

**MODELING AND EVALUATION OF THE INTER-
CONNECTION - DRIVEN REPAIRABILITY
FOR DISTRIBUTED EMBEDDED
MEMORY CORES ON CHIP**

By

BYUNG-HYUN JANG

Bachelor of Science

Sungkyunkwan University

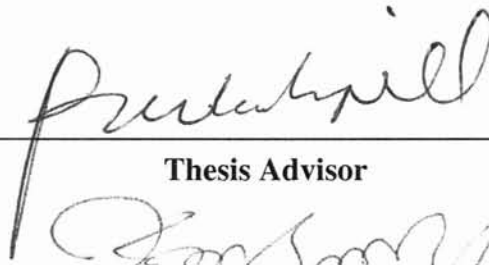
Seoul, Korea

2000

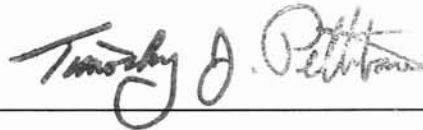
**Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2002**

**MODELING AND EVALUATION OF THE INTER-
CONNECTION - DRIVEN REPAIRABILITY
FOR DISTRIBUTED EMBEDDED
MEMORY CORES ON CHIP**

Thesis Approved :



Thesis Advisor



Dean of the Graduate College

ACKNOWLEDGMENTS

I truly would like to express my appreciation to my advisor, Dr. N. Park, for his intelligent supervision, great guidance, wonderful inspiration, and kind friendship. My sincere appreciation extends to my other committee members, Dr. K. M. George and Dr. G. E. Hedrick for their excellent supervision and guidance. I also would like to thank all my computer science department colleagues for their kind co-operation.

Special thanks go to my father in Heaven and my mother in Korea for their precious support and encouragement. I also would like to express my appreciation to all my family for their wonderful care. I especially would like to thank to my brother, Chan-hoi, for his great sacrifice for my family.

Finally, I would like to thank the Department of Computer Science for their qualified and advanced education.

TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION	1
II. LITERATURE REVIEW	5
1. Memory Repair	5
2. Fault Clustering	5
3. SoC Test & Repair	6
4. Effect of Interconnection in Test & Repair of System	6
III. DISTRIBUTED EMBEDDED MEMORY CORES AND INTERCONNECTION	8
IV. REPAIR ALGORITHM	16
V. RELIABILITY ANALYSIS	24
VI. PARAMETRIC SIMULATION	31
VII. CONCLUSION AND DISCUSSION	37
REFERENCES	39

LIST OF FIGURES

Figure	Page
1. BIST/BISD/BISR model for SoC	2
2. Distributed memory architectural model of SoC	10
3. Serial Interfacing Scheme for BIST/BISD/BISR	12
4. Distributed memory interconnection topology in SoC	13
5. Rerouting overhead after reconfiguration	15
6. Adjacency matrix representation of figure 2	18
7. Adjacency list representation of figure 2	19
8. Shortest distance matrix representation of figure 2	20
9. Spare column table	21
10. The Best Spare Column Selection (BSCS) Algorithm	22
11. The proposed flowchart of distributed embedded memory core repair of SoC with <i>BSCS</i>	23
12. Reliability of SoC at $S = 2, 4, 8, 16$	33
13. Reliability of SoC at $P_{FP} = 0.1, 0.5, 0.9$	34
14. Rerouting overhead	35
15. Reliability of SoC at $\mu = 1, 10, 100$	36

NOMENCLATURE

ASIC	Application Specific Integrated Circuit
BISD	Built in Self Diagnosis
BISR	Built in Self Repair
BIST	Built in Self Test
ECC	Error Correcting Code
IC	Integrated Circuits
ITRS	International Technology Roadmap in Semiconductor Association
MCM	Multi Chip Module
Repairability	The probability that the system can be repaired throughout a repair process
Reliability	The probability that the system operates correctly throughout a complete interval of time
SoC	System-on-Chip
Testability	The ability to test for certain attributes within a system
Topology	Specific physical or logical arrangement of the elements of a network
VLSI	Very Large Scale Integration
Birthday problem	A probability theory about coincidence: it gets its name from the fact that among 22 randomly selected people there is a better than 50% chance that two of these people have the same birthday. For 40 people this chance increases to 90%, and with 70 people it is almost certain

Chordal ring An augmented ring or a circulant graph. Formally it is defined by the pair (n, L) where n is the number of nodes of the ring and L is the set of chords. When each chord $l \in L$ connects every pair of nodes of the ring that are at distance l in the ring, it is called chordal ring of degree l

LIST OF ACRONYMS & NOTATIONS

<i>BSCS</i>	Best Spare Column Selection Algorithm
<i>FP</i>	Fault Prone memory module core
<i>FR</i>	Fault Resistant memory module core
N_m	Number of memory module cores in SoC
N_c	Number of columns in a memory core, excluding spares
N_r	Number of rows in a memory core
S	Number of spare columns
P_{FP}	Probability that a module is FP, $\Pr\{\text{module is FP}\}$
η	Degree of Connectivity
λ	Failure rate of a memory cell in normal state
λ_{FP}	Failure rate of a memory cell in FP memory module core
λ_{FR}	Failure rate of a memory cell in FR memory module core
λ_{inter}	Failure rate of memory interconnection per unit distance
μ	Rerouting overhead (simplified as a distance in this work)

CHAPTER I

INTRODUCTION

System-on-Chip technology is driving the VLSI (Very-Large-Scale-Integration) industry today. Chip designers can implement everything from processors, memories to bus interfaces onto a single silicon chip with deep submicron technology. The advantages of SoC over its multi-chip alternatives are many, among which are higher performance, lower power consumption, and smaller volume and weight. SoC may embed CPUs, application-specific cores, and multiple embedded memory cores, to mention a few.

While SoC offers a variety of advantages, there still exists difficulties with test & repair since it is so much more complex and different from the previous IC generations to be tested by the conventional methods. Current test methods mainly rely on external testers, and as the complexity of IC grows and technology advances, the test & repair cost consequently results in as much as 30 - 40% of the total production cost [3]. Due to these problems, the semiconductor manufacturing industry is heading toward Design-for-Testability (DFT). The irreversibility of SoC fabrication and implementation process doesn't allow embedded cores be reworked physically and hence SoC finds its practical solution for DFT by employing Built-in Self-Test / Diagnosis / Repair (BIST / BISR / BISR).

Such design exploits the multiple small sized embedded memory cores that are distributed throughout the chip area to enhance the space utilization [13]. Figure 1 illustrates the block diagram of BIST / BISR / BISR for the distributed embedded memory cores: A Shared BIST / BISR / BISR that is designed to minimize the area penalty independent of the number of embedded memory cores [12].

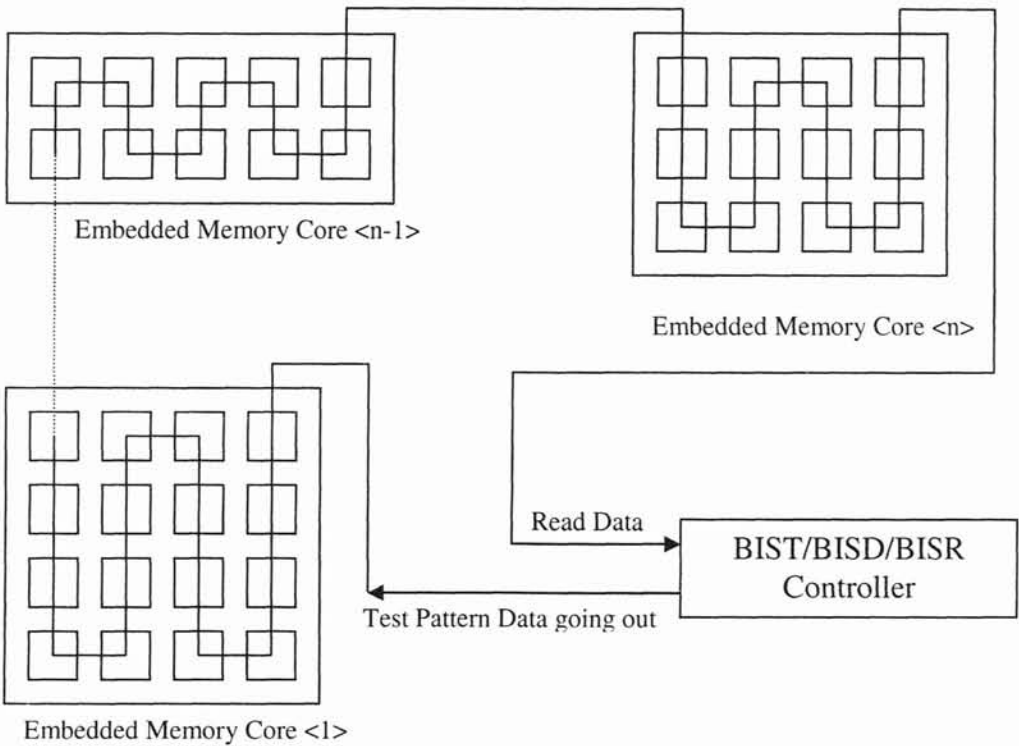


Figure 1. BIST/BISR/BISR model for SoC (adopted from [12])

