TMR based Error Correction Method Considering Trade-off between Area and Soft-Error Tolerance

Harada, Shoji
Graduate School of Information Science and Electrical Engineering, Kyushu University

Yoshimura, Masayoshi
Faculty of Information Science and Electrical Engineering, Kyushu University

Matsunaga, Yusuke
Faculty of Information Science and Electrical Engineering, Kyushu University

TMR based Error Correction Method Considering Trade-off between Area and Soft-Error Tolerance

Shoji Harada*, Masayoshi Yoshimura**, Yusuke Matsunaga**
*Graduate School of ISEE, Kyushu University
Email : s-harada@soc.ait.kyushu-u.ac.jp
**Faculty of ISEE, Kyushu University
Email : {yoshimura, matsunaga}@ait.kyushu-u.ac.jp

Abstract

Recently, the lowering of soft error tolerance of LSI becomes the problem. Soft error is a phenomenon that the output value of a logic gate flips transiently or the preserved value of a storage element flips because of neutron particle strike etc. This paper presents a TMR based error correction method considering trade-off between soft error tolerance and area overhead. Our method corrects an error which occurs in a logic circuit when specific input vectors are given to the circuit. In our method, the degree of loss of soft error tolerance that can be allowed in target circuit is given as a design constraint. The method aims at selecting input vectors which contribute to area minimization of correction circuits under the constraint as unprotected vectors. This paper shows vector selection method for minimizing the area. For results, there are several circuits that it is cost-effective. It seems that there is validity in our approach.

1 Introduction

Recently, the lowering of soft error tolerance of LSI is gradually becoming the problem. Soft error is a phenomenon that the output value of a logic gate flips transiently or the preserved value of a storage element flips because of electric charge occurred by neutron particle strike at transistor. A soft error may cause an incorrect operation of LSI. As semiconductor devices scale down and supply voltages reduce, necessary electric charge for flipping the value of a transistor declines. The probability that a soft error occurs when neutron particle strikes at transistor is increasing. A soft error might become the problem which cannot be bypassed in the future.

In memory circuits, techniques which can achieve high reliability with comparatively low area overhead are established. One of such techniques is error checking and correcting (ECC). On the other hand, in logic circuits, techniques which can achieve high reliability with low area overhead are not established. Therefore, it is thought that techniques for reducing soft error rate in logic circuits becomes more important [1][2].

Two techniques that can be used to reduce soft error rate are error detection and error correction. Error detection method involves using concurrent error detection (CED) [3][4] circuitry that monitors the outputs of a circuit for the occurrence of an error. If an error is detected, the system recovers through rollback and retry thereby preventing a failure. It may not be possible to do retry for realtime systems, thus error correction is the only option. This paper focuses on error correction methods to reduce the soft error rate in logic circuits.

There are error correction methods which apply multiplexing [5][6]. The method in [5] is called triple modular redundancy (TMR). TMR can achieve high reliability. But TMR needs large area overhead. It applies mission critical applications with very high reliability where area overhead is a secondary concern. In other applications, it is necessary to reduce soft error rate to acceptable level with low overhead. The method in [6] is a TMR based error correction method considering trade-off between soft error tolerance and area overhead. This method focuses on a difference of probabilities that an error which occurs at each gate propagates to one of the primary outputs or latches. It applies triplexing from the output side of the circuit to a gate which has the high probability by priority. This method has a problem that it is not too cost-effective.

This paper presents a TMR based error correction method considering trade-off between soft error tolerance and area overhead with a different approach. There is a property that the propagation of an error is different according to input vector. The method focuses on the property. By adding two correction circuits to a target circuit, this method corrects an error which occurs in one of circuits at specific input vector. In [4], it also focuses on the property. However it cannot control the trade-off well. Our method is possible to control the trade-off. In the following, we call input vectors at which our method corrects an error protected vectors and input vectors at which our method does not correct an error unprotected vectors.

