Trading Defect Tolerance for Chip Area in Nanotecnology Implementations of Systolic Arrays

Vladimir Ćirić[1], Vladimir Simić[1], Aleksandar Cvetković[2] and Ivan Milentijević[1]
[1] Faculty of Electronic Engineering, University of Niš, Serbia
[2] Faculty of Mechanical Engineering, University of Belgrade, Serbia
Email: {vladimir.ciric, vladimir.simic, milentijevic}@elfak.ni.ac.rs

***Abstract*—New self-assembling techniques used to build nano-scale architecture prototypes have a drawback of being prone to defects and transient faults. Fault and defect tolerance techniques will be crucial to the use of nano-electronics in the future. However, these techniques usually introduce a significant hardware overhead. In these paper we are proposing a method for trading an architecture tolerance on fabrication defects for chip area. The method will be presented using an architecture with generic topology and illustrated on the example of partially defect tolerant bit-plane semi-systolic array. In order to illustrate the method the results of FPGA implementation of completely fault tolerant bit-plane array, and partially fault tolerant bit-plane array will be given.**

I. Introduction

In 1975 Gordon Moore, cofounder of Intel, predicted that the number of transistors that could be placed on a chip would double every two years [1]. However, Moore's law cannot hold forever. Three obstacles stand in the way: the rising costs of fabrication, the limits of lithography, and the size of the transistor. Given the history of the semiconductor industry, most of these issues can probably be solved with current processes. However, there are two significant exceptions. Physical size limitations and astounding costs may require a shift in the fundamental way integrated circuits are fabricated. Many researchers believe this shift will be to nanoelectronics [2].

Unlike deterministic self-assembly, stochastic self-assembly means that chips will be fabricated with methods that allow components to guide each other in constructing a structure with little or no outside intervention [3]. The lack of outside intervention means that fabrication is more prone to defects and no single part can be absolutely relied on to be functional. Even though nanoelectronics device fabrication is in its infancy, it is clear that defect and fault levels will be much higher than current CMOS technology. The exact level of defect densities is unknown, but it is assumed that 1-15% of the resources on a chip will be defective [4], [5].

With defect rates for current VLSI processes in the range of one part per billion [6], manufacturers can afford to discard any chip that is found to be defective. Predicted defect densities in nanoelectronic mean that no chip will be totally defect free. Reliability will have to be handled at the architecture level instead of the device level, what was once a process engineering problem will now become an architectural dilemma [2].

Application of reliability techniques introduces an overhead in chip resources occupation and signal propagation time [7]. However, many implementations of signal processing algorithms can have a defective portions, and still provide acceptable results. Architectures designed having in mind that some parts may be initially defective, for which can be proven during testing that they still produce acceptable results, are called Error Tolerant (ET) architectures [8], [9].

In this paper we are proposing the trade of the architecture's defect tolerance (DT) for chip area by considering a compromise between DT and ET paradigms in architecture design. The goal of this paper is a reduction of chip area by systematic and controlled decrease of the number of fault-tolerant components within the architecture. The price that will be paid for the reduction of area occupation will be computational correctness. The method will be presented using an architecture with generic topology and illustrated on the example of partially defect tolerant bit-plane (BP) semi-systolic array. In order to illustrate the method the results of FPGA implementation of completely defect tolerant bit-plane array, and partially defect tolerant bit-plane array will be given.

II. Defect tolerance vs. Error tolerance

In order to introduce a tradeoff between an acceptable computation correctness and hardware size, in this section we will give a brief overview of defect and error tolerance.

Fault tolerance (FT) is the ability of a system to continue correct operation of its tasks after hardware or software faults occur [10]. Correct operation typically implies that no errors occur at any system output. Defect tolerance (DT) refers to any circuit implementation that provides higher yield than an implementation that is not defect tolerant, for a given level of defects and process variations. Enhancements in this category include redundancy (often in the form of spares) as well as defect avoidance, in the form of layout and circuit design techniques that reduce circuits' sensitivity to fabrication defects and process variations [2], [10].

