INTERNATIONAL RESEARCH JOURNAL IN ADVANCED ENGINEERING AND TECHNOLOGY (IRJAET) E - ISSN: 2454-4752 P - ISSN : 2454-4744
VOL 2 ISSUE 3 (2016) PAGES 1017-1024
RECEIVED : 29/05/2016. PUBLISHED : 07/06/2016
June 7, 2016

GRAPH-BASED TRANSISTOR NETWORK GENERATION METHOD FOR SUPERGATE DESIGN

[1]S.Vidya, [2]Y.Ravi kiran varma,
[1]PG Scholar, Dept of ECE, Srinivasa Institute of Technology and Management studies, Chittoor
[2]Asst. Prof., Dept of ECE, Srinivasa Institute of Technology and Management studies, Chittoor

Abstract:

Transistor network optimization represents an effective way of improving VLSI circuits. This paper proposes a novel method to automatically generate networks with minimal transistor count, starting from an irredundant sum-of-products (SOP) expression as the input. The method is able to deliver both series-parallel (SP) and non-SP switch arrangements, improving speed, power dissipation and area of CMOS gates. Experimental results demonstrate expected gains in comparison with related approaches. The proposed method starts from a sum-of- products (SOP) form F and produces a reduced transistor network. Transistor-level optimization consists in an effective possibility to increase design quality when generating CMOS logic gates to be inserted in standard cell libraries.

Keyword:sum-of-products (SOP) -parallel (SP), VLSI**,** Transistor-level, CMOS logic gates.

1. INTRODUCTION

Trends in VLSI technology scaling demand that future computing devices be narrowly focused to achieve high performance and high efficiency, yet also target the high volumes and low costs of widely applicable general purpose designs. Transistor network optimization represents aneffective way of improving VLSI circuits. This paper proposes anovel method to automatically generate networks with minimaltransistor count, starting from an irredundant sum-of products **(SOP)** expression as the input. The method is able to deliver both series parallel **(SP)** and non-SP switch arrangements, improvingspeed, power dissipation and area of CMOS gates. Experimentalresults demonstrate expected gains in comparison with relatedapproaches. The proposed method starts from a sum-of- products **(SOP**) form *F* and produces a reduced transistor network. Transistor-level optimization consists in an effective possibility to increase design quality when generating CMOS logic gates to be inserted in standard cell libraries**.**The rest of the paper is organized as follows. In Section II existing & proposed schemes are discussed. The design implementation is discussed in Section III. In Section IV technique is validated by analysis and experiments. Finally Section V cites the conclusion. References are cited in Section V1.

2. EXISTING TECHNIQUE:

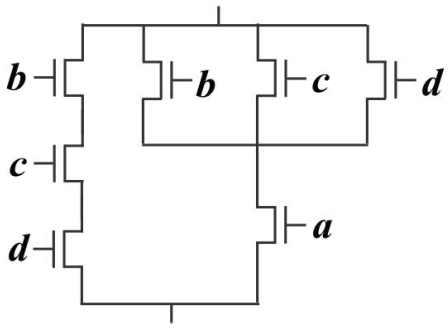


Fig.1. Transistor networks corresponding to SP solution.

INTERNATIONAL RESEARCH JOURNAL IN ADVANCED ENGINEERING AND TECHNOLOGY (IRJAET) E - ISSN: 2454-4752 P - ISSN : 2454-4744
VOL 2 ISSUE 3 (2016) PAGES 1017-1024
RECEIVED : 29/05/2016. PUBLISHED : 07/06/2016
June 7, 2016

Most traditional solutions are based on factoring Boolean expressions, in which only series–parallel (SP) associations of transistors can be obtained from factored forms with seven transistors.Existing graph-based methods are able to provide the NSP solution with seven transistors.

3. PROPOSED TECHNIQUE

1.The proposed method starts from a sum-of-products (SOP) form *F* and produces a reduced transistor network.
2.It comprises two main modules:
1) Kernel identification and
2) Network composition.
3.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*.
4.Results have shown a significant reduction in transistor count when compared with other approaches.
5.Experiments have also demonstrated an improvement in performance, power dissipation and area of CMOS gates as a consequence of such a device saving.

4. DESIGN METHODOLOGY

Therefore efficient algorithms to automatically generate optimized transistor networks are quite useful for designing digital integrated circuits (ICs).