The key idea in our proposed method is to aim at reducing the area of two correction circuits by setting unprotected vectors as don't cares in one of correction circuits. If more unprotected vectors are set as don't cares, the area may be reduce more. But, soft error rate increases in the case. In the method, the degree of loss of soft error tolerance that can be allowed in target circuit is given as a design constraint. The method aims at selecting input vectors which contribute to area minimization of correction circuits under the constraint as unprotected vectors. Thus, how to select unprotected vectors is important.

Many methods for two-level logic circuits which optimizing logic are proposed. The methods in [7][8] is widely known. But these methods optimize logic under the condition that don't cares are given. In our method, it is necessary to select don't cares. this paper shows an algorithm of selecting unprotected vectors for two-level logic circuits. The purpose of the algorithm is to select input vectors which contribute to area minimization under a design constraint as unprotected vectors. Experimental results show the effect of area reduction and the degree of achievement of soft error tolerance. We compare our method and the method in [6]. From results, the tendency that soft error rate reduction is proportional to area overhead doesn't change according to circuits in [6]. The method in [6] is not too cost-effective. In our method, there are several circuits that it is cost-effective. It seems that there is validity in our approach.

The rest of the paper is organized as follows. Section 2 describes the previous works. Section 3 presents our proposed method. Section 4 presents the experimental result. Section 5 presents conclusions.

2 Previous works

In this section, we describe previous works for reducing soft error rate.

TMR[5] is a method which applies triplexing to the target circuit and uses decision by majority, as shown in Fig. 1. TMR can correct an error unless two or more circuits output incorrect values at the same time. TMR can achieve high reliability. The problem of TMR is large area overhead and large increment in power consumption because of two additional circuits which are the same as the target circuit. Therefore, TMR is applied to equipments in the field where reliability is the most important such as aviation, medical, bank etc.

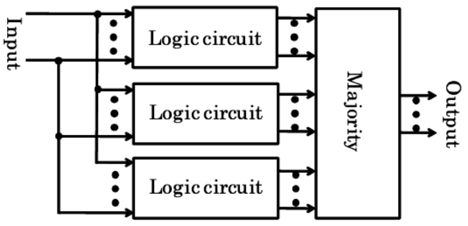


Figure 1: TMR circuit

The method in [6] is a TMR based error correction method considering trade-off between soft error tolerance and area overhead. This method focuses on a difference of probabilities that an error which occurs at each gate propagates to one of the primary outputs or latches. It applies triplexing from the output side of the circuit to the gate which has the high probability by priority. This method can correct an error which occurs at logic gates applied triplexing. It cannot correct an error which occurs at logic gates which is not applied triplexing. The problem of this method is to need large area overhead to achieve desired soft error tolerance.

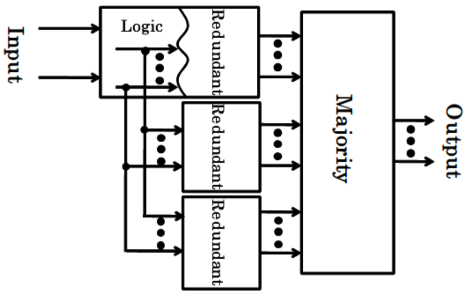


Figure 2: Partial TMR circuit

3 Proposed method

3.1 Overview

Our proposed method is based on TMR. By adding two correction circuits to a target circuit, this method corrects an error which occurs in one of circuits at specific input vector. In the following, we call input vectors at which our method corrects an error protected vectors and input vectors at which our method does not correct an error unprotected vectors.

Our method is composed of a target circuit and two additional correction circuits and a majority circuit, as shown Fig. 3. In Fig. 3, C_1 is a target circuit and C_2, C_3 are correction circuits.

