Yield and Reliability Enhancement for 3D-Stacked ICs

JIANG, Li

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong
September 2013

Thesis/Assessment Committee

Professor LEE Kin Hong (Chair)

Professor XU Qiang (Thesis Supervisor)

Professor WU Yu Liang (Committee Member)

Professor Lee Hsien-hsin (External Examiner)

Abstract

Three-dimensional (3D) stacked integrated circuits (3D-SICs) that stack multiple dies vertically using through-silicon vias (TSVs) have gained tremendous momentum for the semiconductor industry adoption recently. 3D-SICs promise "More-than-Moore" integration by packing more functionalities into a single chip, but the disruptive manufacturing technologies, e.g., die/wafer stacking and TSV formation, inevitably introduce new kinds of manufacturing defects ($t = 0$) and failure mechanisms into the circuit. To be specific, in the die/wafer stacking process, it is likely for a bad die to be stacked with a good die, causing significant stack yield loss. The assembly process, including TSV fabrication, alignment and etc., introduces various types of defects that lead to dramatic assembly yield loss. In addition, electromigration (EM) (one of critical challenges for reliability of metal wires in nano-scale circuit) indeed occurs in TSVs, and even gets exacerbated with the thermal-mechanical stress generated in TSV fabrication process. This wear-out mechanism of TSVs results in accelerated chip failure in the field. No doubt to say, these issues have to be resolved first before TSV-based 3D ICs become commercially viable.

To tackle the above problems, in this thesis, we propose a set of systematic approaches to enhance the yield and reliability for 3D-SICs. We presented the first analytical model for 3D memory circuit that captures the impact of TSV open defects through extensive simulation study, which serves as the preparation step

摘要：

三維堆疊集成電路（3D-stacked ICs）是通過尺寸極短的過矽穿孔 (through-silicon vias)（TSV）把多個晶片堆疊在一起從而行程一個完整芯片。進來，這種集成電路已經逐漸開始得到主流工業界的認可。三維堆疊集成電路可以在一個芯片中集成更多的功能，因此被認為是最有可能成為代表後摩爾定律（More-than-Moore）的技術。但是，三維堆疊芯片的制造工藝不但復雜，而且有一定的破壞性（比如，晶片堆疊過程和 TSV 制造）。這就不可避免的在電路中引入嚴重的製造缺陷以及一些潛在的，暫時無法發現的缺陷。具體的說，在晶片堆疊過程中，一個故障晶片會和一個無故障晶片疊在一起，因此導致嚴重的堆疊良率（stack yield）損失。另一方面，芯片的裝配（assembly）過程（包括 TSV 制造和對齊等）也會引入各種缺陷，從而導致整個三維堆疊集成電路的裝配良率（assembly yield）嚴重下降。而且，電遷移（electromigration）（ 對納米級電路中金屬導線的可靠性造成最大挑戰）也會發生在 TSV 中。更糟的是，製造過程中產生的熱機械應力（thermal-mechanical stress）會惡化 TSV 本身的電遷移過程。這會導致 TSV 加速老化並且試三維堆疊集成電路在運行過程中的失效率大大增加。毫無疑問，只有解決了這些問題，三維堆疊集成電路才有足夠可觀的商業利益。

為了解決這些問題，本論文提出了一套系統性的解決方案來提高三維堆疊芯片的良品率和可靠性。我們通過廣泛的仿真實驗率先提出了一個分析模型來捕捉在三維堆疊存儲器中由於 TSV 斷路而造成的故障模式。這一分析模型作為未來對三維堆疊芯片進行有效的測試和維修奠定了基礎。然後，我們率先提出了三維堆疊片上系統（3D SoC）的測試架構以及相應的優化算法，來盡可能的降低堆疊前測試（pre-bond test）和堆疊後測試（post-bond test）的成本。為了進行堆疊前測試，我們必須要加入額外的測試管腳。而這些測試管腳會占用過多的空間。因此，我們接下來重新優化三維片上系統的測試架構，使其可以在有限的測試管腳下盡可能的降低測試成本。以上這些故障模型和測試方案的提出，可以大大降低故障晶片被堆疊的可能性，從而提高了堆疊良率。不過，一種更有吸引力的做法是在完成堆疊後對那些故障晶片（可能是沒有被堆疊前測試所檢測到）進行修補。但是，只有那些自身帶有冗余資源的產品才能被修復（比如三維存儲器）。我們將這一新穎的概念引入三維存儲器中，並提出了一種迭代的晶片匹配算法來盡可能的提高其堆疊良率。為了提高裝配良率，我們提出了一種新穎的，基於 TSV 陣列的冗余架構以及對應的修復算法。這個方案可以修復 TSV 中的故障（對那種聚集在一起的故障尤其有效），從而極大的提高裝配良率。最後，我們提出了一種可重構的在線 TSV 修復方案。通過巧妙的使用 TSV 冗余資源，它可以有效的解決 TSV 中潛在的故障。這個修復方案中的可重構的冗余 TSV 架構可以讓臨近的 TSV 陣列共享冗余 TSV 資源，而其修復算法則可以支持在線修復。

for further test and repair. We then presented the first 3D SoC test architecture design and optimization technique that takes both pre-bond testing and post-bond testing into consideration. As dedicated test pads introduced for pre-bond testing incur quite high overhead to the circuit, this constraint was considered in our later work in this thesis. With above fault modeling and test solutions, we enhance the stack yield of 3D-SICs by preventing faulty dies from being stacked. A more attractive way is to be able to remedy the bad dies after bonding, and this is only possible for those products with inherent redundancies (e.g., 3D-DRAMs). We introduce this novel concept into 3D DRAMs and presented an iterative die matching strategy that can dramatically enhance the stack yield. In order to enhance the assembly yield, we proposed novel TSV-grid based redundancy architecture and the corresponding repair algorithms to tolerate the manufacturing defects (especially clustered defects) in TSVs. Finally, we describe a reconfigurable in-field repair solution that is able to effectively tolerate latent TSV defects through the judicious use of spares. The proposed solution includes a reconfigurable repair architecture that enables spare TSV sharing between TSV grids, and the corresponding in-field repair algorithms.

Acknowledgement

At the very beginning, I am deeply indebted to my supervisor, Professor Qiang Xu, who patiently motivated me to conceive and develop the main ideas in the thesis and taught me so much in my research. I couldn't achieve these academic works without your supervision. I would like to express to him my sincere gratitude for his seasoned guidance from the very early stage of this research work as well as providing constructive advices throughout the entire study. In particular, I also would like to thank him and his wife for their concerns about my daily life.

Professor YOUNG Fung Yu and WU Yu Liang, thank you for your help and suggestion in my research work. I shall extend my thanks to Prof. Krishnendu Chakrabarty from Duke University for all his kindness and help.

My research partners Yubin Zhang, Feng Yuan, Lin Huang, Xiao Liu, Yuxi Liu, Rong Ye, and Jie Zhang in the The CUhk REliable Computing Laboratory (CURE), thank you for your insightful comments on my research work. I am also grateful to all the colleagues in 506 EDA office, Linfu Xiao, Liang Li, Minqi Jiang, Zaichen Qian, Xiaoqing Yang, Zigang Xiao and Yan Jiang, it is you who bring me a colorful post-graduate study life. Special thanks are also to my friends Lei Shi and my roommate Yongkun Li who helped me a lot in my daily life.

Last but not the least, my mother Yunxia Jiang, my father Caiwang Liu, my wife Yeyun Zhu and all my family members, without your love and support, I cannot achieve anything. I would like to give my greatest appreciation to you.

Contents

Abstract **i**

Acknowledgement **iii**

1 Introduction **1**

1.1 Three-Dimensional Stacked Integrated Circuits 2

1.1.1 Background of 3D-SIC 2

1.1.2 Manufacturing Process 3

1.1.3 Prospects of 3D-SICs 5

1.2 Design for Pre-bond Test 7

1.3 Yield and Reliability Challenges for 3D-SICs 9

1.3.1 Yield and Reliability Crisis in 3D-SICs 10

1.3.2 Yield Modeling 11

1.4 Thesis Motivation and Organization 15

2 Modeling TSV Open Defects in 3D DRAM **18**

2.1 Introduction 18

2.2 Preliminaries 20

2.2.1 Memory Operation 20

2.2.2 Related Work and Motivation 22

2.3 Simulation Methodology 23

2.4 Simulation for Wordline Opens 24
2.4.1 Simulation Setup 24
2.4.2 Simulation Results and Analysis for Write Operation . . . 26
2.4.3 Simulation Results and Analysis for Read Operation . . . 27
2.5 Simulation for Bitline Opens 29
2.5.1 Simulation Setup for Bitline Opens 29
2.5.2 Simulation Results and Analysis for Write Operation . . . 31
2.5.3 Simulation Results and Analysis for Read Operation . . . 32
2.6 Simulation with TSV Coupling from Multiple Layers 33
2.6.1 Simulation Setup 33
2.6.2 Simulation Results 34
2.7 Fault Modeling and Test Implications 36
2.8 Conclusion and Future Work 39

3 3D-SoC Test Architecture Design and Optimization 41
3.1 Introduction 41
3.2 Preliminary 42
3.2.1 Prior Work in SoC Testing 43
3.2.2 Prior Work in Testing 3D ICs 45
3.3 On Test Time and Routing Cost 46
3.3.1 Motivation 46
3.3.2 Problem Formulation 48
3.3.3 Proposed Algorithm 50
3.3.4 Experiments 54
3.4 Pre-bond Test-Pin Constrained TAM Sharing 59
3.4.1 Pre-Bond Test-pin Constraint 59
3.4.2 Motivation 60
3.4.3 Problem Formulation 63

3.4.4 Test Wire Sharing 64
3.4.5 Layout-Driven Test Architecture Design and Optimization 71
3.4.6 Experiments 74
3.5 Conclusion 77

4 Stack Yield Enhancement for 3D DRAM 78
4.1 Introduction 78
4.2 Preliminaries and Motivation 80
4.2.1 Prior work 80
4.2.2 Motivation 81
4.3 3D-Stacked Memory Architecture with Redundancy Sharing . . . 83
4.4 Memory Repair with Redundancy Sharing 84
4.4.1 Problem Formulation 85
4.4.2 Memory Repair Strategy 85
4.5 Matching for Yield Enhancement 86
4.5.1 Problem Formulation 86
4.5.2 Direct Matching 87
4.5.3 Iterative Matching 92
4.6 Experimental Results 93
4.6.1 Experiment Setup 93
4.6.2 Results and Discussion 94
4.7 Conclusion 95

5 TSV Repair for Assembly Yield Enhancement 97
5.1 Introduction 97
5.2 Preliminaries and Motivation 98
5.3 Proposed TSV Redundancy Architecture 102
5.3.1 Overall Structure 102

5.3.2 Switch Design and Repair Path Routing 103
5.4 Proposed Repair Algorithm 104
5.4.1 A Maximum Flow Method Based Approach 105
5.4.2 Additional Delay due to Signal Rerouting 105
5.4.3 Problem Analysis due to Timing Constraint 107
5.4.4 Repair with Length Constraint 108
5.5 TSV Redundancy Architecture Construction 110
5.5.1 Defect Probability Model with Spatial Correlation 112
5.5.2 Reparability Analysis 112
5.5.3 Topology Mapping 117
5.6 Experimental Results 119
5.6.1 Experimental Setup 119
5.6.2 Results and Analysis 120
5.7 Discussion 125
5.7.1 TSV Grid Construction 125
5.7.2 Cost Analysis 126
5.8 Conclusion 127

6 In-Field TSV Repair for Reliability Enhancement 128
6.1 Introduction 128
6.2 TSV Latent Defects 129
6.3 Motivation for In-Field Repair 130
6.4 In-Field TSV Repair Framework 132
6.4.1 Online Test and Diagnosis 133
6.4.2 Spare TSV Sharing and Reconfiguration 133
6.5 Proposed Repair Algorithm 135
6.5.1 Problem Formulation 135
6.5.2 In-Field Repair Algorithm 135

6.5.3 Impact of TSV Redundancy Sharing 138
6.6 Experimental Results 139
6.6.1 Experimental Setup 139
6.6.2 Results and Analysis 142
6.7 Conclusion 146

7 Conclusion and Future Work 147

Bibliography 148

List of Figures

1.1 An Example 3D-SIC. 2
1.2 Bonding Manners. 3
1.3 An Example Manufacturing Process. 4
1.4 Two Probe Manners for Pre-bond Test Access. 7

2.1 Two Example 3D-stacked DRAMs. 19
2.2 3D DRAM Model. 21
2.3 Simulation Model and Two Scenarios for Write/Read Operation with Wordline Opens. 25
2.4 Write Operation with Wordline Open. 26
2.5 Read with Wordline Open: An Example. 27
2.6 Read Operation with Wordline *Border* TSV Open. 28
2.7 Simulation Model and Two Scenarios for Write/Read Operation with Bitline Opens. 29
2.8 Read with Bitline Open: An Example. 30
2.9 Read Operation with Bitline Open. 31
2.10 TSV Coupling from Multiple Layers. 33
2.11 Case 1 for No Access Faults. 35
2.12 Case 2 for Wordline Multiple Access Fault. 36

3.1 An Example for Bus-based TAM and its Optimization. 44

3.2 An Example 3D SoC Test Architecture Optimized for Post-bond Test Only. 46

3.3 The Impact of Pre-Bond Tests. 47

3.4 Main Flow of the Proposed Algorithm. 51

3.5 Inner TAM Width Allocation Procedure. 54

3.6 Detailed Testing Time of p22810. 57

3.7 Pre-Bond Test Pad: (a) C4 Bump as Test Pad, (b) Wire-Bond as Test Pad. 60

3.8 Test Architecture for an Example 3D SoC. 61

3.9 Routing Resource Sharing Example: (a) Test Architecture During Post-Bond Test, (b) Reuse TAM During Pre-Bond Test. 62

3.10 An Example of Post-Bond TAM Routing. 65

3.11 Post-Bond TAM Routing Algorithm 66

3.12 Reusable Routing Resources Represented by Bounding Rectangle. 67

3.13 Greedy Heuristic for Pre-Bond TAM Routing 69

3.14 Example for the Proposed Algorithm to Reuse TAM Routing Resources. 71

3.15 Design Flow for Scheme 1. 72

3.16 Design Flow for Scheme 2. 73

3.17 Pre-Bond TAM Routing in p93791. (a) without Reusing Post-Bond TAMs; (b) Reusing Post-Bond TAMs. 75

4.1 3D-Stacked DRAM 81

4.2 Different Matching Strategy affects the Overall Yield. 82

4.3 Redundancy Sharing using TSVs. 83

4.4 Irrespective Sub-Bipartite Graphs. 86

4.5 Die Matching Algorithm 89

4.6 Matching according to Reparability Condition. 89

4.7 Maximum Matching and Reparability. 91
4.8 Irreparability Condition. 91
4.9 Iterative Matching with Changing Conditions. 93

5.1 Existing TSV Redundancy Solutions: (a) Signal Switching [1]; (b) Signal Shifting [2]; (c) Crossbar [3]. 99
5.2 Results of Existing Repair Schemes Assuming 1/2 Redundancy Ratio. 100
5.3 The Existing Clustered Faults Aware Repair Scheme [4]. 100
5.4 Proposed TSV Redundancy Architecture: (a) An Example Physical Layout of Original TSV bundle and their Signal Entries; (b) A Physical Implementation of Proposed TSV Redundancy Architecture; (c) A Conceptual View of the Proposed TSV Redundancy Architecture; 101
5.5 Switch Design and Routing Capability. (a)Switch Design; (b)TSV grid with Edge-Disjoint Repair Path. 103
5.6 Timing Issue caused by Signal Rerouting. 105
5.7 Problem Transformation. 107
5.8 Proposed Alogrithm 110
5.9 Illustration of the Repair Algorithm: (a) Example TSV Grid; (b) Search Procedure Demonstration 111
5.10 The Reparability Condition:(a) Irreparable Example; (b) Reparable Example; (c) Minimal Orthogonal Cut Equals to Min-cut. . . . 113
5.11 Defect Probability Model:(a) Maximum Flow Method; (b) Length Bounded Search Heuristic. 114
5.12 Reparability Approximation Trends with respect to Average Defect Probability and Cluster Size (from 2×2 to 10×10). 116

5.13 Examples of the Topology Mapping and the Change of Faulty Map in TSV Grid. 117
5.14 Repair Schemes Varying Alpha from 1 to 3. 120
5.15 Repair Schemes Varying TSV Failure Rate with Fixed Alpha. . . . 121
5.16 Time Constrained Repair Varying Alpha from 1 to 3. 123
5.17 Time Constrained Repair Varying TSV Failure Rate with Fixed Alpha. 124
5.18 Mapping Irregular Placed TSVs. 125

6.1 Illustration of some TSV Latent Defects 130
6.2 An Example to Motivate the Need for Careful In-Field Repair. . . 132
6.3 Illustration of the TSV Redundancy Architecture. 134
6.4 Illustration of the Repair Algorithm. 136
6.5 The Impact of TSV Redundancy Sharing on Repair Algorithm. . . 139
6.6 MTTF Results in 4×4 TSV Grid with Varied Aging Coefficients and Fixed Potential Crack or Void Defect Distribution (0.1 kΩm, 0.1 kΩm). 140
6.7 Test Time Results in 4×4 TSV Grid with Varied Aging Coefficients and Fixed Potential Crack or Void Defect Distribution (0.1 kΩm, 0.1 kΩm). 141
6.8 MTTF Results for DES in 4×4 TSV Grid with Varied Potential Crack/void Defect Distribution and Fixed Aging Coefficients (0.05 kΩ/log(s),0.05 kΩ/log(s)). 143
6.9 Experimental Results in 8 × 8 TSV Grid Size Repair Architecture with Varied Aging Coefficients and Fixed Potential Crack/Void Defect Distribution (0.1 kΩ, 0.1 kΩ). 144

6.10 Experimental Results with Varied Rerouting Delay between Two Adjacent Routers (*ps*) and Fixed Aging Coefficients and Potential Crack/Void Defect Distribution. 145

List of Tables

2.1 Fault Modeling for Read Operation with Bitline Open. 38

3.1 Experimental Results for $\alpha = 1$. 55

3.2 Experimental Results for SoC t512505 Considering Both Testing Time and Wire Length. 56

3.3 Experimental Results for 3D SoC p22810 and p34392. 74

4.1 Experimental Parameters for Two Cases. 94

4.2 Experimental Results 96

5.1 TSV Related Experimental Setup. 119

5.2 Cost Comparison for 1k TSVs. 127

Chapter 1

Introduction

The shift towards volume production of 3D-SICs requires their manufacturing yield to be commercially viable and their service life to be as high as possible. However, the disruptive manufacturing process of 3D-SICs causes critical issues on its yield and reliability, posing great threats to their mainstream adoption. For better understanding the content of this thesis, we introduce the related backgrounds of 3D-SICs in this chapter. The remainder of the chapter is organized as follow: In Section 1.1, we first present the background knowledge of 3D-SICs, such as manufacturing process and mainstream prospects. In Section 1.2, we discuss the design for pre-bond testability and its challenges. The current advances for yield enhancement of 3D-SICs and the remaining challenges are then introduced in Section 1.3, wherein some of the descriptions are excerpted from our survey paper [5]. Finally, the motivation of this thesis work and the thesis organization are described in Section 1.4.

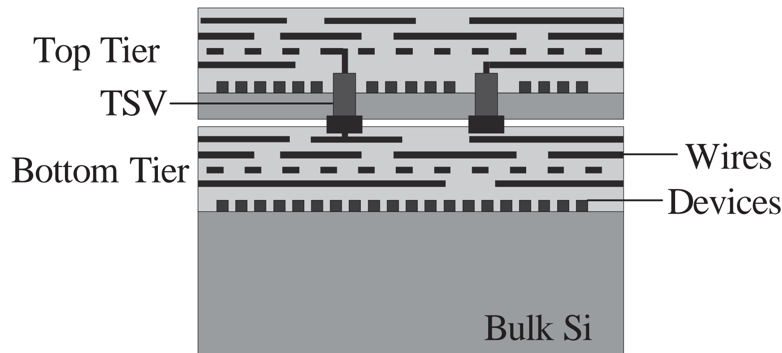


Figure 1.1: An Example 3D-SIC.

1.1 Three-Dimensional Stacked Integrated Circuits

In this section, we first introduce the concept of 3D-SIC and its advantages. We then present the manufacture process and the market prospects for 3D-SICs.

1.1.1 Background of 3D-SIC

In each new generation of electronic products, there are endless quests for higher performance and more functionality with less power consumption. As the technology scales into deep-submicron domain, however, interconnects have become the dominating factor for the performance and the power dissipation. 3D-SIC that provides abundant interconnects with improved performance and less communication energy has been proposed as a promising solution to resolve this problem [6].

3D-SIC integrates multiple silicon layers with short and dense through-silicon vias (TSVs), as shown in Fig. 1.1. This new integration paradigm has plenty of benefits, as stated below. First of all, with the smaller chip footprint and the micrometer-length TSVs, the total wire length can be dramatically reduced, thus leading to improved performance and less communication energy [7, 8]. Secondly, the reduced parasitic for interconnects in 3D-SICs facilitates the circuit design for high-performance applications, such as system-on-chips [9]. Thirdly, 3D dynamic

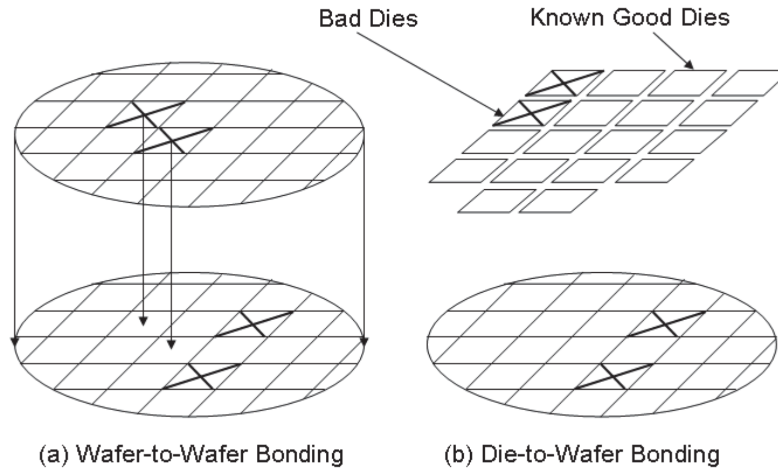


Figure 1.2: Bonding Manners.

random access memory (3D DRAM) [10], with the high bandwidth and low latency access provided by TSVs, becomes a promising solution to overcome the "memory-wall" issue (i.e., the speed and bandwidth gap between the processor and memory) [11]. At last, 3D-SICs also facilitate the integration of devices in heterogeneous technologies as each layer is fabricated separately. Therefore, although there are still critical issues such as yield and heat dissipation to be resolved in 3D integration, it is generally regarded that 3D-SICs will occupy a big market share in the future [12].

3D-SIC offers many unique benefits, but it is not a universal solution, since it incurs additional cost during manufacture and testing. In the remainder of this section, we briefly describe the manufacturing process of 3D-SIC and show the its prospects.

1.1.2 Manufacturing Process

3D-SICs can be built in several manners: wafer-to-wafer (W2W) bonding [13], die-to-wafer (D2W) bonding (only for 3D-SICs built on two semiconductor wafers) [14],

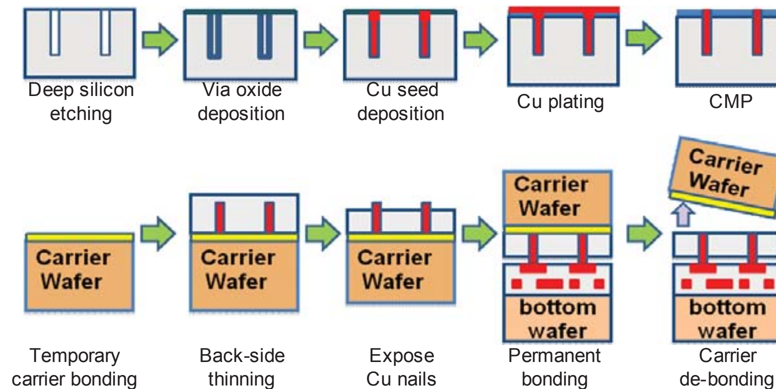


Figure 1.3: An Example Manufacturing Process.

or die-to-die (D2D) bonding [15]. W2W bonding directly stacks wafers together and then dices them into individual die stacks (see Fig. 1.2 (a)). This bonding strategy simplifies the manufacturing process [15], thereby leading to the highest throughput. But it requires that dies at different layers to have the same form factor, and thus, is suitable only for homogeneous integration (e.g. 3D DRAMs). In addition, W2W integration may suffer from significant yield loss due to the stacking of good dies and bad dies, referred to as the "known good die" (KGD) issue. On the contrary, with D2W/D2D integration, bare dies are first diced from a wafer and then stacked to other dies/wafers (see Fig. 1.2 (b)). This bonding strategy requires a more complex manufacturing process that applies pre-bond testing and attaches known good dies only, thus resulting in higher yield compared to W2W bonding [16]. In terms of bonding direction, there can be face-to-face bonding or face-to-back bonding. The former allows more interconnects between active devices on different layers, but it limits the number of stacked dies to be two; the latter is a scalable solution that supports more stacking layers.

The manufacturing process can be coarsely divided into (a) TSV formation [17], (b) wafer thinning and (c) bonding [18]. As depicted in Fig. 1.3(a), TSV formation has to go through a series process: (1) deep silicon etching of TSV holes, (2)

oxide deposition, (3) copper seed deposition, (4) copper plating, and (5) chemical-mechanical polishing (CMP). The wafer needs to be thinned down from the backside to expose the TSV tips. In order to provide sufficient mechanical strength and prevent it from breaking or cracking, we have to temporarily bond the to-be-thinned wafer onto a carrier wafer, prior to thinning, as Fig. 1.3(b) illustrated. Subsequently, the thinned product wafer on its carrier wafer is permanently bonded to the next layer, after which the temporary carrier wafer is removed. Above disruptive process involves tremendous changes of temperature and mechanical stress, and thereby introduces various defects, rendering a serious yield loss.

1.1.3 Prospects of 3D-SICs

The products that can benefit from 3D integration include 1) capacity-driven IC products (e.g., memory, FPGA and etc.); 2) complicated system-on-a-chips (SoCs) with footprint constraints; 3) IC products with high bandwidth requirements (e.g., high performance processors). Some of them have gain mainstream adoption (e.g., 3D DRAM [1]), with others being explored (e.g., 3D SoCs) or to be explored (e.g., Memory-on-logic stacking [10] and logic-on-logic stacking [19]).

Existing 3D DRAM simply uses TSVs as a vertical bus across multiple DRAM layers to link them to the peripheral layer [1, 20], rendering less memory access latency. Such memory organization is simple since the individual structures in each layer are still traditional two-dimensional memory structures. The product yield is also guaranteed by the mature memory self-test and repair mechanism. Specifically, sophisticated functional fault models for random access memory are proposed and a class of tests called march tests [21] has been proved to be able to effectively cover these modeled faults. Redundant rows and columns of storage-cells are added on-chip in advance so as to repair the faulty storage-cells (faulty bits) found in storage-cell arrays by replacing the rows/columns containing the

faulty bits with the redundant ones [22]. To fully exploit the benefits of 3D stacking technology, more aggressive DRAM organization is introduced wherein individual storage-cell arrays are stacked in a 3D fashion. In such a DRAM organization, TSVs are used as bitlines and wordlines to link these storage-cell arrays from different layers to the peripheral logics (e.g. decoder and sense amplifier) in the bottom layer, leading to significant increase of memory bandwidth with much less memory access time [23]. Due to the massive usage of TSVs and the high storage capacity, however, it has higher risk for such memory structure to be hurt by defective TSVs and/or unsuccessful repairing of storage-cell arrays, resulting in significant yield loss.

As SoC designs become increasingly complex, interconnects that emerged as the performance and power limiter can be addressed using 3D stacking, namely 3D SoC. 3D stacking also facilitates the integration of various IP cores with disparate technologies such as microelectromechanical systems (MEMS), various kinds of sensors and other heterogeneous elements demanded by applications, as they can be fabricated on different silicon layers separately before integration, thereby offering a genuine single-chip system solution. To test these core-based 3D SoC, we can partially reuse the basic SoC test infrastructure that is composed of 1) test source and sink, 2) test access mechanism (TAM) and 3) core test wrapper. To test a core in SoC, the test stimuli should be sent into the core-under-test (CUT) first, and the test response should be sent out, which is then compared with the predefined correct response. Test source and sink can be off-chip automatic test equipment (ATE) or on-chip BIST hardware. TAM transports the test stimuli from the source to the CUT and transports the test response out to sink. The test wrapper connects the CUT's terminals to the TAM, isolating the CUT from its environment during test. As the pre-bond test is required, however, the basic SoC test infrastructure faces new challenges, such as the limited test access and the extra test

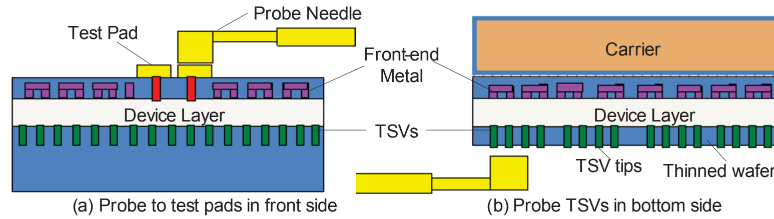


Figure 1.4: Two Probe Manners for Pre-bond Test Access.

cost, rendering the existing TAM architecture and cost optimization procedure inefficient [24]. In next section, we demonstrate the design for pre-bond test and its challenges.

1.2 Design for Pre-bond Test

Pre-bond test is to test individual dies to be stacked before the bonding process. One of the earliest works that address the design for pre-bond test of 3D-SICs is by Lewis and Lee [25]. In this work, the authors considered testing 3D-SICs with fine-grained circuit-level partitioning (i.e., functional blocks spread across multiple dies) and proposed a "scan island" approach to test incomplete circuits. This work pointed out an important observation: test access is the key challenge in pre-bond testing due to the difficulty in TSV probing.

In the state-of-art probe technology, the finest probe needle (tip) has a pitch of $35\mu m$ [26], and the capacity of a probe card (the number of probe needles a probe card contains) is only in an order of magnitude of hundreds. Consider a single die to be probed that fabricates several thousands of TSVs with normally $10\mu m$ pitch. There is a dimension gap between the probe needle and TSV tip that are exposed on the bottom side of the die, rendering an insufficient probe on TSV tips.

There are two potential solutions to tackle above issue [25, 27]:

- *Additional test pads for test*: That is, as can be seen in Fig. 1.4(a) to fabri-

cate a number of test pads on face side of the die (where front-end metals are placed) so that the automatic test equipment (ATE) can probe it during testing. The test pads have to be sized according to today's probing technique. Obviously, this comes with an area penalty, and hence the number of extra pads should be minimized. Also, care must be taken so that the probe force on thinned wafer would not damage the wafer itself [24].

- *Probe technology improvement*: In order to directly probe the TSV tips, as shown in Fig. 1.4(b), advanced probe technology need to be investigated to provide fine-grain probe needles with larger capacity in the probe card.

Most existing test solutions adopt the former approach and consequently provide methods to design the test access mechanism in the die. Lewis et al. [28] proposed several test methods for circuit-level partitioned 3D-SICs. The basic strategy is to establish the control-observe points, by either inserting scan registers to support structural test or selectively partitioning a functional block so that the read/write ports are both available on the same layer so as to use functional test. Instead of inserting scan flip-flops to both ends of TSVs, Li and Xiang [29] proposed to reuse existing primary I/Os or pseudo primary I/Os to provide controllability and observability for signals associated with TSVs. Wu et al. [30] proposed 3D scan chain design that cross multiple dies to minimize the stitching wirelength. For the more practical 3D-SICs that are partitioned at block or core level, modular testing is a natural choice to reduce the required number of test pads. Marinissen et al. [31] proposed to add IEEE 1500-like test wrapper for a die with 3D-specific extensions, such as dedicated test pads for pre-bond probing. Marinissen et al. [32] further extend this die-level wrapper by providing external control interface, which is now being formalized as IEEE P1838 standard [33]. These test solutions, however, have simplified assumptions. For example, modular test access design is limited by the assumption that all the devices are scan testable. Hence, it is neces-

sary to consider more complicated scenarios, e.g., clock generator, analog/RF and non-CMOS devices.

Besides the silicon die testing, the difficulty in TSV probing also raises questions to the feasibility of pre-bond TSV testing since no test access exists at the TSV tips (located on the bottom side of the die). To solve the problem, existing works focus on measuring the I-V characteristics (e.g., resistance and capacitance) with the help of design-for-test (DfT) circuitries built around TSV end-points that located on the face side of the die. The application of on-chip sense amplification was proposed in [34, 35] for detecting capacitive TSV faults. Further, a voltage divider is added for scannable voltage test [36]. Two other methods utilizing leakage current sensor and capacitive bridge were evaluated in [37] to measure TSV resistance and capacitance. As analog devices are utilized in above DfT circuits, the accuracy of measurement is quite sensitive to the tester capability and environmental noises. Moreover, these sophisticated DfT circuits significantly increase the hardware cost.

At last, the pre-bond test complicates the conventional test flow and increases the test cost. Besides pre-bond test, Marinissen [38] discussed the impact of conducting "intermediate stack test" and "pre-package test" before the final post-bond package test, and they are treated as part of pre-bond testing in this thesis.

1.3 Yield and Reliability Challenges for 3D-SICs

All the advantages of 3D-SICs described in Section 1.1 must be translated into cost-effectiveness so that the emerging 3D products (including but not limited to what we mentioned in Section 1.1.3) can be commercially viable. Among the various factors that affect 3D-SIC product cost, manufacturing yield is one of the most (if not the most) crucial ones [16], and it was showed that the functional yield of a rather simple 3-layer chip is only a bit more than 60% [39]. In this section,

we first discuss the yield and reliability crisis in 3D-SICs. Then, a mathematical yield model for 3D-SIC is presented to show the importance of the yield issue.

1.3.1 Yield and Reliability Crisis in 3D-SICs

Generally speaking, there are two kinds of yield losses in manufacturing 3D-SICs.

- *Stack yield loss*, caused by defects in one or more of the stacked dies. Without KGD information, the yield of the W2W bonded 3D chips can be quite low, especially when the die size is large and/or the defect density is high [7]. While D2W/D2D bonding facilitates to achieve higher stack yield than W2W bonding strategy, however requires costly and challenging pre-bond test.
- *Assembly yield loss*, caused by defects occurred during the assembling process. The assembly process for 3D-SICs involves many challenging manufacturing steps, which may cause various types of TSV defects [40]. For example, the wafer thinning possibly leads to degradation of some I-V characteristics, shifts in device performances, and gives rise to yield losses [41, 42]. Insufficiently filling of TSVs is likely to occur during TSV formation, which results in micro-voids inside TSVs. In [43], twelve different types of TSV defects were identified, eight of which involve defects that arise prior to bonding, while the rest is induced due to alignment, bonding, or stress.

For stack yield improvement, needless to say, pre-bond test plays an important role for D2W/D2D bonding and it is critical to achieve high defect coverage to prevent bad dies from being stacked [24]. Even for W2W bonding, with KGD information from pre-bond tests, stack yield improvement can be achieved by conducting selective wafer matching for maximal combination of good dies [44, 45]. While wafer matching is helpful for stack yield improvement, its effectiveness is fundamentally constrained by the defect rate of individual dies.

For assembly yield enhancement, adding redundant TSVs [1, 2, 3, 46] to repair faulty ones is probably the only effective method besides improving the manufacturing process itself. Nevertheless, effective TSV testing is essential to identify the faulty TSV prior-/post bonding process before one can be repaired. A number of TSV redundancy allocation strategies were presented in the literature and they differed in terms of redundancy ratio, repair flexibility and capability, and hardware cost (e.g., [47]). However, their effectiveness is limited either by the high cost of redundant TSVs, or the poor scalability in 3D-SICs with massive use of TSVs.

Not like above manufacturing defects ($t = 0$), latent defects, such as micro-voids and interfacial cracks in TSVs, are induced by the thermal-mechanical stress during the TSV fabrication due to the different coefficients of thermal expansion among diverse materials in 3D stack. These latent defects are usually too small to be detectable in manufacturing test. While at runtime ($t > 0$), they get exacerbated with the electro-migration of TSVs [48, 49, 50], and finally lead to hard-to-predict timing errors on critical paths with TSVs, thereby resulting in accelerated chip failure in the field.

1.3.2 Yield Modeling

The manufacturing yield of a single silicon die based on compound poisson model [51] is as follows:

$$Y_{die} = (1 + \frac{DA_{die}}{\alpha})^{-\alpha} \tag{1.1}$$

wherein D is the defect density, A_{die} is the die area and α is the clustering parameter related to the technology and the design itself (e.g., circuit density and mask steps).

Yield modeling for 3D-stacked ICs is more complicated considering the extra processing steps, various stacking manners, and the impact of die/wafer matching, as discussed in this chapter.

1.3.2.1 Stack Yield Modeling

Consider a 3D-stacked IC product containing N layers, and the yield of ith layer die is Y_{die_i}. Its manufacturing yield using W2W integration (without matching) can be roughly calculated as follows:

$$Y_{stack,W2W} = \prod_{i=1}^{N}[Y_{die_i}] \tag{1.2}$$

With D2W/D2D integration, assuming perfect KGD tests, the yield of 3D-stacked IC product can be estimated as:

$$Y_{stack,D2W/D2D} = \min[Y_{die_i}], 1 \leq i \leq N \tag{1.3}$$

This is because, consider that we have the same number of dies fabricated for each layer, the final good die-stack will be constrained by the layer with the minimum number of good dies. The above clearly demonstrates the yield benefits of D2W/D2D integration [16, 24]. Let us now examine the yield model in detail considering the various factors besides bonding choice.

The impact of footprint: By partitioning a large monolithic 2D-IC into several smaller dies and stacking them together to provide the same functionality, it is in fact beneficial from the yield standpoint since each die now has a much smaller area. On the other hand, there is some additional area overhead for 3D-stacked ICs with TSVs and design-for-testability (DfT) for pre-bond testing. We can roughly obtain the yield model for die i in 3D-stacked IC as follows:

$$Y_{die_i}^{3D} = (1 + \frac{D_i}{\alpha_i}(\frac{A_{die}^{2D}}{N} + O_i))^{-\alpha_i} \tag{1.4}$$

wherein A_{die}^{2D} is area of the monolithic 2D implementation and O_i is the extra area overhead of this particular die.

The impact of KGD test: As discussed earlier, pre-bond testing can be applied to identify known good dies for later bonding. Clearly, the test quality will affect

the final yield of 3D-stacked IC products. The ratio of defective dies that escape pre-bond tests to all the ICs can be derived as [52]:

$$R_{escape} = 1 - Y_{die}^{1-F_c} \tag{1.5}$$

wherein F_c is the fault coverage of pre-bond tests. Thus, if D2W/D2D integration is used, the test escape ratio for the stacked IC can be estimated as [53]:

$$R_{escape,D2W/D2D} = 1 - \prod_{i=1}^{N} Y_{die_i}^{1-F_{c_i}} \tag{1.6}$$

Taking the above yield loss into consideration, the stack yield for D2W/D2D in Eq. 1.3 integration can be revised as:

$$Y_{stack,D2W/D2D} = \min[Y_{die_i} + R_{escape_i}] \cdot \prod_{i=1}^{N} Y_{die_i}^{1-F_{c_i}} \tag{1.7}$$

The impact of wafer matching for W2W integration: In [54], Verbree *et al.* formulated a closed-form mathematical model to approximate the stack yield with wafer matching, by introducing a probability $p(j)$, which denotes the occurrence of matching exactly j faulty dies between two bonding wafers. There are some limitations in this analytical model, e.g., the model assumes a fixed number of faulty dies per stack tier, as pointed out in the paper itself.

1.3.2.2 Assembly Yield Modeling

The assembly yield ($Y_{assembly}$) of 3D-stacked IC products can be calculated as follows:

$$Y_{assembly} = Y_{Bonding} \cdot Y_{TSV} \tag{1.8}$$

wherein $Y_{Bonding}$ is the bonding yield and Y_{TSV} is the TSV yield [53]. Currently, there is still no concrete model for $Y_{Bonding}$ that takes device failure caused by bonding into account and it is typically assumed to be a constant value. For Y_{TSV},

as discussed earlier, TSVs are vulnerable to various kinds of defects introduced during fabrication and stacking process [24, 27]. Without redundancy, Y_{TSV} is simply:

$$Y_{TSV} = (1 - f_{TSV})^{N_{TSV}} \tag{1.9}$$

where f_{TSV} is the TSV failure rate and N_{TSV} is the total number of TSVs. From the above equation, TSV yield is a crucial factor for 3D-stacked IC products, especially when the number of TSVs are large and/or the TSV failure rate is high. Consequently, it is essential to incorporate redundancy for TSV defect-tolerance.

1.3.2.3 Cumulative Yield Model

According to the cumulative yield property [55], the final yield of 3D-stacked SICs Y_{final} can be formulated as follows:

$$Y_{final} = Y_{stack} \prod_{i=1}^{N-1} Y_{assembly(i)} \tag{1.10}$$

where N is the number of layers in the 3D-stacked IC product, Y_{stack} is the stack yield and $Y_{assembly(i)}$ is the assembly yield for the $i^t h$ assembly process.

