ICIC Express Letters
Volume **11**, Number **7**, July **2017**

 ISSN 1881-803X
pp. **1167–1174**

A FAULT TOLERANT TECHNIQUE FOR NANOCOMPUTERS: XOR MULTIPLEXING

Ming Diao[1], Lianhua Yu[1] and Xiaobo Chen[2]

[1]College of Information and Communication Engineering
[2]College of Automation
Harbin Engineering University
No. 145, Nantong Street, Harbin 150001, P. R. China
{ ruying0714; cxiaobo }@hotmail.com

Received January 2017; accepted April 2017

Abstract. *In emerging nanotechnologies, due to the manufacturing process, a significant percentage of components may be unreliable. In order to make systems based on unreliable components reliable, the design of fault tolerant architectures will be necessary. This paper presents a fault tolerant technique for future nanocomputers, namely, XOR/XNOR multiplexing. It is based on a massive duplication of imperfect devices and randomized imperfect interconnections. Dependent on the stage number, the system can function as XOR or XNOR. Bifurcation theory is used to analyze fault tolerant ability of the system and the results show that XOR/XNOR has a high fault tolerant ability. Similar to NAND multiplexing, this fault tolerant technique is potentially useful for future nanoelectronics.*
Keywords: Fault tolerant, XOR multiplexing, XNOR multiplexing

1. **Introduction.** With silicon technology scaling, we are inevitably faced with the question of how to build a reliable system out of unreliable components. To tackle this problem, several fault tolerant techniques based on redundancy have been investigated, such as N-tuple modular redundancy (e.g., triple modular redundancy) [1,2] and reconfiguration [3,4]. However, with these techniques alone, high fault tolerance is hard to achieve for nanocomputers. Thus, von Neumann's multiplexing, which essentially treats unreliable components as an integral part of the system, has received attention again [8]. A wealth of papers reporting performance analysis of multiplexing have been published, and almost all of those studies were concentrated on NAND multiplexing [5-11], majority multiplexing [10-14] and NOR multiplexing [15].

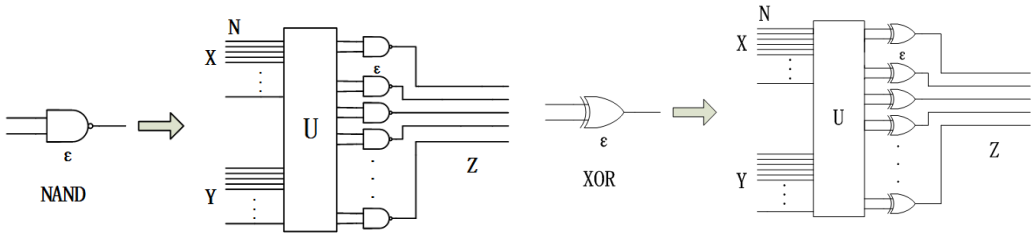
However, none of those multiplexing schemes had ever mentioned how to realize the function of XOR or XNOR. In fact, they are unable to achieve it. As a universal logic gate, XOR and XNOR are widely used in integrated circuits; it should be necessary to study XOR multiplexing or XNOR multiplexing. In this paper, the XOR/XNOR multiplexing for nanocomputers is presented for the first time. The new designed architecture is composed of XOR gates and NAND gates. The XOR gates constitute the execution unit which performs desired function and NAND gates constitute the restoring organs which perform error correction function. The system performance of the architecture is evaluated by studying its fault tolerant ability, which can be defined by gate error threshold and input signal error threshold. The gate error threshold is the maximum gate error probability that the system can still work properly, and the input signal error threshold is the maximum input signal error probability that the system can be tolerant. XOR/XNOR multiplexing has a high fault tolerant ability and depends on the stage number of the system; it can function as XOR or XNOR. Both of XOR and XNOR multiplexing have a

unique feature, and we name it as critical point property, which can indicate the fault tolerant ability of the system.

The rest of paper is arranged as follows. In Section 2, the XOR multiplexing unit and the error distributions in XOR multiplexing unit are presented. In Section 3, we discuss the bifurcation analysis which is followed by Section 4: fault tolerant ability of XOR/XNOR multiplexing system. And Section 5 is conclusion.