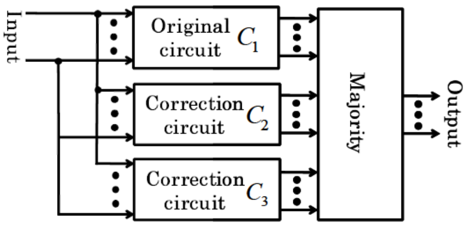


Figure 3: proposed method circuitry

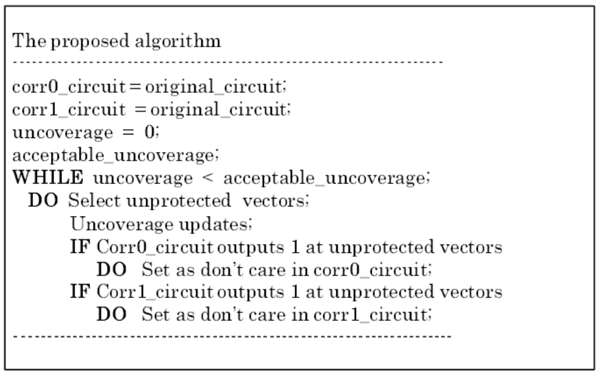
There are two requirements for two correction circuits. First, correction circuit C_2 and C_3 must output the same value as the output value of C_1 when three circuits are given a protected vector. Our method can correct an error which occurs in one of C_1, C_2 and C_3 by using decision by majority. Second, one of correction circuit C_2 and C_3 must output the same values as the output value of C_1 when three circuits are given an unprotected vector. The purpose of this requirement is to prevent final output values from being incorrect by using decision by majority in spite of the fact that a soft error does not occur. Another correction circuit which does not output the same value as the output value of C_1 sets a unprotected vector as don't care. Our method aims at reducing the area of correction circuits by setting don't cares. When the output value of C_1 is 0 at an unprotected vector, correction circuit C_2 sets the vector as don't care. When the output value of C_1 is 1 at an unprotected vector, correction circuit C_3 sets the vector as don't care.

In our method, the degree of loss of soft error tolerance which can be allowed in C_1 is given as a design constraint. The method aims at selecting input vectors which contribute to area minimization of correction circuits under the constraint as unprotected vectors. Thus, how to select unprotected vectors is important. Many methods for two-level logic circuits which optimizing logic are proposed. The methods in the literature [7][8] is widely known. These methods optimize logic under a condition that the don't cares are given. Our method needs to select don't cares.

First process of our proposed method is to equalize two correction circuits C_2 and C_3 with a target circuit C_1. Second process is to select input vectors which can reduce the area of correction circuits as unprotected vectors. Third process is to reduce the area by setting selected unprotected vectors as don't cares according to the output values of correction circuits. Our method repeats second and third process until a certain condition is satisfied. Trade-off between soft error tolerance and area overhead is considered by changing the condition. It shows the pseudo code of our algorithm in Fig. 4. In Fig. 4, $corr0\_circuit$ is C_2 which sets input vectors at which the function values are 0 as don't cares. $corr1\_circuit$ is C_3 which sets input vectors at which the function values are 1 as don't cares. $uncoverage_k$ is the degree of loss of soft error tolerance of C_1. $acceptable\_uncoverage_k$ is acceptable the degree of loss of soft error tolerance of C_1.

$uncoverage_k$ is the degree of loss of soft error tolerance of C_1. Our method considers trade-off between soft error tolerance and area overhead by changing $uncoverage_k$.
$uncoverage_k$ is shown as equation (1).

$$uncoverage_k = 100 \times \left(\frac{\sum_{k_u \in K_u} P_{s\_input}(k_u)}{\sum_{k \in K} P_{s\_input}(k)} \right) [\%] \qquad (1)$$



```
The proposed algorithm
---------------------------------------------------------------------
corr0_circuit = original_circuit;
corr1_circuit  = original_circuit;
uncoverage  =  0;
acceptable_uncoverage;
WHILE uncoverage  <  acceptable_uncoverage;
  DO  Select unprotected  vectors;
        Uncoverage updates;
        IF Corr0_circuit outputs 1 at unprotected vectors
            DO   Set as don't care in corr0_circuit;
        IF Corr1_circuit outputs 1 at unprotected vectors
            DO   Set as don't care in corr1_circuit;
---------------------------------------------------------------------
```

