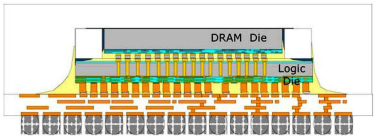
Computer Engineering
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ce.et.tudelft.nl/

2010

MSc THESIS

On 3D Stacked IC Yield Improvement and 3D-DfT Test Architecture

J Verbree



CE-MS-2010-17

Abstract

Three-dimensional stacked ICs (3D-SICs) based on Through-Silicon Vias (TSVs) is an emerging technology. It provides heterogeneous integration, higher performance and bandwidth, and lower power consumption. However, 3D-SICs suffer from lower compound yield, especially those based on Wafer-to-Wafer (W2W) stacking. In addition, testability of such devices is still in its infancy stage. This thesis addresses these two challenges.

To improve the compound yield of W2W 3D-SICs, a technique known as wafer matching will be used. It defines the best matching of top and bottom wafers from repositories of pre-tested wafers. The simulation results show that the compound-yield increase depends on (1) the number of stack tiers, (2) the number of dies per wafer, (3) the die yield, and (4) the repository size. Moreover, they demonstrate that, for realistic cases, relative yield increases of 0.5% to 10% can be achieved.

The thesis also proposes a three-dimensional Design-for-Test (3D-DfT) architecture that solves the testability issue of of 3D-SICs. The architecture is based on a modular approach, in which the various dies, their embedded IP cores, TSV-based interconnect, and external I/O can be tested as separate units, before and/or after bonding. Furthermore, the architecture leverages existing 2D DfT already present in the design, and adds a die-level wrapper based on IEEE Std 1500 augmented with additional features in order to be able to deals with 3D-SIC challenges (e.g., to transport signals up and down through the stack). The architecture is implemented and the simulation results show that it provides the flexibility and the modularity is realized at the cost of less than 0.1% area overhead when considering large industrial chips. The architecture could serve as a basis for further standardization of DfT for 3D-SICs.

TU Delft
Delft University of Technology

Faculty of Electrical Engineering, Mathematics and Computer Science

On 3D Stacked IC Yield Improvement and 3D-DfT Test Architecture

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

by

J Verbree
born in Harderwijk, The Netherlands

Computer Engineering
Department of Electrical Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

On 3D Stacked IC Yield Improvement and 3D-DfT Test Architecture

by J Verbree

Abstract

Three-dimensional stacked ICs (3D-SICs) based on Through-Silicon Vias (TSVs) is an emerging technology. It provides heterogeneous integration, higher performance and bandwidth, and lower power consumption. However, 3D-SICs suffer from lower compound yield, especially those based on Wafer-to-Wafer (W2W) stacking. In addition, testability of such devices is still in its infancy stage. This thesis addresses these two challenges.

To improve the compound yield of W2W 3D-SICs, a technique known as wafer matching will be used. It defines the best matching of top and bottom wafers from repositories of pre-tested wafers. The simulation results show that the compound-yield increase depends on (1) the number of stack tiers, (2) the number of dies per wafer, (3) the die yield, and (4) the repository size. Moreover, they demonstrate that, for realistic cases, relative yield increases of 0.5% to 10% can be achieved.

The thesis also proposes a three-dimensional Design-for-Test (3D-DfT) architecture that solves the testability issue of of 3D-SICs. The architecture is based on a modular approach, in which the various dies, their embedded IP cores, TSV-based interconnect, and external I/O can be tested as separate units, before and/or after bonding. Furthermore, the architecture leverages existing 2D DfT already present in the design, and adds a die-level wrapper based on IEEE Std 1500 augmented with additional features in order to be able to deals with 3D-SIC challenges (e.g., to transport signals up and down through the stack). The architecture is implemented and the simulation results show that it provides the flexibility and the modularity is realized at the cost of less than 0.1% area overhead when considering large industrial chips. The architecture could serve as a basis for further standardization of DfT for 3D-SICs.

Laboratory : Computer Engineering
Codenumber : CE-MS-2010-17

Committee Members :

Advisor:	Erik Jan Marinissen, SSET, IMEC
Advisor:	Said Hamdioui, CE, TU Delft
Chairperson:	Koen Bertels, CE, TU Delft
Member:	Sorin Dan Cotofana, CE, TU Delft
Member:	René van Leuken, CAS, TU Delft

Contents

List of Figures

1 Introduction

The worlds demands for increasingly faster and more capable computer chips have led to a continual growth of the number of transistors on a single chip. This push has put a strain on chip design, to allow for this ever increasing numbers of transistors to work together in a efficient way. The computer industry is currently developing Three-Dimensional Stacked Integrated Circuits (3DSICs) to meet this future demand. The inability of the computer chip manufacturing process to guarantee flawless computer chips necessitates the testing of chips after manufacturing; this still holds for 3D-SICs. This thesis presents two studies into 3D-SICs testing; a method for increasing the stacked yield of 3D-SICs by using test results, and the development of a 3D-DfT test access architecture that enabled 3D-SIC testing.

This chapter presents some basic about 3D-SICs; it also provides the main contributions of the thesis as well as its outline. It is organized as follows. Section 1.1 covers the advantages of 3D-SICs has over two-dimensional designs, but also raises some the problems associated with 3D-SICs in particular in regards to a low stacked yield and poor testability. In Section 1.2 the contributions of this masters project in terms of wafer matching and a 3D-DfT architecture are stated. Section 1.3 states how this thesis is organized into chapters, and provides a broad scope overview of each chapter.

1.1 Potential and challenges of 3D-SICs

3D-SICs are stacks of ‘ordinary’ two-dimensional dies as in production today. The idea behind 3D-SICs, is that combining multiple dies into a stack enables the heterogeneous integration of die technologies into a single product. This allows for a memory die to be manufactured in a process optimized for memory, to be combined with a logic die manufactured in a for logic optimized process into a single chip product. Another advantage is that by placing the dies together in a stack the footprint is reduced compared to two separate chips of equal size, this is of particular interest in markets where PCB area comes at a premium, such as mobile and embedded markets.

In this thesis we will focus on Through Silicon Via (TSV) based 3D-SICs, TSVs provide interconnect between the dies in the stack. With the use of TSVs advantages of 3D-SICs over 2D designs further increase, as the TSVs allow for a low latency, low power, high bandwidth connections between the dies compared to off-chip communication in a wire-bond enabled 3D-SIC or 2D design.

One of the methods for chip stacking to create 3D-SICs is Wafer-to-Wafer stacking. In this method an entire wafer of dies is stacked onto another wafer of dies, bonding all the dies in a single process. This method of stacking has many attractive benefits over other stacking methods,

such as higher production throughput and it allows for the highest bandwidth between dies in a stack [2]. Unfortunately it suffers from a low yield compared to the other stacking methods, even when the dies are tested it is not possible to exclude faulty dies from the stacking process as the wafer is not diced before hand. The faulty dies of one wafer may end up paired to a functioning die on the other wafer, rendering that die worthless as the resulting stack is unsellable.

Testing dies intended to be part of a 3D-SIC is not trivial, as these dies do not have pads intended to be fitted with external pins, but instead have pads intended to connect to the much smaller TSV tips. These TSV landing pads are too small to allow probing equipment access. There also are testing problems for the 3D-SIC after bonding. A die forming a layer in the stack does not necessarily has an external interface after stacking. Testing such a layer has to be carried out through other layers of the same stack. Currently no structured way of accomplishing a stack test of all layers exists.

1.2 Contributions of the project

We propose a method of improving the Wafer-to-Wafer (W2W) yield, called ‘wafer matching’. In this method repositories of wafers are matched, instead of just a single wafer with a single other wafer. The addition of repositories allows an algorithm to use wafer-level test results to find wafers in which faulty dies ‘match’ as much as possible. A ‘match’ is when a faulty die is stacked on top of another faulty stack. Higher concentration of faulty dies into a single stack, increase the chance of other stacks being fault free, leading to an improved stacked yield. We study wafer matching in great detail in this thesis, we show how wafer matching can improve the stacked yield by as much as 10%.

Our contribution to the field of wafer matching is the following

- We have captured the concept of wafer matching into a close-loop mathematical model and identified the number of dies per wafer, faulty dies per wafer, stack height, and repository size as factors that influence the effectiveness of wafer matching. In addition, the model gives a fundamental insight into how W2W matching actually functions, as based on probability theory.
- A simulator has been created to further investigate wafer matching in a realistic production environment, to quantify trends and make it possible to accurately predict the yield benefit wafer matching would provide given a set of production details. We present simulation results, focusing especially on the effects of a varying repository size. We do away with the possibly skewed results caused by using random generators in generation of wafer maps[1, 2] by repeating each experiment 10000 times. Minimizing any misleading effects a single random draw might have, by taking an average over a large number of draws.
- We also provides a cost/benefit analysis for wafer matching, where we balance the additional costs of performing wafer-level tests on the dies to the benefit of additional yield. It considers several testing strategies, and demonstrates the impact of these strategies on the costs per stack. Where a higher yield decreases the cost per die, the addition of tests increases the costs. More aspects have been looked upon, for instance, the introduction of

the wafer level die tests diminishes the need for some of the stack tests that would have taken place at later stages of the production process.

To enable structured testing of every die in a stack during and after manufacturing we introduce a modular Design for Test (DfT) architecture that allows for TSV-based 3D-SICs to be tested. This 3D-DfT architecture covers testing of the layers before and after stacking has occurred, the flexibility it provides allows for various test strategies and optimizations in the test process. We describe this architecture in detail, and show that using IEEE 1500 standard for core testing [3] as a base for the 3D-DfT architecture allows for a leveraging existing 2D DfT all ready present in the dies. By outfitting a 2D dies with a IEEE 1500 style wrapper, thus isolating a die from the stack and allowing access to the isolated die from external IO, it is possible to test a die in a very similar fashion to 2D testing.

Our contribution to 3D-SIC testing is the following

- We list the requirements and constraints for a 3D-DfT architecture to enable 3D-SIC testing during the various stages of its production process. We introduce a 3D-DfT architecture based on IEEE 1500 standard for core testing, that meets these requirements and constraints. Our 3D-DfT architecture leverages as much as possible from existing 2D-DfT commonly implemented in the die, allowing for an easy integration into current chip design, and Electronic Design Automation (EDA) semiconductor design tools.
- To provide verification of the 3D-DfT architecture's functionality, the 3D-DfT architecture has been incorporated into three 2D computer chip designs. The 3D-DfT architecture being based on IEEE 1500, has in part been implemented using IEEE 1500 functionality already provided by the EDA tools. Where EDA support was unavailable, 3D-DfT features have been added to the designs by hand.
- Simulation of the fault coverage for a die incorporated in a stack with the 3D-DfT architecture present compared to the same die tested as a 2D chip has yielded extremely similar results. Thereby proving the 3D-DfT architecture has been effective in enabling 3D-SIC testing in all its facets.
- The cost of implementing our 3D-DfT architecture has been also been researched in terms of additional dedicated DfT probe pads, TSVs and area.
- The verification process has provides valuable information regarding the requirements for EDA tools to support the 3D-DfT and resulted into recommendations for the EDA vendors

1.3 Thesis organisation

This remaining parts of the thesis are organized as follows. Chapter 2 gives an introduction into 3D-SICs. In this chapter characteristics of 3D-SICs and the 3D manufacturing process and its implications on testing will be put forward. Chapter 3 describes the process of wafer matching, the effectiveness of wafer matching, and the implications wafer matching has on production. Chapter 4 entails our 3D-DfT architecture, the requirements, a full description of the architecture

itself, verification of the architecture, and an analysis of costs of implementation. Chapter 5 concludes this Thesis.

2 Background to 3D Stacked ICs

This chapter introduces three-dimensional Stacked Integrated Circuits (3D-SIC), insight into the driving forces behind the industry's push towards 3D-SICs, and summarizes the processes involved with the production of 3D-SICs. The information in this chapter will serve as background to the Wafer-to-Wafer 3D-SIC yield improvement methodology and 3D-DfT architecture covered in chapters 3 and 4.

This chapter is organized as follows, Section 2.1 covers 3D-SICs, what exactly are 3D-SICs, and why do we want or need 3D-SICs. Section 2.2 provides a high-level overview of the manufacturing process for 3D-SICs, the chapter ends with Section 2.3 that covers the specific testing characteristics of 3D-SICs.

2.1 3D-Stacked ICs

Today's integrated circuits are two-dimensional designs, by that is meant that the chips are created on the surface of a flat silicon wafer. On these wafers there are fields of transistors, all laying next to each other, forming flat structures. This is not unlike the way farmers grow crops, and there is a good reason for this. During the growing process of the crops, sunlight, water and the farmer's equipment all need to be able to reach the crops. The same holds for the production of chips, where the transistors are manufactured on a flat silicon wafer and all at the same time, and the required process steps all need access to the transistors.

The world's demands for increasingly faster and more capable computer chips have led to a continual growth of the number of transistors on a chip (Moore's Law). The fact that the average size of a chip has not increased proportionately is due to fact that transistors themselves have decreased in size. For many years, downsizing the transistor to allow for more transistors per given area kept pace with the growing number of transistors. Larger die sizes result in an increase of the probability of a defect being present on the die [4], thereby raising cost. Therefore keeping the dies small kept cost down; hence, in part, kept the progression of Moore's Law sustainable. However, keeping up with the growing number of transistors is becoming more and more difficult as the limits of CMOS technology are close to being reached. Another issue of high density chips, is the amount of interconnecting wires it requires [5]. Adding metal layers contributes to the overall costs of the chip. The length of these interconnecting wires, can easily reach several millimeters, a lot when you consider the size of a transistor. Another downside of long wires is they are slow compared to shorter wires, very long wires can negatively affect the obtainable clock speed of a chip.