2. The XOR Multiplexing Technique.

2.1. **An XOR multiplexing unit.** As shown in Figure 1, XOR multiplexing unit has the same structure with NAND multiplexing unit. Consider an XOR gate with an error probability ε. Duplicate the XOR gate N times, and replace each input of the XOR gate as well as its output by a bundle of N lines.



(a) NAND multiplexing unit (b) XOR multiplexing unit

FIGURE 1. NAND and XOR multiplexing unit

The randomizing unit U performs "random permutation"; through this operation, each input from the first bundle is randomly paired with an input from the second bundle to form the input pair of one of the duplicated XOR gates [6]. Assume that the failure probability ε is a constant. And in this article, we only consider the von Neumann fault. Other types of faults (such as fault model Stuck-at-0 and Stuck-at-1) are not taken into account.

2.2. **Error distribution in XOR multiplexing unit.** The XOR multiplexing unit is shown in Figure 1(b). Assume that (x, y, z) are the probabilities of two inputs being stimulated and of output being stimulated, respectively. If the error probabilities in the two input lines are independent, the probability of the output of XOR gate that is found stimulated is $z = x(1-y) + y(1-x)$ (assuming that the XOR gate is fault free). If each gate has a probability ε of making a von Neumann error, the probability of its output being stimulated is

$$z = (1-\varepsilon)\left[x\left(1-y\right) + y\left(1-x\right)\right] + \varepsilon\left[xy + \left(1-x\right)\left(1-y\right)\right] \tag{1}$$

And the probability of its output to be non-stimulated is $1-z$. If the N XOR gates function independently, then the entire XOR multiplexing unit constitutes a Bernoulli sequence. The distribution of the probability of stimulated output is, therefore, the binomial distribution, the probability of exactly k outputs being stimulated is

$$P(k) = \left(\begin{array}{c} N \\ k \end{array} \right) z^k (1-z)^{N-k} \tag{2}$$

When N is rather large ($N > 1000$), the probability density of k can be obtained now as

$$f\left(k\right) = \frac{1}{\sqrt{2\pi}\sqrt{Nz\left(1-z\right)}} e^{-1/2\left(\left(k-Nz\right)/\sqrt{Nz(1-z)}\right)^2} \tag{3}$$

Therefore, the probability of the number of stimulated outputs of the XOR multiplexing unit could be approximated by a normal distribution when N is extremely large [6].

3. **Bifurcation Analysis of XOR/XNOR Multiplexing System.** Multiplexed systems contain two types of organs. The first type is the executive organ which performs the desired basic operations on the bundles. The second type of organ uses the redundant information available on the input bundle to provide more reliable information on the output bundle. Any logic gate, like NAND gate, NOR gate, AND and OR gates, effectively alternates critical inputs (which produce critical errors) and subcritical inputs (which produce subcritical errors), thereby performing error correction. Among them, NAND gate restoring organ is the first two-layer restoring organ with effective error correction ability. As shown in Figure 2, XOR/XNOR multiplexing system is composed of XOR multiplexing unit and NAND restoring organs. And in order to make system stable, multiply restoring organs would be necessary. Obviously, the system will function as XOR when stage number is odd and function as XNOR when stage number is even.

In order to gain understanding of the associated concept of reliable computation and the system dynamics of probabilistic logics, here, we focus on the operation of individual

