ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 2, January 2017

Design of Supergate based on Grap-based Transistor Network

K. Bindu Madhavi[1] bindumadhav.t@gmail.com[1]

K. Anil Kumar[2] anilkumar10436@gmail.com[2]

[1]Associate Professor, Dept of ECE, HITAM College.
[2]Assistant Professor, Dept of ECE, HITAM College.

***Abstract* - Transistor network optimization represents an effective way of improving VLSI circuits. In VLSI digital design, the signal delay propagation, power dissipation, and area of circuits are strongly related to the number of transistors. This paper proposes a novel method to automatically generate networks with minimal transistor count, starting from an irredundant sum-of-products 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 architecture of this paper will be planned to implemented and also analysis the output current, output voltage, area using Dsch and micro wind.**

Index Terms— Automated synthesis, CMOS gates, digital circuit, switching theory, transistor network.

I. Introduction

Nowadays, VLSI design has definitely established a dominant role in the electronics industry. Automated tools have held designers to manipulate more transistors on a design project and shorten the design cycle. In particular, logic synthesis tools have contributed considerably to reduce the cycle time. In full-custom designs, manual generation of transistor netlists for each functional block is performed, but this is an extremely time-consuming task. In this sense, it becomes comfortable to have efficient algorithms to derive transistor networks automatically.

In VLSI digital design, the signal delay propagation, power dissipation, and area of circuits are strongly related to the number of transistors (switches) [1]–[3]. Hence, transistor arrangement optimization is of special interest when designing standard cell libraries and custom gates [5]. Switch based technologies, such as CMOS, FinFET [6], and carbon nanotubes [7], 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 [8]–[11]. On the other hand, graph-based methods are able to find SP and also non-SP (NSP) arrangements with potential reduction in transistor count [12]–[15].

This paper is organized as follows. Section II Proposed method. Section III presents experimental results regarding transistor count, area estimation, gate performance, and power dissipation. Finally, the conclusion is drawn in Section IV.

II. Proposed System

A. Proposed system.

The proposed method comprises two main modules:
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. 2

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 2, January 2017

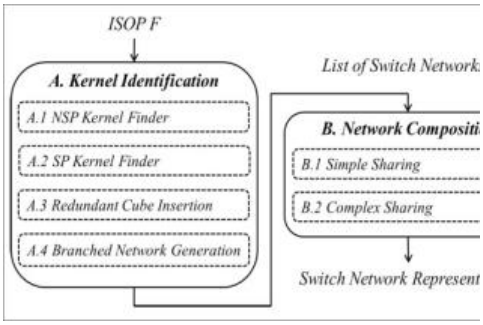


Figure 1: Execution flow of the proposed method.

Kernel Identification 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, . . . ,vm} represent distinct cubes of F. An edge = (vi , v j) ∈ E, i□ j , exists if and only if vi ∩ v j □∅.Such edge e is labeled vi ∩ v j. Using the kernel structure, it is possible to determine the relationship among cubes of Fin 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. The kernel identification module is divided in four steps; 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 shown in Fig. 2(b). Such a graph template is referred as NSP kernel.

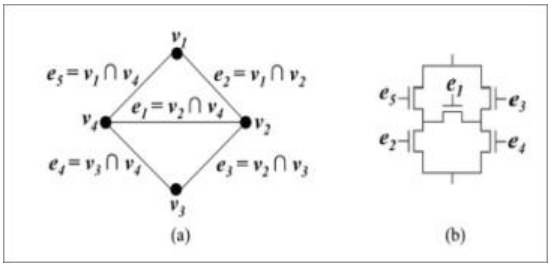


Fig. 2 (a) NSP kernel template. (b) Resulting switch network.

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 graph shown in Fig (a). Such a graph template is referred as SP kernel.

