ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Optimal Transistor Network Synthesis for Super Gate Design

R.Divyabharathi
PG Scholar,
Department of ECE,
Intell Engineering College,
AP, India.

S.Parveen
Assistant Professor,
Department of ECE,
Intell Engineering College,
AP, India.

Abstract
Optimal Transistor network 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

VLSI digital design, the signal delay propagation, power dissipation, and area of circuits are strongly related to the number of transistors (switches). Hence, transistor arrangement optimization is of special interest when designing standard cell libraries and custom gates. Switch based technologies, such as CMOS, Fin FET, and carbon nano tubes, can take advantage of such an improvement. Therefore, efficient algorithms to automatically generate optimized transistor networks are quite useful for designing digital integrated circuits (ICs).

Several methods have been presented in the literature for generating and optimizing transistor networks. Most traditional solutions are based on factoring Boolean expressions, in which only series–parallel (SP) associations of transistors can be obtained from factored forms. On the other hand, graph-based methods are able to find SP and also non-SP (NSP) arrangements with potential reduction in transistor count

Despite the efforts of previous works, there is still a room for improving the generation of transistor networks. For instance, consider a given function represented by the following equation:

$$F = a \cdot b + a \cdot c + a \cdot d + b \cdot c \cdot d. \quad (1)$$

For this function, factorization methods are able to deliver the SP network, comprising seven transistors. Existing graph-based methods, in turn, are able to provide the NSP solution, also with seven transistors. However, the optimal arrangement composed of only five transistors,, is not found by any of these methods

II.PROPOSED SCHEME

The proposed method starts from a sum-of products (SOP) form F and produces a reduced transistor network. It comprises two main modules: 1) kernel identification and 2) network composition. The former aims to find efficient SP and NSP switch networks through graph structures called kernels. The latter receives the partial networks obtained from the first module and performs switch sharing, resulting in a single network representing F. Results have shown a significant reduction in transistor count when compared with other approaches. Experiments have also demonstrated an improvement in performance, power dissipation, and area of CMOS gates as a consequence of such a device saving.

Several different methods have been proposed for implementing switch networks. The resulting networks may present different properties, which are not

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

described in a comprehensive way in the literature. The basic element to implement networks is the switch. This element can be called as direct switch, when it conducts by applying a '1' logic value in its control terminal, or complementary **switch**, when it conducts by applying a '0' logic value in its control terminal. By composing these elements, it is possible to build arrangements, known as logic networks, to allow the interconnection between two different terminals according to a given logic function that this network represents. Depending of the technology used, these switches can be implemented as physical devices. In the currently CMOS technology, they are represented by the **NMOS** transistor (direct switch) and the **PMOS** transistor (complementary switch).

The proposed method comprises two main modules:

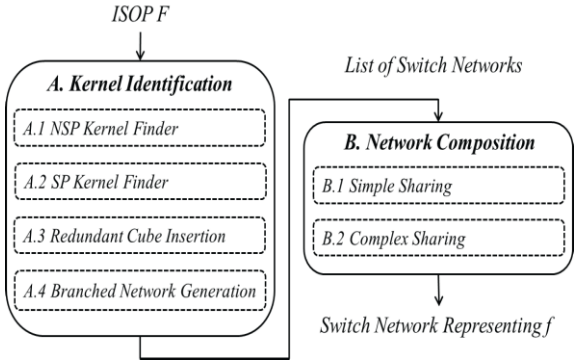

Fig.1. Execution flow of the proposed method.

A. Kernel Identification:

1) The kernel identification and 2) the switch network composition.

The former receives an ISOP F and identifies individual NSP and SP switch networks, representing sub functions of f. The latter composes those networks into a single network by performing logic sharing. The provided output is an optimized switch network representing the target function f. The execution flow of the method is presented in Fig.1.

During the kernel identification module, an intermediate data structure called kernel is used to search for possible SP and NSP networks. The kernel identification module is divided in four steps,each step is responsible for finding switch networks representing sub functions of the target function f. The NSP kernel finder step aims to obtain optimized NSP networks from an input ISOP F. When a switch network is found, the cubes used to achieve such network are removed from F. Such removal may lead to a simpler ISOP F1. The SP kernel finder step, in turn, searches for SP networks using as the input F1. Similarly to the first step, the cubes of the found SP networks are removed from F1, resulting F2. Since the remaining cubes of F2 were not useful to produce NSP or SP networks, redundant cubes are added into the kernels in order to find NSP arrangements with redundant paths. Therefore, the cubes leading to NSP networks with redundant paths are removed from F2, resulting F3. The last step produces branched switch networks, which comprises parallel paths corresponding to cubes from F3. Finally, a list of switch networks is produced as output of the kernel identification module. Each step of this first module is detailed presented below.