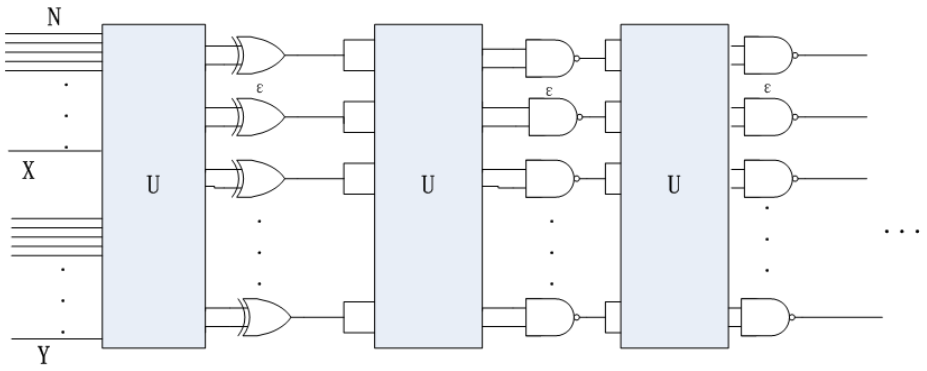


FIGURE 2. XOR/XNOR multiplexing system

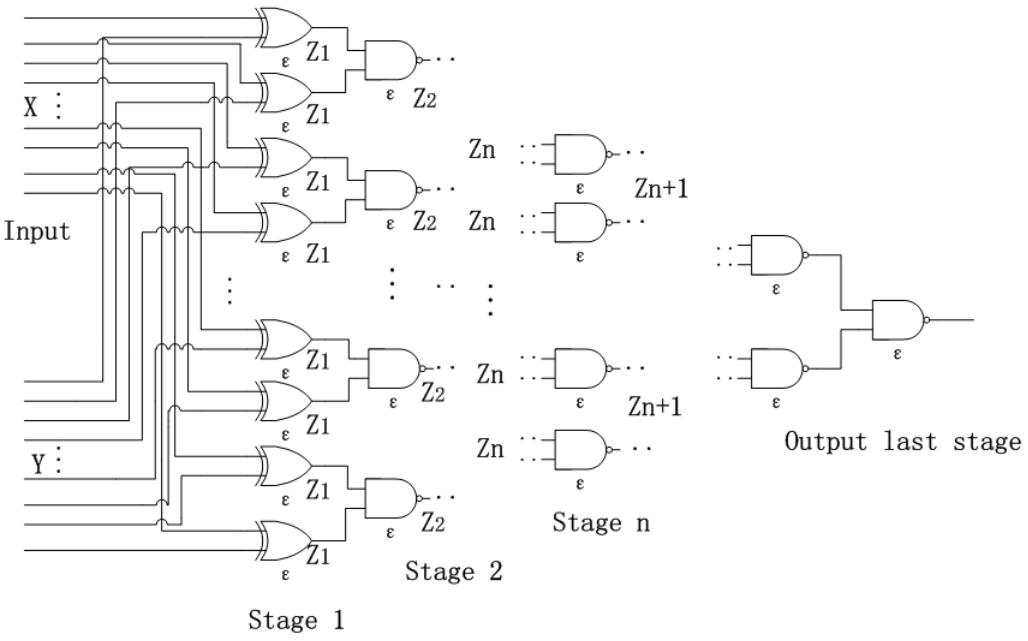


FIGURE 3. Schematic of a full binary tree whose nodes are unreliable two inputs XOR gates and NAND gates with gate error probability ε

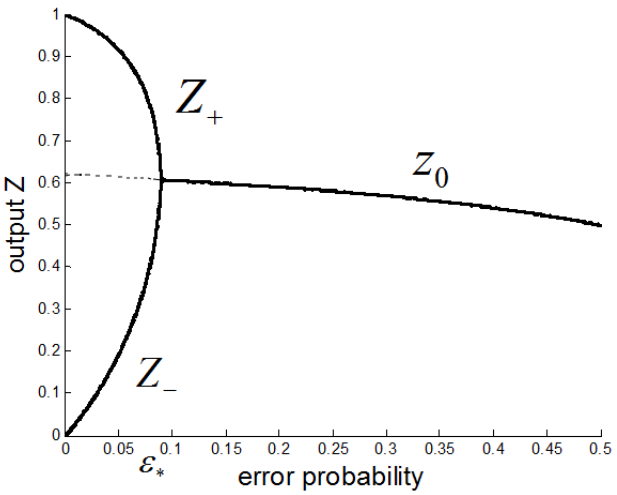


FIGURE 4. Bifurcation diagram for cascaded XOR gates and NAND gates

two-input unreliable XOR gates and NAND gates in a binary tree of cascaded XOR gates and NAND gates as shown in Figure 3 [8]. Assume that the NAND gates have the same error probability to make a von Neumann error while the input and output lines function reliably. Let us denote the probabilities of the two inputs of XOR gate being stimulated by X, Y and further assume that the two inputs are independent. Then the probability of the output of XOR gate being stimulated is

$$Z_1 = (1-\varepsilon)\left[X\left(1-Y\right)+Y\left(1-X\right)\right]+\varepsilon\left[XY+\left(1-X\right)\left(1-Y\right)\right] \tag{4}$$