Let us do a simple mathematical calculation to show the importance of yield issue for 3D-SICs. Given a 3D-SIC containing six stacked dies, each of which has a yield 90%, we assume the test coverage of pre-bond test for each die equals to 80%. According to Equation (1.7), the stack yield of this 3D-SIC is 81%. We further assume there are 1000 TSVs with 500dppm (defect part per million) implemented in this 3D-SIC. And we can get the assembly yield as 77.8%, according to Equation (1.9). The final yield is now 63% based on Equation (1.10), rendering a significant yield loss for this 3D-SIC. This simple example shows that the yield enhancement technique is essential for the mainstream semiconductor industry to adopt 3D-SICs.

1.4 Thesis Motivation and Organization

In this thesis, we propose a systematic set of solutions, providing effective and efficient designs and algorithms, to enhance the yield and reliability of 3D-SICs.

First of all, extensive works have studied the new types of defects introduced in 3D integration and found that conventional fault models can cover a majority of them. There is little work discussing TSV coupling effect [56], especially for future 3D DRAMs that employ a large amount of TSVs. This motivate us to present an analytical model to capture the faulty behavior of TSVs in 3D DRAMs structures. This model serves as the first step towards further yield and reliability enhancement techniques, as demonstrated as follow:

To improve the stack yield, pre-bond test is essential to prevent faulty dies from being stacked. However, the basic test architecture of 2D SoC is not able to support the pre-bond test for 3D SoCs. Motivated by this, in this thesis, we propose an efficient test architecture design and optimization methodology to facilitate the pre-bond test for 3D SoCs. In addition, pre-bond test requires additional test pins, as depicted in Section 1.2, leading to significant hardware cost. This motivates this thesis to optimization the test architecture under pre-bond test-pin-count constraint.

With enhanced pre-bond testability, there is less chance for a faulty die to be mistakenly stacked with good dies. However, 3D-SICs still suffer yield loss due to the discarded faulty dies. A more attractive way is to make use of bad dies after stacking, and it is possible only if the die has inherent redundant resources (e.g., memory). Consider conventional 2D memories that a memory block cannot borrow redundant resources from its neighbors for repair due to the routing complexities and the change of electrical properties. A 2D memory chip is deemed to be faulty if any memory block cannot repair itself with its own redundant resources (self-irreparable). In 3D memories, however, it is possible for a

self-irreparable memory block to use TSVs to borrow redundant resources from its vertical neighbor. Motivated by this, in this thesis, we discover a novel repair framework for 3D DRAMs, including redundancy sharing scheme and an iterative die-matching algorithm that selectively match the DRAM dies to be stacked to enhance the stack yield.

In order to resolve the TSV induced assembly yield loss, effective TSV repair mechanism should be addressed. Existing TSV repair techniques are with limited reparability, and more importantly, are vulnerable to clustered TSV faults. Motivated by above, we propose a novel TSV-grid based redundancy architecture and corresponding algorithm to tackle above problem.

To enhance the reliability and extend the service life of 3D-SICs, the TSV latent faults must be detected and repaired in the runtime, which is, in any case, beyond the capability of all existing TSV repair solutions. Motivated by this, in this thesis, we describe a reconfigurable in-field repair solution that is able to effectively tolerate latent TSV defects through the judicious use of spares. The proposed solution includes a reconfigurable repair architecture that enables spare TSV sharing between TSV grids, and the corresponding in-field repair algorithms.

The remainder of this thesis is organized as follows. Chapter 2 addresses the fault modeling for TSVs in 3D DRAMs. Effective test patterns are also generated to cover the new faulty behaviors. Chapter 3 presents the test architecture design and optimization for 3D SoCs. The pre-bond test-pin-count constraint is also considered in this chapter. We present the repair framework for 3D DRAMs in Chapter 4. In Chapter 5, the TSV-grid based redundancy architecture and repair algorithm are described. Based on above work, in Chapter 6, we propose an efficient in-field repair solution, including the hardware architecture together with an in-field TSV repair algorithm. Finally, chapter 7 concludes this thesis and points out our future research directions.

□ **End of chapter.**

Chapter 2

Modeling TSV Open Defects in 3D DRAM

2.1 Introduction

3D-stacked memories can be implemented in several manners. One possible organization is simply using TSVs to implement a vertical bus across multiple DRAM layers to link them to the processor layer [1, 20, 10], as shown in Fig. 2.1(a). Such memory organization reduces the long memory access latency, but does not provide much bandwidth benefits because the individual structures in each layer are still traditional two-dimensional memory structures.

To fully exploit the benefits of 3D stacking technology, another DRAM organization is introduced wherein individual storage-cell arrays are stacked in a 3D fashion. TSVs are used to link these memory arrays from different layer to peripheral logic (e.g. decoder and sense amplifier) in bottom layer [23], as shown in Fig. 2.1(b). By isolating the peripheral logic implemented with CMOS technology from the DRAM bitcells implemented with NMOS technology, such architecture not only reduces manufacturing complexity, but also enables individual optimiza-

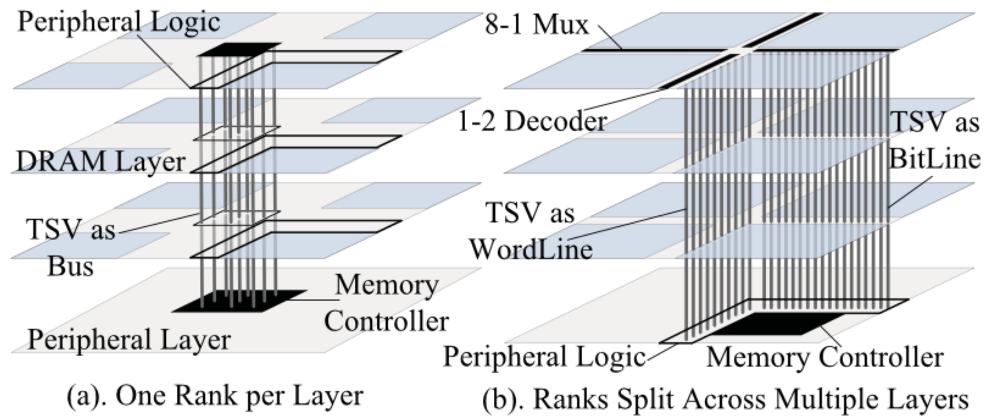


Figure 2.1: Two Example 3D-stacked DRAMs.

tions of logic layer for speed and storage layers for density, thus dramatically reducing memory access time. In addition, TSVs are implemented as part of the bitlines and wordlines, leading to significant increase of memory bandwidth.

Recently, Tezzaron Semiconductor has implemented the above "true" 3D DRAM architecture (see Fig. 2.1(b)). In their design, one TSV is shared by two wordlines through a 1 to 2 decoder at the edge of each memory array. At another edge of each memory array, the 8 to 1 mux are placed to control which bitline can connect to the sense amplifier by TSV [57]. This design not only release the pitch-mismatch problem [57], but also reduce the excessive use of TSVs. However, it still leads to a massive usage of TSVs with density in the range of tens of thousands of TSVs/mm^2, that is, approximately 1.5 million TSVs for 1Gb memory [58]. Consequently, to obtain high manufacturing yield for such 3D DRAM circuits, it is essential to understand the faulty behavior of TSV defects and develop effective test and repair solutions to tolerate such defects.

The primary failure mechanism for TSVs is random open defects (e.g., caused by void after filling) during TSV fabrication [3]. However, we cannot simply model such defects as wordline/bitline stuck-open faults as in [59] for 2D DRAM circuits. This is because the extremely high density of TSVs makes capacitive

coupling effects among them not negligible [60]. To tackle this problem, in this chapter, we conduct extensive simulations to study the faulty behavior of TSV open defects and map them to functional fault models of the memory circuits, which serves as the first step to tackle the test and repair problem for 3D DRAMs.

The remainder of this chapter is organized as follows. Section 2.2 presents preliminaries of this work. In Section 2.3, we describe the simulation methodology employed in this work. Next, the simulation results and analysis for wordline opens and bitline opens due to TSV defects are detailed in Sections 2.4-2.6. In Section 2.7, we map the TSV open defect into memory functional fault models and present corresponding test implications. Finally, Section 2.8 concludes this chapter.

2.2 Preliminaries

2.2.1 Memory Operation

Fig. 2.2 presents an example equivalent circuit with one DRAM layer connecting to the bottom peripheral layer using TSVs. For simplicity, only three wordlines (WL_A-WL_C) are shown in this figure. With the commonly-used folded bitline DRAM architecture [61], each column in the memory array has two bitlines (BL and $\overline{BL}$), intersecting each wordline at two points, wherein one storage-cell is placed in one of these two points. Nevertheless, the storage-cell can be placed in either BL or $\overline{BL}$, for example, storage-cell along WL_A are all in BL while that along WL_B and WL_C are all in $\overline{BL}$ (see Fig. 2.2). Pass transistor connects storage-cell with bitline, and it is controlled by the corresponding wordline.

During write operation, the wordline is driven with logic ‘1’, turning on the pass-transistor. After write enable signal (WE) is turned on, the input data directly drive the bitlines(e.g. drive BL to ‘1’ and drive $\overline{BL}$ to ‘0’ simultaneously), charg-

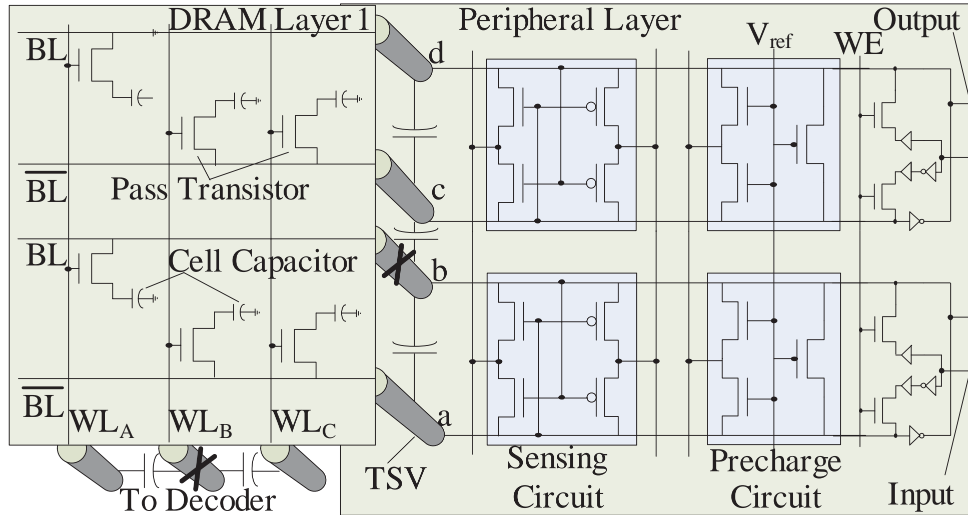


Figure 2.2: 3D DRAM Model.

ing (discharging) the target cell capacitor. During read operation, there are four phases: (i) In *precharge* phase, both BL and $\overline{BL}$ are charged as V_{ref} by precharge circuit. After that, precharge circuit is isolated from bitlines. (ii) In *access* phase, specific wordline is accessed by driven to logic '1, turning its pass-transistors on. Then the storage capacitors begin to charges/discharge the BL while $\overline{BL}$ remains as Vref (i.e. when WL_A is accessed in Fig. 2.2), and vice versa (i.e. when WL_B or WL_C is accessed). (iii) In *sensing* phase, the minute voltage difference between BL and $\overline{BL}$ makes the two transistors in diagonal position in sensing circuit more conductive, leading to a positive feedback, which continually enlarges the voltage difference until one of them is pulled up to '1' while the other dragged down to '0'. (iv) Finally, the result comes out through the output circuit and restored back to the accessed storage-cell.

2.2.2 Related Work and Motivation

Open defects in memory circuits are traditionally modeled as stuck-open faults [59]. With this fault model, when an open spot occurs on the bitline/wordline, the memory cell becomes unaccessible. Using defect injection and SPICE simulation, resistive open defects on the wordlines/bitlines of traditional 2D DRAM circuits have also been studied in several prior works [62, 63]. To the best of our knowledge, there is no work on full open defect in DRAMs.

Generally speaking, a full open defects breaks an interconnect into two parts: one connected to the source, while the other disconnected as a floating net. According to several electrical models for full interconnect opens [64, 65, 66], the parasitic capacitances between the floating net and its neighbors may have a significant impact on the voltage of the floating part, and *Aggressor-Victim* model is commonly used for analysis. For example, in [64], the authors sum up all the coupling capacitance of aggressors having logic value '0/1' as C_0/C_1. Then it determines the voltage of the floating part by comparing C_0 and C_1. For 2D DRAM circuits with bitline/wordline opens, coupling from its surroundings is of a less concern because the coupling capacitance is usually quite small and hence does not affect the DRAM faulty behavior.

When TSVs are used as part of the bitlines/wordlines in 3D DRAMs, however, not only they are more prone to random open defects [3], but also their capacitive coupling is not negligible, as studied in [60]. Previous stuck-open or resistive open fault models hence cannot accurately capture the faulty behavior of TSV open defects in 3D DRAM. In addition, as the coupling effects between aggressor TSVs and victim open TSV vary with the operation of the DRAM, the faulty behavior of TSV open defects in 3D DRAM is affected by many factors, e.g., operation type (i.e., read or write), voltage of its neighboring wordlines/bitlines and coupling capacitance from its surroundings. Consequently, it is almost impossible to

analyze the faulty behavior of TSV open defects statically. The above motivates us to conduct extensive simulations to study TSV open defects in this work.

2.3 Simulation Methodology

Based on [**?**], we set the size of TSV as $1.5\mu m \times 1.5\mu m$. According to the 3D DRAM cell feature size, we estimate the distance between TSVs to be $2-4\mu m$. Based on the previous characteristic extraction for TSVs in [60], we set two reasonable value $0.6fF$ and $1fF$ as the boundary of their coupling capacitance in our simulation, regardless of the TSV open position. The storage-cell intrinsic capacitance is set to be $30fF$. We also set $V_{dd} = 1.8V$ and reference voltage is half of it. The threshold voltage of each pass-transistor is set to be $0.6V$.

In our simulation study, for a victim TSV with open defects, we only consider the capacitive coupling effects from its neighboring TSVs (i.e., without other TSVs in between), because the coupling effects from farther TSVs are usually shielded out by the TSVs in between [60]. We are not concerned about the coupling effects between TSVs used to implement part of wordlines and those on bitlines, that is because they are routed outside of two different borders of each DRAM bank respectively, by which their coupling effects can be ignored. In addition, we differentiate TSVs that are at the border and hence only have neighbors at one side(denoted as *border TSVs*) and TSVs that are in the middle and hence are surrounded by other TSVs (denoted as *middle TSVs*), because they suffer from different coupling effects. For the ease of discussion, we denote “$X\mathbf{w}Y$” as write logic X (i.e., ‘1’ or ‘0’) to a cell with logic Y (‘1’ or ‘0’). Similarly, “$X\mathbf{r}$” denotes read from a cell with logic X. The wordline/bitline with TSV open defect is simply denoted as open wordline/bitline. We also assume that the coupling effect between TSVs and traditional vias are negligible due to the small size of traditional vias.

In the following, we first show our simulation models and results for word-

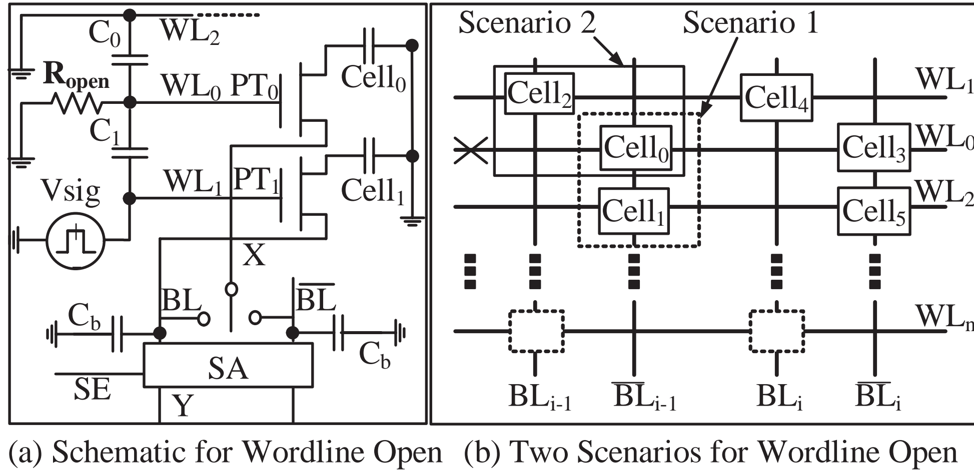
line/bitline opens considering coupling effects from TSVs originating from the same layer, where there are at most two neighboring aggressors to the floating wire (Section 4-5). Then, in Section 6, we extend the simulation model to consider the coupling effects between TSVs passing through different layers with more aggressors.

2.4 Simulation for Wordline Opens

2.4.1 Simulation Setup

For TSV open defect on wordline, the floating part is the wordline in DRAM layer and the pass transistors (MOSFET) on it. According to [65], the gate-to-source voltage (V_{gs}) of a floating pass transistor depends on two factors: voltage on neighboring nets of the floating wordline (V_{mg}) and the drain-to-source voltage (V_{ds}) that determines the operational region of transistor (i.e., off, linear or saturation). As "0**w**0" and "1**w**1" operations will not change the cell capacitor, we only consider the following operations: "1**w**0", "0**w**1", "1**r**", and "0**r**" in our simulation. In addition, consider the example circuit in Fig. 2.2 and suppose WL_B has a TSV open defect, there are two possible cases to be concerned: (i) WL_B is accessed (turned on); (ii) one of WL_B's neighbors, WL_A, is accessed. For the former case, as none of its neighbors can be turned on (only one wordline can be accessed in a memory bank), it behaves as a stuck-open fault. We therefore only need to simulate for the latter case.

Fig. 2.3(a) shows the schematic circuit used in our simulation for wordline opens. The open defect is represented by a very large resistance R_{open} in SPICE simulation. Wordline WL_0 has an open TSV, we simply denote it as an open wordline. Wordline WL_1 (WL_2) is the neighboring wordline being turned on (turned off). PT_0 and PT_1 are the pass transistors. C_0 and C_1 are the corresponding cou-



(a) Schematic for Wordline Open (b) Two Scenarios for Wordline Open

Figure 2.3: Simulation Model and Two Scenarios for Write/Read Operation with Wordline Opens.

pling capacitance between adjacent TSVs, driven by logic ‘1’ and ‘0’, respectively. $Cell_0$ is the floating cell and $Cell_1$ is the accessed cell. C_b is the wire parasitic capacitance in bitline. We use a voltage pulse source (V_{sig}) to represent the voltage change when wordline is accessed. The switch X is used for transfering between two scenarios in terms of the corresponding position of $Cell_0$ and $Cell_1$: (i) connect X to BL such that the accessed cell and floating cell are in the same bitline; (ii) connect X to $\overline{BL}$ such that accessed cell and floating cell are in the complemental bitlines. We use the following example to demonstrate these two scenarios.

Fig. 2.3(b) presents the structural view of accessing neighboring wordline of an open one. Let us consider wordline (WL_0) is open, thus $Cell_0$ is the floating cell. For scenario (i), WL_2 is accessed and $Cell_1$ is the accessed cell. $Cell_0$ and $Cell_1$ are in the same bitline (BL_i). For scenario (ii), WL_1 is accessed and $Cell_2$ is the accessed cell. $Cell_0$ and $Cell_2$ are in complemental bitlines.

During write operation there is no difference between these two scenarios, since there is a source driving both BL and $\overline{BL}$. Thus, we turn off the sense enable signal SE to isolate the sensing amplifier and drive the bitline BL to corresponding

logic ($Y = 1$ or $Y = 0$) (see Fig. 2.3(a)). During read operation, the corresponding position of the accessed cell and floating cell should be considered, because there is no strong source in bitline during sensing phase. Thus the different corresponding positions of accessed cell and floating cell affects the simulation results. As a result, we consider both scenarios during read operation simulation with open wordline.

2.4.2 Simulation Results and Analysis for Write Operation

Fig. 2.4(a)-(b) shows voltage change on $Cell_0$ due to the coupling effect. As $1.2V$ is the maximum voltage can be written into a fault free storage-cell, we set the initial V_{Cell_0} 1.2V for "1**w**0" and $0V$ for "0**w**1". Fig. 2.4(a) shows that for *middle* TSV open, a single "0**w**1" operation to $Cell_1$ can only drive $Cell_0$ to $0.4V$ due to the weak pulling up capability of NMOS transistor while the "1**w**0" operation can pull down the voltage of $Cell_0$ efficiently. Fig. 2.4(b) shows the simulation results by applying the same operation six times (i.e., "0**w**1**w**1**w**1**w**1**w**1**w**1" and "1**w**0**w**0**w**0**w**0**w**0**w**0"). For wordline with *middle* TSV open, the voltage can be written into a cell capacitor in "0**w**1" operation is no more than $60mV$, while the times needed to write cell capacitor to 0 depends on the coupling capacitance be-

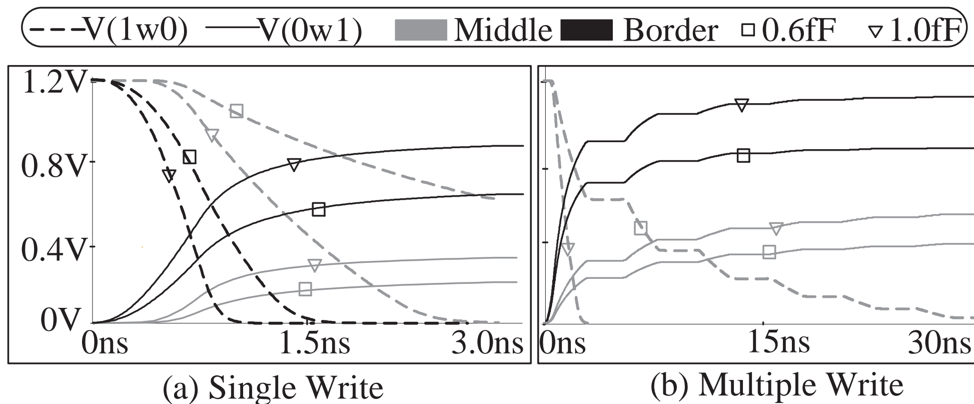


Figure 2.4: Write Operation with Wordline Open.

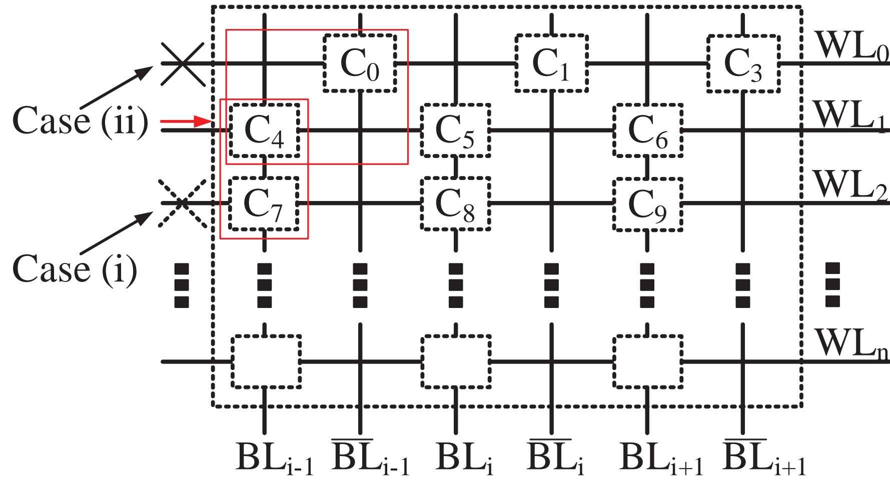


Figure 2.5: Read with Wordline Open: An Example.

cause aggressor TSVs with higher coupling capacitance is more effective to drive the floating cell capacitor. Both Fig. 2.4(a)-(b) shows that it is able to drive cell capacitor in a *border* TSV more aggressively, even with low coupling capacitance $0.6fF$.

2.4.3 Simulation Results and Analysis for Read Operation

Previous results show that an open wordline can turn on the pass-transistor when its neighboring wordline are accessed. Because of this, two storage-cells may be read out at the same time. One is in the accessed wordline (accessed cell) while the other is in the open wordline (floating cell). To simulate logic '1' in accessed cells, we set the initial value to be $1.2V$. The upper-bond of floating cell is set according to the result of previous write operation simulation (in Fig. 2.4).

After simulation, we found that there are only two cases that the output is affected by floating cell: (i) when the floating cell and the accessed cell are in the same bitline, and accessed cell is with logic '1' while the floating cell is with logic '0'; ($C_7 = 0, C_4 = 1$ in Fig. 2.5); (ii) when these two cells are in complemental bitlines, and the two cells are both with logic '0' ($C_0 = 0, C_4 = 0$ in Fig. 2.5).

Furthermore, the output is only affected when $C_0 = 0$ in the wordline with *border* TSV open. The final result is still correct when C_0 is in the wordline with *middle* TSV open. That is because, for *middle* TSV open, the capacitive coupling from logic '1' is equal to that from logic '0', thus, the voltage in the victim wordline is not high enough to fully open the pass transistor, and C_0 cannot affect the bitline effectively.

Fig. 2.6(a)-(b) present the results of cells stay in the same bitline and complemental bitlines when the floating cell is 0. In both cases, the result is incorrect when coupling capacitance is $1fF$. In Fig. 2.6(a), C_4 and C_7 are fighting with each other on the same bitline after wordline access. Bitline is first discharged by C_7 leading to voltage drop, then the charging effect from C_4 become dominant and pull up the voltage of bitline. However, it cannot reach the V_{ref} at last, leading to incorrect result. The correct voltage of BL_{i-1} should be logic '1' but now it is '0'. The correct voltage of C_4 should be logic '1' (the voltage in accessed cell remain the same during read operation), but now it is also logic '0'.

In Fig. 2.6(b), C_0 and C_4 are fighting on two complemental bitlines, which are being charged from V_{ref} (0.9V). Along with the charging, V_{ds} of pass transistor (equal to voltage difference between bitline and cell capacitor) is reduced,

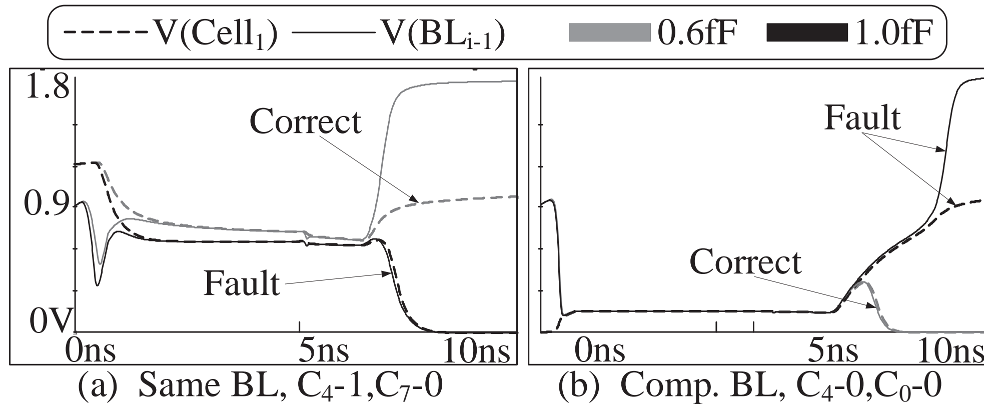


Figure 2.6: Read Operation with Wordline *Border* TSV Open.

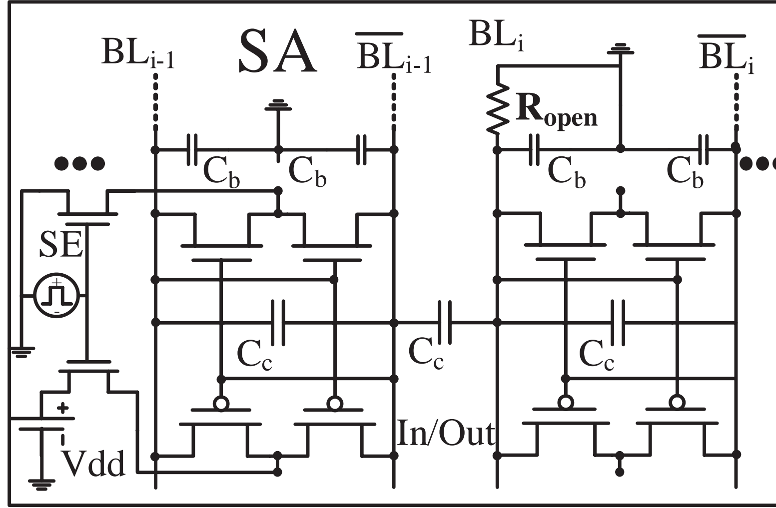


Figure 2.7: Simulation Model and Two Scenarios for Write/Read Operation with Bitline Opens.

making the pass transistor stay in linear region. Since it is weak to charge the floating cell $Cell_0$ to logic '1', the voltage in $Cell_0$ is lower than that in $Cell_4$. Thus, at last in the saturation region, the voltage in $\overline{BL_{i-1}}$ is lower than that in BL_{i-1}, leading to incorrect result. The correct voltage of BL_{i-1} should be logic '0', but now it is '1'. The correct voltage of C_4 should be logic '0', but now it is also '1'.

2.5 Simulation for Bitline Opens

2.5.1 Simulation Setup for Bitline Opens

For write and read operations under open bitline, we do the simulation in two cases. (i) directly access the open bitline; (ii) access its neighboring bitline. Fig. 2.7 presents the schematic circuit used in our simulation for bitline BL_i with TSV open defect. The coupling capacitance C_c is between adjacent TSVs. We set the bitline capacitance to a feasible value $20fF$. The simulation setup for write operation is less complicated, because sense amplifier is shut down during write operation,

only the accessed storage-cell is affected. To simulate **'1'**, we link a voltage pulse source to the corresponding bitline. For case (i), we add this voltage pulse source to the open bitline. For case (ii), we link it to the open bitline's neighboring bitlines. Before sensing phase, all bitlines are floating without any strong source.

Fig. 2.8 presents the structural view for this situation. Assume bitline BL_i suffers TSV open defect. For read operation with this open bitline, let us consider two cases: (i) If we access WL_0, bitline $\overline{BL_i}$ is driven by C_1. Although bitline BL_i is floating with reference voltage due to open defect, the sense amplifier can still sense the voltage difference between $\overline{BL_i}$ and BL_i and then drive both bitlines to opposite voltages. In this case, the open defect will not influence the output of read operation. (ii) However, if we access WL_1, bitline $\overline{BL_i}$ is floating because there is no storage-cell being accessed in $\overline{BL_i}$. At the same time, bitline BL_i is also floating due to the TSV open defect. Thus, bitline BL_i will be influenced by C_4 on bitline BL_{i-1} while bitline $\overline{BL_i}$ will be influenced by C_6 on bitline BL_{i+1} (see the arrows in

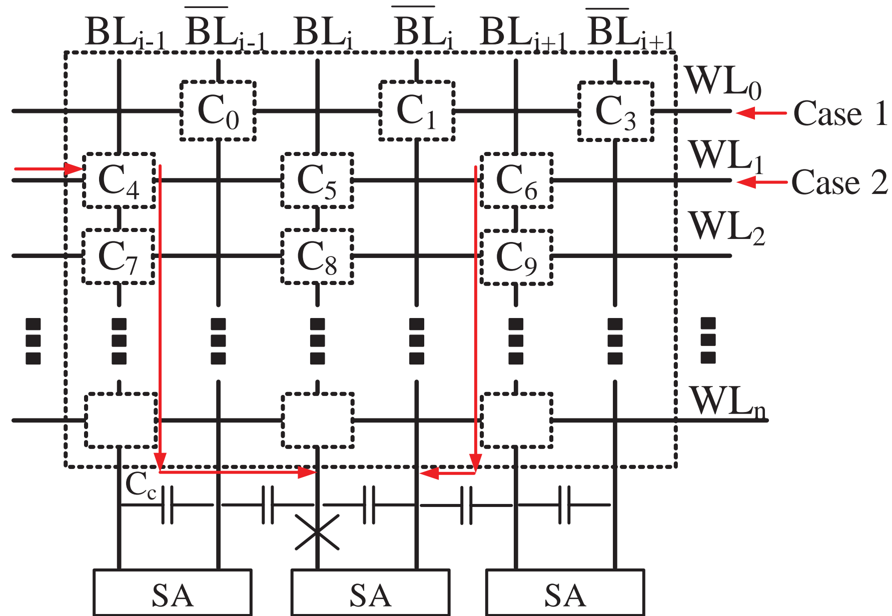


Figure 2.8: Read with Bitline Open: An Example.

Fig. 2.8). As a result, we run simulation for the latter case to see the corresponding fault behavior. It should be notes that, before read operation's sensing phase, all bitlines are floating at V_{ref} since there is no strong source.

2.5.2 Simulation Results and Analysis for Write Operation

For write operation under bitline open defect, there are two scenarios as mentioned in section 2.5.1. (i) We directly drive the open bitline with logic '1' and '0' respectively, and find that the voltage change of storage-cell in the open bitline is negligible. (ii) We write logic '1' to the neighboring bitlines of open bitline and expecting the storage-cell in open bitline to be driven to high voltage through the capacitive coupling between bitlines. We find that it can only be driven to less than $60mV$. If the initial voltage of the accessed cell in open bitline is logic '1', no matter what logic is written into the neighboring bitline, the voltage is reduced almost by half. Because both the parasitic capacity and the storage-cell capacity in the open bitline is quite large while the coupling effect from neighboring bitlines is negligible. In addition, the sense amplifier is shut down during write operation. As a result, the voltage drop in accessed cell is caused by discharging from accessed

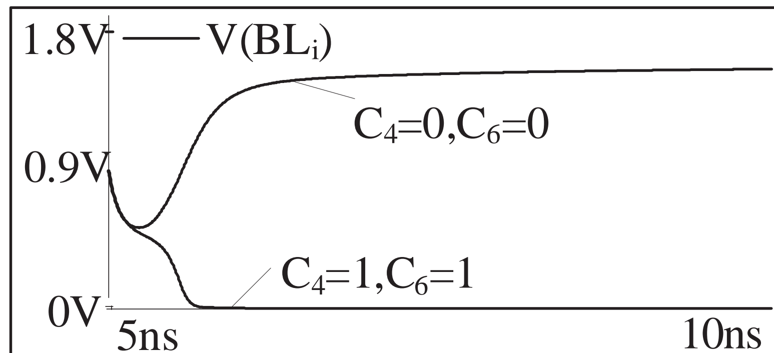


Figure 2.9: Read Operation with Bitline Open.

cell to parasitic capacitance of open bitline. In practical, both write operation on open bitline and its neighbor will not change the behavior of open bitline.

2.5.3 Simulation Results and Analysis for Read Operation

For case (ii) in Fig. 2.8, the simulation result shows that, when the neighbors BL_{i-1} and BL_{i+1} drive BL_i and $\overline{BL_i}$ with opposite logic values (one is '1' while the other is '0'), the voltage difference between BL_i and $\overline{BL_i}$ is amplified by sensing circuit, charging BL_i and $\overline{BL_i}$ to opposite logic values. The output of open bitline is reinforced by both its neighboring bitlines. We denote this fault behavior as a *compatible coupling*. When two neighbors, BL_{i-1} and BL_{i+1} drive BL_i and $\overline{BL_i}$ with the same logic value (both '1' or both '0'). From the simulation result (see Fig. 2.9), we know the output of open bitline BL_i is determined by C_6 since C_6 is coupled with $\overline{BL_i}$, the complemental bitline of BL_i. We denote the *driving force* as the capability to drive the open bitline through capacitive coupling. The larger distance between the aggressor bitline (TSVs) and the open bitline (TSV) is, the smaller coupling capacitance is. Since BL_{i-1} and BL_{i+1} have the same storage-cell value during the read operation, they pull up (or pull down) BL_i and $\overline{BL_i}$ at the same time. The coupling capacitance from BL_{i+1} to $\overline{BL_i}$ is larger than that from BL_{i-1} to BL_i, the driving force of BL_{i+1} is larger, making BL_{i+1} dominates the logic value in BL_i. We denote this fault behavior as a *Competitive coupling*.

We vary the coupling capacitance C_c from $0.1fF$ to $1fF$ and find similar result. The reason is that, the sense amplifiers are active during sensing phase of read operation. Although the coupling capacitance is small ($0.1fF$) between neighboring bitlines, any small voltage changes in open bitline generated by this coupling capacitance can be magnified by sense amplifier.

2.6 Simulation with TSV Coupling from Multiple Layers

Previous simulation studies assume no capacitive coupling from aggressor TSVs on multiple layers. In this section, we consider such coupling effect and present their faulty behaviors. Here, we only consider the case of wordline opens. This is because the faulty behavior of bitline opens are quite similar to what we have studied in Section 5. We just need to know the driving forces among all the neighbors of the open bitline. If the driving force for logic ‘1’ is larger than that for logic ‘0’, the open bitline read as logic ‘1’ , otherwise logic ‘0’.

2.6.1 Simulation Setup

Fig. 2.10(a) presents a possible wordline routing scheme from three vertically stacked memory layers, with its cut-view shown in Fig. 2.10(b). With TSV cou-

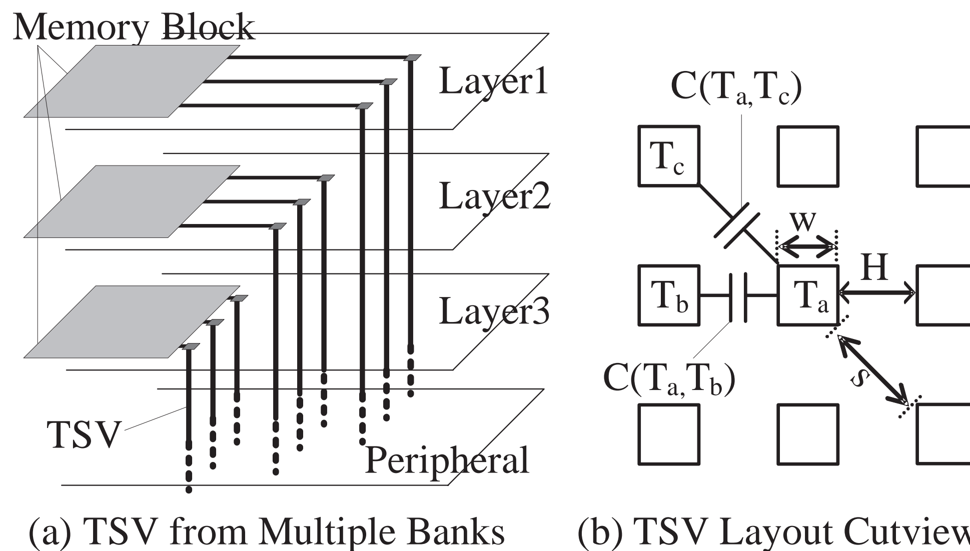


Figure 2.10: TSV Coupling from Multiple Layers.

pling from multiple layers, a *middle* TSV may have eight neighboring TSVs, in which four of them are at orthogonal direction, while the others are at diagonal direction (TSVs from the same layer are arranged in the same column, see Fig. 2.11(a) and Fig. 2.12(a)).

Using Aggressor-Victim Model [64], we can extend our earlier simulation model to consider such additional TSV coupling effects. That is, based on the capacitance extraction model presented in [67], we first calculate the sum of capacitance coupled to all possible neighboring TSVs driven by logic '1' and '0', respectively. Then we change the value of coupling capacitance parameters (C_1 and C_0) with the calculated values and conduct simulation using the same schematic circuit in Fig. 2.3. Since driving different logic values on these neighboring TSVs leads to different C_1 and C_0, without loss of generality, we consider two cases that may cause different faulty behaviors, when compared to the case that aggressor TSVs are from a single layer, as shown in Fig. 2.11-Fig. 2.12.

2.6.2 Simulation Results

As discussed in Section 4, when aggressor TSVs are from the same layer, accessing the open wordline behave as a stuck-open fault (only one wordline can be accessed in a memory bank). When the aggressor TSVs are from multiple layers, however, this is not true since wordlines from different layers can be accessed at the same time. To simulate this case, as shown in Fig. 2.11(a), for open TSV i, its two aggressor TSVs from other layers are driven to be logic '1' and we are interested in the behavior of the corresponding wordline WL_i.

Fig. 2.11(b) presents the voltage change of floating cell $Cell_0$ in WL_i when "0**w**1" and "1**w**0" are applied to wordline WL_i. As can be observed from the simulation results, even though this wordline is open, its storage cell can be written with logic '0' by repeating "**w**0" operations several times, but it cannot write logic'1'

into the storage cell due to the weak pulling up capability of pass transistors.

Fig. 2.11(c)-(d) shows the simulation results when we read the storage cell on open wordline under the same condition. We can get a correct result for "0**r**" operation. However, for "1**r**" operation, the result depends on the coupling capacitance. When the coupling capacitance is $1fF$, logic '1' can be successfully read out. When the coupling capacitance is $0.6fF$, however, we read an incorrect logic '0' out. This is because the low coupling capacitance drives insufficient voltage on the pass transistor and hence limits its conductivity to charge the bitline. Fig. 2.12 presents the simulation results for the faults shown in Fig. 2.6(a)-(b), in which the floating cell $Cell_0$ is in the open wordline WL_i while the accessed cell $Cell_1$ is on the accessed wordline WL_m.

