Fault Tolerance Design by Accurate SER Estimation for Nano-Scale Circuits

YU CHANGHONG
Department of Information & Electronic Engineering
Zhejiang Gongshang University
310018 Hangzhou
CHINA
hangzhouyu@gmail.com

Abstract: - As the transistor sizes continue to shrink, quantum effects will significantly affect the circuit behavior. The inherent unreliability of nano-electronics will have significantly impact on the way of circuits design, so defects and faults of nano-scale circuit technologies have to be taken into account early in the design of digital systems. Fault-tolerant architectures may become a necessity to ensure that the underlying circuit could function properly. In CAD software, a same logic can be made out with different circuits but different design methodology can reach different soft error tolerance ability, so we must find a way to estimate the error rate of the circuit efficiently to make the design more fault tolerant.

In this paper, a new way to fault tolerance design in nano-scale circuit by accurate soft error rate (SER) estimation is proposed. Transform matrix is used for SER computation and a design criteria is then proposed. Simulation results show that the proposed transform matrix model is effective for nano-scale circuits and the criteria delivered is suitable CAD tools development in nano-system design.

Key-Words: - Nano-system, SER estimation, Transform matrix, Matrix analysis, Condition number, Fault tolerant design.

1 Introduction

As device scale shrinks, traditional CMOS based devices are reaching their physical limits. Quantum effects occurring at the nano-scale devices have been the pullback for further scaling down of CMOS based electronic systems. Integrated circuit (IC) technology scaling has brought forth heightened device sensitivity to a different kind of error, which is called as soft errors or transient errors. Soft errors are caused by external noise or radiation which temporarily affects circuit behavior without permanently damaging the hardware. Soft errors can directly occur on state elements such as memories, flip-flops and latches and change their state. Furthermore, state elements can latch incorrect values propagated from soft errors that occur in combinational and sequential elements. For soft errors in nano systems and circuits are widespread it is gaining increasing attention and it is expected to become as important as directly induced errors on state elements [1].

To deal with soft error (faults or defects) in nano-scale circuits, a variety of new devices have been proposed in recent years, including carbon nanotubes (CNTs)[2,3], quantum-dot cellular automata (QCA)[4,5], resonant tunneling devices (RTDs)[6,7], single electron tunneling (SETs) devices[8,9] etc. These devices differ from CMOS in both structure and functionality with advantages including less power dissipation, smaller dimensions, greater performance, etc. But CNTs and RTDs operate near the thermal limit of $k_B T$ (k_B is the Boltzmann constant, T is room temperature)[10], so they will experience high error probabilities. QCAs have two main sources of error, one is delay, when electrons that store information are lost to the environment and the other is switching error which means when the electrons do not properly switch from one state to another due to background noise or voltage fluctuations[11,12]. The disadvantage with all these different types of nano devices is that the output produced by the nano logic may not be as reliable as CMOS.

Another way to deal with soft errors in nano-scale circuit is fault tolerance design which is firstly proposed in 1950s[13]. This design methodology is always employed for transistor-level, circuit-level and system-level.

Van Neumann initially proposed building fault tolerant design using unreliable components[13], who used majority logic gates as a primitive building block and randomizing networks to prevent clusters of failures from overwhelming the fault tolerance of majority logic. However, the redundance in this design is too much, so the hardware efficiency is very low. .S. Roy and V. Beiu optimized Van Neumann's design and use majority multiplexing-economical redundant fault-tolerant designs for nano-architectures, which eliminates the redundance dramatically[14], but for majority logic gates and nand arrays are used in design, there is still a lot of redundance in the circuits, this methodology is not very widely used.

TMR (Triple Modular Redundancy)[15], NMR (N modular redundancy)[16] and CTMR (cascaded triple modular redundancy)[17] are investigated using majority logic to make decision which signal is used for finally output. But when the soft error occurs in MAJ, the TMR and NMR won't fault tolerance.

Recently, some design methodology based on generalized reliability analysis techniques are proposed for nano-scale circuits design.

Markov random field (MRF) model was proposed initially in [18] and extended in [19, 20]. MRF was developed to support optimization of a set of random variables so that their overall joint probability is a global maximum. This methodology uses global optimization for circuit and has very low redundance, but there is no guideline for researcher to follow this design method.

NANOLAB is proposed in [21, 22] which is a MATLAB based reliability analysis tool that uses probability distribution of signal energy levels and entropy as reliability metrics. Belief Propagation algorithm is investigated that can compute signal energy distributions and entropy at the primary/intermediate outputs and interconnects of combinational circuits.

NANOPRISM is referred in [23] which is a tool built on the probabilistic model checker PRISM that applies Markovian techniques to automatically evaluate performance measures of nano-architectures in the presence of defects and transient faults. T. Rejimon uses probabilistic error model for unreliable nano-logic gates[24] He proposes a framework based on probabilistic transfer matrices (PTMs) that can be used for computing the output probabilities for combinational circuits.

J. Han uses probabilistic gate model for nano-circuits[25], J Chen also uses ensemble dependent matrix to model the nano system's behavior[26]. These methods analyze the behavior of the nano system and make fault tolerance design.

This paper analyzes relationship between input and output of the basic gates for SER estimation. Then we get a transform matrix for SER computation. By matrix analysis, we finally propose an efficient way for SER avoidance in fault tolerance design.