Traditional hardware and software fault detection techniques have used N-modular redundancy for fault tolerance. Two or more (up to N) copies of hardware and/or software systems independently perform computations. The result is usually obtained by majority voting [10]. If $N = 3$ the technique is called Triple Modular Redundancy (TMR). Error tolerance is an alternative concept. Error tolerant systems neither detect nor correct error. The circuit is said to be an ET, in respect to an application, if it contains defects that cause internal and may cause external errors, while the system that incorporates this circuit produces acceptable results [9].

Hardware N-modular redundancy significantly increases resources usage, such as area and power. Even when only parts of a processor are duplicated, as in the IBM G5, the resource overhead can be substantial [11]. In general, different levels of N-modular redundancy hardware can be expensive in area, which will not be sustainable in the emerging power- and area-constrained design era. Simply employing error detection and correction mechanisms broadly in a chip is not sufficient in todays robust systems. Future systems requires appropriate, cost-effective mechanisms for a more robust system design. In particular, a comprehensive understanding of the vulnerabilities associated with various units on the chip regarding application workload behavior is required. When such information is available, appropriate approaches can then provide efficient reliability protection [11], [12].

By knowing the architectures' vulnerabilities, defect tolerance can be traded for chip area. The aim of the trade is the design of an architecture that consumes less chip area, but can potentially have some errors at the outputs. Such an architecture has two partitions. One partition consists of vulnerable elements protected by DT methods. The other partition can tolerate existence of errors (ET partition). Because of the partitioning, the architecture is called Partially Defect Tolerant architecture (PDT) [13], [14].

III. Partial defect tolerance architecture design procedure

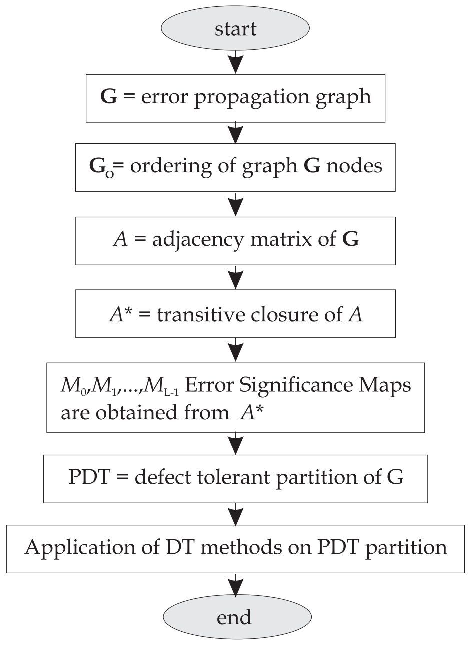
The potential architecture's vulnerabilities are architecture's building blocks from which an error can propagate to the most significant outputs [13]. The significance of the outputs depends on the application of the architecture, and usually is quantified by subjective analysis [8]. The significance of the outputs is not addressed in this paper.

Flowchart of partially defect tolerant architecture design process is shown in Fig. 1. The process will be described on the example of data flow graph of a generic architecture.

Possible error propagation paths within an architecture can be represented by graph $\mathbf{G}$. Given a directed graph $\mathbf{G} = (V, E)$, where $\mathbf{V} = \{v_1, \ldots, v_n\}$ is a finite set of vertices and $\mathbf{E}$ is a finite set of edges; An edge $e_{i,j} \in \mathbf{E}$ is an ordered pair (v_i, v_j), where $v_i, v_j \in \mathbf{V}$ and an edge (v_i, v_j) means that vertices v_i and v_j are connected.

Let the possible error propagation paths of an architecture be represented by the graph $\mathbf{G}$ from Fig. 2. The architecture's vertices are ordered in 2D space using the ordering function $f_0 : \mathbf{V} \to \mathbf{N}^2$ (Fig. 2) [13].