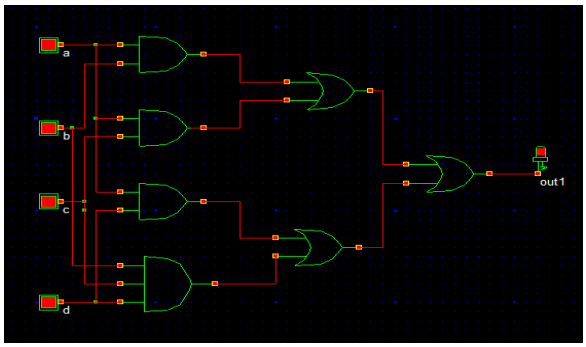


Fig.2. Diagram for proposed SOP function F

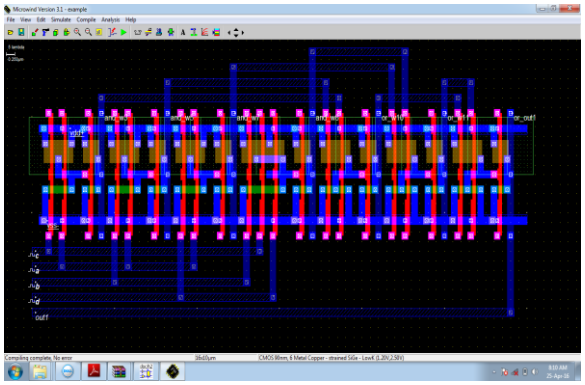


Fig.3. Input and Output Fioor-plan for proposed SOP function F

INTERNATIONAL RESEARCH JOURNAL IN ADVANCED ENGINEERING AND TECHNOLOGY (IRJAET) E - ISSN: 2454-4752 P - ISSN : 2454-4744
VOL 2 ISSUE 3 (2016) PAGES 1017-1024
RECEIVED : 29/05/2016. PUBLISHED : 07/06/2016
June 7, 2016

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)

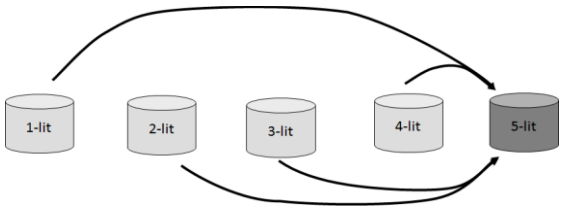


Fig.4.General functions

associations of transistors can be obtained from factored form. On the other hand, graph-based methods are able to find SP and also non-SP(NSP) arrangements with potential reduction in transistor count.

Generation of functions contained in the 5-literal bucket combining the functions in the 1-lit and 4-lit buckets and 2-lit and 3-lit buckets.

DEFINITION OF SOP:

The procedure for SOP: (Sum of Product)

1. Find out the correct truth table.
2. Find out rows lead to result of 1.
3. If the input = 1, then write the input; if the input = 0, write the input's complement. (i.e input is A and B, if A = 0, B = 1, then we write A'B). Each row should be written in a form of product.
4. Add all the products.

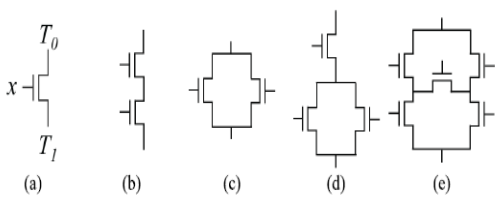
Examples:

The Truth TABLE:

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

Since there is only one row leads to F = 1. The last row: A = 1, B = 1, F = 1. Thus apply the rules listed above, IF INPUT=1, WRITE THE INPUT, IF INPUT=0, WRITE THE INPUT'S COMPLEMENT. We have: F=AB

REPRESENTATION OF SWITCH ELEMENTS:



INTERNATIONAL RESEARCH JOURNAL IN ADVANCED ENGINEERING AND TECHNOLOGY (IRJAET) E - ISSN: 2454-4752 P - ISSN : 2454-4744
VOL 2 ISSUE 3 (2016) PAGES 1017-1024
RECEIVED : 29/05/2016. PUBLISHED : 07/06/2016
June 7, 2016

Fig.5. Representation of switch elements

5. BLOCK OVERVIEW

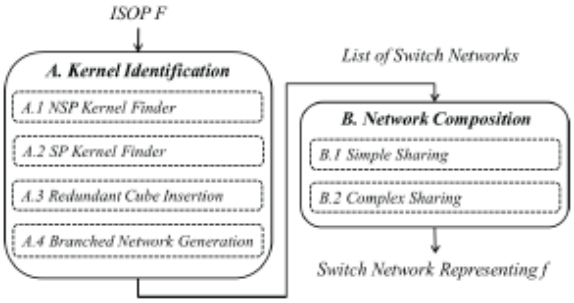


Fig.6. Execution flow of the proposed method.