In the following analysis, we assume that this circuit is a discrete time system. Also assume that all inputs to the XOR gates are independent and each two inputs of those XOR gates have the same probabilities X and Y of being stimulated. This structure guarantees that the inputs to all NAND gates at an arbitrary stage n are also independent and have equal probabilities of being stimulated, which we denote by Z_n [8]. Then for the second stage, the first stage of NAND gates, the probability of the output being stimulated is

$$Z_2 = \varepsilon {Z_1}^2 + (1-\varepsilon)\left(1-{Z_1}^2\right) = (1-\varepsilon)+(2\varepsilon-1)\,{Z_1}^2 \tag{5}$$

For such a construction, (5) reduces to a simple iterative equation

$$Z_{n+1} = (1-\varepsilon)+(2\varepsilon-1){Z_n}^2 \tag{6}$$

In order to find out how the signal propagation in this circuit depends on ε, bifurcation analysis is used to analyze (6) [7]. For any fixed $0 \le \varepsilon \le 1/2$, we choose an arbitrary initial condition X, Y and generate sequences Z_i, $i = 1, 2, \ldots, n, \ldots$, by iterating (6) until it converges to an attractor. Those attractors for the sequences are then plotted against each ε [7,8]. This leads to the bifurcation diagram in Figure 4 ($\Delta\varepsilon = 0.001$). The bifurcation point divides the diagram into two regions, which are: 1) $0 \le \varepsilon < \varepsilon_*$ and 2) $\varepsilon_* \le \varepsilon \le 1/2$. When $\varepsilon_* \le \varepsilon \le 1/2$, the system has a stable fixed point solution, and by solving the equation $z_0 = (1-\varepsilon)+(2\varepsilon-1){z_0}^2$, then we get

$$z_0 = \frac{1-\sqrt{1-4\left(2\varepsilon-1\right)\left(1-\varepsilon\right)}}{2\left(2\varepsilon-1\right)} \tag{7}$$

By stability, it means that for any arbitrary initial inputs condition X and Y, the output Z_n will converge to z_0 when n is large. In other words, in this region, the system no longer functions as XOR or XNOR any more. When $0 \le \varepsilon < \varepsilon_*$, z_0 loses stability, and

the motion is periodic with period 2. We denote those two points by Z_+ and Z_-. And at nth stage, when Z_+ is input, then Z_- would be output and vice versa [7,8]. That means

$$Z_+ = 1 - \varepsilon + (2\varepsilon - 1){Z_-}^2 \tag{8}$$

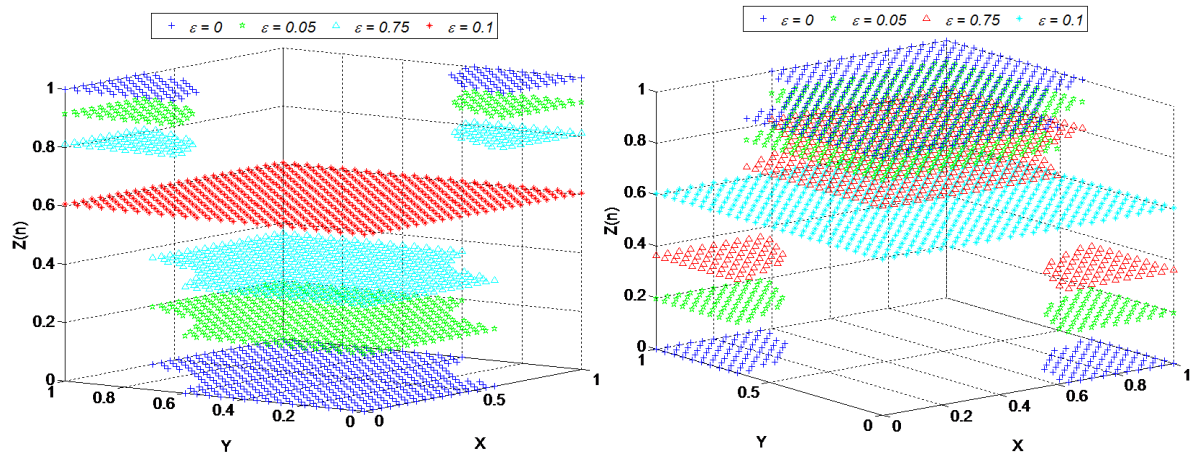
$$Z_- = 1 - \varepsilon + (2\varepsilon - 1){Z_+}^2 \tag{9}$$