In order to determine all vertices that can propagate an error to particular output, transitive closure of the graph should be obtained. Transitive closure of a graph is a matrix in which element $a^*_{i,j} = 1$ only if there is a path from vertex v_i to the vertex v_j, and 0 otherwise [13]. Transitive closure A^* can be obtained using adjacency matrix A of graph $\mathbf{G}_0$.



start
G = error propagation graph
G0= ordering of graph G nodes
A = adjacency matrix of G
A* = transitive closure of A
M0,M1,...,ML-1 Error Significance Maps are obtained from A*
PDT = defect tolerant partition of G
Application of DT methods on PDT partition
end

Fig. 1. Flowchart of partial defect tolerance design procedure

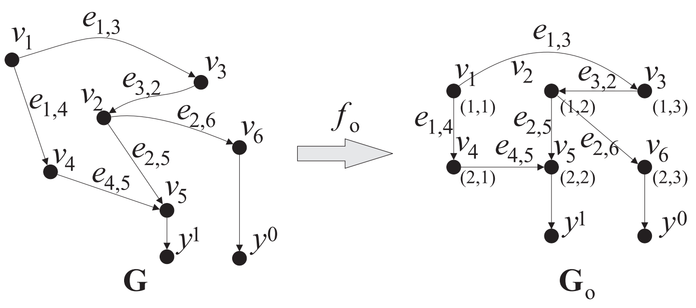


Fig. 2. Error propagation graph $\mathbf{G}$ ordered by the function f_0

The adjacency matrix of the graph $\mathbf{G}_0$ from Fig. 2 is

$$A = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array} \left[\begin{array}{cccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & y_0 & y_1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array} \right], \tag{1}$$

while the transitive closure matrix has column that corresponds to the output y_0

$$Column_{y_0}(A^*) = [1 \;\; 1 \;\; 1 \;\; 0 \;\; 0 \;\; 1 \;\; 1 \;\; 0]^T. \tag{2}$$

From (2) it can be concluded that vertices v_1, v_2, v_3 and v_6 have paths to the output y_0, which is shown in the Fig. 3. The elements of the column (2) written in matrix form using the ordering f_0 are called Error Significance Map (ESM) [13]. ESM that corresponds to the output y_0 of the graph $\mathbf{G}$, denoted as M_0, is shown in Fig. 5. The graph partition marked by ESM M_0 is called Error Significance Set (ESS). ESSs of both ordered and unordered graphs $\mathbf{G}_0$ and $\mathbf{G}$ are shown in Fig. 3.

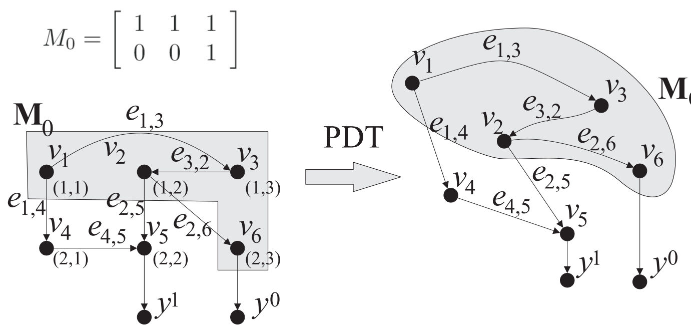


Fig. 3. The Error Significance Matrix M_0 and the Error Significance Set $\mathbf{M}_0$

The union of all ESSs denotes the part of the architecture on which DT methods should be applied. The defect tolerant partition of the architecture is a set of vertices

$$\mathbf{P}_{DT}(\alpha) = \bigcup_{\eta=L-\alpha}^{L-1} \mathbf{M}_{\eta}, \tag{3}$$

where α is the number of the most significant outputs.

IV. Example of partial error-tolerant bit-plane array

A directed graph $\mathbf{G}$ of well-known bit-plane array with $k_C = 2$ coefficients, and coefficient length $m = 2$ is shown in Fig. 4(a). A transitive closure of the graph is obtained and shown in Fig. 4(b) [13]. The dimensions of A^* for the given example are 30x30 (enumerated as $0, 1, \ldots, 29$), because there is a total of 30 nodes within the graph $\mathbf{G}$, including the output nodes y^5, y^4, y^3, and y^2. Fig. 4(b) shows the upper-right corner of transitive closure A^* of the graph $\mathbf{G}$ in Fig. 4(a).

