Fault Tolerance of Decomposed PLAs

O. Keren*and I. Levin**
** Bar Ilan University/School of Engineering, Ramat Gan, Israel*
*** Tel Aviv University/School of Education, Tel Aviv, Israel*
kereno@eng.biu.ac.il, ilia1@post.tau.ac.il

Abstract

The paper deals with the fault tolerance of finite state machines (FSMs) implemented by nanoelectronic programmable logic arrays (PLAs). The paper studies a fault tolerant nano-PLA structure, which is based on implementing an initial FSM in a form of three interacting dense PLAs. The paper provides experimental benchmarks results for estimation of fault tolerance properties of the proposed solution. The results indicate a high efficiency of the proposed decomposition approach.

1. Introduction

The reliability of nanoelectronic systems becomes a critical bottleneck when they are utilized for the practical design. Both the manufacturing defects and a lot of possible run-time faults may disturb proper functioning of the nano-PLA circuits. The high probability of both the presence and the appearance of faults in the circuits define specific challenges of the research, to achieve effective use of the nano-PLA technology in the modern logic design.

In the case of nano-PLA, the number of faults is expected to become so large that the conventional ideology of fault detection (to detect a fault as soon as possible) becomes doubtable. Moreover, satisfying such a requirement may even disturb the normal functioning of a nano circuit. To allow the circuit to function even in the presence of a fault, a fault-tolerance technique has to be applied. One well-known way to increase fault-tolerance of a circuit is increasing its redundancy to guarantee functioning of the circuit in presence of faults. Due to the significant increase of fault occurrence in nanoelectronic circuits, and in particular in nano-PLAs, intensive fault tolerant techniques are required. Two main techniques were proposed for providing fault-tolerance to the nano-PLAs: *on-line repair* [12] and *fault masking* [7]. The fault masking-based techniques are preferable, due to their on-line nature and lower hardware penalties.

Fault tolerant techniques for nano-PLAs were studied in [13], [14]. The authors developed a *tautology technique* and a corresponding PLA architecture. Following [13], denote by 'A' the AND plane, and denote by 'O' the OR plane of the initial logic system description. Four different fault tolerant PLA architectures were introduced: A-O, A-A-O-O, A-O-O, A-O-O-A. The area overhead required for each of the tautology architectures may be easily estimated. Generally, these techniques provide better overhead penalty than the well-known triple modular redundancy (TMR) method. However, in many cases the approach [13] is inefficient due to the high area overhead. In the present paper, we discuss a method that allows increasing efficiency of the tautology-based techniques for a widely used class of digital circuits – the combinational part of FSMs.

PLA crosspoints may be with or without devices. PLA including a small percent of devices in both of its planes is considered to have low density. It has been observed ([4], [11], [13]) that, in nano-PLAs, the device missing dominants. Obviously, between two PLAs of identical square, the PLA with a more number of devices (dense PLA) is less fault-tolerant. Consequently, both the number of PLA devices and the PLA area overhead has to be considered as optimization criterion in synthesis of logic circuits by nano-PLA.

All the tautology methods are based on doubling rows and/or columns of PLA planes. These methods basically double empty crosspoints of the PLA, which results in unreasonable overhead. Known methods for synthesis of the tautology based PLAs don't use the density parameter to reduce the resulting area overhead.

In [3], we introduced an approach for synthesis of fault tolerant circuits, which is highly suitable in non-dense PLAs. In this sense, the concept presented in [3] is analogous to the concept of designing immune communication systems. In communication systems, the data is compressed and then redundancy is added to the compressed data in order to protect it against cannel error. In [4], the compression is done by decomposition. The initial PLA is decomposed into a number of high-density component PLAs. These components are then transformed into a fault tolerant form, without excessive doubling of PLA's empty cross points. The approach [3] provides the fault tolerant property with about 15% additional area overhead in respect to the conventional non-fault tolerant implementation of the PLA based FSM. In addition to the area minimization, the proposed in [3] architecture provides an *area/device* trade-off, which allows achieving a solution suitable for a certain case. However, both of the optimization parameters (the area and the number of devices) studied in [3] has to be considered as overhead while the main property of the PLA structure, which is the fault tolerance was not evaluated. In the context of nanotechnology, the fault tolerance of a nano-PLA is the most crucial characteristics of the design. Surprisingly, it was not investigated for known Nano PLA solutions. The present paper is intended to fill this vacuum. In the paper, we continue developing the approach [3] by study fault tolerance characteristics of the proposed PLA structures.