The BIST controller sends the input test data through serially connected memory modules, and BISR diagnoses the output data coming back to the controller. BISR controller then conducts a repair process by mapping the addresses of the memory cells diagnosed as faulty onto new addresses by utilizing available spare columns.

Memory cores are the most space-dominant among numerous cores in SoC. ITRS (International Technology Roadmap for Semiconductor) forecasts that more than 90% of the SoC area will be populated with embedded memory cores in a decade. Due not only to this fact but the fact that memory is a fault-prone component on SoC, the reliability of memory cores play a critical role in overall reliability of SoC.

In general, memory repair consists of two mechanisms : one is *hard repair* usually done by laser fuse box controls: fuses / antifuses / laser programming are used to disconnect rows and / or columns with faulty bits and replaces them with redundant rows and / or columns. The second mechanism is *soft repair*, such as software-driven reconfiguration: soft repair addresses a mapping procedure to cover faulty address location by redundant rows and / or columns. Hard repair mainly takes place during fabrication time by memory vendors to enhance its manufacturing yield and soft repair takes place during field time to maintain high reliability of the system. Every time power is switched on, memory test & repair processes are conducted under BIST / BISR / BISR controller's control.

The objective of this paper is to study the repair process of distributed embedded memory cores in SoC. An efficient repair algorithm driven by the *spare line borrowing*

(software-side reconfiguration) technique is proposed with a focus on the effect of interconnection topology on the reliability of distributed embedded memory cores. The overhead for the proposed repair algorithm will be analyzed through combinatorial modeling and extensive parametric simulations to justify the effectiveness of the proposed interconnection topology.

CHAPTER II

LITERATURE REVIEW

1. Memory Repair

Since memory modules play a critical role in electronic devices and they are the fault-prone component in the system, there have been significant efforts made on memory repair to achieve higher reliability. Lombadi, F. and Huang, W.K. [7] proposed the approaches for the repair of redundant RAMs in which redundant rows and columns are utilized as spares and Stapper, C.H. and Lee, H.-S. [6] proposed that the combined use of redundant circuits and error-correcting codes (ECC) can achieve a fault-tolerance scheme that is far more effective than an individual employment of either one of these schemes separately (so called *synergistic fault tolerance*), which is obtained by eliminating the so-called “birthday problem” which limits the effectiveness of error-correcting codes.

2. Fault Clustering

Based on the fact that faults on a chip tends to occur in clustering patterns, fault clustering has been studied extensively along with memory faults to take into account practical fault patterns. Blough, D.M. [1][2] proposed a clustered failure model under which the problem of array reconfiguration is studied. Blough, D.M. [2] adopted center-

satellite approach while Blough, D.M. [1] adopted quadrat-based fault model. The quadrat-based model is preferred to the center-satellite model in practice since many parameters used in the center-satellite model makes its parameter estimation difficult [1]. Choi, M. and Park, N. etc. [8] proposed a reliability model of fault-tolerant onboard memory system under fault clustering where spare columns and spare modules are implemented.

3. SoC Test & Repair

With the emergence of SoC in the semiconductor industry, test & repair of SoC and reliability of SoC have become very important issues during the development process. Marinissen, E.J. and Zorian, Y. [9] addressed the challenges in testing core-based system ICs by describing and comparing the differences between traditional test methods and core-based test methods. Marinissen, E.J. and Zorian, Y. [9] identified and summarized the challenges in SoC such as testing, standardization, tool development, insertion of a wrapper around a given core, compliancy check of core plus wrapper, interconnect test pattern generation, test access planning and synthesis and so on. Due to these difficulties, little research has been done on this topic even though many researchers are interested in it.

4. Effect of Interconnection in Test & Repair of System

There have been a few works which deal with the effect of interconnections in the system. Choi, M. and Park, N. etc. [10] proposed a repair method based on the connectivity of the chips on MCM (Multi-chip Module) in which yield degradation due to neighboring chips and interconnect structures were modeled and analyzed. Several MCM repair scheduling strategies based on the number of interconnections and the number of neighboring chips are shown in [10], and they evaluated the impact of connectivity-based repair scheduling on the overall yield of MCMs. In [11] the reliability of SoC design when bus errors affect the SoC interconnection architecture is addressed and an approach to enhance the error detection and correction mechanism of the system bus is proposed based on the concept of distributed bus guardians.

There is no adequate work reported on the repair of distributed embedded memory cores on SoC and the effect of interconnections on the overall reliability.

CHAPTER III

DISTRIBUTED EMBEDDED MEMORY CORES AND INTERCONNECTION

In this chapter, the model of distributed embedded memory cores under investigation is introduced and analyzed.

The chordal ring of degree 4 distributed embedded memory core architecture in SoC is introduced in figure 2. Note that, for simplicity, only distributed memory module cores and BIST / BISR / BISR controller is shown. Each memory core module has S number of spare columns to be used for repair of either its own faults or other module's faults and is connected with two adjacent modules and two distant modules; hence, a total four adjacent connected modules so the degree of connectivity is four. The number on each interconnection line in the figure indicates the distance between two connected modules.

Single Built-In Self-Repair (BISR) Controller coupled with single Built-In Self-Test (BIST) and Built-In Self-Diagnosis (BISR) is preferred in SoC design due to the space problem (A shared BIST / BISR / BISR) [12]. Memory tests are conducted under the control of the BIST, and then the BISR diagnoses and locates the fault addresses which are passed over to the BISR to be repaired by spare columns. It is assumed that BIST / BISR can detect all of the faults in the distributed memory module cores. Testing overhead such as testing time and algorithm complexity is not considered here. There are a number of efficient BIST / BISR algorithms available such as March, RSMarch [13],

and DiagRSMarch [4]. It is also assumed that each memory core is the same size and of the same kind to make the spare line borrowing technique simple. Reconfiguration would become far more complicated if they are of different sizes or different kinds. Each Memory core has only column redundancy to keep the repair algorithm complexity tractable and maximize the hardware utilization. Note that column redundancy has a greater functional fault coverage than row redundancy [14]. Spare columns are effective for faults in a bit line, column multiplexer, sense amplifier, data input register, data output register and column line decoder; whereas, spare rows are effective for faults in a word line, word line driver and word line decoder [14].