1) Non-series–Parallel Kernel Finder:

Let f be a Boolean function given in ISOP form F = c1 + • • + cm, where m denotes the number of cubes in F. In order to identify NSP kernels, the combination of m cubes are taken four at a time, i.e., four-combination of cubes. The sum of such four cubes results in an ISOP H, which represents h that is a sub function of f. A kernel with four vertices is obtained from H. To ensure that the generated kernel results in a NSP switch network, two rules must be checked.

Rule 1: Let Ev be the set of edges connected to the vertex v ∈ V. For each cube (vertex) v ∈ V, all literals from v must be shared through the edges e ∈ Ev.

Rule 2: The kernel obtained from H must be isomorphic to the graph. Such a graph template is referred as NSP kernel. An NSP kernel is mapped to a switch network by applying an edge swapping over three edges of the kernel. For instance, let us consider the generic NSP kernel. To map this kernel to a network, the edge e2 is moved to the place of e4, e4 is moved to the place of e3, and e3 is moved to the place

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

of e2. By applying such a reordering, it is possible to achieve the network. The reordering procedure is necessary to ensure that each path of the switch network represents a cube from the sub function h.

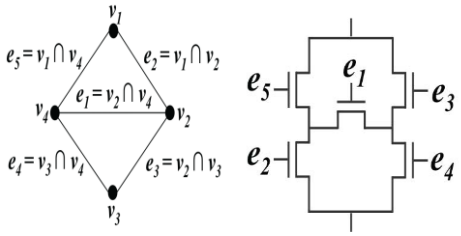

Fig.2.1NSP kernel template. Fig.2.2Resulting switch network

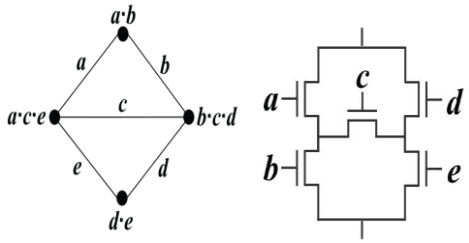

Fig.2.3NSP kernel. Fig.2.4Resulting switch network.

Example 1: Consider the following ISOP as the input to the NSP kernel finder step:

F = a · b + a · c · e + d · e + b · c · d. (2)

The resulting kernel K1, satisfies Rule 1 and Rule 2, and can be mapped by edge reordering to the switch network S1.

Example 2: By combining cubes four at a time, the NSP kernel finder procedure can find more than one kernel per ISOP. For instance, consider the following equation:

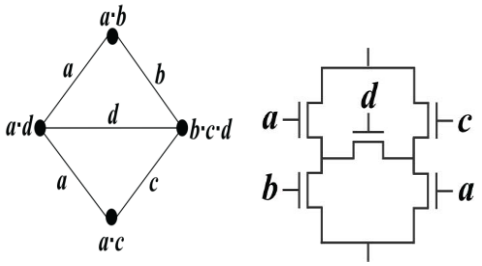

Fig. 2.5.NSP kernel. Fig.2.6Resulting switch network.

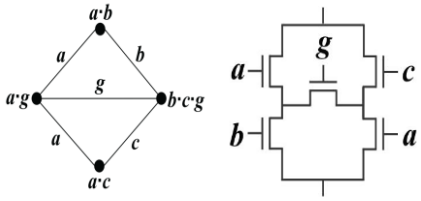

Fig. 2.7.NSP kernel. Fig.2.8Resulting switch network

F = a · b + a · c + c · e + a · d + b · c · d + a · g + b · c · g. (3)

For this ISOP, only two combinations of four cubes satisfy both Rule 1 and Rule 2, resulting in the NSP kernels.

2) Series–Parallel Kernel Finder:

Let F1 be an ISOP form that represents all the cubes of F that were not used to build switch networks in the NSP kernel finder step. To identify SP kernels, combination of m1 cubes from F1 are taken four at a time. A kernel with four vertices is then obtained. To ensure that the obtained kernel results in a valid SP network, Rule 1 and the following Rule 3 must be checked. Rule 3: The obtained kernel must be isomorphic to the network. Such a graph template is referred as SP kernel. Similarly to previous step, the SP kernel finder step must apply some transformations over the kernel in order to achieve a switch network. First, the kernel edges shown are mapped to an auxiliary template graph, Afterward, a switch network is obtained by applying the edge reordering subroutine over the auxiliary template.