Figure 4: The proposed algorithm

In equation (1), K is a set of all input vectors. K_u is a set of selected unprotected vectors. $P_{s\_input}(k)$ is a probability that an error which occurs in circuits propagates to one of primary outputs or latches when circuits are given at input vector k. For example, if $uncoverage_k$ is 10, our method can correct more than 90 percent of an error occurs in a target circuit. $P_{s\_input}(k)$ is shown as equation (2).

$$P_{s\_input}(k) = \frac{1}{|G|} \sum_{g \in G} P_{prop}(g, k) \qquad (2)$$

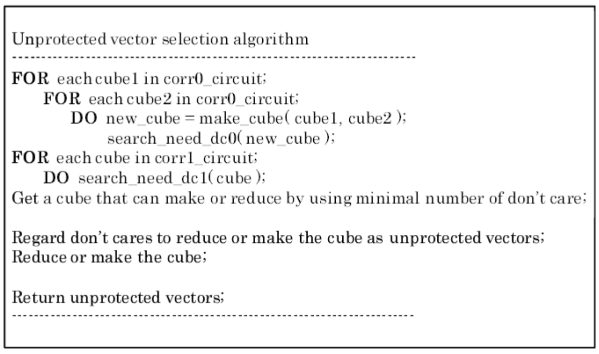
In equation (2), G is a set of all gates in a circuit. $P_{prop}(g, k)$ is a probability that an error occurs at gate g propagates to one of primary outputs or latches when circuits are given at input vector k. $P_{s\_input}(k)$ is a quantitative metrics which shows the easiness of propagation of an error which occurs at a input vector. Our method can consider trade-off between soft error tolerance and area overhead by using $P_{s\_input}(k)$.

3.2 Vector selection algorithm

Our method aims to select don't cares for reducing the area minimization. For two-level logic circuits, the reduction of the number of cubes and literals in the logical expression of a circuit leads to area reduction. Previous works optimize logic under a condition that don't cares are given[7][8]. In our method, it is necessary to select don't cares which contribute to the reduction of the number of cubes and literals.

In the following, an algorithm of selecting unprotected vectors for two-level logic circuits is described. First process is to make a minimal cube which covers two cubes in the logical expression of correction circuit C_2. There are several cubes which cover two cubes. A minimal cube is unique in their cubes. Second process is to get the number of necessary don't cares to make a minimal cube by setting input vectors which correspond to 0 minterms of C_2 as don't care and to get the number of literals which can be reduced when the minimal cube is made. This method executes first and second process for all pairs of cubes in C_2.Third process is to get the number of necessary don't cares to reduce a cube in C_3 by setting input vectors which correspond to 1 minterms of C_3 as don't care, and the number of literals which can be reduced when the cube is reduced. This method executes third process for all cubes in C_3. Forth process is to get a cube which is made or reduced by using the minimal number of necessary don't cares. When the number of necessary don't cares is the same as that of the others, this method defines a cube that the number of literals which can be reduced is the largest as a target. Final process is to select necessary don't cares for reducing or making the target cube as unprotected vectors.

From stated above, the pseudo code of unprotected vector selection algorithm is shown in Fig. 5. In Fig. 5,



```
Unprotected vector selection algorithm
-----------------------------------------------------------------------
FOR eachcube1 in corr0_circuit;
    FOR each cube2 in corr0_circuit;
        DO new_cube = make_cube( cube1, cube2 );
            search_need_dc0( new_cube );
FOR each cube in corr1_circuit;
    DO search_need_dc1( cube );
Get a cube that can make or reduce by using minimal number of don't care;

Regard don't cares to reduce or make the cube as unprotected vectors;
Reduce or make the cube;

Return unprotected vectors;
-----------------------------------------------------------------------
```

Figure 5: Algorithm of selecting unprotected vectors

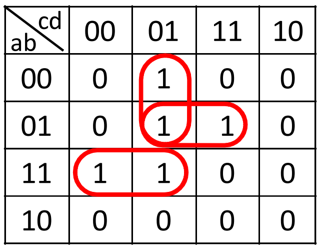
$make\_cube$ is a function which makes a minimal cube which covers two given cubes. $search\_need\_dc0$ is a function which gets the number of necessary don't cares to make a given cube by setting input vectors which correspond to 0 minterms as don't care, and the number of literals which can be reduced when the cube is made. $search\_need\_dc1$ is a function which gets the number of necessary don't cares to reduce a given cube by setting input vectors which correspond to 1 minterms as don't care, and the number of literals which can be reduced when the cube is reduced.