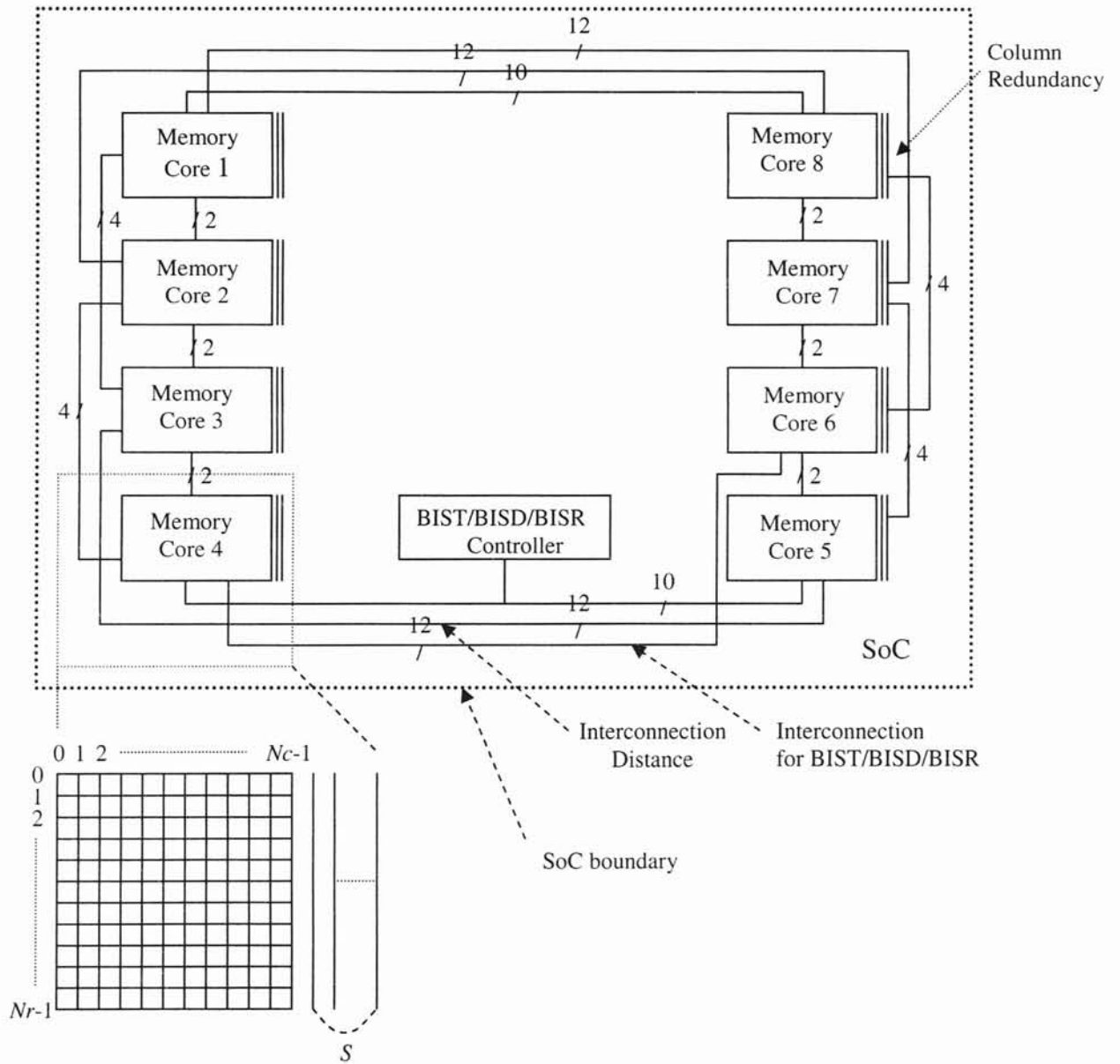


Figure 2. Distributed memory architectural model of SoC
(Chordal ring of degree 4 topology)

A serial interfacing scheme [5] also is preferred in SoC. Figure 3 shows only two extra test lines are added to the original memory architecture so that the hardware overhead can be tolerable.

The interconnection topology investigated in this work is ring based (i.e. degree of connectivity 2) and it goes to higher degree of connectivity such as chordal ring of degree 3 (degree of connectivity 3), chordal ring of degree 4 (degree of connectivity 4) etc. Ring based interconnection topology provides the benefits: its dedicated bus scheme and symmetric interconnection structure is efficient to analyze and show the effect of connectivity on the repairability. Figure 4 shows several topologies of distributed embedded memory module cores in SoC.

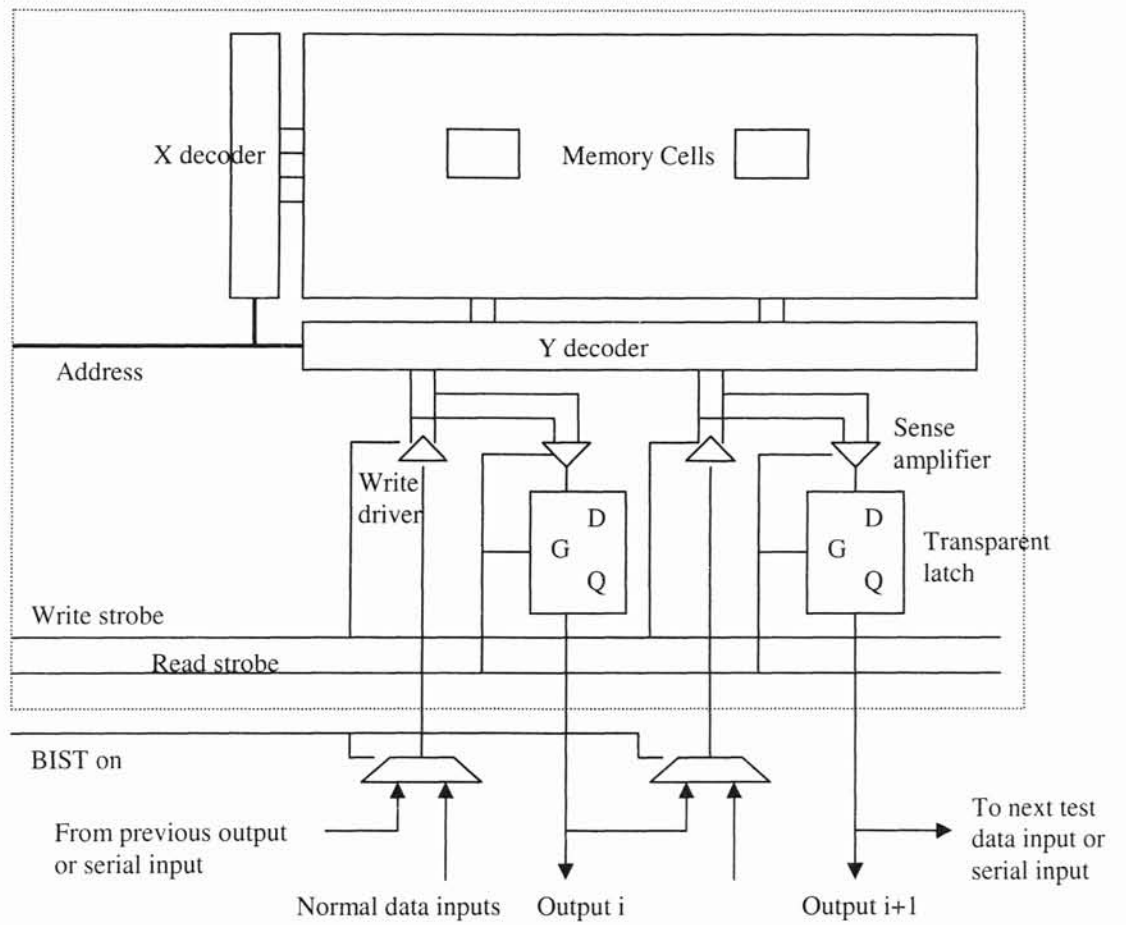
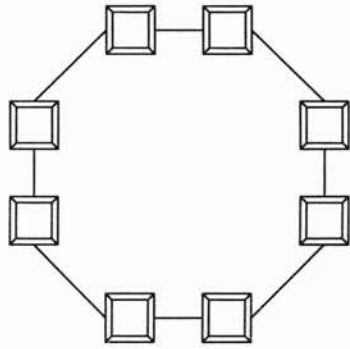
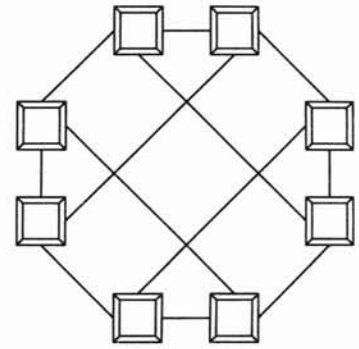