The paper is organized as follows. Preliminaries and basics of the previous research are presented in Section 2. The target Nano PLA architecture is described in Section 3. Experimental benchmarks results and the estimation of the proposed solution are presented in Section 4. Conclusions are provided in Section 5.

2. Preliminaries and Related Work

There are two main challenges in FSM synthesis by PLA: minimization of a number of standard PLA blocks implementing a given FSM, and minimization of the area overhead required for the implementation of the FSM. The majority of the works ([2], [8], [9]) utilizes the following properties of FSMs in order to optimize the resulting solution:

- A system of logic functions, corresponding to a combinational part of FSM is defined by a set of disjoint cubes. This property was used for on-line checking of PLAs ([1], [5], [6], [10]).
- The number of input variables affecting transitions between two certain FSM's states is much smaller than the total number of the FSM input variables. Moreover, the number of input variables affecting all transitions from a certain state is also much smaller than the total number of the FSM input variables.
- The number of possible FSM output vectors is limited and known in advance.
- The number of "ones" in output vectors is much less than a number of "zeros".

The above FSM properties may be interpreted as low density of the corresponding PLA. They were utilized in a number of solutions toward effective implementation of FSMs both as a network of standard PLAs having a minimized number of elements, and as a homogeneous matrix structure with the minimized area.

Consider the FSM described by its true table shown in Table I.

Table I. Table representation of FSM

			X						T			D		Y
x1	x2	x3	x4	x5	x6	x7	x8	t1	t2	t3	d1	d2	d3	Y
1	x	x	x	x	x	x	x	1	1	0	1	0	1	Y8
0	x	x	x	x	x	x	x	1	1	0	1	1	0	Y0
x	x	x	x	x	x	x	x	0	0	0	1	0	1	Y4
x	x	x	x	x	x	1	x	1	0	1	1	0	0	Y4
x	x	x	x	x	x	0	1	1	0	1	0	0	1	Y0
x	x	x	x	x	x	0	0	1	0	1	0	0	0	Y1
1	x	1	x	x	x	x	x	0	1	0	1	1	0	Y2
1	x	0	x	x	x	x	x	0	1	0	1	0	1	Y3
0	1	x	x	x	x	x	x	0	1	0	1	1	1	Y2
0	0	x	x	x	x	x	x	0	1	0	0	1	0	Y0
1	x	x	1	x	x	x	x	0	1	1	0	1	0	Y9
1	x	x	0	x	x	x	x	0	1	1	0	1	1	Y9
0	x	x	x	x	x	x	x	0	1	1	0	0	1	Y5
x	x	x	x	1	x	x	x	1	0	0	0	0	0	Y4
x	x	x	x	0	x	1	x	1	0	0	1	0	1	Y10
x	x	x	x	0	x	0	x	1	0	0	0	0	1	Y4
x	x	x	x	x	x	x	x	1	1	1	0	1	1	Y6
x	x	x	x	x	1	x	x	0	0	1	0	0	1	Y6

The two left hand columns of the table correspond to inputs of FSM combinational portion. The two right hand columns correspond to outputs of the FSM combinational portion. The set of inputs includes external inputs $x_1,\ldots,x_8$ and internal inputs t_1, t_2, t_3, which are the present state variables.

The set of outputs includes the next state variables d_1, d_2, d_3 and external outputs forming the set $Y_1,\ldots,Y_{11}$.