The reduction of the number of cubes and literals by using don't cares is concretely shown. It assume that the logic function of a target circuit C_1 which has 4-inputsa, b, c, d is shown f_1, as shown equation (3).

$$f_1 = \bar{a}\bar{c}d + \bar{a}bd + ab\bar{c} \tag{3}$$

The number of literals of f_1 is 9. The karnaugh map of f_1 is shown in Fig. 6.

First, there is how to select input vectors which correspond to 0 minterms as don't cares. The reduction of the number of cubes is executed by making a cube which covers cubes in the logical expression. The logic

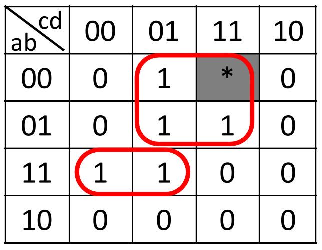


ab \ cd	00	01	11	10
00	0	1	0	0
01	0	1	1	0
11	1	1	0	0
10	0	0	0	0

Figure 6: The karnaugh map of f_1

function sets necessary input vectors which correspond to 0 minterms for making the cube as don't cares. In f_1, A cube $\bar{a}d$ can be made with the minimum number of don't cares. A cube $\bar{a}d$ can be made by setting 0 minterm $\bar{a}\bar{b}cd$ which f_1 does not cover as don't care. If we newly make $\bar{a}d$, f_1 can cover two cubes $\bar{a}\bar{c}d$ and $\bar{a}bd$. If f_1 sets $\bar{a}\bar{b}cd$ as don't care, f_1 changes into f_2, as shown equation (4). The number of literals of f_2 is 5. The karnaugh map of f_2 is shown in Fig. 7.

$$f_2 = \bar{a}d + ab\bar{c} \tag{4}$$



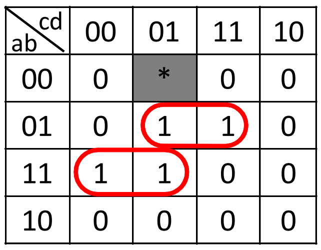
ab \ cd	00	01	11	10
00	0	1	*	0
01	0	1	1	0
11	1	1	0	0
10	0	0	0	0

Figure 7: The karnaugh map of f_2

Next, there is how to select input vectors which correspond to 1 minterms as don't cares. It is necessary to reduce a cube itself in a logic expression. That is, the reduction of the number of cubes is executed by setting minterms which are covered only by a cube for reducing as don't cares. A minterm which is covered only by $\bar{a}\bar{c}d$ is $\bar{a}\bar{b}\bar{c}d$. A cube $\bar{a}\bar{c}d$ can be reduced with the minimum number of don't cares in f_1. The number of cubes is reduced by setting the input vector corresponding to $\bar{a}\bar{b}\bar{c}d$ as don't care. In this case, f_1 changes into f_3, as shown equation (5). The number of literals of f_3 is 6. The karnaugh map of f_3 is shown in Fig. 8.

$$f_3 = \bar{a}bd + ab\bar{c} \tag{5}$$

As stated above, our method aims at selecting don't care the number of cubes of correction circuits by setting input vectors as don't cares.



ab \ cd	00	01	11	10
00	0	*	0	0
01	0	1	1	0
11	1	1	0	0
10	0	0	0	0

Figure 8: The karnaugh map of f_3

4 Experimental results

The purpose of this experiment is to evaluate the effect of area reduction by unprotected vectors and the degree of gain of soft error tolerance. We compare our method and the method in [6].