Another trend in microelectronics industry is to move dies ever closer to each other. The Printed

Circuit Board (PCB) has individually packaged dies placed on a board, has been joined by the Multi-Chip Package (MCP) and later the System-in-Package (SiP). In Figure 2.1 shows how the PCB, MCP and SiP relate to one another. In the MCP, the dies are placed next to each other in a single package,while in SiP, the dies are packed even closer together; they are stacked on top of one another. The communication between these stacked dies is performed through wire bonds. The older PCB and MCP style layout are still widely used today. However, in areas where space is at a premium while the functionality requirements are high like in the mobile market, SiP technology is commonplace. From these sectors there is still a push to package the chips tighter than ever before. This has been one of the motivators of the Three-Dimensional Stacked IC (3D-SIC) as has been shown in Figure 2.1(d). 3D-SIC is similar to the SiP in concept, but instead of using external wire bonds, the interconnect is made through the dies themselves.

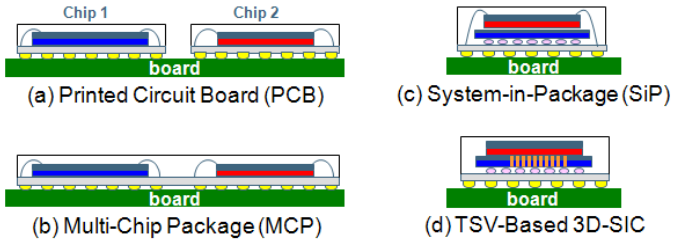


Figure 2.1: Dies are being placed ever closer to each other.

The microelectronics industry is researching the use of Through Silicon Vias (TSVs), as a method for enabling the interconnect between these stacked dies forming tiers in the 3D-SIC. TSVs connect metal layers together just as ordinary vias do, but in this case, the metal layers belong to two different tiers within the stacked IC. TSVs are inter-tier vias, as they stick out perpendicular to the metal layers right through the substrate of their tier, enabling a connection to the metal layers of another tier. The large benefit TSVs have in comparison to wire bonds is their size. Wire bonds are considered expensive, slow and energy-inefficient as interconnects due to their bulk. In contrast, TSVs are much smaller and are comparable to on-die interconnects. The downside is the costs for the additional production steps necessary to create the TSVs.

3D-SICs based on TSVs are a possible answer to the issues raised above ; 3D-SICs consist of multiple tiers of two-dimensional chips, directly connected to each other. By going three-dimensional, closely related memory and logic segments can physically be placed much nearer to each other, just one level in the Z-direction compared to vast distances in the X and/or Y direction. The tiers in a 3D-SIC may only be 50 μm apart, where a long interconnect can be in the order of several millimeters apart. Going three-dimensional eases congestion on buses and reduces average wires lengths [5], thereby increasing speed and lowering power consumption. An additional advantage is that the individual tiers can be optimized for logic or memory and still be integrated together into a 3D-SIC. In Figure 2.2 an example of the integration of logic and memory in a single device is shown. In the figure we see IMEC's prototype chip ETNA currently being manufactured and evaluated, a memory die is placed on top of a logic die, this logic die retains its metalization layers and external IO via the Ball Grid Array.

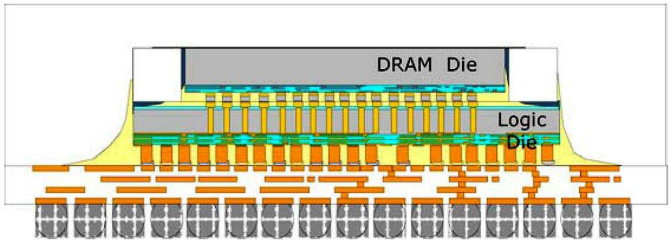


Figure 2.2: Heterogeneous integration on IMEC's ETNA chip.

2.2 Manufacturing 3D-Stacked ICs

This section covers the some of the additional production steps required to build 3D-SICs. The first subsection will cover the creation of the TSVs themselves. The second subsection is devoted to the thinning process necessary to be able to uncover and use the TSVs; While the last subsection addresses the process of stacking the dies.

2.2.1 Through Silicon Vias

IMEC is prototyping and evaluating 3D-SICs based on TSVs. It has been found that there are three categories of TSVs, Depending on where in the production process the TSV manufacturing takes place, the 'Via First', 'Via Middle', and 'Via Last' TSVs. TSV manufacturing can take place before, between, or after the creation of the transistors and the metal layer creation processes. Where the 'Via Last' methodology is relatively easy to perform, it is quite expensive per TSV, and the TSVs are rather large. Large TSVs suffers from a large pitch, high power consumption, slow speed, and have to be put relatively far away from active logic. The two other types of TSVs, made earlier in the production process, are much smaller and thus have a smaller pitch, consume less power, are faster, and can be placed neared to active logic. But do have the disadvantage of being harder to incorporate in the production process, and because of that these categories of TSVs can currently only be found in prototype chips. 3D-SICs based on 'Via Last' TSVs are commercially available, an example are CMOS image sensors [6]. The cost of additional TSVs of the 'Via first' or'Via Middle' categories into a design that already has them is very low, while the setup costs for TSV manufacturing are quite high. Hence, for these catagories of TSVs, it makes sense to have many TSVs, enabling high bandwidth communication between the dies.

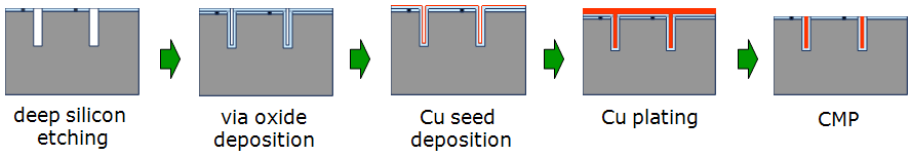


Figure 2.3: Fundamental production steps for creating TSVs.

The TSV length is variable, as it depends on the ratio of the diameter and the length of the TSVs

and the diameter itself. Advanced research facilities are typically finding a TSV diameter/length ratio of 1:5 already quite challenging although IMEC is pushing towards 1:10. Longer thinner TSVs are hard to fabricate, as the holes etched in the silicon are very hard to properly fill with copper. Figure 2.3 shows several steps of the TSV manufacturing process, first holes are etched into the silicon substrate, second is the addition of insulation in the hole to prevent leakage into the substrate. In subsequent steps the holes are filled with the copper which forms the actual TSV, Chemical/Mechanical Planarization (CMP) removes the excess copper.

2.2.2 Thinning Wafers

The limited obtainable aspect ratios of TSVs makes the length of the TSV almost solely dependent on the diameter of the TSV. In order for TSVs to be fast and limit the impact on the original chip design, the diameter and closely related TSV pitch must be kept as small as possible. The length of TSVs is therefore limited to no more than 50μm. On one hand, this is quite good because it means the wire length to the next tier in the stack is also short. And short wires have lower power requirements and can operate at higher speeds compared to longer wires. On the other hand 50μm leaves the TSVs buried deeply into the silicon substrate, the 770μm thick wafer therefore has to be thinned down to such a thickness that the TSV will stick out of the backside. This means reducing the thickness of the wafer to about 5%˜10% of its original thickness. This process involves grinding the wafer down to the top of TSVs, at which time chemicals are used to remove a little more of the substrate leaving the TSV tips to physically stick out from the back of the thinned wafer. To prevent the very thin wafer from cracking under the mechanical stresses of the thinning process, it is bonded to a temporary full thickness carrier wafer before the thinning process commences. Figure 2.4 describes the process of wafer thinning. First the tier is bonded to a carrier wafer, then thinned in several stages untill the TSVs are exposed. The thinned die is subsequently stacked onto another die (designated as 'Bottom Die' in the figure), connecting to the landing pads, directly to the metal layers. Finally the carrier wafer is removed, when the bonding has been completed.

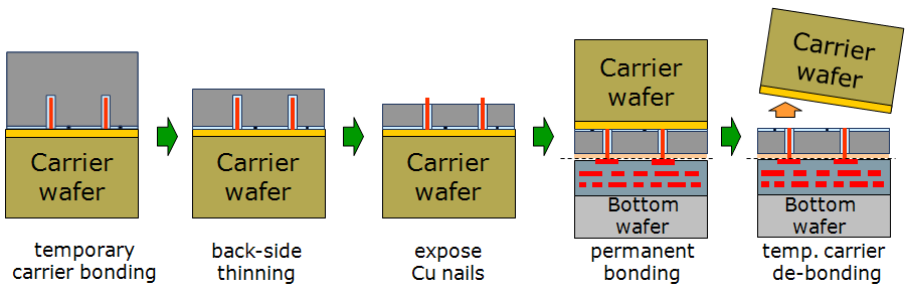
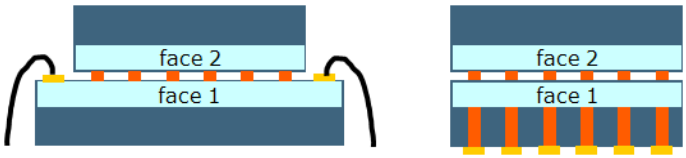


Figure 2.4: Production steps related to the thinning and bonding process.

2.2.3 Stacking Dies

Considering dies with metal layers and pads on the front side and TSV tips on the backside, there are three basic ways to stack two dies together. One may consider Face-to-Face stacking as de-

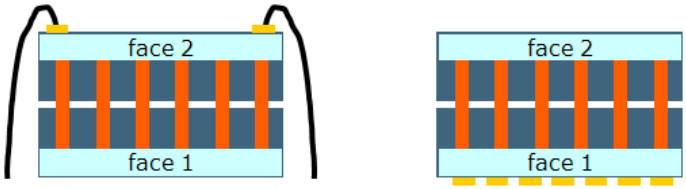
picted in Figure 2.5, the front sides of two chips are bonded together and off-chip communication is performed via wire bonds(a) or via TSVs(b) in a fashion similar to flip-chip.



(a) m (b)

Figure 2.5: Face-to-Face stacking with (a) wire bonds or (b) TSVs.

Alternatives to Face-to-Face stacking are Back-to-Back and Back-to-Face stacking, as shown in Figures 2.6 and 2.7. In Back-to-Back stacking, the backside of one chip is bonded to the backside of the another chip while with Back-to-Face it is bonded to the frontside of another chip. After stacking a TSV tip is supposed to connect to a TSV landing pad on the receiving tier. But in order to successfully connect thousands of TSVs per mm^2, the TSV tips needs to stick out of the backside by exactly the same amount. Also the contact surfaces need the be extremely flat, and not warped in any way. Failure to meet these requirements will result into only a portion of TSVs making good contact. In order to alleviate these problems copper-tin (CuSn) micro bumps are placed on the TSV tips; this relatively flexible material allows the connection to be made even though there are slight differences in height. Figure 2.8(a) shows such a micro bump and how it forms to suit the requirements. The drawback is that the micro bumps are much larger than the TSV tips were, reducing the maximum density of the TSVs severely. Moreover, the addition of metal to this via will make it slower to respond and consume more power. When the technology surrounding 3D-SICs matures, IMEC hopes to investigate TSV interconnects without micro bumps. A proposed process is called copper to copper, referring to the copper TSV tips and copper TSV landing pads. In Figure 2.8(b) one of these TSV tips is shown, and a size comparison can be made to the CuSn micro bump.

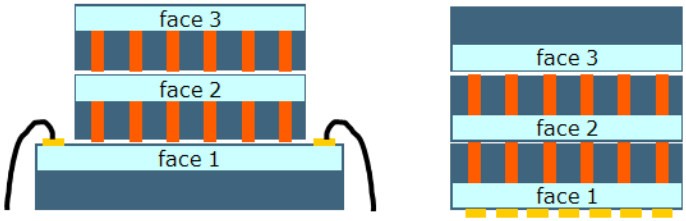


(a) m (b)

Figure 2.6: Back-to-Back stacking with (a) wire bonds or (b) with a Flip-Chip Layout.

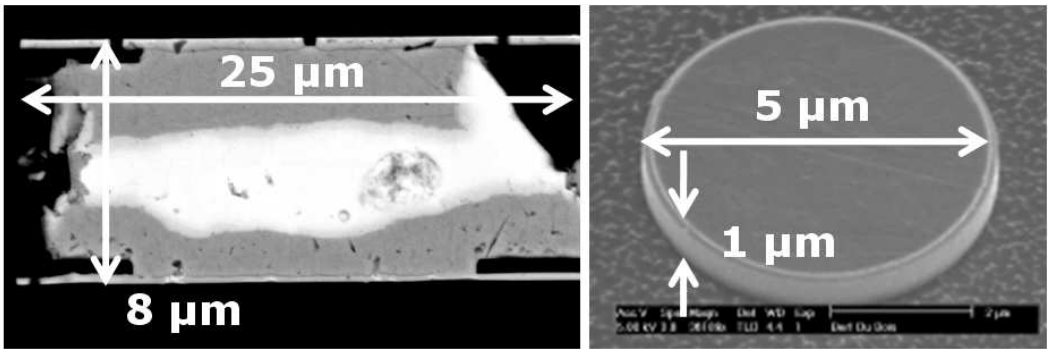
2.3 Challenges in Testing 3D-Stacked ICs

3D-SICs testing differs from conventional two-dimension chip testing in three areas: *Test Content*, *Test Access*, and *Test Flow*; they are discussed next.



(a) (b)

Figure 2.7: Back-to-Back stacking with (a) wire bonds or (b) with a Flip-Chip Layout.



(a) (b)

Figure 2.8: (a) copper-tin-copper micro bump, (b) TSV tip.

Test Content

A 3D-SIC is susceptible to all the manufacturing defects we are used to in two dimensional chips. This is because the individual tiers are essentially 2D chips with additional 3-D specific hardware. This additional hardware needs to be tested also. Some of this hardware is regular hardware being used to perform a 3D task, and should be tested as such. But access to this hardware may be troublesome, as it is part of a bridge between tiers, requiring the tiers to work together in order to perform a test. Some of the hardware however may be completely new, like TSVs for example. They form the interconnect between tiers, and fit somewhere in the middle between on-die interconnect and off-die wire bonds. This results in unique properties and variabilities, for which specific tests may be required. It is also possible that the steps required for the creation of the TSVs or the stacking process introduces defects into the product.