The most significant ESM M_5 can be obtained from the transitive closure A^* by rewriting the 24^{th} column, which corresponds to the output bit y^5, Fig. 4(c). The nodes of the graph from Fig. 4(a) are enumerated as $0, 1, \ldots, 29$, as shown in Fig. 4(c). The elements of the 24^{th} column of A^* are rewritten in the form of a matrix, and mapped to the nodes of graph $\mathbf{G}$ (Fig. 4c). The nodes that have influence on the most significant bit of the result are shaded. Fig. 4(d) shows the error significance map M_5 of the BP array.

If acceptable results of BP array with $k_C = 2$, and $m = 2$ are defined as $Y_{acceptable} = Y_{correct} \pm (2^5 - 1)$, then matrix M_5 in Fig. 4(d) shows the part of the BP array which must be error-free in order for the BP architecture to produce *acceptable results*.

Error significance maps M_4, M_3, and M_2 are developed in the same manner, and are shown in Fig. 4(e). For example, $\mathbf{P}_{DT}(2)$ can be obtained making the cells, marked by M_5 and M_4, defect-tolerant.

V. Implementation results

In order to illustrate newly established tradeoff by partial employment of defect tolerance in the array, we choose a TMR as a DT scheme to be involved in PDT BP array implementation.

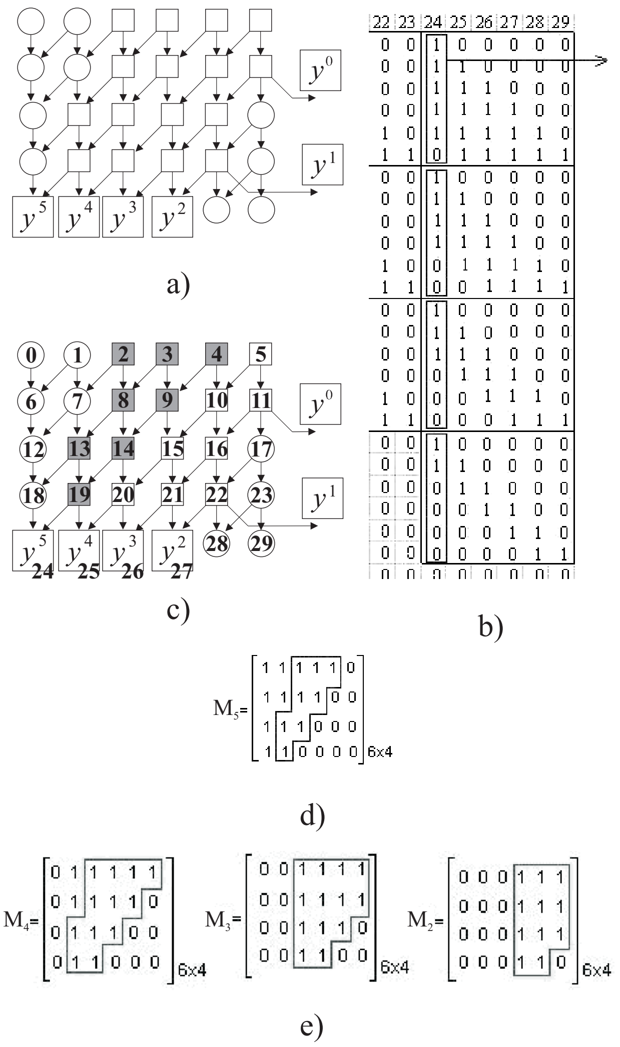


Fig. 4. Implementation example: (a) directed graph $\mathbf{G}$ of bit-plane array with $k_C = 2$ coefficients, and coefficient length $m = 2$; (b) transitive closure A^*; (c) error significance set $\mathbf{M}_5$; (d) error significance map M_5; (e) error significance maps M_4, M_3, and M_2

For the sake of comparison, both basic BP architecture, and partially defect tolerant architecture $\mathbf{P}_{DT}(\alpha), 1 \leq \alpha \leq l_0$, where l_0 is the total number of BP outputs, were described in VHDL and implemented on Virtex4 FPGA. We implemented three arrays of different sizes. Dimensions of the implemented arrays are given in Table I. Values of the arrays widths (l_0), and heights ($m \cdot k_C$) are given in bold. Arrays differ in input word width (n), which implies the different array widths and the same array heights. All three arrays are implemented for parameter α in the range $0 \leq \alpha \leq l_0$.