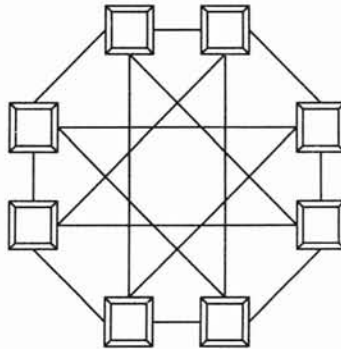
Figure 3. Serial Interfacing Scheme for BIST/BISD/BISR (adopted from [5])



Ring topology



Chordal ring of degree 3



Chordal ring of degree 4

Figure 4. Distributed memory interconnection topology in SoC

(adopted from [15])

Rerouting overhead is an important factor to consider during software-driven reconfiguration. Excessive rerouting overhead is undesirable because it degrades the system performance significantly under the stringent time-to-market constraint. The rerouting overhead under consideration in this work is illustrated in figure 5. Suppose the CPU is to retrieve the data at address “i” in the memory module “X” (note that “i” has been diagnosed to be faulty), and then it has been replaced with a location in a spare column in the memory module “Y” after the reconfiguration process. The memory request for address “i” checks memory module “X” first, and then the memory reconfiguration control unit (i.e. BISR) rerouts it to memory module “Y” for the new location of address “i”. This is where rerouting overhead occurs.

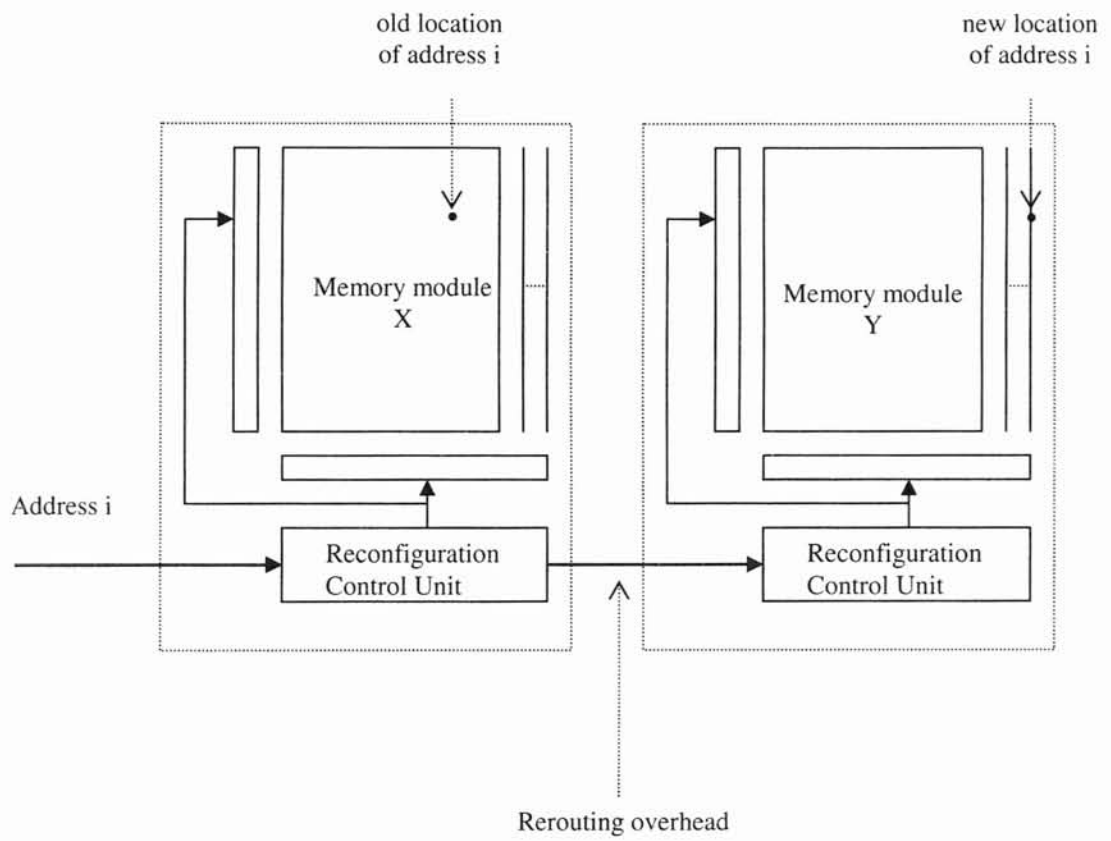


Figure 5. Rerouting overhead after reconfiguration

CHAPTER IV

REPAIR ALGORITHM

In this section, the proposed repair algorithm is presented.

When a module does not have its own spare columns available, it should borrow spare lines from other modules through a reconfiguration process. In order to optimize the rerouting process caused by repair (reconfiguration), in each repair cycle, selecting an appropriate borrowing candidate spare line with the least rerouting overhead is a desirable choice. From figure 5, rerouting overhead is defined by the distance from the module where its destination address is originally located, to the module where the spare line is borrowed from. In order to reduce this rerouting overhead, the reconfiguration controller should choose the nearest available spare column as a repair candidate. For this implementation, the weighted graph representation such as an adjacency matrix or adjacency list can be used. Figure 6 shows the adjacency matrix representation for Figure 2. Each cell in the adjacency matrix denotes the distance between the two modules, each of which is represented by the i 'th row and j 'th column in the matrix, respectively. The cell which has a " ∞ " value indicates that two corresponding row and column modules are not directly connected, and local rerouting overhead is represented by "0".

In an adjacency matrix representation we have $O(N^2)$ space requirement where N is the number of memory modules in the SoC. If the memory module is sparsely interconnected,

then an adjacency list representation would be more efficient in terms of its space requirement which is $O(N + E)$ where N is the number of memory modules and E is the number of edges between memory modules in the SoC. Figure 7 shows the adjacency list representation of figure 2.

By using this adjacency matrix representation the shortest distance from each module can be obtained. There are a number of algorithms to find the shortest path, and in this work a modified version of Warshall's algorithm is used. Figure 8 shows the shortest distance matrix for figure 2.

	M 1	M 2	M 3	M 4	M 5	M 6	M 7	M 8
M 1	0	2	4	∞	∞	∞	12	10
M 2	2	0	2	4	∞	∞	∞	12
M 3	4	2	0	2	12	∞	∞	∞
M 4	∞	4	2	0	10	12	∞	∞
M 5	∞	∞	12	10	0	2	4	∞
M 6	∞	∞	∞	12	2	0	2	4
M 7	12	∞	∞	∞	4	2	0	2
M 8	10	12	∞	∞	∞	4	2	0

Figure 6. Adjacency matrix representation of figure 2.