From (8) and (9), one obtains

$$Z_\pm = \frac{1 \pm \sqrt{4(1-\varepsilon)(1-2\varepsilon) - 3}}{2(1-2\varepsilon)} \tag{10}$$

Clearly, when $\varepsilon = \varepsilon_*$, we have $Z_+ = Z_-$ and it can be derived that $\varepsilon_* = \left(3 - \sqrt{7}\right)/4 = 0.08856\cdots$. Now it is easy to see that $0 \le \varepsilon < \varepsilon_*$ is the parameter interval where the system functions and ε_* is the gate error threshold. When $\varepsilon > \varepsilon_*$, the outputs converge to stable fixed point z_0 regardless of what the initial inputs are.

Fix error probability ε from 0 to 0.1, and plot the 3-D diagrams of X, Y and Z for both XOR and XNOR, which leads to Figure 5. From Figure 5, we can clearly see the transformation of output from two distinct states to a fixed point when we fixed error probability ε from 0 to 0.1, with ε_* as the "bifurcation" point.



(a) XOR, fixed ε from 0 to 0.1 (b) XNOR, fixed ε from 0 to 0.1

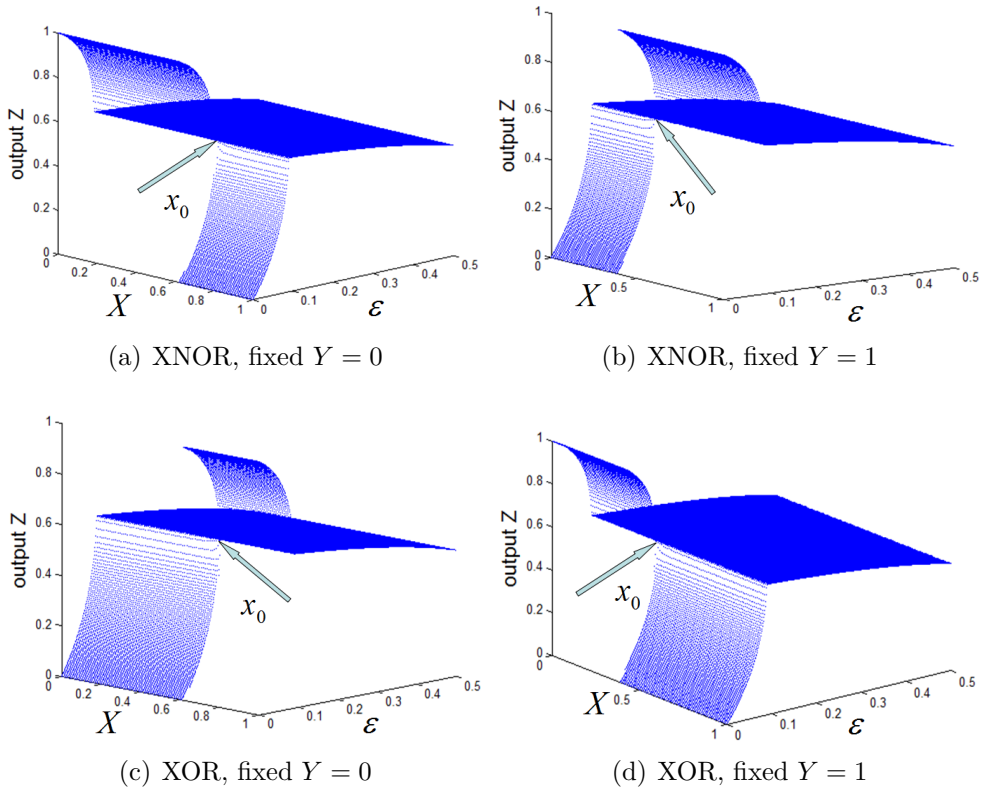
FIGURE 5. 3-D diagrams of XOR and XNOR for X, Y and Z