The rows of the table correspond to transitions of the FSM. The input portion of each transition is represented as a cube in of the Boolean space. The external output portion of each transition is denoted by a certain $Y_t \in Y$ corresponding to an output binary vector.

An FSM presented by the table can be directly implemented by the PLA structure. Rows of the PLA are defined by Boolean cubes of a small rank. This leads to a low density of the PLA. At the same time, there are some places of high density. Such places correspond to the binary code of the FSM states.

3. Dense architecture for Nano-PLA

The general structure of the dense PLA architecture is presented in Fig. 1.

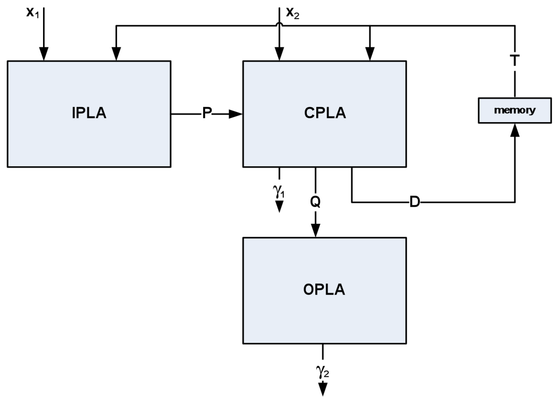

Figure 1. Dense PLA architecture

Assume that the combinational part of the initial FSM implements the transformation $(X \cup T) \Rightarrow (Y \cup D)$, where: $X = \{x_1,\ldots,x_L\}$; is the set of input variables, and $Y = \{y_1,\ldots,y_N\}$; is the set of output variables,

Let the sets of the input and the output variables be divided into two disjoint sets (X_1, X_2), and (Y_1, Y_2) respectively. That is,

$$X = X_1 \cup X_2,\ X_1 \cap X_2 = \varnothing\ ,$$

$$Y = \Upsilon_1 \cup \Upsilon_2,\ \Upsilon_1 \cap \Upsilon_2 = \varnothing\,.$$

Let $T = \{t_1,\ldots,t_R\}$; be the set of present state variables and let $D=\{d_1,\dots,d_R\}$ be the set of the next state variables.

Two main transformations form the dense architecture: a) the transformation of input variables, which can be referred to as input compression, and b) the transformation of output variables, which can be referred to as output decoding. Both of the transformations are implemented by corresponding PLAs. As a result, the final dense structure comprises three PLAs: an input transformation PLA (IPLA), a core transformation PLA (CPLA) and an output transformation PLA (OPLA). The two transformations, IPLA and OPLA, perform the compression, while CPLA implements the functionality of the initial FSM.

The transformations performed by the PLAs are the following:

1. Inputs transformation $(X_1 \cup T) \Rightarrow P$.
2. Core transformation:
   $(X_2 \cup P \cup T) \Rightarrow (Q \cup \Upsilon_1 \cup D)$;
3. Outputs transformation: $Q \Rightarrow \Upsilon_2$.

Two additional sets of auxiliary variables are introduced into the dense architecture: $P=\{p_1,\dots,p_K\}$ is the set of output variables of IPLA (auxiliary input variables of the CPLA), and $Q=\{q_1,\dots,q_M\}$ is the set of output variables of CPLA (auxiliary input variables of the OPLA).

1.2 Inputs transformation PLA

Let $X(a_i)$ be a set of input variables that determine the transitions from state a_i. In our example:

$X(a_1)=\{x_1\}$; $X(a_5)=\{x_4,x_6\}$; $X(a_2)=\varnothing$;
$X(a_6)=\{x_5,x_7\}$; $X(a_3)=\{x_7,x_8\}$; $X(a_7)=\varnothing$;
$X(a_4)=\{x_1,x_2,x_3\}$; $X(a_8)=\{x_6\}$.

The idea of transformation of the original input variables can be formulated as a problem of replacing the set of variables X with a smaller set of new variables P. The number of variables in the set P may be smaller or equal to the maximal number of variables in sets $X(a_i)$. In our example: $|P|\le 3$. The value of the P is a function of the input variables and of state variables.