During the kernel identification module, an intermediate data structure called kernel is used to search for possible SP and NSP networks. A kernel of an ISOP F with m cubes is an undirected graph $G = (V, E)$, where vertices in $V = \{v1, v2, \ldots, vm\}$ represent distinct cubes of F. An edge $e = (vi , v j) \in E$, $i \_= j$, exists if and only if $vi \cap v j \_= \emptyset$. Such edge e is labeled $vi \cap v j$. Using the kernel structure, it is possible to determine the relationship among cubes of F in order to perform logic sharing. This way, each step of the kernel identification module aims to extract kernels from F that leads to optimized switch count.

1. Non-series–Parallel Kernel Finer:

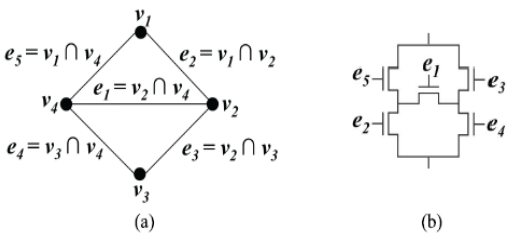
Let *f* be a Boolean function given in ISOP form F

$F = c1 + \cdot\cdot\cdot + cm$, where m denotes the number of cubes in F. In order to identifyNSP kernels the combination of m cubes are taken four at atime i.e., four-combination of cubes 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 $v \in V$. For each cube (vertex) $v \in V$, all literals from v must be shared through the edges $e \in Ev$. This rule is satisfied if and only if the following equation results the value 1:

$$\prod_{v \in V} \left(\left(\bigcup_{e \in E_v} e \right) = v \right). \qquad (2)$$

Rule 2: The kernel obtained from H must be isomorphic to the graph shown in Fig. 4(a). Such a graph template is referred as NSP kernel.



(a) NSP kernel template. (b) Resulting switch network.

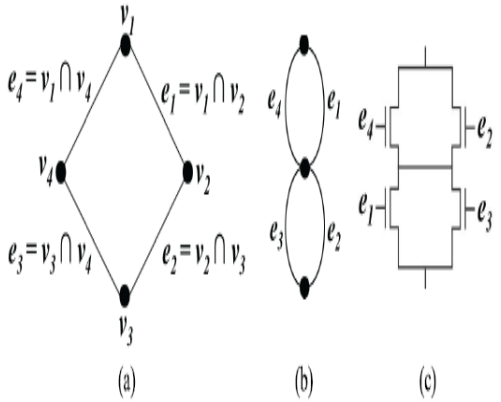
INTERNATIONAL RESEARCH JOURNAL IN ADVANCED ENGINEERING AND TECHNOLOGY (IRJAET) E - ISSN: 2454-4752 P - ISSN : 2454-4744
VOL 2 ISSUE 3 (2016) PAGES 1017-1024
RECEIVED : 29/05/2016. PUBLISHED : 07/06/2016
June 7, 2016

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 shown in Fig. (a). To map this kernel to a network the edge *e*2 is moved to the place of *e*4, *e*4 is moved to the place of *e*3, and *e*3 is moved to the place of *e*2. By applying such a reordering, it is possible to achieve the network shown in Fig.(b). The reordering procedure is necessary to ensure that each path of the switch network represents a cube from the sub-function *h*.

2) Series–Parallel Kernel Finder:

Let *F*1 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 *m*1 cubesfrom *F*1 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 graph shown in Fig (a). 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 in Fig (a) are mapped to an auxiliary template graph, as shown in Fig (b). Afterward, a switch network is obtained by applying the edge reordering subroutine over the auxiliary template graph, as shown in Fig(c).