F = a · c + b · c + b · d + a · d(4)

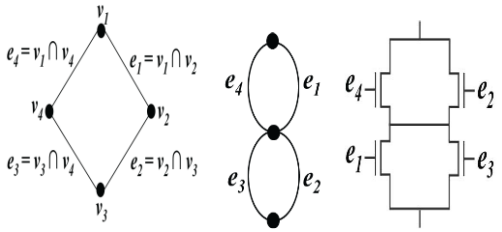

Fig3.1SP Kernel template. Fig3.2Auxiliary template. Fig 3.3Resulting Switch Network.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

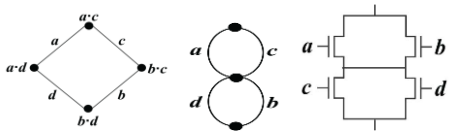

Fig3.4SP Kernel template. Fig3.5Auxiliary template
Fig 3.6Resulting Switch Network

3) Redundant Cube Insertion:

In some cases, it is useful to build NSP arrangements with redundant cubes instead of using SP associations. Thus, when there still cubes not represented through NSP and SP networks, the redundant cube insertion step tries to build NSP kernels by combining remaining cubes with redundant cubes. Let F be an ISOP representing the Boolean function f. A cube c is redundant if F + c = f . Consider a switch network representing an ISOP f. An implementation of a redundant cube c in such a network leads to a redundant logic path, i.e., the path does not contribute to the logic behavior of the network. Even though, redundant paths allow efficient logic sharing in NSP networks. The redundant cube insertion step works over an ISOP F2 representing the cubes that were not implemented by NSP and SP kernel finder steps. To obtain NSP kernels with redundant cubes, combinations of m2 cubes are taken three at a time, where m2 is the number of cubes in F2. A kernel with three vertices is then obtained for each combination.

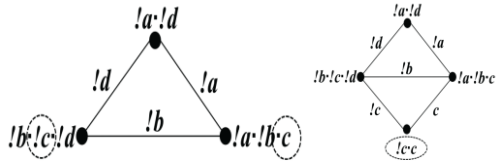

Fig 4.1graph obtained from above equation. 4.2NSP kernels with a redundant cube.

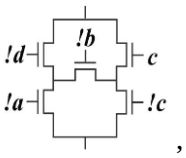

Fig 4.3Resulting Switch

4)Branched Network Generation:

Cubes from ISOP F are removed when a network implementation representing it is found. Even though previous steps are very efficient in finding logic sharing, there may still cubes not represented through any of the found networks. In this sense, the remaining cubes in F3 are implemented as a single switch network. Therefore, the branched network generation step translates each remaining cube in F3 to a branch of switches associate in series.

F = a .b . c .! d +! a .b . !c + a .!b .d (5)

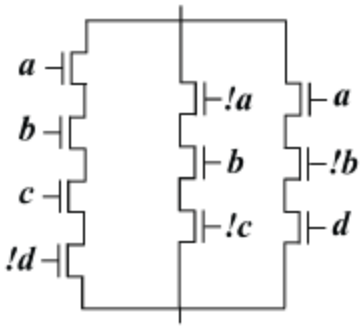

Fig 5.1Branched network

B. Network Composition:

The network composition module receives the function F and a list of partial switch networks S, generated during the kernel identification module. This module composes the networks from S in an iterative process by performing logic sharing among such networks. The target network starts empty and, for each network s ∈ S a parallel association is performed together with simple and complex sharing strategies.

The simple and the complex switch sharing are applied in order to remove redundant switches in the target network. The make Parallel. Association subroutine, in line 4, just places two networks in parallel. This way, this subroutine runs in constant time O(1). The simple and the complex switch sharing steps are presented in the following sections 1) Simple Sharing and 2) Complex Sharing together with their respective time complexities.

1) Simple Sharing:

The simple sharing step implements the edge sharing technique. Basically, the method traverses the switch network searching for equivalent switches, i.e.,

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

switches that are controlled by the same literal. The network is then restructured in such a way that one common node between equivalent switches is available. In some cases, the equivalent switches must be swapped in the networks in order to share a common node. When a common node between equivalent switches is available, only one switch is necessary, leading to a reduction in the number of switches. After performing a switch sharing, the logic behavior of the network must be checked to ensure an accurate implementation of the target function. The switch sharing is accepted only if the logic behavior of the network is maintained. This optimization and validation process is applied iteratively over the network until there is no more feasible switch sharing to be applied.