1.3 Outputs transformation PLA

The outputs transformation PLA transforms (decodes) the set of auxiliary variables Q back to the subset of the original output variables Υ_2.

In our example, there are 11 output vectors denoted as $Y_0,\mathrm{K}\ ,Y_{11}$. Let the on-sets value in each of vector be:

$Y_0=\varnothing$; $Y_1=\{y_2,y_5,y_{10}\}$;$Y_2=\{y_3,y_4\}$;$Y_3=\{y_1,y_3,y_4\}$;
$Y_4=\{y_{10},y_{11}\}$;$Y_5=\{y_6,y_8\}$;$Y_6=\{y_1,y_3\}$;
$Y_7=\{y_9,y_{14}\}$;$Y_8=\{y_7\}$;$Y_9=\{y_6,y_{13}\}$;$Y_{10}=\{y_{12}\}$

For example, the output Y_4 is the binary vector (00000000011000).

Let $K(Y_j)$ be a code (binary vector) associated with the Y_j-set, and let B_j be the corresponding minterm. Each of Y_j-sets can be mapped to minterm B_j in the variables $q_1,\dots,q_M$ such that $B_j=1$, when Y_j-set is activated, otherwise $B_j=0$. The output variables $y_i\ (i=1,\dots,N)$ can be expressed as a sum of minterms in the variables $q_1,\dots,q_M$. In our example:

$y_1=B_3+B_6$; $y_2=B_1$; $y_3=B_2+B_3+B_6$;
$y_4=B_2+B_3$; $y_5=B_1$; $y_6=B_5+B_9$; $y_7=B_8$;
$y_8=B_5$; $y_9=B_7$; $y_{10}=B_1+B_4$; $y_{11}=B_4$;
$y_{12}=B_{10}$; $y_{13}=B_9$; $y_{14}=B_7$.

Obviously, these expressions can be implemented by a PLA structure.

The set Υ_1 consists of all the output variables produced by single product terms. The remaining output variables form the set Υ_2. The output variables of Υ_2 define a set of punctured Y_j-sets. In our example, $\Upsilon_2=\{y_1,y_3,y_4,y_6,y_{10},y_{11},y_{13}\}$ and $\Upsilon_1=Y\setminus\Upsilon_2$. Therefore the punctured Y_1-set corresponding to the original Y_1-set equals to $Y_1=\{y_{10}\}$.

Each punctured Y_j-set is associated with a certain code. The number of the coded bits (denoted by M) is $M=\log_2\left|\{Y_j\}\right|$. Since $M\le N$, the PLA performs compression. In our case: $\left|\{Y_j\}\right|=8$, hence each code is represented by a minterm of the variables q_1,q_2,q_3, clearly 3<14.

1.4 Core PLA

The core transformation PLA implements the Core FSM, namely, the transformation $(X_2 \cup P \cup T) \Rightarrow (Q \cup \Upsilon_1 \cup D)$. The Core FSM differs from the initial FSM by its inputs and outputs. It uses the auxiliary P variables instead of the input variables X_1, and the auxiliary Q variables instead of the output variables Υ_2. The transition functions and the next state functions of the core FSM remain the same as in the initial FSM. The core FSM for our example is presented by Table II.

Table II. The Core FSM

	T			P		X		D		Q
t1	t2	t3	p1	p2	p3	x3	d1	d2	d3	Q
1	1	0	1	x	x	x	1	0	1	B8
1	1	0	0	x	x	x	1	1	0	B6
0	0	0	x	x	x	x	1	0	1	B4
1	0	1	x	1	x	x	1	0	0	B4
1	0	1	0	1	x	x	0	0	1	B6
1	0	1	0	0	x	x	0	0	0	B1
0	1	0	1	x	1	1	1	1	0	B2
0	1	0	1	x	0	0	1	0	1	B3
0	1	0	0	1	x	x	1	1	1	B3
0	1	0	0	0	x	x	0	1	0	B6
0	1	1	1	1	x	x	0	1	0	B9
0	1	1	1	0	x	x	0	1	1	B9
0	1	1	0	x	x	x	0	0	1	B6
1	0	0	x	1	x	x	0	0	0	B4
1	0	0	1	0	x	x	1	0	1	B10
1	0	0	0	0	x	x	0	0	1	B1
1	1	1	x	x	x	x	0	1	1	B6
0	0	1	x	1	x	x	0	0	1	B6

