Computing with Nano-Crossbar Arrays: Logic Synthesis and Fault Tolerance

Mustafa Altun
Dept. of Electronics and Communication Engineering
Istanbul Technical University, Turkey
altunmus@itu.edu.tr

Valentina Ciriani
Dipartimento di Informatica
Università degli Studi di Milano, Italy
valentina.ciriani@unimi.it

Mehdi Tahoori
Karlsruhe Institute of Technology
Karlsruhe, Germany
tahoori@ira.uka.de

***Abstract*—Nano-crossbar arrays have emerged as a strong candidate technology to replace CMOS in near future. They are regular and dense structures, and can be fabricated such that each crosspoint can be used as a conventional electronic component such as a diode, a FET, or a switch. This is a unique opportunity that allows us to integrate well developed conventional circuit design techniques into nano-crossbar arrays. Motivated by this, our project aims to develop a complete synthesis and performance optimization methodology for switching nano-crossbar arrays that leads to the design and construction of an emerging nanocomputer. First two work packages of the project are presented in this paper. These packages are on logic synthesis that aims to implement Boolean functions with nano-crossbar arrays with area optimization, and fault tolerance that aims to provide a full methodology in the presence of high fault densities and extreme parametric variations in nano-crossbar architectures.**

I. Introduction

In 1965, Gordon Moore made an influential prediction about CMOS size shrinking, formulated as the Moore Law stating that the number of transistors on a chip doubles every 18 to 24 months. His prediction has kept its validity for decades. Nowadays this trend has reached a critical point and it is widely accepted that the trend will end in the next decade [1]. Even Gordon accepted that his prediction would lose it validity in near future [8]. At this point, research is shifting to novel forms of nanoarchitectures including nano-crossbar arrays [13]. Nano-crossbar arrays are regular and dense structures that are generally fabricated by self-assembly as opposed to lithography based conventional and relatively costly CMOS fabrication techniques [16].

This study targets nanoarrays where each crosspoint behaves as a switch, either two-terminal or four-terminal. This is illustrated in Figure 1. Depending on the used technology mostly with nanowires, a two-terminal switch based crosspoint can behave as a diode [10] or a FET [12]. Additionally, a four-terminal switch based crosspoint can be implemented with a crossed nanowires with an addition of an insulated controlling input [2]. Note that both diode and FET based crosspoints conduct current in one direction. However, four-terminal switches conduct current in multiple directions. In this study, we implement Boolean functions by considering array sizes. We propose synthesis techniques with performing area optimization. The results show that four-terminal switch based implementations offer favorably better crossbar sizes.

Along with the advantages in terms of circuit size and fabrication, nano-crossbar arrays have drawbacks including reliability issues, standing against commercial production. While reliability of nano-crossbars have been satisfactorily improved using new defect tolerance techniques [15], architectural level of reliability is still a problem. To tolerate high defect rates and variations, our approach is to integrate defect tolerance to improve the manufacturing yield (for fabrication defects), fault tolerance to ensure the lifetime reliability (for errors during normal operation), and variation tolerance to ensure the predictability and performance (for parametric variations), in the design methodologies. Adaptive and built-in defect, variation and fault tolerant design flows, fundamentally different from conventional approaches, are proposed in which the objective is to ensure high manufacturing yield and runtime reliability of the circuit at extremely low costs. We plan to exploit the opportunities created by the nano-crossbar technology such as reprogrammability and abundance of programmable resources to provide defect, variation and fault tolerance.

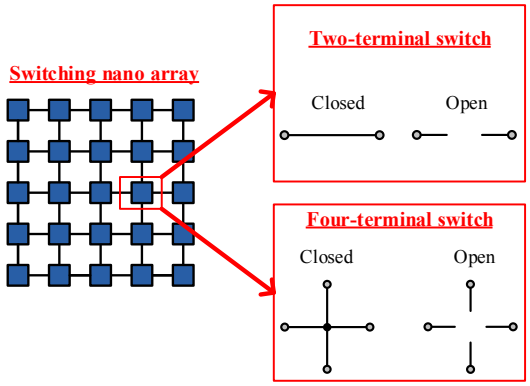

Fig. 1. A switching crossbar nanoarray modeled with two-terminal and four-terminal switches.

Organization of the paper is as follows. Section II provides a general overview of our EU-H2020-RISE project NANOxCOMP # 691178. The following sections summarize the obtained results and the main objectives identified in the first two work packages of the project. In particular, Section III investigates logic synthesis techniques for two-terminal and four-terminal switch based nanoarrays by comparing array sizes needed to implement given Boolean functions. Section IV discusses the development of defect, variation and fault tolerant techniques in the presence of high defect densities and extreme parametric variations for nano-crossbar architectures. Section V concludes the paper with future directions.

