SYSTEM-ON-CHIP DESIGN FOR RELIABILITY

By

MINSU CHOI

Bachelor of Science
Oklahoma State University
Stillwater, Oklahoma
1995

Master of Science
Oklahoma State University
Stillwater, Oklahoma
1998

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
AUGUST 2002

SYSTEM-ON-CHIP DESIGN FOR RELIABILITY

Thesis Approved:

Thesis Advisor

Timothy J. Pettibone

Dean of the Graduate College

ACKNOWLEDGMENTS

I wish to express my sincere appreciation to my dissertation advisor, Dr. Nohpill Park for his excellent supervision. My appreciation extends to the other dissertation committee members, Dr. K.M. George, Dr. G.E. Hedrick and Dr. J.M. Chung. I would like to give my special thanks to my family as well; my wife, Sujeong; my father, Jinsung; and my mother, Daeja.

TABLE OF CONTENTS

1 INTRODUCTION 1

1.1 Reliability of Embedded Cores 2

1.2 Reliability Assurance Techniques for Embedded-core-based SoC . . . 3

1.3 Reliability-driven Defect and Fault Tolerance in SoC 7

2 EMBEDDED MEMORY CORE ARCHITECTURE FOR ENHANCED MANUFACTURING YIELD AND FIELD RELIABILITY 10

2.1 Fault-Tolerant Memory Core Organization for SoC Applications . . . 12

2.2 Repair Algorithm and Procedure 14

2.3 Fabrication-Time Yield Analysis of the Proposed Fault-Tolerant Memory Core 17

2.4 Field Reliability Assurance of the Proposed Fault-tolerant Memory Core 22

2.5 Quality Assurance of the Proposed Fault-Tolerant Memory Core . . . 27

2.6 Summary 29

3 REDUNDANCY PARTITIONING FOR BALANCED MANUFACTURING YIELD AND FIELD RELIABILITY 35

3.1 Preliminaries 36

3.2 Single Dimensional Redundancy Case 37

3.3 Two Dimensional Redundancy Case 40

3.4 Parametric Simulations and Results 44

3.5 Discussion 47

4 CONNECTIVITY-BASED RECONFIGURABLE SOC REPAIR 57

4.1 Review and Preliminaries 58

4.2 Connectivity-based RSoC Repair Process 63

4.3 Connectivity-based RSoC Repair Scheduling 67

4.4 Parametric Analysis 69

4.5 Discussion 74

5 HW/SW CO-RELIABILITY IN CONFIGURABLE MULTI-PROCESSOR-MEMORY-ON-CHIP 80

5.1 Review and Preliminaries on Field Reconfigurable Multi-Processor-Memory SoCs 82

5.2 HW/SW Co-reliability Analysis 87

5.3 System Utilization Categories of the Proposed System 90

5.4 Parametric Co-Reliability Analysis 91

5.5 Discussion and Conclusions 92

6 CONCLUSION 96

REFERENCES 100

LIST OF FIGURES

2.1 SoC area share forecast of embedded memory cores from 1999 to 2014 by ITRS2000 11
2.2 Architecture of the memory block and segment 12
2.3 Logical view of the proposed fault-tolerant memory core components 14
2.4 Logical flow of the BIST/BISD/BISR procedure 16
2.5 Yield analysis results of the sample 1Mbit core 30
2.6 Reliability analysis results of the sample 16K block 31
2.7 Reliability analysis results of the sample 16K x 8 (128K) Segment ($N_s = 0$) 31
2.8 Reliability analysis results of the sample 16K x 8 (128K) Segment ($N_s = 1$) 32
2.9 Reliability analysis results of the sample 16K x 8 (128K) Segment ($N_s = 2$) 32
2.10 Analysis results of the sample 16K x 8 x 8 (1M) Core ($N_s = 0$) 33
2.11 Analysis results of the sample 16K x 8 x 8 (1M) Core ($N_s = 1$) 33
2.12 Analysis results of the sample 16K x 8 x 8 (1M) Core ($N_s = 2$) 34
2.13 MTTF analysis results of the sample 1Mbit core 34

3.1 Example of redundancy partitioning for single dimensional redundancy 40
3.2 Example of redundancy partitioning for two dimensional redundancy 41
3.3 Symmetric 1D dependability graph 49
3.4 Balanced partitioning result [4,4] 49
3.5 Reliability-intensive 1D dependability graph 50
3.6 Balanced partitioning result [3,5] 50
3.7 Yield-intensive 1D dependability graph 51
3.8 Balanced partitioning result [5,3] 51
3.9 An example 1D dependability graph 52
3.10 Balanced partitioning result [6,2] 52
3.11 Symmetric 2D dependability graph 53
3.12 Balanced partitioning result {[4,4],[4,4]} 53
3.13 Reliability-intensive 2D dependability graph 54
3.14 Balanced partitioning result {[3,5],[3,5]} 54
3.15 Yield-intensive 2D dependability graph 55
3.16 Balanced partitioning result {[5,3],[5,3]} 55
3.17 An example of 2D dependability graph 56
3.18 Balanced partitioning result {[2,6],[2,6]} 56

4.1 An example of an RSoC with reconfigurable interconnect network and redundancy 60
4.2 An example of a repaired RSoC 61
4.3 Adjacency list representation of Figure (4.1). 68
4.4 Flow chart of the SNIF and SNCF 70
4.5 Yield of SNIF and LNIF at $\alpha, \beta = 0.05$ 75
4.6 Yield of SNIF and LNIF at $\alpha, \beta = 0.1$ 76
4.7 Yield of SNIF and LNIF at $\alpha, \beta = 0.25$ 76
4.8 Yield of SNIF and LNIF at $\alpha, \beta = 0.5$ 77
4.9 Yield of SNCF and LNCF at $\alpha, \beta = 0.05$ 77
4.10 Yield of SNCF and LNCF at $\alpha, \beta = 0.1$ 78
4.11 Yield of SNCF and LNCF at $\alpha, \beta = 0.25$ 78
4.12 Yield of SNCF and LNCF at $\alpha, \beta = 0.5$ 79

5.1 Investigated Field Reconfigurable Multiprocessor SoC Model. 84
5.2 (1,2,2) System Configuration (2 1-processor 2-memory PPUs). 86
5.3 Repair of faulty processor and memory throughout modular redundancy reconfiguration. 86
5.4 Reliability comparison of the PPU without modular redundancy and PPUs with different numbers of modular redundancy. 89
5.5 HW/SW Co-reliability of the Coarse-Grained Systems. 94
5.6 HW/SW Co-reliability of the Medium-Grained Systems. 94
5.7 HW/SW Co-reliability of the Fine-Grained Systems. 95

LIST OF TABLES

2.1 Simulation parameters for quality assurance 28

3.1 Simulation parameters for a one dimensional case 44
3.2 Simulation parameters for a two dimensional case 44

4.1 Adjacency matrix representation of Figure (4.1). 67
4.2 Y and $\overline{Y}$ of the given RSoC without repair 71
4.3 Performance comparison of the proposed repair strategies at each repair cycle where $\alpha, \beta = 0.05$ 72
4.4 Performance comparison of the proposed repair strategies at each repair cycle where $\alpha, \beta = 0.5$ 72
4.5 Repair performance of SNIF for the given parameters 73
4.6 Repair performance of SNCF for the given parameters 73

5.1 Common simulation parameters 92
5.2 3-tuple simulation parameters 93

CHAPTER 1
INTRODUCTION

System-on-Chip (SoC) is one of the most emerging Ultra-Large-Scale-Integration (ULSI) technologies in which numerous pre-designed and verified system components commonly known as *cores* can be seamlessly integrated into a single chip [8, 10, 13, 16, 20, 21, 22, 30, 31]. Since a complex application-specific system can be rapidly designed and fabricated on a single chip by dedicated Computer-Aided-Design (CAD) tools with vendor-supplied core designs, the SoC-based system requires less power, smaller space and shorter time-to-market [2]. Hand-held devices, wireless communication devices, multimedia devices, PDAs (Personal Data Assistants), computer-controlled vehicles, unmanned autonomous vehicles, MEMS (Micro-Electo-Mechanical Systems) and embedded systems are commonly referred applications for which the SoC is the most suitable technology choice [8].

Two requirements must be realized in order for a SoC to be a viable and practical technology. First, the SoC fabrication line must be capable of mass production at costs that are competitive with alternative methods such as printed circuit boards (PCBs) and multichip modules (MCMs) to mention a few. This implies that the *expected yield* of SoCs must be high enough to be cost-effective [20]. Second, the cores and supporting circuitry must be capable of performing their intended function throughout their intended useful life [22]. The second requirement is commonly referred to as *reliability*.

The yield of a SoC is mainly determined by its manufacturing defect density [20]. As the more complicated fabrication technique is applied, the higher defect density would be induced [28]. For example, memory cells in a DRAM (Dynamic Random Access Memory) core are more likely to experience defects than the other logic gates due to imperfect manufacturing processes and their complex three dimensional cell structure [40, 41, 42, 43, 45]. The reliability of a SoC is a function of time and fault arrival rate. In fact, cores cannot be physically repaired in the field, once they are packaged into a single chip. Therefore, a single unreliable core may devastate the intended functionality of the whole SoC.

As the need for SoC technology increases, high yield and solid reliability are becoming critical requirements for SoCs because insignificant degradation or defect of core components may result in seriously unacceptable low manufacturing yield. Furthermore, field reliability must be also assured because numerous unreplaceable/unrepairable cores are integrated onto a single chip to implement a complex monolithic system. In this context, core-based yield and reliability assurance techniques must be developed to achieve high yield and reliability for SoCs.

1.1 Reliability of Embedded Cores

Embedded cores are reusable modules to be embedded on a single chip to build an on-chip system [8, 10, 16, 20, 21, 22, 30, 31]. These cores are provided in the form of rich libraries of pre-designed and pre-verified building blocks. Cores are designed to embed hardware descriptions of today's standard ICs such as digital signal processor (DSP), microprocessor, or DRAM core.

Cores are categorized by *soft* (register-transfer level), *firm* (netlist level), and *hard* (technology-dependent layout) cores [26]. Soft cores are technology-independent for the system designer to be flexible in system-integration, which is best for implementation of functions with frequent customization needs [21]. Hard cores are designed to be optimized for the given technology constraints such as area and performance, while its weakness mainly stems from the lack of flexibility. Hard cores are usually used for timing and function-critical components of the system such as the CPU or analog elements [30]. Firm cores are designed in an intermediate form of soft and hard cores. Firm cores are appropriate for implementing functions without timing requirements that dictate custom layout, but requiring intellectual property encryption [30]. Each core category provides different advantages and disadvantages due to its unique features, and hence requires different modeling and testing approaches.

A major advantage of the SoC is its customizability and reconfigurability by using its reusable predesigned/preverified and ideally preoptimized core components and interconnects for integration and manufacturing [8]. This yields tremendous cost and performance enhancements. However, integrating and manufacturing a SoC with reusable cores requires new fault-tolerance methodologies because of the unique characteristics created by the SoC with wide variation in reliability requirements for the heterogeneous cores, test structures and defect and fault tolerance [26]. Thus, the quality of each core and interconnect structure of a SoC must be assured and configured properly to achieve the desired higher manufacturing yield and field reliability.

1.2 Reliability Assurance Techniques for Embedded-core-based SoC

SoCs using deep-sub-micron technology such as 100-nm technology, gigahertz clock frequency, and less than 1-V power supply by year 2003 to 2006 will cause serious on-chip noise due to increased cross-coupling of capacitances, inductances, and electromagnetic fields between bus interconnects [26]. New fault models, testing/diagnosis methods, and reliability assurance techniques should be developed for the realization of reliable SoCs [1, 16, 20, 26, 28].

Testing strategies and methodologies for embedded-core-based SoCs have been investigated in [26] in which the criticality of manufacturing test and design debugging in the embedded-core-based SoC has been addressed. In [5], a system tool was introduced to explore and compare different embedded-systems architectures. Another fast and low-cost testing technique for core-based system-chips was also proposed in [22].

In traditional PCB or MCM-based systems, component providers are responsible for chip design, manufacturing and testing, whereas system integrators are responsible for board-level design, integration and testing under the assumption of fault-free components. Thus, only the interconnects between components are to be tested [11].

Core testing is a joint responsibility of the core provider whereas the system integrator in embedded-core-based systems [10, 15, 16, 22, 26]. The core provider basically only transfers the description of a module and the system integrator manufactures a target system by testing the cores provided by the core providers and integrating them together through laying out and testing interconnects between the cores [8, 11, 10, 15, 16, 22, 26].

In core-level testing, a core is often dealt with as a black box to the system integrator if the core is hard or encrypted as an intellectual-property. The core provider can deliver to the system integrator the core test which generally consists of the design-for-test structures, built-in or out, and the corresponding test patterns [10, 12, 15, 16, 22, 26]. In [12], for example, an instruction-level design-for-test methodology for testing processor and IP cores in a SoC was proposed and validated. However, the core provider has little or no prior information about the manufacturing environment and the requirements of the target system, such as test methods, fault models, fault-coverage and quality levels, because different system integration and manufacturing processes have different requirements due to different defect densities and distributions [1, 7, 20, 26, 28].

The accessibility of the component terminals (i.e., the primary inputs and outputs of chips and cores) of an embedded-core-based SoC is different from traditional systems [5, 51, 16, 19]. With PCB-based systems, the chips are tested as stand-alone units, and during tests, they can be accessed through direct physical access to chip pins. In contrast, cores are often deeply embedded in a SoC, so direct physical access is not available [5, 51, 16, 19, 26]. The chip design must then provide an electronic test access infrastructure from the chip pins to the terminals of the embedded core [14, 19, 26]. Infrastructure for testing the hardware in between the cores and isolating a core from its surroundings should also be provided [19, 26].

A system-level test is an integration and coordination of on-chip test and diagnosis capabilities, which is far more complex than the traditional PCB-based assembly test. It consists of individual tests for cores, tests for user-defined logics, and tests for interconnect logics and wires [1]. The SoC's composite test requires adequate test scheduling to coordinate several chip-level requirements, such as test time, power dissipation and area overhead. It is also necessary to run intra-core and inter-core tests in a certain order to avoid affecting the initialization and final content of individual cores [16]. In [7], delay fault testing using symbolic path modeling was proposed to test interconnection delay faults in SoCs. Deep submicron chip testing issues should be also taken into account to meet sufficient defect/fault coverage requirements, overall test cost and time-to-market [28].

Even though preverified cores are integrated, validating and debugging the functionality and timing of the SoC could be another challenge due to unique features such as the heterogeneous structure of the SoC; the need for software and hardware co-simulation; the design for integrated functionality; timing validation and debugging; and the accessibility problems of deeply embedded cores [5, 51, 16, 19, 26]. High-level prototype-level validation is usually co-simulated with silicon-level validation that requires at-speed validation [26]. In [11], for example, high-level crosstalk defect simulation techniques were proposed for SoC interconnects. Unfortunately, current high-level validation techniques cannot catch up with the silicon-level validation speed of a SoC yet, which gives a new challenge for achieving an accurate validation of the quality of SoCs [26].

On-chip testing circuitry takes a much larger area than the main circuitry resulting in reduced manufacturing yield and increased cost and low area utilization. Also, test circuitry for each core often exhibits an irregular structure making it difficult to generate core-specific deterministic test patterns on-chip at acceptable area costs even though it provides better accuracy and defect coverage. In practice, not all test patterns can be generated on-chip in a cost-effective manner [6, 7, 8, 12, 51, 15, 26]. In the meantime, it is becoming increasingly more difficult for off-chip ATE (automated test equipment) to keep up with today's high-speed, high-density, and mixed technology SoCs to sufficiently test their quality-related defects [6, 51, 19]. Hence, a combined ATE and BIST (built-in-self-test) approach is recommended to provide a capability for testing all different types of defects and faults and to meet the above-mentioned cost and quality-effectiveness [1, 6, 14, 16, 19, 26, 28, 29].

Embedded cores in a SoC cannot be replaced once they are fabricated even though the reliability of the end-product SoC is mainly determined by the reliabilities of the cores. Thus, repair of the SoC for more graceful reliability degradation usually requires advanced fault-tolerance techniques such as the Built-In-Self-Test, Diagnosis and Repair (BIST/BISD/BISR) [35, 36, 37]. Self-testability has been commonly exploited for the systems with limited ATE testability [6, 14, 15, 29, 37] because a SoC has severe limitations on ATE accessibility. Test results gathered by either ATE or BIST or both then need to be diagnosed to suggest the most reliability enhancing repair configuration. External diagnosis softwares are usually employed to analyze different possible repair options. Once a SoC is packaged, self-diagnosability is desirable to provide field diagnosis of the BIST data. Finally, redundancy is to be switched in and configured as instructed by the diagnosis softwares or BISD to repair the SoC. Redundancy configuration has been done by external equipment such as a laser fuse blower, while self-repairability is also needed to enhance the reliability of the packaged SoC in field.

The self-testability/diagnosability/repairability is not free from penalty such as redundancy and test/diagnosis/repair circuitry overhead since it consumes valuable chip space, thereby increases manufacturing cost significantly [6, 7, 8, 12, 51, 15, 26]. Thus, minimization of the overhead due to the self-repairability as well as optimized utilization of the redundancy, while achieving maximum reliability, is desirable to maintain the cost-effectiveness of the SoC technology.

1.3 Reliability-driven Defect and Fault Tolerance in SoC

A SoC can be modeled as a system which consists of a number of different cores and fully customized interconnect structures. As the more complex design is implemented in a SoC, its reliability assurance becomes a more critical issue since cost-effectiveness of the design and implementation is mainly affected by the reliability of embedded cores and interconnect structures [11, 20, 26]. The lifetime of a SoC can be categorized into three stages with respect to its reliability: *design stage*, *fabrication stage* and *field stage*. In the design stage, parametric reliability estimations are usually done, and core-level or chip-level defect and fault tolerance techniques are introduced on the fly. In the fabrication stage, ATE performs a wafer probe to cope with the manufacturing defects on the SoC, and embedded redundancy, if any, can be utilized to repair the defects due to imperfect manufacturing. Thus, the yield (i.e., the reliability of a SoC at the fabrication stage) can be modeled by testability and repairability. In the field stage, BIST/BISD/BISR are exploited to further enhance the reliability of the SoC.

The main objectives of this work is to propose reliability assurance and reliability-driven defect and fault tolerance in SoC system, thereby realizing more dependable and cost-effective mass production of SoCs. The following specific problems will be addressed and resolved in this work.

- *Reliability Assurance Techniques* : Core-based assurance techniques are to be proposed to model and evaluate the reliability.

- *Reliability-Driven Defect & Fault Tolerance* : Defect and fault-tolerance techniques are to be developed driven by the proposed modeling and evaluation techniques.

- *Redundancy Architecture and Reconfiguration Algorithms* : Ad-hoc design of core-specific redundancy architectures and their reconfiguration algorithms are to be proposed.

- *Reliability Enhancement Strategies in Configurable SoC* : Reliability improving configuration strategies in configurable SoCs are to be investigated.

- *Connectivity-Based Repair Scheduling Algorithms of reconfigurable SoCs* : Repair algorithms selecting more reliable repair schedules are to be proposed for reconfigurable SoCs.

- *HW/SW Co-Reliability in SoC* : Reliability characteristics of hardware configuration, affected by application design software ,are to be investigated.

The organization of the dissertation is as follows. In Chapter 2, a reliable embedded memory core and its yield and reliability assurance techniques are proposed. Hierarchical redundancy, ATE-based fabrication-time repairability and BIST/BISD/BISR-based field repairability are employed to achieve acceptable manufacturing yield and field reliability. Then in Chapter 3, a novel optimal redundancy partitioning technique will be proposed as well. In the following Chapter 4, connectivity-based reconfiguration algorithms for SoCs are proposed and validated by the reliability assurance techniques. It proposes how proper repair scheduling can guarantee the optimal reliability of a SoC. Then, HW/SW Co-reliability assurance and balancing techniques are proposed in Chapter 5. Reliability issues associated with the hardware configurations set up by the dynamic software demands are investigated. Finally, the conclusion is presented in the last chapter.

CHAPTER 2

EMBEDDED MEMORY CORE ARCHITECTURE FOR ENHANCED MANUFACTURING YIELD AND FIELD RELIABILITY

Among the cores for SoC integration, one of the most vulnerable cores is the embedded memory core since memory cells are commonly considered as most prone to defects and faults [35, 36, 37, 39, 41, 42, 43, 45]. As the SoC fabrication process goes toward the era of deep-sub-micron technology such as $0.13\mu m$, the need for high yield and ultra reliable embedded memory cores has become more urgent than ever before. According to the Semiconductor Industry Association and ITRS (International Technology Roadmap for Semiconductor) 2000, embedded memory will continue to dominate SoC content in the next several years, approaching 94% of the die area by 2014 as shown in Figure (2.1) [38]. The issues surrounding high-density multi-megabit memory dependability (i.e., the probability to be manufactured as good AND not failing during the time interval [0,t]) must be resolved in order to facilitate this trend and to produce a cost effective SoC product.

Traditionally, reconfiguration (repair) of memory arrays using spare memory lines is the most common technique for yield enhancement of memories with faults [35, 36, 37, 39, 41, 42, 43, 45]. In [37], a simple built-in-self-analysis-repair scheme for embedded DRAM was proposed and a row-column self-repair scheme for embedded SRAM for Alpha 21264 was shown in [39]. A shared built-in self-repair analysis scheme for multiple embedded memory cores in the SoC is proposed in [35] to realize minimum area penalty independent of the number of embedded memory cores. Software-based self-testing methodology for processor cores was reported in [15]. Virtual-socket archi-

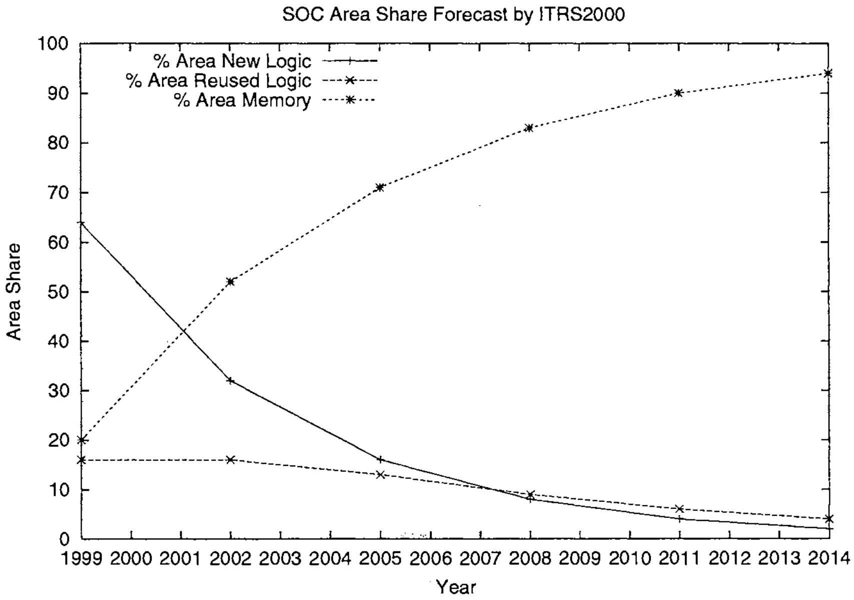

Figure 2.1: SoC area share forecast of embedded memory cores from 1999 to 2014 by ITRS2000

tecture for embedded DRAM cores was proposed in [17] in which embedded DRAM cores with different sizes can be seamlessly integrated into a SoC by utilizing the core wrapper called virtual socket. Application specific synthesis of SoCs with flexible memory size was proposed in [27]. A programmable BIST core for embedded DRAM was introduced in [29]. [32] also reported a design methodology for embedded DRAM in multimedia SoCs. Error-correcting-code (ECC) has been also used to develop more reliable memory devices in terms of data integrity and dependability [43, 44]. Especially a synergistic fault-tolerance of the combination of ECC and row/column redundancy was discovered in [43]. An on-chip march pattern generator for testing embedded memory cores was proposed in [6] as well. However, a comprehensive research work on fault-tolerant embedded memory core organization and its quality assurance has not been reported yet.

The main motivation of this chapter is to propose a high yield, ultra reliable embedded memory core organization for SoC applications which is capable of both factory and field repair to enhance its factory yield and field reliability at the same time, by establishing a series of accurate quality assurance techniques for the proposed memory core architecture.

The organization of this chapter is as follows: In the following section (Section 2.1), a fault-tolerant memory core organization which is capable of both factory and field repair will be proposed. Then, in section 2.2, the detailed factory and field repair procedure will be shown. Yield and reliability assurance techniques for the proposed embedded memory core will be proposed in chapters 2.3 and 2.4, respectively. Parametric simulations and results will be shown in section 2.5. Finally, discussion and conclusions will be given in Section 2.6.

2.1 Fault-Tolerant Memory Core Organization for SoC Applications

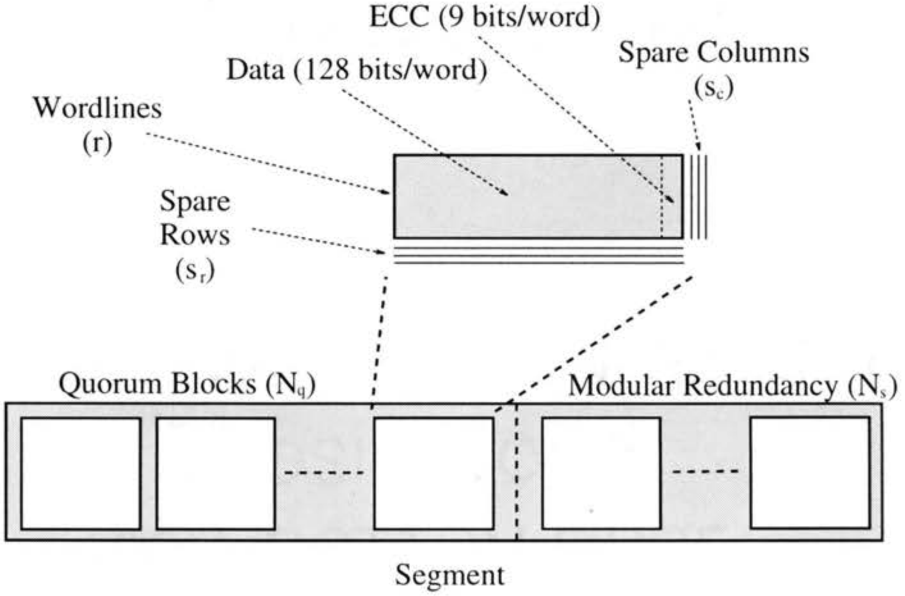

Figure 2.2: Architecture of the memory block and segment

The atomic unit of the proposed memory core architecture is referred to as *block*. To enhance not only data integrity but also manufacturing yield and field reliability, both the double-error-detection, single-error-correction (DED-SEC) error-correcting-code (ECC) and the spare row/column redundancy are built into each block of the memory. 128 bit data word with 9 bit ECC has been chosen while the other configuration of ECC is technically feasible as well. Numerous error correcting codes were categorized in [44]. Each memory block has r ECC words, S_c spare columns, and S_r spare rows for the sake of fault-and-defect-tolerance of the core as shown in Figure (2.2). Almost every numerical parameter of the proposed memory core is left as a variable to facilitate the maximum flexibility in the resultant core, thereby allowing cost-effective and target-specific memory core design optimization. The next unit of the proposed memory core organization is the *segment*. Segment consists of N_q quorum ECC memory blocks with row/column redundancy. The SoC designer can assign N_s spare memory blocks to overcome costly block-wise failures. Figure (2.2) shows an illustration of a segment with N_q quorum memory blocks with N_s spare memory blocks. This kind of modular redundancy technique will enhance core dependability significantly. The detailed reconfiguration algorithm and repair procedure for the proposed memory core will be covered in the next section. The next memory unit is the *segmented memory array* which consists of N_{seg} segments. Therefore, the total size of the memory core is $128 \times r \times N_q \times N_{seg}$ bits. The proposed memory core consists of the following components as shown in Figure (2.3).