The core FSM can be implemented by the PLA structure directly. It is easy to see that the core transformation PLA is a dense matrix. The resulting scheme for example has the total area equals 479 cross point. The direct matrix implementation requires 608 cross points. So, using the dense scheme provides 22% area reduction.

4. Experiments and Discussion

4.1 Cost in area and number of devices

Three main characteristics are commonly used as criteria in synthesis by Nano-PLAs: the fault tolerance, the required area and the total number of devices. We have shown in [3] that while known methods require more than 100% additional area and devices, our method requires just about 15% additional area.
The experimental results reported in [4] are summarized in Tables III (area overhead) and IV (devices overhead). The first two columns of the tables III and IV are: 1) the overhead of the non- fault tolerant FSM implementation, 2) the overhead for the TMR scheme. The last four columns correspond to the four tautology schemes respectively.

Table III. Benchmark results for average area overhead

	Org	TMR	A-O	A-O-O	A-A-O-O	A-O-O-A
V_{dir}	1	3.026	2.91	2.006	2.0255	3.107
V_{denser}	0.575	1.742	1.98	1.154	1.179	1.482

Rows of Tables III and IV correspond to average overheads for the direct and dense solutions normalized by the corresponding overheads required for the direct PLA implementation of the original FSM.

Table IV Benchmark results for average devices overhead

	Org	TMR	A-O	A-O-O	A-A-O-O	A-O-O-A
W_{dir}	1	3.185	3.332	2.041	2.184	2.792
W_{dense}	1.103	3.441	3.711	2.235	2.423	2.994

The benchmark results show that all the fault tolerant FSMs that were implemented according to the dense architecture have significantly lower area overhead than the FSMs implemented by conventional fault-tolerant techniques. For one and the same fault-tolerant technique, the proposed method provides the area overhead reduction of about 50%. However, it requires a certain amount of additional devices to be introduced into the resulting scheme. For the majority of the benchmarks, the additional devices overhead does not exceed 10%, which is a small penalty for achieving the above area reduction.
In what follows we use a combined complexity measure – an Area-Devices-Factor:

$$ADF = Area \times Devices\,.$$

Obviously, a low ADF is preferred. Figure 2 shows the average factor (over the set of benchmark functions) for each the conventional implementation and for the proposed approach for each fault tolerance technique.

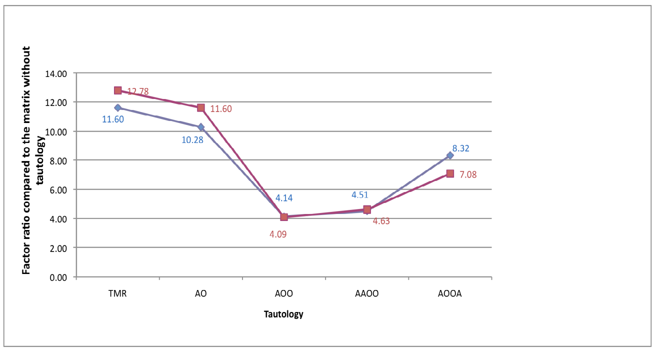

Figure 2. Area-Devices factor. Direct (red line) and Dense implementations (blue line)

It is clear from the figure, that A-O-O and A-A-O-O are the preferable techniques for both, conventional and dense scheme.