Test Access

Test access on a 3D-SIC will differ a great deal from 2D test access, principally because the non-bottom tiers are not designed to have external pins. These tiers are designed with a TSV-based interface in mind, and thus only have the TSV tips or TSV landing pads. Also the tiers can be tested before and after bonding to other tiers, when the tiers is on its own the TSV tips/pads are exposed. But after bonding these TSV tips/pads are covered by the other tier. We therefor

identify two separate tests, the *pre-bond* test and the *post-bond* test. In pre-bond testing, there is access to the TSV tips/pads; these are much smaller then regular access pins or pads. The TSV tips/pads are too small to be probed upon by a wafer prober. Thus the tiers become untestable unless another access mechanism is introduced. In Post-bond testing, the TSV tips/pads are completely covered, and access to a tier is only possible through the tier it is bonded to, right down to the bottom tier which has the external I/O pins. All these tiers will therefore need 3D specific DFT-hardware, which is able to provide this access to the tiers from the I/O pins and through the stack. A simple solution that provides access to all tiers is shown in Figure 2.9. In this figure we see separate test data flows through the stack to and from each tier, each tier has its own dedicated path to external IO. This solution does require all tiers to be designed for a particular location in the stack, and most likely require the tiers to be designed in conjunction with each other, and it does not handle the testing of the interconnect between the tiers in the stack. Other more applicable solutions for the encountered problems regarding pre-bond and post-bond testing will be presented in Chapter 4.

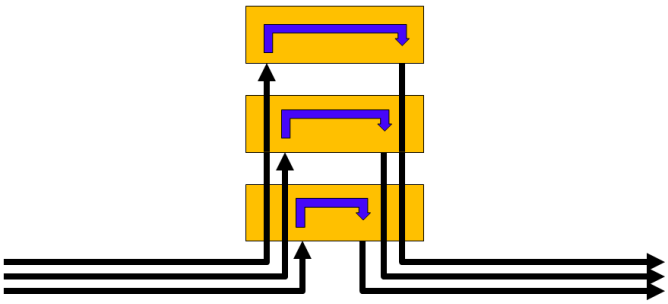


Figure 2.9: Access to non-bottom tiers has to be through other tiers.

Test Flow

The test flow of a 3D-SIC is quite different to the test flow of a conventional 2D chip. Two-dimensional chips are only testable when the whole product has been completed, either at wafer level or after packaging. In contrast 3D-SICs have multiple production moments as individual tiers are manufactured seperately; hence for the first time it is possible to test an unfinished product. This has an impact on the test flow. Instead of testing only after the final 3D-SIC product, now intermediate testing can be performed. The tiers can be tested before bonding in pre-bond die tests, and the stack can be (re)tested after each stacking step has been performed in the post-bond stack tests. The revised test flow for a 3D-SICs that includes these pre-bond die tests and post-bond (partical) stack tests is shown in Figure 2.10(b). A comparison can be made with the typical test flow for a conventional 2D chip in Figure 2.10(a). That figure also illustrates multiple post-bond stack tests, one after each tier is added to the stack.

Performing all these tests allows the manufacturer to select only those tiers or stacks that have passed their tests for further stacking, discarding the failing components. If 50% of the tiers is functional, stacking two arbitrary tiers will result in a 25% chance the stack is functional, with the selecting mechanism only the 50% functional tiers are stacked to for 100% functional stacks.

Taking the disregarded tiers in account the overall yield is now 50%, a vast improvement over the original 25% yield. Because of this effect 3D-stacking can leads to higher overall yields compared to two-dimensional-chips with the same amount of transistors, in 2D chips all the parts are all made in parallel, and swapping out parts of the chips is not possible, leading to the same overall yields as for the stack where there is no selection mechanism. As the tiers and stacks become larger and thus more expensive, it stresses the value of tier- and stack-tests even more, as a faulty tier not only renders itself but also the other tiers in the stack worthless if the faulty tier is introduced in the stack. Thus the value of the resulting stack and not just the tier should be considered.

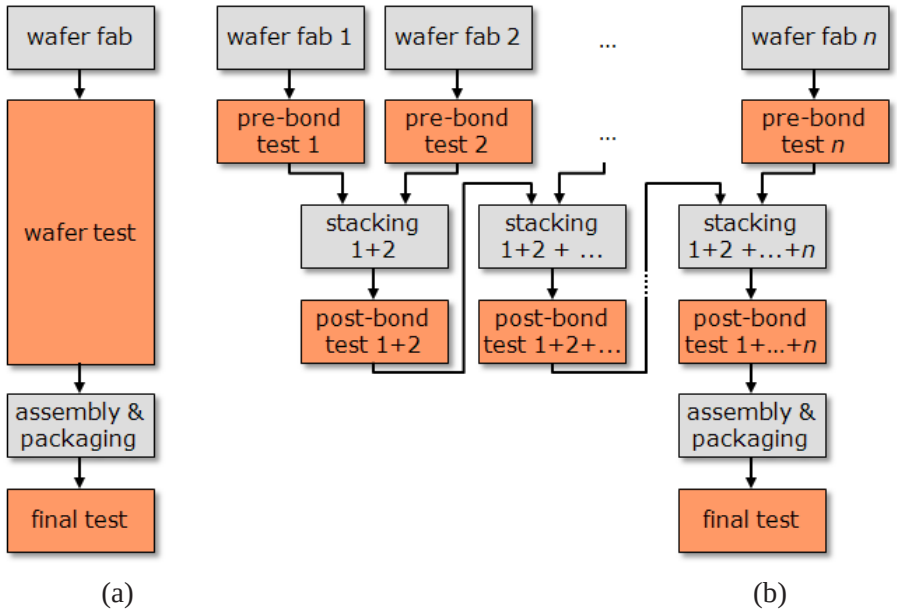


Figure 2.10: (a) typical 2D test flow, (b) 3D-SIC test flow.

3 Effects of Wafer Matching

The wafer-to-wafer (W2W) method to stack tiers to create 3D-SICS promises the highest throughput and lowest costs, and allows for the thinnest wafers and highest TSV density [1, 2] compared to other stacking methods. Unfortunately W2W stacking suffers from a low stacked yield compared to the alternative methods. This chapter covers a study into wafer matching, a technique that raises the 3D-SIC yield for the W2W stacking process.

In Section 3.1, we will introduce the concept of wafer matching for W2W stacking, how it works and what the consequences are for incorporating wafer matching into production. The remaining part of this chapter is organized as follows; Section 3.2 presents prior work carried out in this field, followed by Section 3.3 which is devoted to a theoretical model and describes the fundamental issues related to W2W matching and identifies key parameters related to the effectiveness of the process. Section 3.4 presents a simulation solution to W2W matching, building on the principles of Section 3.3 while doing away with some of its limitations. In that section simulator itself as well as its inner workings will be explained, followed by the presentation and interpretation of the simulation results. In Section 3.5 the costs involved with W2W matching are evaluated against the benefits in a cost model, Section 3.6 concludes this chapter with an overview and summary of the presented work.

3.1 Introduction

In the manufacturing of 3D-SICs, there are three methods of stacking two subsequent tiers. There is the Die-to-Die (D2D) method in which individual dies are singulated, and bonded. The Die-to-Wafer (D2W) method is a process in which dies of one layer are cut of their wafer and bonded to the dies of the other layer while those are still part of a wafer. The third method is the Wafer-to-Wafer (W2W) stacking process. In W2W entire wafers are bonded to each other in a single step and where the individual stacks are singulated from the (stacked) wafers after the bonding process has completed. The W2W method promises the highest throughput and lowest costs due to having to perform an alignment step only once per wafer, where the other methods have to perform such a step for every die. In addition, W2W allows for the thinnest wafers and highest TSV density [1, 2].

The different stacking methods do have an influence on the expected yield of the completed stack given equal yields of the individual tiers. In the D2D and D2W methods, the dies on the wafers can be tested before being bonded, and by selecting only those dies that pass the test, the overall stack yield is maximized. In the W2W method, using only dies that have passed the test is not possible, as the failing dies cannot be excluded from the passing dies. As an example let us define the wafer yield of all layers to be 80%, and the correctly functioning dies have been

identified by testing. In the D2D and D2W processes, the test information is used and only the functioning dies are selected to be part of a stack. The faulty dies can be discarded and as a result the stack yield is still 80% taking the number of dies of a wafer as a base. In W2W, the flexibility to match individual dies to each other does not exist; the wafers are bonded in their entirety. If we assume the functioning dies are distributed randomly across the wafer, then the chance of both dies in the stack are functioning correctly is 80% $\times$ 80% = 64%. The stack yield will suffer even more when we consider more than two tiers forming a stack, as the stack yield fundamentally is equal to the product of all the wafer yields of the wafers that form tiers in that stack.

This chapter shows a method of improving the W2W stacked yield through the use of wafer matching. A method in which multiple wafers predestined to form a certain layer in the stack, are compared to the wafers predestined to become other layers in the stack. In new production flow 3.1 prebond dies tests are performed at wafer level, the wafer maps are matched, the output of this matching process is utilized to reorder wafer containers so the wafers that are matched also arrive at the bonding stage in the same sequence.

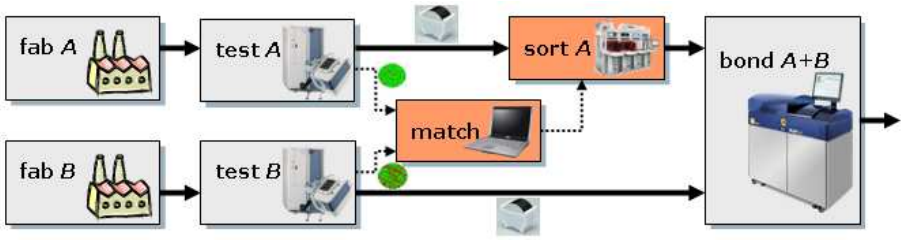


Figure 3.1: W2W production flow with additional matching and sorting steps after testing.

The idea behind wafer matching is to try to put as many of the dies that pass the prebond tests of fab A on top of other succsefully tested dies from fab b, thereby maximizing the stacked yield. Theoretically given infinit wafers with randomly distributed non functioning dies available to the matching process, there will be one with an identical wafer map. If this one is found and stacked there would be no additional yield loss, and the stacked yield will be equal to the wafer yield of 80%. We will look at how much of this effect can be utilized not given infinite amounts of wafers, but a fairly limited number stipulated by the reality of the manufacturing process. The wafer matching process has been repeated with a variety of hypothetical wafer maps, varying the die size, number of layers and individual layer yield to show the effects on the stacked yield and thereby the effectiveness of the W2W matching process.

3.2 Related Prior Work

The idea of wafer to wafer matching to increase the yield in the W2W process has first been put forward in[1], in this work the principle of W2W matching is presented as well as an experiment into the effectiveness of the process. Realistic parameters have been chosen; 300mm wafers containing 1320 dies each and a repository size of 25 which is equal to the size of a single wafer container as used by the IC industry. However the paper fails to show the influence of the individual variables on the effectiveness of W2W matching, moreover as his experiment is performed with a random generator as a source of wafer maps, and the experiment is only performed ones, the results are heavily dependent on the singular outcome of that random generator. The author also questions if the relative modest gains in terms of stack yield justify the expense of the wafer-level tests, but does not look further into the matter.

In [2] the authors take a more profound look at the W2W matching problem, and classifies it as a $\mathcal{NP}$-Hard problem. The authors did so by stating that the problem can be reduced to a known $\mathcal{NP}$-Hard problem [7]. It identified several parameters; wafer yield, die area, numbers of tiers in the stack and the number of wafers that are considered during matching as having

an influence on the ability of W2W matching to increase the stack yield. Furthermore it has looked at multiple algorithms that solve this problem, some are capable of solving the problem optimally as long as the problem size is kept within bounds, while others solve the problem non-optimally but are capable of solving larger instances of the W2W matching problem. The paper gives a comparison between these algorithms in terms of test time and effectiveness, but is hampered by the fact that also in this case I a random generator is used in combination with a limited sampling size, only 5 simulations have taken place per setting and algorithm. Resulting in that the outcome is highly dependent on those particular outputs of the generator.

- Our contribution over the prior work is the following The formulation of a close-form mathematical model describing the W2W matching process, approximates its effectiveness in practice, identifies the individual parameters that are affecting the expected stack yield. In addition, gives a fundamental insight into how W2W matching actually functions, as based on probability theory.
- We present simulation results, indicating clearly the effects of the individual parameters on the effectiveness W2W matching, especially focusing on the effects of a varying repository size. We do away with the possibly skewed results caused by using random generators in generation of wafer maps[1, 2] by repeating each experiment 10000 times. Minimizing any misleading effects a single random draw might have, by taking an average over a large number of draws.
- Thirdly we will present a cost model, it weighs the additional costs of W2W matching to the positive effect it has on stack yield. It considers several testing strategies, and demonstrates the impact of these strategies on the costs per die. Where a higher yield decreases the cost per die, the addition of tests increases the costs. More aspects have been looked upon, for instance, the introduction of the wafer level die tests diminishes the need for some of the stack tests that would have taken place at later stages of the production process.

3.3 Mathematical Model

In this section, we derive a closed-form mathematical equation that approximates the expected stack yield that can be obtained through wafer matching as a function of the number of stack tiers, the number of dies per wafer, the yield per wafer, and the wafer repository size.

3.3.1 Stacking Two Individual Wafers

First, we model the expected yield of stacking two individual wafers, each containing d dies. The *top* and *bottom* wafer have respectively f_t and f_b faulty dies, as determined by their pre-bond die tests. We assume $0 \leq f_t \leq f_b \leq d$. This assumption is without loss of generality; in case $f_b < f_t$, we simply relabel top and bottom.

Let us consider the case with $d = 6$, $f_b = 3$, and $f_t = 2$ as a simplified but illustrative example running throughout this section. Two example wafers that fulfill these settings are depicted in

Figure 3.2.