The paper is organized as follows: Section 2 delivers basic idea where we analyze the SER relationship of input and output and transform matrix is proposed. In Section 3, we discuss circuit design by SER analysis, where Frobenius norm is employed and a criteria is proposed for fault tolerance design. Simulation was presented in Section 4 for checking our model and criteria. We finally conclude our presentation in Section 5.

2 Transform Matrix for Circuit

For physical limitations, nano-scale devices have inherent defect, soft errors will occur in most part of the nano-system. By now, there is no good model to describe individual device and system soft error for nano-scale circuit. We use matrix to model input and output signals of the nano-system, the change of the signals in this model can be described as transition matrix.

2.1 Input and Output Analysis

Use nand as an example, as described in Fig. 1, we can see that there are two input signals with one output signal. The output value is determined by facts of input value and work status of nand.

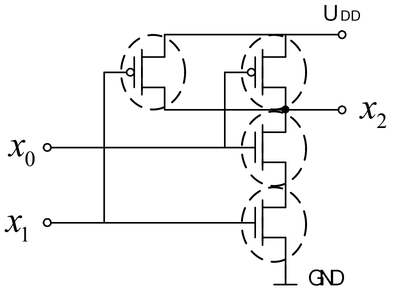


Fig. 1 Determination for nand's output

As all the signals in the gate are continuous, we must make discretization of them to digital signal "0" and "1". Thus at any time, transmission of output signal can be described as Fig. 2.

In Fig. 2 $P_{0|0,0}$ means probability when input state is $x_1 = 0, x_0 = 0$ and output state is $x_2 = 0$ which means nand works incorrectly. Likewise $P_{1|0,0}$ means probability that input state is $x_1 = 0, x_0 = 0$ and output state is $x_2 = 1$ which means the nand works correctly. When nand is in normal operation $P_{0|0,0}$, $P_{0|0,1}$, $P_{1|1,0}$ and $P_{1|1,1}$ should be 0 while $P_{1|0,0}$, $P_{1|0,1}$, $P_{1|1,0}$ and $P_{0|1,1}$ would be 1, otherwise all of them will have value between $[0,1]$. For with all kinds of inputs the output is either $x_2 = 0$ or $x_2 = 1$, then we can get that:

$$\left.\begin{array}{l} p_{0|0,0} + p_{1|0,0} = 1 \\ p_{0|0,1} + p_{1|0,1} = 1 \\ p_{0|1,0} + p_{1|1,0} = 1 \\ p_{0|1,1} + p_{1|1,1} = 1 \end{array}\right\} \qquad (1)$$

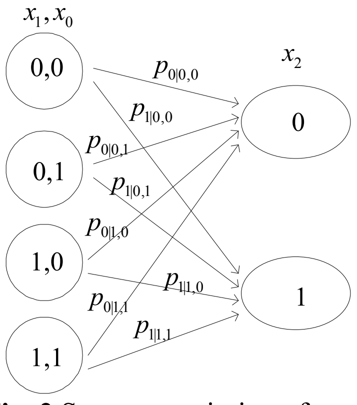


Fig. 2 State transmission of nand

To simplify our discussion, we assume that different inputs work status are the same which incorrect probability is ic and correct probability is c. So it is easy to get:

$$\left.\begin{array}{l} p_{0|0,0} = ic, p_{1|0,0} = c \\ p_{0|0,1} = ic, p_{1|0,1} = c \\ p_{0|1,0} = ic, p_{1|1,0} = c \\ p_{0|1,1} = c, p_{1|1,1} = ic \end{array}\right\} \Rightarrow \left.\begin{array}{l} p_{0|0,0} = 1-c, p_{1|0,0} = c \\ p_{0|0,1} = 1-c, p_{1|0,1} = c \\ p_{0|1,0} = 1-c, p_{1|1,0} = c \\ p_{0|1,1} = c, p_{1|1,1} = 1-c \end{array}\right\} \quad (2)$$

We use $p(x_2 = 0)$ to describe the probability that output is $x_2 = 0$, then $p(x_2 = 0)$ can be calculated as

$$p(x_1 = 0, x_0 = 0) * p_{0|0,0} + p(x_1 = 0, x_0 = 1) * p_{0|0,1} + \\ p(x_1 = 1, x_0 = 0) * p_{0|1,0} + p(x_1 = 1, x_0 = 1) * p_{0|1,1},$$

and $p(x_2 = 0)$ can be described as

$$p(x_1 = 0, x_0 = 0) * p_{1|0,0} + p(x_1 = 0, x_0 = 1) * p_{1|0,1} + \\ p(x_1 = 1, x_0 = 0) * p_{1|1,0} + p(x_1 = 1, x_0 = 1) * p_{1|1,1}.$$

Then $\begin{bmatrix} p(x_2 = 0) \\ p(x_2 = 1) \end{bmatrix}$ can be described as

$$\begin{bmatrix} p_{0|0,0} & p_{0|0,1} & p_{0|1,0} & p_{0|1,1} \\ p_{1|0,0} & p_{1|0,1} & p_{1|1,0} & p_{1|1,1} \end{bmatrix} \begin{bmatrix} p(x_1 = 0, x_0 = 0) \\ p(x_1 = 0, x_0 = 1) \\ p(x_1 = 1, x_0 = 0) \\ p(x_1 = 1, x_0 = 1) \end{bmatrix}.$$

Use symbols X, Y and A as:

$$Y : \begin{bmatrix} p(x_2 = 0) \\ p(x_2 = 1) \end{bmatrix} \Leftrightarrow \begin{bmatrix} y_0 \\ y_1 \end{bmatrix},$$

$$X : \begin{bmatrix} p(x_1 = 0, x_0 = 0) \\ p(x_1 = 0, x_0 = 1) \\ p(x_1 = 1, x_0 = 0) \\ p(x_1 = 1, x_0 = 1) \end{bmatrix} \Leftrightarrow \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}, \text{ and}$$

$A : \begin{bmatrix} p_{0|0,0} & p_{0|0,1} & p_{0|1,0} & p_{0|1,1} \\ p_{1|0,0} & p_{1|0,1} & p_{1|1,0} & p_{1|1,1} \end{bmatrix}$ (for convenience, we rewrite it as $\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \end{bmatrix}$).