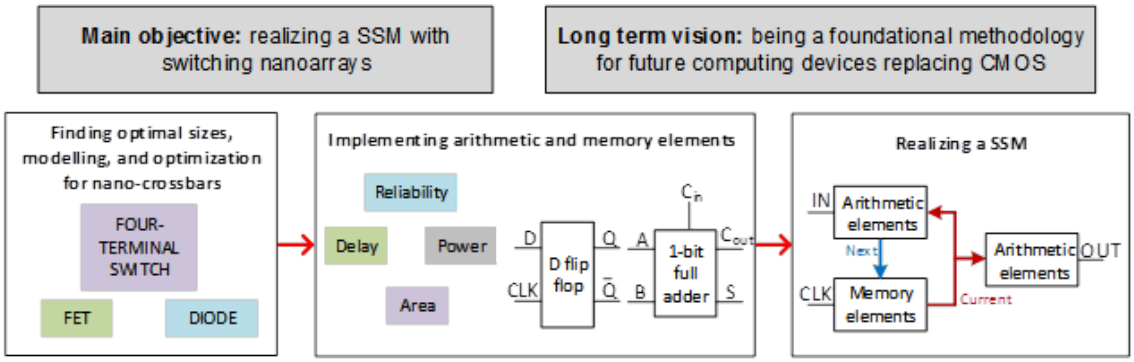

Fig. 2. Project overview with main objectives.

II. Overview of the Project

The main goal of this project is developing a complete synthesis and optimization methodology for switching nano-crossbar arrays that leads to the design and construction of an emerging nanocomputer. New models for diode, FET, and four-terminal switch based nanoarrays are developed. The proposed methodology implements both arithmetic and memory elements, necessitated by achieving a computer, by considering performance parameters such as area, delay, power dissipation, and reliability. With combination of arithmetic and memory elements a synchronous state machine (SSM), representation of a computer, is realized. The proposed methodology targets variety of emerging technologies including nanowire/nanotube crossbar arrays, magnetic switch-based structures, and crossbar memories. The results of this project will be a foundation of nano-crossbar based circuit design techniques and greatly contribute to the construction of emerging computers beyond CMOS. The topic of this project can be considered under the research area of "Emerging Computing Models" or "Computational Nanoelectronics", more specifically the design, modeling, and simulation of nanoscale switches beyond CMOS. The project overview with main objectives is illustrated in Figure 2.

One of the major promises of emerging nanotechnologies for on-chip applications is ultimate integration density, and the reduction of manufacturing and power consumption costs. However, there is a big gap in 1) extending the existing electronic design automation (EDA) flow for emerging technologies in order to introduce them in the architecture and system design in a systematic-way, and 2) novel computer architecture systems based on emerging technologies to provide high performance and minimize power consumption at the same time. This project includes the introduction of hybrid EDA flow as well as emerging computer architectures by gathering well respected experts working in these broad fields.

The proposed research methodology involves all aspects of computer-aided circuit and system design that constitutes the "computational part" of the project. The methodology also involves electrical and physical characteristics of the applicable emerging technologies that constitutes the "technological part" of the project. This project is interdisciplinary in nature. There will be a continuous information flow between its technological and computational parts. European beneficiary organizations' expertise is mostly on computational part and collaborations will be made for the technological part.

III. Logic Synthesis Techniques

In this section, we investigate two-terminal or four-terminal switch based implementation methodologies.

A. Two-terminal Switch based Implementations

These implementations consider each crosspoint of an array as a two-terminal switch that behaves like a diode or a FET. Since diodes and FET's conduct current through their two terminals that are anode & cathode for diodes and source & drain for FETs, they are fundamentally two-terminal switches. Boolean functions are implemented by mainly using conventional techniques that are diode-resistor logic and CMOS logic with an important constraint: Boolean functions should be implemented in their sum-of-products (SOP) forms; other forms such as factored or BDD (Binary Decision Diagram) cannot be used since these forms require manipulation/wiring of switches that is not applicable for nanoarrays [11].