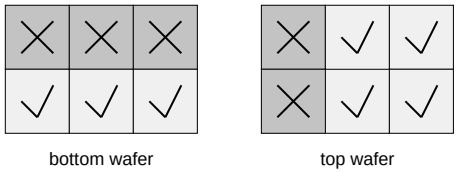


Figure 3.2: Simplified running example: $d = 6$, $f_b = 3$, and $f_t = 2$.

Function $y(i)$ denotes the compound yield of the stacked wafers[1] for exactly i matching faulty dies (with $0 \leq i \leq f_t$):

$$y(0) = \frac{1}{6} \quad , \qquad y(1) = \frac{2}{6} \quad , \qquad y(2) = \frac{3}{6}$$

If we generalize this, we get

$$y(i) = \frac{\max(d - f_b - f_t + i, 0)}{d} \tag{3.1}$$

Function $p(i)$ denotes the probability of the occurrence of exactly i matching faulty dies. $p(i)$ can be expressed as (1) the number of possibilities to match i faulty top dies to f_b faulty bottom dies, multiplied by (2) the number of possibilities to match the remaining $f_t - i$ faulty top dies to $d - f_b$ good bottom dies, divided by (3) the number of possibilities to match all f_t faulty top dies to all d bottom dies:

$$p(i) = \binom{f_b}{i} \cdot \binom{d - f_b}{f_t - i} / \binom{d}{f_t} \tag{3.2}$$

For our running example, the complete function $p(i)$ is defined as follows:

$$p(0) = \binom{3}{0} \cdot \binom{6-3}{2-0} / \binom{6}{2} = \frac{6}{30}$$
$$p(1) = \binom{3}{1} \cdot \binom{6-3}{2-1} / \binom{6}{2} = \frac{18}{30}$$
$$p(2) = \binom{3}{2} \cdot \binom{6-3}{2-2} / \binom{6}{2} = \frac{6}{30}$$

Note that it must always hold that $\sum_{i=0}^{f_t} p(i) = 1$.

The expected yield Y can now be expressed as

$$Y = \sum_{i=0}^{f_t} (y(i) \cdot p(i)) \tag{3.3}$$

[1] In Sections 3.3 and 3.4, the stack yield explicitly disregards yield loss due to stack interconnect faults and intra-tier faults introduced into tiers by the stacking process itself. Only in Section 3.5, this yield loss is again taken into account.

For our running example, this equates to

$$Y = \sum_{i=0}^{2} (y(i) \cdot p(i)) = \frac{1}{6} \cdot \frac{6}{30} + \frac{2}{6} \cdot \frac{18}{30} + \frac{3}{6} \cdot \frac{6}{30} \approx 33.33\%. \tag{3.4}$$

3.3.2 Matching to a Repository of Wafers

In this section, we consider the case in which a top wafer selected out of a repository of k top wafers is stacked onto a given bottom wafer with f_b faulty dies. These top wafers are all assumed to have an equal number f_t of faulty dies per wafer, but the wafer maps of the individual top wafers can be different. This setting allows to improve the yield, by selecting *that* top wafer out of the repository for which its wafer map best matches the wafer map of the bottom wafer.

We generalize the probability function $p(i)$ from Section 3.3.1 into a function $p(i,k)$ which denotes the probability of occurrence of exactly i matching faulty dies, given a repository of k top wafers. Note that $p(i) \equiv p(i,1)$. Let $q(i) = \sum_{j=0}^{i} (p(j))$ denote the probability of at most i matching faulty dies. Then, the generalized probability function can be expressed as follows.

$$p(i,k) = q(i)^k - q(i-1)^k \qquad \text{for } i > 0 \tag{3.5}$$

$$p(i,k) = q(0)^k \qquad \text{for } i = 0 \tag{3.6}$$

For our running example, for a repository size $k = 5$:

$$p(0,5) = q(0)^5 = 0.2^5 = 0.00032$$
$$p(1,5) = q(1)^5 - q(0)^5 = 0.8^5 - 0.2^5 = 0.32736$$
$$p(2,5) = q(2)^5 - q(1)^5 = 1^5 - 0.8^5 = 0.67232$$

Table 3.1 shows how an increasing repository size k increases the probability of matching a larger number of faulty dies in both wafers.

$\boldsymbol{p(i,k)}$	$\boldsymbol{k=1}$	$\boldsymbol{k=3}$	$\boldsymbol{k=5}$	$\boldsymbol{k=8}$
$i=0$	0.2	0.008	0.00032	0.00000256
$i=1$	0.6	0.504	0.32736	0.16776960
$i=2$	0.2	0.488	0.67232	0.83222784

Table 3.1: Larger repository size k improves the probability of more faulty die matches i.

We define the expected yield $Y(k)$ for a repository of k top wafers as follows:

$$Y(k) = \sum_{i=0}^{f_t} (y(i) \cdot p(i,k)) \tag{3.7}$$

For the running example, the expected yield increases from 33.33% for a single-wafer repository to the following value for $k = 5$:

$$\begin{aligned} Y = Y(1) &= \sum_{i=0}^{2} (y(i) \cdot p(i,1)) \\ &= \frac{1}{6} \cdot p(0,1) + \frac{2}{6} \cdot p(1,1) + \frac{3}{6} \cdot p(2,1) \approx 33.33\% Y(5) \quad = \sum_{i=0}^{2} (y(i) \cdot p(i,5)) \\ &= \frac{1}{6} \cdot p(0,5) + \frac{2}{6} \cdot p(1,5) + \frac{3}{6} \cdot p(2,5) \approx 44.73\% \end{aligned}$$

3.3.3 Wafer Stacks With More Than Two Tiers

We now consider stacks of n tiers. We assume that the stacking is done in an iterative fashion. The first two tiers are stacked first. Each stack result serves again as the bottom in the stacking process with the next tier. Every wafer that is stacked onto a bottom wafer or bottom stack is selected from a repository of k wafers which have been pre-tested and for which the wafer map with good and faulty dies is known. Each wafer contains d dies. A wafer at tierl j ($1 \leq j \leq n$) has f_j faulty dies.

We approximate the expected yield $Y(n,k)$ for a stack of n tiers ($n \geq 2$) and a repository size of k wafers in a recursive way as follows:

- Base : $Y(2,k) = \sum_{i=0}^{f_t} (y(i) \cdot p(i,k))$ (3.8)

  where $f_t = f_2$ and $f_b = f_1$ (3.9)

- Step : $Y(n,k) = \sum_{i=0}^{f_t} (y(i) \cdot p(i,k))$ (3.10)

  where $f_t = f_n$ and $f_b = \lfloor (1 - Y(n-1,k)) \cdot d + 0.5 \rfloor$ (3.11)

Do note, that all though f_b and f_t are not part of the $Y(2,k)$ and $Y(n,k)$ functions directly, they are part of the $(y(i)$ and $(p(i)$ which ultimately do influence $Y(2,k)$ and $Y(n,k)$.

3.3.4 Implementation

To evaluate this model we have implemented it in Visual Basic for Applications (VBA), using the results directly in Microsoft Excel 2007. At the core of the model is the probability of the occurrence of exactly i matching faulty dies, as given in Equation (3.2), consisting of three binomial coefficients. These are relative easy to solve for a small values of d, f_b, and f_t. However, trying to calculate the binomial coefficients with bigger realistic values will very quickly result in VBA being unable to handle the extremely large numbers. VBAs Double datatype can only hold values between E^{-324} and E^{308} in the positive range. The upper limit of this range is already reached with a mere 175 dies on a wafer.

We can rewrite the binomial coefficients of Equation (3.12) into a set of divisions as shown below, until we end up with a set like Equation (3.14).

$$p(i) = \binom{f_b}{i} \cdot \binom{d-f_b}{f_t-i} / \binom{d}{f_t} \tag{3.12}$$

$$= \frac{f_b!}{i! \cdot (f_b-i)!} \cdot \frac{(d-f_b)!}{(f_t-1)! \cdot (d-f_b-f_t+i)!} \cdot \frac{f_t! \cdot (d-f_t)!}{d!} \tag{3.13}$$

$$= \frac{f_t!}{i! \cdot (f_t-i)!} \cdot \frac{f_b!}{(f_b-i)!} \cdot \frac{(d-f_b)!}{(d-f_b-f_t+i)!} \cdot \frac{(d-f_t)!}{d!} \tag{3.14}$$

Whenever possible these products of sequences have been implemented as loops. As a division in the form of $\frac{x!}{(x-C)!}$ is equal to the product of sequence $(x-C+1)\times(x-C+2)\times\ldots\times(X)$. Looking closely at the new termed equations it is clear that very large numbers will arise both in the numerator as well as the denominator. In this we have found a solution for the problem handling large numbers VBA has, if both the numerator and denominator are very large numbers then we can divide both by the same term, reducing their size without affecting the result of the division itself.

The two biggest contributors are the $\frac{(d-f_b)!}{(d-f_b-d_t+i)!}$ factor and the $\frac{(d-f_t)!}{d!}$ factor in the numerator and denominator of 3.14respectively. In order to let both the numerator and denominator grow as equally as possible, we alternate performing calculation steps on these two. In both cases the length of the sequence is dependent on the number of faults in the top wafer. By knowing how long these sequences are, we can determine when and how many times we can safely alternate between the two sequences. Meanwhile we keep checking the size of the numerator and denominator, and take appropriate action if they do become too large. With $p(j)$ now computable, we use Equation $q(i) = \sum_{j=0}^{i}(p(j))$, to calculate all possible values of $Q(i)$ Subsequently Equations (3.5) and (3.6) are used to calculate the of likelihood of an number of matches i for a repository size k, given the likelihood of a repository size of 1 (Equation (3.2)). The last step is the relative simple summation of Equation (3.7), with the $y(i)$ from Equation (3.1), this gives us the expected yield results presented in the next section.

3.3.5 Calculated Results

For the experimental results throughout this chapter, we consider a *reference process* with 300 mm wafers with 3 mm edge clearance, a defect density $d_0 = 0.5$ defects/cm^2 and a defect clustering parameter $\alpha = 0.5$ [8]. Our *reference design* consists of a two-tier stack of equal-sized square dies with area $A = 50$ mm^2. These inputs give $d = 1278$ dies per wafer [9], and a die yield $y = (1 + A \cdot d_0/\alpha)^{-\alpha} = 81.65\%$ [4], which roughly corresponds to $f_j = 235$ for all tiers j.

In Figure 3.3, the curves show the relative increase factor of the calculated expected stacked yield, based on the model presented earlier in this section, for our reference process and reference design, for varying repository sizes $k \in \{1,\ldots,50\}$ and various wafer yields 3 $y \in \{70\%,\ldots,90\%\}$. At the right-hand side of the figure, the absolute stack yields for $k = 1$

(minimum) and $k = 50$ (maximum) are given. The figure shows that significant yield increases of several percent points can be obtained for sufficiently large repository sizes. The higher the wafer yield is for each tier, the lower the compound stack yield is, but the larger the relative yield benefit due to wafer matching for a given repository size. The figure also shows that the incremental stacked yield benefit of wafer matching diminishes for larger repository sizes. Hence, the wafer repositories do not need to be large to harvest a major share of yield improvement offered by wafer matching.

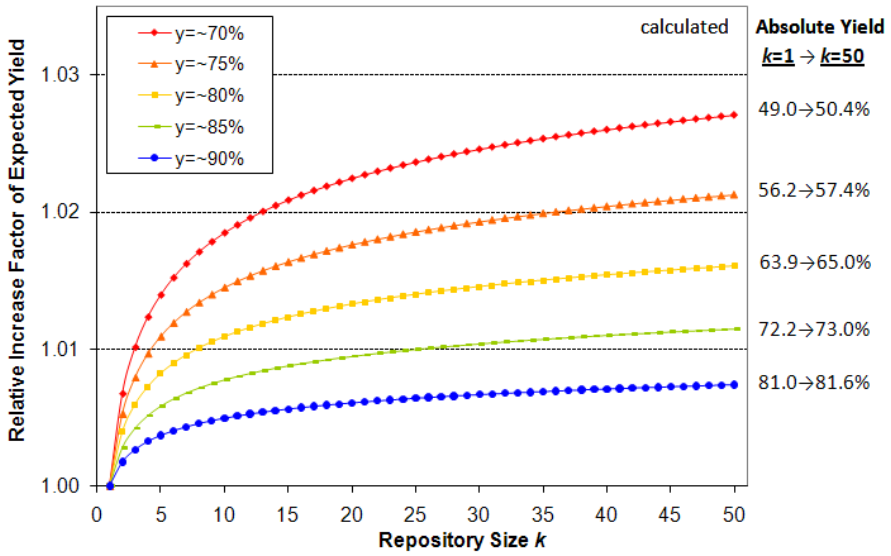


Figure 3.3: Relative increase factor of calculated expected stack yield for varying wafer repository size k and for various wafer yields.

In a second experiment, we vary the die size A. We keep the process maturity of the wafer fab constant, which implies that the yield per die decreases for larger dies. Table 3.3 shows, for the die sizes we selected, the resulting d and f values which served as inputs for our mathmatical model. The results of this calculation are depicted in Figure 3.4.

A	y [4]	d [9]	f	$(d-f)/d$
25 mm^2	89.44%	2597	274	89.45%
50 mm^2	81.65%	1278	235	81.61%
75 mm^2	75.59%	838	205	75.54%
100 mm^2	70.71%	622	182	70.74%
125 mm^2	66.67%	491	164	66.60%

Table 3.2: Calculation of d and f parameters, that served as inputs for our second simulation experiment.