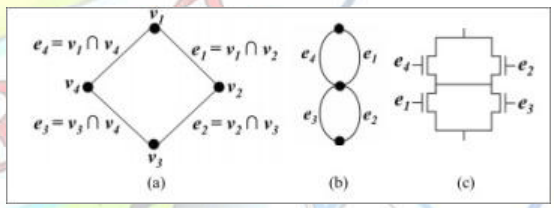


Fig 3(a) are mapped to an auxiliary template graph, as shown in Fig (b). Afterward, a switch network is obtained by applying the edge reordering subroutine over the auxiliary template graph, as shown in Fig(c). 3).

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 3(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

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 2, January 2017

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. [4] proposed a system which contributes the complex parallelism mechanism to protect the information by using Advanced Encryption Standard (AES) Technique. AES is an encryption algorithm which uses 128 bit as a data and generates a secured data. In Encryption, when cipher key is inserted, the plain text is converted into cipher text by using complex parallelism. Similarly, in decryption, the cipher text is converted into original one by removing a cipher key. The complex parallelism technique involves the process of Substitution Byte, Shift Row, Mix Column and Add Round Key. The above four techniques are used to involve the process of shuffling the message. The complex parallelism is highly secured and the information is not broken by any other intruder.

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. 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 ∈V. For each cube (vertex) v ∈V , the literals from v that were not shared through the edges e ∈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 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.

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 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

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 2, January 2017

literal. The network is then restructured in such a algorithm of 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 network until 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 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.

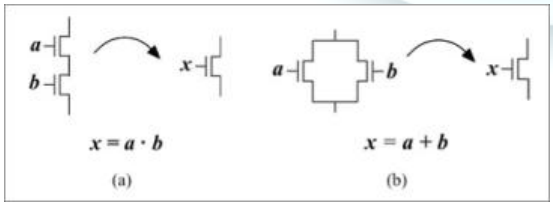


Figure 4 : (a) Series switch compression. (b) Parallel switch compression.

Transistor stack bounding:

Switch networks can be exploited by switch-based technologies, which present some restrictions or guidelines to be followed by designers. For example, in the conventional CMOS design technology, the maximum number of stacked transistors is usually limited to four. Such restriction is done in order to avoid performance degradation. Notice that there is a lower bound on the stacked transistors in switch networks. This lower bound corresponds to the minimum decision chain (MDC) property of the represented Boolean functions. In this sense, an interesting feature to control (or to limit) the number of stacked transistors was included in our method. The method can operate in two execution modes, bounded and unbounded, as described below.

1. Bounded Mode

In this execution mode, a bound variable is used as reference to control the maximum number of transistors in series. The bound value must be equal or greater than the number of literals of the smallest cube from F, i.e., the maximum number of literals in a single cube. When the method is running in the bounded mode, the kernel identification module accepts only switch networks in which maximum stacked transistors do not exceed the bound value. Hence, the networks satisfying such a bound are added to the list S of found networks. This control is also performed during the network composition module when applying switch sharing, since it can increase the transistor stack.

2. Unbounded Mode

When running in the unbounded mode, there is no restriction of transistor stacking, i.e., the bound variable is not considered. Basically, just the total transistor count of the network is taken as metric cost. Hence, there are cases that the networks generated through the unbounded mode result fewer transistors when compared with bounded solutions. Moreover, these different modes are quite useful to explore the tradeoff between circuit area and performance.

Extension.

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

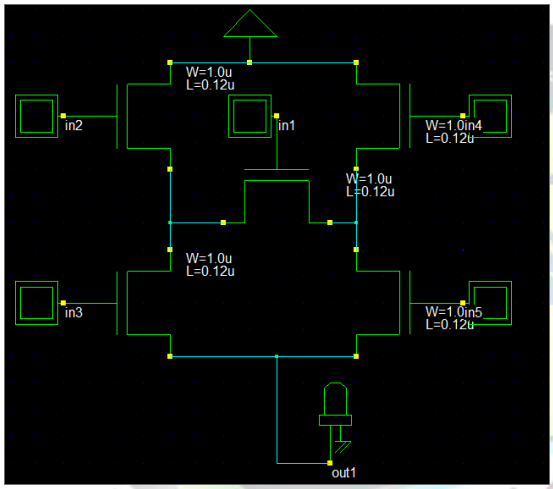
ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 2, January 2017

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 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.