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
*F*2 representing the cubes that were not implemented by NSP and SP kernel finder steps. To obtain NSP kernels with redundant cubes, combinations of *m*2 cubes are taken three at a time, where *m*2 is

INTERNATIONAL RESEARCH JOURNAL IN ADVANCED ENGINEERING AND TECHNOLOGY (IRJAET) E - ISSN: 2454-4752 P - ISSN : 2454-4744
VOL 2 ISSUE 3 (2016) PAGES 1017-1024
RECEIVED : 29/05/2016. PUBLISHED : 07/06/2016
June 7, 2016

the number of cubes in *F*2. A kernel with three vertices is then obtained for each combination. Thus a fourth cube (vertex) *vz* is inserted into the kernel according to the following rule.

Rule 4:Let *Ev* be the set of edges connected to the vertex $v \in V$. For each cube (vertex) $v \in V$, the literals from *v* that were not shared through the edges $e \in Ev$ are inserted in *vz* . Hence, the literals of the new vertex *vz* are obtained by

$$v_z = \prod_{v \in V} \left(v - \bigcup_{e \in E_v} e \right)$$

Where minus signal (−) denotes relative complement. Therefore, after building the redundant cube *vz* , Rule 1 and Rule 2 are applied over the resulting kernel in order to check if the cubes share all their literals through the edges.

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 *F*3 are implemented as a single switch network. Therefore, the branched network generation step translates each remaining cube in *F*3 to a branch of switches associate in series.

B. Network Composition:

The network composition module receives the function *F* and a list of partial switch networks *S*, generated during the kernelidentification 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 $s \in S$ 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 network composition is presented in algorithm . The make Parallel Association subroutine, in line, 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 .Application transistor networks are quite useful for designing digital integrated circuits (ICs).

The simple sharing step implements the edge sharing technique presented. Basically, the method traverses the switch network searching for equivalent switches, i.e., switches that are controlled by the same literal. The network is then restructured in such a algorithmof the Simple Sharing Step. 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 networkuntil there is no more feasible switch sharing to be applied.

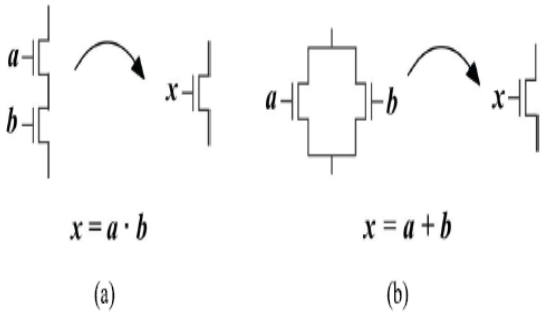
A high level description of the simple sharing step is presented in algorithm. Among all operations and subroutines needed to perform simple switch sharing, the highest time complexity is

INTERNATIONAL RESEARCH JOURNAL IN ADVANCED ENGINEERING AND TECHNOLOGY (IRJAET) E - ISSN: 2454-4752 P - ISSN : 2454-4744
VOL 2 ISSUE 3 (2016) PAGES 1017-1024
RECEIVED : 29/05/2016. PUBLISHED : 07/06/2016
June 7, 2016

given by the logical Equivalence Checking subroutine, in line 8. This procedure verify all logic paths of the network, requiring a time complexity of *O(2e/2)*, where *e* is the number of switches (edges) in the network. Thus, the simple sharing step is bounded by *O(2e/2)*.

1) 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 performed, as shown in Fig. (a) and (b), respectively.

1. LOGIC SYNTHESIS

an irredundant sum-of-products **(SOP)** expression have been designed in Verilog described at the RTL level, synthesizedwith XILINK ISE 14.1 technology library.