Then we can get a simple formula $Y = AX$. We call matrix A, which reveal relationship between input vector and output vector, as *transform matrix*.

As discussed previous, function of different gates or circuits can be described with three essential factors: input vector matrix X, output vector matrix Y and transform matrix A. For example, transform matrix of nand can be represented as $\begin{bmatrix} 1-c & 1-c & 1-c & c \\ c & c & c & 1-c \end{bmatrix}$, and we can use $\begin{bmatrix} 1-c & c & c & c \\ c & 1-c & 1-c & 1-c \end{bmatrix}$ for nor gate while use $\begin{bmatrix} 1-c & c \\ c & 1-c \end{bmatrix}$ for inverter. But how to get the transform matrix when the circuit is complex?

2.2 Analysis for Complex Circuits

Use circuit in Fig. 3 as an example. There are five inputs with two outputs in this circuit, so there are 2^5 different kinds of input combinations, and there will be 2^5 columns for transform matrix. For there are two outputs the transform matrix will have 2^2 rows.

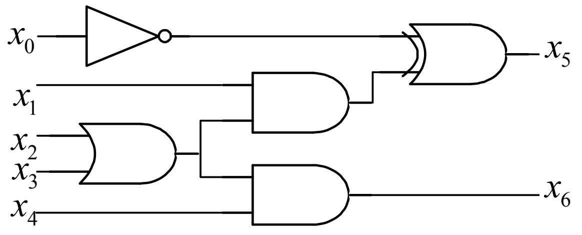


Fig. 3 A circuit for transform matrix computation

As discussed before we have known the transform of nand gate, nor gate and inverter gate, so we can use these elements to construct complex circuit as following steps which is called as "divide and conquer":

Step 1: *Divide* the circuit as serial components as Fig. 4.

We can see that S1, S2, S3, S4, S5 and S6 are six parts which are serial connected, and each part are composed with basic gates and wire. It is easy to get global transform matrix if we can get transform matrix of each part.

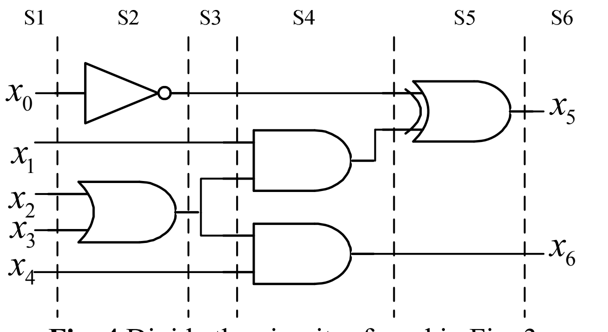
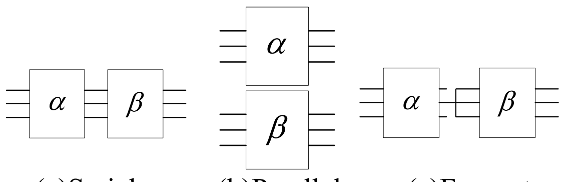


Fig. 4 Divide the circuit referred in Fig. 3

Step 2: (*Conquer*) Get components' transform.

As shown in Fig. 5, logic circuits with different topologies can be constructed through a serial (in which a circuit acts on the outputs of another circuit), parallel and fan-out connections.



(a)Serial (b)Parallel (c)Fan-out

Fig. 5 Topologies of the combinational circuits

As described in [24], for a circuit consisting of two serial sub-circuits α and β, its transform matrix is the product of the two sub-matrices $A = A(\alpha) * A(\beta)$.

The transform matrix for a circuit consisting of two parallel sub-circuits is the tensor product, or the Kronecker product of two individual sub-matrices $A = A(\alpha) \otimes A(\beta)$. For example, transform matrix of two nand gates in parallel is

$$\begin{bmatrix} 1-c & 1-c & 1-c & c \\ c & c & c & 1-c \end{bmatrix} \otimes \begin{bmatrix} 1-c & 1-c & 1-c & c \\ c & c & c & 1-c \end{bmatrix}.$$

Note that the order in the Kronecker product is important. $A \otimes B \neq B \otimes A$, which represents the order of the circuits. The rule of thumb is that $A(\alpha) \otimes B(\beta)$ describes the ensemble transform matrix when sub-circuit α is on top of sub-circuit β.