Fig. 2.12(b) shows the voltage of bitline connecting to $Cell_1$ when we access

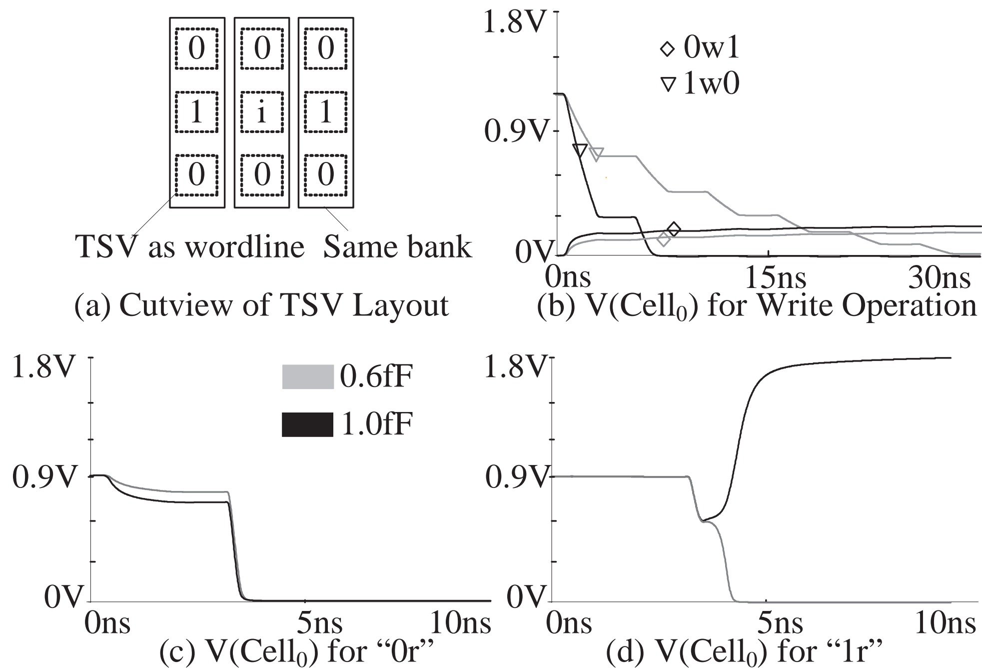


Figure 2.11: Case 1 for No Access Faults.

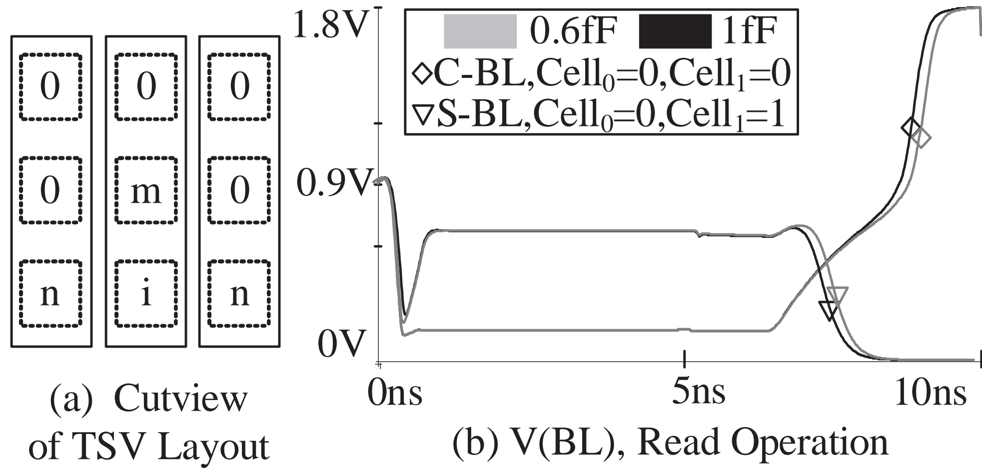


Figure 2.12: Case 2 for Wordline Multiple Access Fault.

WL_m, wherein both WL_n are also accessed at the same time. $C-BL, Cell_0$-0, $Cell_1$-0 represent that $Cell_0$ and $Cell_1$ are on complemental bitlines with logic value '0'. $S-BL, Cell_0$-0,$Cell_1$-1 represent that $Cell_0$ and $Cell_1$ are on the same bitline, with logic value '0' and '1', respectively. The results show that logic value read from bitline of $Cell_1$ is always opposite to the logic value stored in $Cell_1$, which leads to incorrect result.

2.7 Fault Modeling and Test Implications

In this section, we map the TSV open defect into memory functional fault models according to our simulation.

Wordline Open:

When we access the wordline with TSV open defects, we cannot access the corresponding cells, denoted as *No Access Behavior*. Consequently, write operation cannot change the values in the storage cell, and such fault behaves the same as a transition fault (TF_0 and TF_1). To sensitize these faults, we need to access its

neighboring wordline and repeatedly write the same logic value several times (the total number of writes depends on the coupling capacitance) to make sure the cell capacitor fully charged.

When we access the storage-cell (denoted as accessed cell) on the neighboring wordline of the open wordline, the floating cell on this open wordline is accessed too. This faulty behavior is defined as *Multiple Access Behavior*. The write operation in accessed cell forces the floating cell to be written as the same logic, which behaves like a coupling fault (*CF*). For read operation, the logic value in bitline (*BL*) and the content in accessed cell ($Cell_1$) are both incorrect (Fig. 2.6(a)-(b), *border* TSV Open), denoted as a *Read Disturb Faults* (*RDF*) [62]. To sensitize the above faults, we need to set the floating cell to be the opposite logic value of that in the accessed cell. Since every time we write logic value into the accessed cell, the floating cell is written with the same value due to capacitive coupling, making the above sensitizing condition unsatisfied. As a result, the *RDF* faults are difficult to detect when we only consider the coupling effects from aggressor TSVs originating from the same layer. It should be noted that, the refresh operation only affect the fault behavior when the neighboring wordline of *border* wordline is refreshed. Refreshing other wordlines will not change the fault behavior. Because, the neighboring wordline of the open wordline is now driven by '0', the capacitive coupling is too weak to drive the pass transistor in the open wordline.

Bitline Open:

Since the write operation with bitline opens has negligible impact on its accessed cell, we only model the fault of read operation with bitline opens. As discussed in section 2.4.3, there are two types of coupling faults caused by bitline with open TSV, denoted as compatible coupling faults and competitive coupling faults. Suppose BL_i is open and given the logic value in BL_{i-1} and BL_{i+1}, Tab. 2.1 presents the logic value of BL_i generated by capacitive coupling. Once a wordline

is selected, all storage-cells in that wordline are accessed. These accessed cells have two types of positions: either in bitlines (BL_1, BL_2,...,BL_i) or in complemental bitlines ($\overline{BL_1}$, $\overline{BL_2}$,...,$\overline{BL_i}$). As mentioned in section 2.4.3, compatible coupling faults are sensitized when two neighbors of open bitline (BL_i) has opposite logic value while competitive coupling faults are sensitized when these two neighbors has the same logic value (see Tab. 2.1 column 2, 4 in compatible coupling and column 6, 8 in competitive coupling). Take the first row in compatible coupling faults (see Tab. 2.1) as an example. When WL_1 is selected in Fig. 2.8, C_4, C_6 are accessed, driving BL_{i-1} and BL_{i+1} to logic '1' and '0' respectively. Then BL_{i-1} drives BL_i to logic '1' and BL_{i+1} drives $\overline{BL_i}$ to logic '0'. As a result, the output of BL_i is logic '1'. Other output of BL_i in Tab. 2.1 can be derived in a similar way.

The fault behavior in read operation with bitline open can be mapped to one of the functional memory fault model, that is, neighborhood pattern sensitive fault (NPSF). We develop three March algorithms targeting one these patterns. All the fault behavior in Tab. 2.1 can be detected by following three test algorithms:

$$\alpha : \{\Uparrow (w0); \Uparrow (r0, w1, r1)\}$$
$$\beta : \{\Uparrow (w1); \Uparrow (r1, w0, r0)\}$$
$$\gamma : \{\Uparrow (w1); \Uparrow (r1)\}$$

Considering the refresh operation, although voltage in open bitline is affected by its neighboring bitlines, the result is not sent out. During the refresh write-back

	Compatible Coupling				Competitive Coupling			
CellPosition	BL_{i-1}	(open)BL_i	BL_{i+1}	Test	BL_{i-1}	(open)BL_i	BL_{i+1}	Test
	1	***1***	0	α	1	***0***	1	γ
In *Bitline*	0	***0***	1	β	0	***1***	0	γ
	1	***0***	0	α	1	***0***	1	γ
In $\overline{Bitline}$	0	***1***	1	β	0	***1***	0	γ

Table 2.1: Fault Modeling for Read Operation with Bitline Open.

phase, the storage-cells in open bitline is not affected according to our simulation. As a result, the refresh operation dose not change the behavior of bitline open defects. It should be noted that we did not consider the column multiplexing as in [57]. Because for those bitlines not selected by decoder, the behavior just like the refresh operation (e.g. read phase, write-back phase). Whenever the bitlines are switched on or off, there is no difference.

Wordline and Bitline Open with TSVs from Multiple Layer:

To test RDF, we need to apply the following operations corresponding to Fig. 2.12(a). We first drive WL_n with logic '1' and WL_m with logic '1' to write logic '0' to $Cell_1$. Due to the TSV coupling effects, floating cell $Cell_0$ in WL_i is strongly driven to logic '0'. Then, we set both wordlines WL_n='0' and write logic '1' to $Cell_1$. Based on our simulation results, under the above condition, the floating cell $Cell_0$ in wordline WL_i can only be weakly pulled up (see Fig. 2.11(b)). Finally, we read $Cell_1$ out. If we obtain an incorrect value, the wordline is faulty; it is not otherwise.

Moreover, this crossing bank effect changes the normal memory test algorithms dramatically. That is because, in 2D memory, the test in different memory bank are independent, making it possible to test the memory chip in a highly parallel manner. However, in 3D memory, we have to write test pattern to memory cells in multiple banks to satisfy the test condition.

2.8 Conclusion and Future Work

3D-stacked memory is emerging as a promising solution to tackle the "Memory Wall" problem. The large number of TSVs implemented in 3D DRAM circuits, however, are prone to open defects and coupling noises, leading to new test challenges. In this chapter, we model the faulty behaviors of such defects with ex-

tensive simulation studies, map them into memory functional fault models, and present the corresponding test implications.

Since existing March algorithms are not able to detect many faults caused by TSV open defects. In our future work, we plan to develop new test algorithms and the associated BIST structure to address this problem. In addition, this work focuses on full open defects of TSVs. As TSVs can also be resistive open [62] and may exhibit different faulty behavior, we plan to also consider resistive open wordline/bitline in our future work.

□ **End of chapter.**

Chapter 3

3D-SoC Test Architecture Design and Optimization

3.1 Introduction

In this chapter, we design the test architecture for D2W/D2D 3D SoCs due to its high yield as mentioned in chapter 1. To get the information of the "Known good die" information, pre-bond wafer-level test is a good choice. After pre-bond test, those bad dies are discarded and only good dies are bonded. During the procedure of D2D/D2W bonding, however, the manufacture defects are also introduced. Furthermore, after packaging procedure, the functionality of the whole SoC need to be verified. As a result, the traditional package test is also necessary, denoted as **post-bond test**. The test architecture design and optimization problem for 3D SoCs is formulated by considering both post-bond test and pre-bond wafer-level tests. As the test time and routing compose the major part of the test cost [53], we first propose an test access mechanisms (TAM), wherein the test wires for post-bond package test can be fully reused for pre-bond test. Then, we propose efficient and effective heuristics to optimize the testing time and the routing cost associated with

the proposed TAM, based on simulated annealing technique.

In order to enable pre-bond tests and improve manufacturing yield for 3-D ICs, as introduced in Chapter 1, we have to fabricate additional test pads on the silicon die. However, the number of test pads must be limited due to the large area each test pad consumes. As a result, it is essential to take the pre-bond test-pin-count constraint into consideration during test planning. Our previously proposed TAM, however, tries to integrate pre-bond tests and post-bond test together and may lead to high test pad requirement for certain dies. To tackle the pre-bond test-pin-count constraint problems, in this chapter, we further propose to design the TAMs for pre-bond tests and post-bond test separately so that the test-pin-count constraint can be satisfied in pre-bond tests. By doing so, however, the routing cost for TAMs may be dramatically increased as pre-bond tests and post-bond test have different TAMs. To address this issue, we propose optimization methods that allow us to share routing resources between pre-bond tests and post-bond test as much as possible. Also, we show how to optimize test architectures to further reduce TAM routing cost with little impact on testing time.

The remainder of this chapter is organized as follow: Section 3.2 first introduces related works on this domain. In Section 3.3, the 3D SoC test architecture design and optimization is proposed to minimize the test time and routing cost. The test architecture design and optimization considering the pre-bond test-pin constraint is then addressed in Section 3.4. At last, we conclude this chapter in Section 7.

3.2 Preliminary

3.2.1 Prior Work in SoC Testing

When testing SoC devices, embedded cores are typically tested as separate, stand-alone units, wherein test wrappers are constructed to isolate them from the environment during test while dedicated test access mechanisms (TAMs) facilitate the transportation of test stimuli/responses between the core-under-test (CUT) and test source/sink (e.g., tester). Within a test wrapper, scan-chains (SCs) are the key components to deliver test stimuli and collect test response to/from the circuits while sophisticated registers and I/O controllers are used to facilitate the switching among normal functional mode and various test modes. These details have been standardized in IEEE P1500 [68] and are out of the scope of this thesis. However, it is important to note that the test application time of CUT depends on the longest wapper SCs, which in turn, if optimized, depends on the TAM bandwidth [69].

Among various TAMs proposed in literatures, dedicated bus-based TAM becomes the most popular test access mechanism. We use an example in Fig. 3.1(a) to demonstrate more details for the test bus based TAM [70]. The total TAM width W is distributed to two TAMs each of which has TAM width w_1 and w_2 respectively. TAM 1 connects $Core_1$, $Core_2$, and $Core_3$ while TAM 2 connect $Core_4$ and $Core_5$. The cores in different TAMs can be tested independently. Multiplexes are added to select which core is actually connected to the TAM output pins. Only one core can be accessed at a time. Hence the total test application time is the sum of all the individual core test application times.

Since both wrapper design and TAM optimization have direct impacts on the SoC test cost, they should be considered in conjunction to achieve the best result. We use the same example to demonstrate the importance of test architecture optimization integrating wrapper design and TAM optimization. Fig. 3.1 (a') shows the corresponding schedule, where x-axis denotes the test time and y-axis denotes the width of each TAM, and the total test time is determined by the maximum test

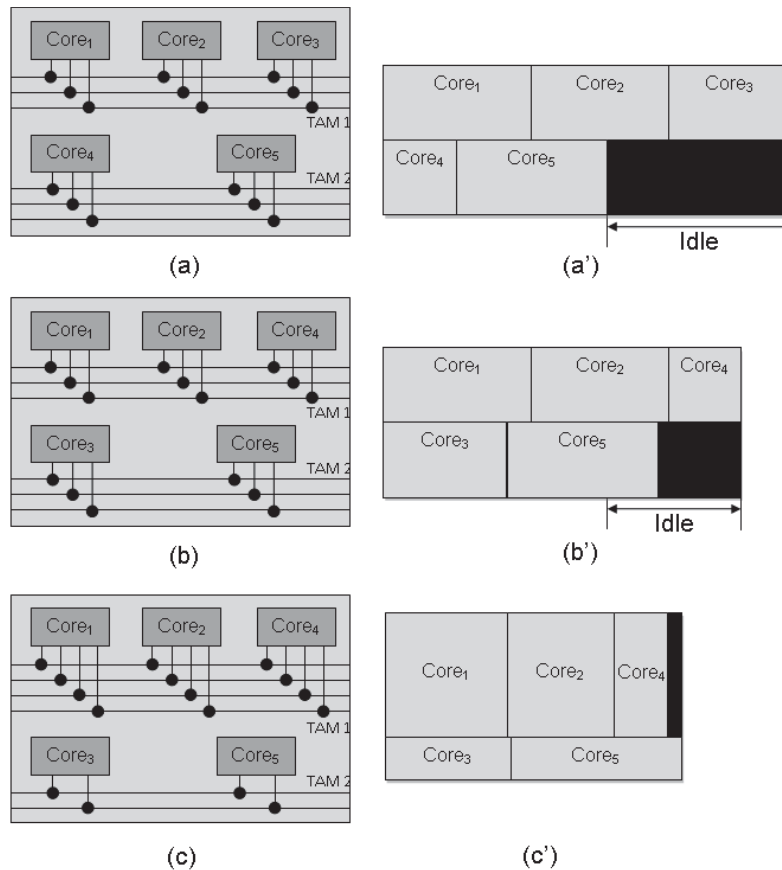


Figure 3.1: An Example for Bus-based TAM and its Optimization.

time among all TAMs. In fig. 3.1 (a'), there is idle time in TAM 2 which no test are applying, leading to a waste of test resource. In Fig. 3.1 (b), we swap the assignment of $Core_3$ and $Core_4$, leading to the reduction of total test time (see Fig. 3.1 (b')). Observing that the test time of TAM 1 is still larger than that of TAM 2, we take one test wire from TAM 2, and assign it to TAM 1, which decrease the test time of TAM 1 but increase the test time of TAM 2 (see Fig. 3.1 (c)). However, the total test time is further reduced (see Fig. 3.1 (c')). Through this example, we know that a good test architecture can significantly reduce the test application time.

For an SoC with specified parameters for its cores, a test architecture and a test schedule need to be designed to minimize the test cost, which must account for the test application time and design-for-testability (DfT) overhead. The optimization of the above modular test architectures and test scheduling have been subject to extensive research [71].

3.2.2 Prior Work in Testing 3D ICs

Test techniques and design-for-testability (DfT) solutions for 3D ICs are critical issues for the success of 3D technology, as pointed out in [72, 27]. However, only limited work has been done in this emerging area.

Lewis and Lee [25] proposed a scan-island-based design to enable pre-bond tests for incomplete circuits at the architecture level. Wu *et al.* [30] studied several scan chain design approaches for 3D ICs and compared their routing costs. The above works mainly target 3D ICs that put functional blocks in different silicon layers.

For 3D SoCs with entire embedded cores on different layers, modular testing is an attractive solution as it facilitates the reuse of test patterns. While test architecture design and optimization for two-dimensional SoCs have been subject to extensive research [71], these solutions are not readily applicable for testing 3D SoCs. Marinissen *et al.* [31] proposed a novel test wrapper design for cores in 3D SoCs, but it did not address the test architecture optimization problem. Recently, a test architecture optimization technique was proposed in [73] to minimize the testing time of 3D SOCs, under limits on the number of TSVs utilized by TAMs. However, pre-bond tests were not considered in this work and hence it can only provide cost-effective solutions for 3D SoCs manufactured with W2W bonding technology. Noia *et al.* [74] considered the case with given fixed or yet-to-be-designed test architecture on each die, which is able to optimize for post-bond test

only.

3.3 On Test Time and Routing Cost

The remainder of this section is organized as follows. Section 3.3.1 motivates this section. The test architecture design and optimization problem for test time and routing cost reduction is then formulated in Section 3.3.2. In Section 3.3.3, we detail our simulated annealing-based solution for the above problem. Experimental results on benchmark circuits are then shown in Section 3.3.4.

3.3.1 Motivation

We use the following motivating example to illustrate the problem investigated in this section. Consider a core-based 3D SoC as shown in Fig. 3.2, wherein six cores are placed on two silicon layers. With a traditional 2D test architecture optimizer that tries to minimize the testing time in post-bond test, we obtain the TAM architecture shown also in Fig. 3.3. That is, there are totally three TAMs for this example SoC: TAM_1 for core 5, TAM_2 for cores 1, 2, and 3, and TAM_3 for

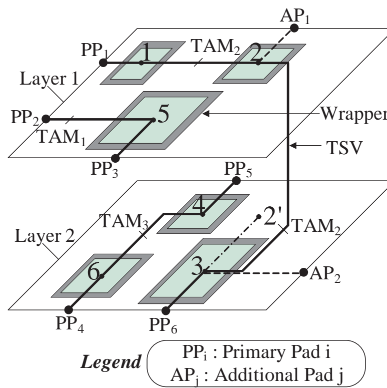


Figure 3.2: An Example 3D SoC Test Architecture Optimized for Post-bond Test Only.

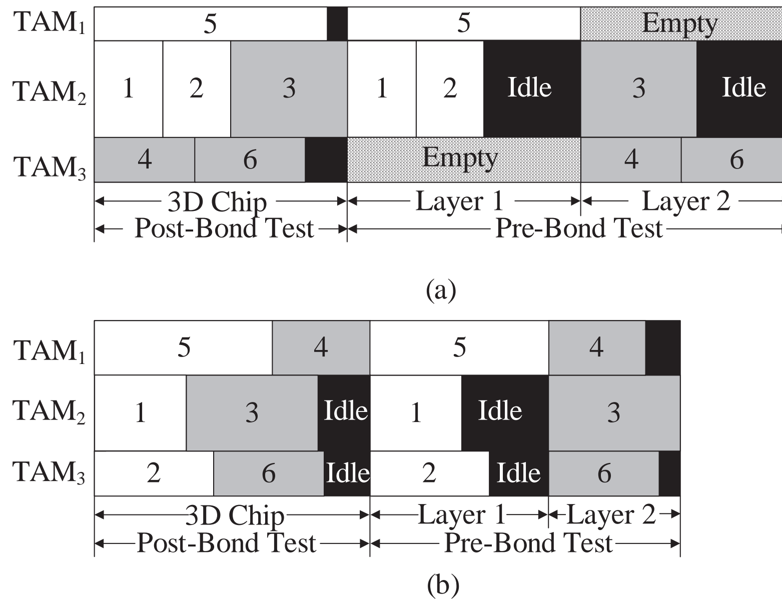


Figure 3.3: The Impact of Pre-Bond Tests.

core 4 and core 6. In particular, TAM_2 traverses two layers in this example.

When pre-bond tests at the wafer-level are required, however, the test cost model for the 3D SoC changes. For instance, the testing time of the chip is the sum of each layer's pre-bond testing time and the post-bond testing time of the entire chip. That is, for the example shown in Fig. 3.2, it contains three parts: the pre-bond testing time for layer 1, the pre-bond testing time for layer 2, and the post-bond testing time for the entire chip, represented as three bins in Fig. 3.3(a), respectively. The cores in different layers are shown in different gray scales, and the TAM can be empty if no cores in that layer are assigned to it. From this figure, it is obvious that the test architecture optimized only for post-bond test in 3D SoCs incurs long idle time on their pre-bond tests (see TAM2 on layer 1 and layer 2). If we are able to design a 3D-aware test architecture as shown in Fig. 3.3(b), the total testing time for the 3D SoC can be dramatically reduced, although the testing time of the post-bond test becomes longer. In addition, the routing cost associated with TAMs for 3D SoCs is also different from that of the planar 2D SoCs, as TAMs

can use TSVs to go through several layers. Therefore, we should add additional TAM wires (dashed lines) and additional test pads (AP_1, AP_2) for those incomplete TAMs during pre-bond test (TAM 2), as shown in Fig. 3.2.

The above motivates us to investigate the test architecture design and optimization problem for D2W/D2D bonding fabricated 3D SoCs, as formulated in the following section.

3.3.2 Problem Formulation

3.3.2.1 Test Cost Model

The test cost model for 3D SoCs to evaluate different test architectures is shown in the following.

$$C_{total} = C_{Test_Time} \times \alpha + C_{Wire_Length} \times (1-\alpha) \quad (3.1)$$

where, C_{Test_Time} is the total testing time for both pre-bond tests and post-bond test, while C_{Wire_Length} is the total TAM wire length. α is a weighting factor designated by users. For the example test architecture and the associated test schedule shown in Fig. 3.3(a), C_{Test_Time} is the sum of three terms: $(T_1+T_2+T_3)$ for post-bond entire chip, T_5 for pre-bond layer 1 and (T_4+T_6) for pre-bond layer 2, where T_i is the testing time of core i.

The computation of C_{Wire_Length}, however, is non-trivial. In this section, we assume a TAM involved in several layers will route through all cores tested with this TAM on one layer before it goes through TSVs to connect cores in other layers. Accordingly, we calculate C_{Wire_Length} as follows.

C_{Wire_Length} for a TAM that involves several layers contains two parts: the intra-layer wire length and the inter-layer one. For the former one, the TAM is broken into several segments, each on a single layer. For each segment (or TAM that is on one layer only), we use the algorithm in [75] to compute its wire length[1]. As for

[1]Note that additional pads (APs) and wires (dash lines in Fig. 3.2) for test may be necessary when reusing chip-level TAMs for wafer-level testing.

the inter-layer wire lengths, they are calculated as the Manhattan distance between the corner cores in different layers, e.g., for TAM_2 in Fig. 3.2, the inter-layer wire length is the Manhattan distance between core 3 and the core 2 mirrored on layer 2 (i.e., the dot dash line). The wire length for TSVs is ignored due to their tiny sizes.

It is important to note that our proposed algorithms (detailed in Section 3.3.3) can be applied to other cost models as well. For example, if a different TAM routing strategy is used [30], partial pre-bond testing is applied [76] or multi-site testing is considered [77]. Designers can just update the above test cost model accordingly and apply our proposed technique.

3.3.2.2 Problem Definition

The problem addressed in this section can be formulated as follows:

Problem: Given

- a set of cores C, and the test parameters for each core $c \in C$, that is, the number of wrapper input cells in_c, the number of wrapper output cells out_c, the number of wrapper bidirectional cells bi_c, the number of test patterns p_c, the number of internal scan chains sc_c, and for each scan chain i, the lengths of scan chain in flip-flops $l_{c,i}$;
- the physical position of every core c, i.e., which layer it sits on and its X-Y coordinate on that layer;
- the maximum available TAM width W_{TAM};

Determine the number of TAMs, the cores assignment associated with each TAM, and the width of each TAM to minimize the total test cost as shown in Eq. (1). Note, test wrapper design and optimization is a subproblem of the above problem, and we use the algorithm in [78] to address it.

3.3.3 Proposed Algorithm

In this section, we introduce the proposed simulated annealing-based algorithm for tackling the problem presented in Section 3.3.2.2. Here, we mainly consider the TestBus architecture for the sake of simplicity. The proposed method, however, can be easily extended to a TestRail architecture.

3.3.3.1 Outline of The Proposed Algorithm

The test architecture design and optimization problem for 2D SoCs has been proven to be a NP-hard problem [78]. To reduce computational time, prior work mainly resorts to deterministic heuristics to address this problem (e.g., [79, 80, 81, 82]). As shown earlier, for 2D SoCs without considering pre-bond test, generally one single TAM is the bottleneck that determines the SoC testing time. Consequently, greedily optimizing the bottleneck TAM by assigning cores to different TAMs and/or allocating more TAM wires to the bottleneck TAM can lead to close-to-optimal solution [79]. The above deterministic optimization strategies, however, are difficult to apply to optimize 3D SoC test architectures as we need to consider both pre-bond tests and post-bond test, which can have multiple bottleneck TAMs in terms of testing time (e.g., TAM_1 for layer 1 pre-bond test and TAM_2 for post-bond test in Fig. 3.3(a)). We therefore propose to use simulated annealing (SA) based stochastic search algorithms to tackle the problem described in Section 3.2.

One of the most straightforward methods to address this problem is then to construct a unified solution representation including both core assignment and TAM width allocation, and perform simulated annealing on it. That is, we can represent a solution as a few core sets and a TAM width for each set. This method, in spite of its simplicity, is not quite effective due to the huge solution space and the difficulty to specify neighboring solutions in the SA process.

Fortunately, we notice that, given a fixed core assignment for each TAM, it is easy to determine close-to-optimal TAM widths for each TAM by properly ad-

```
1  Set TAM_Num_min and TAM_Num_max
2  For Tam_Num = TAM_Num_min to TAM_Num_max
       // Simulated Annealing
3      Get initial core assignment with no empty TAM
4      Perform inner TAM width allocation algorithm
       and compute initial cost
5      Cost_best ← initial cost, Cost_old ← initial cost
6      Set temperature T as a high value
7      While (T > end temperature T_end)
           // Run a few iterations at same T
8          For each iteration
9              Random move to reach a new core assignment
10             Perform inner TAM width allocation
11             Compute Cost_new
12             If Cost_new < Cost_old or e^(ΔCost/T) > rand()
                   // Accept Move
13                 Cost_old ← Cost_new
14                 Update core assignment solution
15                 If Cost_new < Cost_best
16                     Cost_best ← Cost_new
17                     Record the best solution
18             Else
19                 Restore old solution
20         Decrease temperature T
```

Figure 3.4: Main Flow of the Proposed Algorithm.

justing existing deterministic heuristics. Based on this observation, for a given number of TAMs, we propose to separate our optimization procedure into two

nested parts: the *outer SA-based core assignment* (Section 3.3.3.2) and the *inner heuristic-based TAM width allocation* (Section 3.3.3.3). By doing so, the solution space explored by simulated annealing shrinks to the various core assignment solutions only, without loss of its quality. The above outer and inner procedures are for a given number of TAMs, we therefore need to enumerate this value in our algorithm.

The overall algorithm to tackle 3D SoC test architecture design and optimization problem is shown in Fig. 3.4. In the beginning, we set the minimum number of TAMs (TAM_Num_{min}) and the maximum number of TAMs (TAM_Num_{max}) to be explored in our algorithm. Typically, $TAM_Num_{min} = 1$ while TAM_Num_{max} is set to be a small number that is much less than $\min\{|C|, W_{TAM}\}$[2]. Then, for a given number of TAMs, we start from a random initial core assignment, and keep on searching for its neighbor solutions. Once a feasible solution is obtained, TAM width allocation is conducted by inner deterministic algorithm. Core assignment solutions are evaluated using the cost model in Section 3.3.2.1. Then algorithm conducts like a typical simulated annealing algorithm. Finally, the algorithm outputs the best solution that is obtained during the stochastic search process.

3.3.3.2 SA-Based Core Assignment

Suppose we are performing core assignment for m TAMs. Remind there are totally $|C|$ cores to be tested, the problem comes down to dispatching cores to m sets. As a consequence, a solution can be represented as a series of sets $\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_m$, where $\mathcal{A}_i$ is the core set assigned to TAM i. For instance, assume there are two TAMs and five cores, a valid solution can be $\{(1,3),(2,4,5)\}$, meaning that cores 1 and 3 are assigned to one TAM, and cores 2, 4, and 5 are assigned to the other one.

$\{(1,3),(2,4,5)\}$ and $\{(2,4,5),(1,3)\}$ are both valid representations, but they

[2] A large number of TAMs typically results in excessive testing time and hence is not preferred.

essentially correspond to the same solution. To eliminate this redundancy and provide a one-to-one mapping between a representation and its corresponding solution, we always keep the smallest core index assigned to TAM i smaller than that assigned to TAM j, provided i is smaller than j. Let α_i be the minimum core index of TAM set $\mathcal{A}_i$. This rule can be expressed as $\forall i < j : \alpha_i < \alpha_j$. According to it, $\{(2,4,5),(1,3)\}$ will be deemed as an invalid solution in our annealing process. With the help of the above rule, the solution space shrinks to $\frac{1}{m!}$ of that without this rule. Also, we do not allow empty sets, because any solution with n empty sets achieved in the iteration where TAM number is set to be m can be revisited in the iteration where TAM number is $(m-n)$ without empty sets.

The only move defined in our procedure to find a neighbor solution is M1: pick up a core from a random set $\mathcal{A}_i$ which contains more than one cores, and put it into another randomly selected set $\mathcal{A}_j$. The completeness of the above move can be effectively proved by: starting from a valid solution $\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_m$, we are able to reach any other solution $\mathcal{B}_1, \mathcal{B}_2, \cdots, \mathcal{B}_m$ after finite times of M1 move.

3.3.3.3 Heuristic-Based TAM Width Allocation

With given number of TAMs and the core assignment for each TAM, it is actually possible to obtain the optimal TAM width allocation using techniques such as linear programming (e.g., [78]). The inner TAM width allocation procedure however needs to be called every time we have a feasible core assignment solution during the simulated annealing process. Consequently, the running time for this inner procedure needs to be very short so that we can explore a large number of core assignment solution space. Because of this, instead of acquiring exact optimal solution for the inner TAM width allocation process, we use a greedy heuristic as shown in Fig. 3.5 to obtain a close-to-optimal solution. Similar to [79], this procedure iteratively assign one-bit wire ($b = 1$) to a TAM that leads to the lowest total test cost (Line 6). If this one-bit TAM wire cannot result in cost reduction,

```
1   Allocate one bit width to every TAM
2   Set b = 1
3   While no more unassigned TAM width
4   Cost_min ← ∞
5       For each TAM
6           Allocate b bit width to this TAM;
7           Compute the cost of entire TAM architecture
8           If Cost < Cost_min
9               Cost_min = Cost
10              Keep this TAM as the only candidate
11          Restore this b bit width
12      If Cost_min reduces
13          Allocate b bit to the recorded TAM
14          Set b = 1
15      Else
16          Increase b by one
```

Figure 3.5: Inner TAM Width Allocation Procedure.

we will not allocate it in this iteration (Line 11). Instead, we increase the width of the to-be-assigned TAM wire by one until a lower cost is found (Lines 12-16).

3.3.4 Experiments

To demonstrate the effectiveness of the proposed solution for testing 3D SoCs, we present experimental results for two ITC'02 benchmark SoCs[3] (p22810 and t512505). We map these two SoCs onto three silicon layers randomly and try to balance the total area of each layer, where a core's area is estimated based on

[3]Due to space limitation, results for other benchmark circuits are not reported here, but they show similar trends.

Width	p22810									
	Total Testing Time			Ratio (%)		Wire Length			Ratio (%)	
(bit)	TR-1	TR-2	SA	Δ_1^T	Δ_2^T	TR-1	TR-2	SA	Δ_1^W	Δ_2^W
16	1888667	1327398	1062281	43.75	14.04	5386	13694	12095	-120.66	13.21
24	1302783	1046844	780763	40.07	20.42	7729	17041	14614	-104.41	7.29
32	1031366	836821	627148	39.19	20.33	7693	15051	18062	-126.97	-16.01
40	802558	723546	534329	33.42	23.58	8582	21328	24926	-124.91	9.50
48	703142	639377	500868	28.77	19.70	9184	17380	32084	-245.90	-82.78
56	593840	562916	435664	26.64	21.43	9709	22240	37302	-333.20	-89.11
64	527732	490439	421677	20.10	13.03	10560	26862	35286	-241.11	-34.10

Width	t512505									
	Total Testing Time			Ratio (%)		Wire Length			Ratio (%)	
(bit)	TR-1	TR-2	SA	Δ_1^T	Δ_2^T	TR-1	TR-2	SA	Δ_1^W	Δ_2^W
16	47658726	30913144	27190110	42.95	7.82	2743	4560	13234	-382.46	-190.22
24	34450110	30758094	26778656	22.27	11.55	5459	5824	21245	-289.17	-264.78
32	33867447	18036401	17510811	48.30	1.55	4596	11590	8621	-87.58	25.62
40	23417347	18890284	13028909	44.36	25.03	6313	8460	17540	-177.84	-107.33
48	23417347	18890284	12967247	44.63	25.29	7171	9724	11775	-64.20	-21.09
56	23417347	18890284	12967247	44.63	25.29	7797	10988	13605	-74.49	-23.82
64	23417347	18890284	12967247	44.63	25.29	8985	12252	15846	-76.36	-29.33

Δ_1^T / Δ_2^T: Difference ratio on total testing time between SA and TR-1 / TR-2;
Δ_1^W / Δ_2^W: Difference ratio on wire length between SA and TR-1 / TR-2.

Table 3.1: Experimental Results for $\alpha = 1$.

the number of internal inputs/outputs and scan cells (if any). An academic floorplanner is utilized to get the coordinates for each core, to be used for wire length calculation. As mentioned in Section 3.3.3, we focus on the TestBus architecture in our experiments. Other benchmarks, such as p93791, have similar results.

We compare the proposed algorithm with two baseline solutions, constructed from a traditional 2D optimization algorithm TR-ARCHITECT [79]. In the first one (referred as TR-1), we apply TR-ARCHITECT algorithm to the 3D SoC layer by layer, i.e., no TAM wires is allowed to traverse multiple silicon layers, and we adjust the TAM width among layers iteratively until the testing time of these layers are as balanced as possible. In the second baseline solution (referred as TR-2), we simply apply TR-ARCHITECT algorithm to the whole 3D chip, minimizing the testing time of the post-bond test. In this section, we do not compare against [73] because [73] is essentially an optimizer for post-bond test with TSV constraints, but we do not consider TSV as constraints in our work. Also, the layout used in [73] is not available to us.

Table 3.1 presents our experimental results when considering testing time only. As expected, our algorithm leads to significant improvement in terms of testing time reduction when compared with the other two baseline solution. The benefit can be as high as 48.3% compared with TR-1, and 25.3% compared with TR-2 for SoC t512505. TR-1 leads to longer testing time because TAMs are not allowed

	$\alpha = 0.6$										$\alpha = 0.4$									
Width	Total Testing Time			Ratio (%)		Wire Length			Ratio (%)		Total Testing Time			Ratio (%)		Wire Length			Ratio (%)	
(bit)	TR-1	TR-2	SA	Δ_1^T	Δ_2^T	TR-1	TR-2	SA	Δ_1^W	Δ_2^W	TR-1	TR-2	SA	Δ_1^T	Δ_2^T	TR-1	TR-2	SA	Δ_1^W	Δ_2^W
16	47658726	30913144	25854529	45.75	10.61	2743	4560	7272	-165.11	-59.47	47658726	30913144	27556426	42.18	7.04	2743	4560	3957	-44.26	13.22
24	34450110	30758094	24742962	28.18	17.46	5459	5824	4714	13.65	19.06	34450110	30758094	28320290	17.79	7.08	5459	5824	4144	24.09	28.85
32	33867447	18036401	14449906	57.33	10.59	4596	11590	9933	-116.12	14.30	33867447	18036401	27355499	19.23	-27.52	4596	11590	3986	13.27	65.61
40	23417347	18890284	13359723	42.95	23.62	6313	8460	7354	-16.49	13.07	23417347	18890284	27360934	-16.84	-36.17	6313	8460	4029	36.18	52.38
48	23417347	18890284	13359680	42.95	23.62	7171	9724	7440	-3.75	23.49	23417347	18890284	27350658	-16.80	-36.13	7171	9724	4175	41.78	57.06
56	23417347	18890284	13361473	42.94	23.61	7797	10988	7470	4.19	32.02	23417347	18890284	27350409	-16.80	-36.13	7797	10988	4102	47.39	62.67
64	23417347	18890284	13372901	42.89	23.56	8985	12252	7374	17.93	39.81	23417347	18890284	28867718	-23.27	-42.61	8985	12252	4043	55.00	67.00

Table 3.2: Experimental Results for SoC t512505 Considering Both Testing Time and Wire Length.

to walk through different layers, which significantly constrains the solution exploration space. At the same time, as TAM wire length is not considered in this experiment (i.e., $\alpha = 1$), typically long TAM wires are obtained using our algorithm, especially when compared to TR-1.

When the total TAM width gets larger, the total testing time decreases for p22810. However, for t512505, after the TAM width is larger than 40, its testing time does not decrease any more, mainly due to a bottleneck core in the system. Also, we can observe that when TAM width increases to be more than 32, the testing time of TR-2 becomes even higher, mainly due to the increase of its pre-bond testing time.

Fig. 3.6 shows the testing time of the pre-bond test for each layer and the post-bond test for the entire chip for SoC p22810. With TR-1, we can observe balanced testing time among all layers (as expected), while the other two algorithms do not have this feature. Compared to TR-2, the proposed algorithm often has longer testing time for the post-bond test, but achieves a much shorter testing time in the pre-bond tests (e.g., when TAM width is 16), thus resulting in improved total testing time.

In Table 3.2, we present experimental results for SoC t512505 with two sets of weight parameter: $\alpha = 0.6$ and $\alpha = 0.4$, in which the former one is associated with balanced testing time and TAM wire length cost factors while the latter one

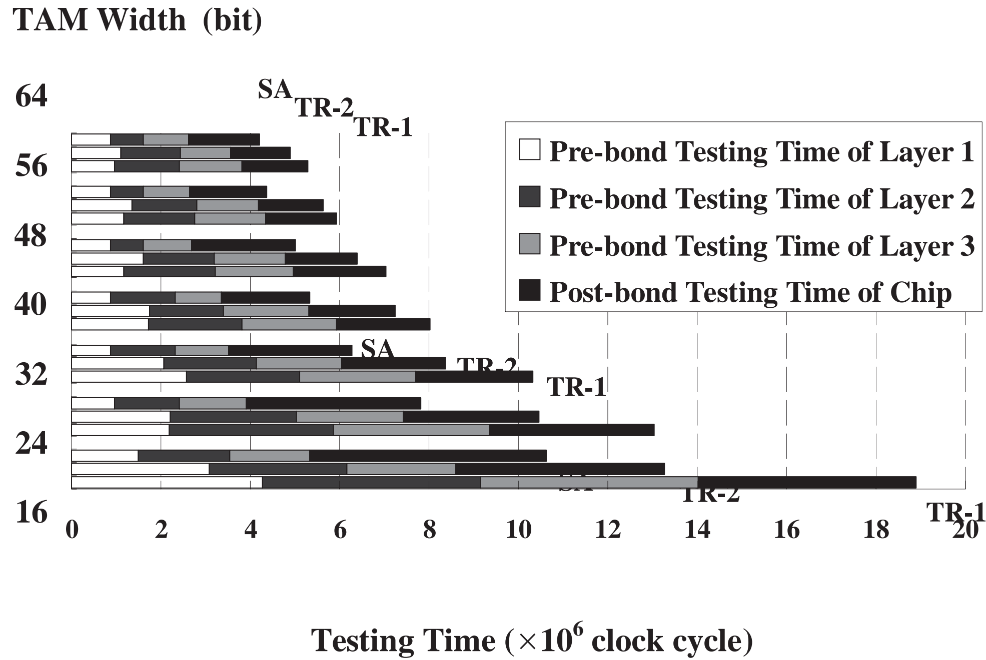


Figure 3.6: Detailed Testing Time of p22810.

emphasizes more on the impact of wire length cost.