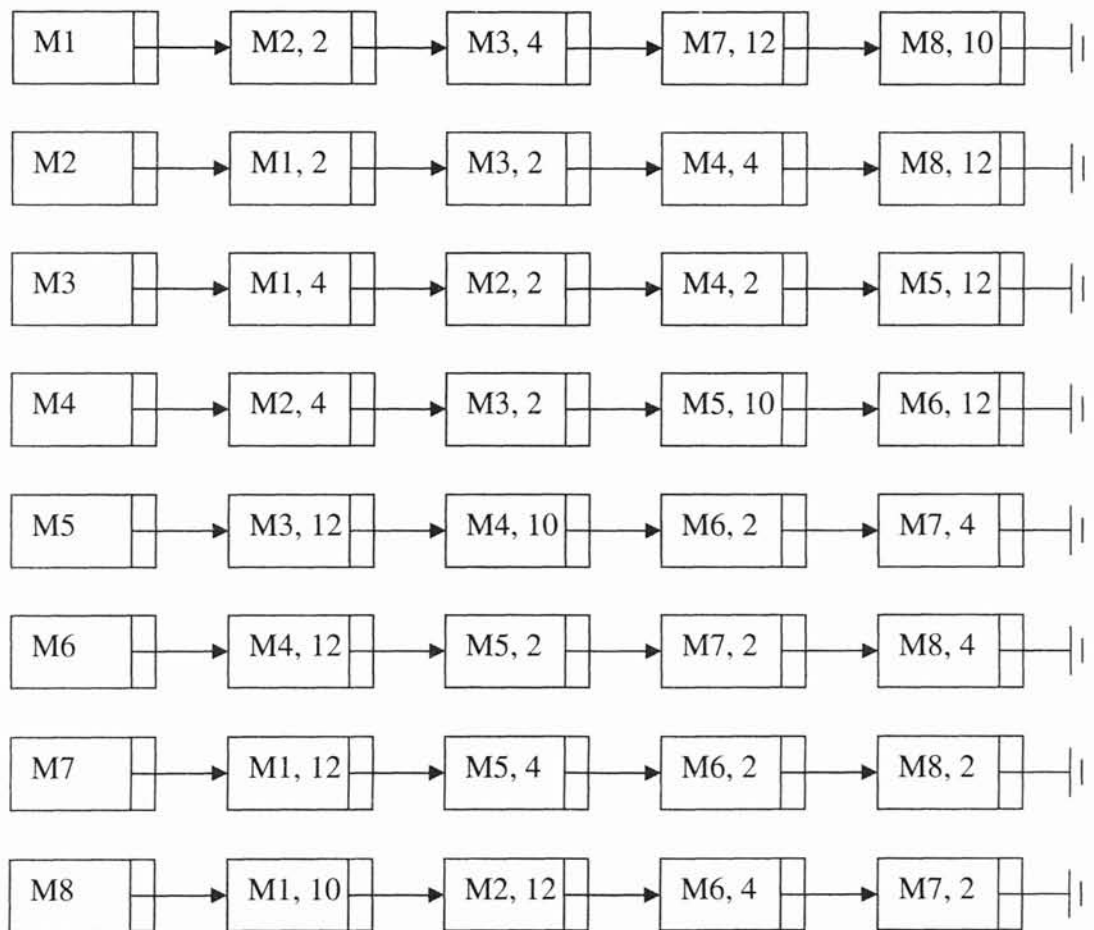


Figure 7. Adjacency list representation of figure 2.

	M 1	M 2	M 3	M 4	M 5	M 6	M 7	M 8
M 1	0	2	4	6	16	14	12	10
M 2	2	0	2	4	14	16	14	12
M 3	4	2	0	2	12	14	16	14
M 4	6	4	2	0	10	12	14	16
M 5	16	14	12	10	0	2	4	6
M 6	14	16	14	12	2	0	2	4
M 7	12	14	16	14	4	2	0	2
M 8	10	12	14	16	6	4	2	0

Figure 8. Shortest distance matrix representation of figure 2

After establishing the shortest distance matrix, the algorithm checks the availability of spares to borrow with the aid of the spare column table in figure 9. The spare column table looks like a one dimensional array as shown in figure 9 and it keeps track of the number of spare columns available for each memory module by decreasing by one whenever it is used for a repair.

M1	M2	M3	M4	M5	M6	M7	M8
S	S	S	S	S	S	S	S

Figure 9. Spare column table

Based on the proposed manipulation of the adjacency matrix, shortest distance matrix, and spare column table, an algorithm which utilizes the spare columns optimally, referred to as the Best Spare Column Selection Algorithm (*BSCS*), is proposed in figure 10 along with its overall flowchart of distributed memory repair in SoC shown in figure 11.

Algorithm BSCS

Input : FM (faulty module), Adjacency Matrix Adj[[]], Spare Column Table ST[[]]

Output : A pair (Boolean R, module number of best spare column)

Begin

Path[[]] = Adj[[]];

for k = 0 to N-1

for i = 0 to N-1

for j = 0 to N-1

Path[i][j] = min(Path[i][j], (Path[i][k] + Path[k][j]));

if ST[0] ≠ 0

candidate = 0;

else

candidate = null;

for i = 0 to N-1

if (Path[FM-1][i+1] < Path[FM-1][i]) && (ST[i+1] ≠ 0)

candidate = i+1;

if candidate = null

return (false, null);

else

ST[candidate] = ST[candidate] - 1;

return (true, (candidate + 1));

End of Algorithm *BSCS*

Figure 10. The Best Spare Column Selection (*BSCS*) Algorithm

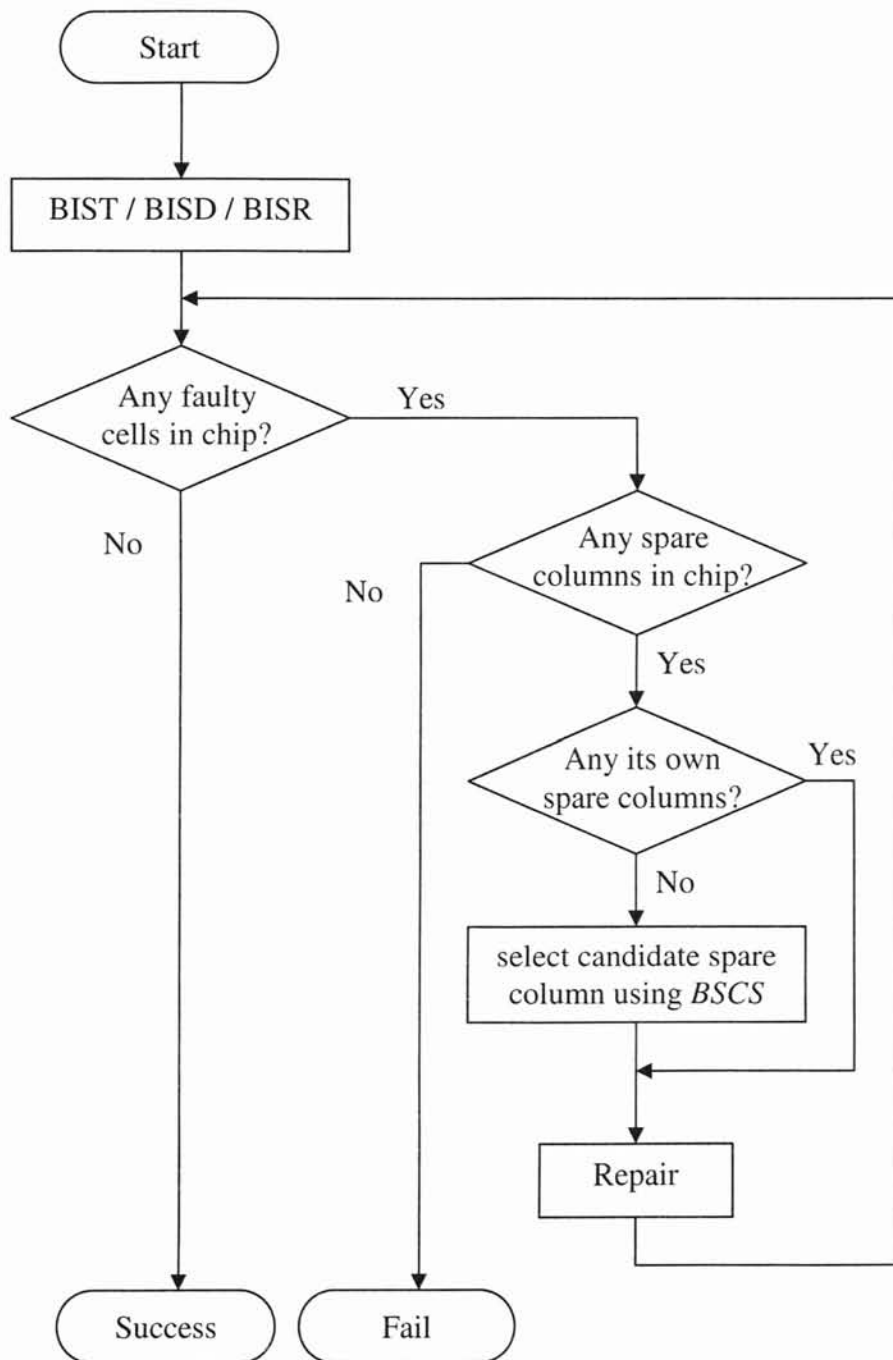


Figure 11. The proposed flowchart of distributed embedded memory core repair of SoC with *BSCS*