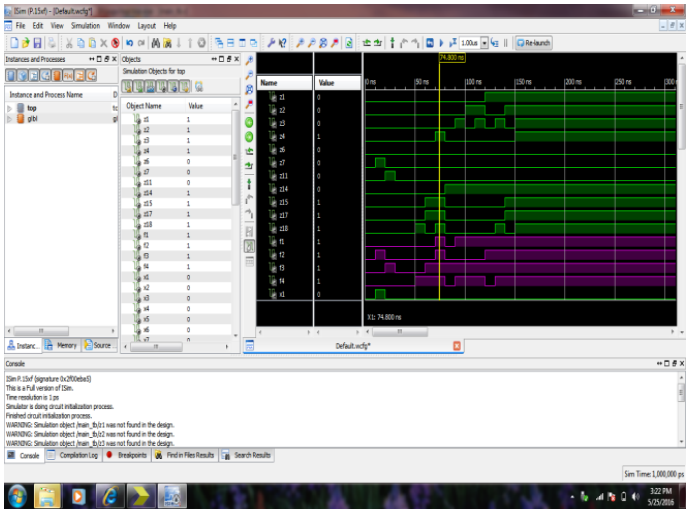
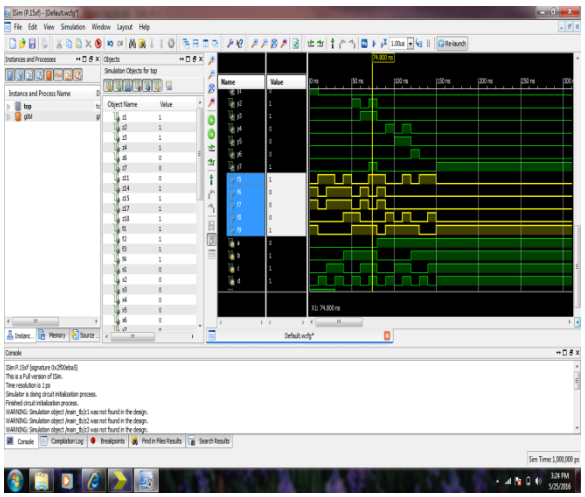


Fig.7.Working figure

INTERNATIONAL RESEARCH JOURNAL IN ADVANCED ENGINEERING AND TECHNOLOGY (IRJAET) E - ISSN: 2454-4752 P - ISSN : 2454-4744
VOL 2 ISSUE 3 (2016) PAGES 1017-1024
RECEIVED : 29/05/2016. PUBLISHED : 07/06/2016
June 7, 2016



CONCLUSION

This paper described an efficient graph-based method to generate optimized transistor (switch) networks. Our approach generates more general arrangements than the usual SP associations. Experimental results demonstrated a significant reduction in the number of transistor needed to implement logic networks, when compared with the ones generated by existing related approaches.It is known that the transistor count minimization in CMOS gates may improve the performance, power dissipation, and area of digital ICs. In a general point-of-view, the proposed method produces efficient switch arrangements quite useful to be explored by different IC technologies based on switch theory.

REFERENCES

[1] Y.-T. Lai,Y.-C.Jiang, and H.-M. Chu,"BDD decomposition for mixed CMOS/PTL logic circuit synthesis," in *Proc. IEEE Int. Symp. CircuitsSyst. (ISCAS)*, vol. 6. May 2005, pp. 5649–5652.

[2] H. Al-Hertani, D. Al-Khalili, and C. Rozon, "Accurate total static leakage current estimation in transistor stacks," in *Proc. IEEE Int. Conf.Comput. Syst. Appl.*, Mar. 2006, pp. 262–265.

[3] T. J. Thorp, G. S. Yee, and C. M. Sechen, "Design and synthesis of dynamic circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 1, pp. 141–149, Feb. 2003.

[4] A. I. Reis and O. C. Andersen, "Library sizing," U.S. Patent 8 015 517, Jun. 5, 2009.

[5] R. Roy, D. Bhattacharya, and V. Boppana, "Transistor-level optimization of digital designs with flex cells," *Computer*, vol. 38, no. 2, pp. 53–61, Feb. 2005.

[6] M. Rostami and K.Mohanram, "Dual-*vth* independent-gate FinFETs for low power logic circuits," *IEEE Trans. Comput.-Aided Design Integr.Circuits Syst.*, vol. 30, no. 3, pp. 337–349, Mar. 2011.

[7] M. H.Ben-Jamaa, K. Mohanram, and G. De Micheli,"An efficient gate library for ambipolar CNTFET logic," *IEEE Trans.Comput.-AidedDesign Integr. Circuits Syst.*, vol. 30, no. 2, pp. 242–255, Feb. 2011.

[8] M. C. Golumbic, A. Mintz, and U. Rotics, "An improvement on the complexity of factoring read-once Boolean functions," *Discrete Appl.Math.*, vol. 156, no. 10, pp. 1633–1636, May 2008.