A fan-out topology is that output of one component is connected to multiple inputs of its following components. With this kind of structure we can eliminate the columns in the transform ensemble matrix when the elements in the matrix correspond to same interconnections via same wire. As an example, input signal "a" in Fig. 6 fan-outs to both inverters, so the input combination won't occur "01" and "10", thus the input combination will be only "00" and "11". The transform matrix is then

$$\begin{bmatrix} 1-c & c \\ 1-c & 1-c \\ 1-c & 1-c \\ c & 1-c \end{bmatrix}.$$

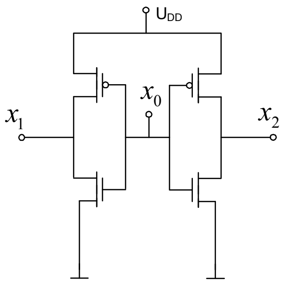


Fig. 6 An example of fan-out topology

Step 3: Get a whole transform matrix.

With step 1 and step 2 we get each component's transform matrix A_i. The components are divided as serial topology as Fig. 7.

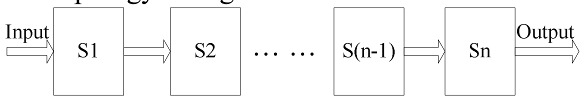


Fig. 7 Get a whole transform matrix

For all the components are serial connected, the global transform can be described as $A = A_n * A_{n-1} * \cdots * A_2 * A_1$.

Use Fig. 8 as an example, we can divide it to four sub-circuits with three stages as shown in Fig 8. The first sub-circuit is fan-out structure where x_1 fan-outs to inputs d_1 and d_2. The second sub-circuit is parallel structure which is constructed by two nand gates. Three stages in the circuit are serial structure. So the ensemble transform matrix can be described as $A = A_{inverter} * A_{nor} * (A_{nand} \otimes A_{nand})$. The two parallel nand gates in this transform matrix is $A_1 = A_{nand} \otimes A_{nand} =$

$$\begin{bmatrix} 1-c & 1-c & 1-c & c \\ c & c & c & 1-c \end{bmatrix} \otimes \begin{bmatrix} 1-c & 1-c & 1-c & c \\ c & c & c & 1-c \end{bmatrix}.$$

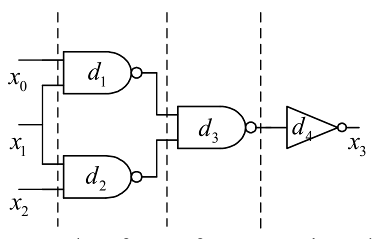


Fig. 8 An example of transform matrix calculation

For the first stage is fan-out structure, we can eliminate columns of input combination "X01X"and "X10X", then A_1 can be simplified as

$$\begin{bmatrix} (1-c)^2 & (1-c)^2 & (1-c)^2 & c(1-c) & (1-c)^2 & (1-c)^2 & c(1-c) & c^2 \\ c(1-c) & c(1-c) & c(1-c) & (1-c)^2 & c(1-c) & c(1-c) & c^2 & c(1-c) \\ c(1-c) & c(1-c) & c(1-c) & c^2 & c(1-c) & c(1-c) & c^2 & c(1-c) \\ c^2 & c^2 & c^2 & c(1-c) & c^2 & c^2 & c(1-c) & (1-c)^2 \end{bmatrix}.$$

3 Fault Tolerance Design Based on SER Estimation

In this section we analysis transform matrix of circuits and present criteria for fault tolerant design. The design criteria presented can assist us to design CAD software to find optimal nano-scale design.

3.1 Input and Output Status Analysis

As discussed before, the transform probability of inputs to outputs can be described by transform matrix as:

$$Y = AX \qquad (3)$$

where Y is matrix for distribution probability of outputs vector $\left[y_1, y_2, \cdots, y_{2^n}\right]^T$, X is matrix for distribution probability of inputs vector $\left[x_1, x_2, \cdots, x_{2^m}\right]^T$ and A is transform matrix $\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{12^m} \\ a_{21} & a_{22} & \cdots & a_{22^m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{2^n1} & a_{2^n2} & \cdots & a_{2^n2^m} \end{bmatrix}$ of the circuit.

Then we can get probability when output vector $y(i)$ is through transform matrix as:

$$y_i = a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{i2^m}x_{2^m} \qquad (4)$$

where x_j is probability when input is vector $x(j)$.

Assume that transform matrix is $\tilde{A}$ when gate works without error, and $\tilde{Y}$ is probability matrix for output vector, then we can get the relationship between input and output as $\tilde{Y} = \tilde{A}X$.

We call the work status without error as ideal status when the transform matrix is $\tilde{A} = \begin{bmatrix} \tilde{a}_{11} & \tilde{a}_{12} & \cdots & \tilde{a}_{12^m} \\ \tilde{a}_{21} & \tilde{a}_{22} & \cdots & \tilde{a}_{22^m} \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{a}_{2^n1} & \tilde{a}_{2^n2} & \cdots & \tilde{a}_{2^n2^m} \end{bmatrix}$. And we can get the following equation by the same way:

$$\tilde{y}_i = \tilde{a}_{i1}x_1 + \tilde{a}_{i2}x_2 + \cdots + \hat{a}_{i2^m}x_{2^m} \qquad (5)$$

(4) and (5) are the equations when the circuits works without error or with. Note that y_i and $\tilde{y}_i$ stand for probability of $y(i)$ and $\tilde{y}(i)$, so by analyzing the difference between y_i and $\tilde{y}_i$ we can get the SER when there are some soft error occurring in the circuit.