4.2 Performance in the presence of fault-events

We call a *fault-event* an event in which some of devices are missing. We assume that the missing devices are spread all over the system's area and that they are not concentrated in a single matrix. There is a difference between our definition for fault-events and the conventional definition. Our definition, which allows missing elements in **all parts** of the system, implies that each matrix may produce an erroneous output. The conventinal concurrent error detection techniques, e.g. the TMR, assume that a fault event effects a **single sub-structure**. In particular, the TMR scheme is designed to cope with faults in which the missing elements are concentrated in one of the three 'copies' of the original structure. Thus only a single part of the system produces erroneous output. Under this assumption, there is no residual error, that is, all the errors caused by missing devices are correctable. In this paper, we study the behavior of fault tolerant systems when there is no restriction on the location of the missing devices. We find this model more suitable for faults in nano-structures.

A fault event affects the output of the overall system. In the presence of a fault event, and depending on value of the system inputs, one of the following cases may happen

1. The fault event will not distort the output. That is, the system will produce a correct output.
2. The fault event will cause an undetectable error at the output. That is, there will be no indication in none of the sub-matrices that a fault event has occurred. Experimental results show that the probability of undetected error in the presence of 10-20% missing devices is about 3%.
3. The fault event will cause errors that can be corrected by the fault tolerant system.
4. The fault event will cause an error that cannot be corrected by the fault tolerance system. In this case the output will be erroneous. The probability of having an erroneous output is referred to as the residual error probability.

The immunity of a tautology-based fault tolerant scheme depends on its structure. Figure 3 shows the average (over the set of benchmark functions) residual error probability, in the presence of fault events in which 10% of the total devices are missing. The missing devices are uniformly distributes over all the system. It is clear from the figure that A-O and A-O-O outperform the TMR. However, the A-A-O-O and the A-O-O-A solutions have higher residual error than the TMR. Their poor immunity results from their structure: The original PLA consists of a single AND matrix and a single OR matrices. The four tautologies consist of AND and OR matrices, some of the matrices are diagonal matrices. Diagonal AND/OR matrices are more vulnerable to missing devices than dense matrices Table V shows the structure of the four schemes. The results, shown in Figure 3, support the hypothesis that there is a correlation between the number of the diagonal matrices and the residual error probability.

Table V. Structure of fault tautology based schemes

Scheme	Structure			
	AND	Diagonal AND	OR	Diagonal OR
original	1	-	1	-
TMR	3	-	3	-
A-O	4	-	2	-
A-O-O	2	-	2	1
A-A-O-O	2	1	2	1
A-O-O-A	2	1	4	2

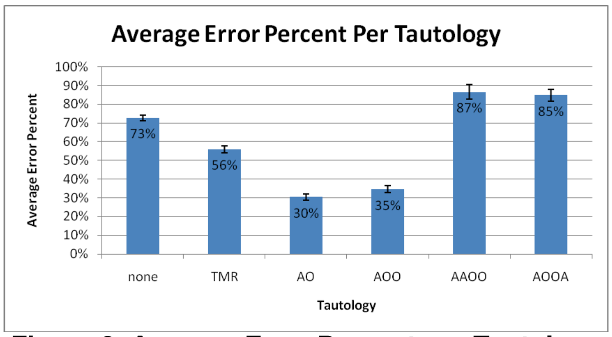

Figure 3. Average Error Percent per Tautology Scheme

Figures 4 and 5 show the residual error probability at the presence of 10% uniformly distributed missing devices for four benchmark functions. These two figures correspond to the conventional approach and the dense PLA approach, respectively. The residual error is shown as a function of the ADF. It is clear from the figures that the A-O scheme achieves the lower error probability. The A-O-O has both, low implementation cost and a low residual error probability. Notice that the original scheme has the lowest ADF, however, about 70% of its outputs are erroneous in the presence of 10% missing devices.