In a third experiment we have look at the effect of multiple tiers. Again we used our reference process, for which we vary the repository size $k \in \{1, \ldots, 50\}$ and the number of tiers $n \in$

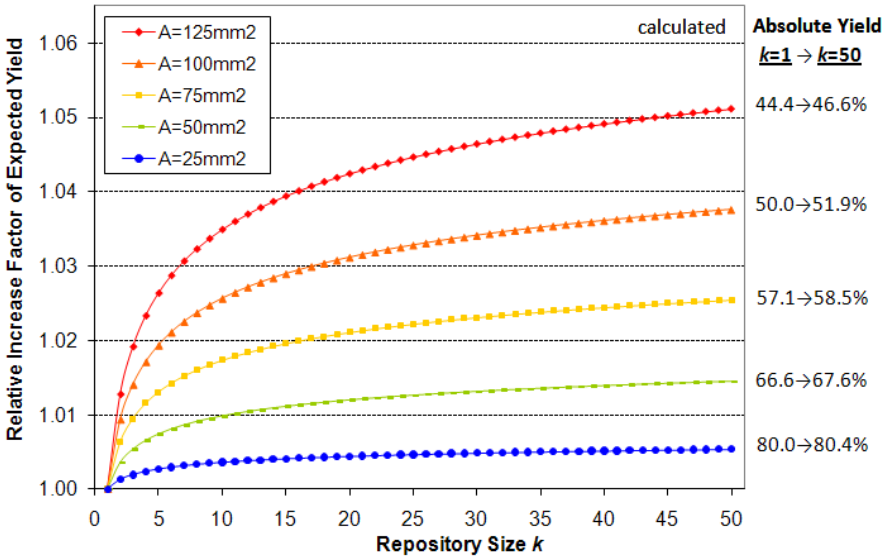


Figure 3.4: Relative increase factor of calculated expected stack yield for a varying wafer repository size k and for various die sizes.

$\{2, \ldots, 6\}$.

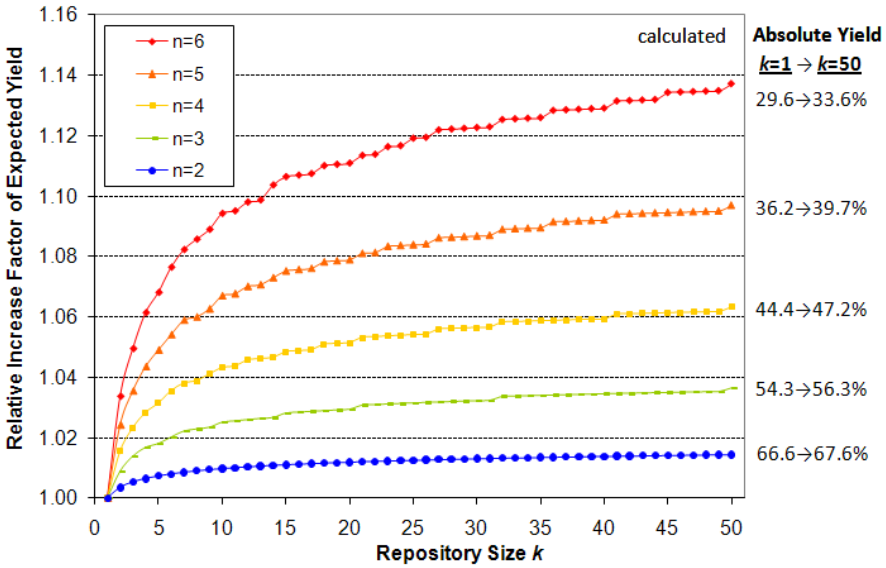


Figure 3.5: Relative increase factor of calculated expected stack yield for varying wafer repository size k and for various numbers of stack tiers.

We see very similar behavior in all three of our experiments, the majority of the gain W2W matching provides is achieved in the left hand side of the graphs. Further increasing the repos-

itory size is beneficial, but the gain per additional repository space is ever decreasing. Also we consistently find that the lowest base stacked yield, benefits more from the W2W matching compared to higher stacked yields.

The theoretical model presented in this section has a number of limitations. First and for most the the model assumes that a single bottom wafer is stacked to the best matching wafer out of k top wafers, while the remaining top wafers are discarded; in reality, entire cassettes of wafers need to be matched to each other, as we want to use them all, and not discard any wafers. another limitation is that the model assumes a fixed number of faulty dies f_j per stack tier; in reality, there will be variations in that number. A final limitation of the model, is that fact that it can only work with natural values. This raises a problem when we consider a stack size greater then two. For we use the intermediate expected stacked yield of the first two tiers to form a bottom wafer to match with a third. We therefor have to take the intermediate expected stacked yield and round to the nearest f (see Eq. (3.11)); this assumption is responsible for the clearly visible step behavior in Figure 3.5, which becomes more pronounced for larger values of n. These limitations are addressed in our simulation model presented in the next section.

3.4 Simulation

To assess the yield benefit of wafer matching through simulation, we have built two software solutions: (1) a wafer map generator, and (2) a wafer matching algorithm.

In order to perform wafer matching a matching algorithm requires wafer maps. In an actual industrial application, the wafer maps would be based on actual test results [10, 11, **?**]. In our case these are provided by a wafer map generator. This wafer map generator mimics real life wafer maps for simulation purposes, by generating pass/fail wafer maps using the internal C++ pseudo-random generator function and the fraction $(d - f)/d$ to determine whether the die was assumed to pass or fail its pre-bond test. In our system, wafer maps are represented as binary arrays of length d, the number of dies per wafer; a value of zero (resp. one) represents that the corresponding die on the wafer passed (resp. failed) its pre-bond test. This set-up guarantees that over a long run of generated wafer maps, the average die yield is indeed $(d - f)/d$, but allows for random variations per wafer, just as is the case for real-life wafer maps. To eliminate the bias a individual randomly generated wafer maps might have, experiments are executed 10,000 times and averaged.

The wafer matching algorithm we have implemented matches the entire content of n wafer repositories of size k to each other; in other words, no wafers are scrapped. The algorithm iteratively matches wafers, tier by tier. The algorithm first fully matches the Tier 1 and Tier 2 wafer repositories; subsequently, the Tier 3 repository is matched to the repository of Tiers 1+2 stacks; and so on, until Tier n is also matched and stacked. In matching a repository of k new top wafers to a repository of k bottom wafers or wafer stacks, we use the following algorithm. All k^2 combinations are evaluated and the highest-yielding match is selected. Subsequently, we consider the remaining repositories of size $k - 1$, repeat the procedure, and so on. Our matching algorithm is an iterative version of the ‘greedy’ heuristic for only two tiers as described in [2]. Note that this software component can also be used to match real-life wafer maps, instead of

the pseudo-randomly generated ones used in our experiments. Also, our matching algorithm can be replaced by another matching algorithm, for example the heuristic or exhaustive algorithms described in [2].

3.4.1 Simulator

The generalized flow through the simulator is depicted in Figure 3.6.

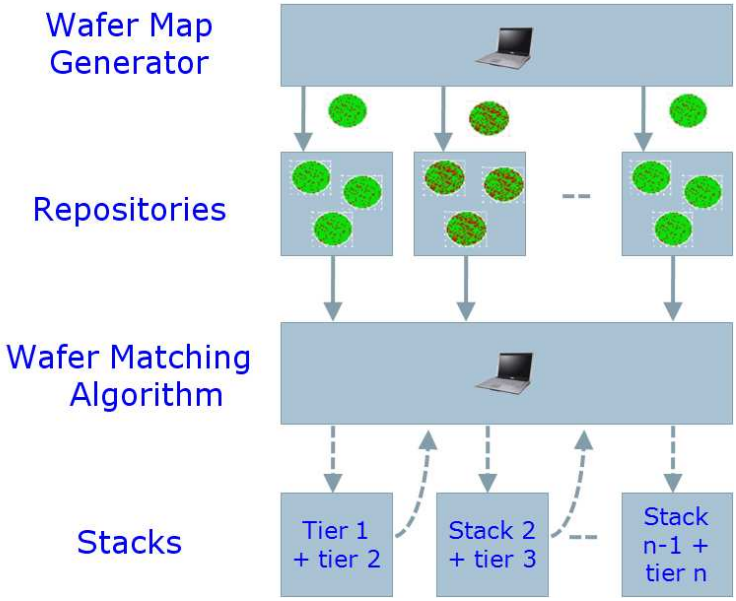


Figure 3.6: Operational flow of the W2W simulator.

A simulation starts with a user defining the experimental parameters, using a Graphical User Interface, see Figure 3.7. As the mathematical model in the previous section has shown, the effectiveness of W2W matching is dependent on four input parameters:

- the number n of stack tiers
- the number d of dies per wafer
- the number f_j of faulty dies per wafer (for $1 \leq j \leq n$)
- the wafer repository size k

In addition to these parameters, we also include settings for the number of simulations we want to run. Shorter simulation runs were used to verify functionality of the simulator, and confirm correct operation. Longer simulations provide us with statistically stable results we require for analysis.

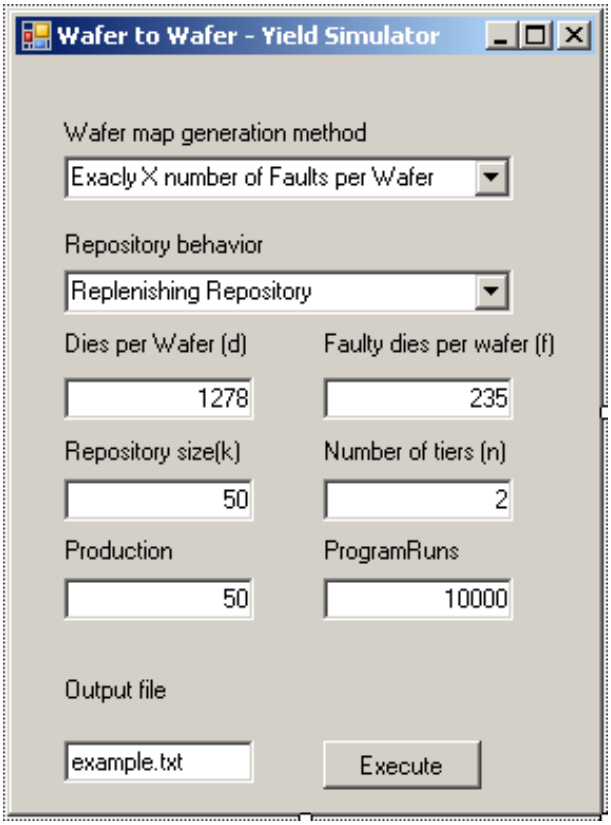


Figure 3.7: Graphical User Interface of the W2W simulator.

The GUI also provides settings that specify how the wafer map generator functions; (1) a set number of faulty dies randomly dispersed over the wafer map, leading to exactly f_j faults on a wafer, or (2) every die on the wafer map has a fraction $(d - f)/d$ chance of being faulty. Function (2) also leads to the same average yield over longer simulation runs as function (1) does, but allows for individual wafers to have lower and higher yields. This is one of the few settings that allows us to choose between the two main functions of the Simulator. In one mode it can be used to verify the mathematical model, while in the other it allows us to mimic real life conditions more closely. Other GUI settings include the number of stacks the are produced per simulation ('production'), this enables the simulator to discard $k - 1$ wafers in the repositories after the first match has been selected, this behavior is identical to that of our mathematical model. The simulator GUI also allows us to specify the name of the output file, that will hold the simulation results.

There are two main data structures, one that holds all the wafer repositories, and one that holds repositories with the (partial) stacks as compound wafers. Both are implemented as three-dimensional arrays of the form[a][b][c]: the first position denotes the repository, the second the position of a wafer in the repository, and the third the die on that wafer.

Algorithm 1 shows the steps the simulator goes through for a single simulation, we will look at the individual steps later on.

Algorithm 1 [YIELD SIMULATOR, SINGLE SIMULATION]

```
1  INITIALIZE ALL REPOSITORIES;
2  COMPUTE POSSIBLE MATCHINGS;
3  SELECT BEST MATCHINGS;
4  MEASURE RESULTING STACK;
```

In Algorithm 2 we initialize the repositories, this includes the generation of the wafer maps for each of the wafers in each of the repository. We will only concern ourselves with the 'waferstock' array which holds the repositories of wafers, the 'stack' array with partial stack is initialized but remains empty until the bonding procedures take place.

Algorithm 2 [INITIALIZATION OF THE REPOSITORIES]

```
1  initialize array
2  /* Step through the repositories */
3  for n := 1 to #repositories {
4      /* Step through the wafers */
5      for k := 1 to #wafers_per_repository {
6          if Algorithm == variable_number_of_Faults {
7              /* Step through the dies */
8              for j := 1 to #dies_ per_ wafer {
9                  if (d * randomvalue[0.00..1.00]) > f_i {
10                     waferrepositories[n][k][j] = 0; /* good die */
11                 }
12                 else {
13                     waferrepositories[n][k][j] = 1; /* faulty die */
14                 }
15             }
16         }
17         else (Algorithm == fixed number of Faults) {
18             while #faults not reached do {
19                 waferrepositories init to 0;
19                 int die = (randomvalue[0.00..1.00] * d);
20                 if waferrepositories[n][k][die]] =! 1 {
21                     waferrepositories[n][k][die]] = 1;
22                     #faults++;
23                 }
24             }
25         }
26     }
27 }
```

We only consider the matching of two repositories of wafers at the time, one repository is the provider of the bottom wafers, the other of the top wafers. In Algorithm 3 we add all possible k^2 combinations of a top and bottom wafer to a Linked list called 'matchlist'. 'Matchlist' is responsible for ordering these matchings by the stacked yield, which results from bonding those two wafers. After the first combination has been chosen, there are $k-1$ bottom wafers and $k-1$ top wafers left, leading to $(k-1)^2$ possible combinations for the second combination to be selected from.

These $(k-1)^2$ combinations are a subset from the original k^2 combination, as all $k-1$ bottom and top wafers where part of the two sets of k wafers. Realizing this we have made the choice of not repeating the calculations bet reuse the result from the original k^2 matrix of calculations, instead we remove the invalid entries from this matrix after a particular match is selected. This is why we use a Linked List, a Linked list can easily hold all the combinations of the matrix, keep them in order by resulting yield, and the removal of invalid combinations is performed by traveling through the list only once.