As can be observed from this table, for the former case, the TAM wire length of the proposed solution is still higher than that of TR-1 in many cases, but already much smaller than the case in Table 1. In several cases (when TAM width is 24, 56 and 64), we can achieve both testing time improvement and wire length reduction.

As the total TAM width increases, the total testing time of the proposed algorithm declines first, and then increases (see Column 4). We attribute it to the fact that wire length accounting for more share in the cost function with the increment of TAM width.

For the latter case, where wire length has a much heavier weight, since the wire length increases dramatically with TAM width increment, when the TAM width is large, we can achieve much shorter wire length compared to that with TR-1 or TR-2. For instance, when TAM width is 64, the difference ratio on wire length

between the proposed algorithm and TR-1 / TR-2 is as high as 55.00% / 67.00%.

Finally, the computational time of the proposed technique is in the range of minutes for all our experiments, and the number of TSVs remains in the magnitude of tens. Both are acceptable for test architecture design and optimization.

3.4 Pre-bond Test-Pin Constrained TAM Sharing

The remainder of this sections is organized as follow: Section 3.4.1 demonstrate the cause of test-pin constraint. We motivate this section in Section 3.4.2. The problem is then formulated in Section 3.4.3. Next, our proposed test wire sharing scheme, and layout-driven test architecture design and optimization techniques are shown in Section 3.4.4 and Section 3.4.5, respectively. Section 3.4.6 presents our experimental results for benchmark circuits.

3.4.1 Pre-Bond Test-pin Constraint

When conducting pre-bond tests for silicon dies at wafer-level, one of the biggest challenges is how to probe the silicon die effectively. As shown in [83], since fine-grained touchdown probe needles are not available in the next decade, producing dense probe arrays to connect to the ATE is not a viable solution, at least for the near future. Consequently, we have to fabricate test pads (C4 bump or wire bond, see Fig. 3.7) on silicon dies and rely on conventional probing techniques to connect them to the ATE during pre-bond testing [72]. At the same time, however, it is not possible to fabricate a large number of test pads for pre-bond testing in 3D ICs. This is because of the following reasons. According to [83], the pitch for C4 bumps is around 120μm, which is much larger than that of TSV (1.7μm as shown in [84] and this figure keeps shrinking with technology improvements). In other words, one single test pad can consume area equivalent to hundreds of TSVs (see Fig. 3.7). As these test pads have to be put at the “keep-out area” for TSVs (i.e., TSVs need to keep some distance from any other component), the benefits of exploiting dense TSVs for interconnecting active devices between layers are significantly diminished with the increase of test pads [24]. In 3D technology, except for the bottom layer, the silicon bulks in other layers are thinned for the ease of TSV

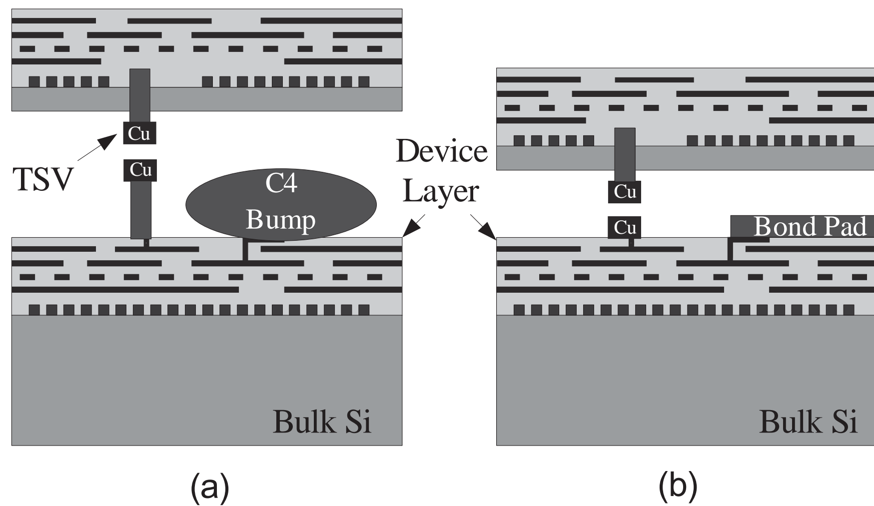


Figure 3.7: Pre-Bond Test Pad: (a) C4 Bump as Test Pad, (b) Wire-Bond as Test Pad.

fabrication. If we conduct pre-bond tests before thinning, we may not be able to detect the failures introduced during the chemical mechanical polishing (CMP) process. If, however, we probe the thinned wafer instead, the probe force (typically $3-10g$ per probe and $60-120kg$ per wafer) during testing becomes a serious concern as these thinned wafers are not mechanically strong enough. Again, it is desired to have less test pads (probes/touchdowns) for silicon dies in pre-bond testing.

3.4.2 Motivation

In Section 3.3, the same TAMs that traverse multiple layers in post-bond testing are fully reused for pre-bond tests. Consequently, TAMs can be divided into multiple parts and distributed among the different silicon layers. As all the TAM segments in a particular silicon layer need to be probed during pre-bond testing, a large amount of test pads may be required for those silicon dies that contain many TAM

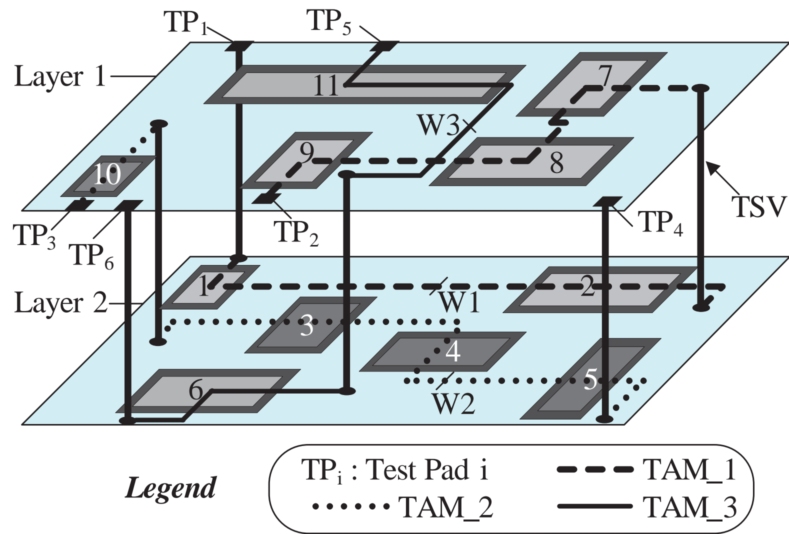


Figure 3.8: Test Architecture for an Example 3D SoC.

segments. This can be a serious issue in pre-bond testing.

The most straightforward solution to take pre-bond test-pin-count constraint into consideration during the 3D SoC test architecture design and optimization process is to design *separate* test architectures for pre-bond tests and post-bond test. By doing so, however, the total TAM routing cost for 3D SoCs can be quite high as we have dedicated TAMs for pre-bond tests, resulting in degradation of the chip's routability. As we need to link cores using both pre-bond TAMs and post-bond TAMs and they are used at different times, a natural question is whether we can share some of the routing resources between the two types of TAMs.

We use the following example to demonstrate the possibility of sharing routing resources and its potential benefits. Consider a two-layer 3D SoC containing 11 cores, in which six of them (C1 to C6) are on the bottom layer while the other five cores (C7 to C11) are on the top layer. Similar to Section 3.3, for the sake of TSV count consideration, we assume a post-bond TAM involved in several layers will route through all cores tested with this TAM on one layer before it goes through

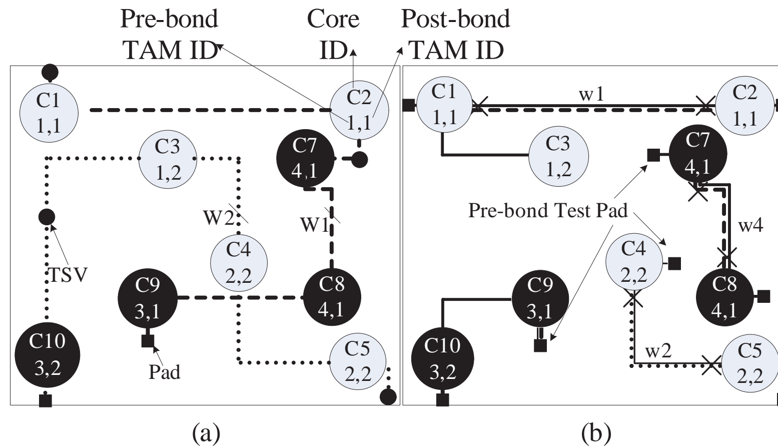


Figure 3.9: Routing Resource Sharing Example: (a) Test Architecture During Post-Bond Test, (b) Reuse TAM During Pre-Bond Test.

TSVs to connect cores in other layers. In this example 3D SoC, three TAMs are used for post-bond testing and they are shown in Fig. 3.8. As an example, TAM_1 connects C1, C2, C7, C8, and C9 with TAM width $W1$, starting from test pad TP_1 and ending at test pad TP_2.

For the ease of discussion, we map a few cores in the 3D SoC onto one layer as shown in Fig. 3.9(a). In this figure, each vertex represents a core, in which the upper label is the core ID, while the lower one denotes the pre-bond TAM ID and post-bond TAM ID that this core belongs to. In Fig. 3.9(b), we show how pre-bond TAMs can reuse the existing test wires for post-bond testing, wherein the solid lines are pre-bond TAMs. It can be easily observed that those wires having both solid and dashed/doted lines can be shared between pre-bond test and post-bond test, which can significantly reduce the total routing cost for TAMs in 3D SoCs. Note that, during pre-bond test, the end points of each TAM are directly routed to deliver test data on its own silicon layer. Here, we assume that these test pad is near the end point, so that we can ignore the distance between end points and test pads.

Clearly, some design-for-testability (DfT) circuitries need to be introduced to enable the routing resource sharing between pre-bond test and post-bond. To be specific, we need: (i) certain multiplexers to select the different test data source for pre-bond test and post-bond test (see the "×" point shown in Fig. 3.9(b)); (ii) reconfigurable test wrappers for cores that have different TAM width between pre-bond test and post-bond test (e.g., [85, 82]); (iii) the necessary control mechanisms (typically by introducing extra instructions in test wrapper and JTAG controller).

3.4.3 Problem Formulation

Problem: Given

- the set of cores C on the 3D SoC, and the test parameters for each core $c \in C$;
- the layout of the 3D SoC, i.e., the physical position of every core c, including which layer it sits on and its X-Y coordinate on that layer;
- the maximum available TAM width for post-bond test W_{post};
- the pre-bond test-pin-count constraint W_{pre};

Our objective is to determine the number of pre-bond TAMs N_{pre}, the number of post-bond TAM N_{post}, the core assignment associated with each TAM t_i, and the width of each TAMs W_{t_i}, so that the total testing cost, $C_{total} = C_{time} \times \alpha + C_{route} \times (1-\alpha)$, is minimized ($\alpha$ is a weighting factor designated by users). Note thatC_{time} and C_{route} represent the total testing time of the 3D SoC and the total TAM wire length for the 3D SoC, respectively.

For the case that the pre-bond TAMs and post-bond TAMs are not shared, the routing cost C_{route} is simply the total wire length of the two kind of TAMs, that is,

$$C_{route} = \sum_{i=0}^{i<N_{pre}} W_{t_i} \times L_{t_i} + \sum_{i=0}^{i<N_{post}} W_{t_i} \times L_{t_i} \tag{3.2}$$

Here, L_{t_i} denotes the wire length for TAM t_i, and we calculate it using the sum of Manhattan distance between adjacent cores in this TAM.

When considering the sharing of pre-bond TAMs and post-bond TAMs, suppose the total length for the shared wires is C_{reused}. The routing cost C_{route} becomes

$$C_{route} = \sum_{i=0}^{i<N_{pre}} W_{t_i} \times L_{t_i} + \sum_{i=0}^{i<N_{post}} (W_{t_i} \times L_{t_i}) - C_{reused} \tag{3.3}$$

It is worth noting that test wrapper design and optimization is a subproblem of the above problem, and we use the algorithms in [78, 85] to optimize the IEEE Std. 1500-compliant test wrapper and the reconfigurable test wrapper, respectively.

3.4.4 Test Wire Sharing

It is not trivial to solve the above problem due to its complexity. Thus, in this section, we address the test wire sharing scheme, provided that the TAMs for post-bond test and pre-bond test are already given. The first step of our test wire sharing scheme is to derive the post-bond TAM routing, after which, we try to identify those wire segments in post-bond TAM that can be reused by pre-bond TAM. At last, we propose a greedy heuristic for pre-bond TAM routing to minimize its routing cost, leveraging the wire sharing.

3.4.4.1 Post-Bond TAM Routing

Given a post-bond TAM with several cores tested on it, we first map these cores belonging to different layers onto one virtual layer (as shown in Section 2.3). For the connections that link two cores on different layers, we can ignore the routing cost for the TSVs due to its short length. Given the test architecture, we are to route all the cores belonging to the same TAM sequentially, and hence the routing problem is equivalent to the NP-Hard Traveling Salesman (TSP) problem [75].

To tackle this problem, we first construct a complete graph for all the cores (vertices) belonging to the TAM and the weight of each edge represent its routing cost (i.e., the distance between the two cores multiply the TAM width) between the linked two cores (denoted as $\mathcal{SG}$, see Fig. 3.10(a)). We have another acyclic graph used to store the final routing result (initialized with no edges), denoted as $\mathcal{EG}$. We consider all the edges in $\mathcal{SG}$ as candidate TAM segments and gradually build $\mathcal{EG}$, using the algorithm shown in Fig. 3.11.

The input to the post-bond TAM routing algorithm is a set of cores which belong to a TAM, and the output is a core sequence indicating the routing order for the cores on this TAM and its routing cost. In lines 1-4, we construct the completed graph $\mathcal{SG}$ and assign the weight ($L_{ti} \times W_{ti}$) on them. In line 5, we sort all the edges according to their weights. Then, in every iteration (lines 6-10), we move edges from $\mathcal{SG}$ to $\mathcal{EG}$ in a greedy manner (i.e., we move the edge with the smallest weight), e.g., the solid line A-B with length 1 in Fig. 3.10(b) and the solid line B-C with length 3 in Fig. 3.10(c)). As the procedure going, more edges (i.e., TAM segments) are moved into $\mathcal{EG}$ and they are gradually linked together as a path (e.g., the linked path A-B-C in Fig. 3.10(c)). It is important to note that, there are two kinds of redundant edges that should be deleted in every iteration after a

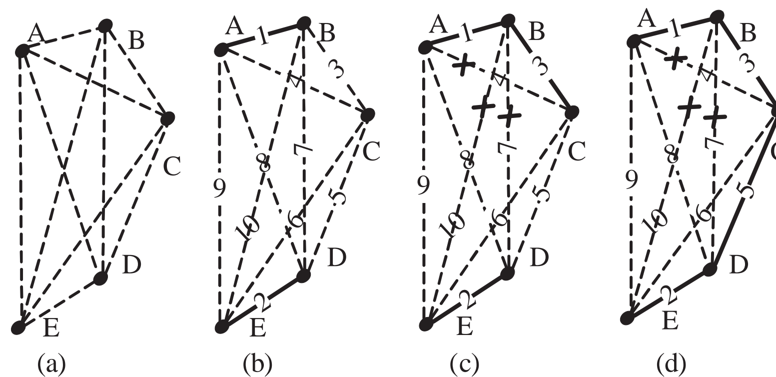


Figure 3.10: An Example of Post-Bond TAM Routing.

$C = \{core_1, ..., core_n\} \rightarrow C' = \{core'_1, ..., core'_n\}$

1 Construct a complete graph $\mathcal{SG} = (V, E)$ from all cores in C;
2 **for all** edges $e_{ij} \in E\{$
3 $\quad W(e_{ij}) := w(r) \cdot d(m_i, m_j);\}$
4 $Sum = 0$;
5 **SORT** all the edges $e_{ij} \in E \rightarrow E'$;
6 **while** $E' \neq Empty\{$
7 $\quad$ Pop first edge e_{kl} from E';
8 $\quad$ Add e_{kl} into result graph $\mathcal{EG}$;
9 $\quad$ Sum += $W(e_{kl})$;
10 $\quad$ delete redundant edges in $\mathcal{SG}$;
11 Obtain C' from $\mathcal{EG}$;
12 Obtain Sum;

Figure 3.11: Post-Bond TAM Routing Algorithm

new edge is moved into $\mathcal{EG}$ (line 10): (i) Any edge should be deleted if either of its vertex is an *internal vertex* (all vertices except two end points of a path) in current $\mathcal{EG}$; (ii) Any edge that would generate a cycle in $\mathcal{EG}$ should be deleted since $\mathcal{EG}$ should be acyclic all the time. Taking Fig. 3.10(c) as an example, edge B-E and B-D are the first kind of redundant edges, since vertex B is a internal vertex. Edge A-C is the second kind of redundant edge, since A-B-C-A will become a cyclic graph if we move A-C into $\mathcal{EG}$. Finally, all the paths are linked together to form one single path (e.g., solid lines A-B-C-D-E in Fig. 3.10(d)), which gives us the final TAM routing order and its cost, as returned from lines 11-12.

3.4.4.2 Identification of Reusable TAM Segments

Given the TAM routing for post-bond test, our problem now is to route the pre-bond TAM in such a manner that we can reuse the post-bond TAM test wires as much as possible. To reduce the problem complexity, we first divide every TAM into a set of TAM segments, each linking two adjacent cores on the same silicon

layer belonging to the TAM. After routing all the post-bond TAMs, every post-bond TAM segment is considered to be reusable by any pre-bond TAM segment that is on the same silicon layer. For the sake of simplicity, we consider each TAM segment of pre-bond test can reuse test wires from only one TAM segment of post-bond test, and each TAM segment of post-bond test can be reused by only one TAM segment of pre-bond test. It should be noted that, we have excluded those TAM segments that link two cores on different layers.

We examine several scenarios to illustrate how we can reuse post-bond TAM routing resources for pre-bond tests (see Fig. 3.12). With given core layout position (modeled as the center point of the cores), we can draw a bounding rectangle for each TAM segment as shown in Fig. 3.12(a). To connect these two cores, we can have any routes within this bounding rectangle as long as there is no detour (e.g., route A, B or C), and the Manhattan distance of these routes (i.e., the half perimeter of the bounding rectangle) are all the same.

Let us consider the 3D SoC example demonstrated in Fig. 3.8. TAM segments C1-C2 and C3-C4 are from post-bond TAM 1 and TAM 3 (see Fig. 3.12(b)-(d)). Fig. 3.12(b) depicts the case that a pre-bond TAM segment C1-C3 needs to be routed; while Fig. 3.12(c) presents another case that we need to route another pre-bond TAM segment C3-C4.

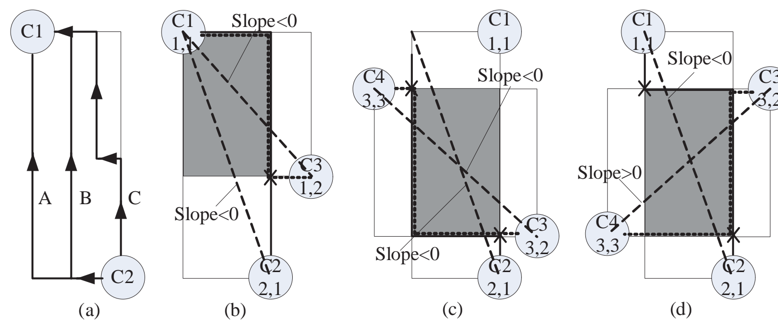


Figure 3.12: Reusable Routing Resources Represented by Bounding Rectangle.

Considering the bounding rectangles for TAM segments discussed above, it is clear that the *coincided rectangle* is where we can route the pre-bond TAM that reuses post-bond TAM routing resources (i.e., the grey parts in Fig. 3.12(b)-(d)). It is important to note that, however, the reusable wire length is not always the half perimeter of the coincided rectangle (see Fig. 3.12(d)), and it is calculated as follows.

Let us denote the slope of diagonal line with its two end points placed from up-left to bottom-right as negative (e.g., C1-C2 of Fig. 3.12(b)-(d), C1-C3 of Fig. 3.12(b)-(c) and C4-C3 of Fig. 3.12(c)). On the contrary, the slope of diagonal line with its two end points placed from up-right to bottom-left is positive (C3-C4 of Fig. 3.12(d)). Considering two TAM segments that share test wires, if the slopes of their diagonal lines are the same (i.e., all negative or all positive), as shown in Fig. 3.12(b)-(c), the reusable wire length is half perimeter of coincided rectangle. If the slopes are different (i.e., one is negative while the other is positive), however, the reusable wire length is the longer edge of the coincided rectangle, as shown in Fig. 3.12(d).

In view of the above discussion, the problem of reusing post-bond TAM routing resources in a pre-bond TAM cane be stated in terms of how to combine the one-to-one pairs of TAM segments (one from pre-bond test and the other from post-bond test) so that the total routing cost is minimized. We propose a greedy heuristic to solve this problem, as described next.

3.4.4.3 Greedy Heuristic for Pre-Bond TAM Routing

Fig. 3.13 presents our proposed greedy heuristic for pre-bond TAM routing, which tries to reuse the routing resources of post-bond TAMs as much as possible.

In line 1, we acquire the possible reusable TAM segments for post-bond test. Similar to the post-bond TAM routing algorithm shown in Fig. 3.11, we move the

```
{C_1,...,C_n} ∈ Layer_i → {C'_1,...,C'_n} ∈ Layer_i
1  Get the set of reusable post-bond TAM segment F;
2  for all TAM_i in this layer{
3       Construct a complete graph G_i = (V_i, E_i) from all
        cores in C_i belong to TAM_i;
4       put all G_i together into SG;}
5  for all edges e_ij ∈ SG{
6       W(e_ij) := w(r) · d(m_i, m_j);
7       add W(e_ij, ∅) into list WList(e_ij);
8       for all edges f_kl ∈ F{
9            calculate the routing cost after reusing
             W(e_ij, f_kl) = minWidth(e_ij, f_kl) × L_reuse(e_ij, f_kl);
10           record the routing cost W(e_ij, f_kl) and
             corresponding post-bond TAM segment f_kl
             into list WList(e_ij);}
11      SORT all the results in WList(e_ij) in ascending order;}
12 Sum = 0;
13 while E ≠ Empty{
14      find edge e_ij ∈ E with the minimum
             routing cost W(e_ij, f_x) ∈ WList(e_ij);
15      delete e_ij from E and add it into edge list: TAM_i;
16      Sum += W(e_ij);
17      for all e_kl ∈ E{
18           if exist f_x, remove W(e_kl, f_x) from WList(e_kl);}
19      delete redundant edges from SG;
20 Obtain {C'_1,...,C'_n} from {TAM_1,...,TAM_n};
21 Obtain Sum;
```

Figure 3.13: Greedy Heuristic for Pre-Bond TAM Routing

edge with the lowest routing cost from $\mathcal{SG}$ to $\mathcal{EG}$ iteratively in a greedy manner. In lines 2-4, we construct a completed graph for every TAM for pre-bond test in the layer, and put all these complete graphs together into $\mathcal{SG}$. The reason behind this is that a reusable post-bond TAM segment can be a reusable candidate for

TAM segments from more than one pre-bond TAMs. Since each TAM segment of pre-bond test has more than one reusable candidates and each reusable candidates can only be reused at most once, we build a list for each TAM segment of pre-bond test, and store all possible reusable candidates into the list (lines 8-10).

To be specific, if one edge cannot reuse any of the TAM segments of post-bond test in that layer, we simply use the original routing cost, calculated by the wire length multiplied by its TAM width, and add it into the list (lines 6-7). If it has a reusable candidate, its routing cost is updated accordingly. In addition, the TAM widths can be different between pre-bond TAM and post-bond TAM. We choose the smaller one to calculate the routing cost of reused wires by multiplying it with the reused wire length. And then we use the original routing cost minus the routing cost of reused wire as the new routing cost of this edge (line 9).

Here, we maintain the list for each edge to keep all the possible reusable TAM segments, and their corresponding updated routing costs (line 10). After sorting the list according to their routing cost (line 11), the head item of each list is the edge with the least routing cost (either with or without reuse strategy). In every iteration, we choose the edge with least routing cost (i.e. value of head item in the list linking to this edge) and move it into $\mathcal{EG}$ (lines 12-15). Since every reusable candidate can only be reused for at most once, we delete this reused segment from all other edges in $\mathcal{SG}$ (lines 17-18). Finally, we obtain the routing result and its cost (lines 19-21).

We take an example extended from Fig. 3.9 to elaborate the above process. After constructing the complete graph from all cores in pre-bond test, we obtain the list of TAM segments shown in Fig. 3.14, which keeps all possible reusable post-bond TAM segment in the list of every pre-bond TAM segment. For example, for segment [TAM1(C2,C3)], there are two reusable candidates in its list, that is, post-bond TAM segment (C1,C2) and (C3,C4), The corresponding routing costs

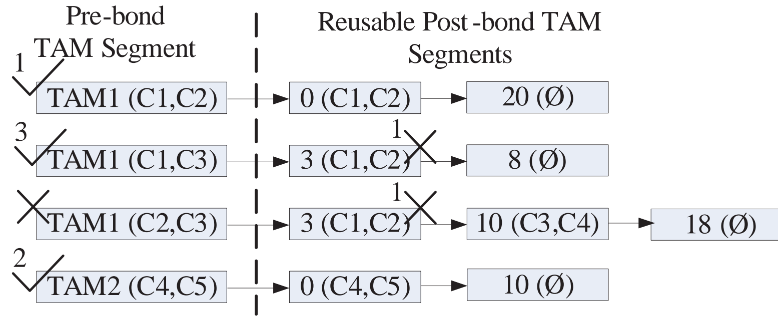


Figure 3.14: Example for the Proposed Algorithm to Reuse TAM Routing Resources.

are 3 and 10, respectively. The original routing cost is 18. In this example, we pick up the pre-bond TAM segment [TAM1(C1,C2)] first since it has the minimum routing cost 0 by reusing the post-bond TAM segment [0(C1,C2)]. We then move [TAM1(C1,C2)] from $\mathcal{SG}$ to the $\mathcal{EG}$. Afterwards, the reusable post-bond TAM segments [(C1,C2)] are deleted from all the lists. Then, in the second iteration, [TAM2(C4,C5)] is chosen, and the reusable post-bond TAM segment [(C4,C5)] is deleted. Next, we know that [TAM1(C1,C3)] has the minimum routing cost of 8 without any reuse. Again, it is moved to $\mathcal{EG}$. At last, we delete the redundant segment [TAM1(C2,C3)].

3.4.5 Layout-Driven Test Architecture Design and Optimization

With the test wire sharing scheme, we address the test architecture optimization problem progressively in this section, denoted as Scheme 1 and Scheme 2, respectively.

In Scheme 1, we consider the case that test architectures for both pre-bond tests and post-bond test are fixed, and we propose a greedy heuristic to share test wires

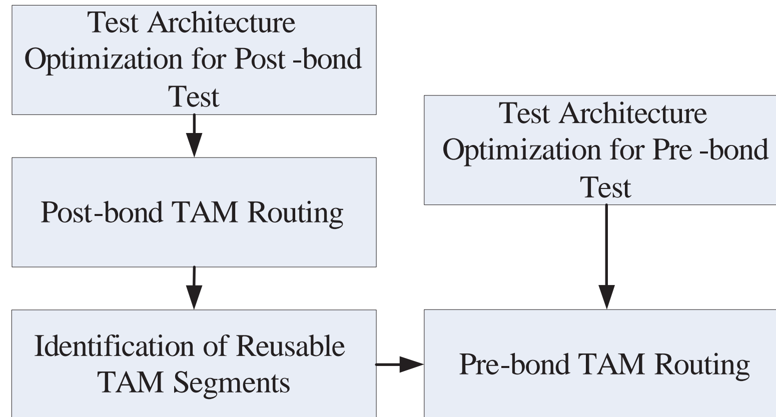


Figure 3.15: Design Flow for Scheme 1.

between pre-bond TAM and post-bond TAM as much as possible. The design flow for this scheme is shown in Fig. 3.15. Firstly, we optimize the test architecture for both pre-bond tests and post-bond test, which gives us the number of TAMs, the width for each TAM, and the cores tested on each TAM for each kind of test. Then, we take the 3D SoC layout into consideration and conduct post-bond TAM routing (as stated in Section 3.4.4.1). Next, we identify reusable TAM segments out of the post-bond TAM and conduct pre-bond TAM routing to share them as much as possible (as stated in Section 3.4.4.2 and Section 3.4.4.3 respectively).

In Scheme 2, to further reduce routing cost, we consider flexible pre-bond test architecture while keeping the post-bond test architecture and its TAM routing unchanged, and then we optimize the pre-bond test architecture so that the total cost C_{total} is minimized. The reason behind the above strategy is: (i) making both pre-bond test architecture and post-bond test architecture flexible would lead to an extremely large solution space, and hence it is rather difficult to find a good solution within limited computational time; (ii) making pre-bond test architecture (instead of the post-bond test architecture) flexible has the benefit that it only affects the routing in one layer.

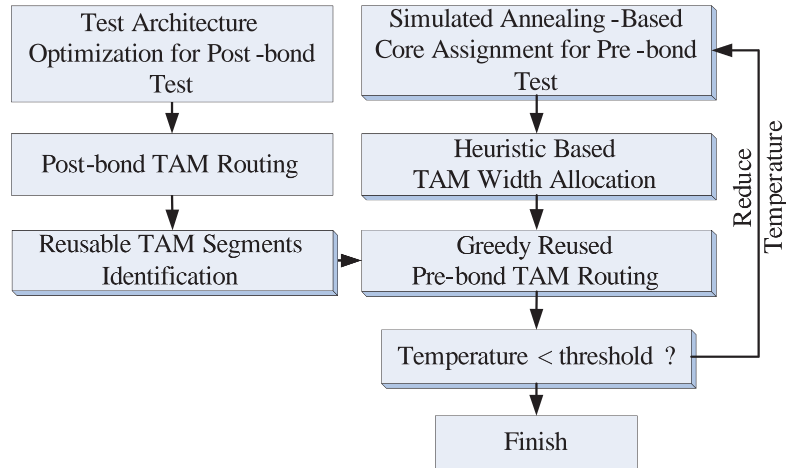


Figure 3.16: Design Flow for Scheme 2.

To further reduce their routing cost, it is possible to share more routing resources from post-bond TAM, by changing the test architecture for pre-bond tests. However, it may lead to the increase of testing time, since we change the test architecture which is previously optimized to a near optimal solution in terms of testing time cost. As a result, we would like to sacrifice only limited testing time to obtain much better routing cost. In order to achieve the above objective, we extend the simulated annealing-based 3D SoC test architecture optimization procedure presented in Section 3.3. Our design flow for Scheme 2 is shown in Fig. 3.16.

The optimization procedure is comprised of two parts: the *outer SA-based core assignment* and the *inner heuristic-based TAM width allocation*, which is almost the same with that in Similar to Section 3.3, except that we use proposed greedy reusable heuristic (as shown in Fig. 3.13) to calculate the pre-bond routing cost, and obtain the total test cost including both routing cost and testing time cost.

Width	Total Testing Time			Ratio	Routing Cost			Ratio		Total Testing Time			Ratio	Routing Cost			Ratio	
(bit)	No Reuse	Reuse	SA	Δ^T (%)	No Reuse	Reuse	SA	Δ_1^W (%)	Δ_2^W (%)	NoReuse	Reuse	SA	Δ^T (%)	No Reuse	Reuse	SA	Δ_1^W (%)	Δ_2^W (%)
	p22810									p34392								
16	948714	948714	959753	1.16	18677	15922	12648	-14.7	-32.3	2100312	2100312	2088962	-0.54	15154	13595	7760	-10.29	-48.79
24	730866	730866	738800	1.09	19549	17744	14567	-9.23	-25.5	1802341	1802341	1791078	-0.62	27624	22766	17422	-17.59	-36.93
32	647043	647043	653060	0.93	17289	15445	10898	-10.6	-36.9	1592668	1592668	1595286	0.16	25220	20946	13949	-16.95	-44.69
40	608139	608139	611087	0.48	14547	11826	8240	-18.7	-43.4	1579078	1579078	1538795	-2.55	43384	34172	24439	-21.23	-43.67
48	584380	584380	587328	0.50	25532	23402	19181	-8.34	-24.9	1572840	1572840	1567728	-0.33	33544	28460	23705	-15.16	-29.33
56	562561	562561	563882	0.23	17138	15739	10754	-8.16	-37.3	1571728	1571728	1819571	15.77	34068	29669	23423	-12.91	-31.25
64	550549	550549	553497	0.54	17642	16548	11320	-6.20	-35.8	1571728	1571728	1758060	11.86	34068	29669	23145	-12.91	-32.06
	p93791									t512505								
16	3631177	3631177	3726714	2.63	35709	32674	18803	-8.5	-47.34	23494872	23494872	23494872	0	2693	2409	1974	-10.55	-26.70
24	2782827	2782827	2791223	0.30	42034	36818	23727	-12.41	-43.55	23417347	23417347	23494872	0.33	2687	2403	1968	-10.57	-26.76
32	2347166	2347166	2390750	1.86	41403	38383	20953	-7.29	-49.39	18232745	18232745	18310270	0.43	1723	1441	1004	-16.37	-41.73
40	2099123	2099123	2153380	2.58	44550	40509	24709	-9.07	-44.54	18192297	18192297	18438359	1.35	2721	2405	2002	-11.61	-26.42
48	1932476	1932476	1946263	0.71	43546	37730	22749	-13.36	-47.76	18192297	18192297	18352952	0.88	2721	2405	2002	-11.61	-26.42
56	1814195	1814195	1842030	1.53	61754	54135	33205	-12.34	-46.23	18192297	18192297	18269822	0.43	2721	2405	2002	-11.61	-26.42
64	1708376	1708376	1737586	1.71	52686	46992	26974	-10.81	-48.80	18192297	18192297	18482093	1.59	2721	2405	2002	-11.61	-26.42

Δ_1^T: Difference ratio on total testing time between SA and reuse (testing of reuse and N-reuse is the same);
Δ_1^W / Δ_2^W: Difference ratio on wire length between Reuse and No Reuse / SA and No Reuse.

Table 3.3: Experimental Results for 3D SoC p22810 and p34392.

3.4.6 Experiments

To demonstrate the effectiveness of the proposed layout-driven 3D SoC test architecture design and optimization technique, we present experimental results for four revised ITC'02 benchmark SoCs (p22810, p34392, p93791, and t512505). The setup of benchmarks is the same with that in Section 3.3. In addition, the pre-bond TAM width is fixed to be 16 by taking the test-pin-count constraint into consideration. We compare three kinds of test architecture design and optimization solutions. The first one (denoted as *No Reuse*), implemented the algorithm in [86] to optimize testing time and it uses the TAM routing algorithm shown in Fig. 3.11 to route both post-bond TAMs and pre-bond TAMs without sharing routing resources between the two kinds of TAMs. The second one (denoted as *Reuse*), resorts to the same heuristic to optimize testing time as No Reuse, but uses the greedy heuristic algorithm shown in Fig. 3.13 to route pre-bond TAMs. The last scheme (denoted as *SA*), has the same procedures as *Reuse* to optimize post-bond testing time and post-bond TAM routing, and it uses the SA-based optimization procedure shown in Section 4.2 to adjust the pre-bond test architectures for further

test cost reduction.

We do not compare against our prior work in Section 3.3, because it does not take test-pin-count constraint into consideration and hence may result in pre-bond test architecture with more than 16-bit TAMs, preventing fair comparison.

As can be observed from Table 3.3, the testing time of *No Reuse* scheme and *Reuse* scheme (i.e., Scheme 1) are the same as they employ the same test architecture (with different routing strategies only). In most cases, *SA* scheme (i.e., Scheme 2) slightly increase the pre-bond testing time since it sacrifices some testing time to achieve reduced routing cost, but no more than 1% or 2% except for p34392 with large post-bond TAM width. In very few cases, both the testing time and the routing cost for the *SA* scheme are the smallest among the three schemes (e.g., p34392 with TAM width 40).

The routing cost reduction brought by the greedy TAM reusing algorithm used

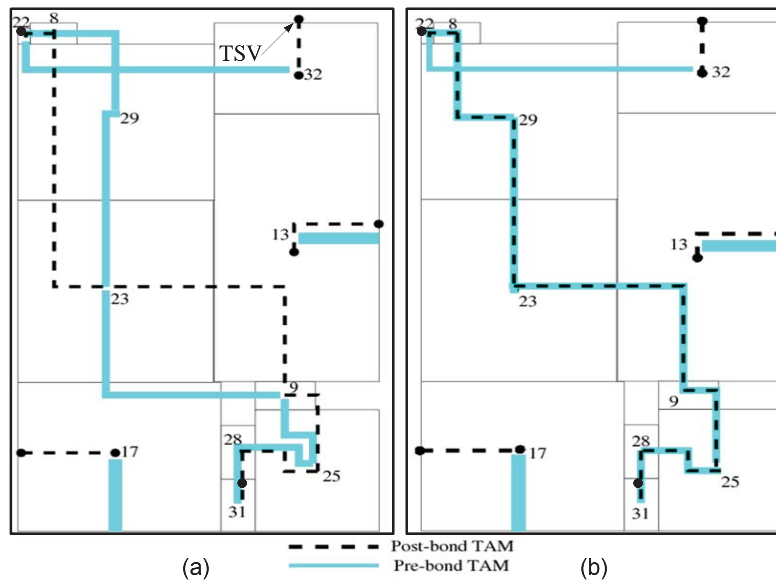


Figure 3.17: Pre-Bond TAM Routing in p93791. (a) without Reusing Post-Bond TAMs; (b) Reusing Post-Bond TAMs.

in *Reuse* scheme is considerable when comparing with *No Reuse* scheme. The reduction ratio can be as high as -21.23% for p34392 with TAM width 40. With flexible pre-bond test architecture used in the *SA* scheme, the savings in routing cost is even larger, in the range between -24.87% and -49.39%. The average routing reduction is around 33%, 38%, 46%, and 28% for the four benchmark SoCs. p93791 produces better results because there is no stand-out large core in this SoC, which can serve as a bottleneck during the optimization process. By contrast, t512505 has a large core that alone requires a large post-bond TAM width on its own, which essentially reduces the reusable TAM segments.

Generally speaking, with the growth of post-bond TAM width from 16 to 64, the routing cost reduction ratio increases in the beginning and drops at the end. The main reason is that, when post-bond TAM width grows, the width of reusable TAM segments also goes up; while the TAM width keeps to be 16 for pre-bond tests, the demanded reusable TAM width does not increase, leaving more reusable TAM width idle.

In Fig. 3.17, we present the layout of one layer in 3D SoC p93791 to demonstrate the effectiveness of our TAM reuse strategy (see). The TAM segments of post-bond TAM on this layer is shown as dashed lines; while the solid lines are the pre-bond TAMs in this layer. Note that, if a TAM only goes through one single core in this layer, it cannot be reused for pre-bond TAM (e.g., the one for cores 13). Fig. 3.17(a) depicts the test wires without reusing any post-bond TAM segments. With our reuse methodology as shown in Fig. 3.17(b), we can see that the the routing overhead for TAMs can be significantly reduced.

3.5 Conclusion

Although applying pre-bond test for 3D SoCs with D2W/D2D bonding technique needs additional test effort, it is critical for 3D SoCs yield enhancement and the final cost (the manufacture cost plus the test cost). In this chapter, we combine the pre-bond test into the 3D SoCs test architecture design, and propose efficient simulated-annealing based optimization methods to reduce test cost. Further, the pin-count constraint of pre-bond test is considered. It is a practical problem during test and our proposed solution can tackle it efficiently.

□ **End of chapter.**

Chapter 4

Stack Yield Enhancement for 3D DRAM

4.1 Introduction

To fulfill the ever-increasing demands of storage capacity, many semiconductor companies have implemented 3D-stacked memories, by using TSVs as vertical bus across multiple DRAM layers and it is believed that such memory products will be commercialized in the near future [1, 10].

Due to their extreme high density, memory circuits are prone to manufacturing defects. Consequently, to avoid yield loss, redundant rows and columns are added on-chip so that most faults can be repaired by replacing the rows/columns containing faulty bits with redundant ones. Since we can only implement limited amount of redundant resources on-chip for cost considerations and we cannot afford lengthy repair time for throughput considerations, numerous redundancy analysis and repair algorithms have been presented in the literature for effective and efficient memory repair [22].

Memory circuits typically contain multiple memory blocks and spare rows/columns are added to each memory block. If one of the blocks is not reparable, the entire memory circuit has to be discarded. Intuitively, we can increase memory yield

by letting neighboring blocks share the precious redundant rows/columns [87], but this strategy involves quite high routing overhead and hence is not used in practice for traditional two-dimensional (2D) memory circuits. With the emerging 3D-stacked memory, however, sharing redundant resources between neighboring vertical blocks for yield enhancement becomes feasible because short TSVs can be used for routing.