Array sizes for diode and FET based nanoarrays: Given a target Boolean function f, we derive formulas of the array sizes. This is shown in Figure 3. For diode based implementations, each product of f requires a row (horizontal line), and each literal of f requires a column (vertical line) in an array. Additionally, one extra column is needed to obtain the output. For FET based implementations, each product of f and its dual, f^D, requires a column, and each literal of f requires a row in an array. As an example, consider a target function $f = x_1\overline{x_2} + \overline{x_1}x_2$ having 4 literals and 2 products; $f^D = x_1x_2 + \overline{x_1}\ \overline{x_2}$ has 2 products. This results in array sizes of 2×5 and 4×4 for diode and FET based implementations, respectively. Note that both formulas always result in optimal array sizes; no further reduction is possible.

Type	Array Size Formulas (Optimal)
Diode	(number of products in f) x ("number of literals in f" + 1)
FET	(number of literals in f) x ("number of products in f" + "number of products in f^D")

Fig. 3. Size formulas for diode and FET based implementations.

B. Four-terminal Switch based Implementations

Four-terminal switch based implementation considers each crosspoint of an array as a four-terminal switch [2]. Four terminals of the switch are all either mutually connected (ON-logic 1 applied) or disconnected (OFF-logic 0 applied). Boolean functions are implemented with top-to-bottom paths in an array by taking the sum (OR) of the product (AND) of literals along

each path. This makes Boolean functions implemented in their sum-of-products (SOP) forms. An example of a four-terminal switch based implementation is given in Figure 4.

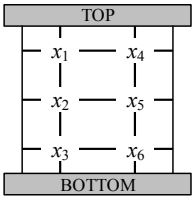

Fig. 4. Four-terminal switch based array/lattice implementing the Boolean function $x_1x_2x_3 + x_1x_2x_5x_6 + x_2x_3x_4x_5 + x_4x_5x_6$.

The problem of synthesising Boolean functions with four-terminal switch based arrays or lattices is first presented in [3]. In this study, array size formula is derived by considering that each product of a target function f and its dual, f^D, require an array column and an array row, respectively. This is given in Figure 5. As an example, consider a target function $f = x_1\overline{x_2} + \overline{x_1}x_2$ and $f^D = x_1x_2 + \overline{x_1}\ \overline{x_2}$ both having 2 products. This results in an array size of 2×2.

Type	Array Size Formula (Non-optimal)
Four-terminal	(number of products in $\boldsymbol{f}$) x (number of products in $\boldsymbol{f^D}$)

Fig. 5. Size formula for four-terminal switch based implementation.

Examining the array size formulas in Figure 3 and Figure 5, we see that while the formulas in Figure 3 always result in optimal sizes, the sizes derived from the formula in Figure 5 for four-terminal switch based arrays are not necessarily optimal. In the following part, we present two preprocessing synthesis techniques to reduce sizes of four-terminal switch based arrays.

1) Preprocessing with function decomposition:
In the framework of switching lattices synthesis, the available minimization tools in [3], [2], [9], [11] are not yet as mature as CMOS synthesis tools. We show that the computational cost for the synthesis of a switching lattice could be directly reduced by decomposing an input Boolean function. We first implement decomposed functions/blocks in separate lattices and then we merge them together in a lattice that describes the entire function. We focus on a particular decomposition method that gives rise to the bounded-level logic networks called P-circuits [7]. We recall from [3] that given the switching lattices implementing the functions f and g, we can easily construct the lattices representing their disjunction $f+g$ and their conjunction $f \cdot g$ using a padding column of 0's and a padding row of 1's, respectively. We can use these rules to derive a lattice describing a P-circuit decomposition [5]:

$$\text{P-circuit}(f) = (\overline{x}_i \oplus p)\, f^{=} + (x_i \oplus p)\, f^{\neq} + f^{I}$$

where I is the intersection of the projections of f onto the two sets $x_i = p$ and $x_i \neq p$, and

1) $(f|_{x_i=p} \setminus I) \subseteq f^{=} \subseteq f|_{x_i=p}$
2) $(f|_{x_i\neq p} \setminus I) \subseteq f^{\neq} \subseteq f|_{x_i\neq p}$
3) $\emptyset \subseteq f^{I} \subseteq I$.

This definition can be easily generalized to incompletely specified Boolean functions. Thus, the synthesis idea of P-circuits is to construct a network for f by appropriately choosing the sets $f^{=}$, $f^{\neq}$, and f^{I} as building blocks.

The same idea can be exploited in the switching lattice framework: the sub-functions $f^{=}$, $f^{\neq}$, and f^{I} depend on $n-1$ variables instead of n, they have a smaller on-set than f, and their lattice synthesis should produce lattices of reduced area. Therefore, the overall lattice for f derived composing minimal lattices for $f^{=}$, $f^{\neq}$, and f^{I}, could be smaller than the one derived for f without exploiting its P-circuits decomposition. This expectation has been confirmed by a set of experimental results, where the utility of the decomposition-based approach has been evaluated applying the two synthesis methods presented in [2] and in [9].