Assume that the SER of the circuit can be described as $p(Y \neq \tilde{Y})$, which indicates probability deviation for actual output and expected (or ideal) output. Then we can get $y_i - \tilde{y}_i = (a_{i1} - \tilde{a}_{i1})x_1 + (a_{i2} - \tilde{a}_{i2})x_2 + \cdots + (a_{i2^m} - \hat{a}_{i2^m})x_{n^m}$ and $Y - \tilde{Y} = AX - \tilde{A}X = (A - \tilde{A})X$.

Define SER vector $e_i = y_i - \tilde{y}_i = \sum_{k=1}^{k=2^m} (a_{ik} - \tilde{a}_{ik})x_k$ which describes the probability deviation of expected and ideal distribution, MSE(error of mean square) of the signal vector is then:

$$\varepsilon = \|e_i\|^2 = (\sum_{k=1}^{k=2^m} (a_{ik} - \tilde{a}_{ik})x_k)^2 = x_i^{'}(\delta A)^{'}(\delta A)x_i \quad (6)$$

where $\delta A = A - \tilde{A}$, which we call it as error transform matrix, A' is the conjugate transpose matrix of A.

Now, we can see that the transform matrix can reveal the work status of the circuit and get information about SER when the circuit constructed by gates with soft error.

3.2 Fault Tolerance Design Criteria

Different circuit has different character for fault tolerant when the soft error occurs. Our design is aim to minimize the SER for circuit. We want to have the actual transform matrix A be as close as possible to the desired ideal matrix $\tilde{A}$. Note that MSE for input vector $x(i)$ is $\varepsilon = \|e_i\|^2 = x_i^{'}(\delta A)^{'}(\delta A)x_i$. Then for input matrix X we can get equation as:

$$\varepsilon = \sum_{i=1}^{i=2^n} \|e_1\|^2 = \sum_{i=1}^{i=2^n} x_i^{'}(\delta A)^{'}(\delta A)x_i = X^{'}(\delta A)^{'}(\delta A)X \quad (7)$$

In (7), let $\Lambda = (\delta A)^{'}(\delta A)$, we can see that with same input, MSE is determined by matrix Λ which has $n \times n$ elements.

Our goal is to minimize the norm of Λ. There are many definitions of the norm for a matrix and one widely used is the Frobenius norm

$$\|A\|_F = (\sum_{j=1}^{m}\sum_{i=1}^{n}|a_{ij}|^2)^{\frac{1}{2}} \quad (8)$$

Condition number for A is $cond(A) = \|A\|\|A^{-1}\|$ which is always simplified as $cond(A) = \lambda_{\max}/\lambda_{\min}$, where $\lambda_{\max}$ and $\lambda_{\min}$ is maxim and minimal eigen-value for spectral norm of transform matrix A.

In general, the condition number can be used to measure how singular a matrix is. If the condition number is large, it indicates that the matrix is nearly singular. Small condition numbers indicate good fault tolerance quality of a circuit and large condition numbers indicate the circuit is more sensitive to fault.

Our simulation shows that when $cond(\Lambda)$ of transform matrix is smaller the circuit is more fault tolerance.

4 Simulation Results

In this section we present some simulations to show validity of our criteria described previous sections. We use a HSPICE toolbox for MATLAB developed by Silicon Laboratories Inc (Http://www-mtl.mit.edu/~perrott). The toolbox is employed to analyze the simulation results generated by HSPICE and MATLAB. We then calculate the actual error rate of individual circuits. The simulation consists of four steps:

1) Use MATLAB to generate two sets of control signals with designated error rate. The data is later used in HSPICE.

2) Utilize HSPICE to simulate the output of individual circuits with the control signals created at the first step.

3) Employ MATLAB to analyze the output data from HSPICE and calculating the actual error rate of various circuits.

4) Repeat previous three steps several times and recording the actual error rate of each circuit during each run. Get the averaged results and comparing them with the theoretical SER values calculated according to the matrix method.

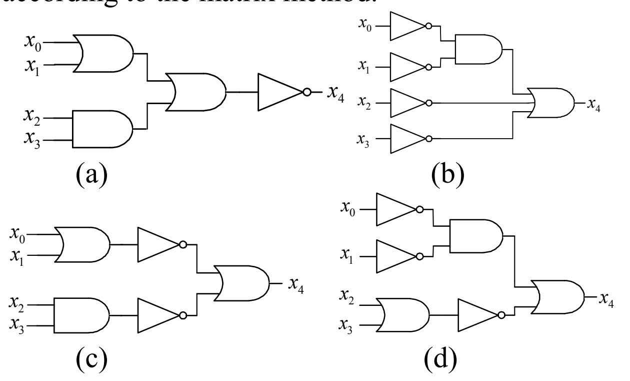


Fig. 8 Four circuits for function $x_4 = \bar{x}_0 \bullet \bar{x}_1 + \bar{x}_2 + \bar{x}_3$

In Fig. 8 we show four circuits with four inputs and one output which achieve same logic function $x_4 = \bar{x}_0 \bullet \bar{x}_1 + \bar{x}_2 + \bar{x}_3$ .The simulation results are shown in Table 1 and Table 2.

Use circuit "a" for instance, the simulation results with two different ways is shown in Fig. 9.