4. **Fault Tolerant Ability Analysis.** In the last section, we analyzed the tolerant ability of the gate error probability (gate error threshold). Now let us analyze the tolerant ability of input signal error probability (input signal error threshold). In order to map each output probability to a logic state, we need a threshold z_*. According to Figure 4, it is not so difficult to find out the truth that $z_0(\varepsilon_*)$ is a good choice for z_*. It is simple and effective. Substituting $\varepsilon = \varepsilon_*$ into (7) then we have

$$z_0(\varepsilon_*) = \frac{1 - \sqrt{1 - 4(2\varepsilon_* - 1)(1 - \varepsilon_*)}}{2(2\varepsilon_* - 1)} = 0.6076 \tag{11}$$

Below, we shall interpret $[0, z_*)$ as non-stimulated state and $(z_*, 1]$ as stimulated state. Fix the input $Y = 1$ and $Y = 0$, and then we can get 3-D diagrams as shown in Figure 6. Clearly, for XOR and XNOR multiplexing, system has higher fault tolerant ability when inputs are both stimulated or both non-stimulated. Seen from Figure 6, the effectiveness of this threshold is obvious.

Note that for each different fixed Y, a different value of X (here we name it as critical point and denote it by x_0) divides the output into two states when $0 \le \varepsilon < \varepsilon_*$. Take



(a) XNOR, fixed $Y = 0$ (b) XNOR, fixed $Y = 1$

(c) XOR, fixed $Y = 0$ (d) XOR, fixed $Y = 1$

FIGURE 6. 3-D diagrams of XNOR and XOR multiplexing

$Y = 0$ as an example, in the interval $0 \leq \varepsilon < \varepsilon_*$, when $X < x_0$, the output would be non-stimulated (for XOR) or stimulated (for XNOR), and when $X > x_0$, the output would be stimulated (for XOR) or non-stimulated (for XNOR). The calculation of critical point can help us more intuitively understand the fault tolerant ability of system. Since when n is large enough and $0 \leq \varepsilon < \varepsilon_*$, the output only depends on the input condition: input X and Y have the same logic state (both stimulated or both non-stimulated) or have different logic state (one of the inputs is stimulated and the other one is non-stimulated). Let us denote the probability that two input X and Y have different logic state by P_1, and denote the probability that two input X and Y have the same logic state by P_2. So the ratio of P_1 and P_2 will be a key parameter to determine the final output Z_n is larger than z_* or not. X and Y are the probabilities of inputs being stimulated, then $1 - X$ and $1 - Y$ are the probabilities of inputs being non-stimulated. P_1 and P_2 are shown as follows.

$$P_1 = X(1 - Y) + Y(1 - X) \tag{12}$$

$$P_2 = XY + (1 - X)(1 - Y) \tag{13}$$

For XOR (XNOR), if we need the output to be stimulated, then P_1/P_2 must be larger (smaller) than a specific value which is greater than one. Since the output logic state is associated with the output threshold z_*, the specific value will be a function of z_* and the mathematic relation between them is shown below.

$$\frac{P_1}{P_2} = \frac{X(1 - Y) + Y(1 - X)}{XY + (1 - X)(1 - Y)} > \frac{z_*}{1 - z_*} \tag{14}$$