Algorithm 3 [COMPUTE ALL POSSIBLE MATCHINGS]

```
1  initialize matchlist
2  /* Step through the bottom wafers */
3  for h := 1 to k {
4      /* Step through the top wafers */
5      for j := 1 to k {
6          int score = WaferPairScore(waferrepositories[bottom][h], waferrepositories[top][j]);
7          matchlist.add(h,j,score);
8      };
9  };
```

When all possible matchings have been added to the matchlist, a new phase can begin. As matchlist kept the matchings in order, the best yielding combination will be at the head of the linked list. This best matching is removed for the list, and the two involved wafers are bonded and stored in the 'stack' array. Algorithm 4 shows how this is performed. Once the stack had been created it can be used as a source for a new matching process together with a new top wafer from one of the 'wafer stock' repositories, repeating Algorithm 3. Alternatively it can be a final product and be tested as Algorithm 5 describes.

Algorithm 4 [SELECT BEST MATCHINGS]

```
1  /* continue selecting until all wafers are used */
2  for i := 1 to #wafers_per_repository {
3      match = matchlist.retrieve();
4      new_stack = bond(match.top, match.bottom);
5      stacks[partial_stack][i] = new_stack;
6  };
```

Algorithm 5 [MEASURE RESULTING STACKS]

```
1  /* Count working dies until all wafers are checked */
2  for i := 1 to #wafers_per_repository {
3      wafer_yield = WaferScore(stacks[#tiers][i]);
4      total_yield = total_yield + wafer_yield;
5  };
6  yield_percentage = total_yield/#wafers_per_repository;
```

3.4.2 Matchlist

Matchlist is the data structure that holds all the results of all possible matchings between two repositories. For a given repository size of k wafers, k^2 scores will be added to matchlist. These entries will have four parameters, the first two denoting the two wafers in the matching, the third is ”Score” or the number of good dies that would result, and the final variable is a pointer to the next matching in the list. Matchlist initializes as an empty linked list after which all the scores will be added one by one, while keeping the list ordered, the first best scores are kept at the head of the list, while the worst scores are at the tail of the list. How this is performed is made clear in Algorithm 6

Algorithm 6 [MATCHLIST.ADD]

```
1   /* add(int bottomwafer, int topwafer, int score) */
2   matchlist new_item = new Matchlist(bottomwafer, topwafer, score);
3   *Previous = *Head;
4   *Current = *Head;
5   while Current.score > new_item.score do {
6         *Previous = *Current;
7         *Current = *Current.next();
8   }
9   *Previous.next = *new_item;
10  *new_item.next = *Current;
```

When the linked list is completed, the so called ‘Retrieve_procedure’ is used. This Algorithm in 7 returns the head of the linked list (the first highest score), and stores the two wafers used in that matching. Subsequently all entries of the list that share one of both of these wafers with the head in the queue are invalid and are removed from the linked list.

Algorithm 7 [MATCHLIST.RETRIEVE]

```
1   /* Retrieve() */
2   int bottomwafer = Head.bottomwafer;
3   int topwafer = Head.topwafer;
4   *Previous = *Head;
5   *Current = *Head;
6   while Current.next =! null do {
7         if Current.topwafer == topwafer & Current.bottomwafer == bottomwafer {
8               *Previous.next = *Current.Next;
9               *Current = *Previous.next;
10        }
11        else{
12              *Current = *Current.next();
13        }
14  return Head;
15  Head = Head.Next()
16  };
```

For each of our experiments we have done 10,000 simulations, and using repository sizes k ranging for 1 to 50. To this end each simulation run, initiated by the GUI, will loop through Algorithm 1 500000 times. In in the case for a stack consisting of more then two tiers, multiple loops per simulation through Algorithm 3 and Algorithm 4.

3.4.3 Experimental Results

In this section we present three simulation experiments, performed with the simulator described in the previous section. The experiments are based on our reference process and reference design (see Section 3.3.5); in each experiment we vary repository size k and one of three other input parameter: (1) the number f_j of faulty dies per wafer (for $1 \leq j \leq n$) , (2) the number d of dies per wafer, (3) the number n of stack tiers.

In the first simulation experiment, we vary f while maintaining the die size A constant; this leads to variations in the die yield $y = (d - f)/d$. This experiment simulates the effect of varying process maturity of the wafer fab. Our reference design in our reference process has a yield of ∼82%. We have experimented with approximate die yields of 50% ($f = 639$), 60% ($f = 511$), 70% ($f = 383$), 80% ($f = 256$), and 90% ($f = 128$). The results are depicted in Figure 3.8.

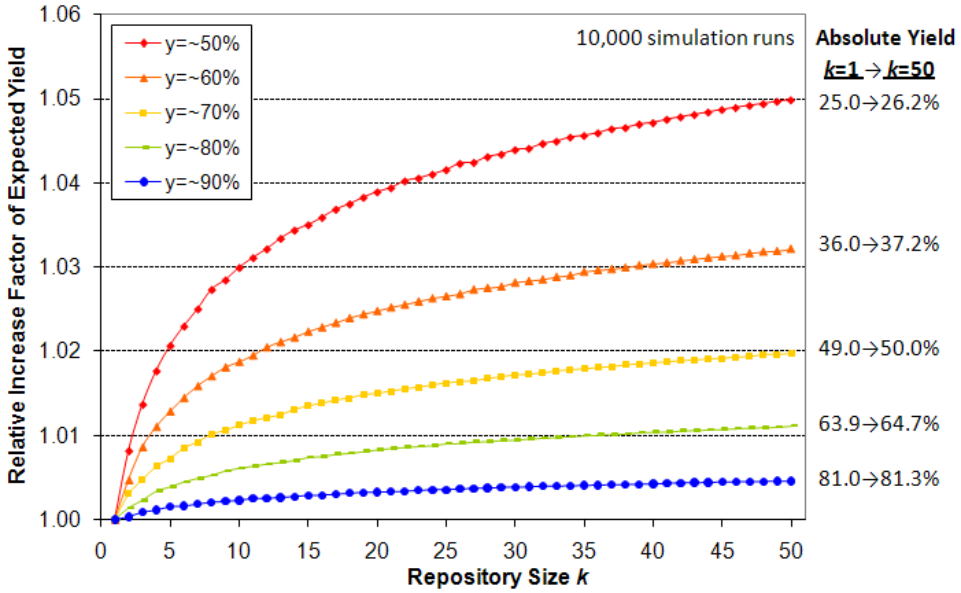


Figure 3.8: Relative increase factor of simulated expected stack yield for varying wafer repository size k for various die yields.

In the second simulation experiment, we vary the die size A. We keep the process maturity of the wafer fab constant, which implies that the yield per die decreases for larger dies. Table 3.3

shows, for the selected die sizes, the resulting d and f values which served as inputs for our simulations. The simulation results are depicted in Figure 3.9.

A	y [4]	d [9]	f	$(d-f)/d$
25 mm^2	89.44%	2597	274	89.45%
50 mm^2	81.65%	1278	235	81.61%
75 mm^2	75.59%	838	205	75.54%
100 mm^2	70.71%	622	182	70.74%
125 mm^2	66.67%	491	164	66.60%

Table 3.3: Calculation of d and f parameters, that served as inputs for our second simulation experiment.

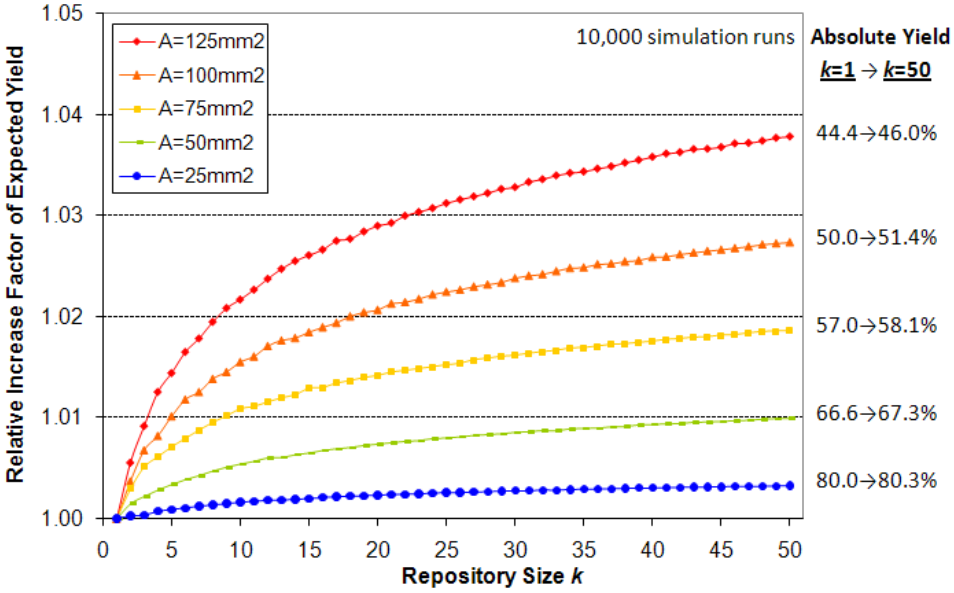


Figure 3.9: Relative increase factor of simulated expected stack yield for varying wafer repository size k for various die areas.

In the third simulation experiment, we use the reference process and reference design and vary the number of tiers $n \in \{2, \ldots, 6\}$. The simulation results are shown in Figure 3.10.

All three experiments lead to similar observations. The absolute compound stack yield of W2W stacking decreases for increasing number of tiers, decreasing die yield, and increasing die area; in some cases drastically. Hence, if the number of tiers is large, the die yield is low, and/or the die area is large, it is rather unattractive to do W2W stacking. However, the lower the compound stack yield, the higher the relative yield increase that can be obtained through wafer matching. In our experiments, the observed relative yield increase varies between 0.5% and 10%.

We also repeated the experiment in Smith et a. l[1] with our simulator. Settings for this experiment included 1320 dies per wafer, 25 wafers per repository, multiple stack heights, and a

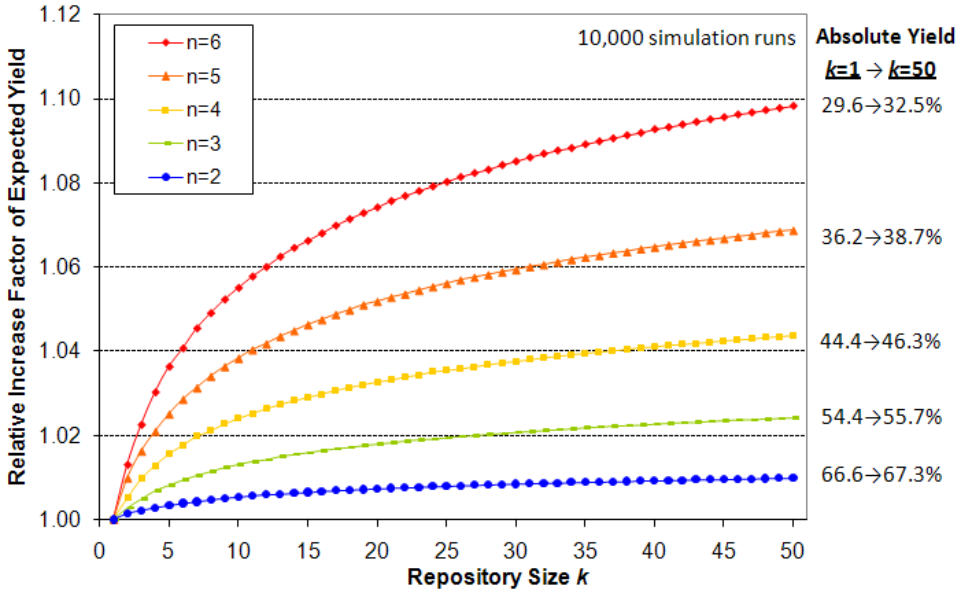


Figure 3.10: Relative increase factor of simulated expected stack yield for varying wafer repository size k for various numbers of stack tiers.

varying die yield of 5% to 95%. In the experiment this is signified as 1254 to 66 faulty dies in steps of 5% or 66 faulty dies. The result of this experiment is shown in Figure 3.11.

From this experiment we see the benefit of performing a large number of simulations, as it shows how the benefit of W2W matching is effected by the yield of the resulting stack. The smoothness of the curves mean that realistic estimates can be made even for repository sizes outside our scope by extrapolation. In Figure 3.11 we see that there is an optimum, which is to be found near a stack yield of 20%, meaning that that for $n = 2$, the yield of the individual tiers is much lower then would be the case for $n = 6$. Also note that we are not manipulating the stack yield directly, instead it is controlled indirectly by manipulating the tier yield.

3.5 Cost Benefit Analysis

The previous sections have demonstrated the benefit of the wafer matching approach in W2W stacking in terms of yield increase. However, wafer matching is only possible if pre-bond test results are available for all dies. In case the wafer fabrication and stacking is performed by different companies, and the wafers are delivered to the stacking company under a *Known-Good Die* (KGD) agreement, die testing will be necessary to guarantee the outgoing product quality of the wafer fab. In other cases, the question is whether the additional costs of performing pre-bond die tests are compensated by the obtained yield increases [1, 2].