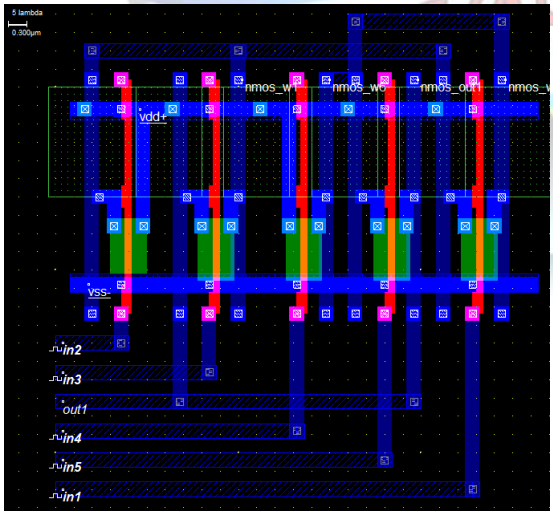
III.RESULTS

Proposed system.

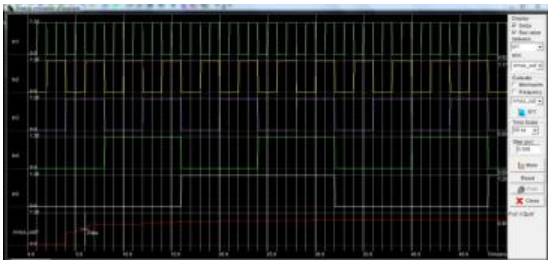
Schematic.



Layout Design.

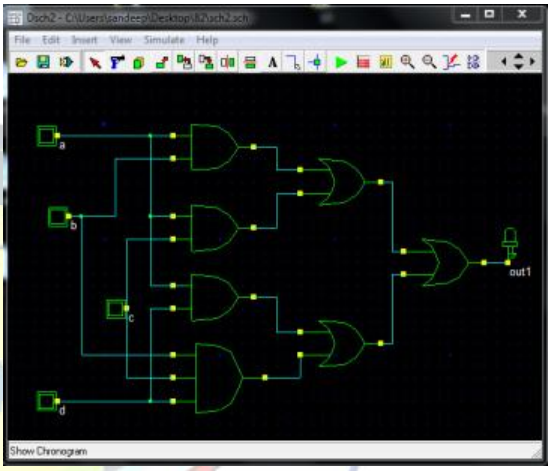


Simulation.

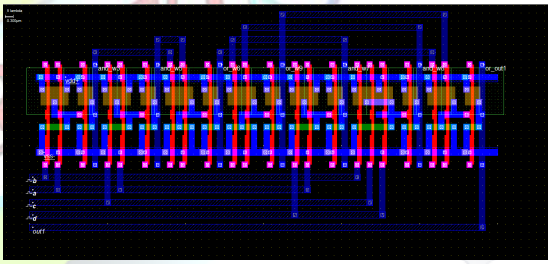


Extension

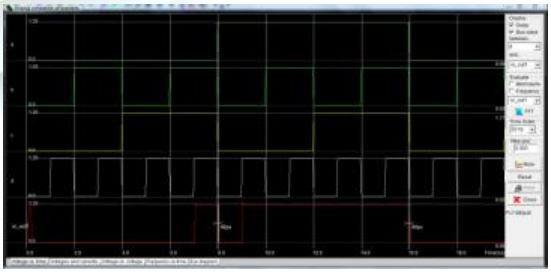
Schematic.



Layout design.



Simulation.



IV. Conclusion

This paper described an efficient graph-based method to generate optimized transistor (switch) networks. Our approach generates more

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 2, January 2017

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. Circuits Syst. (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] Christo Ananth, H.Anusuya Baby, "High Efficient Complex Parallelism for Cryptography", IOSR Journal of Computer Engineering (IOSR-JCE), Volume 16, Issue 2, Ver. III (Mar-Apr. 2014), PP 01-07.