CHAPTER V

RELIABILITY ANALYSIS

In this section the reliability of the distributed embedded memory cores is analyzed based on the proposed repair algorithm. The expected number of memory cell failures for a given time period Δt is numerically defined by the *failure rate*. The exponential relationship between reliability and time is the *exponential failure law* which claims that for a constant failure rate, the component reliability changes exponentially as a function of time t . Therefore, with a fault arrival rate λ , the reliability is denoted as

$$R(t) = e^{-\lambda t}$$

The structure function of a system can be modeled by using the series and parallel configuration of the components. In a series configuration, each component of the system is required to operate correctly for the whole system to operate correctly. In a parallel configuration, on the other hand, at least one of the parallel components should be operational for the whole system to perform its functions correctly. The series configuration is suitable to model a system which contains no spare components so the reliability of the series system become the probability that all of the elements are working properly as expressed below.

$$R_{\text{series}}(t) = R_1(t) \cdot R_2(t) \cdots R_N(t)$$

$$\text{or } R_{\text{series}}(t) = \prod_{i=1}^N R_i(t)$$

On the other hand, in parallel configuration, at least one of N identical components are required for the whole system to be operational as expressed as follows,

$$R_{\text{parallel}}(t) = 1.0 - \prod_{i=1}^N (1.0 - R_i(t))$$

Based on the above two reliability expressions, the reliability of $M - N$ systems where M components should operate properly in order for the whole system to be operational can be expressed as follows, using the binomial distribution without loss of generality.

$$R_{M-N}(t) = \sum_{i=0}^{N-M} \binom{N}{i} (R(t))^{N-i} (1.0 - R(t))^i$$

Distributed embedded memory core architecture in SoC consists of memory modules and interconnections between modules, and these two components are fully dependent on each other with regard to whole memory system reliability, which can be modeled as a series system. Therefore, the reliability of distributed embedded memory cores can be expressed as follows.

$$R_{\text{memory_system}}(t) = R_{\text{module}}(t) \cdot R_{\text{interconnection}}(t)$$

For simplicity, it is assumed that the failure rate of each interconnection line is proportional to the distance (i.e. interconnection length). In order to model the degradation between the reliability before and after the repair, the initial reliability of an interconnection line prior to repair is considered to be 1 and the reliability degradation caused by rerouting overhead only is

$$R_{\text{interconnection}}(t) = e^{-\mu \lambda_{\text{inter}} t}.$$

The reliability of a memory module with S spare columns can be expressed by a combinatorial model:

$$R_{\text{module}}(t) = \sum_{i=0}^S \binom{N_c + S}{i} (R_{\text{column}}(t))^{N_c + S - i} (1.0 - R_{\text{column}}(t))^i$$

With the proposed spare line borrowing technique, the number of spare columns available for each module may be greater than S depending on the interconnection topology. The probability that a column in a module is faulty is denoted as follows.

$$\begin{aligned} & \Pr \{ \text{a column in a module is faulty} \} \\ &= 1 - \Pr \{ \text{a column in a module is fault-free} \} \end{aligned}$$

Since each column contains N_r cells, the probability that a column in a module is fault-free is $(1 - \lambda)^{N_r}$. Thus, $\Pr \{ \text{a column in a module is faulty} \}$ is

$$1 - (1 - \lambda)^{N_r}$$

The total expected number of faulty columns in a module can be expressed as follows.

$$N_c \cdot (1 - (1 - \lambda)^{N_r})$$

Suppose there are N_m memory modules, and each memory module has the same fault rate λ . The spare columns that a module can lend to other modules are the remaining ones after its own repair. The total number of spare columns available to a module in this case then is as follows.

$$S' = N_m \cdot \{ S - (N_c \cdot (1 - (1 - \lambda)^{N_r})) \}$$

The reliability of a module can be expressed as follows.

$$R'_{\text{module}}(t) = \sum_{i=0}^{S'} \binom{N_c + S'}{i} (R_{\text{column}}(t))^{N_c + S' - i} (1.0 - R_{\text{column}}(t))^i$$

Therefore, the overall reliability of the distributed embedded memory cores can be expressed as follows.

$$R_{\text{memory_system}}(t) = \left(\sum_{i=0}^{S'} \binom{N_c + S'}{i} (R_{\text{column}}(t))^{N_c + S' - i} (1.0 - R_{\text{column}}(t))^i \right) \cdot (e^{-\mu \cdot \lambda_{\text{inter}} \cdot t})$$

In case of fault clustering, we have *FP* modules with probability P_{FP} and *FR* modules with probability $(1 - P_{FP})$. For a *FP* module, we have the probability that a column in a *FP* module is expressed as follows.

$$\begin{aligned} & \Pr \{ \text{a column in a } FP \text{ module is faulty} \} \\ &= 1 - \Pr \{ \text{a column in a } FP \text{ module is fault-free} \} \\ &= 1 - (1 - \lambda_{FP})^{N_c} \end{aligned}$$

Then the total expected number of faulty columns in a *FP* module can be expressed as follows.

$$N_c \cdot (1 - (1 - \lambda_{FP})^{N_c})$$

Similarly, for each *FR* module, $\Pr \{ \text{a column in a } FR \text{ module is faulty} \} = 1 - \Pr \{ \text{a column in a } FR \text{ module is fault-free} \}$. That is,

$$1 - (1 - \lambda_{FR})^{N_c}$$

Then the total expected number of faulty columns in a FP module can be expressed as follows.

$$N_c \cdot (1 - (1 - \lambda_{FR})^{N_r})$$

Since we have the probability that a module is a FP is P_{FP} , the total expected number of faulty columns in a module is expressed as follows.

$$P_{FP} \cdot (N_c \cdot (1 - (1 - \lambda_{FP})^{N_r})) + (1 - P_{FP}) \cdot (N_c \cdot (1 - (1 - \lambda_{FR})^{N_r}))$$

Then the total number of spare columns available to a module in a fault clustering case is as follows

$$S' = N_m \cdot \{S - \{P_{FP} \cdot (N_c \cdot (1 - (1 - \lambda_{FP})^{N_r})) + (1 - P_{FP}) \cdot (N_c \cdot (1 - (1 - \lambda_{FR})^{N_r}))\}\}$$

The reliability of a module can be expressed as follows.

$$R'_{\text{module}}(t) = \sum_{i=0}^{S'} \binom{N_c + S'}{i} (R_{\text{column}}(t))^{N_c + S' - i} (1.0 - R_{\text{column}}(t))^i$$

Therefore, the overall reliability of the distributed embedded memory cores in a fault clustering case can be expressed as follows.

$$R_{\text{memory_system}}(t) = \left(\sum_{i=0}^{S'} \binom{Nc+S'}{i} (R_{\text{column}}(t))^{Nc+S'-i} (1.0 - R_{\text{column}}(t))^i \right) \cdot (e^{-\mu \cdot \lambda_{\text{inter}} \cdot t})$$

CHAPTER VI

PARAMETRIC SIMULATION

In this chapter, the effect of the interconnection topology on system reliability is investigated through parametric simulation.

Figure 12 shows the reliability of a memory system in SoC versus the number of spare columns in which the same fault rate is assumed for each module; that is, no fault clustering. Higher reliability can be obtained as spare columns are added in each module. Taking into consideration that time at x-axis are months as industrial experiments indicate, the memory module where spare columns equal 16 can be tolerable up to 52 months, while the memory module where spare columns equal 2 can be tolerable for only 5 months. Figure 13 shows the reliability of a memory system in SoC according to the probability that a module is fault prone. Similarly, the reliability of memory module cores in SoC drops very quickly when the probability that a module is fault prone is high. With $P_{FP} = 0.1$, the memory module cores stay in a reliable state for up to 65 months while the memory module cores stay in a reliable state for only 15 months with $P_{FP} = 0.9$.

To demonstrate the effectiveness of the proposed repair algorithm, a random spare column selection algorithm is simulated and compared with the proposed *BSCS* algorithm. Figure 14 shows the rerouting overhead of the model in figure 2 where $S = 3$ is assumed. Figure 14 shows that the *BSCS* algorithm reduces the rerouting overhead compared to the

random selection algorithm, and note that the *BSCS* further reduces the rerouting overhead in the range where the number of faulty columns are small.