- BIST/BISD/BISR Processor : This system component governs self-test, self-diagnosis, and self-repair procedures.

- Laser Fuse : A set of laser reconfigurable fuses to permanently program the given redundancy resources.

- EEPROM Fuse : A programmable, non-volatile memory to store additional reconfiguration signature generated by the BIST/BISD/BISR processor in field.
- IEEE JTAG (Joint Test Action Group) 1149.1 : External boundary scan test equipment interface.
- Memory Array Interface : This component connects the segmented memory array and the BIST/BISD/BISR Processor together. Data, address, control and repair data flow via this component.

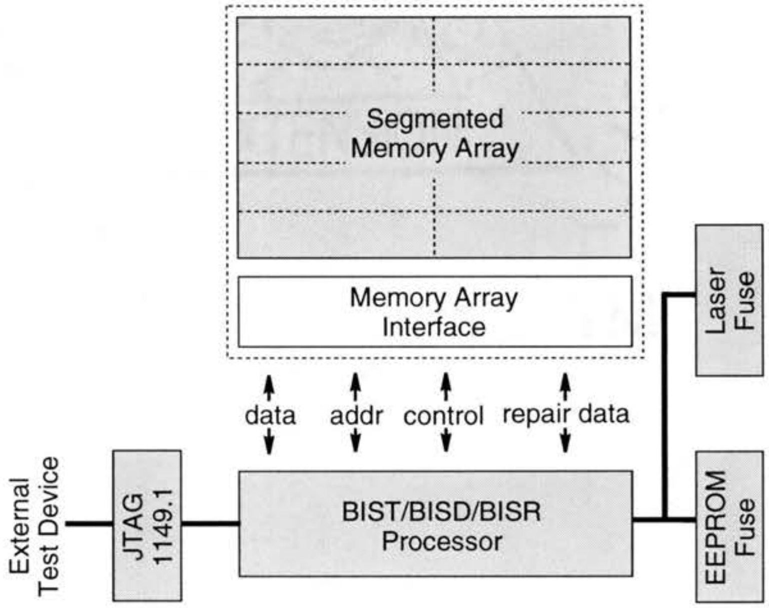

Figure 2.3: Logical view of the proposed fault-tolerant memory core components

2.2 Repair Algorithm and Procedure

Memory reconfiguration by using row/column redundancy is one of the well-known NP-complete problems in VLSI (Very Large Scale Integration) design [41, 42, 44, 45].

Thus, numerous approximation algorithms guaranteeing near-optimal solutions have been introduced with a quite unrealistic assumption, i.e., high faulty cell probability, while world-class semiconductor manufacturers produce quality memories which contain usually one or less faulty cell per million cells [43]. Also, most of the memory reconfiguration algorithms are not suitable to be embedded in the SoC in reality due to their high computational complexity. Thus, a very simple yet effective hierarchical repair algorithm is used for the proposed embedded memory core, while various reconfiguration algorithms can be applied as well. Quality assurance techniques for the proposed embedded memory core will be realized based on the following requirements:

1. *ECC* : single-error-correction and double-error-detection (SEC-DED) error correcting code covers one-bit permanent or transient faults and detects 2-bit permanent or transient faults.

2. *Row Redundancy* : If more than 2 failing bits in a row or the support circuitry of the row are diagnosed as faulty, a row spare replaces the faulty memory row.

3. *Column Redundancy* : Likewise, if more than 2 failing bits in a column or the support circuit of the column are diagnosed as faulty, a column spare replaces the faulty memory column.

4. *Block-wise Modular Redundancy* : Within a segment, a failing block also can be repaired by a spare block. This kind of repair method is commonly referred to as *modular redundancy technique.*

The proposed fault-tolerant memory core provides not only factory repairability but also field self-test and repair capabilities. The first repair process takes place during the wafer probe in the factory. The repair signature created by the proposed memory core is sent to an external test equipment via the IEEE JTAG 1149.1, which is a common connectivity interface between the external tester and the embedded BIST/BISD/BISR processor [19, 35, 37, 39]. Then, conventional laser repair equipment blows the fuse box with the required reconfiguration information. Afterwards, the core contains the permanent memory repair signature unique to this particular SoC. The fabrication-time repair also has the advantage of performing test and repair at the wafer level. The wafer is subject to a variety of stringent conditions during testing which helps insure high memory and SoC dependability (i.e., extended voltage, temperature and frequency conditions).

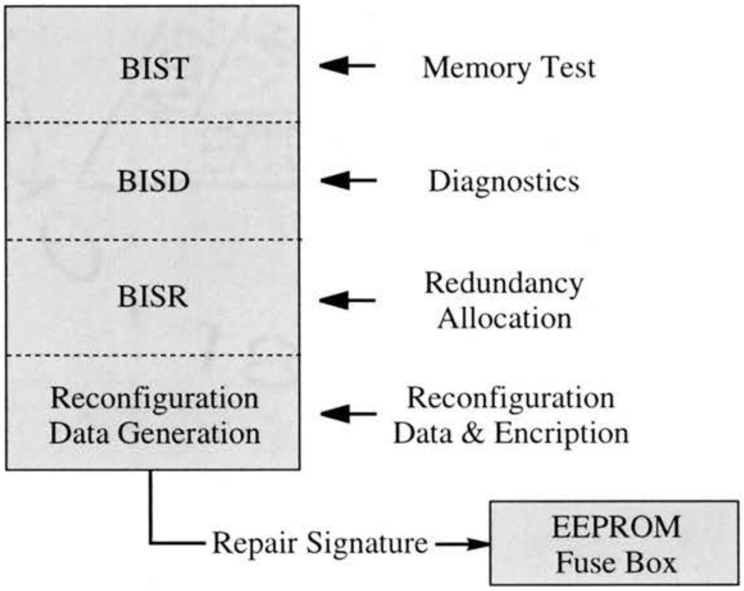

Figure 2.4: Logical flow of the BIST/BISD/BISR procedure

The field repair operation tests and repairs memory instances each time the end product starts up or is reset. Logical view of the field repair procedure is given in Figure(2.4). The BIST/BISD/BISR Processor initiates and operates the test program, determines defective memory locations (if any), allocates redundancy resources, and produces a new repair signature that resides in the EEPROM fuse on the memory core (similar to the EEPROM-controlled redundancy allocation technique proposed in [36]). This non-volatile repair signature is applied to the redundancy control logic and remains there until a new repair signature is overwritten. As a result, it is possible to design an SoC that incorporates the proposed memory core with both fabrication-time and field repair in which the fabrication-time repair process repairs manufacturing defects, and the field repair operation covers any subsequent problems that may materialize over the life of the end SoC product.

2.3 Fabrication-Time Yield Analysis of the Proposed Fault-Tolerant Memory Core

The following parameters are used for the analysis:

- d : Number of data bits per ECC word.
- c : Number of code bits per ECC word.
- n_c : Number of columns per block without spare columns. ($n_c = d + c$)
- n_r : Number of rows per block without spare rows.
- s_r (s_c) : Number of spare rows (columns) per block.
- N_q : Number of quorum blocks per segment.
- N_s : Number of modular spare blocks per segment.
- N_{seg} : Number of segments per memory system.
- λ_m : Expected number of manufacturing failures per memory cell.

The following assumptions are also made to analyze the manufacturing yield of the proposed fault-tolerant memory core:

- A failing bit in a word is supposed to be corrected by Single-Error-Correction & Double-Error-Detection (SEC-DED) Error-Correcting-Code (ECC).
- Multiple faults in a word can be repaired by row redundancy.
- Multiple faults in a column can be repaired by column redundancy.
- Block-wise failure can be repaired by the block modular redundancy.

The yield of a single cell can be written as

$$Y_{cell} = e^{-\lambda_m} \tag{2.1}$$

where λ_m is the expected number of faults per memory cell due to imperfect fabrication. The probability of not having a failing cell in an n_c bit error-correcting-code word is then given by $(Y_{cell})^{n_c}$ and the probability of having one of the n_c bits failing can be represented by $n_c \cdot (Y_{cell})^{(n_c-1)} \cdot (1 - Y_{cell})$. Therefore, the yield of each word without ECC is

$$Y_{word} = (Y_{cell})^{n_c} \quad \text{if ECC is not used.} \tag{2.2}$$

Since each ECC word can tolerate one failing bit, the yield of each word enhanced by ECC is given by

$$Y_{word} = (Y_{cell})^{n_c} + n_c \cdot (Y_{cell})^{(n_c-1)} \cdot (1 - Y_{cell}) \quad \text{if ECC is used.} \tag{2.3}$$

There are n_r words in each block of the proposed memory core. If we start with a fault-free block, the chances that the first failing cell will not occur coincident with another failing cell are absolute certainty. The probability for the second faulty cell not to occur in the same word as the first faulty cell is $(n_r - 1)/r$. For the third one, the probability is $(n_r - 2)/n_r$, etc,. Thus, for the N single-cell faults in a block the probability of non-alignment can be written as [43]

$$P(no\ alignment|N) = \prod_{i=1}^{N} \frac{(n_r + 1) - i}{n_r} \tag{2.4}$$

which can be rewritten in the form

$$P(no\ alignment|N) = \frac{n_r!}{n_r^N (n_r - N)!} \tag{2.5}$$

For example, as shown in [43], the calculations made with either one of the formulas given above show that with 100 randomly failing cells in a 16Mbit memory chip, there is a better than 97% chance that two such failing cells will not occur in the same error-correcting-code word of 137 (i.e, 128 data bits + 9 ECC bits) bits. Thus, the yield associated with all these defects would be more than 97%. Nevertheless, when there are 1,000 randomly failing cells on the memory, the probability for non-alignment of cells in any error-correcting-code word is less than 2%. This represents a 2% yield for such faults.

The ECC circuitry covers single-bit faults perfectly. However, its limitation mainly occurs when multi-bit faults hit on a single word. In addition, some of these faults affect the chip support circuitry, and the rows and columns. Because of this, the proposed fault-tolerant memory core also has redundant rows and columns to replace the defective ones entirely.

The probability of having two or more single-cell failures in a word is given by

$$P_{mfw} = \sum_{i=2}^{n_c} \binom{n_c}{i} (Y_{cell})^{(n_c - i)} \cdot (1 - Y_{cell})^i \tag{2.6}$$

where the subscript mfw is used to indicate that this probability is associated with the event of multiple faults in a word. This equation can be further simplified by Equation (2.3) and the binomial theorem as

$$P_{mfw} = 1 - Y_{word} \tag{2.7}$$

if ECC is used to cover one faulty cell in a word. Then, the probability of finding from one to s_r words with multiple faults can be written as

$$P_{s_r} = \sum_{j=1}^{s_r} \binom{n_r}{j} (P_{mfw})^j \cdot (1 - P_{mfw})^{(n_r - j)} \tag{2.8}$$

where the subscript s_r is used to indicate that these are the probabilities for the words that need to be replaced, if there are redundant rows available. The probability to fix those words with multiple faulty bits solely depends on the availability of fault-free spare rows. So, the probability of having j or more fault-free spare rows is given by

$$P_{s_r \geq j} = \sum_{k=j}^{s_r} \binom{s_r}{k} (Y_{word})^k \cdot (1 - Y_{word})^{(s_r - k)} \tag{2.9}$$

where the subscript $s_r \geq j$ is self explanatory.

The probability of replacing the words with multiple faulty cells in a block is obtained by combining Equation (2.8) and (2.9) as follows

$$Y_{s_r} = \sum_{j=1}^{s_r} \sum_{k=j}^{s_r} \binom{n_r}{j} \binom{s_r}{k} (P_{mfw})^j \cdot (1 - P_{mfw})^{(n_r - j)} \cdot (Y_{word})^k \cdot (1 - Y_{word})^{(s_r - k)} \tag{2.10}$$

where Y_{s_r} is used to indicate that it is the yield of repaired block by spare rows. Taking the faults corrected with the SEC ECC circuitry into account, the yield of a block is improved from $(Y_{word})^{n_r}$ to

$$Y_{block:s_r} = (Y_{word})^{n_r} + Y_{s_r} \tag{2.11}$$

The yield of a block can be further enhanced with the spare columns. It is assumed that the spare columns are used to repair the block which cannot be repaired by the SEC ECC and spare rows. Using Equation (2.11), the probability of having a faulty column can be written as

$$P_{fc} = 1 - \log_{n_c} Y_{block} \tag{2.12}$$

Following the same manner which has been used to evaluate the effect of the spare rows, with s_c redundant columns given in a block, the probability of finding from one to s_c faulty columns in a block is given by

$$P_{s_c} = \sum_{j=1}^{s_c} \binom{n_c}{j} (P_{fc})^j \cdot (1 - P_{fc})^{(n_c - j)} \tag{2.13}$$

The subscript s_c is also used to indicate that these are the probabilities for the columns that need to be replaced, if there are redundant columns available. Then, the probability of having j or more fault-free spare columns can be also given by

$$P_{s_c \geq j} = \sum_{k=j}^{s_c} \binom{s_c}{k} (Y_{cell})^{(n_r \times k)} \cdot (1 - (Y_{cell})^{n_r})^{(s_r - k)} \tag{2.14}$$

Then, the probability of replacing the faulty columns in a block can be obtained by combining Equation (2.13) and (2.14) as follows

$$Y_{s_c} = \sum_{j=1}^{s_c}\sum_{k=j}^{s_c}\binom{n_c}{j}\binom{s_c}{k}(P_{fc})^j \cdot (1-P_{fc})^{(n_c-j)} \cdot (Y_{cell})^{(n_r \times k)} \cdot (1-(Y_{cell})^{n_r})^{(s_r-k)} \quad (2.15)$$

where Y_{s_c} is used to indicate that it is the yield of repaired block by spare columns. Taking the faults corrected with the SEC ECC circuitry and spare rows into account, the yield of a block is improved from $(Y_{word})^{n_r}$ to

$$Y_{block} = Y_{block:s_r} + Y_{s_c} \quad (2.16)$$

Each segment of the proposed fault-tolerant embedded memory core consists of $N_q + N_s$ blocks, where N_q is the number of quorum blocks that must be operational to guarantee the functionality of the segment and N_s is the number of spare blocks. By exploitation of this N_q-out-of-$(N_q + N_s)$ modular redundancy technique, the yield of each segment can be enhanced from $(Y_{block})^{N_q}$ to

$$Y_{seg} = \sum_{i=0}^{N_s}\binom{N_q+N_s}{i}(Y_{block})^{N_q+N_s-i} \cdot (1-Y_{block})^i \quad (2.17)$$

Finally, the overall yield of the proposed fault-tolerant memory system can be expressed as

$$Y = (Y_{seg})^{N_{seg}} \quad (2.18)$$

2.4 Field Reliability Assurance of the Proposed Fault-tolerant Memory Core

As described in Section 2.2, the proposed fault-tolerant memory core is able to perform Built-In-Self-Test (BIST), Built-In-Self-Diagnosis (BISD), and Built-in-Self-Repair (BISR) in the field. These special capabilities of the proposed memory core architecture further enhance its reliability during field operation. Thus, rapid reliability degradation of the host SoC product can be prevented. In this section, reliability assurance of the proposed memory core architecture will be presented based on the following assumptions:

- As the yield of the laser-repaired core at the factory wafer probe stage has been analyzed based on the expected number of failures due to manufacturing defects, the failure arrival rate per unit time will be used to estimate the reliability of the proposed fault-tolerant memory core used in the SoC in the field. To differentiate the expected number of failures due to the manufacturing defects and the failure arrival rate per unit time during field operation, λ_m and λ_f are used in this chapter, respectively.

- Embedded BIST/BISD/BISR processor tests, analyzes and repairs the faulty memory core in the field when it is reset or restarted. For simplicity, the reset/restart event is assumed to occur once in every unit interval time (i.e., Δt).

The following parameters in addition to the ones introduced in Section 2.3 are used for the reliability assurance analysis of the proposed fault-tolerant memory core:

- t : time.

- Δt : Unit time interval (e.g, day, week or month, etc).

- λ_f : Field failure arrival rate of memory cell per unit time interval.

- $R(t)$: Reliability of the core at time t.

- $D(t)$: Overall dependability of the core at time t (i.e., the probability to be manufactured as good AND not failing during the time interval $[t_0, t]$).

The *reliability* of a system is a function of time which is defined by the conditional probability that the system performs correctly throughout the interval of time $[t_0, t]$, given that the system was performing flawlessly at the initial time t_0 [40]. If we assume that the failure arrival rate of the system has a constant value of λ, the reliability of the system can be expressed as follows

$$R(t) = e^{-\lambda t} \tag{2.19}$$

The exponential relationship between the reliability and time is known as the *exponential failure law* which states that for a constant failure arrival rate function, the reliability varies exponentially as a function of time. The exponential failure law is quite valuable to analyze electronic components and is a very well-known and widely-used relationship between the reliability and time.

Since the yield can be viewed as the reliability of the system at the initial time t_0, the formulas for the yield assurance given in the previous section can be extended to derive the reliability formulas with some appropriate modifications in this section. At the initial time t_0, the probability that a single cell is fault-free is $Y_{cell} = e^{-\lambda_m}$. Therefore, the reliability of a single cell (i.e., the conditional probability that a single cell is fault-free from t_0 to t) can be written as

$$R_{cell}(t) = Y_{cell} \cdot e^{-\lambda_f t} \tag{2.20}$$

where Y_{cell} is the yield of a single cell and λ_f is the failure arrival rate of a single memory cell in the field.

Since the BIST/BISD/BIST shares the same redundancy structure and repair mechanism with the fabrication-time repair, the yield assurance formulas proposed in the previous section can be used if $R_{cell}(t)$ replaces Y_{cell}. Thus, the field reliability assurance formulas will be listed without detailed explanations.

$$R_{word}(t) = R_{cell}(t)^{n_c} \quad \text{if ECC is not used.} \tag{2.21}$$

$$R_{word}(t) = R_{cell}(t)^{n_c} + 137\, R_{cell}(t)^{(n_c-1)}(1 - R_{cell}(t)) \quad \text{if ECC is used.} \tag{2.22}$$

$$P_{mfw}(t) = \sum_{i=2}^{n_c} \binom{n_c}{i} (R_{cell}(t))^{(n_c-i)} \cdot (1 - R_{cell}(t))^i \tag{2.23}$$

$$P_{mfw}(t) = 1 - R_{word}(t) \tag{2.24}$$

$$P_{s_r}(t) = \sum_{j=1}^{s_r} \binom{n_r}{j} (P_{mfw}(t))^j \cdot (1 - P_{mfw}(t))^{(n_r-j)} \tag{2.25}$$

$$P_{s_r \geq j}(t) = \sum_{k=j}^{s_r} \binom{s_r}{k} (R_{word}(t))^k \cdot (1 - R_{word}(t))^{(s_r-k)} \tag{2.26}$$

$$R_{s_r}(t) = \sum_{j=1}^{s_r} \sum_{k=j}^{s_r} \binom{n_r}{j} \binom{s_r}{k} (P_{mfw}(t))^j \cdot (1 - P_{mfw}(t))^{(n_r-j)} \cdot (R_{word}(t))^k \cdot (1 - R_{word}(t))^{(s_r-k)} \tag{2.27}$$

$$R_{block:s_r}(t) = R_{word}(t)^{n_r} + R_{s_r}(t) \tag{2.28}$$

$$P_{fc}(t) = 1 - \frac{\log(R_{block}(t))}{\log(n_c)} \tag{2.29}$$

$$P_{s_c}(t) = \sum_{j=1}^{s_c} \binom{n_c}{j} (P_{fc}(t))^j \cdot (1 - P_{fc}(t))^{(n_c - j)} \tag{2.30}$$

$$P_{s_c \geq j}(t) = \sum_{k=j}^{s_c} \binom{s_c}{k} (R_{cell}(t))^{(n_r \times k)} \cdot (1 - (R_{cell}(t))^{n_r})^{(s_r - k)} \tag{2.31}$$

$$R_{s_c}(t) = \sum_{j=1}^{s_c} \sum_{k=j}^{s_c} \binom{n_c}{j} \binom{s_c}{k} (P_{fc}(t))^j \cdot (1 - P_{fc}(t))^{(n_c - j)} \cdot (R_{cell}(t))^{(n_r \times k)} \cdot (1 - (R_{cell}(t))^{n_r})^{(s_r - k)} \tag{2.32}$$

$$R_{block}(t) = R_{block:s_r}(t) + R_{s_c}(t) \tag{2.33}$$

$$R_{seg}(t) = \sum_{i=0}^{N_s} \binom{N_q + N_s}{i} (R_{block}(t))^{N_q + N_s - i} \cdot (1 - R_{block}(t))^i \tag{2.34}$$

$$R(t) = (R_{seg}(t))^{N_{seg}} \tag{2.35}$$

The overall dependability (i.e., $Pr\{$manufactured as good AND not failing during the time interval $[0, t]\}$) can be written as follows:

$$D(t) = Y \cdot R(t) \tag{2.36}$$

In addition to the reliability, the mean time to failure ($MTTF$) is another useful measurement to specify the quality of a system since the $MTTF$ is the expected time that a system will operate before the first failure occurs. Therefore, it can be used to measure the *system operation life without failure*. The $MTTF$ is defined in terms of the reliability function as

$$MTTF = \int_0^\infty R(t)dt \tag{2.37}$$

which is valid for any reliability function that satisfies $R(\infty) = 0$. Thus, the reliability function given in Equation (2.35) can be directly applied to Equation (2.37) to obtain the mean-time-to-failure (i.e., expected system operation life without failure) of the proposed fault-tolerant memory core.

2.5 Quality Assurance of the Proposed Fault-Tolerant Memory Core

In this section, the quality of the proposed fault-tolerant memory core in terms of the manufacturing yield, field reliability and mean-time-to-failure is assured by the proposed estimation techniques via a series of parametric simulations. The parameters are summarized as shown in Table 2.1. Although the proposed memory core architecture is flexible enough to implement various embedded memory core configurations, a sample 1Mbit memory core has been used for the parametric simulations. The sample 1Mbit memory core consists of 8 segments and each segment has 8 blocks of 128 × 128 memory array without considering ECC and redundancy. Therefore, each block is a 16Kbit memory array and each segment is 16K × 8 = 128K and the core is 128K × 8 = 1M. The following observations and discussions can be made from the simulations:

- *Manufacturing Yield Enhancement Ability of the Proposed Memory Core* : In Figure (2.5), the manufacturing yield of the block, the segment and the core are initially given (i.e., Figure(2.5-a,b,d)) as a function of S_c and S_r. The ECC-only core configuration (i.e., S_c and S_r =0 and N_s = 0) shows almost zero manufacturing yield while the allocation of 5 spare rows and columns guarantees an almost perfect yielding core. In addition to the spare row/column redundancy,

Table 2.1: Simulation parameters for quality assurance

Parameter	Meaning	Value
λ_m	Expected manufacturing failures per memory cell	$1 \cdot 10^{-3}$
S_r, S_c	# of spare row/column	Variable
N_q	# of quorum block	8
N_s	# of spare block	Variable
N_{seg}	# of segment	8
λ_f	Field failure arrival rate for memory cell	$5 \cdot 10^{-6}$
Δt	Unit time interval	One week

the modular block redundancy also enhances the manufacturing yield significantly as shown in Figure (2.5-e). The combination of 4 spare rows and columns and a spare block per segment amazingly covers almost every manufacturing defect and achieves near 100% core yield. The results given in Figure (2.5) assure the manufacturing yield enhancement ability of the proposed fault-tolerant memory core architecture. Intelligent exploitation of the proposed quality assurance techniques can be also used to optimize the core design in terms of the desirable manufacturing yield while minimizing cost due to redundancy overhead.

- *Field Reliability Enhancement Ability of the Proposed Memory Core* : In Figures (2.6)-(2.12), extensive field reliability simulation results are shown. Numerous combinations of S_c (S_r) from 0 to 8, and N_s from 0 to 2 are used to identify the robust field reliability enhancement ability of the proposed memory block, segment and core. As given in Figure (2.10-a), the ECC-only configuration of the sample core configuration hardly retains its reliability for the first 100 weeks while the S_c, $S_r = 8$ and $N_s = 2$ core configuration (e.g., Figure (2.12-i)) successfully maintains its reliability more than 95% for 1200 weeks. The common reliability requirement for long life applications is $\geq 95\%$ for a 10 year mission period. Thus, the proposed embedded memory core even qualifies for the strict field reliability requirement.

- *Prolonged MTTF* : Mean-time-to-failure (MTTF) is a commonly used measurement of the average system life time. The redundancy configuration of S_c, $S_r = 8$ and $N_s = 1$ has $MTTF \approx 1200$ weeks while the ECC-only configuration has $MTTF \approx 0$ weeks.

2.6 Summary

A high yield, ultra reliable embedded memory core architecture for SoC applications and its quality assurance techniques have been discussed in this chapter. The proposed fault-tolerant memory core organization takes advantage of the double-error-detection and single-error-correction (DED-SEC) error-correcting-code (ECC) to circumvent the single-bit faults while the row/column redundancy repairs more than two simultaneous faulty cells in a memory line or in wordline (bitline) failures. Synergistic fault-tolerance, first discovered in [43], has been also achieved in the proposed memory core design. The modular redundancy technique has been also introduced to overcome the memory-block-wise failures. The proposed memory core provides two distinctive repair mechanisms: the permanent laser redundancy reconfiguration during the wafer probe stage in the factory to enhance its manufacturing yield, and the dynamic BIST/BISD/BISR-based field reconfiguration of the redundant resources in the field to maintain high field reliability of the host SoC. The quality assurance techniques based on the combinatorial modeling scheme, which is suitable for the proposed memory core design, further verifies its abilities of enhancing manufacturing yield, excellent field reliability, improved tolerance against increasing failure arrival rates, and prolonged MTTF.