[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.-Aided Design 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.

[9] E. M. Sentovich et al., "SIS: A system for sequential circuit synthesis," Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/ERL M92/41, May 1992.

[10] M. G. A. Martins, V. Callegaro, L. Machado, R. P. Ribas, and A. I. Reis, "Functional composition and applications," in Int. Workshop Logic Synthesis Tech. Dig. (IWLS), Jun. 2012, pp. 1–8. [Online]. Available: http://www.inf.ufrgs.br/logics/

[11] M. G. A. Martins, L. S. da Rosa, Jr., A. B. Rasmussen, R. P. Ribas, and A. I. Reis, "Boolean factoring with multi-objective goals," in Proc. IEEE Int. Conf. Comput. Design (ICCD), Oct. 2010, pp. 229–234.

[12] L. S. da Rosa, Jr., F. S. Marques, F. R. Schneider, R. P. Ribas, and A. I. Reis, "A comparative study of CMOS gates with minimum transistor stacks," in Proc. 20th Annu. Conf. Integr. Circuits Syst. Design (SBCCI), Sep. 2007, pp. 93–98.

[13] V. N. Possani, R. S. de Souza, J. S. Domingues, Jr., L. V. Agostini, F. S. Marques, and L. S. da Rosa, Jr., "Optimizing transistor networks using a graph-based technique," J. Analog Integr. Circuits Signal Process., vol. 73, no. 3, pp. 841–850, Dec. 2012.

[14] D. Kagaris and T. Haniotakis, "A methodology for transistor-efficient supergate design," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 15, no. 4, pp. 488–492, Apr. 2007.

[15] J. Zhu and M. Abd-El-Barr, "On the optimization of MOS circuits," IEEE Trans. Circuits Syst. I, Fundam. Theory Appl., vol. 40, no. 6, pp. 412–422, Jun. 1993.

[16] R. K. Brayton, A. L. Sangiovanni-Vincentelli, C. T. McMullen, and G. D. Hachtel, Logic Minimization Algorithms for VLSI Synthesis. Norwell, MA, USA: Kluwer, 1984.

[17] T. Sasao, Switching Theory for Logic Synthesis. New York, NY, USA: Springer-Verlag, 1999.

[18] C. Piguet, J. Zahnd, A. Stauffer, and M. Bertarionne, "A metal-oriented layout structure for CMOS logic," IEEE J. Solid-State Circuits, vol. 19, no. 3, pp. 425–436, Jun. 1984.

[19] M. G. A. Martins, V. Callegaro, R. P. Ribas, and A. I. Reis, "Efficient method to compute minimum decision chains of Boolean functions," in Proc. 21st Ed. Great Lakes Symp. VLSI (GLSVLSI), May 2011, pp. 419–422.

[20] Federal Univ. Rio Grande do Sul, Logics Lab. (Oct. 2012). Catalog of 53 Handmade Optimum Switch Networks. [Online]. Available: http://www.inf.ufrgs.br/logics/docman/53_NSP_Catalog.pdf

[21] M. A. Harrison, Introduction to Switching and Automata Theory. New York, NY, USA: McGraw-Hill, 1965, pp. 408–472.

[22] K. Tanaka and Y. Kambayashi, "Transduction method for design of logic cell structure," in Proc. Asia South Pacific Design Autom. Conf. (ASP-DAC), Jan. 2004, pp. 600–603.

[23] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45 nm early design exploration," IEEE Trans. Electron Devices, vol. 53, no. 11, pp. 2816–2823, Nov. 2006. [Online]. Available: http://ptm.asu.edu/

[24] I. E. Sutherland, R. F. Sproull, and D. F. Harris, Logical Effort: Designing Fast CMOS Circuits. San Mateo, CA, USA: Morgan Kaufmann, 1999.

[25] T. Uehara and W. M. Vancleemput, "Optimal layout of CMOS functional arrays," IEEE Trans. Comput., vol. C-30, no. 5, pp. 305–312, May 1981.