Figure 15 shows the overall reliability of the memory system in SoC taking the rerouting overhead into account. From the simulation results, it is observed that the rerouting overhead affects the reliability of the memory system significantly. The reliability is so sensitive to the rerouting overhead. It drops steeply with large amount of rerouting overhead. Therefore, it is concluded that the repair process for distributed embedded memory cores must be driven adequately by their interconnection topology to minimize the rerouting overhead.

λ	λ_{inter}	N_c	N_r	S	η	μ
0.003	0.0005	32	8	2, 4, 8, 16	8	0.01

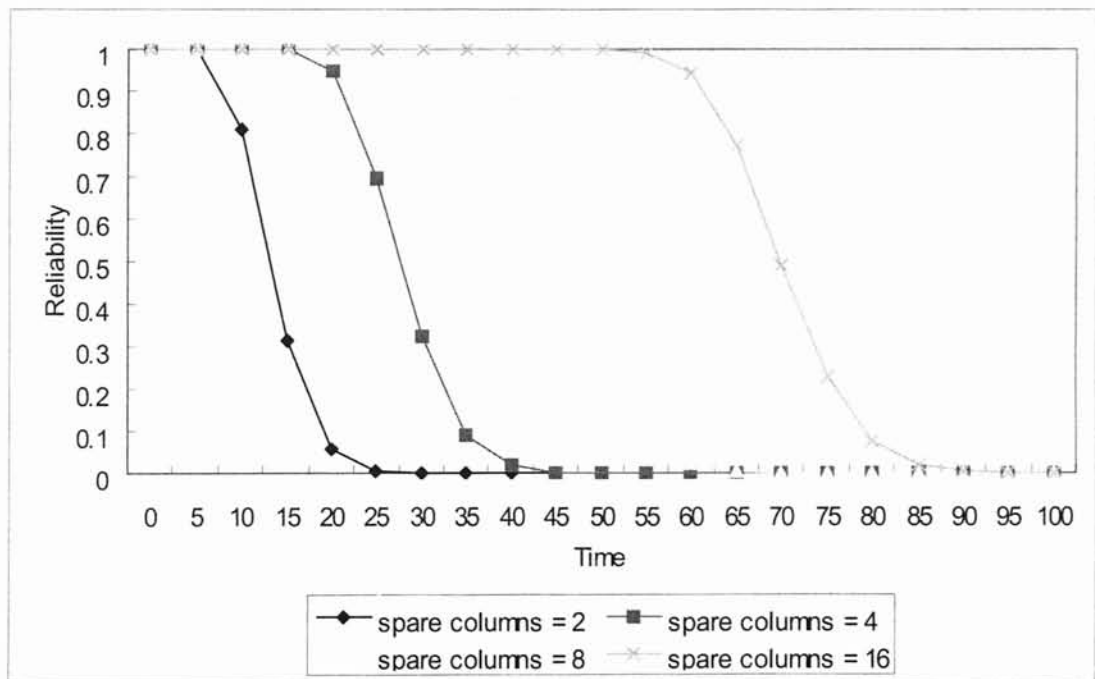


Figure 12. Reliability of SoC at $S = 2, 4, 8, 16$

λ_{FP}	λ_{FR}	λ_{inter}	N_c	N_r	S	η	μ	P_{FP}
0.006	0.001	0.0005	32	8	8	8	0.01	0.1, 0.5, 0.9