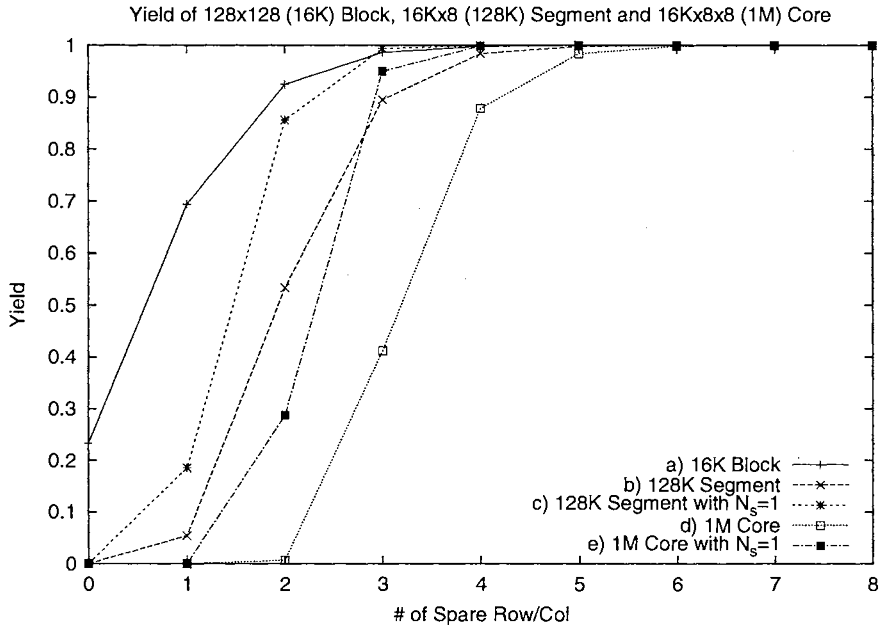

Figure 2.5: Yield analysis results of the sample 1Mbit core

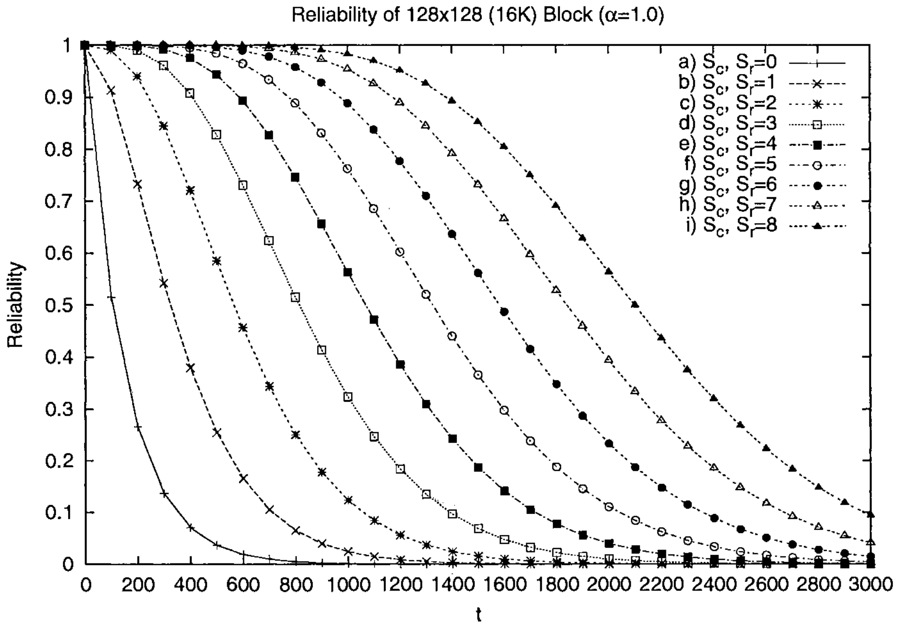

Figure 2.6: Reliability analysis results of the sample 16K block

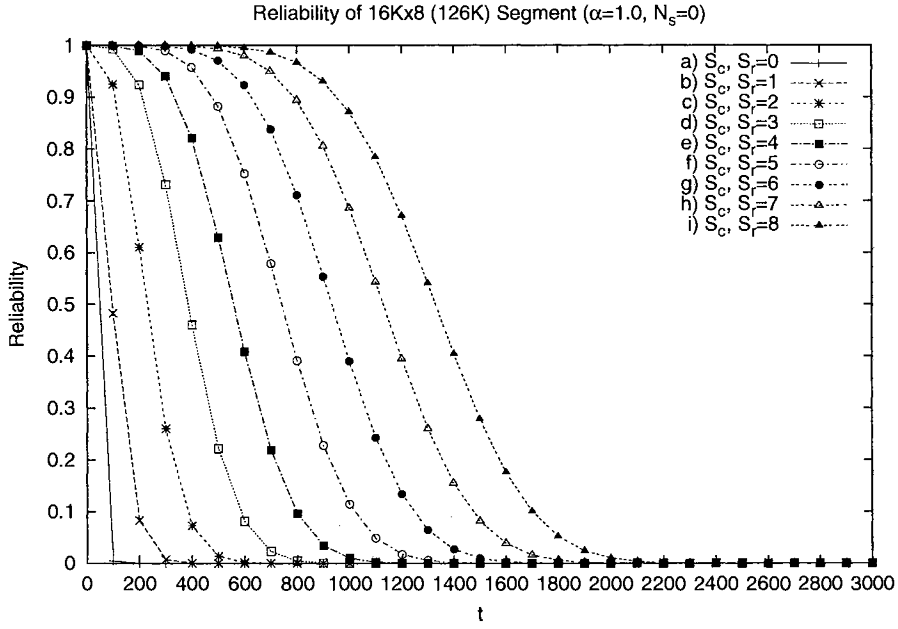

Figure 2.7: Reliability analysis results of the sample 16K x 8 (128K) Segment ($N_s = 0$)

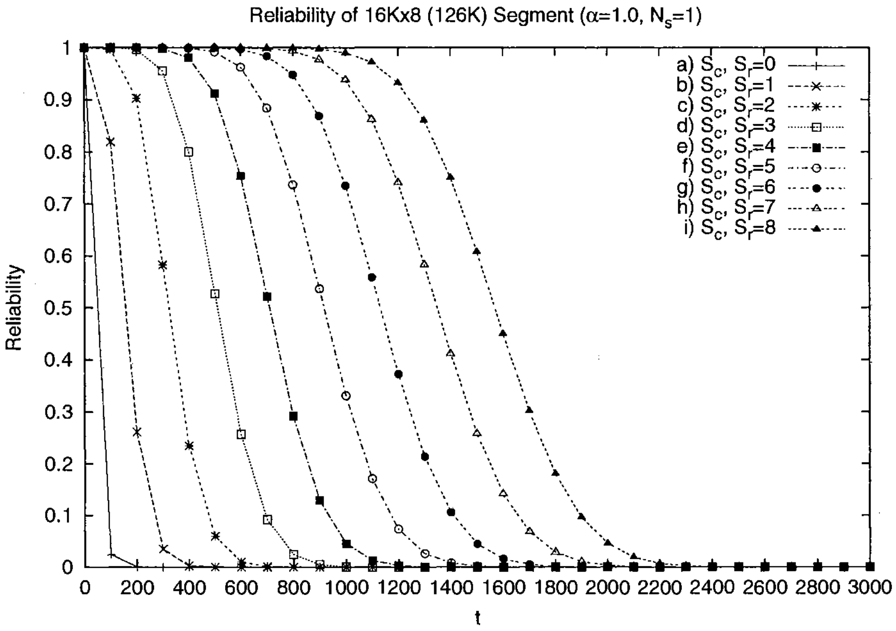

Figure 2.8: Reliability analysis results of the sample 16K x 8 (128K) Segment ($N_s = 1$)

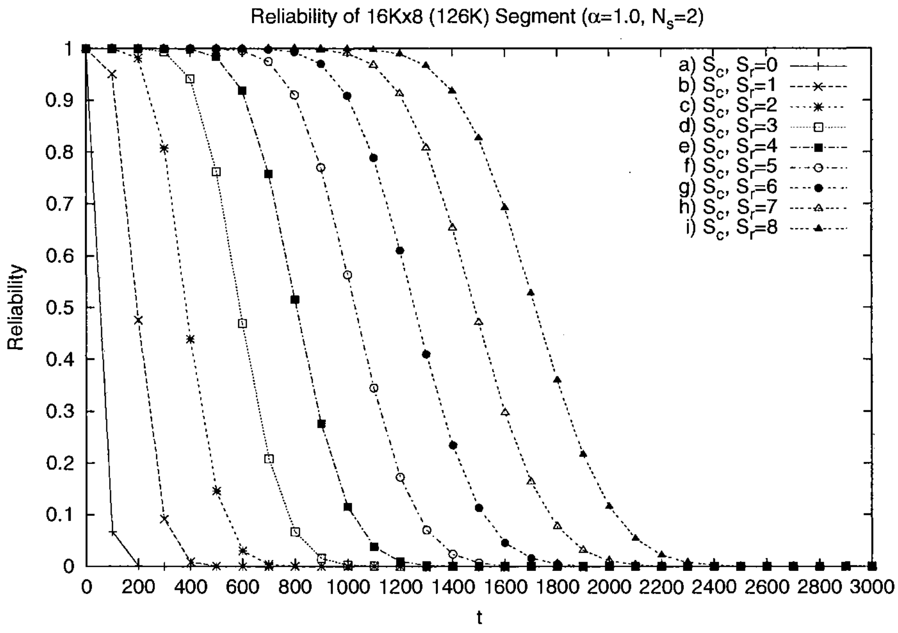

Figure 2.9: Reliability analysis results of the sample 16K x 8 (128K) Segment ($N_s = 2$)

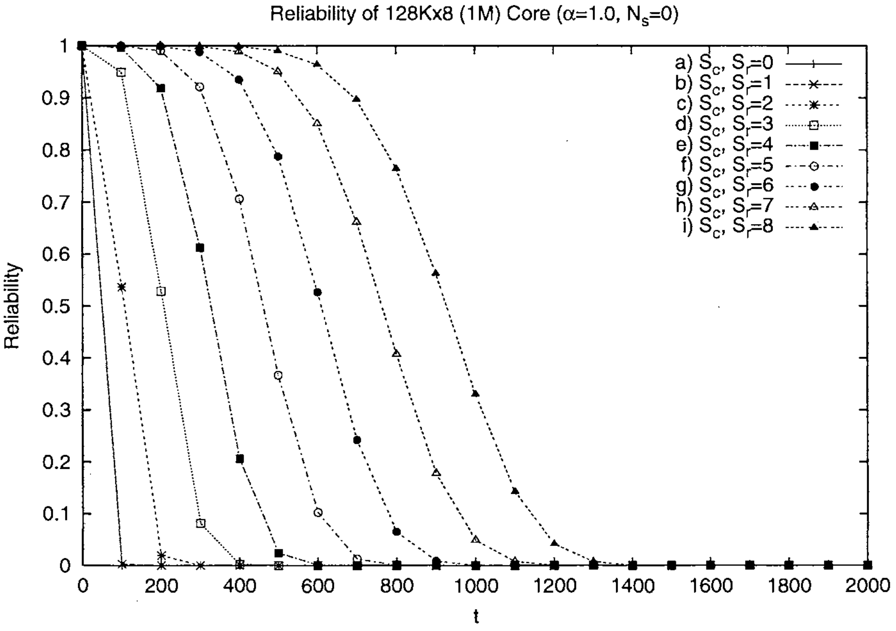

Figure 2.10: Analysis results of the sample 16K x 8 x 8 (1M) Core ($N_s = 0$)

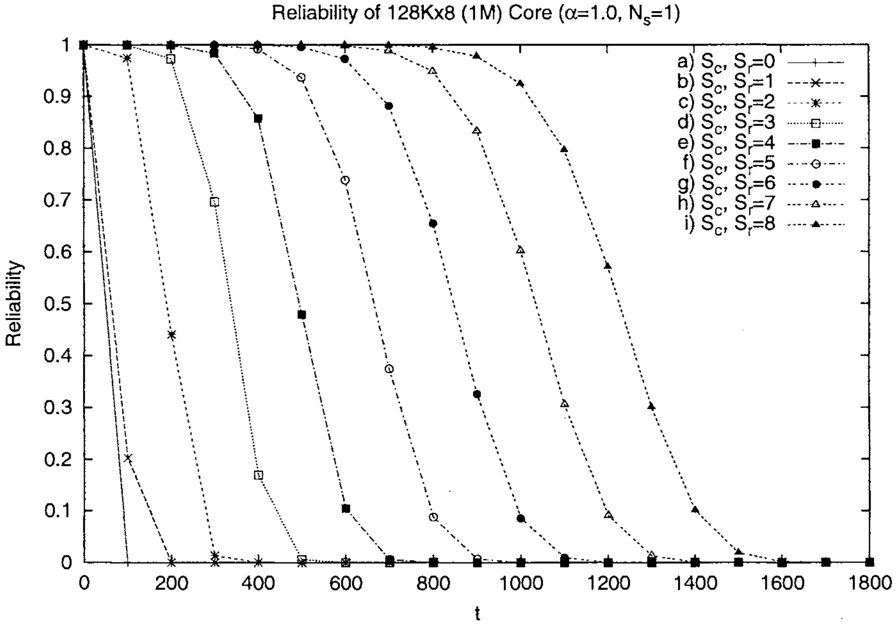

Figure 2.11: Analysis results of the sample 16K x 8 x 8 (1M) Core ($N_s = 1$)

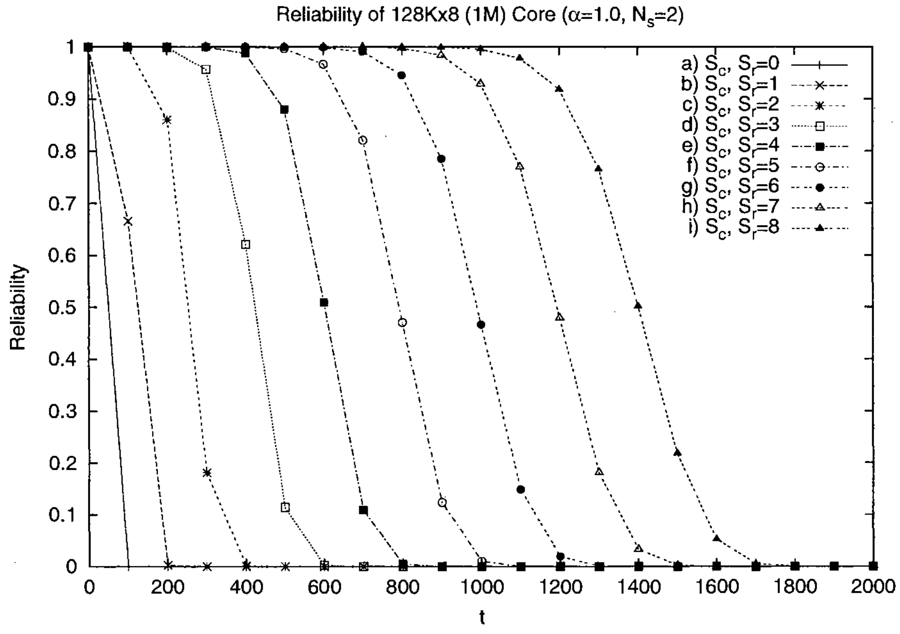

Figure 2.12: Analysis results of the sample 16K x 8 x 8 (1M) Core ($N_s = 2$)

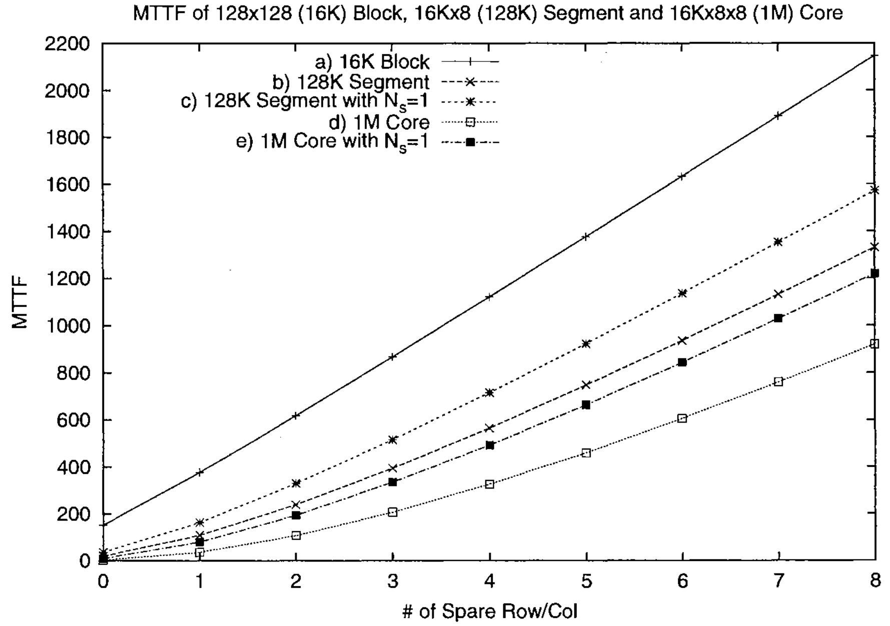

Figure 2.13: MTTF analysis results of the sample 1Mbit core

CHAPTER 3

REDUNDANCY PARTITIONING FOR BALANCED MANUFACTURING YIELD AND FIELD RELIABILITY

As advances in Ultra-Large-Scale-Integration (ULSI) technologies make possible the seamless embedding of numerous cores on a single chip (i.e., Commonly referred to as *System-On-Chip* technology), solid dependability becomes an urgent requirement of such ultra density and high performance systems since insignificant degradation or defects of core components could result in unacceptably low resultant SoC manufacturing yield and field reliability. However, fabricated embedded memory system cores cannot be physically replaced in the field. Thus, built-in self-test, diagnosis and repair circuits are commonly practiced along with ATE-based repair to assure improved manufacturing yield and field reliability of the embedded memory core [36, 35, 37, 39]. Although it is obvious that the combination of ATE and BISR is able to achieve significant *manufacturing yield* [1] and *field reliability* [2] enhancements for embedded memory system core, one problem still remains unsolved:

How the given redundancy can be partitioned into two groups (i.e., one for ATE repair and another one for BISR repair) to balance the manufacturing yield and field reliability for the maximum dependability?

Since ATE (for repairing manufacturing defects) and BISR (for repairing field faults) share the given common redundancy, balanced redundancy partitioning and

[1] the probability of being manufactured and repaired as functional

[2] a function of time which is defined by the conditional probability that the system performs correctly throughout the interval of time $[t_0, t]$ given the system was performing flawlessly at the initial time t_0

utilization techniques are very important to achieve optimal combination of yield and reliability of the embedded memory system core. Thus, dependability evaluation techniques for single and two dimensional redundancy architectures will be initially investigated to unveil the significance of redundancy balancing. Then, balanced redundancy partitioning and utilization techniques for both single and two dimensional redundancy architectures will be investigated. Extensive parametric simulation results will be also shown.

The organization of this chapter is as follows: In the following section (Section 3.1), a conceptual architectural model of the embedded memory core with both ATE and BISR repair capabilities will be shown and significance of redundancy partitioning and utilization for balanced yield and reliability will be discussed. In Sections 3.2 and 3.3, detailed yield and reliability assurance techniques for single and two dimensional redundancy cases will be shown. Then, balanced redundancy partitioning and utilization techniques for both cases will be proposed as well. A set of parametric simulations further verifies the effectiveness of the proposed redundancy balancing techniques in Section 3.4. Finally, discussion and conclusions will be given in Section 3.5.

3.1 Preliminaries

Figure (2.3) shows a model of the embedded memory system core under investigation in which both ATE-based factory repair and BISR-based field repair are practiced for manufacturing yield and field reliability enhancements. The embedded memory system core under investigation consists of the following components:

- IEEE JTAG (Joint Test Action Group) 1149.1 : External ATE interface for factory repair.

- Laser Fuse : A set of laser reconfigurable fuses to permanently program the given redundancy resources in the factory.
- BIST/BISD/BISR Processor : This system component governs the self-test, self-diagnosis, and self-repair procedures.
- Programmable Fuse : A set of programmable fuses to store additional reconfiguration signatures generated by the BIST/BISD/BISR processor in the field.
- Memory Array Interface : This component connects the EAB (embedded array block) array and the BIST/BISD/BISR Processor together. Data, address, control and repair data flow via this component.

The given embedded memory system core is tested and repaired as follows:

- Factory Repair Process: To circumvent the defects due to imperfect manufacturing processes, ATE communicates with the embedded memory system core via the external test equipment interface. Then, the laser fuse is permanently programmed to allocate redundancy to repair manufacturing defects in EABs.
- Field Repair Process: Whenever the host SoC is reset or powered, BISR tests, diagnoses and repairs EABs. The programmable fuse is programmed to store redundancy allocation information.

3.2 Single Dimensional Redundancy Case

The following notations will be used throughout this chapter:

- n_c : Number of columns (i.e., number of bits per word).
- n_r : Number of rows (i.e., number of words).

- s_c : Number of spare columns.
- s_{cm}: Number of spare columns used for manufacturing yield enhancement (i.e., $s_c - s_{cf}$).
- s_{cf}: Number of spare columns used for field reliability enhancement (i.e., $s_c - s_{cm}$).
- λ_m : Expected number of manufacturing defects per memory cell.
- λ_f : Field failure arrival rate of memory cell per unit time interval.
- Y : Manufacturing yield.
- $R(t)$: Field reliability at time t.
- $D(t)$: Overall dependability.

The yield of a single cell can be formulated by the exponential failure law as

$$Y_{cell} = e^{-\lambda_m} \tag{3.1}$$

Then, the probability of having n_r non-defective cells in a column (i.e., the yield of a column) can be written as

$$Y_{column} = (Y_{cell})^{n_r} \tag{3.2}$$

The given memory consists of n_c memory columns and s_{cm} spare memory columns for yield enhancement. The quorum size of n_c of the total of $n_c + s_{cm}$ columns are required to be functional. Thus, the yield of the given memory with column-redundancy can be formulated by the binomial distribution as follows:

$$Y = \sum_{i=0}^{s_{cm}} \binom{n_c + s_{cm}}{i} (Y_{column})^{n_c+s_{cm}-i} \cdot (1.0 - Y_{column})^i \tag{3.3}$$

Reliability assurance equations are similar to the yield assurance equations and can be expressed as follows:

$$R_{cell}(t) = e^{-\lambda_f \cdot t} \tag{3.4}$$

$$R_{column}(t) = (R_{cell})^{n_r} \tag{3.5}$$

$$\begin{aligned} R(t) &= \sum_{i=0}^{s_{cf}} \binom{n_c + s_{cf}}{i} (R_{column}(t))^{n_c+s_{cf}-i} \cdot (1.0 - R_{column}(t))^i && (3.6) \\ &= \sum_{i=0}^{s_c-s_{cm}} \binom{n_c + s_c - s_{cm}}{i} (R_{column}(t))^{n_c+s_c-s_{cm}-i} \cdot (1.0 - R_{column}(t))^i && (3.7) \end{aligned}$$

The conditional probability of having manufactured-as-good (i.e., Y) and not-failing-in-the-field during the time interval $[t_0, t]$ (i.e., $R(t)$) is referred to as *dependability* denoted by $D(t)$. Since Y and $R(t)$ are serial probabilities, the product of equations 3.3 and 3.7 can be used to formulate $D(t)$.

$$\begin{aligned} D(t) &= Y \cdot R(t) && (3.8) \\ &= \sum_{i=0}^{s_{cm}} \binom{n_c + s_{cm}}{i} (Y_{column})^{n_c+s_{cm}-i} \cdot (1.0 - Y_{column})^i \\ &\quad \times \sum_{i=0}^{s_c-s_{cm}} \binom{n_c + s_c - s_{cm}}{i} (R_{column}(t))^{n_c+s_c-s_{cm}-i} \\ &\quad \times (1.0 - R_{column}(t))^i && (3.9) \end{aligned}$$

To find the most balanced s_{cm}, $D(t)$ can be differentiated and solved with respect to s_{cm} as follows.

$$\frac{dD(t)}{ds_{cm}} = 0 \tag{3.10}$$

Note that $s_{cf} = s_c - s_{cm}$ holds and s_{cm} must be an integer value. So, both $\lceil s_{cm} \rceil$ and $\lfloor s_{cm} \rfloor$ must be evaluated to determine the final partitioning position. Figure (3.1) shows an example of an EAB with six spare columns. Later, they are partitioned into two groups: two spare columns for ATE repair and four spare columns for BISR repair.

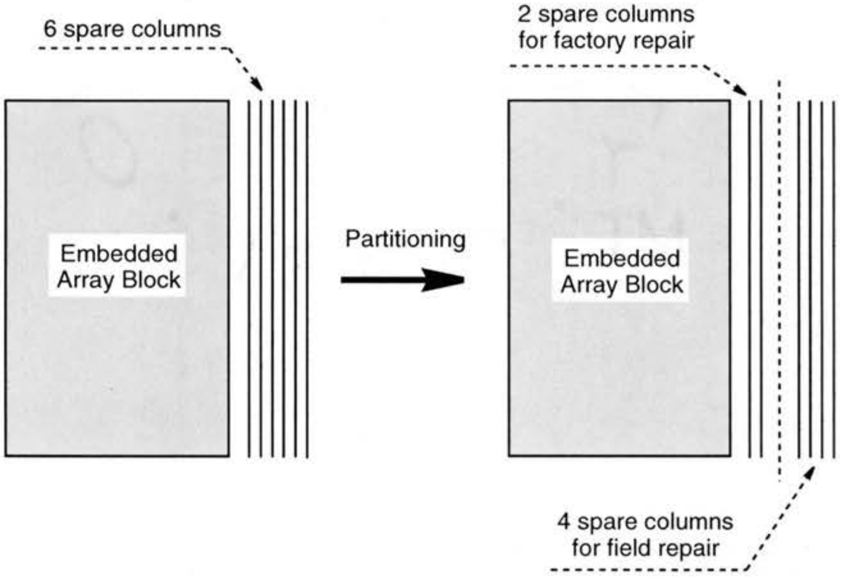

Figure 3.1: Example of redundancy partitioning for single dimensional redundancy

3.3 Two Dimensional Redundancy Case

The following notations will be used in addition to the ones given in the previous section throughout this chapter:

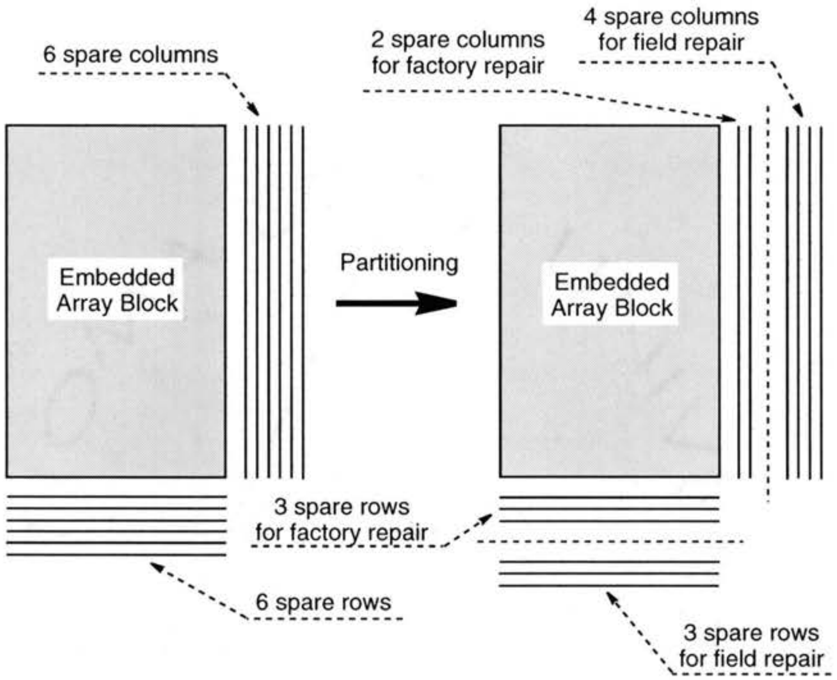
6 spare columns
Embedded Array Block
Partitioning
2 spare columns for factory repair
4 spare columns for field repair
Embedded Array Block
3 spare rows for factory repair
6 spare rows
3 spare rows for field repair

Figure 3.2: Example of redundancy partitioning for two dimensional redundancy

- s_r : Number of spare rows.
- s_{rm}: Number of spare rows used for manufacturing yield enhancement (i.e., $S_r - S_{rf}$).
- s_{rf}: Number of spare rows used for field reliability enhancement (i.e., $S_c - S_{rm}$).
- λ_{cm} : Expected number of manufacturing defects per memory column.
- λ_{rm} : Expected number of manufacturing defects per memory row.
- λ_{cf} : Field failure arrival rate of memory column per unit time interval.
- λ_{rf} : Field failure arrival rate of memory row per unit time interval.