We consider the three production flows [12] in Figure 3.12. All three flows have a final test which re-tests dies and interconnects, and hence result in products with the same final test quality. The

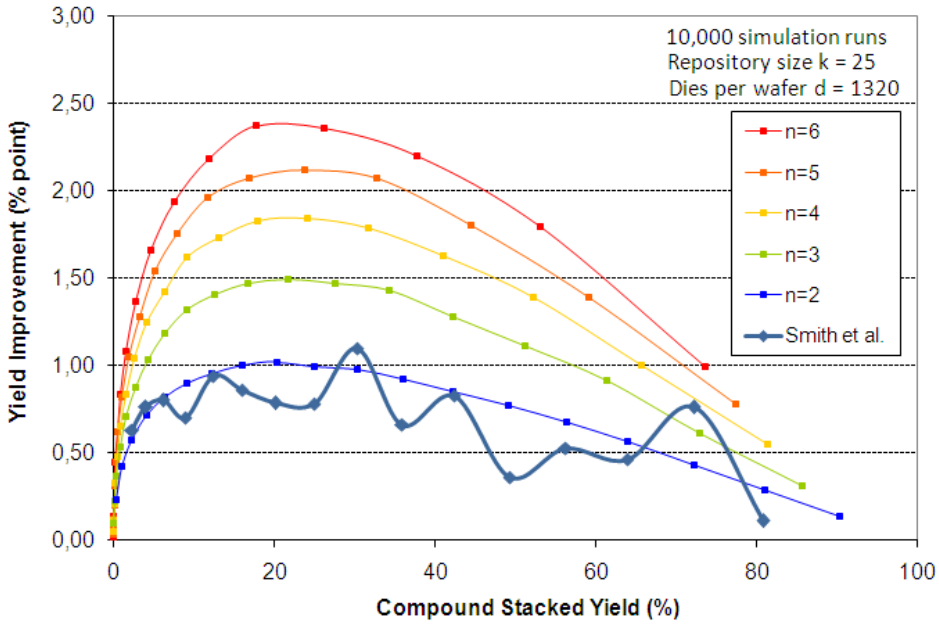


Figure 3.11: A comparison to Smith et al. [1]; Yield improvement, as a function of the stack yield.

differences are in the earlier parts of the flows. Flow (a) is the base flow for W2W stacking: dies and interconnect are tested for the first time only *after* stacking. Flows (b) and (c) both make use of wafer matching, enjoy the corresponding increased yield benefit, but require pre-bond die testing. In Flow (b), the stack is fully tested after stacking but before assembly and packaging, i.e., the newly-formed interconnects between the stacked dies are tested, but also the dies themselves are re-tested. The die re-tests might show limited fall-out, e.g., due to intra-die defects induced by wafer thinning [12]. Flow (c) is an optimized variant of (b), in which during the stack test, only the newly-formed die interconnects are tested. If present, intra-die defects will slip through this stack test, but are assumed to be caught by the final test, hence maintaining a constant quality of the final product but increasing the packaging cost.

Function $t(x)$ denotes the test costs for Flow (x), with $x \in \{a, b, c\}$. $t(x)$ can be expressed as a sum of products. The products express how many items are tested and what the test costs per item are. The sum iterates over the various test stages that are executed: pre-bond die test, post-bond stack test, and final test. In addition to the variables from Section 3.3, we introduce four new variables: t_{die} and t_{int} denote respectively the test costs (in arbitrary units (a.u.), we use this only for relative comparison) for a die test and an interconnect test; and ys_{die} and ys_{int} denote respectively the die and interconnect pass yield during the stack test. We can express $t(x)$

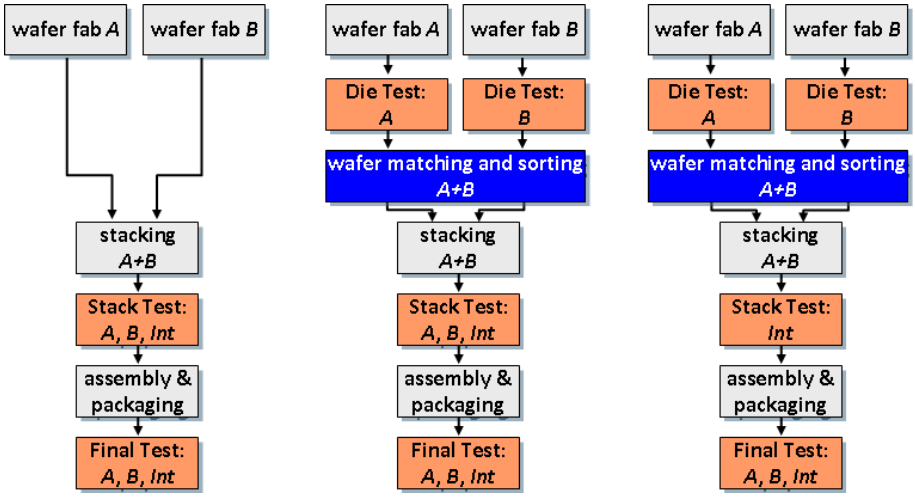


Figure 3.12: Production flows: (a) base flow (without wafer matching), (b) wafer matching flow, (c) optimized wafer matching flow.

now as follows:

$$
\begin{aligned}
& & \text{number of items tested} & & \text{test cost per item} & \\
t(a) = \; & & d & & \cdot (n \cdot t_{\text{die}} + (n-1) \cdot t_{\text{int}}) & \\
& + & d \cdot Y(n,1) \cdot ys_{\text{die}}^{n} \cdot ys_{\text{int}}^{n-1} & & \cdot (n \cdot t_{\text{die}} + (n-1) \cdot t_{\text{int}}) & (3.15) \\
t(b) = \; & & d & & \cdot n \cdot t_{\text{die}} & \\
& + & d \cdot Y(n,k) & & \cdot (n \cdot t_{\text{die}} + (n-1) \cdot t_{\text{int}}) & \\
& + & d \cdot Y(n,k) \cdot ys_{\text{die}}^{n} \cdot ys_{\text{int}}^{n-1} & & \cdot (n \cdot t_{\text{die}} + (n-1) \cdot t_{\text{int}}) & (3.16) \\
t(c) = \; & & d & & \cdot n \cdot t_{\text{die}} & \\
& + & d \cdot Y(n,k) & & \cdot (n-1) \cdot t_{\text{int}} & \\
& + & d \cdot Y(n,k) \cdot ys_{\text{int}}^{n-1} & & \cdot (n \cdot t_{\text{die}} + (n-1) \cdot t_{\text{int}}) & (3.17)
\end{aligned}
$$

For a fair comparison of the test cost per flow, we attribute the total test costs only to the functional (passing) stacks. For Flow (a), the number of functional stacks is $d \cdot Y(n,1) \cdot ys_{\text{die}}^{n} \cdot ys_{\text{int}}^{n-1}$; for Flows (b) and (c), the number of functional stacks is $d \cdot Y(n,k) \cdot ys_{\text{die}}^{n} \cdot ys_{\text{int}}^{n-1}$.

In Figure 3.13 we compare the test costs increase (or decrease) per functional stack for the two wafer-matching flows (Flows (b) and (c)), relative to the base flow (Flow (a)). For these cost calculations, we made the following assumptions: $k = 50$, $t_{\text{die}} = 5,000$ a.u., $t_{\text{int}} = 50$ a.u.[2], $ys_{\text{die}} = 99\%$, $ys_{\text{int}} = 97\%$. The set-up of other parameters for the experiments in Figure 3.13 are equal to the three experiments of Section 3.4, in which we vary respectively yield, die size, and number of stack tiers.

At first glance, one would think that adding a pre-bond die test as a third test to a flow that already contains a post-bond stack test and a final test adds roughly 50% test costs per die. However,

[2]For our comparative study, the actual values of t_{die} and t_{int} are far less important than their ratio.

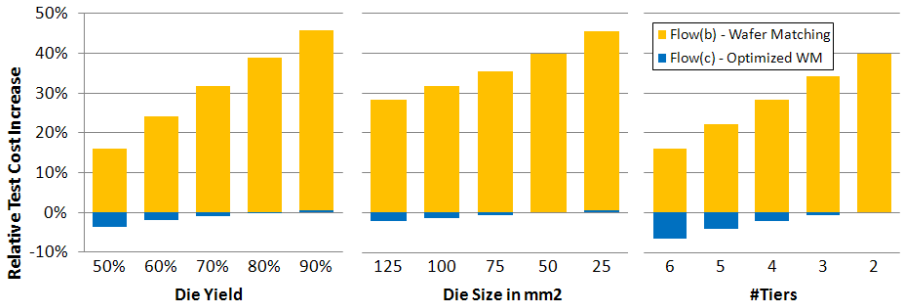


Figure 3.13: Relative test cost per functional stack of wafer-matching flows in comparison to the base W2W flow without wafer matching.

if the wafer maps resulting from the die test are kept, the stack test can simply skip stacks which contain one or more known bad dies, and hence reduce its test costs. This is confirmed by Figure 3.13, which shows that the test cost increase of Flow (b) varies between 15% and 50%. The lower the compound stack yield (due to low die yield, large die size, and/or a large number of tiers), the lower the additional test costs. If the compound stack yield is low, it is beneficial to find that out as early as possible through a pre-bond die test, as further (test) costs downstream can be prevented. Figure 3.13 also shows that the optimized wafer-matching flow (Flow (c)) more or less keeps the test costs per die constant, and in some cases even achieves a cost reduction up to 6.5%. The price to be paid for that is that due to the lower-quality stack test, some faulty stacks get packaged and are only detected as faulty during the final test. This applies to a fraction $(1 - ys_{\text{die}}^{n})$ of the functional stacks; in our example, for $n = 2$, this amounts to roughly 2% of the functional stacks.

3.6 Conclusion

The W2W stacking approach for 3D-SICs offers the highest manufacturing throughput and allows for the smallest die sizes, thinnest wafers, and highest TSV densities. The drawback of W2W stacking is that one cannot avoid that a bad die is stacked onto a good die or vice versa, leading to low compound yields. This drawback is exacerbated by a large number of stack tiers, a small number of (large) dies per wafer, and/or low die yield.

Nevertheless, if one chooses for a W2W approach, wafer matching on the basis of pre-tested wafer repositories can increase the compound yield significantly. Our mathematical yield model shows that the yield increase depends on (1) the number of stack tiers, (2) the number of dies per wafer, (3) the die yield, and (4) the wafer repository size. For reasonable repository sizes of 25 to 50 wafers (one or two cassettes), our simulations demonstrate that relative yield improvements of 0.5% to 10% can be achieved; the benefits are larger when the absolute compound yield is lower. Our experiment also has shown that the that the benefit gained by increasing the size of the repository k, is decreasing. Making small manageable repository sizes of 25 or 50 wafers attractive. The additional investment required to achieve this yield increase is either an relative test cost increase of 15-50% (for the wafer matching Flow (b)), or an additional package cost

of 2-5% (for the optimized wafer matching Flow (c)). This implies that wafer matching is an economically viable approach in the context of W2W stacking.

DfT Architecture for 3D Stacked ICs

4

This chapter describes a structured modular Design for Test (DfT) architecture for 3D-SICs. Without such an architecture testing of a 3D-SIC is only possible if all the contributing dies are designed in conjunction with each other, and only when manufacturing of the stack has been completed. With the modular 3D-DfT architecture we propose, it is possible to take any 3D-DfT enabled design and position in anywhere in the stack, and still perform testing of the entire stack and components. We base this 3D-DfT architecture on the IEEE 1500 [3] architecture for embedded core testing. The proposed 3D-DfT architecture allows for a 3D-SIC to be tested in between various stages of manufacture, as well as allow for final product testing. We also investigate the cost aspect of our 3D-DfT architecture in terms of additional TSV's and dedicated pre-bond probe pads.

This chapter is organized as follows. In Section 4.1, we provide the background to our work in this area, and identify where 2D testing techniques do not suffice in regard to 3D testing. Section 4.2 describes related prior work in test access architectures for 3D-SICs. Section 4.3 provides an overview of test access architecture standards for PCBs and 2D-SOCs, which, like 3D-SICs, are also built from interconnected smaller components. The test requirements and constraints which are unique to 3D-SICs are discussed in Section 4.4. Our proposed 3D-SIC test access architecture is presented in Section 4.5. In Section 4.6 we will validate the 3D-SIC test access architecture with simulation experiments with recommendations for tool vendors in Section 4.7. Section 4.8 concludes this chapter.

4.1 Introduction

Stacking layers of 2D ICs on top of each other to form 3D-SICs has major advantages; heterogeneous integration of dies, small physical size and footprint, and high internal connectivity. The advantages of 3D-SICs over 2D ICs are described in Chapter 2. However, it also brings forth problems in terms of test access; Getting test data to the selected tier of the 3D-SIC is not trivial. In contrast to 2D ICs testing, testing equipment may not have direct access to the die, as it forms a tier in a 3D stack, and other tiers physically prohibit that tier to be probed. Our 3D-DfT architecture solves the access problem for 3D-SICs by allowing scan access to a particular tier through the other tiers in the stack. In Figure 4.1 we show a stack of three dies, each die may have multiple cores or be monolithic in design. The 3D-DfT architecture should provide access to each die, from where 2D DfT can distribute the test data internally.

First and foremost 3D-SICs need to be testable to the same degree as 2D ICs; being unable to properly test 3D-SICs would make them economically inviable. The process of stacking only makes sense for bigger, more expensive dies, making any 3D-SIC valuable. Any customer

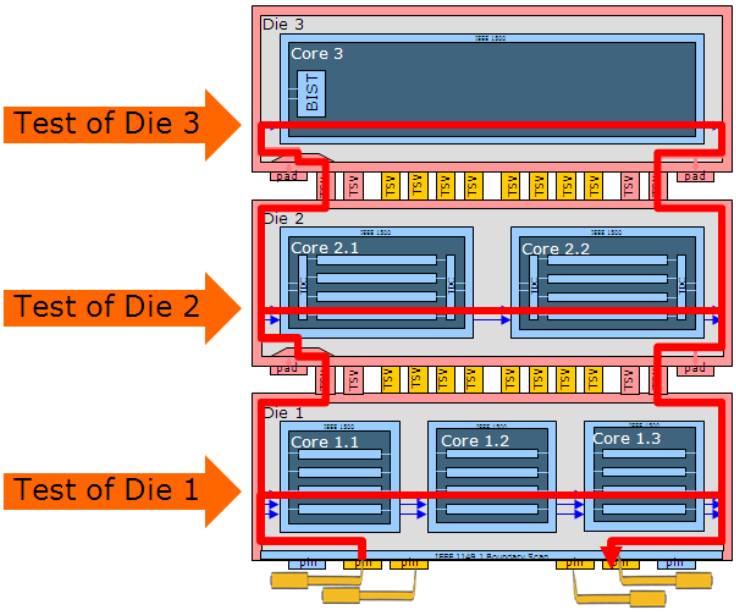


Figure 4.1: 3D-DfT architecture allows for the testing of any die, with only the bottom die accessible.

therefore will be expecting high quality fully tested products, especially for those markets where product failure is unacceptable (military, aeronautical, medical, etc).