First, the method in [6] is described. This method considers trade-off between soft error tolerance and area overhead by using $uncoverage_g$. $uncoverage_g$ is the degree of loss of soft error tolerance. $uncoverage_g$ is shown as equation (6).

$$uncoverage_g = 100 \times \left(\frac{\sum_{g_u \in G_u} P_{s\_gate}(g_u)}{\sum_{g \in G} P_{s\_gate}(g)} \right) [\%] \tag{6}$$

In equation (6), G is a set of all gates. G_u is a set of gates which are applied triplexing. $P_{s\_gate}(g)$ is a probability that an error occurs at gate g propagates to one of primary outputs or latches. For example, if $uncoverage_g$ is 10, this method can correct about 90 percent of an error occurs in a circuit. $P_{s\_gate}(g)$ is shown as equation (7).

$$P_{s\_gate}(g) = \frac{1}{|K|} \sum_{k \in K} P_{prop}(g, k) \tag{7}$$

In equation (7), $P_{prop}(g, k)$ is a probability that an error occurs at gate g propagates to one of primary outputs or latches when input vector k is given to circuits. $P_{s\_gate}(g)$ is a quantitative metrics which shows the influence of an error occurs at a gate. This method can consider trade-off between soft error tolerance and area overhead by using $P_{s\_gate}(g)$.

The metrics of soft error tolerance is a probability that an error occurs in a circuit and it propagates to one of primary outputs or latches. The metrics of area is the number of literals of all circuits expect for a majority circuit. In this experiment, we evaluate on the assumption that an error does not occur in a majority circuit. We also assume that a probability that an error occurs at each gate is equal, and errors occur at gates at the same time. Furthermore, an error is a permanent flip of output value of a gate. The metrics of soft error tolerance is defined as SER and it is shown as equation (8).

$$SER = \frac{1}{|K|} \sum_{k \in K} \sum_{g \in G} P_{occur}(g, k) P_{prop}(g, k) \tag{8}$$

In equation (8), $P_{occur}(g, k)$ is a probability that an error occurs at gate g when circuits are given input vector k. From the above assumptions, $P_{occur}(g, k)$ can be regarded as a constant. $P_{prop}(g, k)$ is a probability that an error which occurs at gate g when circuits are given input vector k.

We explain about processes of this experiment. In our method, first process is to input a target two-level logic circuit, P_{s_input} of each input vector in the target circuit and several kinds of $acceptable_uncoverage$s. Second process is to execute our method for each $acceptable\_uncoverage$ and get a two-level logic circuit which is executed our method. Third process is to execute multi-level optimization to the circuit, and evaluate SER of the circuit and the number of literals. P_{s_input} of each input vector is computed in advance by fault simulation for a target circuit which is executed two-level and multi-level optimization. In the method in [6], first process is to input a target circuit, P_{s_gate} of each gate in the target circuit and several kinds of $acceptable_uncoverage$s. The target circuit is executed two-level and multi-level optimization. Second process is to execute the method for each $acceptable\_uncoverage$ and get a circuit which is executed the method. Third process is to evaluate SER of the circuit and the number of literals. P_{s_gate} of each gate is computed in advance by fault simulation for a target circuit which is executed two-level and multi-level optimization. In this experiment, we use the command of espresso, script.rugged in SIS[9] to execute two-level and multi-level optimization. We give five $acceptable\_uncoverage$ 10, 20, 30, 40, 50. Benchmark circuits are a part of twolexamples of MCNC benchmark set[10].

The experimental result for our method is shown in Table 1. The experimental result for the method in [6] is shown in Table 2. In Table 1 and 2, Benchmark is shown the benchmark name, and #Input, #Output is shown the number of primary inputs and primary outputs. propose/original is shown a value that SER of the circuit which is executed our method is divided by SER of original circuit and a value that the number of literals of the circuit which is executed our method is divided by the number of literals of original circuit. 10, 20, 30, 40, 50 in Table 1 and 2 is a value of $acceptable\_uncoverage$.