Since row/column deletion is one of the NP-complete problems, there is no effective way to derive closed formulas for Y and $R(t)$. So, in this chapter, a line-based fault model is used rather than the cell-based fault model for 2-D case.

The yield of a row and a column can be approximated as $Y_{row} = e^{-\lambda_{rm}}$ and $Y_{column} = e^{-\lambda_{cm}}$.

Then, the yield of rows is

$$Y_{rows} = \sum_{i=0}^{s_{rm}} \binom{n_r + s_{rm}}{i} (Y_{row})^{n_r+s_{rm}-i} \cdot (1.0 - Y_{row})^i \tag{3.11}$$

and the yield of columns is

$$Y_{columns} = \sum_{i=0}^{s_{cm}} \binom{n_c + s_{cm}}{i} (Y_{colmun})^{n_c+s_{cm}-i} \cdot (1.0 - Y_{column})^i \tag{3.12}$$

Thus, the overall yield is

$$Y = Y_{rows} \times Y_{columns} \tag{3.13}$$

Likewise,

$$R_{row}(t) = e^{-\lambda_{rm} \cdot t} \tag{3.14}$$

$$R_{column}(t) = e^{-\lambda_{cm} \cdot t} \tag{3.15}$$

$$\begin{aligned} R_{rows}(t) &= \sum_{i=0}^{s_{rf}} \binom{n_r + s_{rf}}{i} (R_{row}(t))^{n_r+s_{rf}-i} \cdot (1.0 - R_{row}(t))^i \\ &= \sum_{i=0}^{s_r - s_{rm}} \binom{n_r + s_r - s_{rm}}{i} (R_{row}(t))^{n_r+s_r-s_{rm}-i} \cdot (1.0 - R_{row}(t))^i \end{aligned} \tag{3.16}$$

$$\begin{aligned} R_{columns}(t) &= \sum_{i=0}^{s_{cf}} \binom{n_c + s_{cf}}{i} (R_{colmun}(t))^{n_c+s_{cf}-i} \cdot (1.0 - R_{column}(t))^i \\ &= \sum_{i=0}^{s_c - s_{cm}} \binom{n_c + s_c - s_{cm}}{i} (R_{colmun}(t))^{n_c+s_c-s_{cm}-i} \\ &\quad \times (1.0 - R_{column}(t))^i \end{aligned} \tag{3.17}$$

$$R(t) = R_{rows}(t) \times R_{columns}(t) \tag{3.18}$$

The overall dependability, then, can be written as

$$D(t) = Y \times R(t) \tag{3.19}$$

To find the most balanced s_{cm} and s_{rm}, $D(t)$ can be differentiated and solved with respect to s_{cm} and s_{rm} as follows:

$$\frac{d^2 D(t)}{ds_{cm} ds_{rm}} = 0 \tag{3.20}$$

Note that $s_{cf} = s_c - s_{cm}$ and $s_{rf} = s_r - s_{rm}$ hold and s_{cm} and s_{rm} must be integer values. So, the combination of $\lceil s_{cm} \rceil$ & $\lfloor s_{cm} \rfloor$ and $\lceil s_{rm} \rceil$ & $\lfloor s_{rm} \rfloor$ must be evaluated to determine the final partitioning positions. Figure (3.2) shows an example of an EAB with six spare columns and six spare rows. Later, spare columns are partitioned into two groups: two spare columns for ATE repair and four spare columns for BISR repair and spare rows are also partitioned into two groups: three spare columns for ATE repair and three spare columns for BISR repair.

3.4 Parametric Simulations and Results

The effect of the redundancy balancing for both single and two dimensional cases will be studied through numerical experiments in this section. Parameters used in the simulation for the single dimensional redundancy case are summarized in Table 3.1 and for the two dimensional redundancy case are summarized in Table 3.2

Table 3.1: Simulation parameters for a one dimensional case

Parameters	n_c & n_r	s_c	λ_m	λ_f	t
Values	128	8	10^{-4}	10^{-5}	10

Table 3.2: Simulation parameters for a two dimensional case

Parameters	n_c & n_r	s_c & s_r	$\lambda_c m$ & $\lambda_r m$	$\lambda_c f$ & $\lambda_r f$	t
Values	128	8	10^{-2}	10^{-3}	10

Observation 1 *The simulation results for the single dimensional redundancy case are shown in Figures (3.3) - (3.8) and the following observations can be made:*

- *Symmetric Case* : In Figures (3.3) and (3.4), the dependability and its derivative of the given EAB are plotted with respect to s_{cm}. In this case, the parameters are selected in order that Y and $R(t)$ show exactly symmetric behaviors. Thus, the partitioning result is [4,4]: 4 spare columns for factory repair and 4 spare columns for field repair. The spares are evenly partitioned and utilized in this case.

- *Reliability-Intensive Case* : In Figures (3.5) and (3.6), the dependability and its derivative of the given EAB are plotted with respect to s_{cm}. In this case, the parameters are selected in order that more field faults than manufacturing defects are induced (i.e., $t = 10 \rightarrow 20$). Thus, the partitioning result is [3,5]: 3 spare columns for factory repair and 5 spare columns for field repair. The spares are partitioned and utilized in favor of field reliability enhancement in this case.

- *Yield-Intensive Case* : In Figures (3.7) and (3.8), the dependability and its derivative of the given EAB are plotted with respect to s_{cm}. In this case, the parameters are selected in order that more manufacturing defects than field faults are induced (i.e., $\lambda_m = 10^{-4} \rightarrow 2 \times 10^{-4}$). Thus, the partitioning result is [5,3]: 5 spare columns for factory repair and 3 spare columns for field repair. The spares are partitioned and utilized in favor of manufacturing yield enhancement in this case.

Case Study 1 *The redundancy balancing results, s_{cm} and s_{cf}, must be integer values. So, both $\lceil s_{cm} \rceil$ and $\lfloor s_{cm} \rfloor$ must be evaluated to determine the final partitioning position, if the solution to the $\frac{dD(t)}{ds_{cm}} = 0$ is not an integer value. An example is given in Figures (3.9) and (3.10), in which the redundancy optimization result is not an integer. Thus, both $D(t)$ at $\lceil s_{cm} \rceil$ and $D(t)$ at $\lfloor s_{cm} \rfloor$ are compared to find the optimal balancing. As a result, the proposed redundancy balancing technique partitions the redundancy into [6,2].*

Observation 2 *The simulation results for the two dimensional redundancy case are shown in Figures (3.11) - (3.16), the following observations can be extended from the observation 1:*

- *Symmetric Case* : In Figures (3.11) and (3.12), the dependability and its derivative of the given EAB are plotted with respect to s_{cm} and s_{rm}. In this case, the parameters are selected in order that Y and $R(t)$ show exactly symmetric behaviors. Thus, the partitioning result is {[4,4],[4,4]}: 4 spare columns and rows for factory repair and 4 spare columns and rows for field repair. The spares are evenly partitioned and utilized in this case.

- *Reliability-Intensive Case* : In Figures (3.13) and (3.14), the dependability and its derivative of the given EAB are plotted with respect to s_{cm} and s_{rm}. In this case, the parameters are selected in order that more field faults than manufacturing defects are induced (i.e., $t = 10 \rightarrow 20$). Thus, the partitioning result is {[3,5],[3,5]}: 3 spare columns and rows for factory repair and 5 spare columns and rows for field repair. The spares are partitioned and utilized in favor of field reliability enhancement in this case.

- *Yield-Intensive Case* : In Figures (3.15) and (3.16), the dependability and its derivative of the given EAB are plotted with respect to s_{cm} and s_{rm}. In this case, the parameters are selected in order that more manufacturing defects than field faults are induced (i.e., $\lambda_m = 10^{-4} \rightarrow 2 \times 10^{-4}$). Thus, the partitioning result is {[5,3],[5,3]}: 5 spare columns and rows for factory repair and 3 spare columns and rows for field repair. The spares are partitioned and utilized in favor of manufacturing yield enhancement in this case.

Case Study 2 *The 2-D redundancy balancing results, s_{cm} and s_{cf} and s_{rm} and s_{rf}, must be integer values. So, the following four combinations must be evaluated to find the optimal balancing, if the solution to the $\frac{d^2D(t)}{ds_{cm}ds_{rm}} = 0$ is not integer values:*

- $\lceil s_{cm} \rceil$ and $\lceil s_{rm} \rceil$

- $\lceil s_{cm} \rceil$ and $\lfloor s_{rm} \rfloor$

- $\lfloor s_{cm} \rfloor$ and $\lceil s_{rm} \rceil$
- $\lfloor s_{cm} \rfloor$ and $\lfloor s_{rm} \rfloor$

An example is given in Figures (3.17) and (3.18), in which the redundancy optimization result is not a pair of integers. Thus, all $D(t)$'s of the four possible balancing candidates shown above are compared to find the optimal balancing. As a result, the proposed redundancy balancing technique partitions the redundancy into {[2,6],[2,6]}.

3.5 Discussion

Among the cores for SoC integration, one of the most sensitive cores is the embedded memory core since memory cells are commonly considered as more prone to defects and faults than logic cells. Since cores cannot be physically replaced once they are fabricated onto a SoC, a combination of both ATE and BISR is commonly practiced. Proper partitioning and utilization of given shared redundancy is highly desirable to achieve balanced manufacturing yield and field reliability of the embedded memory system core. Thus, yield and reliability assurance techniques have been initially proposed for the single dimensional redundancy case, then extended to the two dimensional redundancy case. Since yield and reliability trade off with each other, dependability (i.e., $Y \times R(t)$) reaches its maximum only if properly partitioned groups of the given redundancy are utilized to repair both manufacturing defects (i.e., ATE-based repair) and field faults (i.e., BISR-based repair). To effectively achieve the balanced redundancy partitioning and utilization, the dependability equations are differentiated and solved with respect to the number of spares used to enhance manufacturing yield. Parametric simulation results have further verified that the proposed redundancy partitioning and utilization techniques for embedded memory system core achieves the theoretically optimal redundancy balancing. The proposed redundancy balancing techniques can be possibly incorporated into the existing CAD compilers for embedded memory system cores, thereby cost-effective partitioning and utilization of the shared redundancy can be realized.

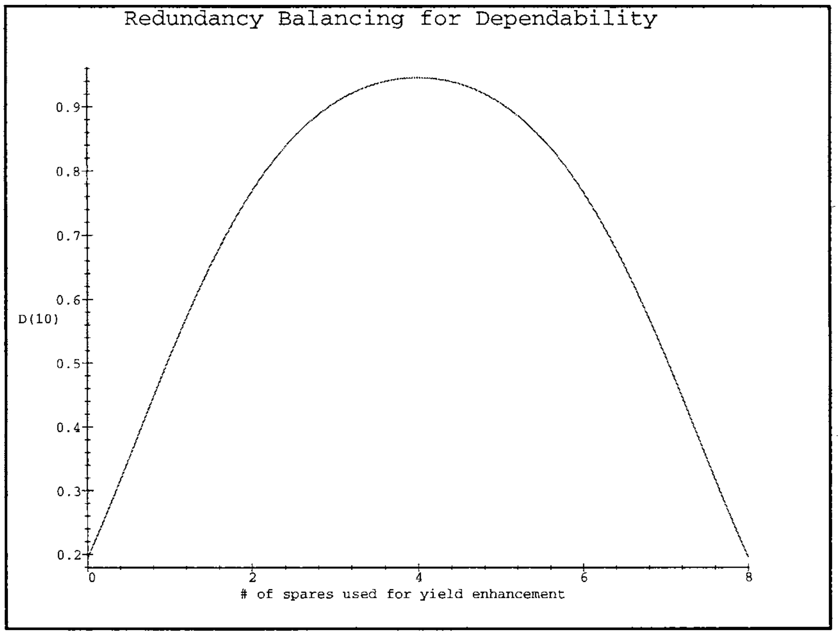

Figure 3.3: Symmetric 1D dependability graph

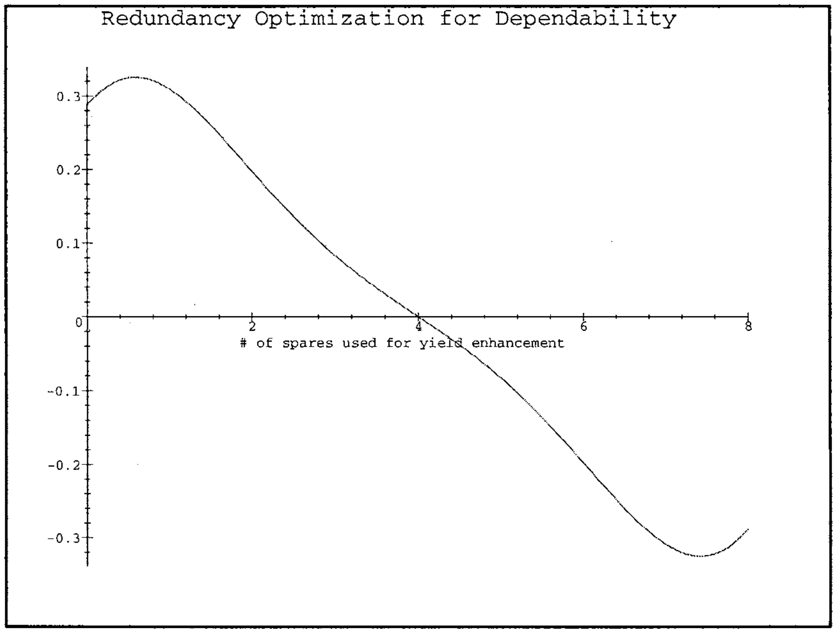

Figure 3.4: Balanced partitioning result [4,4]

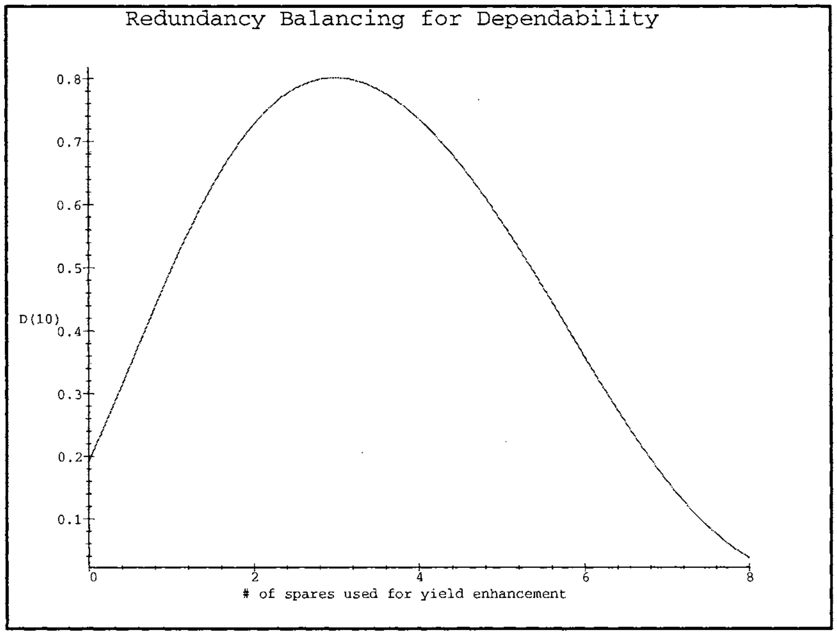

Figure 3.5: Reliability-intensive 1D dependability graph

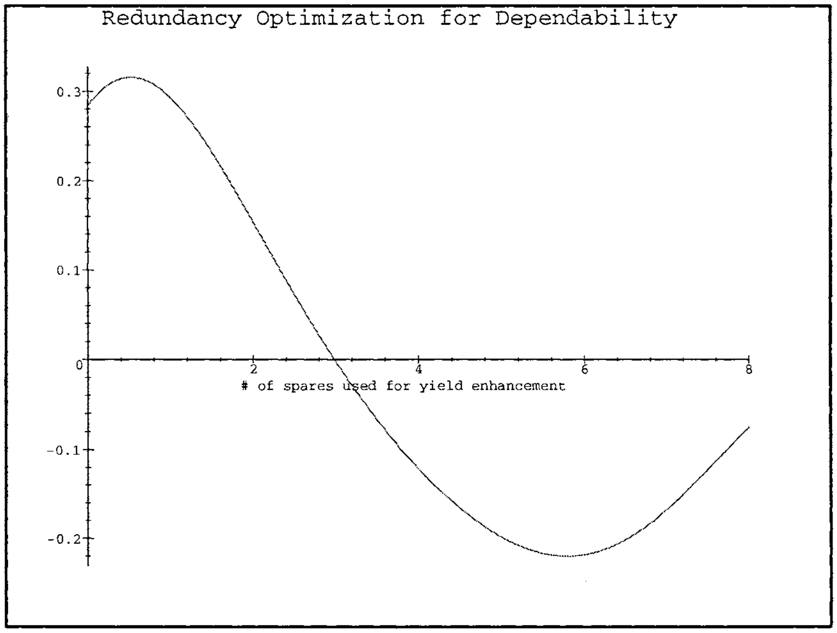

Figure 3.6: Balanced partitioning result [3,5]

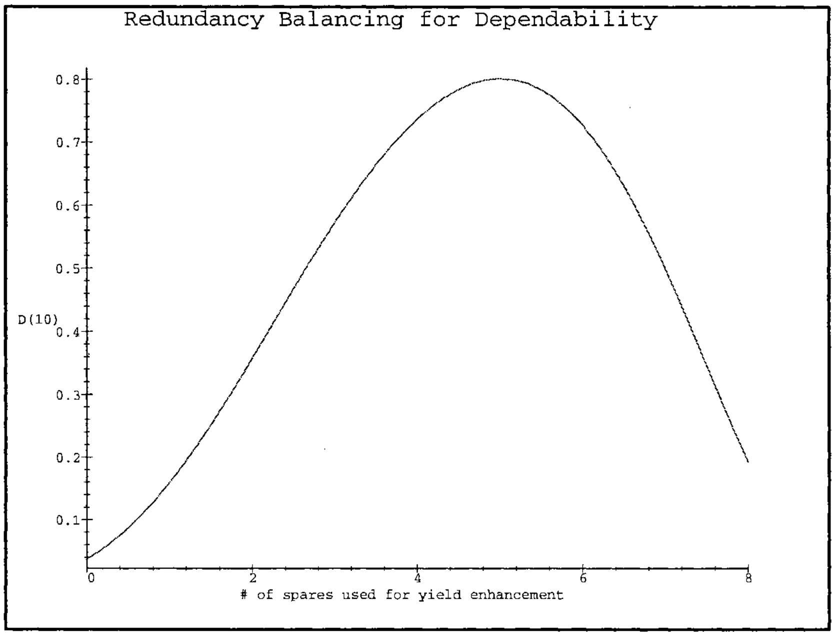

Figure 3.7: Yield-intensive 1D dependability graph

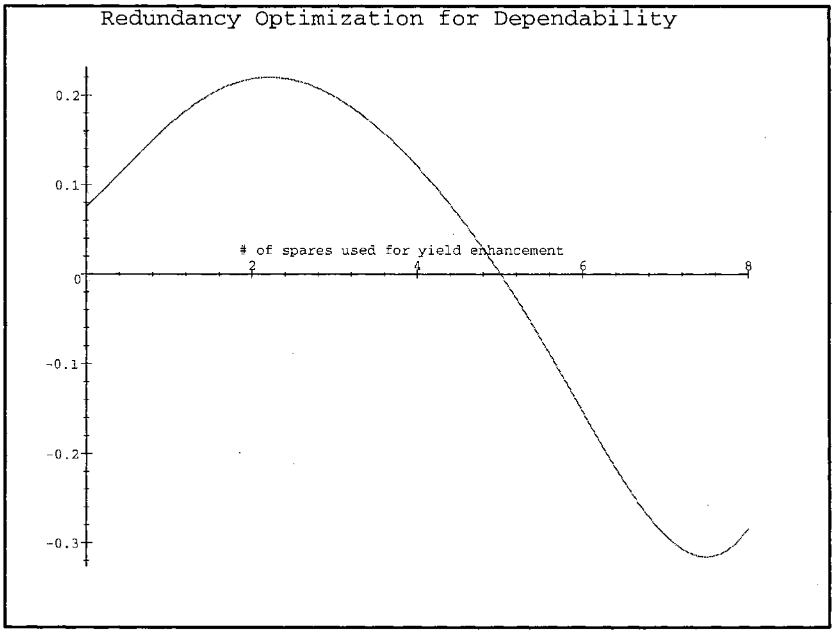

Figure 3.8: Balanced partitioning result [5,3]

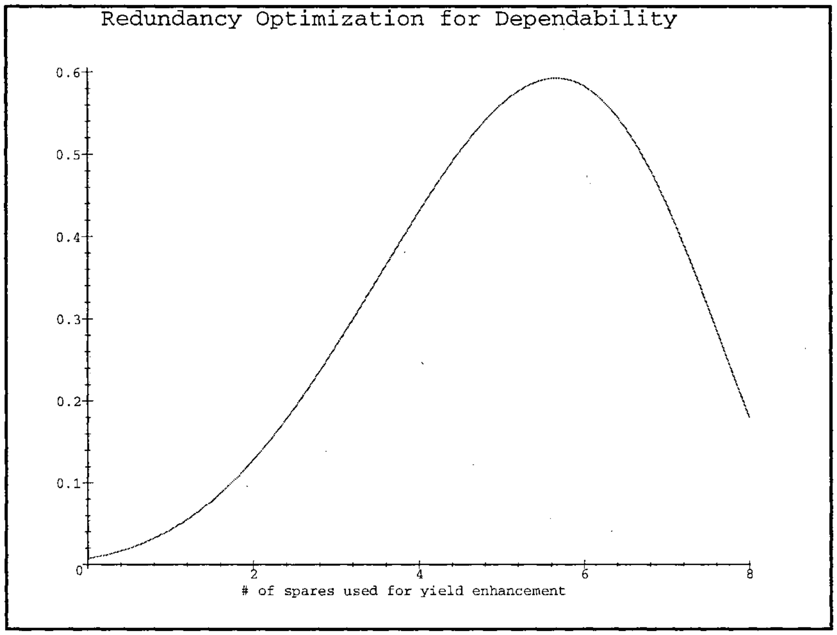

Figure 3.9: An example 1D dependability graph

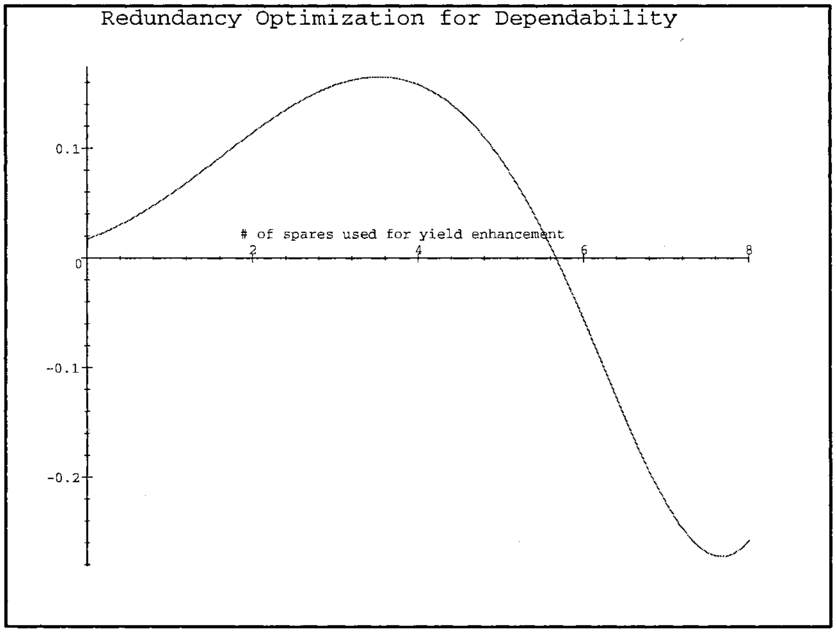

Figure 3.10: Balanced partitioning result [6,2]

