be composed by just one SP kernel; (ii) a network can be composed by a SP kernel, and one or more cubes that are implemented as parallel transistor associations to this kernel; (iii) a network can be composed by multiple SP kernels in a parallel association; (iv) a network can be composed by multiple SP kernels, and one or more cubes that are implemented as parallel transistor associations; (v) there is no SP kernels and the network is implemented through the edges sharing algorithm. For each of these five cases, such topological composition is done gradually until achieving a network that is logically equivalent to the input Boolean function. During the composition process, the edges sharing procedure is applied to the network in order to eliminate redundant switches [12]. Such strategy allows a reduction in the total number of switches.

As an example of network generation composed by a SP kernel and a remaining cube associated in parallel, let us consider the following equation:

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

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

For such ISOP, the *SP Kernel Finder* routine founds the SP kernel illustrated in Fig. 3(a). This kernel may be mapped to the transistor network illustrated in Fig. 3(b). Besides that, the cube ***!a.!b.!c.!d*** was not implemented through the found kernel. Hence, this cube must be associated in parallel with such kernel as shown in Fig. 4(a).

In the next, the SP kernel and the remaining cube are gradually merged, by applying the edges sharing procedure, resulting in the sharing of the switch ***!a*** depicted by the Fig. 4(b). Afterwards, the redundant switch ***!c*** is shared, resulting in the optimized switch network presented in Fig. 5.

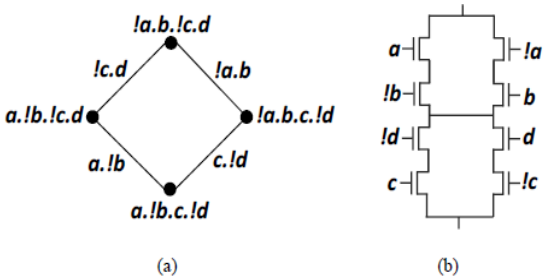

Figure 3. SP kernel (a) obtained from the Equation (4) and resultant network (b).

Notice that, the proposed method starts from SP arrangements and can achieve NSP arrangements, as illustrated in Fig. 5, through the composition procedure and the edges sharing technique. This final network is composed by 10 switches, saving 2 switches when compared to the exact factorization that needs 12 switches to implement the function described by Equation (4).

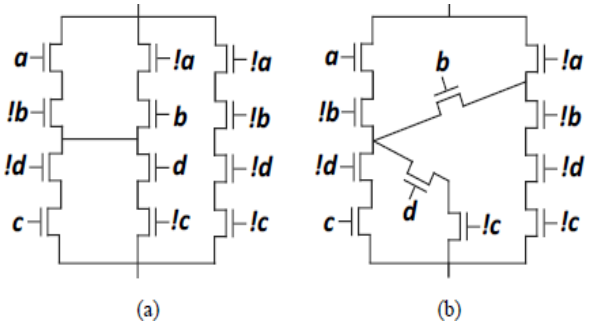

Figure 4. Network derived from the SP kernel and the remaining cube associated in parallel (a), and the intermediate network (b) after sharing the switch ***!a***.

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

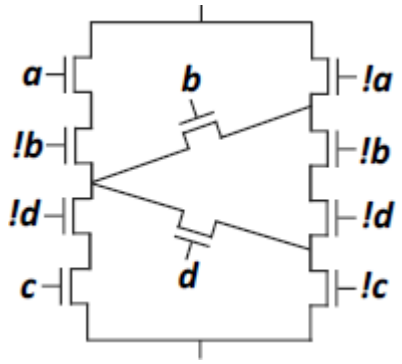

Figure 5. Final network delivered by the proposed approach for Equation (4).

To demonstrate the network generation with multiple SP kernels, let us consider the equation of the 4-input XOR function:

$$f = !a.!b.!c.d + !a.!b.c.!d + !a.b.!c.!d + !a.b.c.d + a.!b.!c.!d + a.!b.c.d + a.b.!c.d + a.b.c.!d$$

The proposed method is able to find two SP kernels. These kernels are illustrated in Fig. 6(a) and in Fig. 7(a). Each kernel may be mapped to a correspondent switch network, as illustrated in Fig. 6(b) and in Fig. 7(b), respectively.

Afterwards, during the kernel composition procedure, these kernels are associated in parallel resulting in the network illustrated by the Fig. 8. Notice that there are some redundant switches between these kernels. Hence, in order to remove the redundancies of the network, the edges sharing routine is applied resulting in a network with a large sharing of the switches, as shown in Fig. 9. The obtained solution represents the minimal switch network to implement the 4-input XOR function.