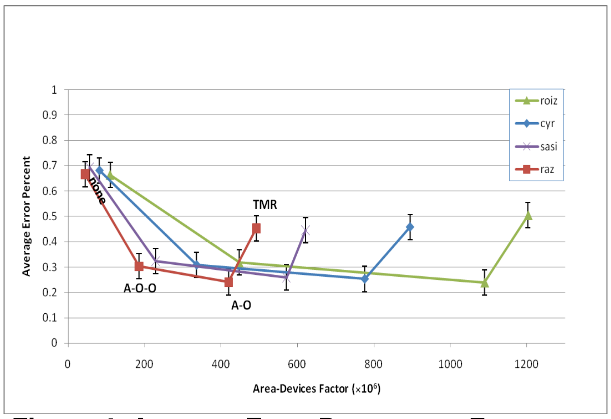

Figure 4. Average Error Percent vs. Factor per benchmark for Direct implementation

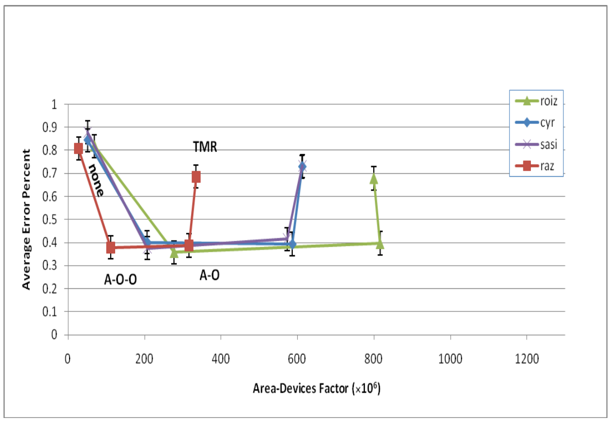

Figure 5. Average Error Percent vs. Factor per benchmark for dense implementation

Another important parameter is the immunity to different types of faults events. In this paper we consider four models:

1. Uniform distribution of missing devices: X percents of the total number of devices are missing. The missing devices are uniformly distributed over all the matrices.
2. Missing devices in columns: X percents of the columns are disconnected, i.e. carry no devices.
3. Missing devices in an entire row.
4. Missing devices in blocks: X percents of the total number of devices are missing. The missing devices are arranged in square blocks of size 10x10, the blocks are uniformly distributed over the total area.

The residual error probabilities of the density based fault tolerant schemes in the presence of error events which induce 5%, 10% and 20% missing devices are presented in Figure 6. The figure shows the average (over the set of benchmark functions) residual error probability for each of the above fault models. The experimental results show that on average the A-O and A-O-O schemes provide better immunity than the TMR.

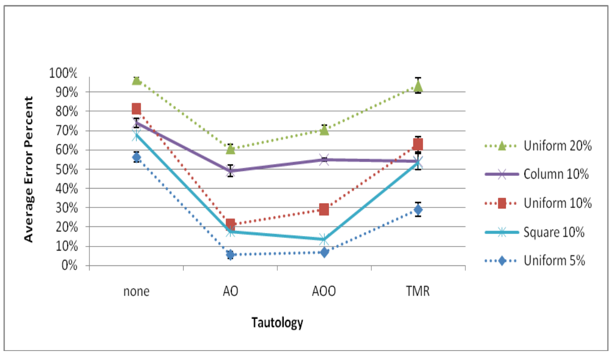

Figure 6. Comparison of Different Fault Models

The quality of a fault tolerant Nano PLA were evaluated by considering three criteria: the fault tolerance property, the total area and the total number of devices. The experimental results indicate that dense implementation combined with AOO tautology is the most suitable in comparison with the other options.

5. Conclusions

During the last years, a number of techniques for synthesis of nanoelectronic PLAs were proposed. The main challenge in the synthesis of nano PLAs is providing a high level of fault tolerance without increasing the hardware overhead. Tautology-based fault masking schemes are often used for this purpose.