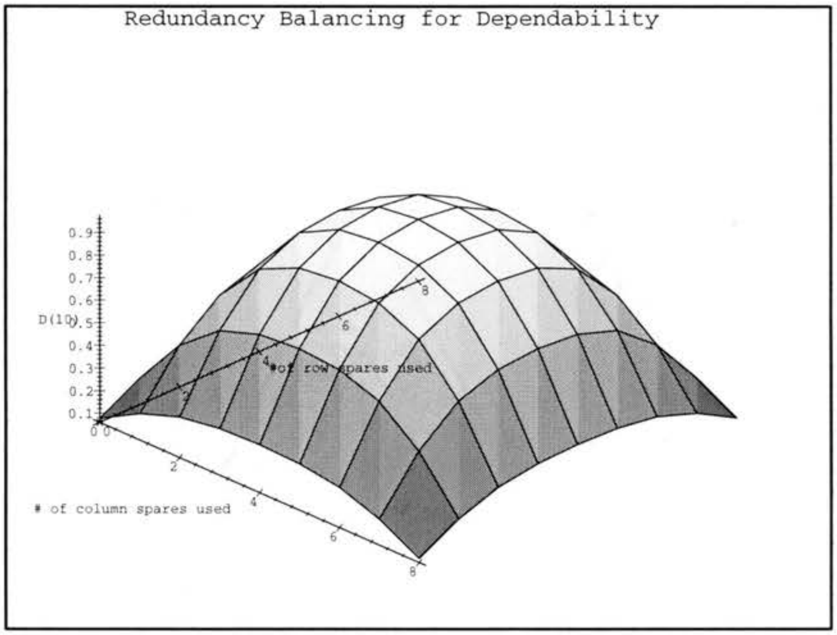

Figure 3.11: Symmetric 2D dependability graph

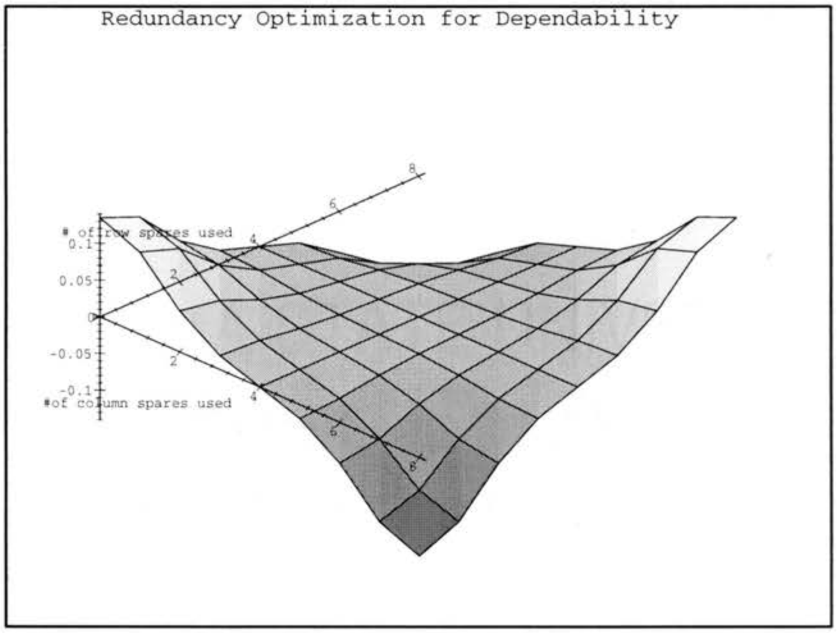

Figure 3.12: Balanced partitioning result {[4,4],[4,4]}

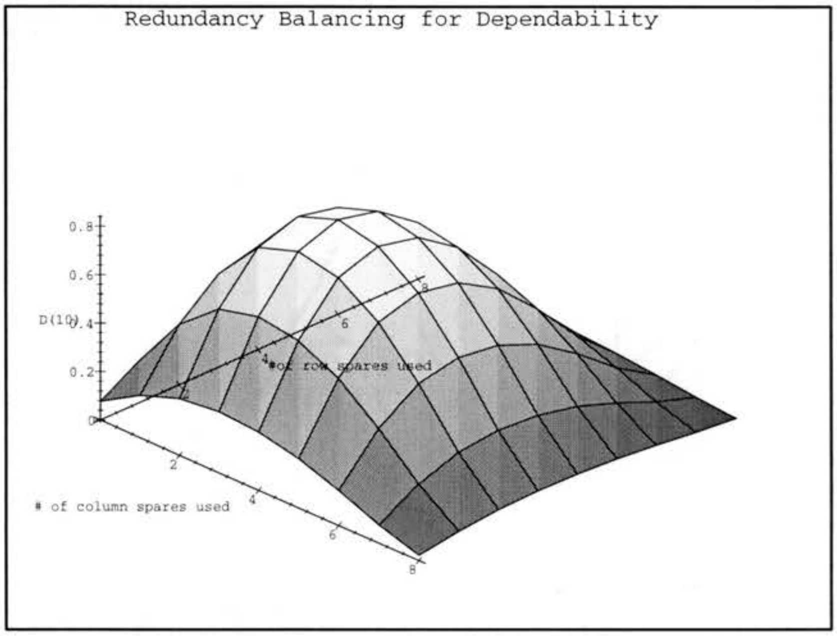

Figure 3.13: Reliability-intensive 2D dependability graph

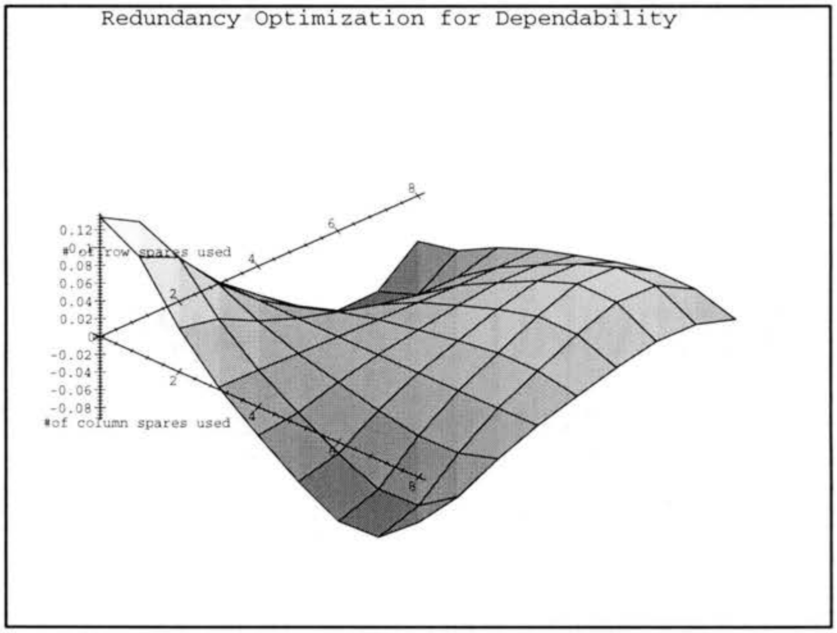

Figure 3.14: Balanced partitioning result {[3,5],[3,5]}

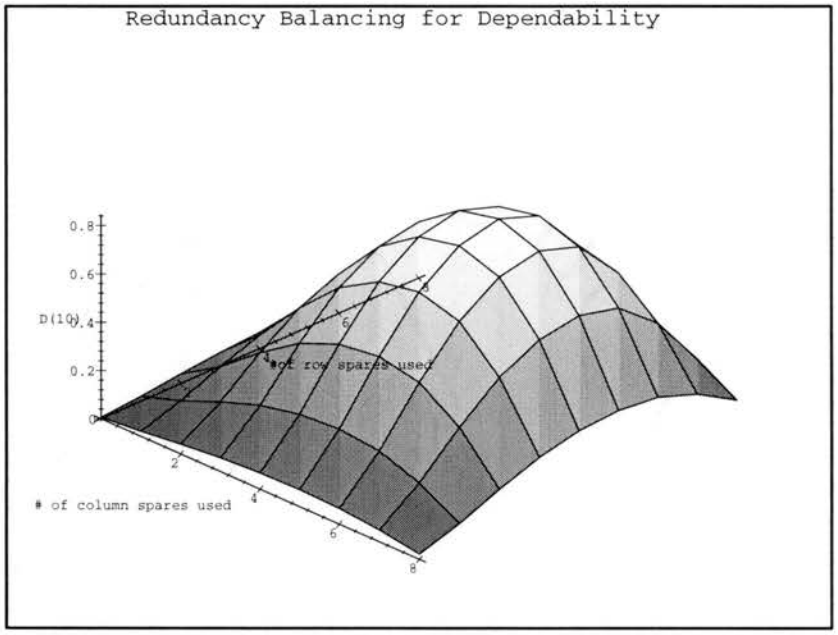

Figure 3.15: Yield-intensive 2D dependability graph

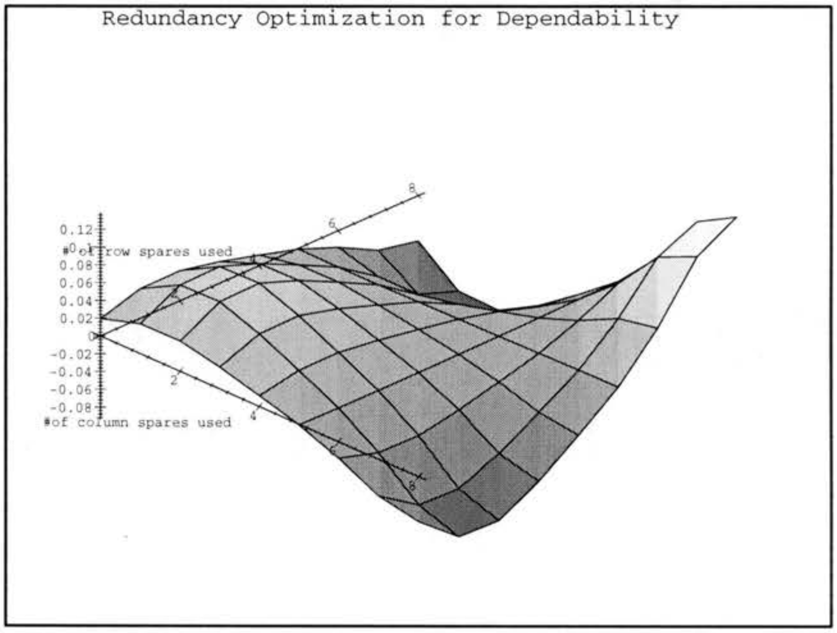

Figure 3.16: Balanced partitioning result {[5,3],[5,3]}

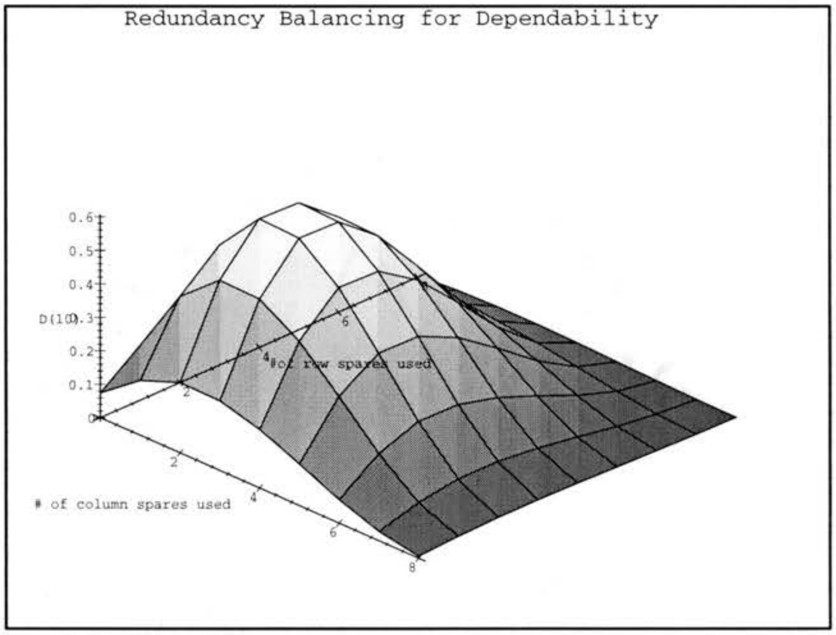

Figure 3.17: An example of 2D dependability graph

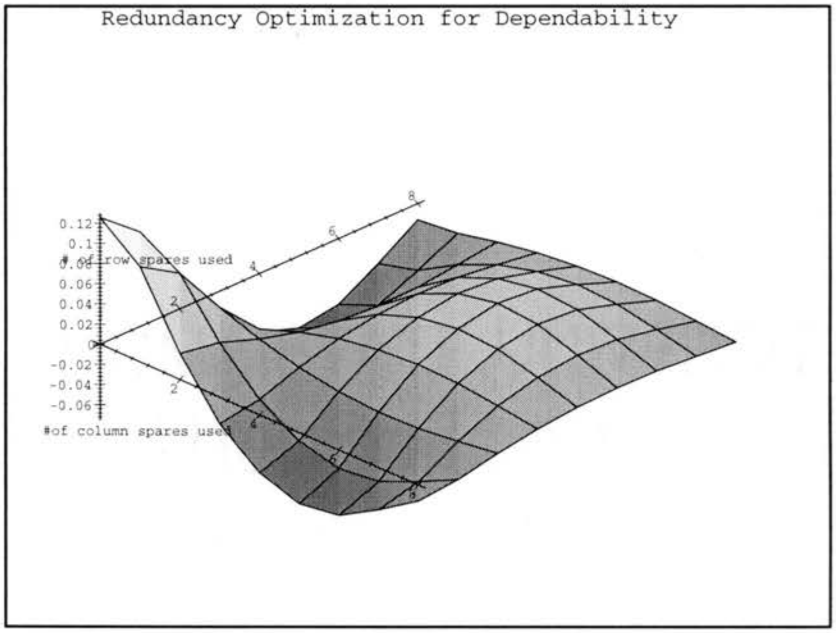

Figure 3.18: Balanced partitioning result {[2,6],[2,6]}

CHAPTER 4

CONNECTIVITY-BASED RECONFIGURABLE SOC REPAIR

The increasing demand on circuit speed and customizability has motivated high performance system development. System-on-Chip (SoC) technology provides potential advantages of high integration density, small interconnection delay and high system performance. For the purpose of customizability and repairability, embedding reconfigurable components along with ordinary cores with fixed functionality are commonly practiced [54, 55, 56, 57, 58, 59, 9, 23, 24, 25, 34]. The SoC with reconfigurable resources is commonly referred to as *Reconfigurable System-on-Chip (RSoC).* In this section, reliability-driven repair algorithms for an RSoC which exploits the reconfigurable redundancy will be discussed.

Test and *repair* are essential processes for achieving high reliability SoCs. After the fabrication phase, each SoC undergoes a test phase where defective cores are diagnosed and identified. Usually, defective cores on RSoCs are deemed to be reworked, which means that defective cores can be repaired by reconfigurable redundancy. The overall quality of the repair process significantly affects the final quality of the repaired RSoC. However, the repair process is not free from penalty since faulty core isolation and reconfiguration processes may affect the system integrity, the reconfigured interconnect structure routability, and the neighboring cores' functionality due to the serious interconnection network reconfiguration and associated programmable logic gate programming. For the extra long interconnects, even signal boosters are required to guarantee the integrity of the routed signals [3]. For more structured and reliable operations, protocol-based interconnction networks are commonly implemented for

SoCs as well [53].

How densely a *repair-candidate core* (i.e., the core to be repaired since it is diagnosed as defective) is connected to the neighboring parts of the RSoC is referred to as *connectivity.* If a repair-candidate core's connectivity is high, the repair process applied to the core may impair the reliability of the RSoC while it repairs the core, because the physically associated components with the core are many. Thus, selection of a repair-candidate core that results in less reliability degradation at each repair cycle seems to be crucial in the RSoC repair process.

The objective of this chapter is to extensively investigate the effect of the connectivity of repair-candidate cores on the overall yield of the repaired RSoC and to propose various repair scheduling strategies. Also, how improper repair scheduling could degrade the overall quality of repaired RSoCs will be studied.

The organization of the chapter is as follows: In the following section, review and preliminaries related to this research work will be given. In section 4.2, analytical characteristics of the RSoC repair process for the proposed repair scheduling strategies will be introduced. Section 4.3 will describe details of the proposed RSoC repair scheduling strategies. In section 4.4, extensive parametric analysis and simulations will be provided to demonstrate and verify the accuracy and efficiency of the proposed approaches.

4.1 Review and Preliminaries

In this work, an RSoC is modeled as a set of cores, their interconnect structure, reconfigurable interconnects, and reconfigurable logic redundancy as shown in Figure (4.1), in which the RSoC has six cores and corresponding interconnect structure. The repair process induces faulty core isolation, programmable logic reconfiguration, and interconnection rerouting. Although the process repairs the RSoC, the unreliability induced by the reconfiguration process (i.e., imperfect faulty core isolation, programmable logic reconfiguration and interconnection rerouting) may have negative effects on the quality of the repaired SoC.

Once fabricated, embedded cores cannot be physically replaced. Thus, embedded redundancy must be practiced for better yielding SoCs. Since a number of embedded hybrid cores are usually involved to design a SoC, the legacy modular redundancy scheme (i.e., embedding of extra cores to repair faulty cores) may require significant die area investment and its redundancy utilization also may be very low (i.e., unused spare cores are likely). The proposed reconfigurable redundancy architecture for SoC repair consists of two key components: *reconfigurable logic redundancy* and *reconfigurable interconnect redundancy.* The embedded cores are tested in order to identify faulty cores, if any. Then, the faulty cores and their interconnects are emulated by the reconfigurable logic and interconnect redundancy to restore the original functionality of the RSoC. The following case study clarifies the proposed RSoC core repair scheme based on the reconfigurable redundancy.

Case Study 3 *Suppose that an RSoC shown in Figure (4.1) is tested and diagnosed, and its core 4 is identified as faulty. Then, an emulated core 4′ is implemented by using the reconfigurable logic redundancy and core 4′'s interconnects are rerouted to the core 4′ via the reconfigurable interconnect redundancy. As a result, the repaired RSoC is shown in Figure (4.2).*

Upon proper fault simulation and analysis, the optimized amount of the reconfigurable redundancy can be determined prior to the fabrication of the RSoC. Thereby, both minimization of the die area overhead due to the redundancy and maximization of the RSoC yield can be achieved. Customized circuits can be also implemented by the reconfigurable redundancy.

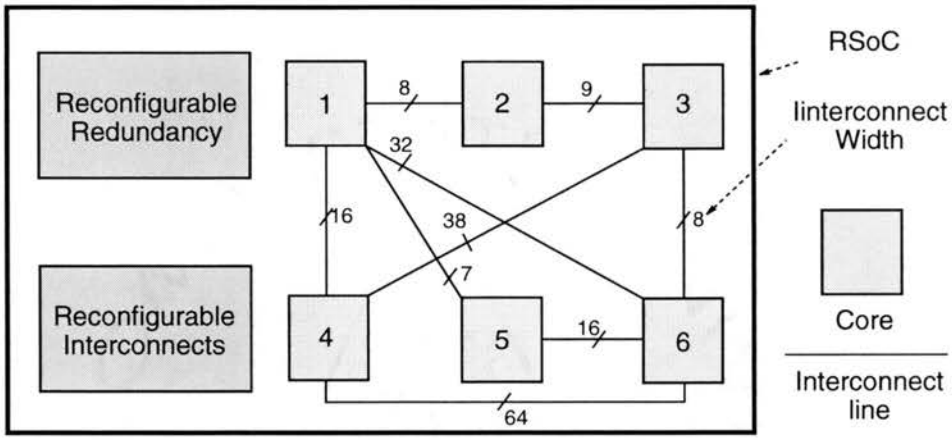

Figure 4.1: An example of an RSoC with reconfigurable interconnect network and redundancy

The following assumptions are made in this work:

- RSoC is fabricated with embedded cores and each core can be tested and diagnosed as faulty or not.
- No escaped cores are considered (i.e., 100% test coverage is assumed).
- Repair process, including defective core isolation, redundancy reconfiguration and interconnect reconfiguration, can be applied to the RSoC.
- Each core may have an uneven number of ports which connects to other core(s) via interconnects.
- Reconfigured and rerouted interconnects are considered as less dependable than the original interconnects due to the complexity of the resulting interconnect configuration.

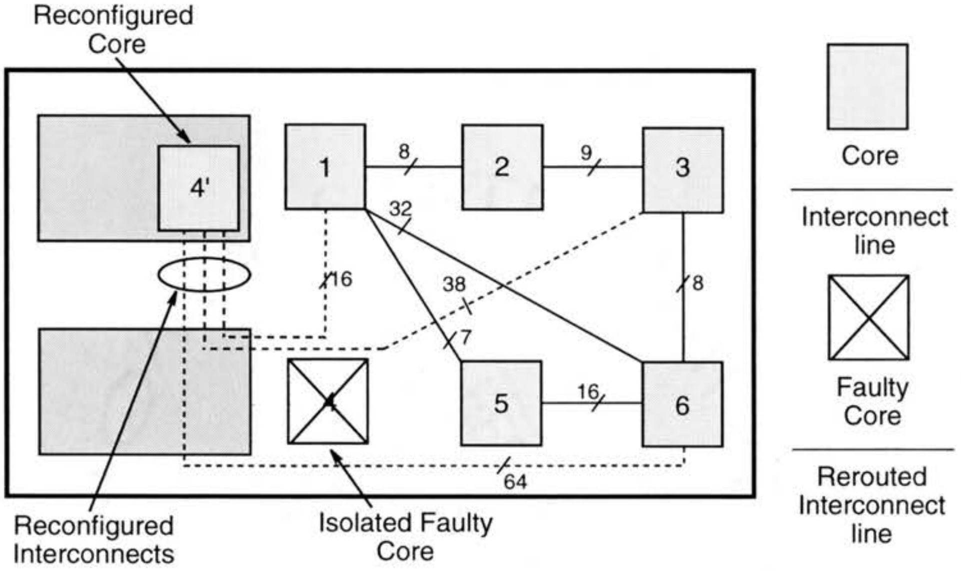

Figure 4.2: An example of a repaired RSoC

As clearly addressed in the assumptions given above, the repair procedure of an RSoC has not only an advantage but also a disadvantage. Proper testing and diagnosis of the embedded cores and reconfigurable redundancy utilization may enhance the overall yield of the RSoC since faulty cores can be replaced by reconfigured cores and rerouted reconfigurable interconnects. However, the reconfigured and rerouted interconnects may be less reliable due to the complexity of the resulting interconnect configuration. The unreliability associated with the reconfigured redundancy is modeled as the *unreliability impact factor (uif)*. For example, in Figure (4.1), repairing core 4 is assumed to affect neighboring (i.e., interconnected) cores 1, 3, 6 and associated interconnect structure. To accurately and effectively model the effect of both advantageous repair process and disadvantageous reconfigured and rerouted interconnects' unreliability, the following parameters are used to model the reliability of RSoC under repair:

- N : Number of cores in the RSoC.

- $y(i)$: Yield of i_{th} individual core.

- r : Maximum number of Repair cycles.

- YC : Overall yield of cores in the RSoC.

- YI : Overall yield of interconnect structure.

- Y : Overall yield of the RSoC which takes into account both the cores and the interconnect structure.

- uif : Base unreliability impact factor due to the repair process penalty.

- uif_{inc} : Incremental rate of uif per repair cycle.

- $cuif(i, j)$: Unreliability impact factor of the neighboring j_{th} core due to repair of the i_{th} core.

- α : Core unreliability impact factor coefficient. $0 \leq \alpha \leq 1$. It is fully dependent on the repair technology used. As $\alpha \to 1$, more reliability degradation due to the repair process is assumed to be applied to neighboring cores of the repair-candidate core.

- $\lambda_{i,j}$: Expected number of interconnect lines between the i_{th} core and the j_{th} core.

- $iuif(i)$: Interconnect structure unreliability impact factor due to repair of the i_{th} core.

- β : Interconnect structure unreliability impact factor coefficient. $0 \leq \beta \leq 1$. It is fully dependent on the repair technology used. As $\beta \rightarrow 1$, more reliability degradation due to the repair process is assumed to be applied to the interconnect structure.

- λ_i : Expected number of interconnect lines of the i_{th} core.

- y_{inc} : Yield increase rate of a core due to repair.

- $INT(i)$: Number of interconnect lines from the i_{th} core.

- $INT(i, j)$: Number of interconnect lines between the i_{th} and the j_{th} cores.

- $F(INT(i, j); \lambda_{i,j})$: cumulative Poisson probability function of $INT(i, j)$ (i.e., $\sum_{y=0}^{INT(i,j)} \frac{e^{-y}\lambda_{i,j}^{y}}{y!}$).

- RY : Yield of overall repair process (e.g., if 8 out of 10 defective RSoCs are repaired during the repair process, $RY = 8/10 = 80\%$).

4.2 Connectivity-based RSoC Repair Process

The repair process of a core enhances the overall yield of the RSoC, but the process is also likely to introduce yield degradation due to the complication of the reconfiguration process and is also prone to impair its neighboring (i.e., interconnected) cores' reliability since serious rerouting of connectivity would be experienced afterwards. Thus, the unreliability impact factor is modeled to be mainly determined by the number of interconnect lines between the repair-candidate core and its neighboring cores.

The characteristics of the repair process, so called *connectivity-based repair*, analyzed in this chapter are given as follows :

1. The i_{th} core is assumed to have initial yield of $y(i)$. Then, YC (i.e., overall yield of cores) of the given RSoC is initially determined by

   $$YC = \prod_{i=1}^{N} y(i) \tag{4.1}$$

   Then, the overall initial yield of RSoC (denoted by Y) is

   $$Y = YC \cdot YI \tag{4.2}$$

   where YI is the yield of the interconnect structure of RSoC.

2. The test and repair processes are performed after the fabrication phase.

3. Repair of a core degrades the reliability of the neighboring cores and the interconnect structure of the RSoC under repair. It is assumed that the unreliability impact factor (denoted by uif) due to the repetition of repair cycles increases as the RSoC undergoes a number of repair cycles. uif_n at the n_{th} repair cycle is given as

   $$uif_n = uif_{n-1} + (1 - uif_{n-1})uif_{inc} \tag{4.3}$$

   where uif_{inc} is the incremental rate of uif at each repair cycle due to the increasing complexity of the repair process as the number of repair cycles increases.

4. The repair of the i_{th} core is assumed to affect the neighboring (i.e., interconnected) core j, if it exists. The unreliability impact factor of the j_{th} core due to the repair of the i_{th} core is denoted by $cuif(i, j)$. $cuif(i, j)$ is modeled as a

function of $INT(i,j)$. The probability that the interconnect lines between the i_{th} and the j_{th} cores consist of exactly $INT(i,j)$ lines is

$$\frac{e^{-INT(i,j)}\lambda_{i,j}^{INT(i,j)}}{INT(i,j)!} \tag{4.4}$$

The increase in the possibility of having more degradation due to an increment of one interconnect line from $INT(i,j)-1$ to $INT(i,j)$ is also modeled to be determined by Equation (4.4), since the occurrence of degrading repair is directly influenced by the number of interconnect lines attached to the repair candidate core. Thus, without loss of generality, the cumulative Poisson probability function of $INT(i,j)$ (i.e., $\sum_{y=0}^{INT(i,j)} \frac{e^{-y}\lambda_{i,j}^{y}}{y!}$) is the reasonable one to simulate an incremental rate of uif imposed by the number of interconnect lines between the i_{th} and j_{th} cores. Thus, $cuif(i,j)$ is

$$uif_n + (1 - uif_n) \cdot \alpha \cdot F(INT(i,j); \lambda_{i,j}) \tag{4.5}$$

where α is a technology-dependent core unreliability impact factor coefficient and $F(INT(i,j); \lambda_{i,j})$ is a cumulative Poisson probability function of $INT(i,j)$. The parameter α simulates the efficiency of the repair process technology. As α approaches to 1 and $INT(i,j)$ increases, more reliability degradation is assumed to take place on the j_{th} core since the $\alpha \cdot F(INT(i,j); \lambda_{i,j})$ part approaches to 1. Thus, the yield of the j_{th} core after the repair of the i_{th} core can be formulated as

$$y(j)_n = y(j)_{n-1}(1 - cuif(i,j)_n) \tag{4.6}$$

5. Repair of the i_{th} core is assumed to impair the interconnect structure as well. The reliability degradation rate of the interconnect structure due to the repair of the i_{th} core is denoted by $iuif(i)$,

$$iuif(i)_n = uif_n + (1 - uif_n) \cdot \beta \cdot F(INT(i); \lambda_i) \tag{4.7}$$

where β is a technology-dependent interconnect structure damage coefficient and $F(INT(i); \lambda_i)$ is a cumulative poisson probability function $(= \sum_{y=0}^{INT(i)} \frac{e^{-y}\lambda_i^y}{y!})$ which simulates the incremental rate of the reliability degradation due to the number of interconnect lines of the i_{th} core. As β approaches to 1 and $INT(i)$ increases, more reliability degradation is assumed to be applied to the j_{th} core since the $\beta \cdot F(INT(i,j); \lambda_{i,j})$ part approaches to 1.