Implementation results are given in Table II. For each array from Table I there are two columns within Table II. The left column stands for the number of basic cells obtained analytically using the proposed method, while the right column stands for FPGA implementation in equivalent number of kilogates (kG).

Let us explain analytically obtained results for *Arr1*. If there are no error tolerant bits in the output result ($\alpha = 0$), then the array is basic BP array, and the number of the required basic cells is equal to $(m \cdot k_C) \cdot l_0 = 8 \cdot 4 \cdot 16 = 512$. If all bits of the output result are error tolerant ($\alpha = l_0 = 16$), FFT array, then all the basic cells have to be triplicated ($512 \cdot 3 = 1536$, Table II). For $\alpha = 1$, 362 (out of 512) basic cells have to be triplicated, thus $362 \cdot 3 + (512 - 362) \cdot 1 = 1236$ basic cells are required for the array implementation (Table. II).

The number of basic cells, as well as the number of gates required for the implementation of FFT arrays, are given in bold in Table II. These values are used for the calculation of achieved savings in the implementation of error tolerant arrays with $\alpha < l_0$, which are graphically shown in Fig. 5. For both the analytically obtained and FPGA implementation results savings are calculated as

$$Saving = \frac{FFT - \mathbf{P}_{DT}(\alpha)}{FFT} \cdot 100\%. \qquad (4)$$

The Analytically obtained savings are shown with dashed lines, while FPGA implementation results are shown with solid lines in Fig. II. It can be noticed that FPGA implementation results are slightly shifted below the corresponding theoretical results. This is due to the optimization of VHDL synthesis tool.

VI. Conclusion

In this paper we proposed the trade of the architecture's defect tolerance for chip area by considering a compromise between DT and ET paradigms. The method for reduction of chip area by systematic and controlled decrease of the number of defect-tolerant components within the architecture is presented. The price paid for the area reduction is computational correctness. The method is presented using an architecture with generic topology and demonstrated on the example of partially defect tolerant bit-plane semi-systolic array. In order to illustrate achieved tradeoff the results of FPGA implementation of completely fault tolerant bit-plane array, and partially fault tolerant bit-plane array are given.

	k_C	m	n	l_0	$m \cdot k_C$	$\mathbf{P}_{DT}(\alpha)$
Arr1	4	8	8	**16**	**32**	$0 \le \alpha \le 16$
Arr2	4	8	16	**24**	**32**	$0 \le \alpha \le 24$
Arr3	4	8	24	**32**	**32**	$0 \le \alpha \le 32$

TABLE I
PARAMETER SETS FOR IMPLEMENTED ARRAYS

α	Arr1		Arr2		Arr3	
	No. of cells	Impl. [kG]	No. of cells	Impl. [kG]	No. of cells	Impl. [kG]
32	/	/	/	/	**3072**	**82.0**
24	/	/	**2304**	**61.6**	2986	80.2
16	**1536**	**41,4**	2218	59.8	2730	74.7
8	1450	39.6	1962	54.3	2304	65.4
4	1344	37.3	1770	50.1	2048	59.9
2	1274	35.8	1658	47.1	1920	57.1
1	1236	34.9	1598	46.4	1856	55.7
0	512	19.4	768	28.6	1024	37.8