With the above redundancy sharing strategy, a memory block that is not self-reparable can borrow spare resources from its vertical neighbors (if any) and possibly becomes reparable after bonding. Consequently, when compared to the case that we only bond self-reparable dies together to form the 3D-stacked memory, we have the opportunity to achieve significant yield enhancement, especially when the defect density is high and/or the redundant resources are limited. At the same time, whether we could realize this opportunity highly depends on the matching strategy for the memory dies. That is, if a self-reparable memory die is bonded with a non-reparable one but they could not form a functional 3D-stacked memory circuit, the memory yield might even be sacrificed. Therefore, with the distinct defect bitmaps of different memory dies obtained with pre-bond testing, how to selectively matching them together to maximize the yield for the bonded 3D-stacked memory is an interesting and relevant problem. In this chapter, we present novel solutions to tackle the above problem. Noted that, we assume D2D bonding is utilized to form the 3D-stacked memory chips, as we are able to attach known good dies so that the manufacturing yield can be significantly higher when compared to W2W bonding. Experimental results with various memory organizations and defect distributions show that the proposed methodology is quite effective for yield enhancement.

The remainder of this chapter is organized as follows. Section 4.2 reviews related work and motivates this chapter. The 3D-stacked memory architecture that supports redundancy sharing across neighboring dies is described in Section 4.3.

Next, we discuss the memory repair strategy with redundancy sharing and our memory die matching algorithms in Section 4.4 and Section 4.5, respectively. Section 4.6 presents experimental results with various memory organizations and defect distributions. Finally, Section 4.7 concludes this chapter.

4.2 Preliminaries and Motivation

4.2.1 Prior work

The memory repair problem can be formulated as a constrained vertex cover sets of bipartite graphs problem and has been proved to be NP-complete in [88]. It is possible to obtain optimal repair solution by exhaustive search, but this is too time-consuming to be used in practice. To address this problem, various redundancy analysis and fast memory repair techniques (e.g., [89, 88]) were presented in the literature to reduce repair time at the cost of slight memory yield loss.

In the above memory repair strategies, they assume a full fault bitmap of the memory is available and the repair is conducted by external memory testers. With the increasing usage of embedded memories, built-in self-repair (BISR) has become more popular. As it is not cost-effective to store the entire fault bitmap before repair, various memory repair techniques with limited fault bitmap are developed in recent years. In particular, a so-called *essential spare pivoting algorithm* (*ESP*) is presented in [90]. In this work, the authors classify faults into three types: 1) suitable for row repair; 2) suitable for column repair; 3) orthogonal fault which can either be repaired by a spare row or a spare column. This work is shown to be quite effective and efficient.

A memory circuit typically contains multiple memory blocks. If one of these memory blocks is irreparable, the entire memory circuit is deemed as defective and has to be discarded. Clearly, if a self-irreparable memory block can borrow some redundant resources from other reparable blocks, memory yield can be increased.

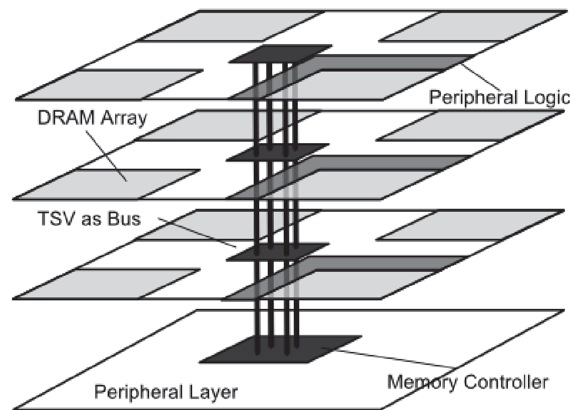


Figure 4.1: 3D-Stacked DRAM

Motivated by the above, [87] proposed a distributed global replaceable redundancy scheme, which allows to use the spare rows/columns in a memory block to repair faults in other memory blocks. In [91], the author proposed a memory built-in self-repair (BISR) algorithm with sharable redundant resources, using a programable decoder. However, such decoder design is much more complex than conventional one, which not only increases area overhead but also significantly deteriorates the routability of memory. Because of this, redundancy sharing is not utilized in practice for today's 2D memory devices.

4.2.2 Motivation

In 3D-stacked memory, bit-arrays are stacked vertically on each other and TSVs are utilized as buses to link the stacked dies together, as shown in Fig. 4.1. Such organization provides us the opportunity to conduct redundancy sharing across neighboring dies with short TSVs without incurring much routing overhead. With redundancy sharing, an irreparable memory die is likely to become reparable by borrowing spares from its neighboring dies, but whether we can realize the above yield enhancement opportunity highly depends on the matching strategy.

Consider the eight memory dies shown in Fig. 4.2, each of them containing

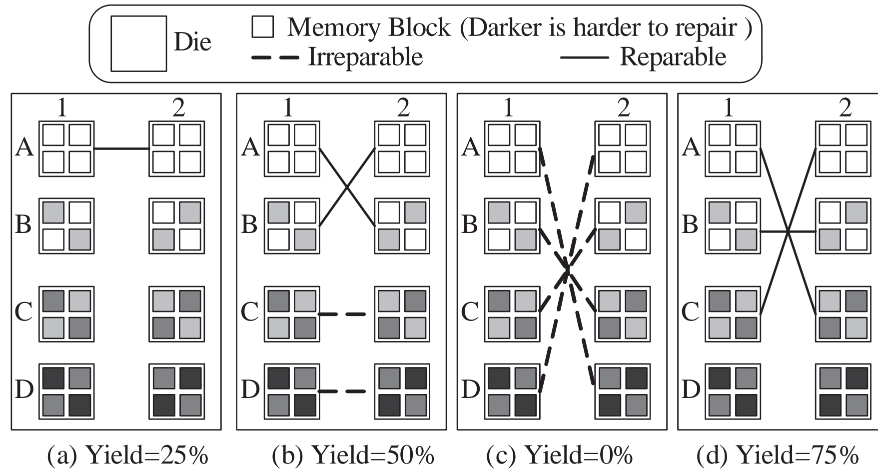


Figure 4.2: Different Matching Strategy affects the Overall Yield.

four memory blocks with different fault density. For simplicity, we classify these dies into four categories.

The memory blocks in Type A are all self-repairable and provide extra spare resources (see A1 and A2 in Fig. 4.2(a)). Type B memory dies cannot repair themselves but become reparable by borrowing a small number of redundant resources from other dies, and we assume that two Type B memory dies stacking together can be reparable. Type C memory dies must borrow plenty of spares to become reparable, while the memory blocks in type D have high fault density and is irreparable even if they are allowed to borrow spares from others.

For this example, suppose we only allow bond self-reparable memory dies only (see Fig. 4.2(a)), the yield is only 25%. As shown in Fig. 4.2(b), stacking A1 to B2 and B1 to A2 produces two reparable 3D-stacked memory circuits, and the overall yield in this case is 50%. Suppose we have an aggressive repair strategy as shown in Fig. 4.2(c), which attempts to repair Type D memory dies using the spare resources in Type A dies, the yield can be 0% in the extreme case. In other words, a bad matching strategy can even reduce the overall memory yield since good dies might be wasted. A good matching strategy as shown in Fig. 4.2(d), on the other

hand, results in a maximum yield of 75%.

From the above, we can conclude that die matching strategy is critical for the final yield of the 3D-stacked memory circuits. Suppose we can quickly know whether the matched 3D-stacked memory is reparable or not for any pair of memory dies, the matching problem is quite similar to the wafer-to-wafer bonding selection problem in [44]. However, with both redundant rows and columns for a memory block, we cannot afford the time used to temporarily running repair algorithm between every possible memory die pairs. Consequently, how to conduct efficient and effective matching to achieve the maximum memory yield is a challenging problem, which motivates this work.

4.3 3D-Stacked Memory Architecture with Redundancy Sharing

Fig. 4.3 depicts the 3D-stacked memory architecture that supports redundancy sharing across dies. Spare rows/columns on each memory die not only connects to the programmable decoder on its own layer, but also link to the decoder of other layers using TSVs. The routing overhead to support redundancy sharing across dies is quite low due to the use of short TSVs as routing mechanisms. The prefabricated multiplexor controls which memory block use the corresponding spare row/column. For a memory block with *n* spare rows and *m* spare columns, sharing

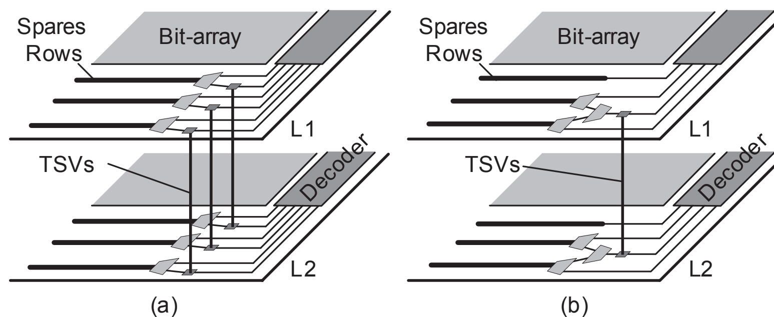


Figure 4.3: Redundancy Sharing using TSVs.

all the redundant resources between two neighboring dies requires $n+m$ TSVs, as shown in Fig. 4.3(a) (only spare rows are shown in this figure). With the continuing improvement of TSV manufacturing technology, this overhead is of a less concern. Moreover, we can leave some spare resources for repairing its own die and/or share the same TSVs for different redundant rows/columns, reducing the amount of TSVs used for redundancy sharing. An example is shown in Fig. 4.3(b).

The above architecture can tolerate some TSV defects. Let us consider memory blocks in Fig. 4.3 (a). If one of the three TSVs is defective, we can leave the corresponding spare row to repair its own memory block, while we still have two spare rows for sharing among neighboring blocks. As long as a memory block does not require to borrow all three redundant rows to become reparable, it is still sufficient.

4.4 Memory Repair with Redundancy Sharing

When conducting self-repair of a memory block, there might be multiple repair strategies to replace the faulty rows/columns with redundant ones, and any one of them suffices. With resource sharing between neighboring dies, however, how to repair each individual die determines whether the other die can be repaired or not. For example, two memory blocks are shown in Fig. 4.4(a)-(b), in which black vertices denote faulty cells and solid/dashed lines denote spare columns/rows, respectively. Memory block in Fig. 4.4 (a) is self-irreparable. Memory block in Fig. 4.4 (b) is self-reparable, but it can lend no spares to the other block if we guarantee to repair itself first. Bonding these two dies together is reparable only if memory block in Fig. 4.4(a) repairs itself with three spare columns (one is borrowed) while memory block in Fig. 4.4(b) repairs itself with two spare rows (one is borrowed) and one spare column. From this example, the memory repair algorithm with redundancy sharing must be aware of the fault information of stacked

dies and be conducted at a global level.

4.4.1 Problem Formulation

The problem of memory repair with redundancy sharing can be formulated as follows:

Problem: Given

- The number of memory blocks N_b on the memory die;
- The fault bitmap of each memory block obtained from pre-bond testing;
- The number of spare rows/columns R_a, C_a in each memory block;

Our objective is to determine the repair scheme of each memory block so that the stacked memory dies can be reparable, whenever possible.

4.4.2 Memory Repair Strategy

A single memory repair problem is formulated as constrained vertex cover sets of bipartite graphs. Based on the fault bitmap of the memory block, we can build corresponding bipartite graphs (see Fig. 4.4). The left set of vertices denote the row number and the right set denote the column number of the faulty bits, respectively. An edge between two vertices represents the position of the corresponding faulty bit. With given number of spare rows and columns as constraints, our objective is to find a set of vertices to cover one end of every edge.

We propose a novel algorithm to tackle this problem, inspired from the concept of so-called *irrespective error matrixes* in [91]. We first extract rows and columns that contain faulty bits from the two memory blocks with sharable redundancies, and integrate them together into a new fault bitmap by putting them in diagonal position without intersection, as shown in Fig. 4.4(c). For the new fault bitmap, our redundant resources are also doubled. By translating the problem of repairing two dies with redundancy sharing into a new memory problem as above, we guarantee

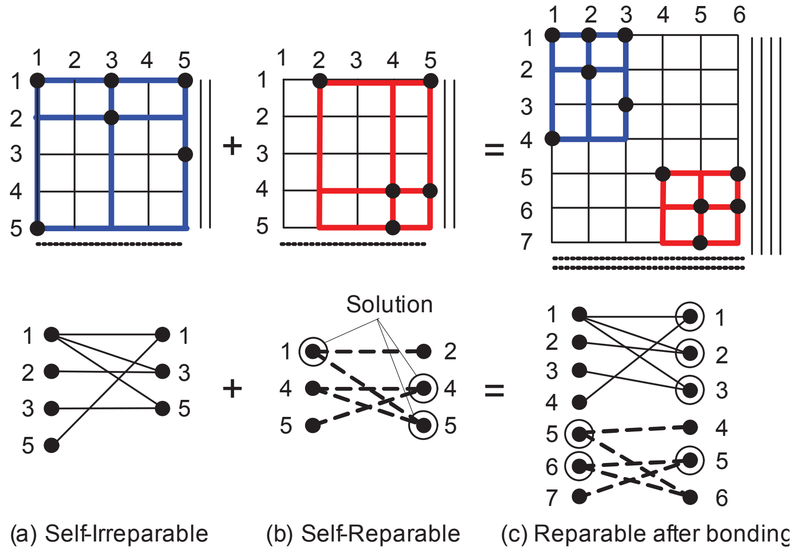


Figure 4.4: Irrespective Sub-Bipartite Graphs.

that a sharable redundant row/column can be flexibly utilized to replace a faulty one in either of the two memory blocks without enumerating all possible repair solutions for each die. Then, we can apply any redundancy analysis and repair algorithm to solve this problem. In this example (see Fig. 4.4(c)), a solution is found so that a self-irreparable block and a self-reparable one forms a reparable stacked memory circuit.

4.5 Matching for Yield Enhancement

4.5.1 Problem Formulation

As discussed earlier, how to conduct efficient and effective die matching has a significant impact on the final yield of the 3D-stacked memory circuits. We model this problem by constructing a graph with these memory dies as vertices and add edges between two vertices if the corresponding memory dies are reparable after stacking. This problem can then be solved in polynomial time as the classical *maximum matching* problem.

The challenge here lies in the fact that it is extremely time-consuming to run repair algorithm between every die pairs and hence it is impractical to obtain the exact edge information as modeled above. We hence need more efficient method to address the die matching problem, as formulated in the following.

Problem: Given a number of memory dies with the following information:

- The number of memory blocks in each die;
- The fault bitmap of each memory block in every die obtained from pre-bond testing;
- The number of spare rows/columns of each memory block;

Our objective is to match the given dies in a *pairwise* manner to maximize the total number of reparable 3D-stacked memory circuits. It is important to note that even though a 3D-stacked memory can contain more than two layers, we consider pairwise matching only in this work. There are two reasons behind: (i). selectively matching more than two dies is a much more complicated problem since the possible combinations grow rapidly; (ii). generally speaking, the die yield cannot be a small value (otherwise the product is not profitable) and a good pairwise matching algorithm is able to recover most self-irreparable dies. By simply combining the matched memory pairs to form 3D-stacked memory circuits with more layers, we can maintain a high memory yield.

4.5.2 Direct Matching

As discussed earlier, we cannot afford the computational complexity to run repair algorithm for every possible pair of memory dies to obtain *exactly* whether two dies matched together can form a reparable stacked circuit (referred to as *matching condition*). If, however, we can *estimate* the *matching conditions* with sufficient accuracy *efficiently*, we should be able to achieve good matching results. Motivated by the above, we develop two kinds of matching conditions to solve our

die matching problem. Before describing them in detail, we first present our die matching algorithm as follows.

4.5.2.1 Die Matching Algorithm

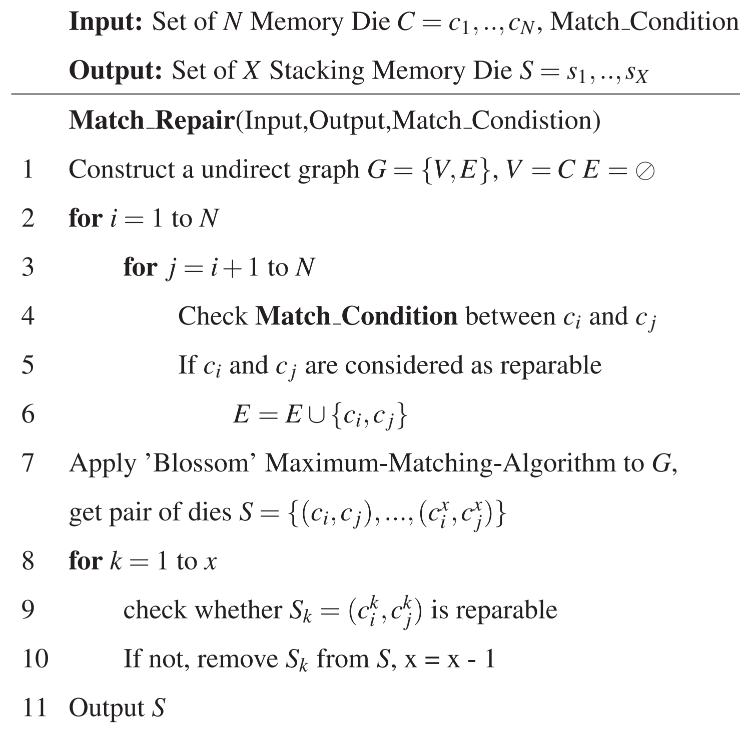
Fig. 4.5 presents the pseudocode of our algorithm. We first construct a graph with each memory die as a vertex (Line 1), and check every pair of memory dies whether they are considered to be reparable according to our matching condition (Line 4). After checking all pairs and adding the corresponding edges, a undirect graph is constructed (Lines 1-6). We then use the classic *'Blossom' maximum matching algorithm* [**?**] to get the matched die pairs (Line 7). Finally, to verify whether every pair is reparable and get the final yield (Lines 8-10), we use a two-step memory repair algorithm: (i) We first utilize the efficient ESP algorithm [90] for repair; (ii) If a solution cannot be found, we resort to a branch-and-bound method for another try and abort to repair if a solution still cannot be found under a runtime limit.

4.5.2.2 Reparability Condition

Let us first consider *reparability condition* which guarantees that any pair of memory dies passing through this condition must be reparable. Apparently, matching self-reparable dies together is one type of reparability condition, as shown in Fig.4.6(a). However, such matching strategy is too conservative, far from the optimal matching solution[1]. We therefore propose to extend the *ESP* algorithm presented in [90] for building more effective reparability condition efficiently.

Similar to [90], for a memory block i with R_i spare rows and C_i spare columns, we classify the faulty bits into three types: (i). F_i^r bits that are suitable for row

[1]The "optimal matched dies" shown in Fig.4.6 refers to the obtained dies in the case that we can run final repair algorithm between every possible memory die pair to determine whether they are reparable.



Input: Set of N Memory Die $C = c_1, .., c_N$, Match_Condition

Output: Set of X Stacking Memory Die $S = s_1, .., s_X$

Match_Repair(Input,Output,Match_Condistion)

1 Construct a undirect graph $G = \{V, E\}$, $V = C$ $E = \oslash$
2 **for** $i = 1$ to N
3   **for** $j = i + 1$ to N
4   Check **Match_Condition** between c_i and c_j
5   If c_i and c_j are considered as reparable
6             $E = E \cup \{c_i, c_j\}$
7 Apply 'Blossom' Maximum-Matching-Algorithm to G,
get pair of dies $S = \{(c_i, c_j), ..., (c_i^x, c_j^x)\}$
8 **for** $k = 1$ to x
9   check whether $S_k = (c_i^k, c_j^k)$ is reparable
10   If not, remove S_k from S, x = x - 1
11 Output S

Figure 4.5: Die Matching Algorithm

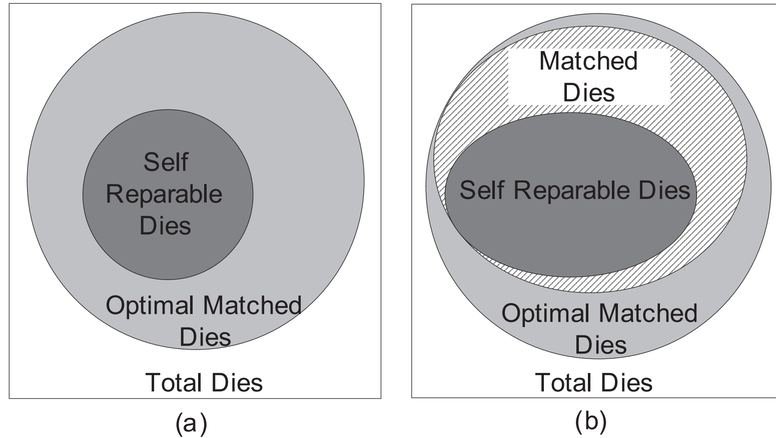


Figure 4.6: Matching according to Reparability Condition.

repair; (ii). F_i^c bits that are suitable for column repair; and (iii). F_i^o orthogonal bits that can be repaired by either spare rows or spare columns. Then, we use the following formula to determine whether two blocks a and b are reparable after

stacking:

$$R_l = (R_a + R_b - F_i^r - F_j^r) \geq 0 \quad (4.1)$$

$$C_l = (C_i + C_j - F_i^c - F_j^c) \geq 0 \quad (4.2)$$

$$R_l + C_l \geq F_i^o + F_j^o \quad (4.3)$$

R_l/C_l are the number of left spare rows/columns after repairing the corresponding faulty cells that require dedicated row/column repair. If their sum can be used to repair the remaining orthogonal faulty bits, we can guarantee that bonding the two memory blocks together is reparable. By checking all pairs of memory blocks between the two memory dies, the above condition is used as our new reparability condition, which can provide yield enhancement when compared to matching self-reparable dies only, as shown in Fig.4.6(b).

4.5.2.3 Irreparability Condition

The above reparability condition guarantees that all found die pairs are reparable. Such stringent requirement prevents us from finding those reparable pairs that cannot pass the previous reparability condition checking and inherently limits the achievable maximum yield. In this subsection, we consider another type of matching conditions based on *irreparability checking*. By only eliminating those pairs that are guaranteed to be irreparable, more possible die pairs can be found but we cannot guarantee they are reparable. We obtain the irreparability condition by analyzing the property of bipartite graph constructed from the fault bitmap.

Given a bipartite graph $G = (V, E)$, where vertices are partitioned into two sets (X and Y, $x_i \subseteq X$, $y_i \subseteq Y$) and each edge $(x_i, y_i,) \subseteq E$, we have the theorem that the minimum number of vertices that cover all the edges is equal to the number of edges in any *maximum bipartite matching* of the graph [88]. For a memory block i with R_i spare rows and C_i spare columns, it is easy to deduct from the above that,

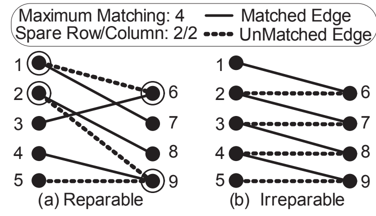


Figure 4.7: Maximum Matching and Reparability.

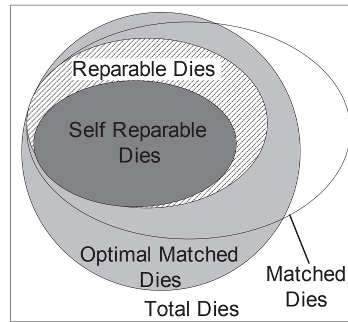


Figure 4.8: Irreparability Condition.

the memory block is not reparable if $R_i + C_i$ is smaller than the number of vertices required for maximum bipartite matching (can be obtained efficiently as in [88]). Now let us consider the matching condition for two memory blocks a and b, we have the following lemma.

Lema 1: Given two memory blocks with spare rows R_a, R_b and spare columns C_a, C_b. The mapped bipartite graph from these two blocks is $G_a = (V_a, E_a), G_b = (V_b, E_b)$. The maximum bipartite matching of G_a, G_b are M_a, M_b. Then, only if

$$|M_a| + |M_a| \leq R_a + R_b + C_a + C_b \tag{4.4}$$

the stacked memory is possibly to become reparable.

The die matching algorithm is then conducted on the bipartite graph constructed according to the above irreparability condition. With this strategy, we are able to find more die pairs that are possibly reparable, but this is not guaranteed. For the two examples shown in Fig. 4.7, both require at least four spares

(either spare row or spare column) to be possibly reparable. Even though both are deemed as possibly reparable according to the above lemma, if in total we have 2 spare rows and 2 columns with redundancy sharing, the faults in (a) are reparable (rows 1,2 and columns 6,9), but the faults in (b) cannot be repaired (a possible repair solution is when we have 4 spare columns). Fig. 4.8 presents the matching effect according to irreparability condition. As can be observed, within the matched dies, only part of them are reparable.

4.5.3 Iterative Matching

Directly matching memory dies according to the previous reparability condition and irreparability condition all have their limitations. In this subsection, we consider to match dies iteratively. That is, we conduct die matching multiple times, and in each iteration, we keep those good pairs while redo the matching for the left dies by changing the matching condition.

We start from the irreparability condition [2] as discussed earlier, since such strategy gives us more possible die pairs and our die matching algorithm will try to match them by aggressively making use of the redundant resources. Then, we keep those pairs that are reparable, and for those found pairs that are not reparable, we conduct matching again with tightening irreparability condition as follows:

$$|M_i| + |M_j| + k \leq R_i + R_j + C_i + C_j \qquad (4.5)$$

, with gradual increase of the value k (initialized as 0).

As shown in Fig. 4.9, every time we tighten the irreparability condition, more reparable pairs are found because the existence of more spare rows/columns between them. At the same time, the number of found pairs decreases, and the above procedure terminates until we cannot find any pairs that can be matched.

[2]Starting from reparability condition does not give us any benefit because all the matched pairs are reparable already.

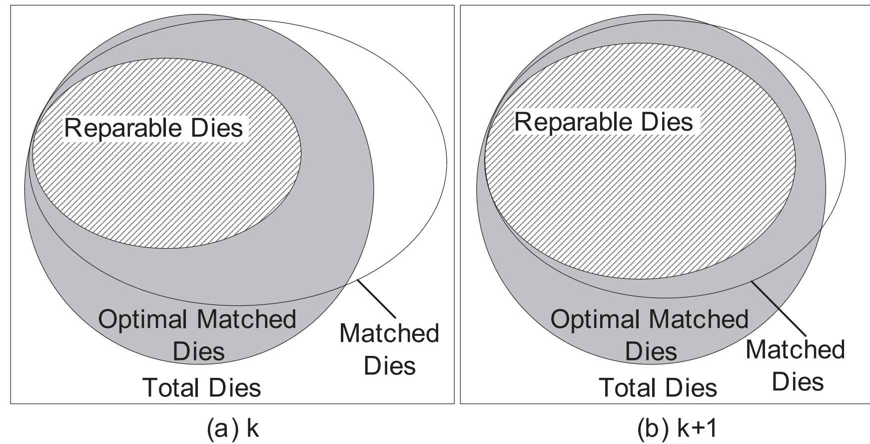


Figure 4.9: Iterative Matching with Changing Conditions.

4.6 Experimental Results

4.6.1 Experiment Setup

We consider a total of 1000 1*Gb* memory dies to be formed as 2-layer memory circuits, and hence we can obtain a maximum of 500 functional stacked memories when the yield is 100%. Each memory die contains 4×4 memory blocks, and each memory block is with the size of $8k \times 8k$ ($Row \times Column$) bit-cells.

For fault injection, we consider two cases to obtain the number of faults in each die: (i). Poisson distribution with $\lambda = 2.130$ [92]; (ii). Polya-Eggenberger distribution with $\lambda = 2.130$ [90]. For Polya-Eggenberger distribution, we also tune another parameter α with (ii) $\alpha = 2.382$ [93] and (iii) $\alpha = 0.6232$ [94], representing the case with clustered faults and that with evenly distributed faults, respectively.

We assume that all the spare rows/columns can be borrowed between neighboring vertical memory blocks, and we inject random TSV faults with faulty rate as 0.1%. Experiments are conducted on two cases with different percentage of six kinds of faults (see the following table).

Fault	Single Cell	Double Cell	Single Row	Single Column	Double Row	Double Column
Case 1	40%	4%	20%	20%	8%	8%
Case 2	70%	4%	8%	8%	5%	5%

Table 4.1: Experimental Parameters for Two Cases.

4.6.2 Results and Discussion

In our experiments, we compare four matching strategies: (i). matching self-reparable dies only; (ii). matching according to reparability condition; (iii). matching according to irreparability condition; and (iv). iterative matching. Tables 4.2 present our experimental results with gradual increase of spare rows/columns.

From this table, we can observe that, with redundancy sharing, all the three proposed matching strategies significantly increase memory yield when compared to the case that we bond self-reparable dies only (up to 35.6% yield enhancement). Generally speaking, the amount of yield improvement decreases with the increase of redundancy, while the number of self-reparable dies grows rapidly. From another perspective, we can observe that for all the cases, iterative matching with 10×10 spares outperforms bonding self-reparable dies only with 18×18 spares. In other words, to achieve the same memory yield, much fewer spare rows/columns are needed with redundancy sharing, which justifies the use of some TSVs to facilitate this repair strategy.

In most cases, iterative matching results in the highest yield, followed by matching according to reparability condition while matching according to irreparability condition is inferior to the other two strategies. This is because, even though the number of matched dies according to irreparability condition is usually quite high (see Column N_{match}), a non-trivial portion of them might be irreparable, especially when the redundant resources are not sufficient. There are also few cases that iterative matching results in less reparable memory circuits when compared to matching according to reparability condition (e.g., Poisson Distribution in Ta-

ble 2(b) with 10×10 spares). This is because, when the first iteration matching according to irreparability condition has returned very high yield (494 out of 500 die pairs are reparable in this case), there are few flexibility for us to match more reparable pairs for the left ones.

From Table 2, we can also see that the yield in Case 2 is always higher than that in Case 1. This is because, there are much more single cell faults in Case 2 and they can be easily repaired with either a spare row or a spare column, and hence the repair algorithm has higher flexibility to fulfill such needs. For polya-eggenberger fault distribution, the yield values with $\alpha = 0.6232$, is much larger than that with $\alpha = 2.38$. This is also expected because, there are more fault clusters when α is large, and hence such memory dies are more difficult to repair.

4.7 Conclusion

In this chapter, we propose to conduct redundancy sharing across neighboring dies for yield enhancement of the emerging 3D-stacked memory circuits. To achieve this objective, we first develop a repair strategy that enables redundancy sharing for any given pair. Next, we present novel solutions that selectively match memory dies together for yield maximization. Experimental results show that the proposed technique is able to greatly enhance memory yield or requires much less redundant resources to achieve similar yield.

☐ **End of chapter.**

Table 4.2: Experimental Results

(a) Case1

Spare	Self Repair		Reparability Condition		Irreparability Condition			Iterative Approach	
$SR \times SC$	N_{repair}	Y_{self}	N_{repair}	ΔY_r	N_{match}	N_{repair}	ΔY_{ir}	N_{repair}	ΔY_{it}
Poisson Distribution									
6×6	203	40.6%	341	27.6%	386	281	15.6%	352	29.8%
8×8	268	53.6%	415	29.4%	475	367	19.8%	446	**35.6%**
10×10	358	71.6%	481	24.6%	496	411	10.6%	**482**	24.8%
12×12	388	77.6%	485	19.4%	498	448	12.0%	489	20.2%
14×14	428	85.6%	496	13.6%	500	477	9.8%	496	13.6%
16×16	457	91.4%	498	8.2%	500	490	6.6%	498	8.2%
18×18	**471**	94.2%	499	5.6%	500	498	5.4%	499	5.6%
Average				18.34%			11.4%		19.7%
Polya-Eggenberger Distribution $\alpha = 0.6232$									
6×6	307	61.4%	412	21.0%	445	404	19.4%	428	**24.2%**
8×8	367	73.4%	436	13.8%	483	407	8.0%	460	18.6%
10×10	410	82.0%	457	9.4%	494	435	5.0%	**480**	14.0%
12×12	434	86.8%	477	8.6%	497	454	4.0%	485	10.2%
14×14	453	90.6%	490	7.4%	499	469	3.2%	490	7.4%
16×16	463	92.6%	495	6.4%	500	484	4.2%	496	6.6%
18×18	**471**	94.2%	497	5.2%	500	489	3.6%	497	5.2%
Average				10.26%			6.77%		12.31%
Polya-Eggenberger Distribution $\alpha = 2.38$									
6×6	111	22.2%	202	18.2%	215	169	11.6%	205	18.8%
8×8	152	30.4%	278	25.2%	318	181	5.8%	286	26.8%
10×10	187	37.4%	340	30.6%	396	179	-1.6%	**350**	**32.6%**
12×12	219	43.8%	354	27.0%	414	254	7.0%	377	31.6%
14×14	257	51.4%	393	27.2%	447	300	8.6%	406	29.8%
16×16	292	58.4%	421	25.8%	457	353	12.2%	430	27.6%
18×18	**319**	63.8%	443	24.8%	477	374	11.0%	450	26.2%
Average				25.54%			7.8%		27.62 %

ΔY_r: Yield improvement according to reparability condition over Y_{self}.
ΔY_{it}: Yield improvement with iterative matching over Y_{self}.

(b) Case 2

Spare	Self Repair		Reparability Condition		Irreparability Condition			Iterative Approach	
$SR \times SC$	N_{repair}	Y_{self}	N_{repair}	ΔY_r	N_{match}	N_{repair}	ΔY_{ir}	N_{repair}	ΔY_{it}
Poisson Distribution									
6×6	291	58.2%	446	31.0%	455	445	30.8%	452	**32.2%**
8×8	376	75.2%	493	23.4%	495	485	21.8%	493	23.4%
10×10	443	88.6%	500	11.4%	500	494	10.2%	**498**	11.0%
12×12	462	92.4%	500	7.6%	500	499	7.4%	499	7.4%
14×14	483	96.6%	500	3.4%	500	500	3.4%	500	3.4%
16×16	494	98.8%	500	1.2%	500	500	1.2%	500	1.2%
18×18	**498**	99.6%	500	0.4%	500	500	0.4%	500	0.4%
Average				11.2%			10.74%		11.29%
Polya-Eggenberger Distribution $\alpha = 0.6232$									
6×6	382	76.4%	468	17.2%	477	469	17.4%	474	**18.4%**
8×8	436	87.2%	486	10.0%	495	479	8.6%	490	11.0%
10×10	468	93.6%	492	4.8%	499	488	4.0%	**495**	5.0%
12×12	476	95.2%	496	4.0%	500	495	3.8%	498	5.0%
14×14	484	96.8%	498	2.8%	500	497	2.6%	498	2.8%
16×16	491	98.2%	499	1.6%	500	498	1.4%	499	1.8%
18×18	**494**	98.8%	499	1.0%	500	499	1.0%	499	1.2%
Average				5.91%			5.54%		6.2%
Polya-Eggenberger Distribution $\alpha = 2.38$									
6×6	177	35.4%	294	23.4%	305	294	23.4%	302	25.0%
8×8	228	45.6%	371	28.6%	389	354	25.2%	378	**30.0%**
10×10	298	59.6%	416	23.6%	439	394	19.2%	**429**	26.2%
12×12	313	62.6%	438	25.0%	452	423	22.0%	444	26.2%
14×14	354	70.8%	456	20.4%	472	441	17.4%	461	21.4%
16×16	382	76.4%	472	18.0%	481	468	17.2%	476	18.8%
18×18	**408**	81.6%	479	14.2%	489	469	12.2%	482	14.8%
Average				21.89%			19.51%		23.2%

ΔY_{ir}: Yield improvement according to irreparability condition over Y_{self}.

Chapter 5

TSV Repair for Assembly Yield Enhancement

5.1 Introduction

The assembly process for 3D-SICs involves many challenging manufacturing steps (e.g., wafer thinning and TSV bonding), which may cause various types of TSV faults [40]. Adding redundant TSVs to repair faulty ones is probably the most effective method to enhance assembly yield besides improving the manufacturing process itself. Several TSV redundancy design techniques have been proposed in the literature [3, 2, 1]. Despite different redundancy allocation strategies used in these works, they all assume uniformly-distributed TSV faults and use neighboring TSVs to replace faulty ones, if any. In practice, however, the bonding quality of TSVs depends not only on the bonding technology, but also the winding level of the thinned wafer and the surface roughness and cleaness of silicon dies. Consequently, if one TSV is defective during the bonding process, it is more likely that its neighboring TSVs are also faulty. Due to such clustering effect, earlier TSV repair techniques are less effective because a signal TSV and its neighboring redundant TSV may be defective at the same time.

In this chapter, we propose a novel TSV repair framework to tackle the above problem. Instead of repairing faulty TSVs by their neighbor TSVs, our technique

enables them to be repaired by redundant TSVs that are “distant”. With the improved repair flexibility, our technique is suitable to repair clustered, faulty TSVs. To guarantee the timing correctness after repair, we also present a new repair algorithm in this chapter. Experimental results show that the proposed solution outperforms prior techniques, especially when the number of TSVs used in the 3D-SICs is large and/or the clustering effect is significant.

The remainder of this chapter is as follows: Section 5.2 presents the preliminaries and motivation of this work. In Section 5.3, we present the hardware architecture of our TSV repair framework. The corresponding repair algorithm is then shown in Section 6.5. Section 5.5 presents the corresponding redundancy architecture construction. Section 5.6 presents the experimental results for various hypothetical 3D-SICs. We then discuss several practical considerations to use the proposed technique in Section 5.7. Finally, we conclude this work in Section 5.8.

5.2 Preliminaries and Motivation

To date, there is no public data on actual TSV failure rates. In fact, they can vary significantly among different foundries because the failure rate of a particular TSV technology depends on its technology maturity level and parameters such as TSV width/height and TSV pitch size. The common belief is that: while the TSV processing technology has advanced significantly over the past several years, TSV yield is still not satisfactory, requiring redundancy for defect-tolerance. Consequently, several TSV redundancy allocation strategies were presented in the literature which differ in terms of redundancy ratio ($\frac{\#Redundant\ TSVs}{\#Signal\ TSVs}$), repair capability, and hardware cost.

In [1], Samsung presented a TSV redundancy strategy used to improve the yield of its 3D memory product, wherein four signal TSVs and two spares are bundled together to form a group of six TSVs (see Fig. 5.1(a)). The redundancy

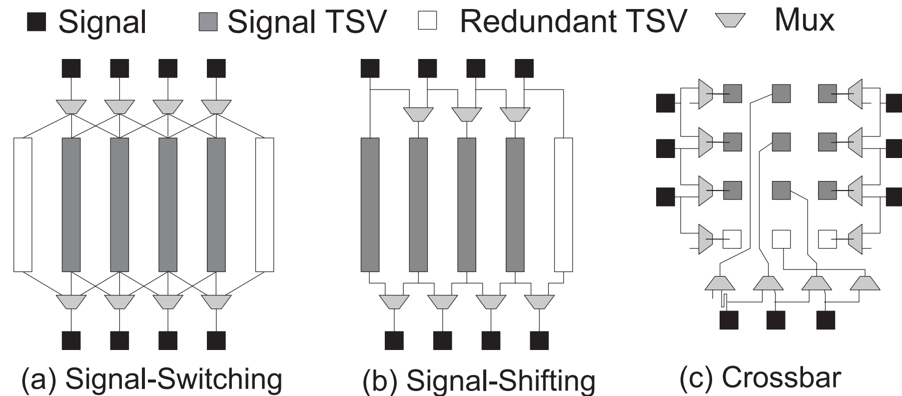


Figure 5.1: Existing TSV Redundancy Solutions: (a) Signal Switching [1]; (b) Signal Shifting [2]; (c) Crossbar [3].

ratio of this technique is 1:2 and it can tolerate any two TSV failures within the group.

Hsieh *et al.* [2] proposed to link signal TSVs in a TSV block with one spare TSV to form a TSV-chain (see Fig. 5.1(b)). If there is one defective TSV in a TSV block, signal shifting is conducted to repair it with the spare. Suppose each TSV block contains N TSVs, the redundancy ratio of this technique is $1 : N$, and it can tolerate one TSV failure in the block.

In *et al.* [3], for a $N \times N$ TSV grid used as NoC links, redundant rows or columns of TSVs are added for defect-tolerance. Suppose a redundant row is added (see Fig. 5.1(c)), each spare TSV is connected to the signal TSVs on its corresponding column through a crossbar and it can be used to repair any defective TSV on that column. Suppose M redundant columns/rows are added, the redundancy ratio of this technique is $M : N$, and it can tolerate any M TSV failure in each row/column in the grid.

While significant yield improvements were achieved in the above works, their analysis, in all cases, was based on the assumption that TSV defects are uniformly-distributed. This assumption may hold true for certain random defects such as void

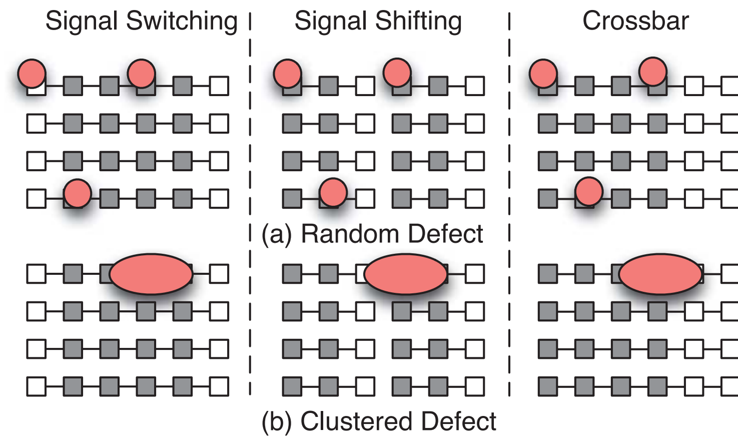


Figure 5.2: Results of Existing Repair Schemes Assuming 1/2 Redundancy Ratio.