Clearly, $P_1 + P_2 = 1$, hence, Equation (14) is equivalent to

$$P_1 = X(1 - Y) + Y(1 - X) > z_* \tag{15}$$

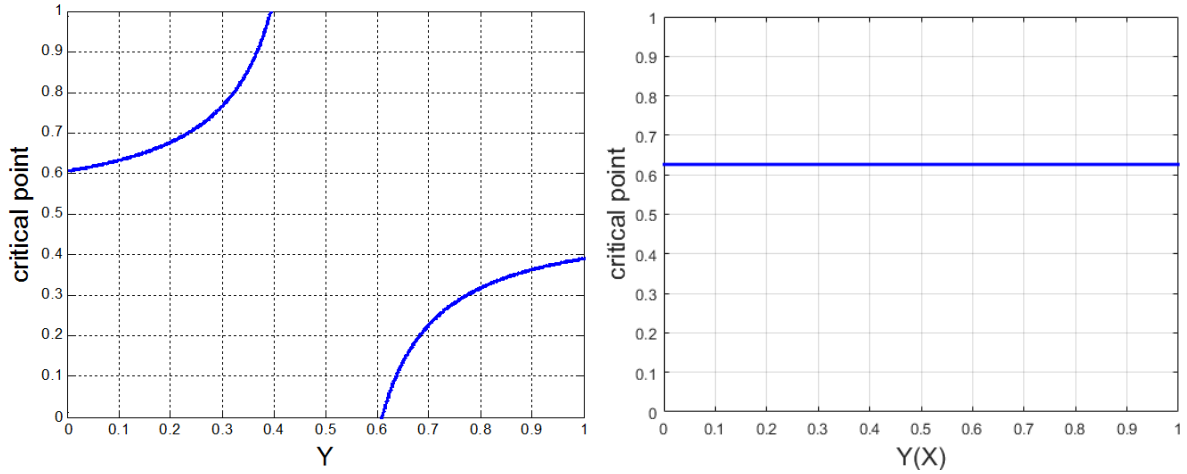
If $P_1 > z_*$, the final output would be larger than threshold (stimulated) for XOR and smaller than threshold (non-stimulated) for XNOR. The inequality becomes

$$P_1 = X(1-Y) + Y(1-X) < z_* \tag{16}$$

The final output would be smaller than threshold (non-stimulated) for XOR and larger than threshold (stimulated) for XNOR. Hence, it is easy to obtain the critical point x_0 for each fixed Y by solving the following equality

$$x_0(1-Y) + Y(1-x_0) = z_* \tag{17}$$

Critical point x_0 is a function of Y; these critical points are then plotted against each Y ($\Delta Y = 0.01$). This leads to Figure 7(a). It shows that the diagram has two regions and for each different Y. Critical point x_0 has different values and there is a parameter interval that makes the system no longer function even though the system is fault free, and the parameter interval approximates $0.3924 < Y < 0.6076$. If the value of one of inputs is in this interval, then the output will always be stimulated for XNOR and non-stimulated for XOR. This property is quite different from NAND multiplexing. In [6-8], they considered the worst scenario of NAND nultiplexing: inputs $x = y$. Under the circumstance, critical point of NAND is a constant, as shown in Figure 7(b), any inputs in the interval $[0, x_0)$ produce an output in the interval $(x_0, 1]$, and vice versa. That means any input condition will produce a valid output.



(a) XOR/XNOR multiplexing (b) NAND multiplexing

FIGURE 7. Critical point against each Y

TABLE 1. Critical points for several fixed Y

	$Y=0$	$Y=0.1$	$Y=0.2$	$Y=0.3$	$Y=0.4$
x_0	0.60760	0.63450	0.67933	0.76900	/
	$Y=0.6$	$Y=0.7$	$Y=0.8$	$Y=0.9$	$Y=1$
x_0	/	0.23100	0.32067	0.36550	0.39240

In order to demonstrate the tolerant ability of input signal error probability of the system more intuitively, we extracted several fixed Y and the corresponding x_0 from Figure 7(a). These lead to Table 1. Let us take $Y = 0.7$ as an example, it can be seen that when input Y has a probability of 70% being stimulated (It means 30% error probability), any stimulated probability smaller than 23.1% of the other input X could be accepted. That is to say the system can tolerate error probabilities of 30% and 23.1% for the inputs Y and X. Other situations are similar so we omit them here. It also can be

obtained that the maximum input signal error probability that the system can tolerate is 0.3924 (39.24%); namely, the input signal error threshold is 0.3924.