6. The yield of the i_{th} core after the repair process is given by

$$y(i)_n = y(i)_{n-1} + (1 - y(i)_{n-1})y_{inc} \tag{4.8}$$

7. The overall yield of the cores on RSoC after the n_{th} repair cycle becomes

$$YC_n = \prod_{i=1}^{N} y(i)_n \tag{4.9}$$

8. The overall yield of the interconnect structure on RSoC after the n_{th} repair cycle can be formulated as

$$YI_n = YI_{n-1}(1 - iuif(i)_n) \tag{4.10}$$

9. The overall yield of the RSoC after the n_{th} repair cycle then becomes

$$Y_n = YC_n \cdot YI_n \tag{4.11}$$

4.3 Connectivity-based RSoC Repair Scheduling

Every core on an RSoC is tested after the fabrication phase. Each core can be tested and diagnosed as non-faulty with the probability of y and as faulty with the probability of $\bar{y}$. If there is only one faulty core detected during the test phase, the core will be isolated and repaired. If more than one bad core is detected during the test phase, the order of repair (referred to as the *repair schedule*) must be properly arranged. In each repair cycle, in other words, selecting an appropriate repair-candidate core which has the least impact on the overall RSoC yield is a natural choice for optimal scheduling.

Table 4.1: Adjacency matrix representation of Figure (4.1).

	Core 1	Core 2	Core 3	Core 4	Core 5	Core 6
Core 1	0	8	0	16	7	32
Core 2	8	0	9	0	0	0
Core 3	0	9	0	38	0	8
Core 4	16	0	38	0	0	64
Core 5	7	0	0	0	0	16
Core 6	32	0	8	64	16	0

The RSoC structure shown in Figure (4.1) can be viewed as a weighted graph with six vertices and nine weighted edges. A simple way to represent the graph is to use a two-dimensional array so called an adjacency matrix representation. The equivalent adjacency matrix of Figure (4.1) is shown in Table (4.1). The space requirement of the representation is $O(N^2)$ where N is the number of cores on the RSoC.

If the RSoC is sparsely interconnected, a better solution is adjacency list representation such as shown in Figure (4.3). The space requirement for this representation

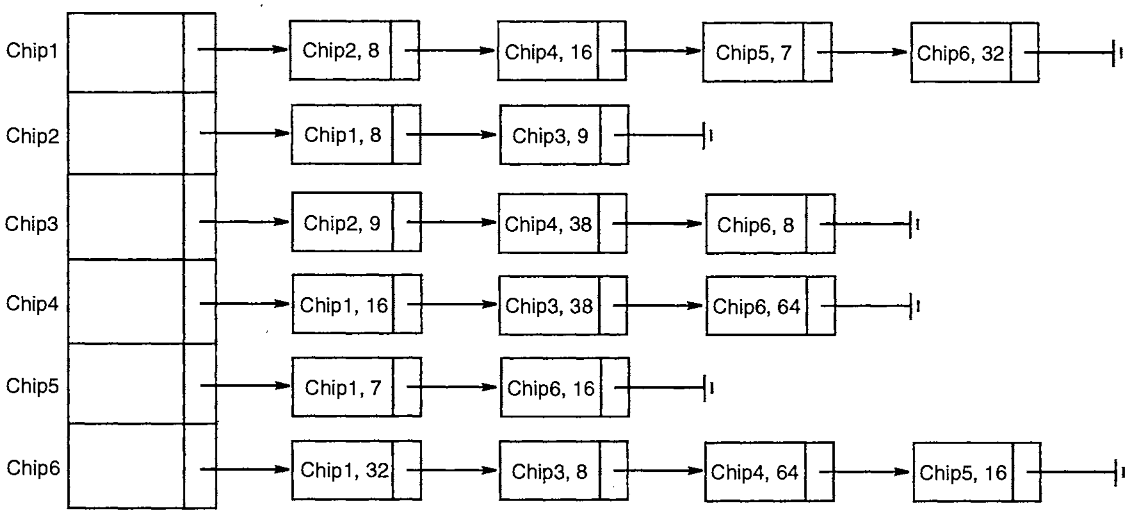

Figure 4.3: Adjacency list representation of Figure (4.1).

is $O(N + E)$ where N is the number of cores and E is the number of edges between cores on the RSoC. For RSoCs with a greater number of cores which are sparsely interconnected, the adjacency list representation can save the space requirement. For RSoCs with a lesser number of cores which are densely interconnected, the adjacency matrix is the choice. One of the two representations can be chosen accordingly, in practice.

For the proposed RSoC model, the number of interconnect lines and the number of neighboring cores attached to a repair-candidate core determines the resulting yield of the RSoC after each repair cycle. Four possible repair scheduling strategies are proposed as follows:

- *Smallest Number of Interconnects First (SNIF)* - Among those diagnosed as faulty cores, the one which has the smallest number of interconnect lines is to be repaired first.

- *Largest Number of Interconnects First (LNIF)* - Among those diagnosed as faulty cores, the one which has the largest number of interconnect lines is to be repaired first.

- *Smallest Number of neighboring Cores First (SNCF)* - Among those diagnosed as faulty cores, the one which has the smallest number of neighboring cores is to be repaired first.

- *Largest Number of neighboring Cores First (LNCF)* - Among those diagnosed as faulty cores, the one which has the largest number of neighboring cores is to be repaired first.

Since $LNIF$ and $LNCF$ repair scheduling strategies are supposed to repair the most reliability degrading core first, they do not have advantages in practice. However, they are also analyzed to be compared with the $SNIF$ and $SNCF$ repair scheduling strategies. The conceptual processes of $SNIF$ and $SNCF$ RSoC repair strategies are depicted in the flow chart shown in Figure (4.4).

4.4 Parametric Analysis

In this section, the effects of the connectivity-based RSoC repair scheduling are investigated through numerical experiments. An RSoC system with $N = 15$, $y = 0.99$, and $YI = 0.9999$ is considered. The yield of the RSoC before an application of the repair process can be calculated as a series product of the $y(i)$ of all the cores (i.e., $\prod_{i=1}^{N} y(i)$) and the yield of the interconnect structure (i.e., YI). In Table (4.2), the overall RSoC yield Y and $\overline{Y}$ are given where $\overline{Y}$ is subdivided into six categories according to the number of defective cores on the RSoC (denoted by dc). The following can be observed from Table (4.2):

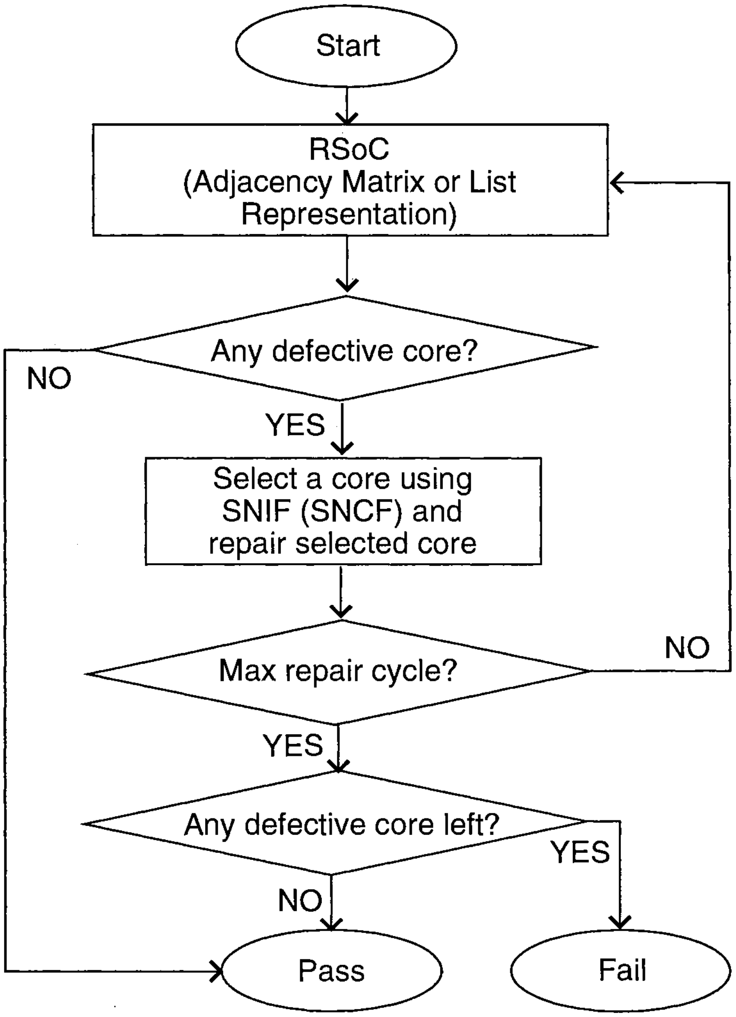

Figure 4.4: Flow chart of the SNIF and SNCF

- Among those 14.0028% RSoCs with defects, 13.0298% of them have one defective core identified, 0.9213% of them have two defective cores identified, 0.000403% of them have three defective cores identified, 0.000012% of them have four defective cores identified, and 0.000016% of them have more than five defective cores identified.

- Since RSoCs with $dc > 5$ are very few and then almost ignorable, the maximum allowed number of repair cycles $r = 5$ is applied.

- 0.00086% RSoCs in the category di (i.e., defective interconnect structure) does not have defective cores, but they have a defective interconnect structure. Since RSoCs in the di category are very few, no repair process is applied in this example.

Table 4.2: Y and $\overline{Y}$ of the given RSoC without repair

Y	$\overline{Y}$					
85.9972%	14.0028%					
	$dc = 1$	$dc = 2$	$dc = 3$	$dc = 4$	$dc >= 5$	di
	13.0298%	0.9213%	0.000403%	0.000012%	0.000016%	0.000086%

To compare the proposed repair scheduling strategies, the values of y_{inc}, uif, and uif_{inc} are set to 0.1 and the value of λ_i is set to 9, arbitrarily. In Tables (4.3) and (4.4), the repair performances of those proposed strategies, measured in the percentage of repaired RSoCs at each repair cycle, are shown. For example, 13.0298% of RSoCs contain one defective core and 62.4108% of them are repaired in the first repair cycle of SNIF, and 0.9213% of RSoCs contain two defective cores and 27.4829% of them are repaired in the second repair cycle, and so on.

Table 4.3: Performance comparison of the proposed repair strategies at each repair cycle where $\alpha, \beta = 0.05$

	r=1	r=2	r=3	r=4	r=5
SNIF	62.4108%	27.4829%	0.0661%	0%	0%
LNIF	33.7273%	0.09249%	0.5%	0%	0%
SNCF	67.2667%	33.2356%	8.1886%	0%	0%
LNCF	36.3137%	9.2478%	1.4888%	0%	0%

Table 4.4: Performance comparison of the proposed repair strategies at each repair cycle where $\alpha, \beta = 0.5$

	r=1	r=2	r=3	r=4	r=5
SNIF	59.3532%	14.6424%	0.2481%	0%	0%
LNIF	0.7552%	0.0109%	0%	0%	0%
SNCF	45.1135%	9.8665%	0.4963%	0%	0%
LNCF	2.5941%	0.0868%	0%	0%	0%

By comparing the results shown in Tables (4.3) and (4.4), the following can be observed:

1. Even with relatively small values of α and $\beta = 0.05$, (i.e., less core and interconnect degradation due to the repair process) the repair scheduling plays an important role in the RSoC repair process. Thus, it is shown that SNIF and SNCF outperform LNIF and LNCF at every repair cycle.

2. As relatively larger values of α and $\beta = 0.5$ are applied (i.e., more core and interconnect degradation due to the repair process), the difference in the repairability between SNIF (SNCF) and LNIF (LNCF) becomes more significant.

3. An appropriate selection of the repair schedule definitely affects repair yield with any values of α and β.

Repairability at each repair cycle and RY (i.e., the yield of the whole repair process, which is $\frac{total\# \ of \ repaired \ RSoCs}{total \ \# \ of \ tested\text{-}as\text{-}bad \ RSoCs}$) of SNIF and SNCF repair strategies at different values of α and β are more extensively experimented with and the results are shown in Tables (4.5) and (4.6). The values of α and β are arbitrarily set to be equal for the simplicity of the analysis.

Table 4.5: Repair performance of SNIF for the given parameters

α, β	r=1	r=2	r=3	r=4	r=5	RY
0.0	62.7546%	27.0378%	5.4590%	0%	0%	60.23%
0.2	61.3831%	21.3936%	1.7369%	0%	0%	58.57%
0.4	60.0254%	16.9108%	0.4962%	0%	0%	57.01%
0.6	58.6839%	12.7862%	0%	0%	0%	55.49%
0.8	57.3569%	9.6059%	0%	0%	0%	54.04%
1.0	56.0461%	7.1746%	0%	0%	0%	52.66%

Table 4.6: Repair performance of SNCF for the given parameters

α, β	r=1	r=2	r=3	r=4	r=5	RY
0.0	69.7278%	37.0888%	10.4218%	0%	0%	67.40%
0.2	59.8819%	23.3257%	3.7220%	0%	0%	57.31%
0.4	50.0361%	13.5135%	0.9925%	0%	0%	43.20%
0.6	40.1909%	6.9467%	0%	0%	0%	37.88%
0.8	30.3450%	2.9632%	0%	0%	0%	28.46%
1.0	20.4999%	0.9009%	0%	0%	0%	19.15%

In Figures (4.5)-(4.8), Y of SNIF at different values of N (i.e., 5, 10, and 15), α and β (i.e., 0.05, 0.1, 0.25 and 0.5), and y (i.e., 0.8 - 1.0) are shown versus LNIF. In Figures (4.9)-(4.12), Y of SNCF at different values of N (i.e., 5, 10, and 15), α and β (i.e., 0.05, 0.1, 0.25 and 0.5), and y (i.e., 0.8 - 1.0) are shown versus LNCF. By comparing the results of Figures (4.5)-(4.12), the following observations can be drawn:

1. Using proper connectivity-based repair scheduling strategies (i.e., SNIF and

SNCF), a higher Y of RSoC can be achieved.

2. SNIF and SNCF always outperform LNIF and LNCF.
3. As α and β increase, the difference between Y of SNIF and Y of LNIF increases. It is the same for SNCF and LNCF.
4. With relatively smaller α and β values, Y of both SNIF and SNCF perform similarly. However, SNIF performs better than SNCF as α and β increase.
5. In practice, SNCF is likely to be the choice when smaller α and β values are applied, since it counts only the number of neighboring chips which is simpler than counting the number of interconnect lines.
6. In practice, SNIF is likely to be the choice when larger α and β values are applied since it has less impact on Y of RSoC than SNCF.

4.5 Discussion

This chapter has presented a new model for analyzing the yield of RSoC systems with repair processes based on the effect of the connectivity of the repair-candidate chip where yield degradation of both neighboring chips and interconnect structure due to the complexity of the reconfigured logic and interconnect redundancy is taken into account. Two approaches, *Smallest Number of Interconnections First (SNIF)* and *Smallest Number of Neighboring Chips First (SNCF)* have been proposed. Two other scheduling policies, *Largest Number of Interconnections First (LNIF)* and *Largest Number of Neighboring Chips First (LNCF)* also have been introduced and analyzed, and it has been shown how improperly scheduled repair processes could impair the yield of RSoCs. Extensive parametric analysis and comparison of the proposed approaches have demonstrated the efficiency of the proposed RSoC repair scheduling strategies (i.e., $SNIF$ and $SNCF$). From the results, it is obvious that a higher repair yield of RSoCs can be expected when proper RSoC repair scheduling strategies such as $SNIF$ and $SNCF$ are applied. Also, it has been shown that $SNIF$ tolerates a higher chip and interconnect yield degradation due to the repair process (i.e., higher α and β) than $SNCF$ does, while $SNCF$ results in a higher yield when less chip and interconnect yield degradation due to the repair process (i.e., lower α and β) is assumed.