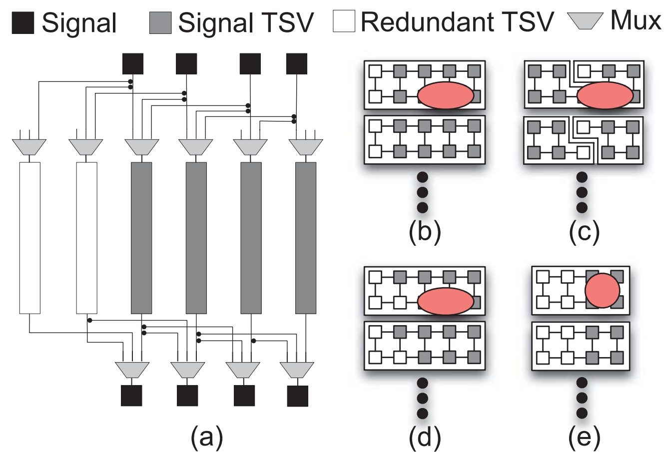


Figure 5.3: The Existing Clustered Faults Aware Repair Scheme [4].

formation [95] and lamination due to thermal induced stress [96]. At the same time, however, many types of TSV defects appear during the imperfect bonding process. Oxidation or contamination of the bond surface, height variation of the TSVs, thinned dies warping [27] and bowing of a wafer can cause large alignment errors [97], leading to clustered, faulty TSVs.

Due to the above, it is likely that, while most signal TSV groups have very few faults and are reparable, there exists one or more signal TSV groups that are vul-

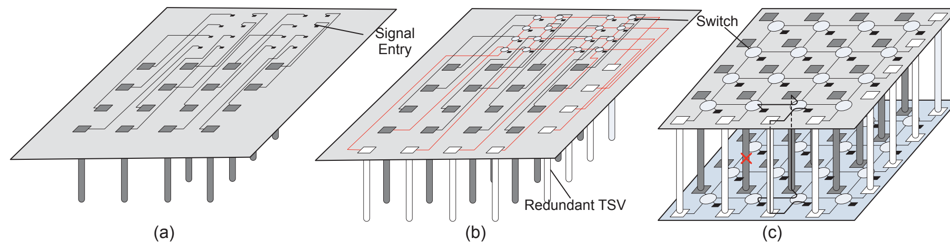


Figure 5.4: Proposed TSV Redundancy Architecture: (a) An Example Physical Layout of Original TSV bundle and their Signal Entries; (b) A Physical Implementation of Proposed TSV Redundancy Architecture; (c) A Conceptual View of the Proposed TSV Redundancy Architecture;

nerable to clustered faults and will become irreparable, although the redundancy ratio is large enough. On the other hand, since prior solutions mainly rely on neighboring spare TSVs for repair, they may suffer from the same clustered faults as the defective one, rendering low repair efficiency. We use two examples in Fig. 5.2 to demonstrate the above concern. For simplicity, the bundled signal TSVs and redundancies are linked by a line in the fault map. Although the redundancy ratio of the prior solutions are all the same (1:2), their repair efficiencies behavior differently after the clustered TSV faults occur. In Fig. 5.2(a), all the prior solutions can successfully repair the random TSV faults that are sparsely located. While in Fig. 5.2(b), all the existing techniques fail to repair the three clustered TSV faults due to the lack of redundancies nearby.

Recently, the clustered TSV faults get noticed by several works [4, 46]. In [4], signal TSVs and redundant TSVs are grouped together as a crossbar scheme. Each signal TSV has r repair candidates if the group is assigned with r redundant TSVs (e.g. Fig. 5.3 (a)). In order to tolerate the clustered faults (e.g. Fig. 5.3 (b)), they either choose to shrink the group size (e.g. Fig. 5.3 (c)) or increase the redundancy ratio (e.g. Fig. 5.3 (d-e)). In order to maximize the yield and minimize the multiplexors, the group size and redundancy ratio are determined through proba-

bility analysis of successful repair. The derived optimal solution, however, is still not "cheap" especially in terms of redundancy cost. That is because the repair structure is fixed in advance, rendering an inflexible repair and hence most of the redundant TSVs remain useless. Motivated by the above, in this chapter, we propose a novel repair framework that is effective and flexible to repair clustered TSV faults with least redundancy cost.

5.3 Proposed TSV Redundancy Architecture

In order to handle clustered TSV fault, our solution is to offer more repair options for each defective TSV. In other words, we try to increase repair flexibility so that a defective TSV can be replaced by a spare that is distant. In this section, we first demonstrate the overall architecture and then detail the switch design and signal re-routing mechanism.

5.3.1 Overall Structure

An example layout of TSVs is shown in Fig. 5.4(a), wherein the signal connecting TSV (noted as signal entry) is located nearby. Around each TSV, there is a "Keep-out-zone" that no logic gate can be placed. Thus, the signal can only be routed into this TSV bundle by wires. It first injects a switch into the TSV-signal link and places the redundant TSVs along two borders of the TSV bundle (Fig. 5.4(b)). Each redundancy TSV links to a switch by wires (shown in red color). Noted that the exactly placement of these redundancy TSVs can be tuned to facilitate the placement and routing in physical design. This physical layout is mapped to a logical TSV grid (see Fig. 5.4(c)) for clarity. In the remaining of this chapter, we will use this logical redundancy architecture instead.

Inspired by the compensation path problem [98], in this architecture, if one signal is disconnected due to a TSV fault (the one with "X" mark), the switches linking two pads of the faulty TSV reroute the signal through a neighbor fault-free

TSV by way of their switches (see the solid line). Since the fault-free TSV is now attached to the previous rerouted signal, its original signal needs to be rerouted as well. This process continues until a redundant TSV on the border is used.

TSVs are usually fabricated in a "regular" manner and grouped as bundles in many 3D-SIC designs [99, 20]; those regularly placed TSVs can be naturally linked together to construct the proposed TSV redundancy architecture. In case that TSVs are not regularly placed, we can also map them into a logical TSV grid and apply our repair architecture (discussed later). Note that, while more hardware resources are needed in the proposed architecture (i.e., additional switches and wires) when compared to earlier TSV redundancy techniques, this hardware cost is well justified by the yield improvement brought with our solution, as shown in our experimental results.

5.3.2 Switch Design and Repair Path Routing

The switch design depends on the placement of redundant TSVs, e.g. in Fig. 5.4(c), which are placed on the east and south borders of the TSV grid. Thus, we constrain

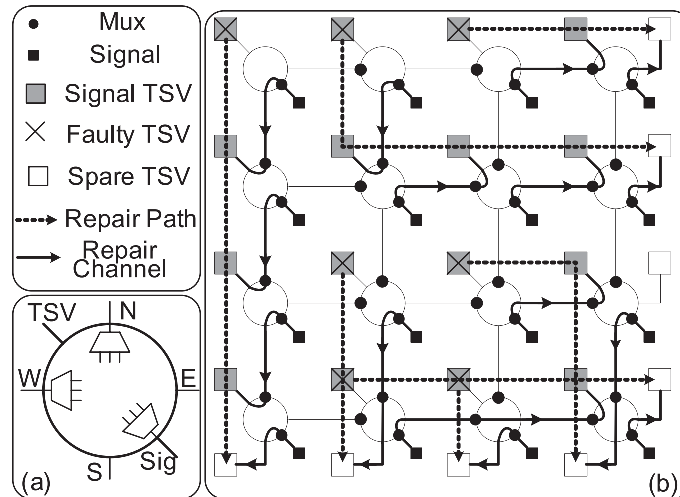


Figure 5.5: Switch Design and Routing Capability. (a)Switch Design; (b)TSV grid with Edge-Disjoint Repair Path.

signals to route from two directions (from west to east or from north to south). Fig. 5.5(a) shows the schematic of the corresponding switch design. The signal entry and its original TSV have two ports in the switch (denoted as *signal port* and *TSV port*). In addition, there are four other ports connecting other switches far apart from four different directions (denoted as *linking ports*). The design principle is that the *signal port* and two *linking ports* (**N**orth and **W**est) have a mux capable of linking to the *TSV port* and the remaining *linking ports* (**E**ast and **S**outh).

We use Fig. 5.5(b) as an example (4×4 grid) to introduce the concept of *repair path* and to show its routing capability. Initially the mux of *signal port* connects to *TSV port*. Once the faulty TSV is detected, the signal port reroutes to another fault free TSV by reconfiguring the connectivity of the switch. This type of physical connection between signal and TSV is represented as a ***repair channel*** (see the solid arrow). Starting from any faulty TSV, there must be a succession of continuous *repair channels* finally terminating in a redundant TSV. We denote this virtual connection from a faulty TSV to a redundant TSV as ***repair path***. For example, the three clustered faults on the top find three disjointed *repair paths* to redundant TSVs (see dashed arrow). The four clustered faults on the bottom also find four disjointed *repair paths*. It is worth noting that the design of the switch guarantees that the *repair paths* can intersect with each other without any contradiction as long as port connections within the switch have no conflict (see the three clustered faults in the bottom).

5.4 Proposed Repair Algorithm

After the fault maps of each TSV grid is obtained via testing [1], a repair algorithm is essential to analyze whether the TSV grid is reparable and generate *repair paths*

[1] Testing is out of the scope of this chapter. Interested readers may refer to [27, 2, 100] for more details.

for each faulty TSV, if possible.

5.4.1 A Maximum Flow Method Based Approach

Consider the TSV grid as a directed graph, wherein each vertex represents a TSV and its corresponding switch while the directed edge connecting two vertices is the wire between two switches (the edge direction depends on the constraint of signal routing directions).

Our problem is to find *edge-disjoint repair paths* for all faulty TSVs, and we can employ the Maximum Flow method [101] to find them. To be specific, we first assign each edge in the graph with an unit capacity "1" to construct a directed flow graph. By adding a super source node that points to those faulty TSVs and merging all the spare TSVs into a target node, the TSV grid is repairable only if the weight of maximum flow is equal to the number of faulty TSVs (see Fig. 5.7(a)).

5.4.2 Additional Delay due to Signal Rerouting

While the above problem formulation and its corresponding solution are simple and effective, it does not take the additional delay introduced during repair into

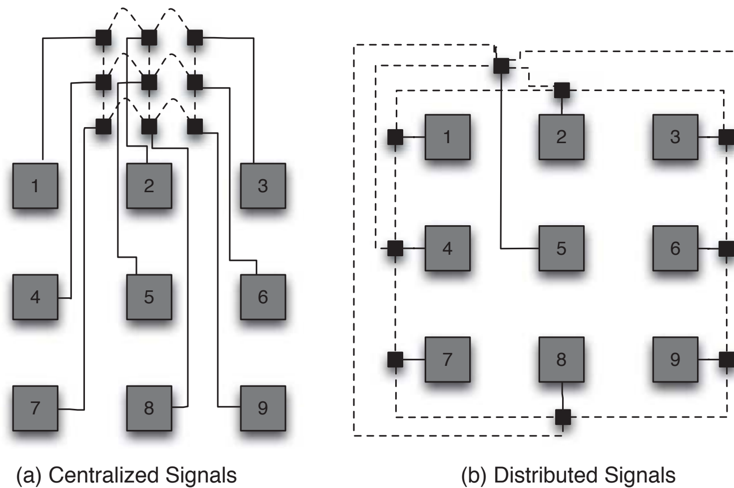


Figure 5.6: Timing Issue caused by Signal Rerouting.

consideration. First, let's take a look at the extra delay caused by the switch inserted between each TSV and its signal entry. Each switch has three Muxes (i.e. six gates). Assuming the average area of a gate is $3125F^2$, the size of the switch is approximately $0.9\mu m \times 0.9\mu m$ in 65nm technology. Comparing to the TSV size (in an order of magnitude of 10μm), it is negligible. To reroute to the neighboring TSV, one signal only need to pass four gates in maximum. Conclusively, it is not the extra hardware in the proposed architecture that contributes the extra delay. Instead, the wire of accessing the neighboring TSV mainly determines the extra delay.

Furthermore, the wire length of rerouting the signal is determined by the layout of signal entries. We classify their layout into two scenarios, one of which is mentioned in Fig. 5.4 and denoted as "centralized signal entries" (see Fig. 5.6(a)), wherein the switches are assumed to be near the signal entry, and thus omitted for clarity. Applications like 3D-NoC and 3D stacked memory prefer to connect TSVs in this way for massive data transformation. Under this circumstance, the extra delay after repairing depends on the different distances of the original TSV and repairing TSV away from the respect signal entries. And this difference is roughly equivalent to the distance between the two TSVs (e.g. repairing TSV 2 using TSV 5). While the extra the delay after repairing the TSV would even decrease in other cases (e.g. repairing TSV 1 by TSV 2). Assuming the TSV distance is in the same order of its pitch ($10\mu m$), this extra delay is limited and negligible, except for some critical signals with tight timing margin.

For the other scenario, the signal entries are distributed near to their respect TSVs (see Fig. 5.6(b)). In this situation, the proposed TSV redundancy architecture confronts with timing issues since we have to take the distance between signal entries (switches) into consideration. For example, repairing TSV 5 by TSV 8 leads the signal to access as far as half of the bundle's perimeter away. A repairing

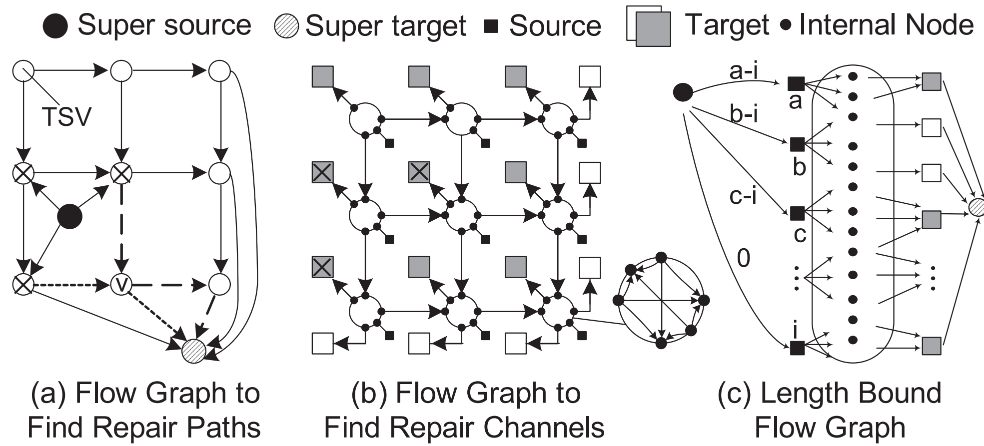


Figure 5.7: Problem Transformation.

is available only if the extra delay of rerouting the signal would not violate the timing constraint. As a result, it is essential to consider timing constraint in the proposed TSV repair algorithm.

5.4.3 Problem Analysis due to Timing Constraint

To guarantee the timing correctness of the circuit after repair without necessarily changing our problem formulation completely, we translate the timing constraint for each "to-be-repaired" fault as length constraints in the flow graph. That is, each wire/mux in the TSV grid is associated with a length weight, and given a length constraint for each signal, the length of *repair channel* (the distance between the signal with the faulty TSV and its reconnected TSV) cannot violate the length constraint.

With the above, directly using maximum flow method to solve our problem is not applicable because: (i). The "flow" in maximum flow method has no sense of length; (ii). If two *repair paths* intersect in the same TSV (e.g. dashed path and doted path intersected in TSV V in Fig. 5.7(a)), a decision has to be made, that one *repair path* should possess the TSV while the other one is bypassed, and

this decision has to consider the timing constraint. This is not a concern in the original maximum flow method. Before introducing our repair algorithm in detail, we prove this problem is a NP problem first, as shown in the following.

First, we transform the flow graph by setting all signals as sources and all the fault-free TSVs as targets (see Fig. 5.7(b)). At the same time, all the ports within switches become internal vertices while all the wires between them are edges in the graph. The edge between TSV and the *TSV port* guarantees that each TSV is used only once, and the length weight is put on these edges. The problem now becomes how to find edge-disjoint paths from all signals to TSVs under the length constraint. Second, we rearrange the flow graph as shown in Fig. 5.7(c), where each source is labeled with length constraint ($a ... i$), among which we suppose i is the minimum length constraint. Then we add a super source/target and links to all the sources/targets. For each link from super source to source, we manipulate a length weight that is the source's length constraint minus the minimum length constraint (e.g., $a - i$). Thus, the original problem becomes a NP-complete *maximum length-bounded flow problem*, which is to find a maximum flow between one source and one target where the length of all flows are bounded by a length constraint [102].

5.4.4 Repair with Length Constraint

For the sake of simplicity, in this chapter, we assume that there is a unified length constraint C, i.e., the one for the most critical signal.

Our heuristic is shown in Fig. 5.8, and the basic idea is to divide sources into groups and apply bounded search for each group. First, we initialize the flow graph by removing those edges that link signals with faulty TSVs and their *TSV port* (line 1). Consider that the *repair channel* can only go towards east and south directions. It is better to conduct a bounded search for those sources from west/north to east/south. This is because the targets and edges chosen for preced-

ing sources are no longer available for the latter sources, thus reducing solution space. At the same time, the sources in the same diagonal lines have no edge connection to each other, which makes it a perfect choice to group them together (line 2). For each source in a group, we first find all possible candidate *repair channels* (c_j) including available edges and targets that satisfies the length constraint using breadth first search (lines 4-5). To avoid an extremely large solution space during exhaustive search, we constrain the candidate *repair channels* in terms of their number of edges/hops. The search bound iteratively increases from "1" as long as it does not exceed some pre-defined maximum bound (line 6). Generally speaking, the larger the search bound is, the more edges the *repair channel* occupies, leaving less solution space for consequent groups. Thus, the *repair channels* with less edges are preferred. We then apply an exhaustive search to find the *repair channels* for each source in this group such that, there is no conflict on edges and targets (line 7). Once such a combination of *repair channels* is found, we confirm this solution by recording the *repair channels* and updating the graph, i.e. deleting the chosen edges and targets in this solution (lines 8-9). We then continue with the next group of sources (line 10). Otherwise, the search bound is increased and the search continues. If no such non-conflict *repair channels* can be found even with the maximum bound, the TSV grid is deemed irreparable (lines 11-12). Otherwise, the heuristic returns the successful repair solution (line 13).

Let us demonstrate how our heuristic works using an example fault map shown in Fig. 5.9(a). The groups are those signals in diagonal lines (doted lines). To simplify the demonstration, we adopt a flow graph like Fig. 5.7(a) and index each node with the row and column numbers ($C_{x,y}$). We mark the faulty TSV as cross ("X") and use a circle to denote the node whose TSV is possessed by previous signals. When $i = 1$, $C_{1,2}$ has faulty TSV and it finds a *repair channel* (dashed gray arrows) to the TSV in node $C_{1,3}$ while the other node in this group finds the *repair channel*

Input : $G = (V, \overrightarrow{E}, w), S, T, \mathcal{L}$
Output : $C = \{c_k(s_k, t_k), s_k \in S, t_k \in T\}$

1 Initialize G, remove faulty TSVs;
2 Classify diagonal sources into groups $P = \{p_i\}$;
3 **For each** group $\{p_i\}$
4   **For each** source $s_j \in p_i$
5   **BFS**: find all possible *repair channels* from s_j to $t \in T$, satisfying the length bound $\mathcal{L}$ and put into c_j;
6   **For** search bound *sb* from 1 to **maximum bound**;
7   find a *repair channel* and target for each source such that no conflict exists;
8   **If** success
9   confirm the solution and update G;
10   **break**;
11   **Else If** not success and $sb = MAX_{hops}(c_j)$
12   The grid is Irreparable;
13 The grid is reparable;

Figure 5.8: Proposed Alogrithm

to its own TSV. When $i = 2$, two signals without their original TSVs find the *repair channels* (dashed black arrows) bounded by 2. The search process is shown in Fig. 5.9(b). The nodes with fault TSV or possessed TSV are shown in grey color and the non-conflicting *repair channels* end on nodes $C_{1,4}$ and $C_{2,4}$ respectively (underlined). After confirming this solution, the edges and TSVs possessed during $i = 2$ are labeled in grey color and no longer available. The process continues for group $i = 3$, and three more *repair channels* are found for this group (solid black arrows).

5.5 TSV Redundancy Architecture Construction

In order to integrate the proposed TSV redundancy technique into the design flow, we plan to insert redundant TSVs and supporting infrastructures, i.e. muxes and

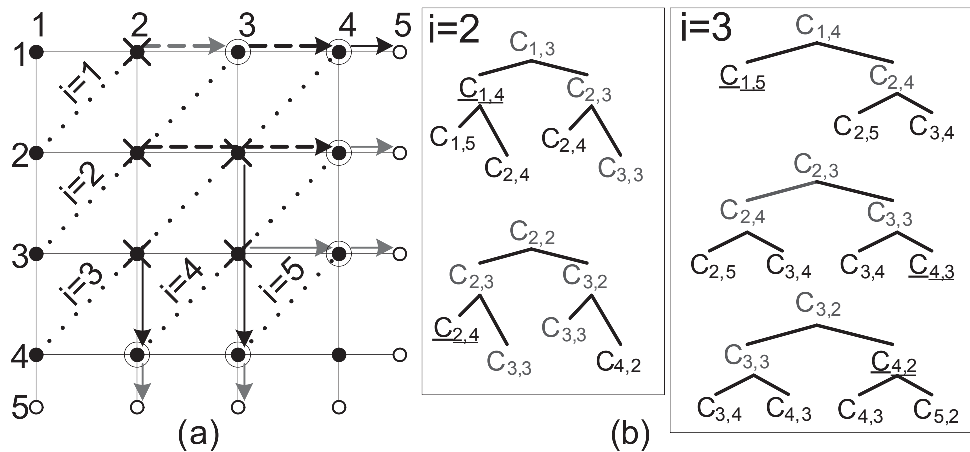


Figure 5.9: Illustration of the Repair Algorithm: (a) Example TSV Grid; (b) Search Procedure Demonstration

wires, right after the TSVs planning but prior to the placement and detail routing. During this process, the key step is to construct the TSV redundancy architecture, i.e. to determine the distance of neighboring TSVs, which influences the reparability in two contradictive aspects. From the perspective of the whole TSV grid, a higher routability indicates a higher reparability since each signal is more flexible to find a replacement. Thus, it is preferred to minimize the *distance* between neighboring TSVs so that more replacement candidate can be reached. On the other side, it is likely that the presence of a single TSV fault increases the chance of more defective TSVs in close vicinity [4, 46]. Once a TSV is found faulty, we are unwillingly to see its successive neighboring TSVs are also faulty, blocking up its repair paths. In that sense, it seems better, on the contrary, to maximize the *distance* between neighboring TSVs as long as this distance fulfill the length constraint. In order to investigate the impact of this distance, we first conduct the probability analysis on the reparability of TSV redundancy grid assuming that the clustered TSV faults are spatially correlated. Then a topology mapping strategy is proposed for TSV grid construction to enhance the reparability.

5.5.1 Defect Probability Model with Spatial Correlation

To analyze the probability of successful repair, i.e. reparability, it is essential to derive a probability model of the clustered TSV faults. The Compound Poisson Distribution [51] is widely accepted to model the clustering effect, in which the defect count follows Poisson distribution compounded with a Gamma function presenting the distribution of defect density. To model the spatial correlation, a center-satellite model [103] is proposed, where the distributions of the cluster centers are described by a two-dimensional distribution function and the distribution of the satellites (defects) with respect to the cluster center is also described by a two-dimensional distribution function. In this case, the defect probabilities in the regions near defect clusters are higher than other regions. Approximately, this defect probability is inversely proportional to the distance from the existing defects [104]. If there are already N_c defects (regarded as cluster centers), the defect probability of TSV_i, p_i can be expressed as

$$p_i \propto \sum_{j=1}^{N_c} (\frac{1}{d_{ic}})^{\alpha} \tag{5.1}$$

where d_{ic} is the distance between TSV_i and existing cluster center and α is the clustering effect.

5.5.2 Reparability Analysis

To analyze the reparability, we first investigate the metrics to determine whether the proposed repair scheme can survive the clustered TSV faults. As stated in the previous section, in order to repair the TSV faults, we have to find edge-disjointed paths from the faulty TSVs to redundant TSVs located in the borders. According to “Max-Flow Min-Cut theorem”, the minimum cut of the induced flow graph must not be less than the number of faulty TSVs. Fig. 5.10(a) shows an irreparable fault map, in which the value of the min-cut (dashed curve) is less than the number

of faulty TSVs. Whereas the fault map in Fig. 5.10(b) has a min-cut equal to the number of faulty TSVs, indicating a reparable solution (black solid arrows). The fault map in Fig. 5.10(c) shows that, on the contrary, although the min-cut of these seven sources is equal to seven, the fault map is still irreparable. As a result, we have the following theory

Theorem 1 *The fault map is reparable if and only if there is no such a sub-set of sources whose min-cut is less than the size of this sub-set.*

Proof *To repair all the faults, we need to find a* repair-path*, as well as edge-disjointed path for each the source. If the number of sources in any sub-set is larger than the min-cut of this subset, then the number of edge-disjointed path is also larger than the min-cut. Since the repair scheme constrain that the* repair-path *can only go towards east and south, based on the "min-cut maximum-flow" theorem, it is no way we can find all edge-disjointed paths for each source that can pass through the min-cut. Thus, the fault map is irreparable.*

With the above theorem, we can model the reparability of the TSV grid by investigating the probability that any fault cluster is reparable. The proposed two repair algorithms are modeled respectively. As [103], the probability of the cluster

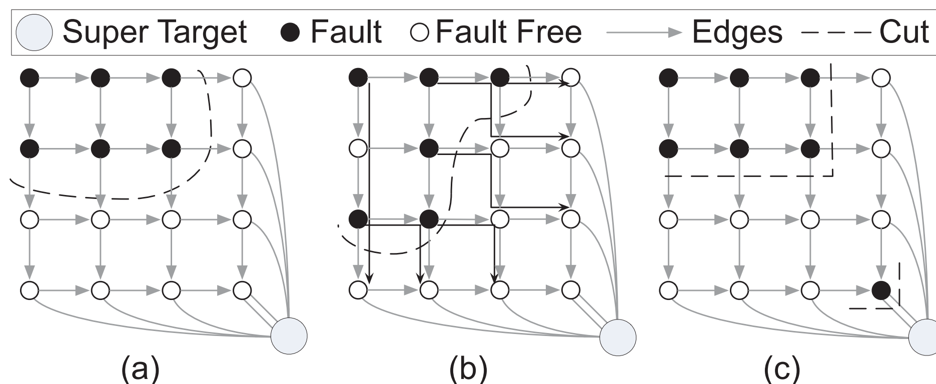


Figure 5.10: The Reparability Condition:(a) Irreparable Example; (b) Reparable Example; (c) Minimal Orthogonal Cut Equals to Min-cut.

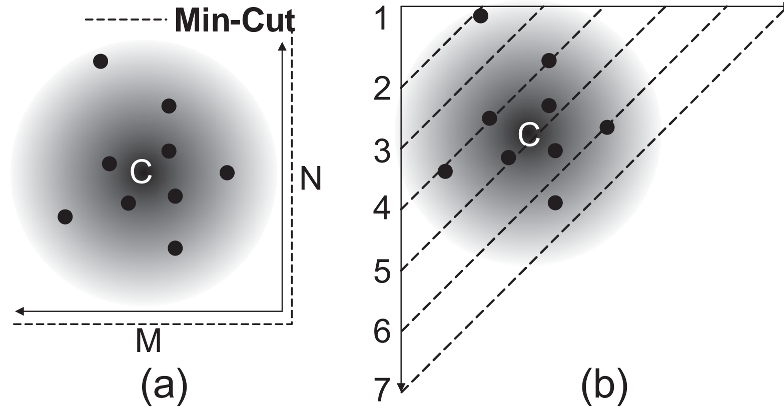


Figure 5.11: Defect Probability Model:(a) Maximum Flow Method; (b) Length Bounded Search Heuristic.

center (c) is assumed to be Poisson distributed. The defect probability of satellite TSVs p_i is given in Eq. 5.1. In order to model the reparability of the maximum flow based algorithm, we assume the size of the fault cluster is M by N and hence the min-cut of the fault cluster is $M+N$ (see Fig. 5.11(a)). The random variable X denotes the number of defective TSVs among totally $M \times N$ TSVs. As mentioned above, this fault cluster is reparable only if the number of faulty TSVs in the cluster is less than the min-cut $M+N$. First we randomly select x $(x \leq M+N)$ TSVs from the total $M \times N$ TSVs as a faulty TSVs set F_x, resulting in totally $\binom{M \times N}{x}$ combinations. We obtain the probability of each combination C_i by calculating the product of defective probabilities for the faulty TSVs selected in F_{x,C_i} as well as the non-defective probabilities for the rest of TSVs that are not selected $(\Omega - F_{x,C_i})$. Then the probability of all the combinations are accumulated for this x. Finally, the reparability is formulated by cumulating the derived probabilities of all possible xs as

$$P(X \leq M+N) = \sum_{x=0}^{M+N} [\sum_{i}^{\binom{M \times N}{x}} [\prod_{k \in F_{x,C_i}} p_k \prod_{j \in \Omega - F_{x,C_i}} (1-p_j)]] \tag{5.2}$$

where p_k and p_j are the defective probability of TSV k and j obtained from Eq. 5.1. For the sake of simplification, we just assume an average defective probability and Eq. 5.2 can be approximated as follow:

$$P(X \leq M+N) = \sum_{x=0}^{M+N} [\binom{M \times N}{x} p^x (1-p)^{M \times N - x}] \tag{5.3}$$

where p are the average defective probability.

In order to apply the "min-cut maximum-flow" theorem to the proposed length bounded search heuristic, we can draw a set of right triangles with their hypotenuses representing the min-cuts. These right triangles are sorted in ascending order of the size until the last one embodies the whole fault cluster (see Fig. 5.11(b)). Empirically, the set of right triangles represent the repair process that the TSV groups in the diagonal lines (i.e. the hypotenuses) are repaired iteratively. Once a faulty TSV is repaired according to the heuristic, a fault-free TSV is occupied in next orthogonal hypotenuse. Thus we need to find another fault-free TSV for this occupied TSV in next iteration. From a higher perspective, the number of occupied TSVs is accumulated in successive hypotenuses iteratively. This process continues until the redundant TSVs are reached. Obviously, the repair scheme fails as long as the number of faulty TSVs embodied by any right triangle is larger than the min-cut (number of edges) long with the hypotenuse. Suppose we need n right triangles $T_n = \{t_1, t_2, ..., t_n\}$ $(Size(t_i) < Size(t_{i+1}))$ to embody the fault cluster. The random variable x_i denotes the number of faulty TSVs inside ith right triangle which contains $Num(t_i) = \frac{i \times (i+1)}{2}$ TSVs in total. The reparability can be approximated in a same manner as follow

$$\prod_{i=1}^{n} P(x_i \leq Mincut(t_i)) = \prod_{i=1}^{n} P(x_i \leq 2i) \tag{5.4}$$

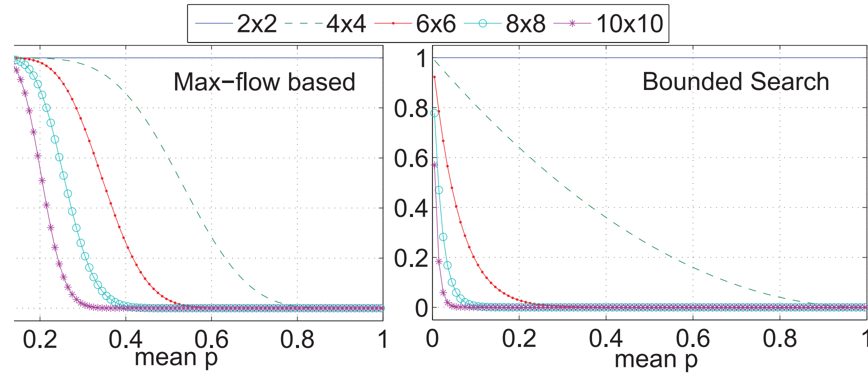


Figure 5.12: Reparability Approximation Trends with respect to Average Defect Probability and Cluster Size (from 2×2 to 10×10).

For any right triangle t_i, the reparable probability is

$$P(x_i \leq 2i) = \sum_{x_i=0}^{2i} [\binom{Num(t_i)}{x_i} p^{x_i}(1-p)^{Num(t_i)-x_i}] \tag{5.5}$$

where p are the average defective probability.

Based on the above equations, we know that the key factor is the distance between TSVs. According to Eq. 5.1, the smaller distance between TSVs indicates that the defect probability p becomes higher. Although, the exact defect probability is related to the clustering effect and defect intensity, the TSVs near the cluster center would have a very high probability to fail. We can approximately assume the average defect probability within this fault cluster is approaching to 1 (i.e. $p \rightarrow 1$), which means the fault has more likelihood to occur. On the contrary, if this distance is larger, p is approaching to a fixed failure rate, which indicates a random defect.

Now let's consider this trend according to the approximated reparability from above equations (see Fig. 5.12). As the average defect probability increases, the reparability drops dramatically. At the same time, it becomes more difficult to repair those fault clusters with larger sizes. Comparing the optimal solution derived by maximum flow algorithm, the bounded search heuristic is more vulnerable to

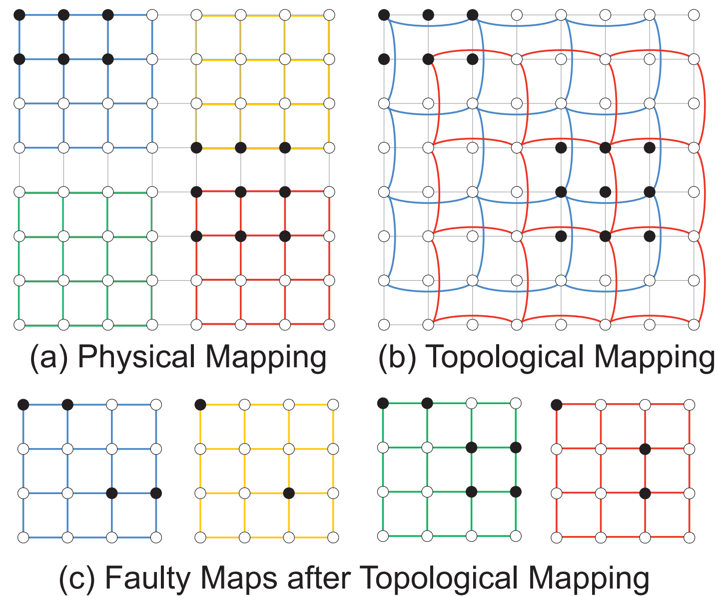


Figure 5.13: Examples of the Topology Mapping and the Change of Faulty Map in TSV Grid.

the clustering effect. In order to enhance the reparability, it is urgent to reduce the average defect probability and reduce the size of fault cluster based on above analysis.

5.5.3 Topology Mapping

In order to enhance the reparability, the basic strategy is either increasing the distance between TSVs or shrinking the size of the faulty TSV cluster. However, the distance of TSVs is normally determined according to the design specification and can hardly be changed afterwards. Similarly, the size of the faulty TSV cluster is also beyond our control. Fortunately, we can achieve both the above strategies by topology mapping. A topology mapping is a process that maps the TSVs from the physical layout to a "logical" TSV grid. Thus, the neighbors in this "logical" grid (topological neighbors) are not necessarily the neighbors in the physical lay-

out. To demonstrate this concept, we first show a straightforward mapping that the topological neighbors are the exact physical neighbors. In Fig. 5.13(a), a 8×8 TSV bundle is split to four 4×4 TSV grids. Although the proposed TSV redundancy architecture is inherently able to tolerate TSV clustering fault, applying the above topology disposes the TSV redundancy architecture to the risk of unsuccessful repair. For example, two faulty TSV clusters (black dots in Fig. 5.13(a)) are located into two TSV grids and make them irreparable. To overcome this vulnerability, the physical neighboring TSVs are distributed apart in derived grid or even into different grids. Fig. 5.13(b) renders such a mapping that the topological neighbors are 2-hop away (with one gapped TSV in between) in physical layout. The derived four 4×4 TSV grids are shown in Fig. 5.13(c) and are able to repair all the faulty TSVs. By employing this topology mapping, the clustered faulty TSVs are distributed, indicating a larger distance between TSVs and smaller cluster size and hence the reparability is improved.

The topology mapping would increase the delay of signal rerouting, however, the affect is limited. As we discussed in Section 5.4.2, the extra delay is roughly equal to the distance between "to-be-repaired" TSV and the repairing candidate TSV, if the switches and signal entries are centralized. Mapping TSVs in 2-hop distance away in original grid into a logical TSV grid would double this extra delay. Since the state-of-art TSV process technology can achieve $< 10\mu$m pitch, doubling this delay after topology mapping would not affect the delay too much. Besides, the topology mapping could also be used upon the proposed timing aware "length bounded search" scheme, which can improve the yield while keep the signals under the timing constraint.

Work	Defect Rate	TSV Pitch	Yield w/o Spare	TSV Number
IBM’05 [105]	13.9E-6	0.4μm	95% 88%	1k$-$10k
IMEC’06 [106]	40.0E-6	10μm	67%	10k
HRI’07 [107]	9.75E-6	-	68%	100k
HRI’09 [39]	7.95E-7	-	$\geq$90%	100k
SAMSUNG’09 [1]	*0.63%*	-	15%	300

Table 5.1: TSV Related Experimental Setup.

5.6 Experimental Results

5.6.1 Experimental Setup

Defect rate and the number of TSVs are two key parameters for yield calculation that also determine the effectiveness of a repair scheme. The TSV number associates with a specific design, while, the defect rate is related to a specific TSV process technology (e.g. the pitch and aspect-ratio of TSV). Various settings of these parameters presented in previous works are summarized in Table 5.1. A commonly acceptable failure rate is between 10^{-5} and 10^{-4} except for two extremes (i.e. Samsung and HRI’09). in this chapter, we vary both these two parameters in a reasonable range.

We set up three types of sample chips (small/medium/large) based on TSV count, that is,1k, 16k and 128k. We group TSVs as bundles (16×16 for small chip and 32×32 for medium and large chip) and place them randomly on the chip. We vary the TSV failure rate in a range from 10^{-5} to 10^{-4}. In the experiments, we use three repair schemes as baseline solutions for comparison. The crossbar scheme [3] with 0.25 redundancy ratio is denoted as “8:2”, which indicates that eight signal TSVs and two redundant TSVs are bundled together. Similarly, the signal-shifting scheme [2] and signal-switching [1] scheme with a 0.5 redundancy ratio are denoted as “2:1” and “4:2” respectively. The proposed scheme allocates one column and one row of redundant TSVs on two borders of each TSV grid,

denoted as "$R \times C : R + C$". For ease of comparison, we set up two TSV grids, "$4 \times 4 : 8$" and "$8 \times 8 : 16$", with redundancy ratio 0.5 and 0.25, respectively. The "2HD" indicates the topological neighbors have two hops distance in physical layout. After topology mapping with 2-hop distance, TSV grids "8×8" contain four 4×4 sub-grids. Similarly, a 16×16 grid has eight 4×4 sub-grids after a 4-hop distance topology mapping. And hence, numbers of redundancy TSVs are 32 and 128 respectively. For the sake of simplicity, the distance between neighboring TSVs within the TSV bundle are all set equally. The length constraint is decreased from 4 to 1 time(s) of this distance (denoted as "$-T(4)$ to $-T(1)$").

We conduct two sets of experiments. In the first set, the repairing efficiency of proposed maximum-flow based method with and without topology mapping are compared with prior techniques. And the proposed length bounded search algorithm is evaluated under diverse length (timing) constraints in the second set. To study repair efficiency on clustered TSV faults, we adopt the mentioned Compound Poisson Distribution combined with spatial correlation in the both sets. The clustering effect parameter (denoted as $Alpha$) is varied from 0 to 3. Particularly, when $Alpha$ is equal to 0, it degrades to a Poisson distribution.

5.6.2 Results and Analysis

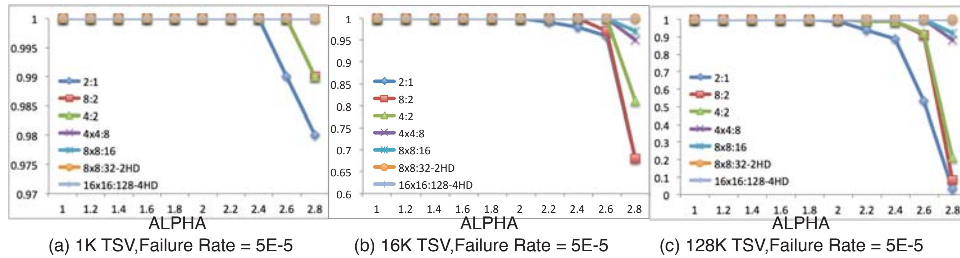


Figure 5.14: Repair Schemes Varying Alpha from 1 to 3.