Notwithstanding a number of attempts to reduce overhead of the tautology-based schemes, nobody investigated fault tolerance of such schemes. We have tried to fill this vacuum by our study. Specifically, we have investigated an FSMs implementation performed by Nano PLAs. Specific properties of FSMs can be utilized to provide efficient fault tolerant solutions. One of possible solutions is a so-called "dense PLA architecture", which is based on transforming both input and output variables of the initial FSM into smaller sets. Such transformations lead to an architecture consisting of three PLA portions having the "dense" property. We have investigated the "dense architecture" from the point of its area overhead, devices' overhead as well as its fault tolerance.

The obtained results demonstrate the high potential of our approach both from the practical and the theoretical points of view. From the practical point of view, we have shown that the dense architecture provides high fault tolerance without significant overheads. From the theoretical point of view, our approach opens a way for new studies that consider a nanoelectronic scheme as a communication system where compression and coding are applied in order to improve the system performance.

6. References

[1] Astaf'ev, M., Levin, I., Matrosova, A., Sinelnikov, V. (2002). "Self-Testing Automaton Networks: Their Design in Programmable Logical Matrices", Automation and Remote Control, 63(10), 1637-1652.

[2] Baranov S., "Logic Synthesis For Control Automata", Kluwer Academic Publishers, Norwell, MA, USA, Pages: 408, 1994.

[3] Baranov S., Levin I., Keren O., Karpovsky M. "Designing Fault Tolerant FSM by Nano-PLA", 15th IEEE International On-Line Testing Symposium, IOLTS 2009, Sesimbra-Lisbon, Portugal, 229-234.

[4] Chen Y., Jung G. Y., Ohlberg D., Stewart D. R., Jeppesen J. 0., Nielsen K. A., Stoddart J. F. and Williams R. S., "Nanoscale Molecular-switch Crossbar Circuits", Nanotechnology, vol. 14, pp. 462-468, 2003.

[5] Fuchs W. K., Chen C. R., Abraham J. A. "Error Detection in Highly Structured Logic Arrays", IEEE Journal of Solid-State Circuits, Vol. 22, n. 4, pp. 583-594, August 1987.

[6] J. Khakbaz, E. J. McCluskey, "Concurrent Error Detection and Testing for Large PLA's". IEEE Journal of Solid-State Circuits, Apr 1982, Volume: 29, Issue: 4, pages: 756- 764.

[7] Lala, P.K., Tao, D.L. "On fault-tolerant PLA design", IEEE Proceedings Southeastcon '90, vol.3, 1990, pp. 945-947.

[8] Levin I. (1986). "Decompositional Design of Automata Based on PLA with Memory", Automatic Control and Computer Sciences.

[9] Levin, I. (1987) "Hierarchical Model of the Interaction of Microprogrammed Automata",. Automatic Control and Computer Sciences, 21(3), 67-73, Vol. 20, No. 2, 61-68.

[10] Levin I., Karpovsky M., "On-line Self-Checking of Microprogram Control Units", (1998) 4-th IEEE International On-line Testing Workshop, Capri, Compendium of papers, 153-159.

[11] H. Naeimi and A. DeHon, "A Greedy Algorithm for Tolerating crosspoints in Nano PLA design", Proceedings IEEE International Conference on Field-Programmable Technology, pp. 49-56, 2004.

[12] Nikolic, K. Sadek, A. Forshaw, M. "Architectures for reliable computing with unreliable nanodevices". Proceedings of the 2001 1st IEEE Conference on Nanotechnology, 2001, pp. 254-259.

[13] Tahoori M. B., D"efect Tolerance in Crossbar Array Nano-Architectures", In: Emerging Nanotechnologies. Test, Defect Tolerance, and Reliability, Series: Frontiers in Electronic Testing, Vol. 37, M. Tehranipoor, (Ed.).

[14] Wenjing Rao, Alex Orailoglu, Ramesh Karri, "Logic Level Fault Tolerance Approaches Targeting Nanoelectronics PLAs", Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE '07, 2007 pp. 1 – 5.

[15] Wenjing Rao, Alex Orailoglu, Ramesh Karri, "Fault Tolerant Approaches to Nanoelectronic Programmable Logic Arrays", 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2007, pp. 216 –224.