TABLE II
REQUIRED RESOURCES FOR BP ARRAYS IMPLEMENTATION

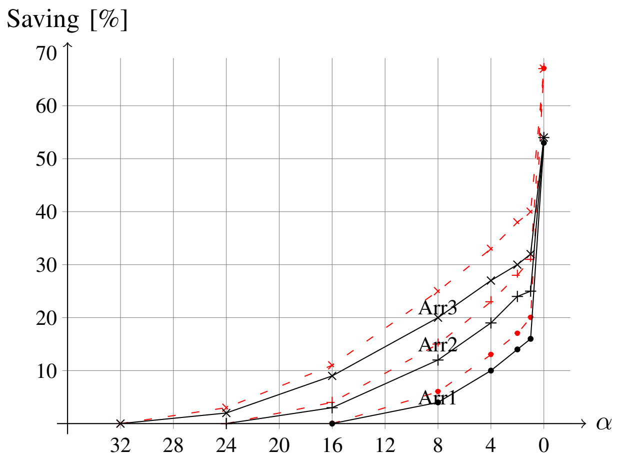


Fig. 5. Saving of the silicon area required for array implementation as the function of bits in the output word that are error-tolerant (α)

References

[1] G.E. Moore, *Cramming more components into integrated circuits*, Electronics, Vol. 38, No. 8, 1965.
[2] Michael Haselman, Scott Hauck, *The future of integrated circuits: A survey of nanoelectronics*, Proceedings of the IEEE, Vol. 98, Issue 1, January 2010, pp. 1138.
[3] D. Whang et al., *Large-Scale Organization of Nanowires Arrays for Integrated Nanosystems*, Nano Letters, Vol. 3, No. 9, 2003, pp. 1255-1259.
[4] Y. Chen et al., *Nanoscale Molecular-Switch Crossbar Circuits*, Nanotechnology, Vol.14, No. 4, 2003, pp. 462-468.
[5] Y. Huang et al., *Directed Assembly of One-Dimensional Nanostructures Into Functional Networks*, Science, Vol. 291, No. 5504, 2001, pp. 630633.
[6] M.R. Stan et al., *Molecular Electronics: From Devices and Interconnect to Circuits and Architectures*, Proc. IEEE, Vol. 91, No. 11, 2003, pp. 1940-1957.
[7] N. Stojanovic, E. Milovanovic, I. Stojmenovic, I. Milovanovic, T. Tokic, *Mapping matrix multiplication algorithm onto fault-tolerant systolic array*, Elsevier, Computers & Mathematics with Applications, Volume 48, Issues 1-2, July 2004, pp. 275289.
[8] M. Breuer, *Multimedia applications and imprecise computation*, In Proceedings on the 8th Euromicro conference on Digital System Design, Euromicro, Porto, Portugal, September 2005.
[9] M. Breuer, S. Gupta, T. Mark, *Defect and error tolerance in the pressence of massive numbers of defects*, IEEE Transactions on Design & Test of Computers, Vol. 21, 2004, pp. 216227.
[10] A. DeHon, *Defect and fault tolerance, in: S. Hauck, A. DeHon (Eds.), Reconfigurable Computing: The Theory and Practice of FPGA-Based Computing*, Morgan Kaufmann, 2008.
[11] J. Rivers, P. Kudva, *Reliability challenges and system performance at the architecture level*, Design & Test of Computers, IEEE, Volume 26, Issue 6, Nov.-Dec. 2009, pp. 62 73.
[12] J. Kolokotronis, V. Ćirić, I. Milentijević, *Error signifcancemap for bit-plane FIR fltering array*, In Proc. 26th International Conference on Microelectronics (MIEL 2008), Vol. 2, Nis, Serbia, May 2008, pp. 429432,
[13] V. Ćirić, J. Kolokotronis, I. Milentijević, *Partial error-tolerance for bit-plane FIR filter architecture*, International Journal of Electronics and Communication, Esevier Science (AEU), Vol. 63, May 2009, pp. 398405.
[14] V. Ćirić, A. Cvetković, I. Milentijević, *Yield analysis of partial defect tolerant bit-plane array. Elsevier*, Int.J., Computers and Mathematics with Applications, Vol. 59(Number 1), January 2010, pp. 98107.