The experimental results with $Alpha < 1$ are not shown in the chapter, as all the repair schemes can guarantee 100% yield. The main reason is that the yield of

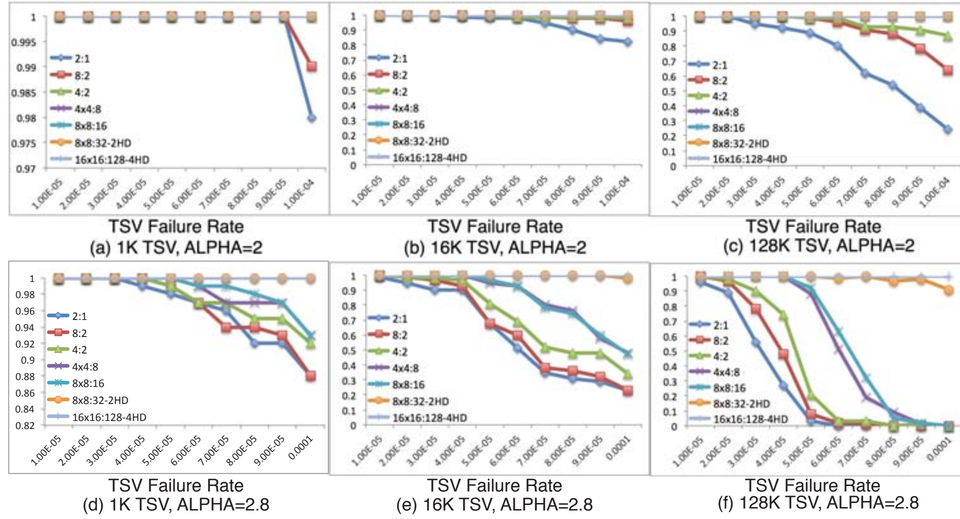


Figure 5.15: Repair Schemes Varying TSV Failure Rate with Fixed Alpha.

a single TSV is quite high in state-of-art process technique. However, the TSVs yield in real product and some prototype test chips is not that optimistic due to many reasons, one of which is the clustering faults.

Fig. 6.6(a)-(c) show the yield comparison by varying the clustering effect *Alpha* with a fixed TSV failure rate set to 5×10^{-5} . It can be observed from the figures that with the increasing clustering effect, all the previous repair schemes suffer notable yield loss. The yield drop becomes more severe as the TSV number increases. However, the proposed maximum-flow based technique has much less vulnerability (at most 5% yield drop in medium chip while 10% yield drop in large chip). After the topology mapping, the proposed technique can achieve a near 100% yield. The vulnerability to clustering fault is different across these repair schemes. To be specific, signal shifting scheme $2:1$ is worse than others because it fails whenever two adjacent TSVs has faults at the same time. The signal switching and cross-bar schemes behavior better, however, their efficiency drop fast as the number of TSVs increases. That's because, there are more chances to occur a triple TSV faults nearby.

Fig. 6.8(a)-(c) show the yield comparison by varying the TSV failure rate with a fixing *Alpha=2*. With the ramping TSV failure rate, the yields of baseline repair schemes drop with different gradients while the maximum-flow based scheme can still keep a very high yield. To be specific, the repair efficiency of signal-shifting scheme is less than other solutions because it can only tolerate one fault in its repair unit. The crossbar is better than the preceding one but worse than signal-switching scheme. The signal-switching scheme performs better for small chip and medium chips with up to 98% yield, but in the case of large chips, the yield drops nearly 10%. To observe closer into the clustering effect, we ramp up the *Alpha* to 2.8 indicating a high likelihood of catastrophic defects (see Fig. 6.8(d)-(e)). As the TSV number increases, all the repair schemes are revolved into three groups. The baseline solutions are converged in the first group with significant yield drop. While the proposed maximum-flow based technique is much better to tolerate TSV clustered faults, however, still suffers when TSV number is large enough for catastrophic defects to happen. After topology mapping, the repair efficiency of the proposed technique significantly increases. Comparing to the two topology mapping strategies, the one with 4-hop distance can keep the yield closer to 100% . Obviously, the 4-hop distance mapping make the topology neighbor distant and tolerate faults with larger cluster size.

From the above results, we can observe that the redundancy ratio is not the only dominating factor for the final yield, e.g., the 8 : 2 scheme is better than the 2 : 1 scheme, even though its redundancy ratio is only half of the latter one. This is because the flexibility for repair has a significant impact on TSV repair efficiency. Let us consider a TSV bundle containing 8 signal TSVs. As long as there are no more than two TSV failures, the cross-bar 8:2 scheme can successfully repair them. However, if two TSV failures occur the same repair unit, the 2:1 signal shifting scheme would fail. When the TSV failure rate is not very high, the

possibility to have more than two TSV failures in the bundle (and can be repaired with 2:1 repair scheme) is lower than the possibility to have irreparable double TSV failures. Therefore, 8:2 repair scheme results in higher yield when compared to 2:1 scheme. Similarly, the $8 \times 8 : 16$ TSV grid has higher yield than the $4 \times 4 : 8$ TSV grid when the cluster effect is strong. This is also expected since the 4×4 TSV grid is more vulnerable to the fault cluster whose area may cover the majority of the 4×4 TSV grid or be even larger than the TSV grid. Under this situation, it is by no mean possible to repair such fault cluster.

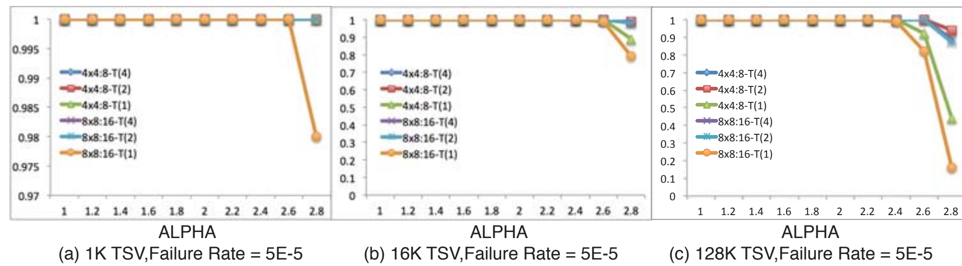


Figure 5.16: Time Constrained Repair Varying Alpha from 1 to 3.

To evaluate the timing effect among the proposed bounded search heuristic, we reproduce the above comparison among different length constraints. Fig. 6.9(a)-(c) show the yield comparison by varying the clustering effect *Alpha* with a fixed TSV failure rate equal to 5×10^{-5}. Both 4×4 and 8×8 TSV grids with 1-hop length constraint ("$T(1)$") suffer from notable yield loss. This is expected because the *repair path* can only reach one hop of TSVs for repair and hence it is vulnerable to any TSV faults in close proximity.

Fig. 6.10(a)-(c) show the yield comparison by varying the TSV failure rate with a fixing *Alpha=2*. With the increase of TSV failure rate, only the proposed repair scheme with 1-hop length constraint suffers the yield drops. After we raise the clustering effect to 2.8 (see Fig. 6.10(d)-(e)), the proposed repair scheme under all pre-defined length constraint suffer the yield drop in different gradience. For

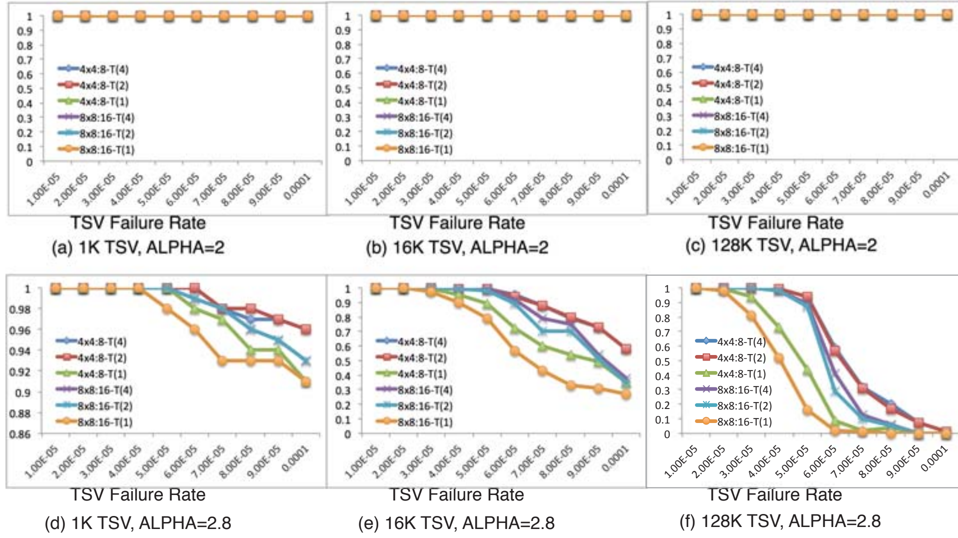


Figure 5.17: Time Constrained Repair Varying TSV Failure Rate with Fixed Alpha.

bounded search algorithm, the 4×4 scheme is always better than the 8×8, which is different from the maximum-flow based method. That is because, the search bound limits the flexibility so that some fault maps, reparable in maximum-flow based method, become irreparable using bound search algorithm. An interesting observation is that, increasing the search bound only improves the yield in a limited degree (see "$T(4)$" and "$T(2)$"). As long as the more-than-1-hop *repair path* is allowed, the proposed timing aware repair scheme can guarantee a very high yield under our experimental settings. This observation also supports the advantage of topology mapping. Suppose the signal can reroute to 4 hops away in original TSV grid, it can only reroute to 2 hops away after a 2-hop distance topology mapping. According to above observation, as long as the timing constraint is larger than the 2-hop distance, it is worthwhile to apply the topology mapping.

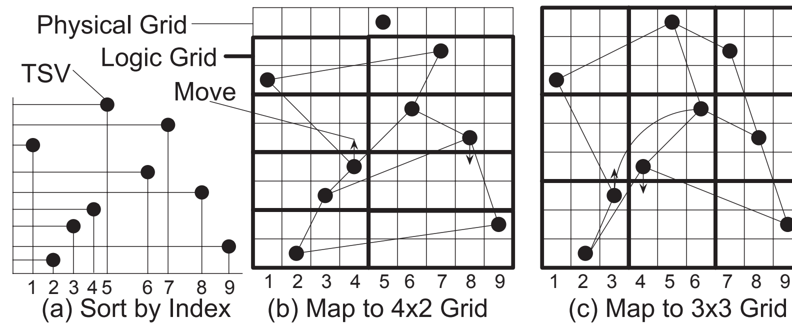


Figure 5.18: Mapping Irregular Placed TSVs.

5.7 Discussion

5.7.1 TSV Grid Construction

TSVs are usually bundled together in a 3D-SIC design, and the proposed TSV redundancy scheme can be directly applied to such designs if we treat each TSV bundle as a grid. However, sometimes the shape of the TSV bundle may not be suitable for efficient repair (e.g., a 1×32 TSV bundle). There are also designs with irregularly-placed TSVs. Under the above circumstances, it is essential to construct "logical TSV grids" for repair.

We suggest a "cut and merge" strategy for TSV grid construction when TSVs are bundled together but their shapes are not suitable for our repair scheme. Given a rough size of TSV grid based on the TSV failure rate and TSV redundancy ratio, we cut the TSV bundle according to the grid's size. For TSV bundles with high aspect-ratio (e.g., 1×32), we can first merge two columns into one column and the cluster becomes 2×16 grid. Such merging process continues until the shape is proper. It should be noted that, the merging process would increase the length of edges in the logical TSV grids.

When TSVs are placed irregularly, we propose a simple mapping method as

follows. First, we divide the layout into blocks until the number of TSVs in each block is roughly equal to the size of logical TSV grid. Then, we index each TSV with its X-/Y-coordinate (see Fig. 5.18(a)), which indicates their relative positions amongst TSVs. Next, we construct grids and place TSVs into them according to the index obtained earlier. After that, we map them into the corresponding logical TSV grids (e.g., 4×2 in Fig. 5.18(b) and 3×3 in Fig. 5.18(c) in bold). Finally, we move TSVs so that each cell of the logical grids is assigned with one TSV only. Since the physical grid maintains the relative position of TSVs, we are able to find a good TSV grid construction with few TSV movements.

There are some other considerations during TSV grid construction. For example, it would be beneficial to have timing critical signals with shorter *repair channels* in a grid so that we can have more repair candidates, which will be explored in our future research work.

5.7.2 Cost Analysis

Under the same redundancy ratio, the proposed repair scheme has higher hardware cost when compared to other TSV repair solutions, i.e., our redundancy scheme requires three 1-to-3 muxes for each TSV, while signal-shifting and signal-switching have one 1-to-3 mux for each signal TSV. However, the extra cost is justified by the corresponding significant TSV yield improvement. It should be noted that, the amount of redundancy is configurable in the proposed architecture. That is, we can adjust the redundancy ratio by either varying the TSV grid size or changing the redundancy allocation scheme (e.g., we can allocate redundant TSVs only on one border). Table 5.7.2 shows a simple comparison of cost assuming 1k TSVs. As calculated above, the size of each switch is approximately $0.9\mu m \times 0.9\mu m$, which is negligible comparing to a TSV ($10\mu m \times 10\mu m$). As shown in the Table II, the dominant factor of area is the TSV, instead of the extra logic gates. More impor-

tantly, the TSV manufacturing cost is much larger than logic gates and the spare TSVs are also suffering the yield problem. From this perspective, the proposed repair scheme requires much less redundant TSVs than existing solutions, rendering less hardware overhead, while, can achieve a higher yield at the same time.

1k TSVs	4:2	8:2	4x4:8	8x8:16	16x16:32
#Spare TSVs	512	256	512	256	128
#Extra Muxes	1k	2k	3k	3k	3k
Areaμm^2	52010	27220	53630	28030	15230

Table 5.2: Cost Comparison for 1k TSVs.

Finally, the runtime of the proposed repair algorithm is very small. For large dies with hundreds of thousands TSVs, it takes only tens of millisecond to obtain the repair solution, if any.

5.8 Conclusion

In this chapter, we propose a novel TSV redundancy architecture and the corresponding repair algorithm for yield enhancement of 3D-stacked ICs. When compared to prior techniques, the proposed solutions enable faulty TSVs to be repaired by spares that are distant, thus is suitable for repairing clustered TSV faults. Experimental results demonstrate the effectiveness of the proposed technique.

□ **End of chapter.**

Chapter 6

In-Field TSV Repair for Reliability Enhancement

6.1 Introduction

As introduced in Chapter 1, TSV fabrication involves several disruptive manufacturing technologies, which leads to new types of defects [108]. These defects are often latent and difficult to screen during manufacturing test, but their impact can be significant during field operation, leading to reduced service life of 3D ICs [96, 109]. Burn-in for screening latent defects during manufacturing is expensive and its effectiveness for new TSV defect types has yet to be thoroughly characterized. Therefore, repair solutions are needed in order to exploit the potential of 3D ICs and facilitate commercialization.

Various TSV redundancy allocation techniques and their corresponding repair algorithms have been proposed in the literature [1, 2, 110, 3, 111, 112, 4, 113, 114, 46, 115]. While effective for repairing TSV manufacturing defects occurred at $t = 0$, these solutions are not readily applicable for in-field repair of TSV latent defects that manifest themselves at $t > 0$. This is because the repair solution obtained with the *deterministic* repair algorithms used in these techniques may not satisfy the timing requirement of the circuit due to circuit aging, thereby rendering the repair solution less effective. To tackle the above problem, this chapter presents a novel

in-field TSV repair solution to enhancement the lifetime reliability of stacked 3D ICs. We first propose an efficient TSV repair algorithm that is able to significantly improve the mean-time-to-failure (MTTF) of TSV grids through the judicious use of spares, as demonstrated by our experimental results. Then, we enhance the TSV redundancy architecture in chapter 5 by allowing redundancy sharing across neighboring TSV grids.

The remainder of this chapter is organized as follows. Section 6.2 presents related works and further motivates this chapter. In Section 6.4 and Section 6.5, we detail the proposed in-field TSV repair framework and the corresponding repair algorithm, respectively. Experimental results on 3D benchmark designs are next presented in Section 6.6. Finally, Section 6.7 concludes this chapter.

6.2 TSV Latent Defects

During TSV fabrication, the temperature is first increased for copper electroplating and then brought down to the ambient temperature. Owing to the large difference in coefficients-of-thermal-expansion (CTE) of the copper TSVs and that of the silicon [116], however, tensile stress inevitably appears on the silicon [117]. Such thermal-mechanical stress is likely to cause TSV interfacial cracks (see Fig. 6.1) that is usually undetectable during manufacturing test [48]. The forces induced by residual stress in the 3D structure cause the crack to grow during field operation, thereby increasing the delay of critical paths with TSVs (if any) and eventually forming an open defect [96]. Moreover, a number of recent works examined the classical electromigration (EM) failure mechanisms in 3D ICs, showing that TSVs are prone to EM-induced voiding effects [48, 49, 50] (see Fig. 6.1). Similar to TSV interfacial cracks caused by thermal-mechanical stress, EM-induced voids increase TSV resistance, causing path delay faults and eventually TSV open defects. Note that TSV-induced stress also reduces the reliability of nearby transistors and

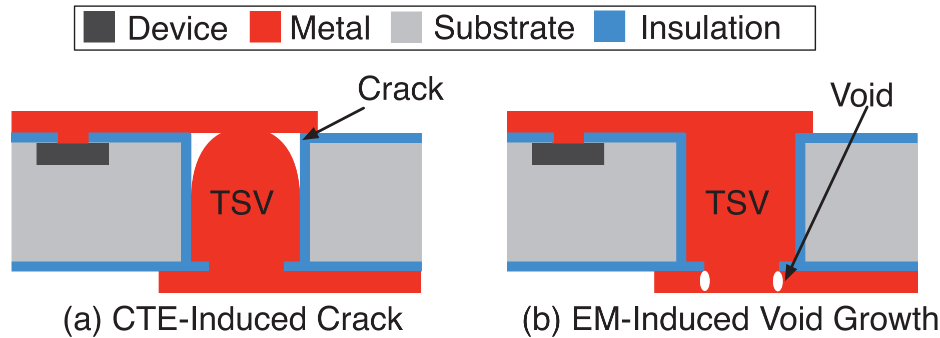


Figure 6.1: Illustration of some TSV Latent Defects

metal wires, and various analytical models and reliability-driven physical design techniques have been presented in the literature to mitigate this problem [108]. However, we limit the scope of this chapter to the repair of TSV latent defects only.

6.3 Motivation for In-Field Repair

Various TSV repair solutions have been proposed in the literature for manufacturing yield enhancement [1, 2, 110, 3, 112, 4, 113, 114, 46, 115]. In chapter 5, a low-cost and flexible TSV redundancy architecture was proposed, which enables defective TSVs to be replaced with distant spares to tolerate clustered faults and it is shown to have higher repair capability than previous methods. Considering the fact that neighboring TSVs usually suffer from similar thermal and mechanical stress, we leverage this TSV redundancy architecture in this work to enable repair of clustered latent faults within a TSV grid.

Unlike TSV repair at $t = 0$ for yield enhancement, the objective of in-field repair for TSV latent faults at $t > 0$ is to increase the MTTF of 3D ICs. This problem is especially difficult due to circuit aging. Pervious TSV repair solutions have focused on the replacement of defective TSVs with fault-free ones, i.e., the repair

algorithms start from faulty TSVs and try to find repair paths to spares, without explicitly considering the impact of the repair solution on signal delays. Such repair methodology is generally applicable for detectable manufacturing defects such as opens and shorts when the distance between the failed TSV and its corresponding spare is not large.

However, both TSVs and other circuit elements wear out during field-operation. On one hand, it is likely that the "replacement-oriented" repair solution provided with existing methods violates signal timing requirements after shifting or rerouting, thereby leading to new "faulty TSVs". On the other hand, a faulty TSV linking to a particular signal might be a good one if it links to another signal instead. This is because, a TSV fault occurring online is not necessarily a catastrophic open/short defect, but often a delay fault that cannot meet the timing requirement of critical paths going through it due to circuit degradation. Consider the example TSV grid shown in Fig. 6.2(a). Signal S_1 needs to be rerouted due to the latent defect that is manifested on its corresponding TSV. However, it may fail again if it is rerouted to use TSV_2 originally linked to S_2, generating a "new" TSV fault even though this TSV is fault-free. Such fault propagation may eventually make the TSV grid irreparable, even though a more sustainable repair solution exists as shown in Fig. 6.2(b)).

Consequently, for in-field TSV repair, we should not focus only on faulty TSV replacement and simply find a repair path for each faulty TSV. Instead, we are to find the set of signal-TSV pairs that satisfy the timing requirement of every signal. Whether a signal and a particular TSV can be paired together is known only after we conduct online testing of those circuit paths going through the TSV, due to the difficulty to predict change of signal timing slacks with circuit aging. The above considerations have motivated the new in-field repair technique investigated in this chapter.

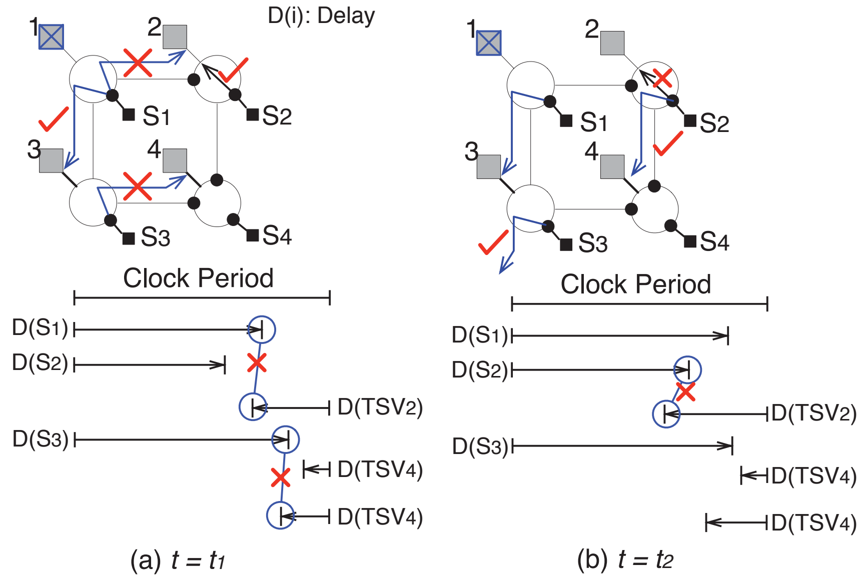


Figure 6.2: An Example to Motivate the Need for Careful In-Field Repair.

6.4 In-Field TSV Repair Framework

In order to conduct in-field repair for TSV latent defects, we first need to be able to test and diagnose faulty TSVs in an online manner. To achieve these objectives, as in [118], we assume the existence of a processor core and non-volatile memory in the system for test and diagnosis purpose (see the conceptual architecture shown in Fig. 6.3). This assumption is reasonable because 3D logic-on-logic ICs or 3D logic-memory designs of the near future are likely to be large multiprocessor system-on-a-chip (MPSoC) designs. Such designs provide the most compelling motivation for high-density 3D integration. To be specific, the non-volatile memory stores the test and diagnosis patterns for TSV faults, our in-field repair algorithm and the repair signature for each TSV grid, while a processor core is called upon for online test and repair, triggered periodically or by events.

6.4.1 Online Test and Diagnosis

As discussed earlier, TSVs suffer from interfacial cracks and EM-induced voids and such latent defects usually manifest themselves as hard-to-predict timing errors on critical paths with TSVs. From this perspective, TSV BIST techniques (e.g., [119]) are insufficient for in-field test and diagnosis because they target on faults occurred in individual TSV structure (and often consider TSV open/short only) instead of delay faults of circuit paths with TSVs. For example, as discussed in Section 2.2, using a fault-free TSV to replace a faulty one does not necessarily lead to a valid repair solution because of the unknown signal timing slack changes with circuit aging.

Consequently, it is important to online test those critical paths that go through TSVs. To be specific, for each TSV, we need to pick one or more long paths that go through it and store the corresponding path delay test patterns in non-volatile memory (in a compressed form to reduce the storage requirement, whenever possible).

Note that, we try to overcome the delay fault on a particular path with TSVs by signal rerouting using other TSVs. Even though this strategy mainly targets TSV degradation/failure, it can also be used to target for path delay faults caused by the degradation of other on-path circuit elements. That is, as long as the identified repair solution is confirmed to be valid with online testing, it is not necessary to root-cause the path delay fault to a particular circuit element.

6.4.2 Spare TSV Sharing and Reconfiguration

Due to the clustering effects of latent faults, unless the redundancy ratio is quite high, we may still run into the situation that some faulty TSV grids lack spare TSVs while the others have many redundant TSVs. We therefore propose to enhance the TSV redundancy architecture presented in chapter 5 by allowing spare TSV

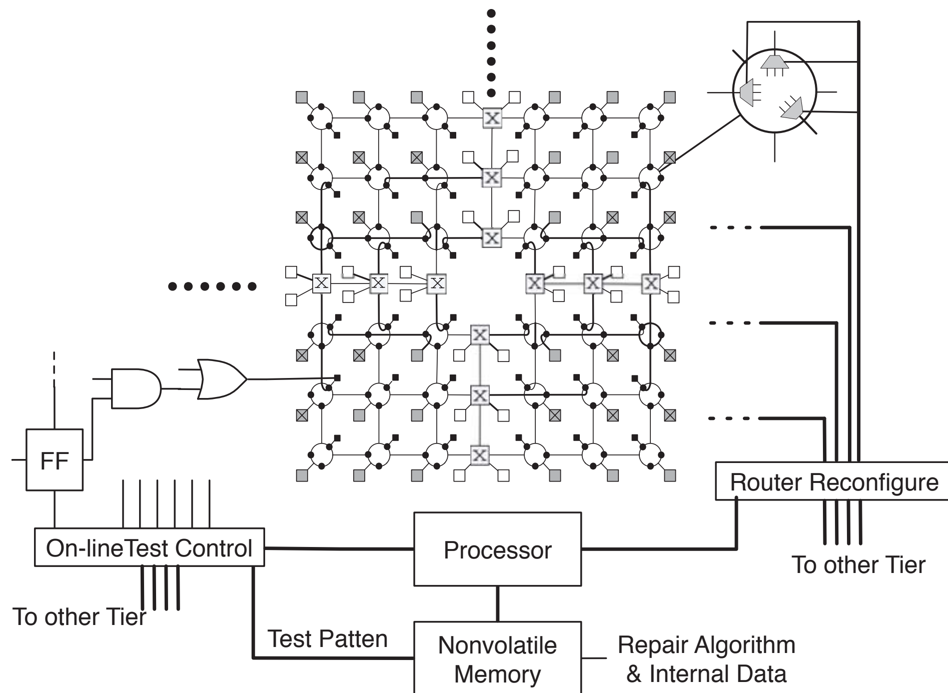


Figure 6.3: Illustration of the TSV Redundancy Architecture.

sharing between TSV grids, as shown in Fig. 6.3.

Given the above TSV redundancy architecture for a 3D IC, the design flow of the proposed in-field repair solution is as follows. With online testing triggered periodically or by events, if a particular path with TSVs is found to be faulty, our TSV repair algorithm (detailed in Section 6.5) is called upon to obtain a possible repair solution. Afterwards, we rerun online testing to check whether this solution is acceptable. The above procedure iterates until a valid repair solution is achieved. The 3D IC is regarded as being irreparable if the circuit is not free of path delay faults after tall the possible repair solutions have been considered.

6.5 Proposed Repair Algorithm

In this section, we first formulate the in-field repair problem and then present details of the proposed repair algorithm.

6.5.1 Problem Formulation

For a 3D IC with TSV redundancy architecture as shown in Fig. 6.3, the in-field TSV repair problem is formulated as follows:

Given the set of signals $\mathbf{S} = \{s_1, s_2, ..., s_n\}$ and the set of TSVs $\mathbf{T} = \{TSV_1, TSV_2, ..., TSV_m\}$ $(n < m)$, our goal is to link every signal in $\mathbf{S}$ with a dedicated TSV in $\mathbf{T}$ under the following conditions: (i) all signal-TSV pairs are routable with the given TSV redundancy architecture; (ii) it is confirmed with online testing with no timing violations.

As we need to invoke the online testing procedure whenever there is a possible repair solution, it is preferable to reduce the number of trials for valid repair.

6.5.2 In-Field Repair Algorithm

In order to solve the above problem, we construct a bipartite graph to store all "possible" signal-TSV pairs, namely *STpair-graph* in this chapter. Fig. 6.4 (a) presents an example *STpair-graph* at $t = 0$. In this graph, one side is the signal set $\mathbf{S}$ while the other side is the TSV set $\mathbf{T}$, and an edge exists for a possible signal-TSV pair that has the following two properties: (i). there is at least one routing path from the signal to the TSV in the flow graph; (ii). there is no *confirmed* timing violation for this signal-TSV pair. *STpair-graph* gets updated with online testing results, i.e., an edge is deleted if the corresponding signal-TSV pair fails path delay test, and a TSV and all its edges are removed when it has a catastrophic failure, e.g., a full open defect.

A valid repair solution is hence a *maximum matching* of the *STpair-graph*

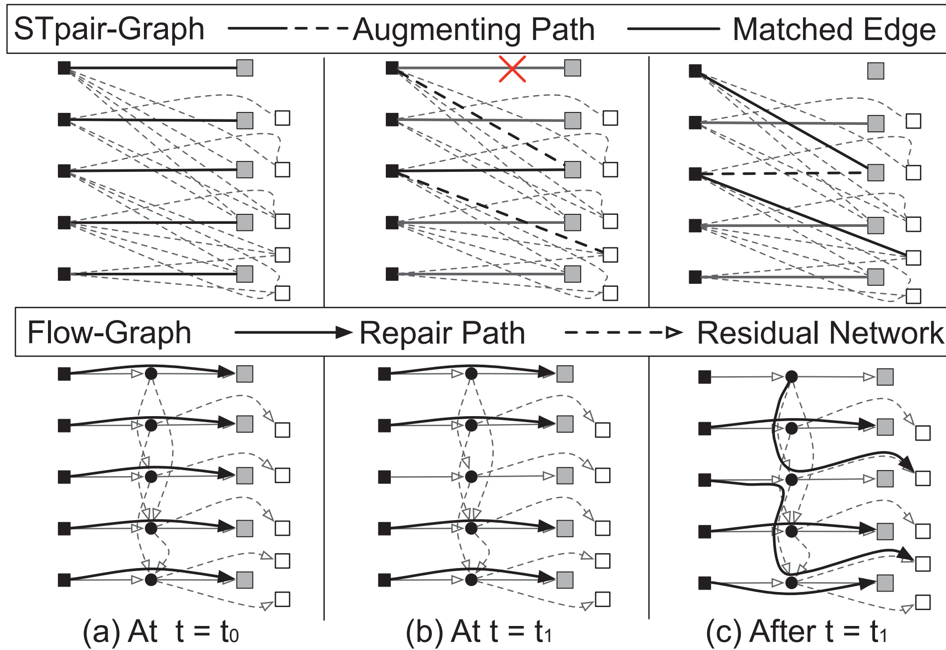


Figure 6.4: Illustration of the Repair Algorithm.

whose matching number is equal to n (i.e., every signal is paired with a dedicated TSV) and all of the signal-TSV pairs are both routable and confirmed to have no timing violations with online testing. With continuous circuit aging, one can imagine that the number of edges in *STpair-graph* keeps decreasing, and the 3D IC is irreparable when the matching number of a *STpair-graph* is less than n.

We use Fig. 6.4 to illustrate one possible repair algorithm. Fig. 6.4(a) presents the matching used in the 3D IC at $t = 0$. Suppose online testing is performed at $t = t_1$, and we found one signal-TSV pair fails its test. We remove this edge from *STpair-graph* and thus the current matching is not maximum any more. In order to find another maximum matching, we resort to Berge's lemma [120], by iteratively finding the shortest *augmenting path*[1] from the unmatched signal to any free TSV. Such a method preserves the signal-TSV pairs in the earlier matching when-

[1]An augmenting path of a matching is defined as a path that starts and ends on free (unmatched) vertices, and alternates between edges in & not in the matching.

ever possible and hence is more likely to be valid when compared to a solution based on a random maximum matching of the updated *STpair-graph*. In addition, routability checking is integrated into the above procedure for efficiency. That is, whenever we add an augmented path, we update the corresponding flow graph and check whether it can be routed in the residual network of the flow graph. To update the flow graph, we cancel the edges in the flow graph possessed by those signal-TSV pairs that are removed in the matching and move them to the residual graph (i.e., the sub-graph of the original one, composing edges with residual capacity) (Fig. 6.4(b)). Then, we can verify routability by finding edge-disjoint paths in the residual network for those signal-TSV pairs that are added into the new matching (Fig. 6.4(c)). If the matching solution is not routable, we find another augmenting path and iterate the above procedure. Otherwise, we invoke online testing to check whether this solution leads to any timing violation. If not, we have obtained a valid repair solution. Otherwise, we update *STpair-graph* by removing those edges whose corresponding signal-TSV pairs fail path delay tests, and repeat the above procedure on the updated *STpair-graph*.

While simple and effective, the above algorithm may invoke online testing many times due to the enumeration of matchings. Let us use M_i to denote the i_{th} maximum matching of the *STpair-graph* (containing the set of all signal-TSV pairs). The above repair algorithm iteratively finds a new maximum matching and performs online testing for it, until a matching (say, M_v) is shown to be valid. Hence, we need to perform v times of online testing. Generally speaking, however, there is usually a significant overlap of the signal-TSV pairs between M_i and M_{i+1} because we tend to preserve many existing valid signal-TSV pairs from the previous solution in each iteration. These preserved pairs are known to be fault-free with previous testing results, which do not need to be tested again.

Motivated by the above discussion, we propose a more efficient algorithm.

Instead of checking one possible matching a time with online testing, we attempt to test "multiple matchings" simultaneously, whenever possible. For example, after testing M_0, for a new maximum matching M_1, we only need to perform online testing for those signal-TSV pairs in $M_1 \backslash M_0$, because the other signal-TSV pairs in $M_1 \bigcap M_0$ have been shown to be valid with previous testing results of M_0. Without loss of generality, let us consider another maximum matching M_2 (if any), there must be some signal-TSV pairs in $M_2 \backslash M_1$ (otherwise M_2 is not a new matching). If some of these signal-TSV pairs have not been tested (i.e., they do not belong to M_0) and they are routable together with those signal-TSV pairs in $M_1 \backslash M_0$, they can be tested simultaneously in one iteration. Such a method reduces the number of online testing because, if a signal-TSV pair in $M_2 \backslash M_1$ is shown to be invalid, not only we do not need to test M_2 any more, but also the corresponding edge is removed from the *STpair-graph* and reduces the possibility to find other invalid matchings.

6.5.3 Impact of TSV Redundancy Sharing

With TSV redundancy sharing between neighboring TSV grids, there might be conflict between the repair requirements between them. We use the example shown in Fig. 6.5 to explain how we resolve this issue. In this example, TSV Grid *A* and TSV Grid *B* are sharing spares TSVs in Fig. 6.5(a)). In this case, for both grids, their *STpair-graphs* contain the shared TSVs. Hence, the two STpair-graphs are connected as shown in Fig. 6.5(b). We still obtain maximum matchings for each grid by finding augmenting paths. When the two grids try to use the same spare TSV with their augmenting paths, a conflict arises (see red line in Fig. 6.5(b)). We then arbitrate which grid owns this spare TSV according to the fault maps of the two grids. In this example, Grid *A* has more faults than Grid *B* and hence this spare TSV will be assigned to Grid *A*. We remove this node from the STpair-graph of

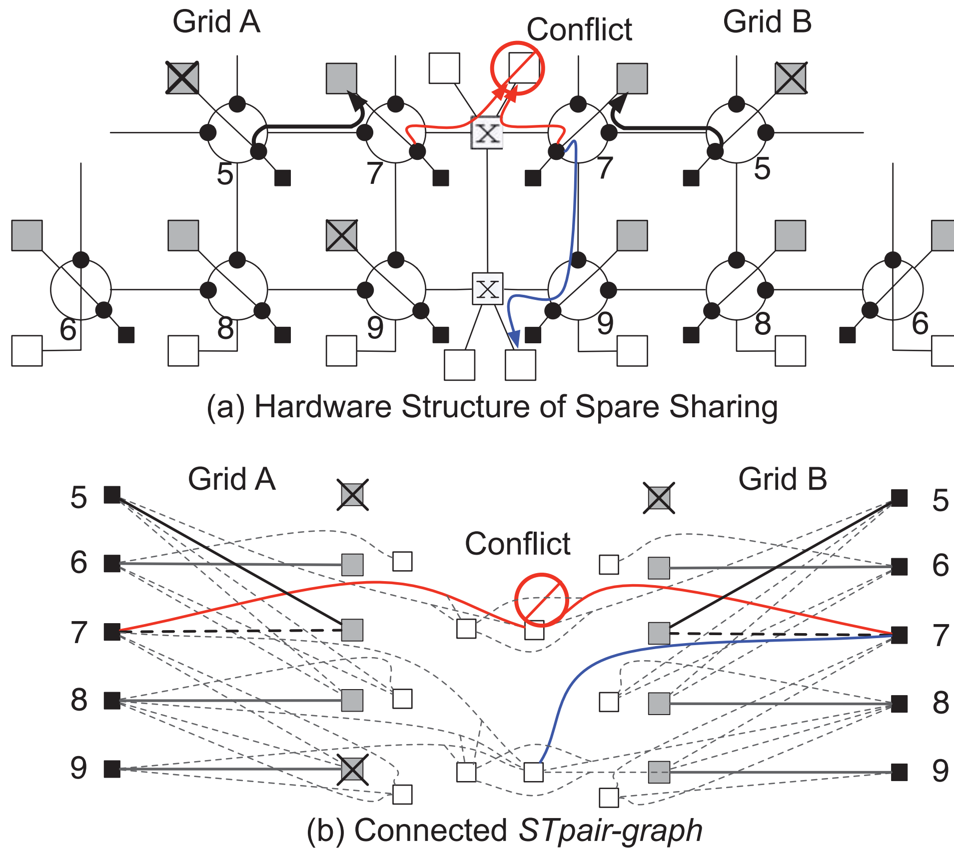


Figure 6.5: The Impact of TSV Redundancy Sharing on Repair Algorithm.

Grid *B* and it will look for a different matching for in-field repair.

6.6 Experimental Results

6.6.1 Experimental Setup

To evaluate the effectiveness and efficiency of the proposed solution, we perform simulation studies and report results on MTTF and test times.

We use the maximum-flow based algorithm presented in chapter 5 as the baseline solution for comparison. As chapter 5 mainly deals with manufacturing defects and the original algorithm uses a static fault map as input, we make the fol-

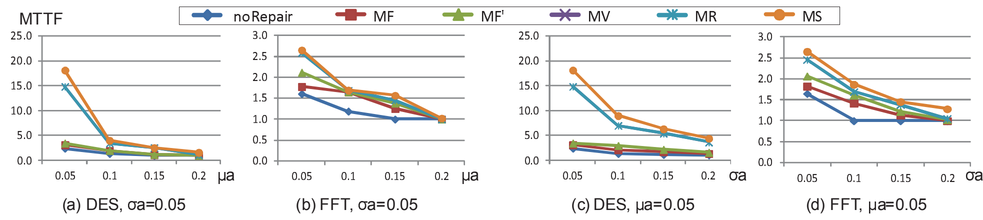


Figure 6.6: MTTF Results in 4×4 TSV Grid with Varied Aging Coefficients and Fixed Potential Crack or Void Defect Distribution (0.1 kΩm, 0.1 kΩm).

lowing changes to generate two types of baseline in-field repair algorithms. The first type simply updates the fault map by marking the corresponding TSV to be "faulty" whenever online testing shows an invalid signal-TSV repair and utilizes the original algorithm to find a new repair solution (if possible), denoted as MF. For the second type, when a signal-TSV pair is shown to be invalid with online testing, it attempts to find another repair path for the faulty TSV instead of marking the TSV as "faulty", denoted as $MF^{'}$. The proposed algorithm based on maximum matching with routability verification is denoted as MV, while the proposed repair algorithm with test time reduction is named as MR. The above results are obtained based on the TSV redundancy architecture in chapter 5. We further present the results based on the proposed TSV redundancy architecture with spare sharing capability, denoted as MS. We compare the MTTF of the above solutions, and a particular 3D IC is deemed to fail when no repair solution can be found for a path delay fault due to aging effects.