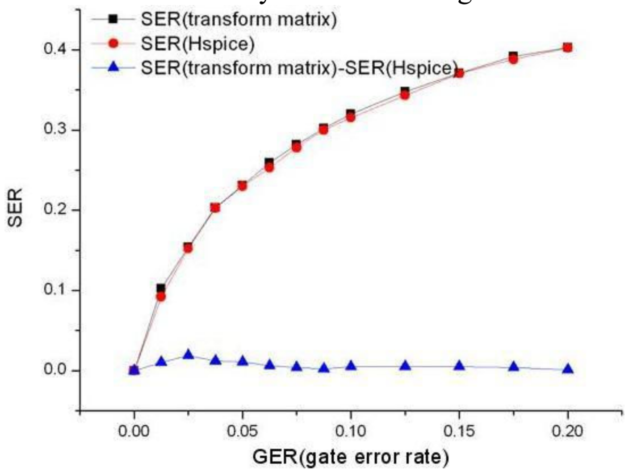
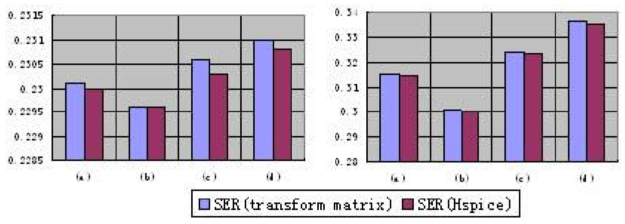
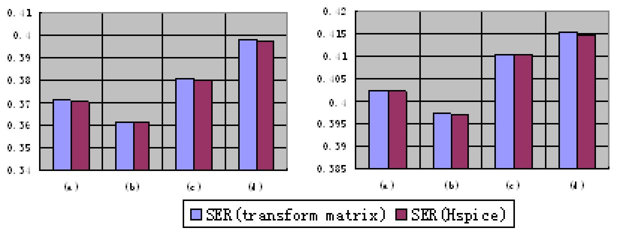


Fig. 9 Comparison of transform matrix and Hspice simulation results for circuit "a"

The simulation results in Fig. 9 shows that SER calculated by transform matrix deviates from Hspice is very small with the peak value at gate error rate (GER) 0.025. And as shown in Fig. 10, with these four circuits Hspice simulation results for SER matches the theoretical analysis using the transform matrix well when soft error occurs. The simulation results suggest that the transform matrix model can



(a) GER is 5% (b) GER is 10%



(c) GER is 15% (d) GER is 20%

Fig. 10 SER for circuits in Fig. 8 with different gate error rate (GER)

correctly model actual circuits' soft error behavior

Fig. 11 shows condition number for circuits "a" to "d". By calculating the condition number we can see clearly that when the gate 's SER is between 0–

25%, circuit “c” is more fault tolerance than others, which matches with Fig. 10 very well.

Table 1 SER for circuits in Fig. 8 by transform matrix computation

SER(q)	Circuit a	Circuit b	Circuit c	Circuit d
5%	0.2301	0.2296	0.2306	0.2310
10%	0.3154	0.3147	0.3157	0.3160
15%	0.3709	0.3701	0.3712	0.3715
20%	0.4025	0.4020	0.4028	0.4030

Table 2 SER for circuits in Fig. 8 by Hspice simulation

SER(q)	Circuit a	Circuit b	Circuit c	Circuit d
5%	0.2300	0.2296	0.2303	0.2308
10%	0.3152	0.3145	0.3154	0.3158
15%	0.3704	0.3699	0.3709	0.3712
20%	0.4024	0.4018	0.4027	0.4028

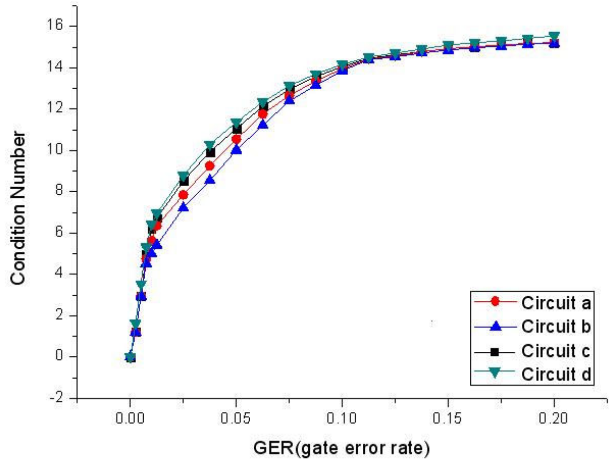


Fig. 11 Condition number of transform matrix with different GER

To show the generality of the transform matrix model, we also perform simulations for several other circuits. In Fig. 12 we show four circuits with five inputs and two outputs which achieve same logic $x_5 = x_0x_2 + x_1(\bar{x}_2 + \bar{x}_3); x_6 = \overline{x_2x_3}(x_1 + x_4)$.

The simulation results are shown in Table 3 and Table 4. Simulation result shows that SER calculated by transform matrix matches Hspice well as shown in Fig. 13 and Fig. 14. Our simulation results suggest that the transform matrix model can correctly model actual circuits’ soft error behavior.