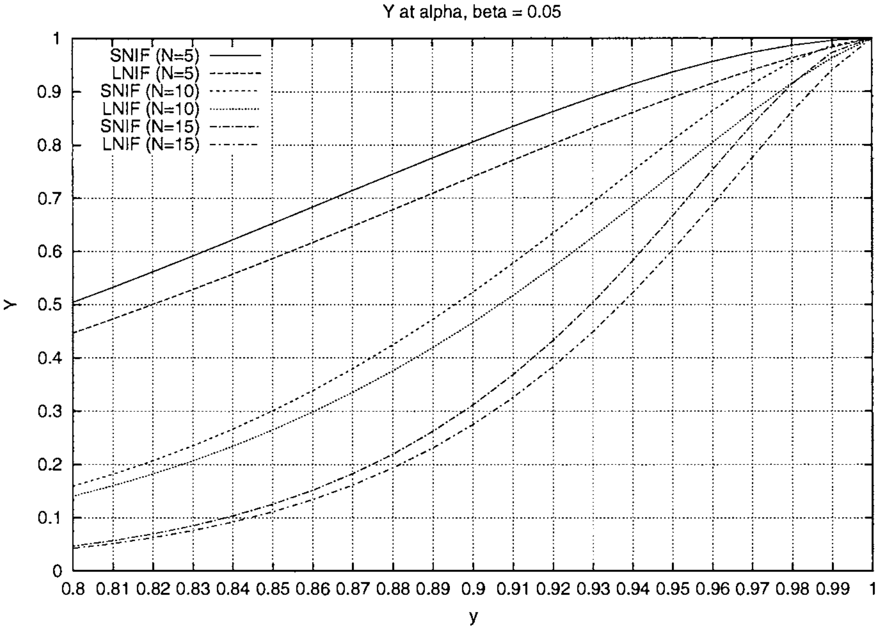

Figure 4.5: Yield of SNIF and LNIF at $\alpha, \beta = 0.05$

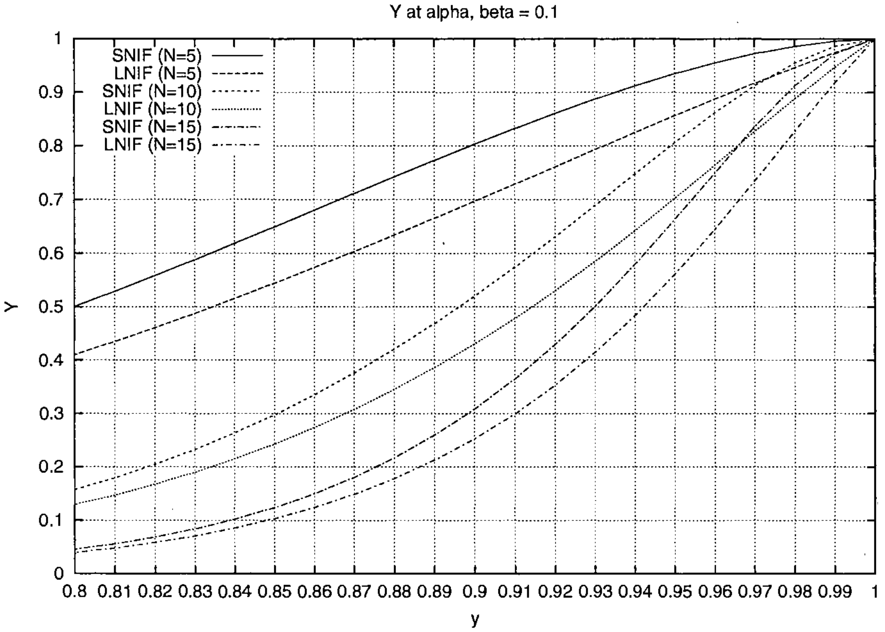

Figure 4.6: Yield of SNIF and LNIF at $\alpha, \beta = 0.1$

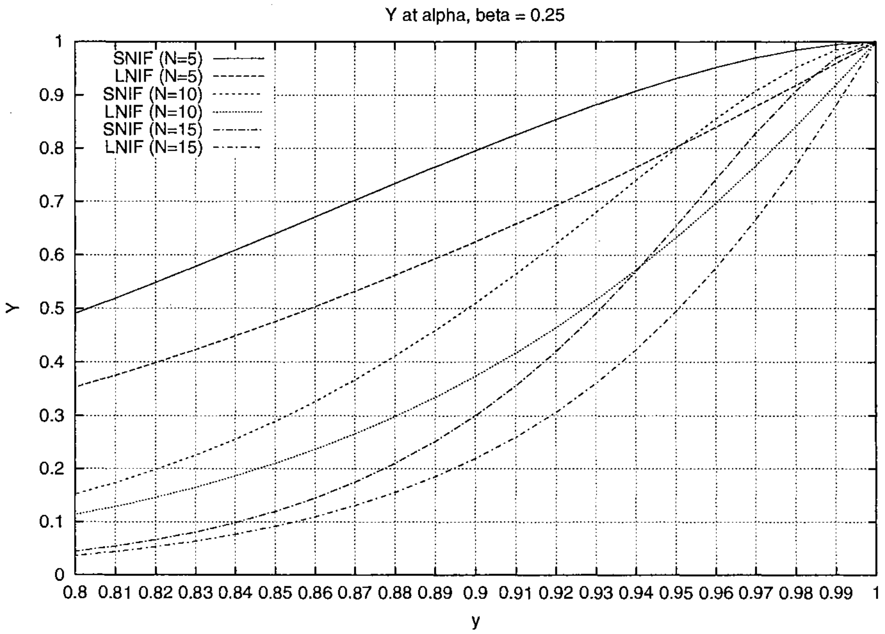

Figure 4.7: Yield of SNIF and LNIF at $\alpha, \beta = 0.25$

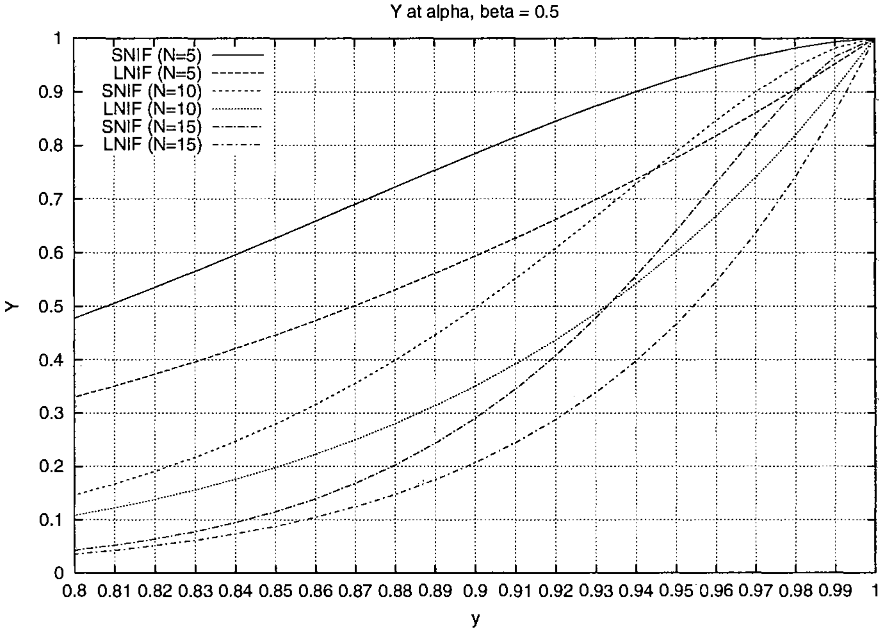

Figure 4.8: Yield of SNIF and LNIF at $\alpha, \beta = 0.5$

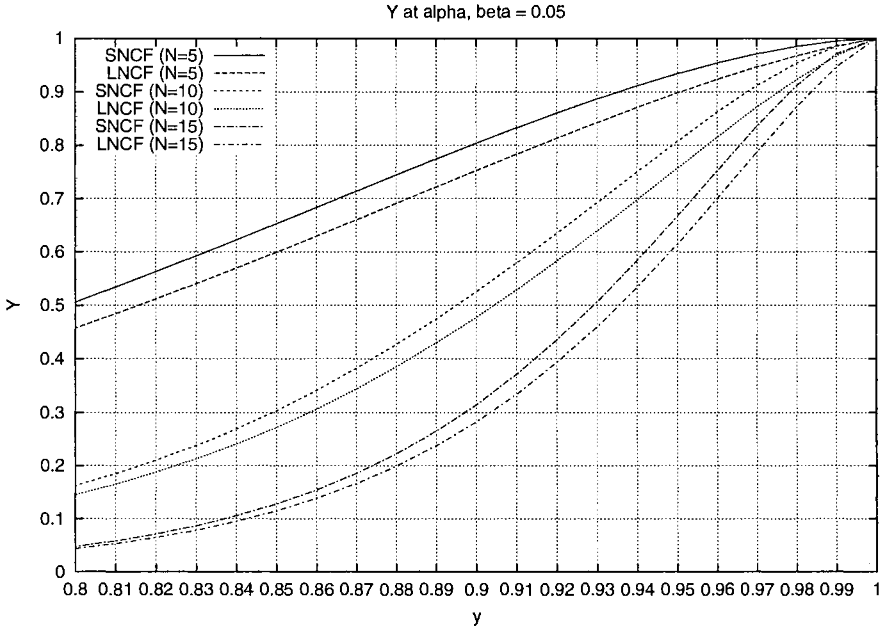

Figure 4.9: Yield of SNCF and LNCF at $\alpha, \beta = 0.05$

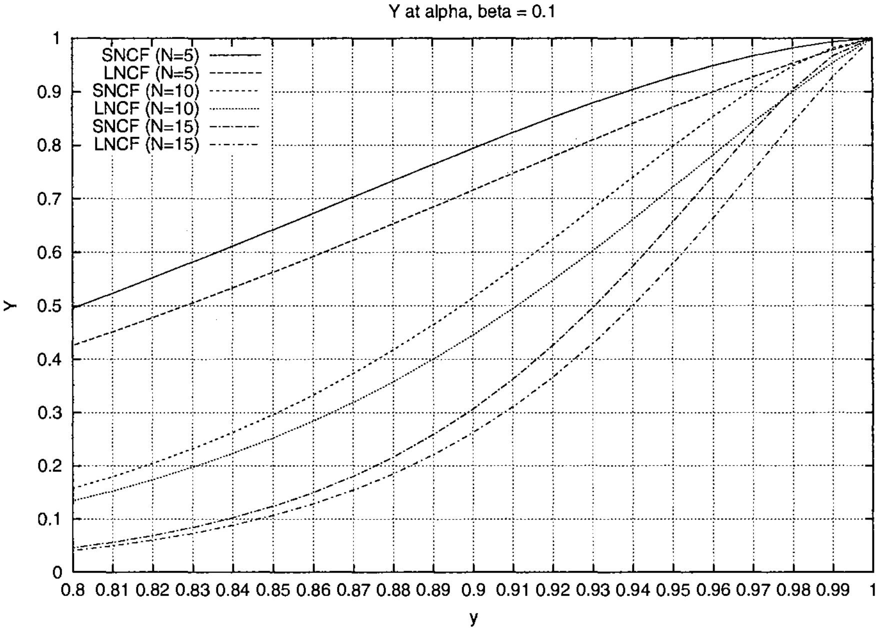

Figure 4.10: Yield of SNCF and LNCF at $\alpha, \beta = 0.1$

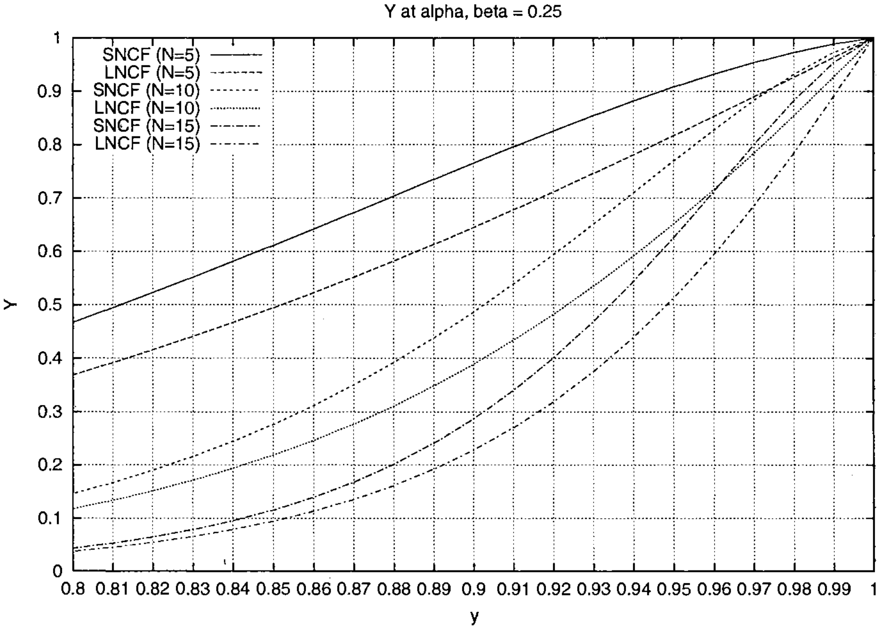

Figure 4.11: Yield of SNCF and LNCF at $\alpha, \beta = 0.25$

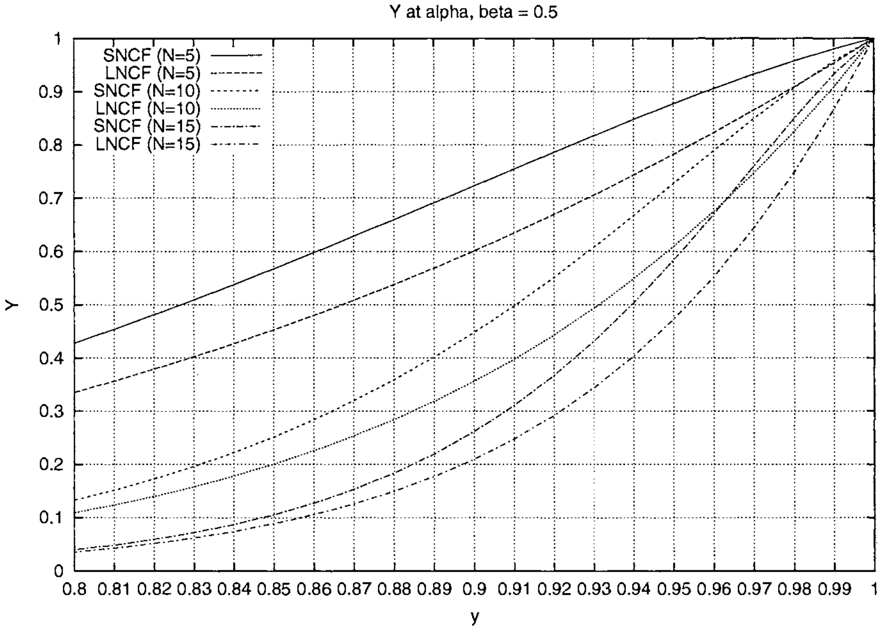

Figure 4.12: Yield of SNCF and LNCF at $\alpha, \beta = 0.5$

CHAPTER 5

HW/SW CO-RELIABILITY IN CONFIGURABLE MULTI-PROCESSOR-MEMORY-ON-CHIP

Advances in field reconfigurable (FC) [54, 55, 56, 57, 58, 59, 9, 23, 24, 25, 34] SoCs have made possible the highly flexible yet ultra parallel multi-processor-memory systems (MPMSs) design on a single chip [60, 61, 62]. As shown in Figure (5.1), a typical architectural model consists of processor/memory cores and an FC Interconnect bus [24] along with a dedicated reconfiguration and repair processor. Reliability is a commonly emphasized requirement of such systems since insignificant degradation of individual components could result in unacceptably low overall system reliability. Acceptable reliability for these component-based systems has been commonly achieved by redundancy utilization and repair processes. Local reconfiguration (repair) of memory arrays using spare memory lines is the most common technique for reliability enhancement of memories with fault, and modular redundancy utilization is one of the well-known techniques used for reliability enhancement of core-based systems. To accurately model the reliability of the MPMSs, a proper reliability model must be introduced. It is well-known that failures of system components can be numerically defined by the *failure rate* which is the expected number of failures per a given time period Δt. The exponential relationship between the reliability and time is also known as the *exponential failure law* which claims that for a constant failure rate, the component reliability changes exponentially as a function of time t.

Several research results regarding multiprocessor systems have been reported. In [49], the fault detection capability and the fault location capability for a multiprocessor system with spare processor cores were discussed and its results showed that these schemes utilize spare capacity more efficiently, thereby improving upon the fault detection and location capabilities of the system. The proposed task allocation method for a multiprocessor system in [51] was a simple yet effective method to allocate the tasks in multiprocessor systems for minimizing the interprocessor communication cost, subject to resource limitations defined by the system and designer. [46] introduced a multiprocessor system with a reconfigurable pipelined bus system for parallel matrix multiplication and reconfiguration algorithms. [48] proposed the dynamic reconfiguration network and a monitoring-at-transmission bus to support dynamic reconfiguration of an N-modular redundancy multiprocessor system in order to provide gracefully degradable operations. Automatic layout of reconfigurable subsystems for SoCs was presented in [4]. An integrated reconfigurable subsystem for data-parallel and computation-intensive applications was reported in [23]. A reconfigurable bus core architecture for SoC applications was also presented in [24]. Quantitative analysis on the impact of design and synthesis options on processor core performance was presented in [10]. In [47], an availability model for multiprocessor systems based on a Markov chain was proposed and validated. Availability and reliability of mesh-connected systems were discussed in [50]. In addition, another efficient technique for computing the reliability of k-to-l-out-of-n systems was also presented in [52] which can be used to estimate reliability of static multiprocessor systems. In [18], an optimal memory allocation method for an application-specific multiprocessor SoC was proposed and validated. Besides, [1] proposed a complete strategy for testing on-chip multiprocessor architecture. However, comprehensive research work on the co-effect of hardware and software on the reliability (referred to as *HW/SW Co-Reliability*) of the field reconfigurable multi-processor-memory SoCs has not yet been reported.

The objective of this chapter is to propose the HW/SW Co-reliability assurance technique based on the proposed general architectural model for field reconfigurable SoC-based multi-processor-memory systems. Ultimately, architecture-general, reliability-assured and cost-effective design and implementation of field reconfiguration for the multi-processor-memory SoCs will be realized by using the proposed methods.

The organization of the chapter is as follows: In the Section 5.1, review and preliminaries related to this research work and the generalized architectural model of a field reconfigurable multi-processor-memory SoC will be introduced and discussed. Introduction to HW/SW Co-reliability and its analysis technique based on the proposed architectural model will be given in Section 5.2. System utilization patterns demanded by software in the field are categorized in Section 5.3. Parametric simulations and their results are shown in Section 5.4. Then, discussion and conclusions are given.

5.1 Review and Preliminaries on Field Reconfigurable Multi-Processor-Memory SoCs

The following notations are used in this work:

- N : Number of total processor/memory cores.
- p : Number of processor cores per parallel computational unit.
- m : Number of memory cores per parallel computational unit.
- u : Number of total parallel computational units.

- (p, m, u) : 3-tuple of system configuration demanded by software.
- n : Number of rows (columns) in a memory core.
- s : number of spare columns embedded in each memory core.
- M_p : Number of unused processor cores which can be utilized as spares. $N - (i \cdot k)$.
- M_m : Number of unused memory cores which can be utilized as spares. $N - (j \cdot k)$.
- λ_p : Fault arrival rate for a processor core.
- λ_m : Fault arrival rate for a memory cell.
- t : time.
- Δt : Unit time interval.
- R : Reliability.

The generalized architectural model for field reconfigurable multi-processor-memory systems given in Figure (5.1) is considered in this chapter. Characteristics of the model can be summarized as follows:

- N FC (Field ReConfigurable) processors and N FC memories communicate via an FC interconnection bus.
- One parallel processing unit (PPU) consists of p processors and m $n \times n$ memories and dedicated reconfigurable interconnects. u noncoherent units are supposed to execute in parallel at the same time.

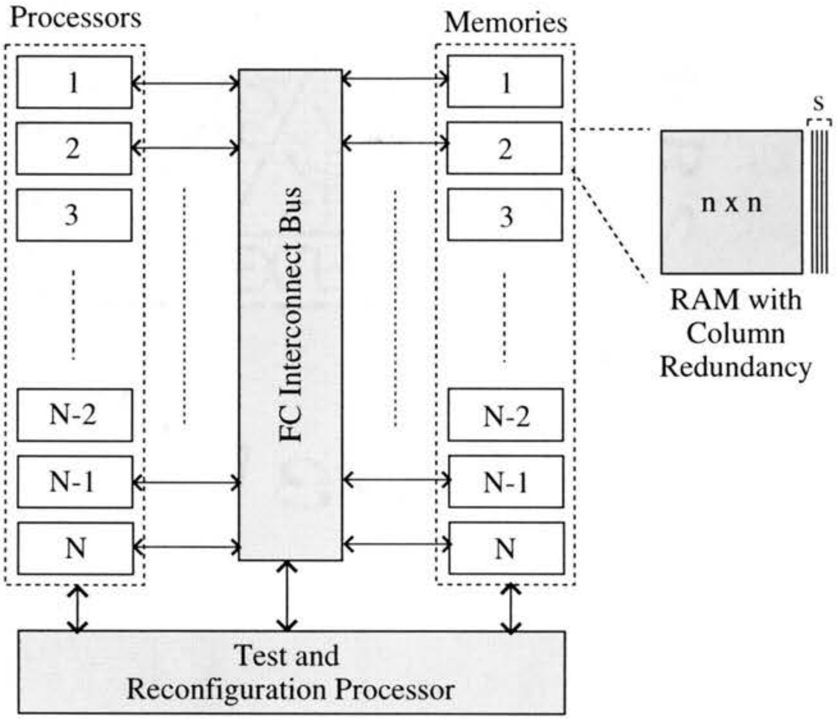
Processors
Memories
1
2
3
N-2
N-1
N
FC Interconnect Bus
s
n x n
RAM with Column Redundancy
Test and Reconfiguration Processor

Figure 5.1: Investigated Field Reconfigurable Multiprocessor SoC Model.

- 3-tuple of the hardware configuration (m, p, u) is demanded by software in the field.

- Faulty cells occurring in a memory can be locally tolerated by field-reconfiguration of the given spare column redundancy.

- Unused processor cores and memory cores can be field-reconfigured to replace faulty processor cores and memory cores in PPUs.

- Reliability of the FC interconnect bus is considered to be perfect since the work emphasizes the computation and memory access intensiveness of SWs.

- PPUs are noncoherent to each other, which means that tasks assigned to PPUs are mutually exclusive.

For instance, a $(1, 2, 2)$ 3-tuple configuration on an $N = 8$ system is shown in Figure (5.2) in which the first PPU consists of processor core 1 and memory cores 1 and 2 and a corresponding virtual interconnection network provided by the FC interconnect bus, and the second PPU uses processor core 2 and memory cores 3 and 4. Since processor cores 3-8 and memory cores 5-8 are unused, they can be used to replace upto six faulty processor cores and upto four faulty memory cores. Figure (5.3) further illustrates the repair process by utilization of modular redundancy. If processor core 2 and memory core 4 are diagnosed as faulty by the Test and Reconfiguration Processor (TRP), it dynamically reallocates processor core 3 and memory core 5 to tolerate such faulty cores. As a result, the system ends up with the configuration of PPU 1 and reconfigured PPU 2 and 5 redundant processor cores and 3 redundant memory cores. Processor 2 and memory core 4 are marked as faulty by TRP and will no longer be used.

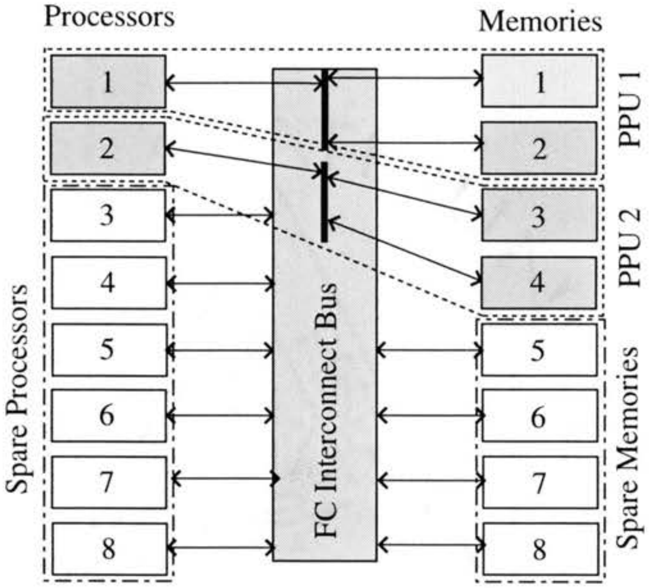

Figure 5.2: (1,2,2) System Configuration (2 1-processor 2-memory PPUs).

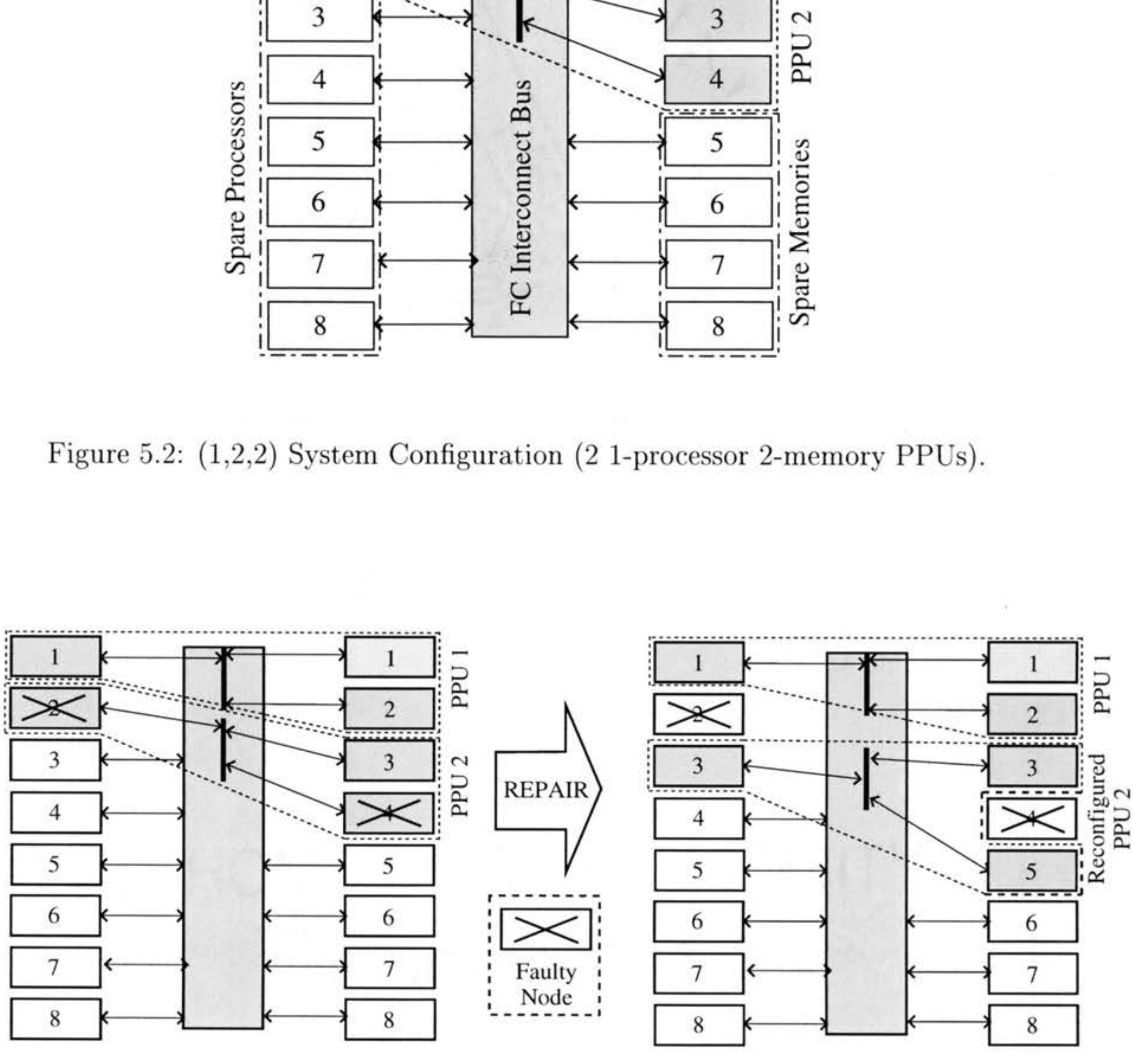

Figure 5.3: Repair of faulty processor and memory throughout modular redundancy reconfiguration.

5.2 HW/SW Co-reliability Analysis

The system reliability which is mainly determined by the 3-tuple hardware configuration demanded by software, and field reconfiguration and repair by utilizing unused processor and memory cores is referred to as *HW/SW Co-reliability.* HW/SW Co-reliability estimation and assurance technique are proposed in this section. Fault arrival rate for processor is λ_p. Then, the reliability of each processor core is determined by the exponential failure law

$$R_p(t) = e^{-\lambda_p t} \tag{5.1}$$

and the given fault arrival rate for the memory cell is λ_m. Thus, the reliability of a memory cell is

$$R_{cell}(t) = e^{-\lambda_m t} \tag{5.2}$$

and the reliability of each column of memory becomes

$$R_{column}(t) = (e^{-\lambda_m t})^n \tag{5.3}$$

Since each memory locally tolerates faulty columns up to s, the reliability of each fault-tolerant memory core is

$$R_m(t) = \sum_{i=0}^{s} \binom{n+s}{i} R_{column}(t)^{n+s-i} \cdot (1.0 - R_{column}(t))^i \tag{5.4}$$

In the field, the software demands 3-tuple reconfiguration of the given hardware: (p, m, u) where p is the number of processors, m is the number of memories in each PPU, and u is the number of PPUs. The reliability of a PPU without repair by modular redundancy can be written as

$$R_{PPU_{norepair}}(t) = R_p(t)^p \cdot R_m(t)^m \tag{5.5}$$

(1, 2, 2) system configuration of $N = 8$ system is given in Figure (5.2) as an example. Since six processors and four memories are not demanded by SW, they can be utilized for modular redundancy, and faulty processors and memories can be replaced by those spare cores. Thus, the reliability of a PPU with redundancy and repair is

$$\begin{aligned} R_{PPU}(t) \quad = \quad & \Big(\sum_{i=0}^{a} \binom{p+a}{i} R_p(t)^{p+a-i} \cdot (1.0 - R_p(t))^i\Big) \\ & \times\Big(\sum_{j=0}^{b} \binom{m+b}{j} R_m(t)^{m+b-j} \cdot (1.0 - R_m(t))^j\Big) \end{aligned} \tag{5.6}$$

where a and b are the number of dedicated spare processors and memories, respectively. The effect of modular redundancy utilization is visualized in Figure (5.4) where $p = 8$, $m = 8$, and different numbers of redundant processor and memory cores (i.e., a and $b = 0$, 2, 4, 6 and 8, respectively) are applied. As shown, even a relatively small amount of redundancy enhances reliability of the PPU gracefully: for example, the PPU with a, $b = 2$ can maintain its reliability well above 95% for $t = 1200$ while another PPU without modular redundancy barely does for $t = 100$.

Overall reliability of the system without considering modular redundancy and repair is a serial product of the reliability of processor and memory cores, such as

$$R_{norepair}(t) = R_p(t)^{(p \cdot u)} \cdot R_m(t)^{(m \cdot u)} \tag{5.7}$$

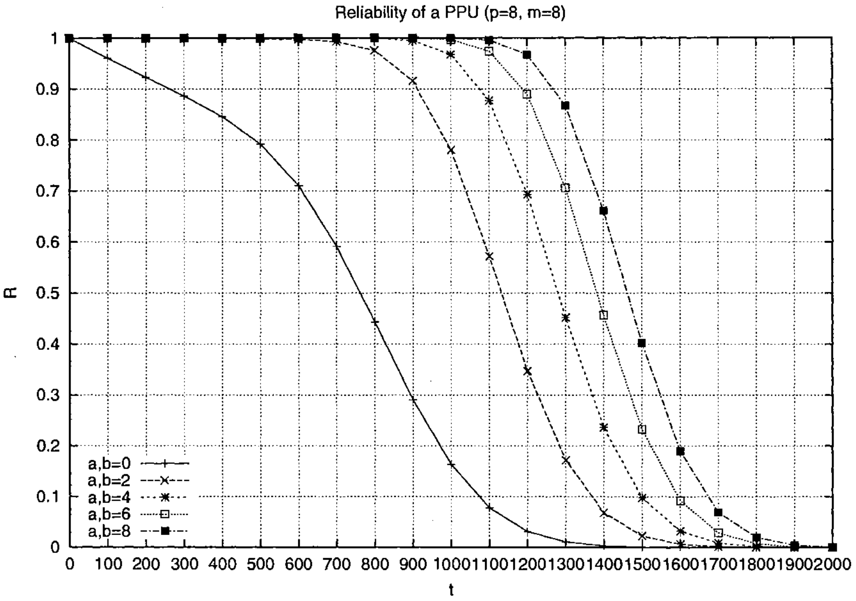

Figure 5.4: Reliability comparison of the PPU without modular redundancy and PPUs with different numbers of modular redundancy.

Finally, overall reliability of the system enhanced by modular redundancy can be expressed as

$$R(t) = \left(\sum_{i=0}^{M_p} \binom{N}{i} R_p(t)^{N-i} \cdot (1.0 - R_p(t))^i\right) \cdot \left(\sum_{j=0}^{M_m} \binom{N}{j} R_m(t)^{N-j} \cdot (1.0 - R_m(t))^j\right) \quad (5.8)$$

where M_p is the number of unused processor cores which can be utilized as spare processor cores, and M_m is the number of unused memory cores which can be used as spare memory cores. The proposed HW/SW Co-reliability estimation and assurance techniques are verified in Section 5.4.

5.3 System Utilization Categories of the Proposed System

Utilization of the proposed field reconfigurable multi-processor-memory system can be categorized in terms of relative PPU size as follows:

1. Coarse-Grained Utilization (CG) : If the PPU size is relatively large, such as a $(9, 9, u)$ system, the system can be categorized as CG. For the given number of processor/memory cores, N, a relatively smaller number of relatively larger PPUs can be allocated. Thus, the level of parallelism is low while the processing/storage capabilities of each PPU are high.

2. Medium-Grained Utilization (MG) : The intermediate category, such as a $(5, 5, u)$ system, between CG and FG. It has a balanced level of parallelism and processing/storage capabilities.

3. Fine-Grained Utilization (FG) : If the PPU size is relatively small, such as a $(2, 2, u)$ system, the system can be categorized as FG. For the given number of processor/memory cores, N, a relatively larger number of relatively smaller PPUs can be allocated. Thus, the level of parallelism is high while processing/storage capabilities of each PPU are low.

Utilization of the proposed system can be also categorized in terms of processor/memory utilization intensity as follows:

1. Processor Intensive Utilization (PI) : If $p > m$, then the system can be categorized as PI. Such PPUs have a higher processing capability and a lower storage capability.

2. Evenly Intensive Utilization (EI) : If $p = m$, then the system can be categorized as EI. Such PPUs have balanced processing and storage capabilities.

3. Memory Intensive Utilization (MI) : If $p < m$, then the system can be categorized as MI. Such PPUs have a lower processing capability and a higher storage capability.

Thus, combination of these six categories further yields nine possible utilization patterns: CGPI, CGEI, CGMI, MGPI, MGEI, MGMI, FGPI, FGEI, and FGMI. Sample 3-tuple configurations which represent those nine categories are shown in Table 5.4. Characteristics of the categories are investigated by extensive parametric simulations in Section 5.4.

5.4 Parametric Co-Reliability Analysis

In this section, the proposed HW/SW Co-reliability estimation and assurance technique is further verified by parametric simulation. Commonly used simulation parameters are shown in Table 5.4 (i.e., 64 processor/memory cores, 64 × 64 memory cores with 8 spare columns, the failure arrival rate for each processor is 5×10^{-4} and the failure arrival rate for each memory cell is 1.4×10^{-6}), and 3-tuple representative parameters for the proposed nine PPU categories are given in Table 5.4. Parametric simulation results are given in Figure (5.5) - (5.7) where HW/SW Co-reliability of both non-fault-tolerant and fault-tolerant systems are compared. For example, the CGEI system configuration enhanced by redundancy utilization can gracefully maintain above 95% reliability for $t = 700$ while the same system without repair hardly does for even $t = 100$. The following observations can be obtained from the results:

1. Field reconfiguration of the unused processor and memory cores to repair faulty PPU cores significantly enhances overall system reliability.

2. Since memory cores are locally fault-tolerant by spare columns, they show more

graceful degradation. Thus, processor intensive configurations (PIs), which yield more memory redundant cores, are more reliable than the other system configurations (i.e., PI is more reliable than EI, and EI is more reliable than MI).

3. Differences between CG, EG and FG in terms of overall system reliability do exist but not so significantly, which means M_p and M_m mainly determine the co-reliability. The only difference between them is the level of parallelism.

4. The proposed nine utilization patterns can be listed with respect to their co-reliability as follows: $MGPI > CGPI$, $FGPI > CGEI > MGMI > CGMI$, $FGMI > FGEI$, $MGEI$.

5. HW/SW Co-reliability estimation technique is an effective and accurate method to assure the system reliability. This technique can be used to optimize M_p, M_m and s for the demanded 3-tuple system configuration by the SW.