5. **Conclusion.** In this paper, we have studied a new fault tolerant architecture: XOR/XNOR multiplexing. The system is expected to work at an acceptable reliability level when inputs have different logic state and expected to work at a much higher reliability level when inputs have the same logic state. Dependent on stage number, the system can function as XOR or XNOR. This architecture is potentially effective in protection against transient faults for systems based on unreliable nanometerscale devices.

Fault tolerant techniques and architectures are indispensable for nanocomputers. In the future work, we will be committed to the study of how to improve the system performance of the proposed multiplexing scheme, such as exploring other restoring organs with better performance. Also, we will study other fault tolerant techniques and focus on hardware redundancy, such as developing new fault tolerant architectures.

Acknowledgment. This work was supported by the National Natural Science Foundation of China (61571149). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] S. C. Anjankar, M. T. Kolte et al., FPGA based multiple fault tolerant and recoverable technique using triple modular redundancy (FRTMR), *Procedia Computer Science*, vol.79, pp.827-834, 2016.

[2] R. Yao, Q. Chen et al., Multi-objective evolutionary design of selective triple modular redundancy systems against SEUs, *Chinese Journal of Aeronautics*, vol.28, no.3, pp.804-813, 2015.

[3] K. Siozios, I. Savidis and D. Soudris, A framework for exploring alternative fault-tolerant schemes targeting 3-D reconfigurable architectures, *International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation*, pp.336-341, 2016.

[4] A. Mukherjee and A. S. Dhar, Choice of granularity for reliable circuit design using dynamic reconfiguration, *Microelectronics Reliability*, vol.63, pp.291-303, 2016.

[5] J. von Neumann, Probabilistic logics and the synthesis of reliable organisms from unreliable components, in *Automata Studies*, C. E. Shannon and J. Mccarthy (eds.), Princeton University Press, 1956.

[6] J. Han and P. Jonker, A system architecture solution for unreliable nanoelectronic devices, *IEEE Trans. Nanotechnology*, vol.1, no.4, pp.201-208, 2002.

[7] Y. Qi and J. B. Gao, Bifurcations and fundamental error bounds for fault-tolerant computations, *IEEE Trans. Nanotechnology*, vol.4, no.4, pp.395-402, 2005.

[8] Y. Qi and J. B. Gao, Markov chains and probabilistic computation – A general framework for multiplexed nanoelectronic systems, *IEEE Trans. Nanotechnology*, vol.4, no.2, pp.194-205, 2005.

[9] G. R. Voicu and S. D. Cotofana, Towards heterogenous 3D-stacked reliable computing with von Neumann multiplexing, *Proc. of IEEE/ACM International Symposium on Nanoscale Architectures*, Los Alamitos, CA, USA, pp.122-127, 2013.

[10] K. Gucwa, On simulation of multiplexed architecture for fault-tolerant nanoelectronic systems, *The 12th IEEE International Conference on Nanotechnology (IEEE-NANO)*, vol.7, no.23, pp.1-4, 2012.

[11] V. Beiu and W. Ibrahim, Devices and input vectors are shaping von Neumann multiplexing, *IEEE Trans. Nanotechnology*, vol.10, no.3, pp.606-616, 2011.

[12] S. Roy and V. Beiu, Majority multiplexing-economical redundant fault-tolerant designs for nanoarchitectures, *IEEE Trans. Nanotechnology*, vol.4, no.4, pp.441-451, 2005.

[13] D. Bhaduri and S. K. Shukla, Reliability evaluation of von Neumann multiplexing based defect-tolerant majority circuits, *The 4th IEEE Conference on Nanotechnology*, pp.599-601, 2004.

[14] D. Bhaduri, S. Shukla et al., Comparing reliability-redundancy tradeoffs for two von Neumann multiplexing architectures, *IEEE Trans. Nanotechnology*, vol.6, no.3, pp.265-279, 2007.

[15] W. Ibrahim, V. Beiu and A. Beg, On NOR-2 von Neumann multiplexing, *The 5th International Design and Test Workshop*, pp.67-72, 2010.