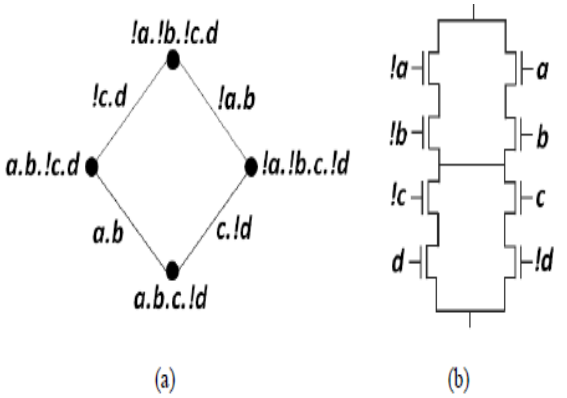

Figure 6. First SP kernel (a) obtained from Equation (5), and the switch network (b) derived from such kernel.

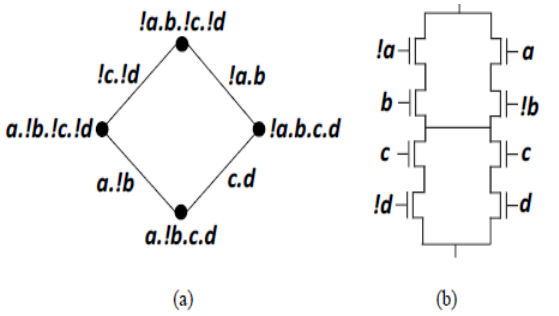

Figure 7. Second SP kernel (a) obtained from Equation (5), and the switch network (b) derived from such kernel.

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

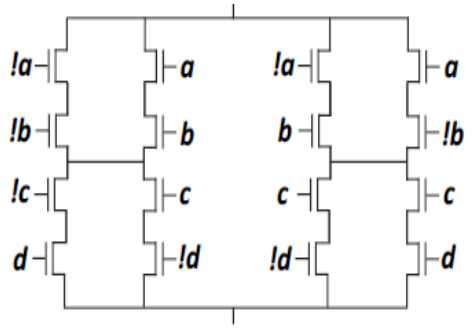

Figure 8. Parallel association of the networks built through the SP kernels found from the Equation (5).

III. EXPERIMENTAL RESULTS

In order to provide a comparison of our methodology to other available solutions described in the literature, the experiments were performed over the set of 4-input P-class logic functions. This set, composed by 3982 functions, was chosen because it contains simple functions that are more likely to be used as logic gates in real designs. We have generated gates for each function of this set, and compared them to other methods available in the literature.

Table I shows the obtained results when considering the total switch count to compute the logic gates. These results also summarize the inverters needed to implement the gates. Inverters are needed to generate the complementary signal for input variables that appears in both polarities. As presented in Table I, our method compares favorably with past approaches.

TABLE I. TOTAL SWITCH COUNT TO COMPUTE LOGIC GATES FOR THE SET OF 4-INPUT P-CLASS LOGIC FUNCTIONS.

	[4]	[9]	[11]	[13]	Proposed Method
Total number of switches	106,162	102,668	103,049	97,174	96,484

Fig. 10 shows the distribution of gains and losses of our approach when comparing to other solutions. This distribution for the set of 4-input P-class logic functions is not available in [13]. Thus, it was not possible to perform the comparison with such technique. As can be seen, the proposed method is able to reduce up to 10 transistors in some generated networks from the set of 3982 logic functions. In general, the gains remain around 1 to 4 switches per network.

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

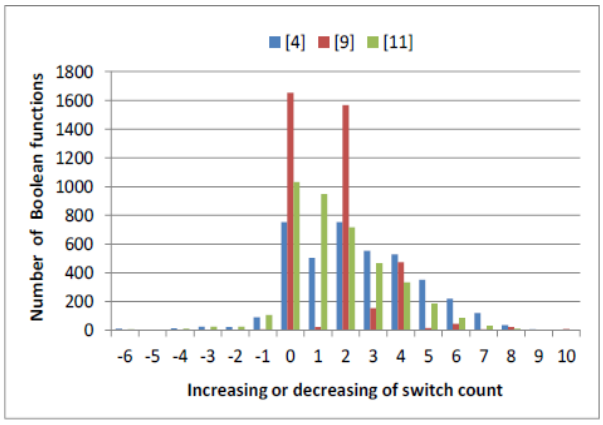

Figure 10. Distribution of switch count when comparing the proposed approach to the other methods available in the literature, considering the set of 4-input P-class logic functions.