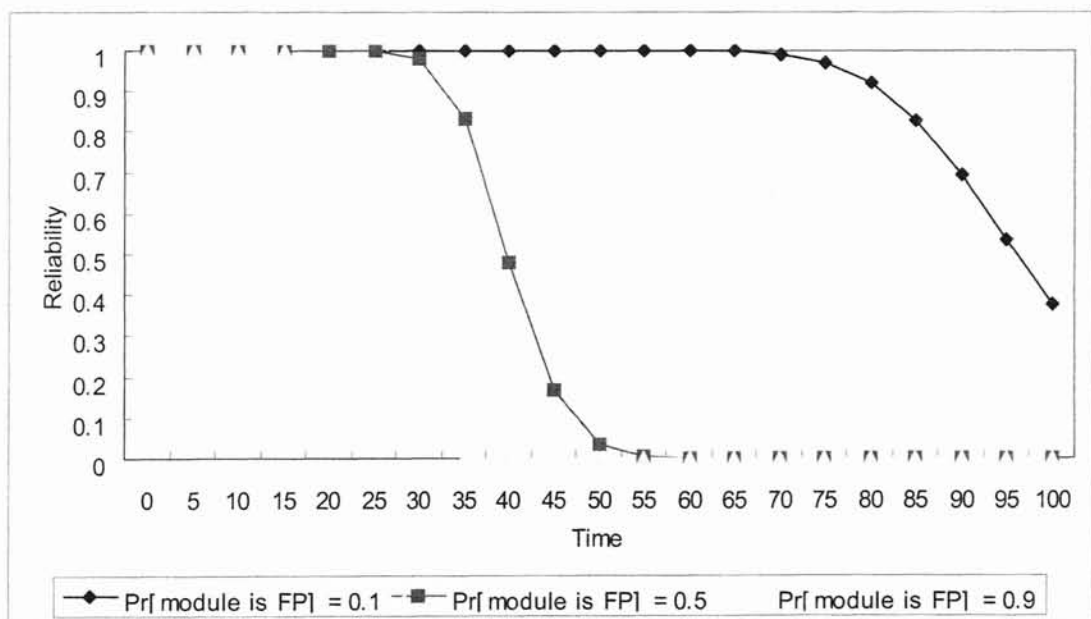


Figure 13. Reliability of SoC at $P_{FP} = 0.1, 0.5, 0.9$

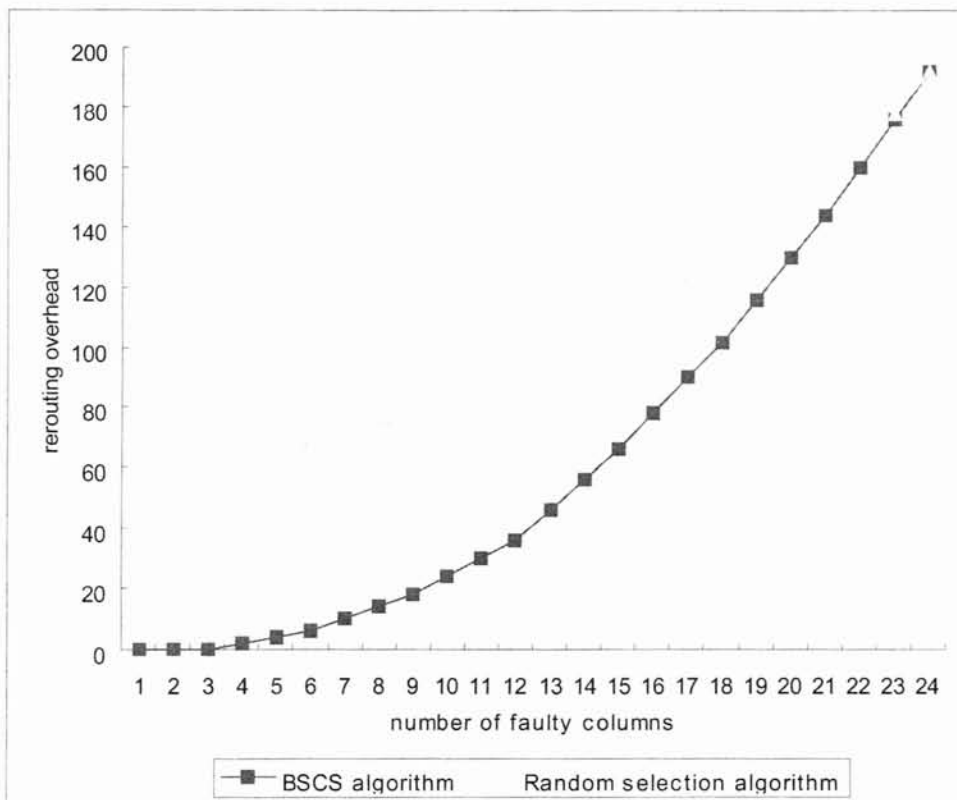


Figure 14. Rerouting overhead

λ	λ_{inter}	N_c	N_r	S	η	μ
0.003	0.0005	32	8	8	8	1, 10, 100

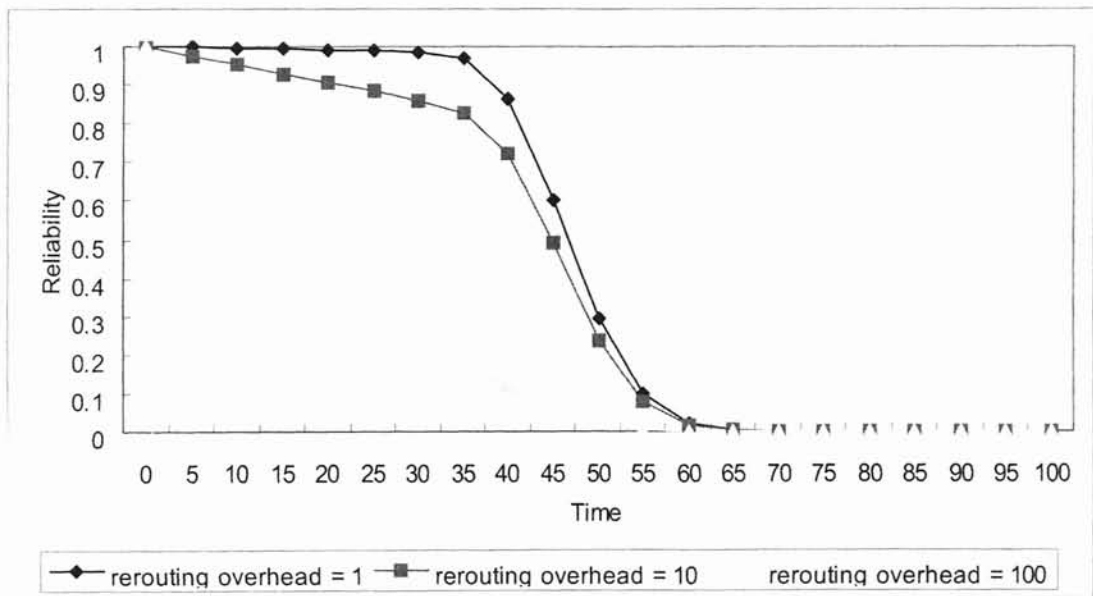


Figure 15. Reliability of SoC at $\mu = 1, 10, 100$

CHAPTER VII

CONCLUSION AND DISCUSSION

This paper has proposed a method to model and evaluate the interconnection-driven repair process for distributed embedded memory cores in SoC. The effect of interconnection topology on the reliability of distributed embedded memory modules has been evaluated through parametric simulation with respect to the proposed repair algorithm to exploit the utilization of distributed spare columns over the interconnection network.

Parametric results demonstrated that memory module interconnection affects the system reliability by utilization of the spare columns in the memory modules distributed over the interconnection network. It has been observed that the reliability of memory system increases as the number of spare columns increase and the probability that a module is fault prone decreases. Especially, it is noticeable that rerouting overhead significantly affects not only the system performance but also system reliability. It also has been shown that the proposed *BSCS* (Best Spare Column Selection) algorithm can reduce the rerouting overhead.

Therefore, it is concluded based on the modeling and analysis in this paper that memory interconnection rerouting overhead is a major factor in determining the feasibility of a

distributed embedded memory core design from the reliability's standpoint, which also requires a practical cost justification.

REFERENCES

- [1] Blough, D.M., "Performance Evaluation of a Reconfiguration-Algorithm for Memory Arrays containing Clustered Faults", *IEEE Transactions on Reliability*, Vol. 45, No. 2, June 1996, Page(s): 274-284.
- [2] Blough, D.M.; Pelc, A., "A Clustered Failure Model for the Memory Array Reconfiguration Problem", *IEEE Transactions on Computers*, Vol. 42, No. 5, May 1993, Page(s): 518-528.
- [3] Chandramouli, R.; Pateras, S., "Testing systems on a chip", *IEEE Spectrum*, Vol. 33, Issue 11, Nov. 1996, Page(s): 42-47.
- [4] Huang, D.C.; Jone, W.B., "A parallel built-in self-diagnostic method for embedded memory arrays", *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, Vol. 21, Issue: 4, April 2002, Page(s): 449-465.
- [5] Nadeau-Dostie, B.; Silburt, A.; Agarwal, V.K., "Serial interfacing for embedded-memory testing", *IEEE Design & Test of Computers*, Vol. 7, Issue: 2, April 1990, Page(s): 52-63.
- [6] Stapper, C.H.; Lee, H.-S., "Synergistic fault-tolerance for memory chips", *Computers, IEEE Transactions on*, Vol. 41, Issue: 9, Sept. 1992, Page(s): 1078-1087.
- [7] Lombardi, F.; Huang, W.K., "Approaches for the repair of VLSI/WSI RRAMs by row / column deletion", *Fault-Tolerant Computing, 1988. FTCS-18, Digest of Papers., Eighteenth International Symposium on*, 1988, Page(s): 342-347.
- [8] Choi, M.; Park, N.; Meyer, F.J.; Lombardi, F.; Piuri, V., "Reliability measurement of fault-tolerant onboard memory system under fault clustering", *Instrumentation and Measurement Technology Conference, 2002. IMTC/2002. Proceedings of the 19th IEEE*, Vol. 2, 2002, Page(s): 1161-1166.
- [9] Marinissen, E.J.; Zorian, Y., "Challenges in testing core-based system ICs", *IEEE Communication Magazine*, Volume: 37, Issue: 6, June 1999, Page(s): 104-109.
- [10] Choi, M.; Park, N.; Meyer, F.; Lombardi, F., "Connectivity-based multichip module repair", *Dependable Computing, 2001. Proceedings. 2001 Pacific Rim International Symposium on*, 2001, Page(s): 19-26.
- [11] Lajolo, M., "Bus guardians: an effective solution for online detection and correction of faults affecting system-on-chip buses", *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, Volume: 9, Issue: 6, Dec. 2001, Page(s): 974-982.

- [12] Ohtani, J.; Ooish, T.; Kawagoe, T.; Niuro, M.; Maruta, M.; Hidaka, H., "A shared built-in self-repair analysis for multiple embedded memories", *Custom Integrated Circuits, 2001, IEEE Conference on.*, 2001, Page(s): 187-190.
- [13] Jone, W.B.; Huang, D.C.; Wu, S.C.; Lee, K.J., "An efficient BIST method for small buffers", *VLSI Test Symposium, 1999. Proceedings. 17 th IEEE*, 1999, Page(s): 246-251.
- [14] Kim, I.; Zorian, Y.; Komoriya, G.; Pham, H.; Higgins, F.P.; Lewandowski, J.L., "Built In Self Repair for Embedded High Density Sram", *Test Conference, 1998. Proceedings., International*, 1998, Page(s): 1112-1119.
- [15] Pradhan, D.K., "Fault-tolerant Computer System Design", *Prentice Hall*, ISBN 0-13-057887-8, 1996.

VITA 2

Byung-hyun Jang

Candidate for the Degree of

Master of Science

Thesis: MODELING AND EVALUATION OF THE INTERCONNECTION-
DRIVEN REPAIRABILITY FOR DISTRIBUTED EMBEDDED
MEMORY CORES ON CHIP

Major Field: Computer Science

Biographical:

Education: Graduated from Chongju High School, Chongju, Korea in 1993;
Received Bachelor of Science Degree in Bio-Mechatronic Engineering
from Sungkyunkwan University, Seoul, Korea in February 2000;
Completed the requirements for the Master of Science degree with a major
in Computer Science at Oklahoma State University, Stillwater, Oklahoma in
December 2002.

Experience: Completed Military Service as a Soldier, Korea Army,
April 1995 to June 1997; Fulfilled Internship at Digital Media R&D center,
Samsung Electronics, Suwon, Korea, June 2002 to July 2002.

Professional Membership: The Korean Computer Scientists and Engineers
Association in America (KOCSEA).