2) Preprocessing of regular functions:
Another possible approach to reduce the dimension of a Boolean function, to be synthesized, is to exploit possible regularities on its structure. For example, D-reducible functions are functions completely contained in an affine space strictly smaller than the entire Boolean space [4]. Therefore, we can represent a D-reducible function f as $f = \chi_A \cdot f_A$, where A is its unique associated affine space, χ_A is the characteristic function of A, and f_A is the projection of f onto A. Notice that f and f_A have the same number of points, but these are now compacted in a smaller space. The D-reducibility of a function f can be exploited in the lattice synthesis process [6]. For this purpose, we can independently find the lattice implementations for the characteristic function of the affine space A and for the projection of the function f onto A. Finally, we compose the obtained lattices in order to construct an overall lattice for f.

IV. Built-in Variation, Defect, and Fault Tolerance

One of the main focuses of this project is development of defect, variation and fault tolerant techniques in the presence of high defect densities and extreme parametric variations, particularly for crossbar array nano-architectures. To tolerate high defect rates and variations, our approach is to integrate defect tolerance to improve the manufacturing yield (for fabrication defects), fault tolerance to ensure the lifetime reliability (for errors during normal operation), and variation tolerance to ensure the predictability and performance (for parametric variations), in the design methodologies for future nanotechnologies. Adaptive and built-in defect, variation and fault tolerant design flows are proposed

A. Built-in Self-testing (BIST) and Self-diagnosis (BISD)

The main novelties of our proposed BIST are 100% exhasutive coverage of all logic-level faults (including stuck-at, bridging, open, and functional faults) and minimality of test vector and configurations set. Our proposed approach is based on implementing single-term functions in all crossbars during the test mode which allows propagation of all sensitized faults to the outputs, and hence, detection [14]. Diagnosis is achieved by selecting the subset of sensitized fault in each test configuration in such a way that the pass/fail outcomes of test configurations uniquely encodes the faulty resources. The number of diagnosis configurations is also logarithmic to the number of faults. The subsets of sensitized faults in diagnosis configurations can be modeled by block codes.

B. Built-in Self-mapping (BISM)

Blind BISM. In this scheme, a random configuration for the crossbar is generated on-the-fly and then application-dependent BIST is used to check whether this configuration is defect-free. Since no application-independent test is performed and no diagnosis is involved (neither application-independent nor application-dependent), blind BISM is very fast and effective for low defect-densities. The self-reconfiguration circuitry is also very simple and small.

Greedy BISM. When defect density is high, blind BISM approach becomes ineffective due to too many configuration retries. In this case, greedy BISM is performed as follows. It starts with a random configuration followed by BIST. If the configuration fails, application-dependent BISD is performed to identify the defective resources utilized in the most recent configuration. The self-reconfiguration uses the diagnosis information to only bypass (reconfigure) the defective parts of that configuration. This process is repeated until the last configuration is defect-free.

Hybrid BISM. This BISM procedure is the combination of the above procedures and works for all defect densities and also various defect density distributions across different crossbars in a nano-chip, i.e. ideal for both global and local defect density variations. In hybrid BISM, the BISM procedure initially starts with blind BISM. If it is not successful after a pre-defined number of retries, it automatically switches to greedy BISM.

C. Application-Independent Defect Tolerant Flow

Most drawbacks of the traditional defect-aware flow, shown in Fig. 6(a), are due to the fact that this method is application dependent, i.e. defects are handled in a per-application basis. In contrast to the defect-aware design flow, we propose a defect-unaware design flow to tolerate defects in crossbar arrays. This design flow is shown in Fig. 6(b). In this flow, defect tolerance is performed once and the same recovered (defect-free) set of resources are used for all applications. In the proposed flow, almost all design steps remain unaware of the existence and the location of defects within the nano-chip. The key idea in the proposed defect-unaware flow is to identify universal defect-free subsets of resources within the original partially-defective nano-chip.