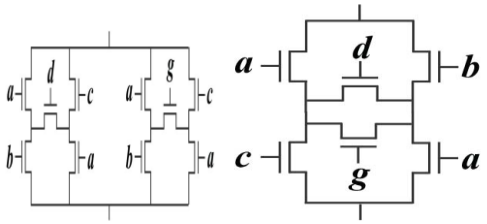

Fig 5.1Parallel association of network. fIg5.2 merge dnetwork

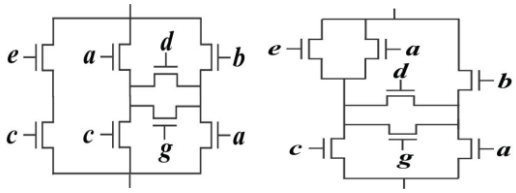

Fig 5.3 Parallel association of the branched network. fig 5.4 Resulting switch network.

2)Complex Sharing:

The complex sharing step receives a preprocessed network provided by the previous step and tries to perform additional optimizations. As mentioned in the simple sharing step, after finding equivalent switches, the procedure checks if the candidate switches have a common node that enables sharing. However, there are some cases where a common node is not directly found due to the position of the switches in the network. Hence, in order to improve the switch sharing, straightforward SP switch compressions are perform. Then, simple switch sharing is applied over the compressed network.

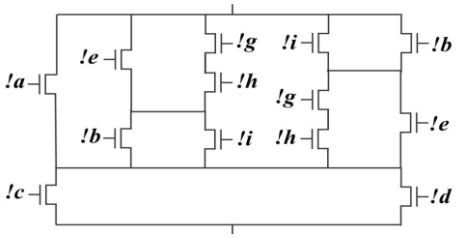

Fig 6.1 Complex sharing

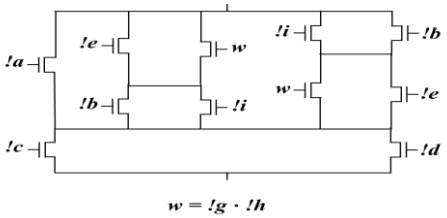

Fig 6.2 Complex sharing

Schematic diagram of a proposed system:

F= a b + b c + c d + d a e + e !f + !f ! g + !g a. (6) based on this equation proposed system.

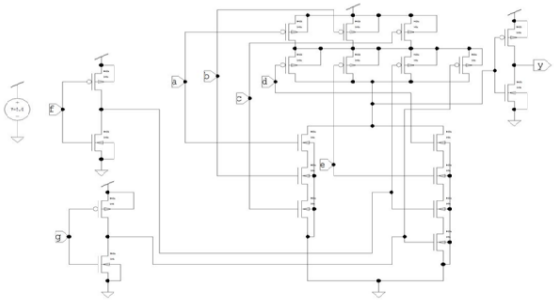

Fig 7.1 schematic diagram.

This is an implemented circuit diagram for proposed system. This proposed system schematic diagram is consists of 20 transistors consists. Compared to the existing system transistor count is less in the proposed system. Here transistors are placed based on the cmos that is pmos transistors are in the upper side and down side nmos transistors it looks like a cmos transistors, then inverters are placed because of complementary so inverters are placed. Here vdd is used and ground is placed. This circuit is designed based on the equation(6). So by using super gate design network is reduced, and transistor count is reduced and area is

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

decreased and increases speed while execution. So it is better to choose optimal transistor network synthesis for super gate design.

III. SIMULATION 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.

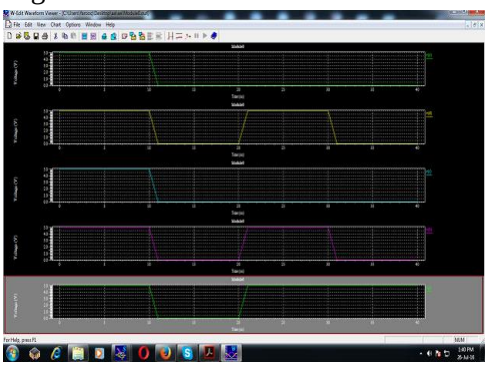

Fig.7.2 Simulation Result.

This is the simulation output of a implemented circuit equation, it is having eight inputs has applied to the different input combinations a, b, c, d and getting a output of a circuit as shown in above wave forms. Proposed system wave forms consists of time in x-axis and voltage in y-axis, in this waveforms a, b, c, d are inputs and y is the output v (in) and v (out) as shown in above wave forms. Proposed system simulation status window consists of file, edit, view, simulation, tools, options, window, and help as shown in above waveforms.

IV. CONCLUSION

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.

V.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.