From Table 1, when $acceptable\_uncoverage$ is 10, the area of most benchmarks is 2.7 times as large as that of original circuit. Thus, it seems that the effect of reduction of area of correction circuits by unprotected vectors is small in most benchmarks. But, SER is reduced from about 80 to 90 percent. The area of misex3c and sao2 is about 2 and 1.4 times as large as that of original circuit. Even when $acceptable\_uncoverage$ is 10, it seems that the effect of reduction of area of correction circuits by unprotected vectors is large in their benchmarks. Moreover, SER is reduced about 90 percent. Therefore, it is thought that our method is effective for misex3c and sao2. As $acceptable\_uncoverage$ becomes 20 or 30, the area of all benchmarks is reducing. SER of all benchmarks is increasing. There is a benchmark that SER increases by about 20% when $uncoverage$ increases by 10. It is thought that this is because SER of the correction circuit in addition to SER of the original circuit increases by unprotected vectors. When $acceptable\_uncoverage$ becomes 40, the area of most benchmarks is about 2 and 1.4 times as large as that of original circuit. But, the effect of SER reduction was various according to the benchmark. When $acceptable\_uncoverage$ becomes 50, the area of most benchmarks is 2 times or less as large as that of original circuit.

In Table 2, it can be confirmed that the result for the method in [6] does not depend on benchmarks. When $acceptable\_uncoverage$ is 10, the method achieves about 90 percent of SER reduction with the area which is 2.8 times that of original circuit. When $acceptable\_uncoverage$ is 20, the method achieves about 80 percent of SER reduction with the area which is 2.6 times that of original circuit. As $acceptable\_uncoverage$ becomes 30, 40, this tendency does not change.

In Table 1 and 2, it is difficult to say our method is more effective than the method in [6]. In sao2, to reduce about 90 percent of SER, our method needs the area which is about 1.4 times the area of original circuit. But, the method in [6] needs the area which is about 2.8 times the area of original circuit. In sao2, it is thought that our method is more effective than the method in [6] It seems that it is similar in misex3c. Thus, it can be confirmed that it is different whether our method is effective according to the circuit. It is thought that two benchmark misex3c and sao2 has some feature. Thus, if some features can be gripped, in other circuits, it may be possible to make our method to a good method by improving.

5 Conclution

In this paper, we proposed a TMR based error correction method considering trade-off between soft error tolerance and area overhead. the method focused on a property. A property is that the easiness of propagation of an error is different according input vectors. We proposed vector selection algorithm which is the key of our method.

In this experiment, we evaluated the effect of area reduction and the degree of gain of soft error tolerance. We compare our method and the method in [6]. From this experimental result, the tendency that soft error rate reduction is proportional to area overhead doesn't change according to circuits in the method in [6]. If a design constraint is defined, the area overhead and the effect of soft error rate reduction. In our method, it seemed that the effect of area reduction of correction circuits was small in most benchmarks. The effect of SER reduction in our method was various according to the benchmarks. In our method, there are circuits that our method is effective.

Our future work is to analyze the circuitry and the function of benchmarks that it is thought that our method is effective. Thus, if some features can be gripped, in other circuits, it may be possible to make our method to a good method by improving. Moreover, our proposed unprotected vector selection algorithm is not practical because of large processing time. There is a circuit that it takes time for several days. Therefore, it is necessary to improve unprotected vector selection algorithm. It is necessary to apply our method to multi-level logic circuits.

References

[1] R. C. Baumann. ”Soft Errors in Advanced Computer Systems,” In IEEE Design and Test of Computers, Vol. 22, Issue 3, 2005.

[2] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, L. Alvisi. ”Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic,” In Proc. of DSN, 2002.

[3] K. Mohanram and N. A. Touba, ”Cost-effective approach for reducing soft error failure rate in logic circuits,” Proc. Intl. Test Conference, pp. 893-901, 2003.

[4] M. Choudhury and K. Mohanram, ”Approximate logic circuits for low overhead, non-intrusive concurrent error detection,” in Proc. Des. Autom. Test Eur., 2008, pp. 903.908.

[5] J. Von. Neumann, ”Probabilistic logic and the synthesis of reliable organisms from unreliable components,” Automata Studies, Ann. of Math. Studies, pp. 43-98, 1956.