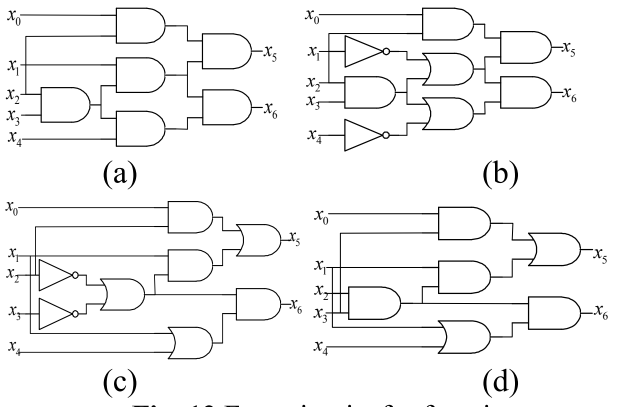
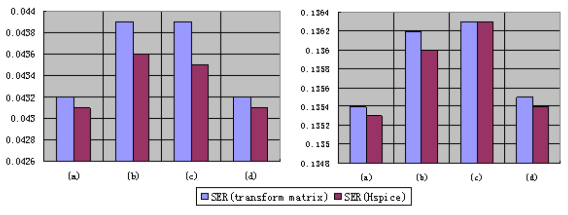
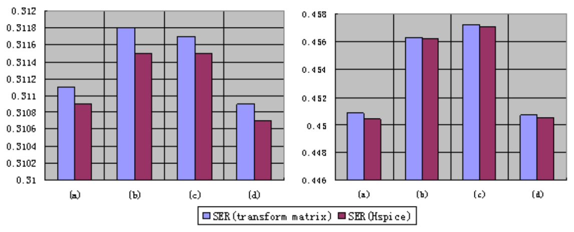


Fig. 12 Four circuits for function
$$x_5 = x_0x_2 + x_1(\bar{x}_2 + \bar{x}_3); x_6 = \overline{x_2x_3}(x_1 + x_4)$$



(a) GER is 1.25% (b) GER is 2.75%



(c) GER is 7.5% (d) GER is 15%

Fig. 13 SER for circuits in Fig. 12 with different gate error rate (GER)

Fig. 15 shows condition number for circuits "a" to "d". Condition number shows clearly that when the gate 's SER is between 0–20%, circuit "a" and "d" are more fault tolerance than others, which matches with Fig. 13 and Fig. 14 very well.

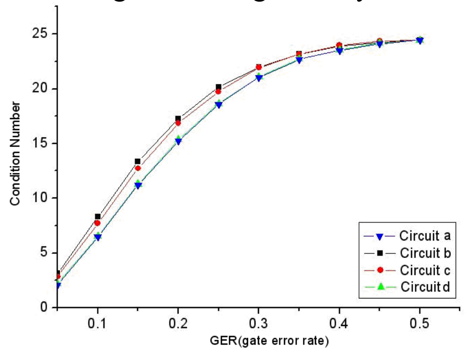
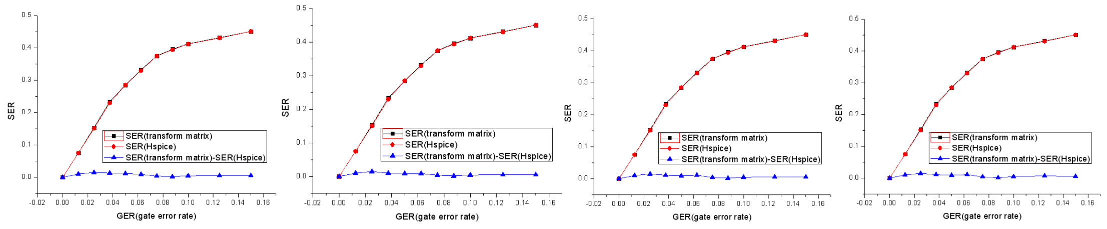


Fig. 15 Condition number of transform matrix with different GER



(a) circuit a (b) circuit b (c) circuit c (d) circuit d

Fig. 14 Comparison of transform matrix and Hspice

Table 3 SER for circuits in Fig. 12 by transform matrix computation

SER(q)	Circuit a	Circuit b	Circuit c	Circuit d
1.25%	0.0432	0.0439	0.0439	0.0432
2.75%	0.1354	0.1362	0.1363	0.1355
7.5%	0.3111	0.3118	0.3117	0.3109
15%	0.4508	0.4563	0.4572	0.4507

Table 4 SER for circuits in Fig. 8 by Hspice simulation

SER(q)	Circuit a	Circuit b	Circuit c	Circuit d
1.25%	0.0431	0.0436	0.0435	0.0431
2.75%	0.1353	0.1360	0.1363	0.1354
7.5%	0.3109	0.3115	0.3115	0.3107
15%	0.4504	0.4562	0.4571	0.4505

5 Conclusion

This paper studies the impact of the unavoidable soft errors on the performance of nano devices and investigates the optimal fault-tolerant of nano-scale circuits. We model the nano-circuits with transform matrix then make out a criteria by analyzing the matrix. The simulation based results show that the model and criteria is very suitable for nano-scale circuit design. Our work provides important guidelines for the development of computer-aided design (CAD) tools for optimal design of nano-scale circuits.

Acknowledgment This work is supported by Scientific Research Fund of Zhejiang Provincial Education Department (Y200805004) and Doctoral Fund of Zhejiang Gongshang University (1120XJ030741).

References:

[1] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi, Modeling the effect of technology trends on the soft error rate of combinational logic, *Proc. of International Conf. on Dependable Systems and Networks*, 2002, pp. 389-398.

[2] R. Saito, G. Dresselhaus, M. S. Dresselhaus, Physical Properties of Carbon Nanotubes, *World Scientific Publishing Company*, 1998.