Table 5.1: Common simulation parameters

Parameter	N	n	s	λ_p	λ_m
Value	64	64	8	5×10^{-4}	1.4×10^{-6}

5.5 Discussion and Conclusions

As advances in field reconfigurable system core technology makes possible the massively parallel multi-processor-memory SoCs, high system reliability becomes a commonly emphasized requirement of such systems since insignificant degradation of some cores could result in unacceptably low overall system reliability. Thus, HW/SW Co-

Table 5.2: 3-tuple simulation parameters

Category	3-tuple
$CGPI$	$(12, 9, 5)$
$CGEI$	$(9, 9, 6)$
$CGMI$	$(9, 12, 5)$
$MGPI$	$(7, 5, 8)$
$MGEI$	$(5, 5, 12)$
$MGMI$	$(5, 7, 8)$
$FGPI$	$(3, 2, 20)$
$FGEI$	$(2, 2, 30)$
$FGMI$	$(2, 3, 20)$

reliability measurement and estimation for the field reconfigurable multi-processor-memory SoCs using a combinatorial modeling method has been proposed and validated through the parametric simulations in this chapter. System configuration patterns demanded by SW can be categorized by the proposed nine representative patterns. Analysis of the nine configuration patterns by using the proposed HW/SW Co-reliability assurance technique reveals that 1) field reconfiguration of the unused processor and memory cores to repair faulty PPU cores significantly enhances overall system reliability and 2) processor/memory intensity is more significant than PPU size and 3) M_p and M_m mainly determines the system reliability. Ultimately, the HW/SW Co-reliability estimation technique can provide an effective and accurate means to assure reasonable system reliability while avoiding unnecessary cost due to larger N. This means that this technique can be used to optimize M_p, M_m and s for the demanded 3-tuple system configuration by a software CAD tool.

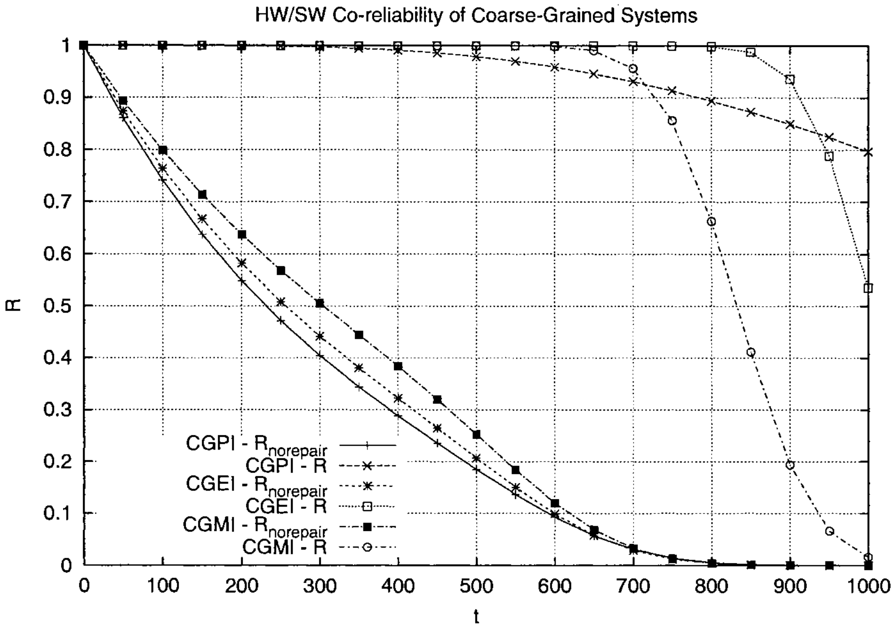

Figure 5.5: HW/SW Co-reliability of the Coarse-Grained Systems.

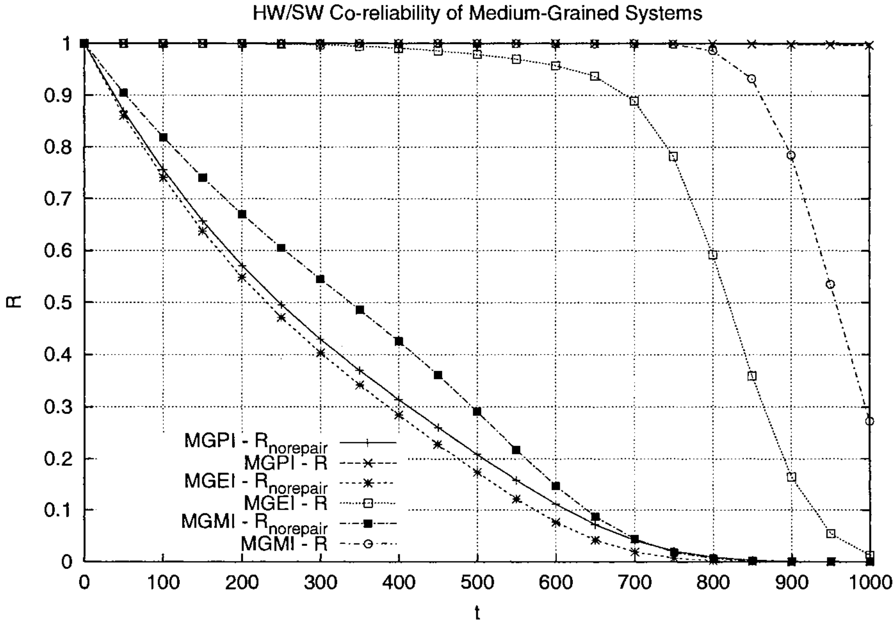

Figure 5.6: HW/SW Co-reliability of the Medium-Grained Systems.

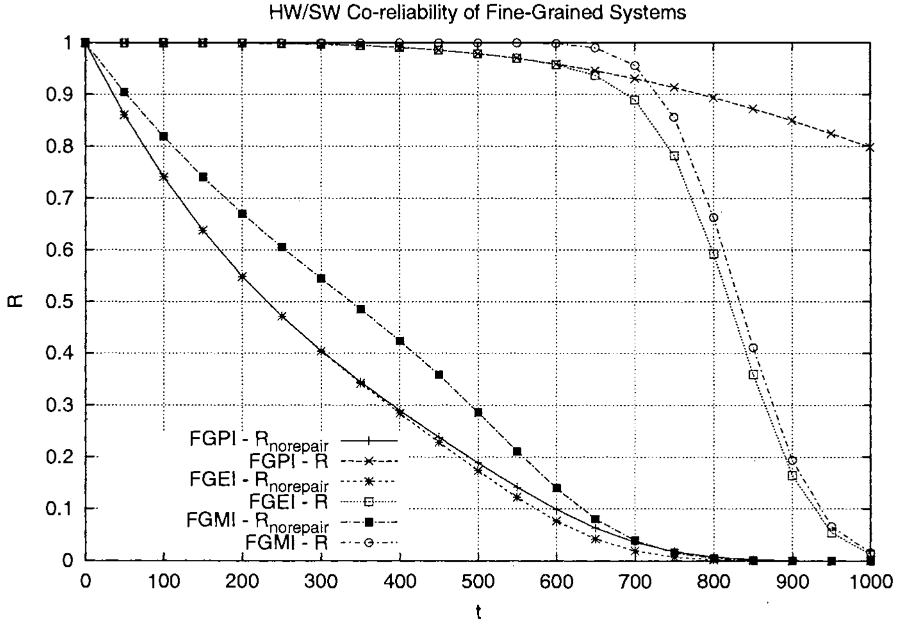

Figure 5.7: HW/SW Co-reliability of the Fine-Grained Systems.

CHAPTER 6

CONCLUSION

In this dissertation, an extensive literature review on the SoC technology and its reliability assurance techniques have been conducted and a number of new issues associated with the SoC technology have been addressed and investigated. Fabricated SoCs must be manufactured in large quantities at costs that are competitive with alternative methods of achieving the circuit and system functions. Hence, the expected yield of good SoCs must be assured to guarantee the cost-effectiveness. Also, the cores and supporting circuitry must perform their function throughout their intended useful lifetime. Although the SoC technology can provide a number of advantages such as low-power consumption, smaller area requirement and faster time-to-market, test/diagnosis/repair of SoCs are not as conventional as in legacy packaging technologies such as PCB and MCM-based systems, because of SoC's new and complex core-based design and monolithic structure. Thus, the reliability assurance, and defect and fault tolerance for reliability enhancement are two of the most imperative research topics for the success of the SoC technology. Four specific research areas have been presented in this dissertation to demonstrate the significance of the SoC reliability assurance and reliability-driven design by using defect and fault tolerance techniques.

A reliable memory core organization and its yield and reliability assurance techniques have been discussed in Chapter 2. The proposed memory core is designed to tolerate cell-based defects and faults by using the proposed hierarchical redundancy, fabrication-time repairability, and field repairability by BIST/BISD/BISR circuitry.

Figures (2.5)-(2.12) show parametric simulation results based on the proposed reliability assurance techniques that are developed for the proposed embedded memory core. The yield assurance results of a sample 1Mbit memory core and its components are shown in Figure (2.5) and the reliability assurance results are given in Figures (2.11) and (2.12). The significance of the reliability assurance, the reliability-driven design and the implementation of defect and fault tolerance has been clearly addressed and justified.

It also has been observed that the given redundancy can be properly balanced between fabrication-time repair and field-time repair to maximize the overall dependability ($D(t) = Y \cdot R(t)$) of the proposed embedded memory core to avoid improper utilization of the redundancy which may result in unnecessarily low dependability $D(t)$ in Chapter 3. The problem of finding the most dependability maximizing balancing is referred to as *optimal redundancy balancing.* Since $D(t)$ is a function of Y and $R(t)$, the yield and reliability assurance techniques proposed in the chapter can be effectively utilized to find the optimal redundancy balancing. The single dimensional redundancy case has been studied and extended to a two dimensional redundancy case. Balanced redundancy partitioning techniques based on the combinatorial modeling and a differential equation have been proposed. The parametric simulation results shown in Figures (3.3)-(3.18) demonstrate that the proposed redundancy balancing technique accurately finds the optimal redundancy partitioning positions for the ultimate combination of the factory yield and field reliability.

Chapter 4 has presented a new model for analyzing the yield of RSoC systems with repair processes, based on the effect of connectivity of the repair-candidate core where reliability degradation of both neighboring cores and interconnect structure is taken into account. Two approaches, *Smallest Number of Interconnects First (SNIF)* and *Smallest Number of Neighboring Cores First (SNCF)* have been proposed. Two other schedulings, *Largest Number of Interconnects First (LNIF)* and *Largest Number of Neighboring Cores First (LNCF)* have also been introduced and analyzed, and it has been shown how an improperly scheduled repair process could impair the yield of RSoCs. Extensive parametric analysis and comparison of the proposed approaches have demonstrated the effectiveness of the proposed RSoC repair scheduling strategies (i.e., $SNIF$ and $SNCF$). The reliability comparison results of SNIF and LNIF at $\alpha, \beta = 0.25$ are shown in Figures (4.5)-(4.8) and the similar results for SNCF and LNCF are given in Figures (4.9)-(4.12). From the results, it is obvious that the higher yield of an RSoC can be exploited when the proper RSoC repair scheduling strategies such as $SNIF$ and $SNCF$ are applied. It can be seen that $SNIF$ tolerates higher core and interconnect degradation due to the repair process (i.e., higher α and β) than $SNCF$ does. However, $SNCF$s show higher yield when lower core and interconnect degradation due to the repair process (i.e., lower α and β) are considered.

In Chapter 5, HW/SW Co-reliability measurement and estimation techniques for the field reconfigurable SoC-based multi-processor-memory systems by using a combinatorial modeling method has been proposed and validated through the parametric simulations. System configuration patterns demanded by SW can be categorized by the proposed nine representative patterns. Analysis of the nine configuration patterns by using the proposed HW/SW Co-reliability assurance technique reveals that 1) field reconfiguration of the unused processor and memory cores to repair faulty PPU cores significantly enhances overall system reliability, 2) processor/memory intensity is more significant than PPU size, and 3) M_p and M_m mainly determines the system reliability. Therefore, the HW/SW Co-reliability estimation technique is an effective and accurate method to optimize M_p, M_m, and s for a 3-tuple system configuration demanded by SW.

REFERENCES

[1] C. Aktouf, "A complete strategy for testing an on-chip multiprocessor architecture," *IEEE Design & Test of Computers*, vol. 19, Issue 1, pp. 18-28, Jan.-Feb. 2002.

[2] P. Magarshack, "Improving SoC design quality through a reproducible design flow," *IEEE Design & Test of Computers*, vol. 19, Issue 1, pp. 76-83, Jan.-Feb. 2002.

[3] A. Nalamalpu, S. Srinivasan and W.P. Burleson, "Boosters for driving long onchip interconnects - design issues, interconnect synthesis, and comparison with repeaters," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, Issue 1, pp. 50-62, Jan. 2002.

[4] S. Phillips and S. Hauck, "Automatic layout of domain-specific reconfigurable subsystems for system-on-a-chip," *Tenth ACM International Symposium on Field-Programmable Gate Arrays*, pp. 165-173, 2002.

[5] A.D. Pimentel, L.O. Hertzbetger, P. Lieverse, P. van der Wolf and E.E. Deprettere, "Exploring embedded-systems architectures with Artemis," *IEEE Computer*, Vol. 34, Issue 11, pp. 57-63, Nov. 2001.

[6] W. Wang and K. Lee and J. Wang "An on-chip march pattern generator for testing embedded memory cores," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 9, Issue 5, pp. 730-735, Oct. 2001.

[7] H. Kim and J.P. Hayes, "Delay fault testing of IP-based designs via symbolic path modeling," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 9, Issue 5, pp. 661-678, Oct. 2001.

[8] R.A. Bergamaschi, S. Bhattacharya, R. Wagner, C. Fellenz, M. Muhlada, F. White, J.-M. Daveau and W.R. Lee, "Automating the design of SoCs using cores," *IEEE Design & Test of Computers*, Vol. 18, Issue 5, pp. 32-45, Sep.-Oct. 2001.

[9] B.I. Hounsell and T. Arslan, "Programmable multiplierless digital filter array for embedded SoC applications," *Electronics Letters*, pp. 735-737, Jun. 2001.

[10] T. Bautista and A. Nunez, "Quantitative study of the impact of design and synthesis options on processor core performance," *19th IEEE Proceedings on VLSI Test Symposium*, pp. 169-175, Apr.-May 2001.

[11] X. Bai and S. Dey, "High-level crosstalk defect simulation for system-on-chip interconnects," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 9, Issue 3, pp. 461-473, Jun. 2001.

[12] W.-C. Lai, K.-T. Cheng, "Instruction-level DFT for testing processor and IP cores in system-on-a-chip," *Proceedings on 2001 Design Automation Conference*, pp. 59-64, Jun. 2001.

[13] Shang-yi Chiang, "Foundries and the dawn of an open IP era," *IEEE Computer*, Vol. 34, Issue 4, pp. 43-46, Apr. 2001.

[14] A. Chandra, K. Chakrabarty, "System-on-a-chip test-data compression and decompression architectures based on Golomb codes," *IEEE Transactions on*

Computer-Aided Design of Integrated Circuits and Systems, Vol. 20, Issue 3, pp. 355-368, Mar. 2001.

[15] Li Chen, S. Dey, "Software-based self-testing methodology for processor cores," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 20, Issue 3, pp. 369-380, Mar. 2001.

[16] S. Ravi, G. Lakshminarayana, N.K. Jha, "Testing of core-based systems-on-a-chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 20, Issue 3, pp. 426-439, Mar. 2001.

[17] T. Yamauchi, M. Kinoshita, T. Amano, K. Dosaka, K. Arimoto, H. Ozaki, M. Yamada and T. Yoshihara, "Design methodology of embedded DRAM with virtual-socket architecture," *IEEE Journal of Solid-State Circuits*, Vol. 36, Issue 1, pp. 46-54, Jan. 2001.

[18] S. Meftali, F. Gharsalli, F. Rousseau, A.A. Jerraya "An optimal memory allocation for application-specific multiprocessor system-on-chip," *Proceedings of the 2001 international symposium on Systems synthesis*, pp. 19-24, 2001.

[19] J.-H. Hong, C.-H. Tsai and C.-W. Wu, "Hierarchical system test by an IEEE 1149.5 MTM-bus slave-module interface core," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 8, Issue 5, pp. 503-516, Oct. 2000.

[20] F.J. Meyer, N. Park, "Predicting the yield efficacy of a defect-tolerant embedded core," *Proceedings on 2000 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 30-38, Oct. 2000.

[21] S. Dey, D. Panigrahi, Li Chen, C.N. Taylor, K. Sekar and P. Sanchez, "Using a soft core in a SoC design: experiences with picoJava," *IEEE Design & Test of Computers*, Vol. 17, Issue 3, pp. 60-71, Jul.-Sep. 2000.

[22] I. Ghosh, S. Dey and N.K. Jha, "A fast and low-cost testing technique for core-based system-chips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 19, Issue 8, pp. 863-877, Aug. 2000.

[23] H. Singh, M.-H. Lee, G. Lu, F.J. Kurdahi, N. Bagherzadeh and E.M. Chaves Filho, "MorphoSys: an integrated reconfigurable system for data-parallel and computation-intensive applications," *IEEE Transactions on Computers*, Vol. 49, Issue 5, pp. 465-481, May 2000.

[24] S. Winegarden, "Bus architecture of a system on a chip with user-configurable system logic," *IEEE Journal of Solid-State Circuits*, vol. 35, Issue 3, pp. 425-433, May 1999.

[25] S. Knapp and D. Tavana, "Field configurable system-on-chip device architecture," *Proceedings of the 2000 IEEE Custom Integrated Circuits Conference*, pp. 155-158, May 2000.

[26] Y. Zorian, E.J. Marinissen and S. Dey, "Testing Embedded-Core-Based System Chips," *IEEE Computer*, Vol.32, Issue 6, Jun. 1999

[27] D. Kirovski, C. Lee, M. Potkonjak and W.H. Mangione-Smith, "Application-driven synthesis of memory-intensive systems-on-chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 18, Issue 9, pp. 104-109, Sep. 1999.

[28] E.J. Marinissen and Y. Zorian, "Challenges in testing core-based system ICs," *IEEE Communications Magazine*, Vol. 37, Issue 6, pp. 104-109, Jun. 1999.

[29] C.-T. Huang, J.-R. Huang, C.-F. Wu, C.-W. Wu and T.-Y. Chang, "A programmable BIST core for embedded DRAM," *IEEE Design & Test of Computers*, Vol. 16, Issue 1, pp. 59-70, Jan.-Mar. 1999.

[30] A.M. Rincon, C. Cherochetti, J.A. Monzel, D.R. Stauffer and M.T. Trick, "Core Design and System-on-a-Chip Integration", *IEEE Design & Test of Computers*, Vol. 14, Issue 4, Oct.-Dec. 1997

[31] R.K. Gupta and Y. Zorian, "Introducing core-based system design," *IEEE Design & Test of Computers*, Vol. 14, Issue 4, pp. 15-25, Oct.-Dec. 1999.

[32] T. Tsuruda, M. Kobayashi, M. Tsukude, T. Yamagata, K. Arimoto and M. Yamada, "High-speed/high-bandwidth design methodologies for on-chip DRAM core multimedia system LSI's," *IEEE Journal of Solid-State Circuits*, Vol. 32, Issue 3, pp. 477-482, Mar. 1997.

[33] T. Lee and J. Cong, "The New Line in IC Design", *IEEE Spectrum*, Vol. 34, Issue 3, Mar. 1997

[34] A. Mohsen, "Programmable Interconnects Speed System Verification", *IEEE Circuits and Devices Magazine*, Vol. 9, Issue 3, May. 1993

[35] J. Ohtani, T. Ooishi, et al., "A Shared Built-In Self-Repair Analysis for Multiple Embedded Memories," *2001 IEEE Conference on Custom Integrated Circuits*, pp. 187-190, May 2001.

[36] R.J. McPartland, D.J. Loeper et al, "SRAM Embedded Memory with Low Cost, FLASH EEPROM-Switch-Controlled Redundancy," *Custom Integrated Circuits*

Conference, 2000. CICC. Proceedings of the IEEE 2000 , Vol. 36, No. 11, pp. 287-289, May 2000.

[37] T. Kawagoe, J. Ohtani, et al., "A Built-In Self-Repair Analyzer (CRESTA) for Embedded DRAMs," *Test Conference, 2000. Proceedings. International*, Vol. 41, No. 9, pp. 567-574, Oct. 2000.

[38] International Technology Roadmap for Semiconductors. "International Technology Roadmap for Semiconductors 2000," *http://public.itrs.net/Files/2000UpdateFinal/2kUdFinal.htm*, 2000.

[39] D.K. Bhavsar, "An Algorithm for Row-Column Self-Repair of RAMs and its Implementation in the Alpha 21264," *Test Conference, 1999. Proceedings. International*, pp. 311-318, Sep. 1999.

[40] S.-K. Lu, S.-Y. Kuo, C.-W. Wu, "Fault-Tolerant Interleaved Memory Systems with Two-Level Redundancy," *IEEE Transactions on Computers*, Vol. 46, Issue 9, pp. 1028-1034, Sep. 1997.

[41] C.P. Low and H.W. Leong, "A New Class of Efficient Algorithms for Reconfiguration of Memory Arrays," *IEEE Transactions on Computers*, Vol 45, No 5, pp. 614-618, 1996.

[42] D. M. Blough, "Performance Evaluation of a Reconfiguration-Alogorithm for Memory Arrays containing Clustered Faults," *IEEE Transactions on Reliability*, Vol. 45, No. 2, pp. 274-284 June 1996.

[43] C.H. Stapper, H.-S. Lee, "Synergistic Fault-Tolerence for Memory Chips," *IEEE Transactions on Computers*, Vol. 41, No. 9, pp. 1078-1087, September 1992.

[44] E. Fujiwara, D.K. Pradhan, "Error-Control Coding in Computers," *IEEE Computer*, Vol. 23, Issue 7, pp. 63-72, Jul. 1990.

[45] S.Y. Kuo and W.K. Fucks, "Efficient Spare Allocation in Reconfigurable Arrays," *IEEE Design and Test*, Vol. 41, Issue 9, pp. 24-31, Feb. 1987.

[46] K.Q. Li and V.Y. Pan, "Parallel matrix multiplication on a linear array with a reconfigurable pipelined bus system," *IEEE Transactions on Computers*, Vol. 50 Issue 5, pp. 519-525, May 2001.

[47] G. Rubino and B. Sericola, "Interval availability analysis using denumerable Markov processes: application to multiprocessor subject to breakdowns and repair," *IEEE Transactions on Computers*, Vol. 44 Issue 2, pp. 286-291, Feb. 1995.

[48] J.C. Liu and K.G. Shin, "Efficient implementation techniques for gracefully degradable multiprocessor systems," *IEEE Transactions on Computers*, Vol. 44 Issue 4, pp. 503-517, Apr. 1995.

[49] S. Tridandapani, A.K. Somani and U.R. Sandadi, "Low overhead multiprocessor allocation strategies exploiting system spare capacity for fault detection and location," *IEEE Transactions on Computers*, Vol. 44 Issue 7, pp. 865-877, Jul. 1995.

[50] P. Mohapatra and C.R. Das, "On dependability evaluation of mesh-connected processors," *IEEE Transactions on Computers*, Vol. 44 Issue 9, pp. 1073-1084, Sep. 1995.

[51] C.-I.H. Chen and V. Cherkassky, "Task allocation and reallocation for fault tolerance in multicomputer systems," *IEEE Transactions onAerospace and Electronic Systems*, Vol. 30 Issue 4, pp. 1094-1104, Oct. 1994.

[52] S.J. Upadhyaya and H. Pham, "Analysis of noncoherent systems and an architecture for the computation of the system reliability," *IEEE Transactions on Computers*, Vol. 42 Issue 4, pp. 484-493, Apr. 1993.

[53] L. Benini and G. De Micheli, "Networks on chips: a new SoC paradigm," *IEEE Computer*, Vol. 35 Issue 1, pp. 70-78, Jan. 2002.

[54] J. Greenbaum, "Reconfigurable logic in SoC systems," *Proceedings of the IEEE 2002 Custom Integrated Circuits Conference*, pp. 5-8, May. 2002.

[55] S. Lee, S. Yoo and K. Choi, "Reconfigurable SoC design with hierarchical FSM and synchronous dataflow model," *Proceedings of the Tenth International Symposium on Hardware/Software Codesign, 2002.*, pp. 199-204, May. 2002.

[56] B. Lewis, I. Bolsens, R. Lauwereins, C. Wheddon, B. Gupta and Y. Tanurhan, "Reconfigurable SoC–what will it look like," *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, 2002.*, pp. 660-662, Mar. 2002.

[57] R. Hartenstein, "Reconfigurable computing: a new business model-and its impact on SoC design," *Proceedings. Euromicro Symposium on Digital Systems Design, 2001.*, pp. 103-110, Sep. 2001.

[58] Z. Huang and S. Malik, "Managing dynamic reconfiguration overhead in systems-on-a-chip design using reconfigurable datapaths and optimized interconnection

networks," *Proceedings of Design, Automation and Test in Europe Conference and Exhibition 2001.*, pp. 735-740, Mar. 2001.

[59] S.J.E. Wilton and R. Saleh, "Programmable logic IP cores in SoC design: opportunities and challenges," *IEEE Conference on Custom Integrated Circuits, 2001.*, pp. 63-66, May. 2001.

[60] P. Gerin, S. Yoo, G. Nicolescu and A.A. Jerraya, "Scalable and flexible cosimulation of SoC designs with heterogeneous multi-processor target architectures," *Proceedings of the Asia and South Pacific Design Automation Conference, 2001.*, pp. 63-68, Feb. 2001.

[61] S. Yoo, G. Nicolescu, D. Lyonnard, A. Baghdadi and A.A Jerraya, "A generic wrapper architecture for multi-processor SoC cosimulation and design," *Proceedings of the Ninth International Symposium on Hardware/Software Codesign, 2001.*, pp. 195-200, Apr. 2001.

[62] B. Clement, R. Hersemeule, E. Lantreibecq, B. Ramanadin, P. Coulomb and F. Pogodalla, "Fast prototyping: a system design flow applied to a complex System-On-Chip multiprocessor design," *Proceedings of the Design Automation Conference, 1999.*, pp. 420-424, Jun. 1999.

VITA 2

Minsu Choi

Candidate for the Degree of

Doctor of Philosophy

Dissertation: SYSTEM-ON-CHIP DESIGN FOR RELIABILITY

Major Field: Computer Science

Biographical:

Personal Data: Born in Seoul, Korea, on September 26, 1970, the son of Jinsung and Daeja Choi.

Education: Graduated from Daesung High School, Seoul, Korea in January 1989; received Bachelor of Science degree and Master of Science degree in Computer Science from Oklahoma State University, Stillwater, Oklahoma in May 1995 and May 1998, respectively. Completed the requirements for the Doctor of Philosophy degree at Oklahoma State University in August 2002.

Experience: Employed as a graduate teaching assistant; Department of Computer Science, Oklahoma State University, 1995 to present.

Professional Memberships: IEEE, Golden Key National Honor Society.