[6] K. Mohanram and N. Touba, ”Partial Error Masking to Reduce Soft Error Failure Rate in Logic Circuits,” IEEE International Symposium on Defect and Fault Tolerance, 2003

[7] E. J. McCluskey, ”Minimization of Boolean functions,” Bell System Tech. J. , vol. 35, pp. 1417-1444, Apr. 1956.

Table 1: Result for our method

Benchmark	#Input	#Output	Area					SER reduction[%]				
			propose/original					propose/original				
			10	20	30	40	50	10	20	30	40	50
5xp1	7	10	2.74	2.29	2.18	2.08	1.86	92.10	80.97	78.42	76.23	68.17
9sym	9	1	2.73	2.47	2.27	1.93	1.74	78.72	58.99	39.69	38.35	26.21
bw	5	28	2.92	2.83	2.61	2.37	1.94	77.89	68.43	60.22	50.57	45.80
clip	8	5	2.69	2.25	2.00	1.87	1.66	91.99	82.75	79.30	72.58	64.39
con1	7	2	2.76	2.47	2.35	2.12	1.74	81.14	66.75	55.33	51.49	39.33
misex1	8	7	2.78	2.47	2.18	2.04	1.92	90.82	81.24	65.09	51.81	43.57
misex3c	14	14	1.99	1.90	1.82	1.74	1.69	90.82	81.24	65.09	51.81	43.57
rd53	5	3	2.62	2.64	2.36	2.11	2.01	89.07	74.62	58.46	50.67	49.24
rd73	7	3	2.62	2.55	2.37	2.21	2.01	89.55	73.27	58.91	44.96	33.93
rd84	8	4	2.71	2.57	2.34	2.10	1.97	80.22	65.07	50.97	50.84	50.67
sao2	10	4	1.38	1.56	1.36	1.24	1.11	92.27	93.34	92.14	88.07	83.07
xor5	5	1	2.77	2.32	2.06	2.00	1.80	77.00	57.00	38.00	34.00	29.00

Table 2: Result for the method in [6]

Benchmark	#Input	#Output	Area					SER reduction[%]				
			propose/original					propose/original				
			10	20	30	40	50	10	20	30	40	50
5xp1	7	10	2.84	2.68	2.48	2.24	2.01	90.98	80.16	70.33	60.59	50.31
9sym	9	1	2.81	2.59	2.37	2.20	1.94	90.14	80.11	70.33	60.34	50.16
bw	5	28	2.78	2.60	2.41	2.26	2.08	90.12	80.28	70.40	60.14	50.07
clip	8	5	2.82	2.58	2.43	2.27	2.02	90.22	80.60	70.44	60.24	50.21
con1	7	2	2.88	2.82	2.65	2.59	2.06	95.04	87.10	69.98	64.02	52.61
misex1	8	7	2.71	2.56	2.19	1.92	1.73	91.44	81.36	70.33	61.02	50.85
misex3c	14	14	2.81	2.59	2.35	2.02	1.66	90.06	80.30	70.03	60.22	50.02
rd53	5	3	2.89	2.74	2.61	2.44	2.24	90.21	80.13	71.10	61.31	50.67
rd73	7	3	2.84	2.65	2.30	2.19	2.05	90.09	80.34	70.11	60.16	50.44
rd84	8	4	2.90	2.57	2.26	2.08	1.84	90.16	80.02	70.10	60.21	50.16
sao2	10	4	2.69	2.44	2.23	2.02	1.87	90.02	80.80	70.55	60.26	50.53
xor5	5	1	2.87	2.75	2.70	2.57	2.32	90.00	80.00	72.00	62.00	51.50

[8] R. K. Brayton, C. McMullen, G. D. Hachtel, and A. Sangiovanni-Vincentelli. ”Logic Minimization Algorithms for VLSI Synthesis,” Kluwer Academic Publishers, 1984.

[9] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Sa voj, P. R. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. ”SIS: A System for Sequential Circuit Synthesis,” Technical Report UCB/ERL M92/41, Electronics Research Lab, Univ. of California, Berkeley, CA 94720, May 1992.

[10] S. Yang. ”Logic Synthesis and Optimization Benchmarks User Guide: Version 3.0,” 1991.