[3] W. Rizwan, S.G. Ansaria, S.K. Young, T.R. Mohantyb, I.H. Hwangb and S. Hyung-Shik, Immobilization of DNA on nano-hydroxyapatite and their interaction with carbon nanotubes, *Synthetic Metals*, vol.159, No.3-4, 2009, pp. 238-245.

[4] X.Q. He, Y. Yan, L.X. Zhang, Flow-induced instability and bifurcation of carbon nanotubes, *Proc. of the 7th IASME / WSEAS International Conference on Fluid Mechanics and Aerodynamics*, 2009, pp. 21-27.

[5] A.S. Shamsabadi, B.S. Ghahfarokhi, K. Zamanifar and N. Movahedinia, Applying inherent capabilities of quantum-dot cellular automata to design: D flip-flop case study, *Journal of Systems Architecture*, Vol.55, No.3, 2009, pp. 180-187.

[6] P. Mazumder, S. Kulkarni, M. Bhattacharya, J. P. Sun, and G. I. Haddad, Digital circuit applications of resonant tunneling devices, *Proc. of the IEEE*, Vol.86, No.4, 1998, pp. 664-686.

[7] T.H. Zheng, W.C.H. Choy, Y.X. Sun, Nanoparticle-induced resonant tunneling behaviors in small molecule organic light-emitting devices, *Applied Physics Letters*, Vol.94, No.12, 123303, 2009.

[8] W. C. P. Henk, T. Tijs, Y. Zhen, G. Milena, D. Cees, Carbon Nanotube Single-Electron Transistors at Room Temperature, *Science*, Vol.293, No. 527, 2001, pp. 76-79.

[9] C.Y. Zhu, Z.Y. Gu, R.P. Dick, L. Shang, G.R. Knobel, Characterization of single-electron tunneling transistors for designing low-power embedded systems, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol.17, No.5, 2009, pp. 646-659.

[10] R. Martel, V. Derycke, J. Appenzeller, S. Wind and Ph. Avouris, Carbon nanotube field-effect transistors and logic circuits, *Proc. of the 39th annual Design Automation Conference*, 2002, pp. 94-98.

[11] M.B. Tahoori, J. Huang, M. Momenzadeh and F. Lombardi, Defects and faults in quantum cellular automata at nanoscale, *22nd IEEE In VLSI Test Symposium*, 2004, pp. 291-296..

[12] R. K. Kummamuru, et al., Operation of Quantum-Dot Cellular Automata (QCA), Shift Registers and Analysis of Errors, *IEEE Transactions on Electron Devices*, Vol.5, No.9, 2003, pp. 1906-1913.

[13] J. von Neumann, *Probabilistic logic and the Synthesis of Reliable Organisms from Unreliable Components*, Princeton University Press, 1956.

[14] S. Roy and V. Beiu, Majority multiplexing: economical redundant fault-tolerant designs for nanoarchitectures, *IEEE Transactions on Nanotechnology*, Vol. 4, No.4, 2005, pp.441-451.

[15] Petivan et al., *Triple modular redundant computer system*, United States Patent, 2001.

[16] J. Biernat, The effect of compensating faults models on NMR system reliability, *IEEE Transactions on Reliability*, Vol.43, No.2, 1994, pp. 294-300.

[17] P.S. Daniel and S.S. Robert, *Reliable Computer Systems: Design and Evaluation*. Digital Press, 2nd Edition, 1992.

[18] J. Chen, J. Mundy, Y. Bai, S. Chan, P. Petrica, and R. I. Bahar, A probabilistic approach to nano-computing, *Proc. of the Second Workshop on Non-Silicon Computing*, 2003, pp.1-8.

[19] N. Kundan, R. Iris Bahar, M. Joseph, W. R. Patterson, A. Zaslavsky, Designing logic circuits for probabilistic computation in the presence of noise, *Design Automation Conference*, 2005, pp.485-490.

[20] I-Chyn Wey, Y.G. Chen, C.H. Yu, J. Chen and A.Y. Wu, A 0.18μm Probabilistic-Based Noise-Tolerate Circuit Design and Implementation with 28.7dB Noise-Immunity Improvement, *IEEE Asian Solid-State Circuits Conference (A-SSCC)*, 2006, pp.291-294.

[21] V. Eichhorn et al., NanoLab: A nanorobotic system for automated pick-and-place handling and characterization of CNTs, *IEEE International Conference on Robotics and Automation*, 2009, pp.1826-1831.

[22] D. Bhaduri and S. Shukla, NANOLAB—a tool for evaluating reliability of defect tolerant nanoarchitectures, *IEEE Transactions on Nanotechnology*, Vol.4, No.4, 2005, pp.381-394.

[23] D. Bhaduri and S. Shukla. *Nanoprism: A tool for evaluating granularity vs. reliability trade-offs in nano-architectures*, In GLSVLSI, Boston, MA, 2004.

[24] K. N. Patel, J. P. Hayes, and I. L. Markov, Evaluating circuit reliability under probabilistic gate-level fault models, *In IWLS,* 2003, pp.59-64.

[25] J. Han and P. Jonker, A system architecture solution for unreliable nanoelectronic devices, *IEEE Transactions on Nanotechnology*, Vol.1, No.4, 2002, pp.201-208.

[26] H.F. Rao et al., Ensemble Dependent Matrix Methodology for Probabilistic-Based Fault-tolerant Nanoscale Circuit Design, *IEEE International Symposium on Circuits and Systems*, 2007, pp.1803-1806.