Moreover, it is important to notice that for a small number of logic functions that compose this set, our method delivers networks with an increasing in the switch count. In these cases, the main reason for that is the bad choice when merging edges during the multiple kernels merging process. An efficient heuristic to choose the edges that should be firstly merged would help to improve the results.

The total execution time to generate all the 3982 networks was less than one second when running in an Intel Core i5 at 2.8GHz with 4 GB of RAM. It demonstrates the feasibility of the proposed method to increase design quality when generating digital CMOS circuits.

IV. CONCLUSIONS

This paper proposed a new graph-based method to generate optimized transistor (switch) networks. The proposed method results in a reduction of transistor count when compared to previous approaches. It is known that reducing transistor count in a logic gate it is possible to achieve better results in terms of signal delay propagation and power consumption. These associated gains were not explicitly investigated in this work, and they are being left as future work at gate, library and circuit design level.

REFERENCES

[1] Y. Lai; Y. Jiang; H. Chu, “BDD Decomposition for Mixed CMOS/PTL Logic Circuit Synthesis”, In: IEEE Int. Symp. on Circuits and Systems (ISCAS 2005), p. 5649-5652.

[2] H. Al-Hertani, D. Al-Khalili and C. Rozon, “Accurate total static leakage current estimation in transistor stacks”, In Proc. Int. Conf. on Computer Systems and Applications, 2006, pp. 262-65.

[3] T. J. Thorp, G. S Yee, C. M Sechen, “Design and synthesis of dynamic circuits”.

IEEE Trans. on VLSI Systems, v. 11, n. 1, p. 141-149, Feb. 2003.

[4] L. S. Da Rosa Junior, F. S. Marques, T. M. G. Cardoso, R. P. Ribas, S. Sapatnekar, A. I. Reis, "Fast Disjoint Transistor Networks from BDDs", In: 19th Symp. on Integrated Circuits and Systems Design (SBCCI 2006), p. 137-142.

[5] A. I. Reis, O. C. Anderson. Library Sizing. US Patent number: 8015517, Filing date: Jun 5, 2009, Issue date: Sep 6, 2011, Application number: 12/479,603.

[6] R. Roy, D. Bhattacharya, V. Boppana, "Transistor-level optimization of digital designs with flex cells," IEEE Trans. on Computers , vol.38, no.2, pp. 53- 61, Feb. 2005.

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

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

[9] M. G. A. Martins, L. S. Da Rosa Junior, A. Rasmussen, R. P. Ribas, A. I. Reis, "Boolean Factoring with Multi-Objective Goals". In: IEEE Int. Conf. on Computer Design (ICCD 2010), p. 229-234.

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

[11] L. S. Da Rosa Junior, F. S. Marques, F. Schneider, R. P. Ribas, A. I. Reis, "A Comparative Study of CMOS Gates with Minimum Transistor Stacks". In: 20th Symp. on Integrated Circuits and Systems Design (SBCCI 2007), p. 93-98.

[12] V. N. Possani, R. S. Souza, J. S. Domingues Junior, L. V. Agostini, F. S. Marques, L. S. Da Rosa Junior, "Optimizing Transistor Networks Using a Graph-Based Technique". Journal of Analog Integrated Circuits and Signal Processing (ALOG), May 2012.

[13] D. Kagaris, T. Haniotakis, "A Methodology for Transistor-Efficient Supergate Design", IEEE Trans. on Very

Large Scale Integration (VLSI) Systems, p. 488-492, 2007.

[14] V. N. Possani, V. Callegaro, A. I. Reis, R. P. Ribas, F. Marques, L. S. Da Rosa Junior, "NSP Kernel Finder - A Methodology to Find and to Build Non-Series-Parallel Transistor Arrangements". In: 25th Symp. on Integrated Circuits and Systems Design (SBCCI 2012), p. 1-6.

AUTHOR 1:-

***P.Kiran** is working as Associate. Professor in Dept of ECE, Balaji Institute of Technology & Science.

AUTHOR 2:-

****G.Sakru** completed his B-tech in Ganapathy Engineering College in 2014 and completed M-Tech in Balaji Institute of Technology & Science