The circuits used in our experiments are the performance-optimized data encryption standard (DES) circuit and the fast-Fourier transform (FFT) circuit from the IWLS 2005 OpenCore benchmarks. The DES circuit contains 26,000 gates and 2,000 flops, while the FFT circuit contains 229,000 gates and 20,000 flops. The DES circuit was partitioned into two-, three-, and four-die stacks using the Nangate open cell library and a placement engine for timing optimization. Given the operational frequency of the benchmark 3D ICs, we extract the timing slacks

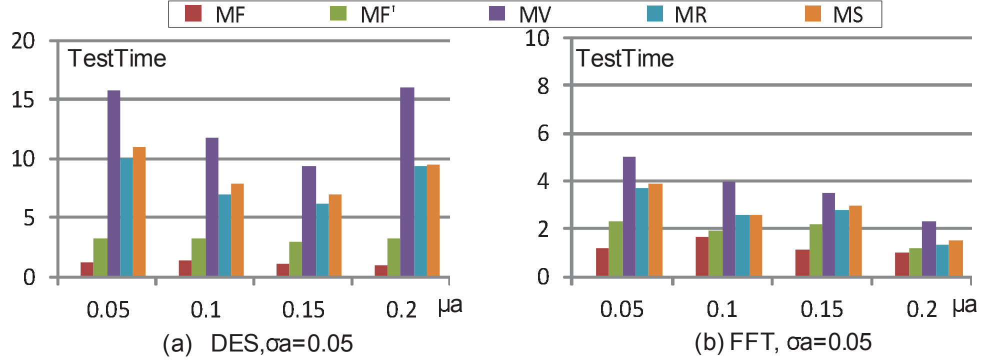


Figure 6.7: Test Time Results in 4×4 TSV Grid with Varied Aging Coefficients and Fixed Potential Crack or Void Defect Distribution (0.1 kΩm, 0.1 kΩm).

for paths with TSVs. Due to the lack of reliability models for stress-induced TSV interfacial cracks in the public literature, we form our model based on an EM reliability model for TSVs and vary its parameter to reflect the impact of TSV interfacial cracks [50, 114]. We also consider initial TSV failures due to manufacturing defects. Aging effects are characterized by additional latent delay in TSVs, reflected as resistance increase in terms of time t, calculated as

$$R(t) - R_0 = A\ln(\frac{t}{t_0}) \tag{6.1}$$

where A is the slope of TSV degradation on a logarithmic scale, and t_0 is the time when the void becomes larger than their TSV section. Note that A and t_0 are affected by multiple parameters, such as the initial resistance R_0 of TSV, TSV barrier resistivity, TSV dimensions, and possibility of voids generated in TSVs. R_0 varies for different TSVs due to process variation and it is assumed to follow a Gaussian distribution. The parameter A indicates the aging rate, which is related to the workloads applied to the 3D IC which in turn determines the temperature and switching activities for TSVs. The dynamic changing of A due to workloads is aggregated in this chapter and we use Gaussian distribution to obtain A.

6.6.2 Results and Analysis

Fig. 6.6(a)-(d) presents the normalized MTTF values, compared to the worst case without any redundancy for in-field repair. We have four configurations for aging coefficients with their mean values (μa) and variances (σa) varying between 0.05 kΩ/log(s) to 0.2 kΩ/log(s). This setting mimics the circuits under different stress. The distribution of the initial resistances R_0 of TSVs that represents the potential Crack or Void Defect Distribution are fixed with a mean value (μr) of 0.1 kΩ and variance (σr) of 0.1 kΩ.

First of all, it can be observed that the two proposed repair algorithms with the TSV redundancy architecture in chapter 5 lead to much higher MTTF values when compared to the two baseline solutions. For example, for DES design, MTTFs are 14.8 for both MV and MR with aging coefficient of (0.05, 0.05), compared to 3.1 using MF and 3.5 using MF'. This is because we are able to search a much larger solution space by exploring all possible signal-TSV pairs while previous methods only target on repair path identification for faulty TSVs. The MTTF value of MF' is slightly better than that of MF because the latter solution regards the TSV from an invalid signal-TSV pair as "faulty", rendering an even smaller solution space. It should be noted that MV and MR have the same MTTF as the solution space are the same for these two algorithms. By adding spare sharing capability with the proposed architecture, the MTTF is increased to 18.2 under the same aging rate.

Secondly, we observe significant MTTF reduction as aging coefficients increase (see Fig. 6.6(a)-(d)), due to the higher TSV failure probability with increasing aging rates. The differences are even larger for the proposed two repair algorithms (MV and MS), wherein the impetus of downtrends is reduced as the aging coefficient increases. This is expected as the solution space shrinks quickly as aging effect becomes more severe, rendering less repair efficiency for all repair algorithms. This indicates that, even with a better TSV redundancy architecture

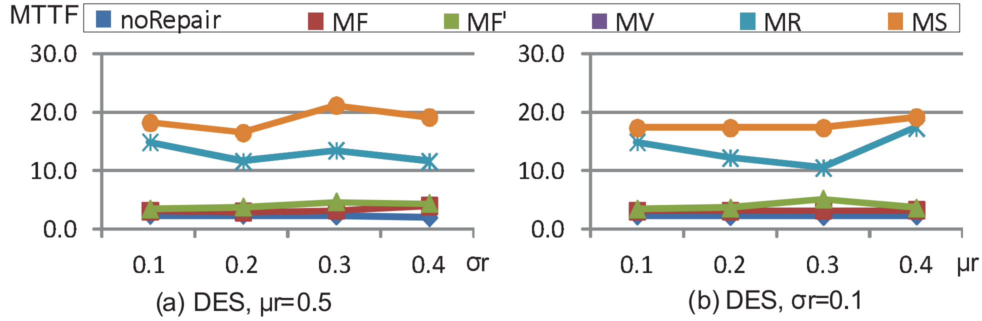


Figure 6.8: MTTF Results for DES in 4×4 TSV Grid with Varied Potential Crack/void Defect Distribution and Fixed Aging Coefficients (0.05 kΩ/log(s),0.05 kΩ/log(s)).

(with spare TSV redundancy sharing), we cannot achieve high MTTF values when the circuit is under severe aging effects.

Thirdly, we compare the results of DES design in Fig. 6.6(a)(c) and FFT design in Fig. 6.6(b)(d). While we can see similar trends for the results of FFT design, but the MTTF differences between the five algorithms are not as significant as that of DES design. This is mainly because the timing slacks of paths with TSVs in FFT design is much tighter, thus leading to less MTTF values.

Fig. 6.7(a)-(b) describe the corresponding test time (in terms of the number of performed online testing) of the circuit with the five repair methods under various aging coefficients, corresponding to Fig. 6.6(a)-(b). The proposed algorithms requires more test time compared to the baseline algorithms due to the fact that more on-line tests are conducted to achieve more successful repair. While the MTTF values for *MV* and *MR* are the same, the test times of *MR* are much smaller. This is because, we try to perform online testing for multiple possible matchings at the same time. With such test time reduction scheme, the test times of *MS* only increase slightly although it has a larger solution space to explore with spare redundancy sharing.

Fig. 6.8 shows the MTTF values of different repair methods when we vary

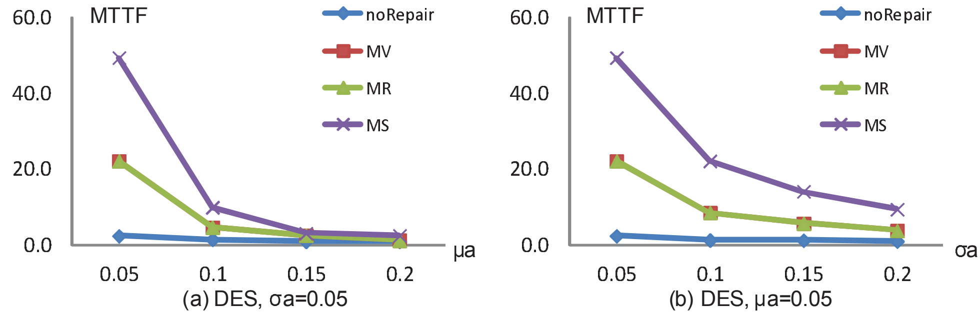


Figure 6.9: Experimental Results in 8 × 8 TSV Grid Size Repair Architecture with Varied Aging Coefficients and Fixed Potential Crack/Void Defect Distribution (0.1 kΩ, 0.1 kΩ).

the initial resistances of TSVs, which demonstrate the impact of undetectable cracks/voids during fabrication on the service life of 3D ICs. Due to space limit, we only report the results of DES circuit. We fix the aging coefficient as (0.05, 0.05), and vary the TSV initial resistances R_0 with its mean values from 0.1 kΩ to 0.4 kΩ and a fixed variance value in Fig. 6.8(a). While in Fig. 6.8(b), we also have four configurations for R_0 with the same variance value of 0.1 kΩ but different mean values ranging from 0.1 kΩ to 0.4 kΩ. From this figure, we can observe that the TSV initial resistance has minor impact on the MTTF values, when compared to the aging coefficients changes shown in Fig. 6.6. This is also expected because TSV voids/cracks that have passed burn-in test, have to grow large enough to affect circuit timing, which is determined more by the aging rates instead of their initial values.

Fig. 6.9 shows the MTTF values of the proposed repair methods with a 8×8 grid size for the repair architecture. The trends are similar to that in Fig. 6.6, however, the MTTF values of *MR* and *MS* are much larger than that in Fig. 6.6. The main reason is that, in this experiment the signal rerouting delay is not considered and hence we have a much larger repair solution space to explore with a larger TSV bundle.

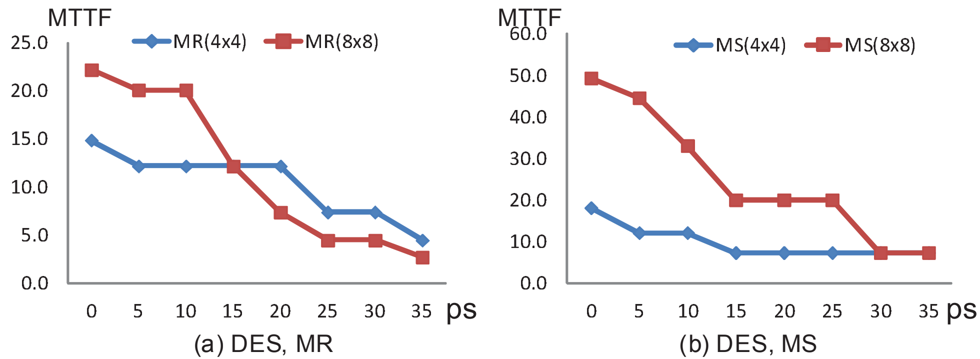


Figure 6.10: Experimental Results with Varied Rerouting Delay between Two Adjacent Routers (*ps*) and Fixed Aging Coefficients and Potential Crack/Void Defect Distribution.

It should be noted that, the extra signal rerouting delay is already taken into consideration in the proposed repair architecture even though the previous simulation studies ignore it. As long as the rerouting delay of a signal-TSV pair exceeds the timing slack of the path containing this signal-TSV pair, the on-line test can detect a timing error and discard this pair from the solution space. Fig. 6.10 investigates the effect of this rerouting delays and shows the MTTF of the proposed repair methods for two different grid size in the repair architecture. For both methods, the architecture with 8×8 grid size performs better when the rerouting delay is small, because it has larger solution space for repair. As rerouting delay increases, the MTTF curves of the two architectures intersect at a point when it has become a bottleneck for signals to be able to reach many TSVs for repair. Beyond this point, the architecture with 4×4 grid size results in more successful repair with higher redundancy ratio. Compared to *MR*, the intersection point of the two architecture occurs later for *MS* because the shared redundant TSVs give each TSV grid more solution space to explore.

6.7 Conclusion

TSV-based 3D ICs have emerged as one of the most promising solutions to overcome interconnect bottleneck in CMOS scaling. The disruptive manufacturing process of TSVs, however, introduce new failure mechanisms such as stress-induced interfacial cracks and EM-induced voids. Such reliability threats reduce the service life of 3D ICs. In this chapter, we have described a novel in-field repair solution that is able to effectively and efficiently tolerate latent TSV defects through the judicious use of spares. Experimental results on 3D benchmark circuits show that the proposed solution is able to significantly increase MTTF when compared to existing TSV repair techniques.

□ **End of chapter.**

Chapter 7

Conclusion and Future Work

The rapid adoption of 3D integration technology seems to be essential and unavoidable [121]. Without efficient and effective solutions to ensure a high yield and long service life for 3D-stacked IC, however, it cannot become a mature industry product. In this thesis, we propose effective and efficient techniques to enhance the yield and reliability of 3D-stacked ICs. We first provide mathematical yield model for 3D-stacked ICs considering various factors, such as stacking process, pre-bond test and TSV fabrication. For better understanding the faulty behavior of defective TSVs, we then provide a fault model for TSVs in future 3D DRAMs. To improve the stack yield, we propose novel test architecture optimization techniques considering the pre-bond test and two most significant test challenges, with which the bad dies are prevented from being stacked. With the enhanced pre-bond testability, we continue to strength the stacking yield of 3D-DRAMs with effective memory repair framework. In order to solve the TSV-induced assembly yield loss, we propose a TSV repair framework, including a novel TSV-grid based redundancy architecture and corresponding repair algorithms. To alleviate the reliability problems induced by TSV latent faults, we propose an efficient in-field repair solution, including the hardware architecture together with an in-field TSV repair algorithm.

By conducting the above research works, we are able to dramatically improve

the yield and reliability of various 3D products (e.g., SoCs, memories and ASICs). As a small percentage of yield loss transfers to millions of dollars of cost increment, while the callback of unreliable products hurts both profit and reputation, the success of this dissertation is of significant value to the main stream adoption of 3D ICs and semiconductor industry.

There are several important topics yet to be explored for future work. Firstly, the TSV induced latent defects still remain mysterious and future fault modeling is required. Secondly, the TSV induced thermal-mechanical stress causes systematic mobility variations in neighboring logic gates that lead to delay degradation. Therefore, efficient design methodologies are essential to tackle this problem in the early stage of the design flow. Last but not the least, this thesis has shown various challenging issues in 3D integration, however, this new integration paradigm provides us new opportunities to facilitate the on-line testing and fault tolerance, e.g., by stacking a FPGA die and a 2D-SoC die with the interposer, the faulty core in SoC can be replaced with partial reconfigured logics in FPGA.

□ **End of chapter.**

Bibliography

[1] Uksong Kang, Hoe-Ju Chung, Seongmoo Heo, Duk-Ha Park, Hoon Lee, Jin Ho Kim, Soon-Hong Ahn, Soo-Ho Cha, Jaesung Ahn, DukMin Kwon, Jae-Wook Lee, Han-Sung Joo, Woo-Seop Kim, Dong Hyeon Jang, Nam Seog Kim, Jung-Hwan Choi, Tae-Gyeong Chung, Jei-Hwan Yoo, Joo Sun Choi, Changhyun Kim, and Young-Hyun Jun. 8 gb 3-d ddr3 dram using through-silicon-via technology. *IEEE Journal of Solid-State Circuits*, 45(1):111 –119, jan. 2010.

[2] A.C. Hsieh, T.T. Hwang, M.T. Chang, M.H. Tsai, C.M. Tseng, and H.-C. Li. Tsv redundancy: Architecture and design issues in 3d ic. In *Proceedings Design, Automation, and Test in Europe Conference Exhibition*, pages 166 –171, march 2010.

[3] I. Loi, S. Mitra, T.H. Lee, S. Fujita, and L. Benini. A low-overhead fault tolerance scheme for tsv-based 3d network on chip links. In *Proceedings International Conference on Computer-Aided Design*, pages 598 –602, nov. 2008.

[4] Y. Zhao, S. Khursheed, and B. Al-Hashimi. Cost-Effective TSV Grouping for Yield Improvement of 3D-ICs. *Asian Test Symposium*, 2011.

[5] Qiang Xu, Li Jiang, Huiyun Li, and Bill Eklow. Yield enhancement for 3d-stacked ics: Recent advances and challenges. In *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*, pages 731–737. IEEE, 2012.

[6] K. Banerjee, S.J. Souri, P. Kapur, and K.C. Saraswat. 3-d ics: a novel chip design for improving deep-submicrometer interconnect performance and systems-on-chip integration. *Proceedings IEEE*, 89(5):602 –633, may 2001.

[7] W.R. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A.M. Sule, M. Steer, and P.D. Franzon. Demystifying 3d ics: the pros and cons of going vertical. *IEEE Design Test of Computers*, 22(6):498 – 510, nov.-dec. 2005.

[8] A. Rahman and R. Reif. System-level performance evaluation of three-dimensional integrated circuits. *IEEE Transactions on Very Large Scale Integration VLSI Systems*, 8(6):671 –678, dec. 2000.

[9] R. Weerasekera, Li-Rong Zheng, D. Pamunuwa, and H. Tenhunen. Extending systems-on-chip to the third dimension: performance, cost and technological tradeoffs. In *Proceedings International Conference on Computer-Aided Design*, pages 212 –219, nov. 2007.

[10] CC Liu, I. Ganusov, M. Burtscher, and S. Tiwari. Bridging the processor-memory performance gap with 3D IC technology. *IEEE Design & Test of Computers*, 22(6):556–564, 2005.

[11] Philip Jacob, Aamir Zia, Okan Erdogan, Paul M Belemjian, Jin-Woo Kim, Michael Chu, Russell P Kraft, John F McDonald, and Kerry Bernstein. Mitigating memory wall effects in high-clock-rate and multicore cmos 3-d processor memory stacks. *Proceedings of the IEEE*, 97(1):108–122, 2009.

[12] Eric Mounier and Yole Développement. Market trends for 3d stacking. *The Global Assembly Journal for SMT and Advanced Packaging Professionals*, 7(7):56, 2007.

[13] Yuan Xie, Gabriel H Loh, Bryan Black, and Kerry Bernstein. Design space exploration for 3d architectures. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 2(2):65–103, 2006.

[14] Takafumi Fukushima, Yusuke Yamada, Hirokazu Kikuchi, and Mitsumasa Koyanagi. New three-dimensional integration technology using chip-to-wafer bonding to achieve ultimate super-chip integration. *Japanese journal of applied physics*, 45:3030, 2006.

[15] Gabriel H. Loh, Yuan Xie, and Bryan Black. Processor design in 3d die-stacking technologies. *IEEE Micro*, 27(3):31 –48, may-june 2007.

[16] G. Smith, L. Smith, S. Hosali, and S. Arkalgud. Yield considerations in the choice of 3d technology. In *International Symposium on Semiconductor Manufacturing*, pages 1 –3, oct. 2007.

[17] Jan Van Olmen, Abdelkarim Mercha, Guruprasad Katti, Cedric Huyghebaert, Joke Van Aelst, Emma Seppala, Zhao Chao, Silvia Armini, Jan Vaes, R Cotrin Teixeira, et al. 3d stacked ic demonstration using a through silicon via first approach. In *Electron Devices Meeting, 2008. IEDM 2008. IEEE International*, pages 1–4. IEEE, 2008.

[18] Bart Swinnen, W Ruythooren, P De Moor, L Bogaerts, L Carbonell, K De Munck, B Eyckens, S Stoukatch, D Sabuncuoglu Tezcan, Z Tokei, et al. 3d integration by cu-cu thermo-compression bonding of extremely thinned bulk-si die containing 10 μm pitch through-si vias. In *Electron Devices Meeting, 2006. IEDM'06. International*, pages 1–4. IEEE, 2006.

[19] B. Black, D.W. Nelson, C. Webb, and N. Samra. 3d processing technology and its impact on ia32 microprocessors. In *Proceedings IEEE International Conference on VLSI Design*, pages 316 – 318, oct. 2004.

[20] M. Kawano, S. Uchiyama, Y. Egawa, N. Takahashi, Y. Kurita, K. Soejima, M. Komuro, S. Matsui, K. Shibata, J. Yamada, M. Ishino, H. Ikeda, Y. Saeki, O. Kato, H. Kikuchi, and T. Mitsuhashi. A 3d packaging technology for 4 gbit stacked dram with 3 gbps data transfer. In *IEEE International Electron Devices Meeting*, pages 1 –4, dec. 2006.

[21] Dong S Suk and Sudhakar M. Reddy. A march test for functional faults in semiconductor random access memories. *Computers, IEEE Transactions on*, 100(12):982–985, 1981.

[22] K. Chakraborty and P. Mazumder. *Fault-tolerance and reliability techniques for high-density random-access memories*. Prentice Hall PTR, 2002.

[23] G.H. Loh. 3d-stacked memory architectures for multi-core processors. In *Proceedings International Symposium on Computer Architecture*, pages 453 –464, june 2008.

[24] H.-H.S. Lee and K. Chakrabarty. Test challenges for 3d integrated circuits. *IEEE Design Test of Computers*, 26(5):26 –35, sept.-oct. 2009.

[25] Dean L Lewis and H-HS Lee. A scanisland based design enabling prebond testability in die-stacked microprocessors. In *Test Conference, 2007. ITC 2007. IEEE International*, pages 1–8. IEEE, 2007.

[26] William R Mann, Frederick L Taber, Philip W Seitzer, and Jerry J Broz. The leading edge of production wafer probe test technology. In *Test Conference, 2004. Proceedings. ITC 2004. International*, pages 1168–1195. IEEE, 2004.

[27] E.J. Marinissen and Y. Zorian. Testing 3d chips containing through-silicon vias. In *Proceedings IEEE International Test Conference*, pages 1 –11, nov. 2009.

[28] Dean L Lewis and H-HS Lee. Testing circuit-partitioned 3d ic designs. In *VLSI, 2009. ISVLSI'09. IEEE Computer Society Annual Symposium on*, pages 139–144. IEEE, 2009.

[29] J. Li and D. Xiang. Dft optimization for pre-bond testing of 3d-sics containing tsvs. In *Computer Design (ICCD), 2010 IEEE International Conference on*, pages 474–479. IEEE, 2010.

[30] X.X Wu, Y.B. Chen, K Chakrabarty, and Y Xie. Test-Access Mechanism Optimization for Core-Based Three-Dimensional SOCs. In *International Conference on Computer Design*, pages 212–218, 2008.

[31] Erik Jan Marinissen, Jouke Verbree, and Mario Konijnenburg. A structured and scalable test access architecture for tsv-based 3d stacked ics. In *VLSI Test Symposium (VTS), 2010 28th*, pages 269–274. IEEE, 2010.

[32] Erik Jan Marinissen, Chun-Chuan Chi, Jouke Verbree, and Mario Konijnenburg. 3d dft architecture for pre-bond and post-bond testing. In *3D Systems Integration Conference (3DIC), 2010 IEEE International*, pages 1–8. IEEE, 2010.

[33] Test access architecture for three-dimensional stacked integrated circuits. *proposed IEEE standard P1838*.

[34] Po-Yuan Chen, Cheng-Wen Wu, and Ding-Ming Kwai. On-chip tsv testing for 3d ic before bonding using sense amplification. In *Asian Test Symposium, 2009. ATS'09.*, pages 450–455. IEEE, 2009.

[35] Po-Yuan Chen, Cheng-Wen Wu, and Ding-Ming Kwai. On-chip testing of blind and open-sleeve tsvs for 3d ic before bonding. In *VLSI Test Symposium (VTS), 2010 28th*, pages 263–268. IEEE, 2010.

[36] Minki Cho, Chang Liu, Dae Hyun Kim, Sung Kyu Lim, and Saibal Mukhopadhyay. Design method and test structure to characterize and repair tsv defect induced signal degradation in 3d system. In *Proceedings of the International Conference on Computer-Aided Design*, pages 694–697. IEEE Press, 2010.

[37] Yi Lou, Zhuo Yan, Fan Zhang, and Paul D Franzon. Comparing through-silicon-via (tsv) void/pinhole defect self-test methods. *Journal of Electronic Testing*, 28(1):27–38, 2012.

[38] Erik Jan Marinissen. Challenges in testing tsv-based 3d stacked ics: test flows, test contents, and test access. In *Circuits and Systems (APCCAS), 2010 IEEE Asia Pacific Conference on*, pages 544–547. IEEE, 2010.

[39] N. Miyakawa. A 3d prototyping chip based on a wafer-level stacking technology. In *Proceedings IEEE Asia South Pacific Design Automation Conference*, pages 416–420, jan. 2009.

[40] J. U. Knickerbocker, P. S. Andry, B. Dang, R. R. Horton, M. J. Interrante, C. S. Patel, R. J. Polastre, K. Sakuma, R. Sirdeshmukh, E. J. Sprogis, S. M. Sri-Jayantha, A. M. Stephens, A. W. Topol, C. K. Tsang, B. C. Webb, and S. L. Wright. Three-dimensional silicon integration. *IBM Journal of Research and Development*, 52(6):553 –569, nov. 2008.

[41] Akihiro Ikeda et al. Design and measurements of test element group wafer thinned to 10um for 3d system in package. In *Proc. IEEE International Conference on Microelectronic Test Structures*, pages 161 –164, march. 2004.

[42] Dan Perry et al. Impact of thinning and packaging on a deep sub-micron cmos product. In *Digest of DATE Workshop on 3D Integration*, pages 282,, april. 2009.

[43] H Chen, JY Shih, SW Li, HC Lin, MJ Wang, and CN Peng. Electrical tests for three-dimensional ics (3dics) with tsvs. In *Informal Proc. International Test Conference 3D-Test Workshop*, 2010.

[44] S. Reda, G. Smith, and L. Smith. Maximizing the functional yield of wafer-to-wafer 3-d integration. *IEEE Transactions on Very Large Scale Integration VLSI Systems*, 17(9):1357 –1362, sept. 2009.

[45] Cesare Ferri, Sherief Reda, and R Bahar. Parametric yield management for 3d ics: Models and strategies for improvement. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 4(4):19, 2008.

[46] L. Jiang, Q. Xu, and B. Eklow. On Effective TSV Repair for 3D-Stacked ICs. In *Proc. IEEE/ACM Design, Automation, and Test in Europe*, pages 6–11, 2012.

[47] Ang-Chih Hsieh and TingTing Hwang. Tsv redundancy: Architecture and design issues in 3-d ic. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 20(4):711–722, 2012.

[48] Suk-Kyu Ryu, Kuan-Hsun Lu, Xuefeng Zhang, Jang-Hi Im, Paul S Ho, and Rui Huang. Impact of near-surface thermal stresses on interfacial reliability of through-silicon vias for 3-d interconnects. *Device and Materials Reliability, IEEE Transactions on*, 11(1):35–43, 2011.

[49] YC Tan, Cher Ming Tan, XW Zhang, Tai Chong Chai, and DQ Yu. Electromigration performance of through silicon via (tsv)–a modeling approach. *Microelectronics Reliability*, 50(9):1336–1340, 2010.

[50] T Frank, C Chappaz, P Leduc, L Arnaud, F Lorut, S Moreau, A Thuaire, R El Farhane, and L Anghel. Resistance increase due to electromigration induced depletion under tsv. In *Reliability Physics Symposium (IRPS), 2011 IEEE International*, pages 3F–4. IEEE, 2011.

[51] I. Koren and Z. Koren. Defect tolerance in vlsi circuits: techniques and yield analysis. *Proceedings of the IEEE*, 86(9):1819–1838, 1998.

[52] T. W. Williams and N.C. Brown. Defect level as a function of fault coverage. *IEEE Transactions on Computers*, 100(12):987–988, 1981.

[53] Y. Chen, D. Niu, Y. Xie, and K. Chakrabarty. Cost-effective integration of three-dimensional (3d) ics emphasizing testing cost analysis. In *Proceedings International Conference on Computer-Aided Design*, pages 471 –476, nov. 2010.

[54] J. Verbree, E.J. Marinissen, P. Roussel, and D. Velenis. On the cost-effectiveness of matching repositories of pre-tested wafers for wafer-to-wafer 3d chip stacking. In *IEEE European Test Symposium*, pages 36 –41, may 2010.

[55] Yangdong Deng and W.P. Maly. 2.5-dimensional vlsi system integration. *IEEE Transactions on Very Large Scale Integration VLSI Systems*, 13(6):668 –677, june 2005.

[56] Ioannis Savidis and Eby G Friedman. Closed-form expressions of 3-d via resistance, inductance, and capacitance. *Electron Devices, IEEE Transactions on*, 56(9):1873–1881, 2009.

[57] Mark Francis Hilbert. High-density memory utilizing multiplexers to reduce bit line pitch constraints, April 23 2002. US Patent 6,377,504.

[58] Hong Sangki. 3D Super-Via for Memory Applications. In *Micro-Systems Packaging Initiative (MSPI) Packaging Workshop 2007.www. tezzaron. com/ about/ papers/ Sangki_2007.pd*, 2007.

[59] Rob Dekker, Frans Beenker, and Loek Thijssen. Fault modeling and test algorithm development for static random access memories. In *International Test Conference*, pages 343–352. IEEE, 1988.

[60] I. Savidis and E.G. Friedman. Electrical modeling and characterization of 3-D vias. In *IEEE International Symposium on Circuits and Systems*, pages 784–787, 2008.

[61] Bruce Jacob, Spencer Ng, and David Wang. *Memory systems: cache, DRAM, disk*. Morgan Kaufmann, 2010.

[62] Zaid Al-Ars and A van de Goor. Static and dynamic behavior of memory cell array opens and shorts in embedded drams. In *Design, automation and test in Europe*, pages 496–503. IEEE Press, 2001.

[63] Z. Al-Ars and A.J. van de Goor. Approximating infinite dynamic behavior for DRAM cell defects. In *IEEE VLSI Test Symposium, 2002*, pages 401–406, 2002.

[64] Y. Sato, L. Yamazaki, H. Yamanaka, T. Ikeda, and M. Takakura. A persistent diagnostic technique for unstable defects. In *International Test Conference.*, pages 242–249, 2002.

[65] M. Renovell and GN Cambon. Electrical analysis and modeling of floating-gate fault. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(11):1450–1458, 1992.

[66] H. Konuk. Voltage-and current-based fault simulation for interconnect open defects. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(12):1768–1779, 1999.

[67] S.-C. Wong, G.-Y. Lee, and D.-J. Ma. Modeling of interconnect capacitance, delay, and crosstalk in vlsi. *IEEE Transactions on Semiconductor Manufacturing*, 13(1):108–111, 2000.

[68] Erik Jan Marinissen et al. On IEEE P1500's Standard for Embedded Core Test. In Krishnendu Chakrabarty, editor, *SOC (System-on-a-Chip) Testing for Plug and Play Test Automation*, volume 21 of *Frontiers in Electronics Testing*, pages 1–19. Kluwer Academic Publishers, September 2002.

[69] Erik Jan Marinissen, Sandeep Kumar Goel, and Maurice Lousberg. Wrapper Design for Embedded Core Test. In *Proceedings IEEE International Test Conference (ITC)*, pages 911–920, Atlantic City, NJ, October 2000.

[70] Prab Varma and Sandeep Bhatia. A Structured Test Re-Use Methodology for Core-Based System Chips. In *Proceedings IEEE International Test Conference (ITC)*, pages 294–302, Washington, DC, October 1998.

[71] Qiang Xu and Nicola Nicolici. Resource-Constrained System-on-a-Chip Test: A Survey. 152(1):67–81, January 2005.

[72] T.M. Mak, P. Garrou, C. Bower, and P. Ramm. *Testing of 3D Circuites*. In Handbook of 3D Integration:Technology and Applications using 3D Integrated Circuits.Wiley-CVH, 2008.

[73] X. Wu, Y. Chen, K. Chakrabarty, and Y. Xie. Test-access mechanism optimization for core-based three-dimensional socs. In *Proceedings International Conference on Computer Design*, to appear, 2008.

[74] B. Noia, S.K. Goel, K. Chakrabarty, E.J. Marinissen, and J. Verbree. Test-Architecture Optimization for TSV-Based 3D Stacked ICs. In *IEEE European Test Symposium*, pages 24–29, 2010.

[75] S.K. Goel and E.J. Marinissen. Layout-driven soc test architecture design for test time and wire length minimization. In *Proceedings Design, Automation, and Test in Europe Conference Exhibition*, pages 738 – 743, 2003.

[76] S. Bahukudumbi and K. Chakrabarty. Wafer-level modular testing of core-based socs. *IEEE Transactions on Very Large Scale Integration VLSI Systems*, 15(10):1144 –1154, oct. 2007.

[77] Vikram Iyengar, Sandeep Kumar Goel, Krishnendu Chakrabarty, and Erik Jan Marinissen. Test Resource Optimization for Multi-Site Testing of SOCs Under ATE Memory Depth Constraints. In *Proceedings IEEE International Test Conference (ITC)*, pages 1159–1168, Baltimore, MD, October 2002.

[78] Vikram Iyengar, Krishnendu Chakrabarty, and Erik Jan Marinissen. Co-Optimization of Test Wrapper and Test Access Architecture for Embedded Cores. *Journal of Electronic Testing: Theory and Applications*, 18(2):213–230, April 2002.

[79] Sandeep Kumar Goel and Erik Jan Marinissen. Effective and Efficient Test Architecture Design for SOCs. In *Proceedings IEEE International Test Conference (ITC)*, pages 529–538, Baltimore, MD, October 2002.

[80] Y. et al. Huang. Optimal core wrapper width selecti on and soc test scheduling based on 3-d bin packing algorithm. In *Proceedings IEEE International Test Conference (ITC)*, pages 74–82, 2002.

[81] Vikram Iyengar, Krishnendu Chakrabarty, and Erik Jan Marinissen. Integrated Wrapper/TAM Co-Optimization, Constraint-Driven Test Scheduling, and Tester Data Volume Reduction for SOCs. In *Proceedings ACM/IEEE Design Automation Conference (DAC)*, pages 685–690, New Orleans, LO, June 2002.

[82] Erik Larsson and Zebo Peng. A Reconfigurable Power-Conscious Core Wrapper and its Application to SOC Test Scheduling. In *Proceedings IEEE International Test Conference (ITC)*, pages 1135–1144, Charlotte, NC, September 2003.

[83] International Technology Roadmap for Semiconductors (ITRS). 2009.

[84] Li Zhou, C. Wakayama, and C.-J.R. Shi. Cascade: A standard supercell design methodology with congestion-driven placement for three-dimensional interconnect-heavy very large-scale integrated circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(7):1270 –1282, july 2007.

[85] S. Koranne. Design of Reconfigurable Access Wrappers for Embedded Core Based SoC Test. 11(5):955–960, 2003.

[86] Sandeep Kumar Goel and Erik Jan Marinissen. SOC Test Architecture Design for Efficient Utilization of Test Bandwidth. *ACM Transactions on Design Automation of Electronic Systems*, 8(4):399–429, October 2003.

[87] T. Yamagata, H. Sato, K. Fujita, Y. Nishimura, and K. Anami. A distributed globally replaceable redundancy scheme for sub-half-micron ULSI memories and beyond. *IEEE Journal of Solid-State Circuits*, 31(2):195–201, 1996.

[88] S.Y. Kuo and W.K. Fuchs. Efficient spare allocation in reconfigurable arrays. In *ACM/IEEE conference on Design automation*, pages 385–390. IEEE Press Piscataway, NJ, USA, 1986.

[89] H. Fernau and R. Niedermeier. An efficient exact algorithm for constraint bipartite vertex cover. *Journal of Algorithms*, 38(2):374–410, 2001.

[90] C.T. Huang, C.F. Wu, J.F. Li, and C.W. Wu. Built-in redundancy analysis for memory yield improvement. *IEEE Transactions on Reliability*, 52(4):386–399, 2003.

[91] S. Bahl. A Sharable Built-in Self-repair for Semiconductor Memories with 2-D Redundancy Scheme. In *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2007.

[92] R.F. Huang, J.F. Li, J.C. Yeh, and C.W. Wu. A simulator for evaluating redundancy analysis algorithms of repairable embedded memories. In *Proc. IEEE Int. Workshop on Memory Technology, Design and Testing (MTDT)*, pages 68–73.

[93] CH Stapper, AN McLaren, and M. Dreckmann. Yield model for productivity optimization of VLSI memory chips with redundancy and partially good product. *IBM J. Res. Develop*, 24(3):398–409, 1980.

[94] C.L. Wey and F. Lombardi. On the repair of redundant RAM's. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 6(2):222–231, 1987.

[95] B. Kim, C. Sharbono, T. Ritzdorf, and D. Schmauch. Factors affecting copper filling process within high aspect ratio deep vias for 3d chip stacking. In *Electronic Components and Technology Conference*, pages 6–pp., 0-0 2006.

[96] A.P. Karmarkar, Xiaopeng Xu, and V. Moroz. Performanace and reliability analysis of 3d-integration structures employing through silicon via (tsv). In *IEEE International Reliability Physics Symposium*, pages 682 –687, april 2009.

[97] A. W. Topol, D. C. La Tulipe, L. Shi, D. J. Frank, K. Bernstein, S. E. Steen, A. Kumar, G. U. Singco, A. M. Young, K. W. Guarini, and M. Ieong. Three-dimensional integrated circuits. *IBM Journal of Research and Development*, 50(4.5):491 –506, july 2006.

[98] J.S.N. Jean, H.C. Fu, and S.Y. Kung. Yield enhancement for wsi array processors using two-and-half-track switches. In *Proceedings International Conference on Wafer Scale Integration*, pages 243 –250, jan 1990.

[99] T. Zhang, Kui Wang, Yi Feng, Xiaodi Song, Lian Duan, Yuan Xie, Xu Cheng, and Youn-Long Lin. A customized design of dram controller for on-chip 3d dram stacking. In *IEEE Conference on. Custom Integrated Circuits,*, pages 1 –4, sept. 2010.

[100] L. Jiang, L. Huang, and Q. Xu. Test architecture design and optimization for three-dimensional socs. In *Proceedings Design, Automation, and Test in Europe Conference Exhibition*, pages 220 –225, april 2009.

[101] A.V. Goldberg and S. Rao. Beyond the flow decomposition barrier. *Journal of the ACM (JACM)*, 45(5):783–797, 1998.

[102] G. Baier, T. Erlebach, A. Hall, E. Kohler, H. Schilling, and M. Skutella. Length-bounded cuts and flows. *Automata, languages and programming*, pages 679–690, 2006.

[103] F.J. Meyer and D.K. Pradhan. Modeling defect spatial distribution. *IEEE Transactions on Computers*, 38(4):538–546, 1989.

[104] M.B. Tahoori. Defects, yield, and design in sublithographic nano-electronics. In *Proceedings IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, pages 3–11, 2005.

[105] A.W. Topol, D.C. La Tulipe, L. Shi, S.M. Alam, D.J. Frank, S.E. Steen, J. Vichiconti, D. Posillico, M. Cobb, S. Medd, J. Patel, S. Goma, D. DiMilia, M.T. Robson, E. Duch, M. Farinelli, C. Wang, R.A. Conti, D.M. Canaperi, L. Deligianni, A. Kumar, K.T. Kwietniak, C. D'Emic, J. Ott, A.M. Young, K.W. Guarini, and M. Ieong. Enabling soi-based assembly technology for three-dimensional (3d) integrated circuits (ics). In *IEEE International Electron Devices Meeting*, pages 352 –355, dec. 2005.

[106] Bart Swinnen, W Ruythooren, P De Moor, L Bogaerts, L Carbonell, K De Munck, B Eyckens, S Stoukatch, D Sabuncuoglu Tezcan, Z Tokei, et al. 3d integration by cu-cu thermo-compression bonding of extremely thinned bulk-si die containing 10 μm pitch through-si vias. In *International Electron Devices Meeting.*, pages 1–4. IEEE, 2006.

[107] N Miyakawa, T Maebashi, N Nakamura, S Nakayama, E Hashimoto, and S Toyoda. New multi-layer stacking technology and trial manufacture. *3D Architectures for Semiconductor Integration and Packaging*, 2007.

[108] D.Z. Pan, S.K. Lim, K. Athikulwongse, M. Jung, J. Mitra, J. Pak, M. Pathak, and J. Yang. Design for manufacturability and reliability for tsv-based 3d ics. In *Asia and South Pacific Design Automation Conference*, pages 750–755. IEEE, 2012.

[109] KN Tu. Reliability challenges in 3d ic packaging technology. *Microelectronics Reliability*, 51(3):517–523, 2011.

[110] M. Nicolaidis, V. Pasca, and L. Anghel. Through-silicon-via built-in self-repair for aggressive 3d integration. In *IEEE International On-Line Testing Symposium*, pages 91–96. IEEE, 2012.

[111] L. Jiang, R. Ye, and Xu Q. Yield enhancement for 3d-stacked memory by redundancy sharing across dies. In *Proceedings International Conference on Computer-Aided Design*, pages 230 –234, nov. 2010.

[112] Y-J Huang and J-F Li. Built-in self-repair scheme for the tsvs in 3-d ics. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(10):1600–1613, 2012.

[113] J. Xie, Y. Wang, and Y. Xie. Yield-aware time-efficient testing and self-fixing design for tsv-based 3d ics. In *Asia and South Pacific Design Automation Conference*, pages 738–743. IEEE, 2012.

[114] F. Ye and K. Chakrabarty. Tsv open defects in 3d integrated circuits: Characterization, test, and optimal spare allocation. In *Design Automation Conference*, pages 1024–1030. ACM, 2012.

[115] L. Jiang, Q. Xu, and B. Eklow. On effective through-silicon via repair for 3-d-stacked ics. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(4):559–571, 2013.

[116] T. Dao, D.H. Triyoso, M. Petras, and M. Canonico. Through silicon via stress characterization. In *IEEE International Conference on IC Design and Technology*, 2009.

[117] C.S. Selvanayagam, J.H. Lau, X. Zhang, S. Seah, K. Vaidyanathan, and T.C. Chai. Nonlinear thermal stress/strain analyses of copper filled tsv (through silicon via) and their flip-chip microbumps. *IEEE Transactions on Advanced Packaging*, 32(4):720–728, 2009.

[118] Y. Li, S. Makar, and S. Mitra. Casp: concurrent autonomous chip self-test using stored test patterns. In *Proceedings of the conference on Design, automation and test in Europe*, pages 885–890. ACM, 2008.

[119] Y.-J. Huang, J.-F. Li, J.-J. Chen, D.-M. Kwai, Y.-F. Chou, and C.-W. Wu. A built-in self-test scheme for the post-bond test of tsvs in 3d ics. In *VLSI Test Symposium*, pages 20–25. IEEE, 2011.

[120] Claude Berge. Two theorems in graph theory. *Proceedings of the National Academy of Sciences of the United States of America*, 43(9):842, 1957.

[121] J.-Q. Lu, K. Rose, and S. Vitkavage. 3d integration: Why, what, who, when? *Future Fab Int*, 23:25–26, 2007.