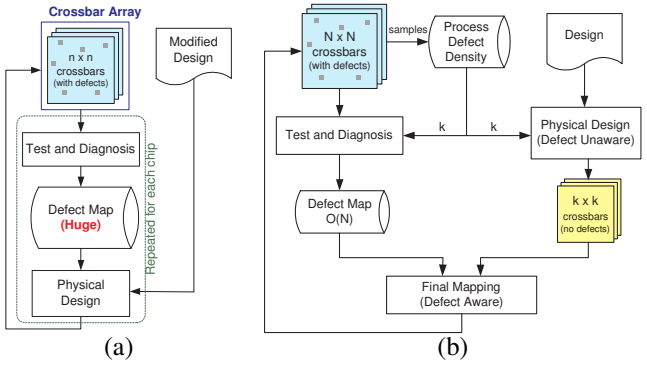

Fig. 6. (a) Traditional defect-aware design flow (b) The proposed defect-unaware design flow

V. Conclusion

The main objective of our project is developing a complete synthesis methodology for nanoscale switching crossbars that leads to the design and construction of an emerging computer. To achieve this objective, we follow a roadmap with sub-objectives. In this paper, we present our preliminary results corresponding to the first two sub-objectives that are 1) Finding optimal nano-crossbar sizes, modelling, and optimization, and 2) Achieving reliable computing. As future work, we have the sub-objectives: 3) Implementing arithmetic and memory elements by considering reliability, area, delay, and power dissipation of the crossbars, and 4) Realizing a nano-crossbar based synchronous state machine (SSM) by integrating arithmetic and logic elements as well as using technology parameters.

VI. Acknowledgments

This work is supported by the EU-H2020-RISE project NANOxCOMP # 691178 and the TUBITAK-Career project # 113E760.

References

[1] "Overall Technology Roadmap Characteristics," International Technology Roadmap for Semiconductors, Tech. Rep., 2010, retrieved 2013.

[2] M. Altun and M. D. Riedel, "Logic Synthesis for Switching Lattices," *IEEE Transactions on Computers*, vol. 61, no. 11, pp. 1588–1600, 2012.

[3] M. Altun and M. Riedel, "Lattice-based computation of boolean functions," in *Proceedings of the 47th Design Automation Conference*. ACM, 2010, pp. 609–612.

[4] A. Bernasconi and V. Ciriani, "Dimension-reducible boolean functions based on affine spaces," *ACM Trans. Design Autom. Electr. Syst.*, vol. 16, no. 2, p. 13, 2011.

[5] A. Bernasconi, V. Ciriani, L. Frontini, V. Liberali, G. Trucco, and T. Villa, "Logic synthesis for switching lattices by decomposition with p-circuits," 2016.

[6] A. Bernasconi, V. Ciriani, L. Frontini, and G. Trucco, "Synthesis on switching lattices of dimension-reducible boolean functions," 2016.

[7] A. Bernasconi, V. Ciriani, G. Trucco, and T. Villa, "Using flexibility in p-circuits by boolean relations," *IEEE Trans. Computers*, vol. 64, no. 12, pp. 3605–3618, 2015.

[8] M. Dubash, "Mooreâs Law is Dead, Says Gordon Moore," *Techworld.com*, no. 13, 2005.

[9] G. Gange, H. Søndergaard, and P. J. Stuckey, "Synthesizing Optimal Switching Lattices," *ACM Trans. Design Autom. Electr. Syst.*, vol. 20, no. 1, pp. 6:1–6:14, 2014.

[10] Y. Huang, X. Duan, Y. Cui, L. J. Lauhon, K.-H. Kim, and C. M. Lieber, "Logic gates and computation from assembled nanowire building blocks," *Science*, vol. 294, no. 5545, pp. 1313–1317, 2001.

[11] M. C. Morgul and M. Altun, "Synthesis and optimization of switching nanoarrays," in *Design and Diagnostics of Electronic Circuits and Systems (DDECS), 2015 IEEE International Symposium on*. IEEE, 2015, pp. 161–164.

[12] G. Snider, "Computing with hysteretic resistor crossbars," *Appl. Phys. A*, vol. 80, pp. 1165 – 1172, 2005.

[13] D. B. Strukov and K. K. Likharev, "Reconfigurable nano-crossbar architectures," *Nanoelectronics and Information Technology*, pp. 543–562, 2012.

[14] M. Tahoori, "Application-Dependent Testing of FPGAs," in *IEEE Transaction on Very Large Scale Integrated Circuits* , 2006.

[15] O. Tunali and M. Altun, "Permanent and transient fault tolerance for reconfigurable nano-crossbar arrays," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, DOI: 10.1109/TCAD.2016.2602804*, 2016.

[16] G. M. Whitesides and B. Grzybowski, "Self-Assembly at All Scales," *Science*, vol. 295, no. 5564, pp. 2418–2421, 2002.