Fundamentally, a 3D design can simply be seen as a System-on-Chip (SoC). A SoC of which the individual cores are distributed among the tiers. If a stack is seen as single SoC, the stack can be tested as a normal SoC and 2D-DfT would suffice. While approach does work for completed stacks it does not cater for pre-bond die tests or partial stack tests. In these tests, sections of the stack are simply not there yet, so they cannot be used to propagate test data through. This is inherited from 2D design, where all components are finished at the same time; there is no need to be able to test a individual core while the rest has not been build yet. In 3D-SICs we do want to perform intermediate testing; we want to know if a tier is functional before putting it through the expensive stacking procedure. Also if a non-functional tier is stacked onto a functional tier, the resulting stack would be unsalable. Hence, the costs associated with the introducing a non-functional tier into a stack is not just the costs of that tier and the stacking procedure but also the cost involved with the other tiers in that stack. To enable pre-bond die tests and partial stack tests, we therefore require an active DfT architecture that controls the flow of test data depending on the stacks composition.

We also want to take a look further ahead, and provide an architecture that allows for the reuse of tiers into other stacks. We see more and more reuse of Intellectual Property (IP), for instance when entire cores are reused in several designs, even amongst different companies. We foresee this phenomenon to continue and spread into reused tiers, where stacking companies make use of tiers from several companies in much the same way as commodity memories are bought in by system makers today. A standardized modular approach in the DfT architecture should help 3D-SICs in these matters; Using test-compatible tiers would mean a stack maker is able to test

the tiers without having to negotiate with each and every tier provider who propagates which signal and where to, further improving the re-usability of a tier.

In this chapter, we focus on the design of a structured and scalable test access architecture. The architecture supports pre-bond tier testing, post-bond stack testing, and final packaged-product testing. It allows for modular testing [13] of intra-tier circuitry and inter-tier TSV-based interconnects. Hence, it supports stacks of possibly heterogeneous and/or black-boxed dies, flexible test flow creation during the various maturity stages of a 3D-SIC product, and easy diagnosis. The architecture reuses commonly encountered DfT structures within the various tiers as much as possible. It leverages two existing DfT standards, viz. IEEE 1149.1 [14, 15] for chips on a Printed Circuit Board (PCB) and IEEE 1500 [3, 16] for embedded cores within an System-On-Chip (SOC). The test access architecture initially targets 3D-SICs consisting of scan-testable digital logic and memory tiers, but can be extended to include other types of tiers as well.

4.2 Related Prior Work

The first paper dedicated to testability of 3D-SICs was [17]. It focuses on pre-bond die testing, required to achieve acceptable compound stack yields. Testing incomplete products as formed by the various stack tiers is identified as a potential problem. In our opinion, this does *not* have to pose a problem really, provided that (1) structural, not functional tests are applied, (2) a modular test strategy is followed, and (3) the infrastructure (power/ground, clocks) can be made operational per die. [17] proposes a 'scan island' approach, which is essentially the wrapper technology from IEEE 1149.1 [14, 15] and IEEE 1500 [3, 16, 13] under a different name.

Many of the other publications on 3D-SIC testing implicitly propose a test access architecture, while focusing on optimizing the design parameters of that architecture to minimize the resulting test length and/or the associated wire length. Wu et al. [18] describe three scan chain optimization approaches for 3D-SICs. Implicitly, this paper assumes that a single logic test unit is partitioned over multiple tiers, which seems rather unrealistic. Therefore, in [19], Wu et al. propose a core-based design and test approach (as common for 2D-SOCs) in which each core resides on a single tier. The paper proposes an ILP-based Test Access Mechanism (TAM) optimization approach, which tries to minimize the resulting test length under a constraint for the number of additional 'test TSVs'. Both papers [18, 19] focus exclusively on *post-bond stack testing*, and ignore the requirements for pre-bond die testing.

Jiang et al. [20] describe a TAM optimization approach based on simulated annealing that minimizes test length and TAM wire length with a user-defined cost weight factor. They assume a modular core-based 3D-SIC test approach and take both pre-bond and post-bond test lengths into account. The paper lacks *realistic constraints* on wafer and packaged stack test access, due to which it unrealistically allows TAMs to start and end at any stack tier. Successor paper [21] remedies this partly, by working with *pre-bond tests* that are applied through *dedicated probe pads* at the die in question, for which a maximum count is assumed. The paper proposes heuristics that determine a post-bond stack test architecture, from which segments are reused as much as possible to build additional die-level test architectures for the pre-bond tests, while meeting

the maximum probe pad count constraint and minimizing test length and TAM wire length.

In contrast to the prior work by others, we start out by identifying *realistic constraints and requirements* set forward by, among others, *wafer probe technology* and *test flow set-ups*. Subsequently, we focus on the design of a generic and structured test access architecture. The architecture is scalable in the sense that its design parameters can be optimized for varying core, die, and stack parameters, but the focus of our work is *not* on those optimization procedures. The prior work published until now has focused emphonly on testing the cores in the various dies constituting the 3D-SIC. However, testing the circuitry within the tiers, beside the cores, has not been covered, and neither is the (TSV-based) inter-die interconnects between the tiers. The prior work also did not identify how existing DfT standards and test access architectures can be leveraged. Finally, test control and instructions were ignored in the prior work. We address all the above issues.

4.3 Related Test Access Standards

4.3.1 Test Access Architecture for PCBs

The commonly-used test access architecture for PCBs is based on IEEE Std. 1149.1, Boundary Scan (a.k.a. 'JTAG') [14, 15]. In order for chips to be compliant to IEEE 1149.1, a small hardware wrapper is added to them as shown in Figure 4.4. The instruction register (IR) is used to hold instructions which determine the flow of data through via the switch boxes. These switch boxes enable or disable internal scan chains, Boundary Scan Register (BSR) and the bypass register. The IR is controlled by a finite state machine, the TAP Controller, which gets its control signals externally. IEEE 1149.1 works through a narrow single-bit interface, as every JTAG terminal requires an additional chip pin and these are considered expensive. Fortunately, the prime focus of IEEE 1149.1 is PCB interconnect testing, and that requires only a small number of test patterns [22]. The single-bit interface pins are called TDI (Test Data Input) and TDO (Test Data Output), and they transport both instructions and test data. The control interface consists of the pins TCK (Test ClocK), TMS(Test Mode Select (and optionally TRSTN (Test ReSeT Nagetively triggered)).

For an example PCB containing three 1149.1 chips, a common JTAG-based test access architecture is depicted in Figure 4.3. The control signals are broadcast to all chips, while the TDI-TDO pins are concatenated through the chips. The broadcast control signals can configure the TAP Controller in a mode in which it is willing to receive instructions, which are subsequently scanned into the Instruction Register (IR) via the daisychained TDI-TDO interface. Note that this allows for different instructions for different chips; for example, Chip B can be configured in INTEST mode, while Chips A and C are configured in BYPASS mode. Then, the chips are brought into their instructed test modes via the broadcast control signals and test data is scanned in and out again via the daisychained TDI-TDO interface. The selected test data register (e.g., the bypass register, a Boundary Scan Register (BSR), or a chip-internal scan chain) depends on the instruction, and can be different for different chips; in any case, it is a single shift register, as shown in Figure 4.3.

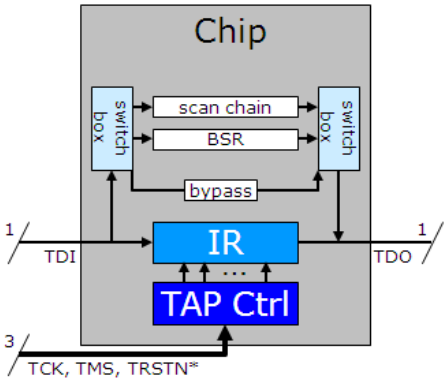


Figure 4.2: IEEE 1149.1 compliant chip.

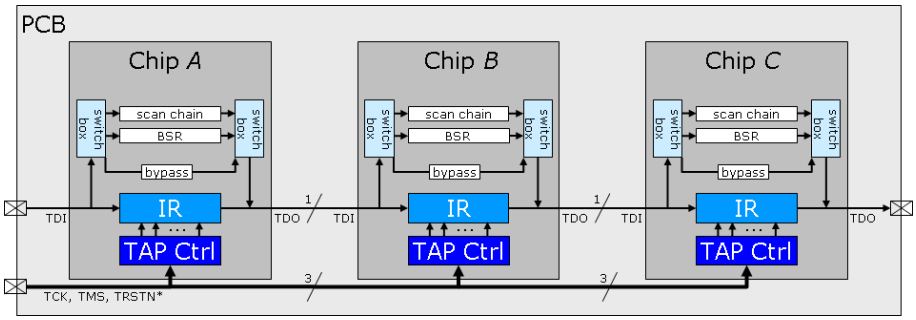


Figure 4.3: Board-level test access architecture for chips based on IEEE 1149.1.

4.3.2 Test Access Architecture for 2D-SOCs

The commonly-used test access architecture for (two-dimensional) SoCs containing embedded IP cores is based on IEEE Std. 1500 [3, 16, 13]. As shown in Figure 4.4 , IEEE 1500 adds a small hardware wrapper around the module-under-test. Similar to the IEEE 1149.1 wrapper, IEEE 1500 adds switch boxes that control the flow of data through the core, but it must cater for the n-bit parallel test data interface as well. Instead of TAP controller and IR, IEEE 1500 uses a Wrapper Instruction Register (WIR) which still has an IR internally but does not require a finite state machine. To provide control over the IR IEEE 1500 instead relies on the Wrapper Serial Control (WSC) signals directly; A faster solution then a IEEE 1149.1 TAP controller, but does require addition pins.

As shown in Figure 4.5, the test access architecture for an IEEE 1500-based SOC shows more similarities to IEEE 1149.1-based PCBs. Control signals are broadcast to all cores. Once configured in the appropriate mode, instructions are shifted into the Wrapper Instruction Register (WIR) via the daisychained WSI-WSO interface. The WSI-WSO interface can also be used as a single-bit test data interface, once the WIR has been put into the desired mode. For board-level compatibility we have linked the IEEE 1500 cores, to a SoC-level IEEE 1149.1 interface, from which we also derive the control signals. This is a typical case, where the IEEE 1149.1 interface is preferred amongst SoCs and other PCB components to enable board-level interconnect

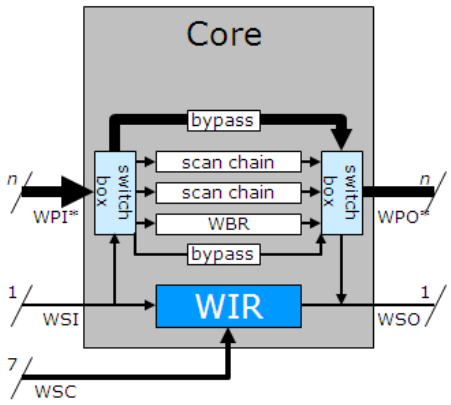


Figure 4.4: IEEE 1500 compliant core .

testing while keeping the external pin count low. While within the SoC IEEE 1500 is used to take advantage of the high-bandwidth parallel test interface it provides. Further advantages of IEEE 1500 has over IEEE 1149.1 is the reduced complexity stemming from not using an FSM as a control mechanism, but the WSC directly, at the expense of cheap internal terminals.

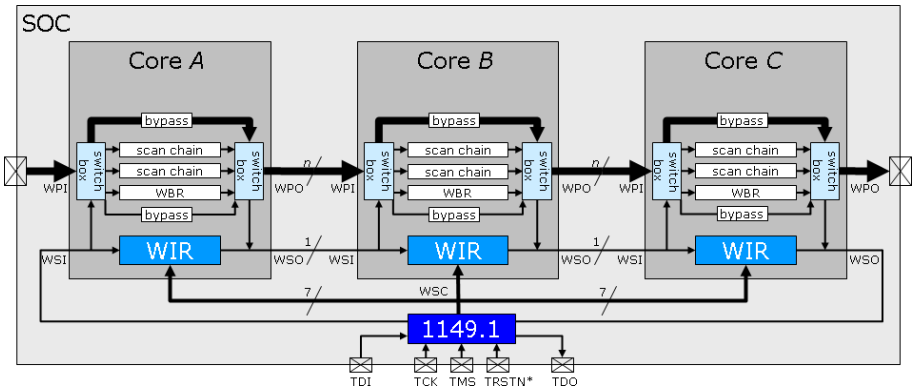


Figure 4.5: SOC-level test access architecture for cores based on IEEE 1500.

Next to the similarities and some of the differences between IEEE 1149.1- and IEEE 1500-based test access architectures given earlier. We also provide a more comprehensive list of differences between these two test access architectures, the list below covers the most important ones.

- Unlike IEEE 1149.1, the focus of IEEE 1500 is not (only) on testing wiring interconnects between cores. First of all, the interconnect circuitry in between IP cores typically does not consist only of wires, but is often formed by deep sequential logic [23]. In addition, IEEE 1500 is meant to support also the testing of the cores themselves, and IP cores are often significantly-sized and complex design entities. Therefore, the test data volumes involved are typically quite large, and a single-bit test data interface would not suffice. Hence, IEEE 1500 has an optional n-bit ('parallel') test data interface (named WPI and WPO), where n can be scaled by the user to match the test data volume needs of the IP core in question.

- Adding wider interfaces to embedded IP cores does not add *chip pins* as in IEEE 1149.1, but only *core terminals*; and they are considered to be significantly less expensive than chip pins.
- IEEE 1149.1 has two (or three) standardized control pins, which are expanded within the chip by the TAP Controller. IEEE 1500 has no TAP Controller, but receives it control signals directly. These are six (or seven) signals: WRCK, WRSTN, SELECTWIR, SHIFTWR, CAPTUREWR, UPDATEWR (and optionally TRANSFERDR) [3, 16, 13].

Figure 4.5 also features a parallel wrapper bypass. This bypass is not mandated by IEEE 1500, but often implemented to shorten the test access path to other cores in the same TAM [24]. It is the task of the *switch boxes* in Figure 4.5 to make an effective mapping between the active WIR instruction mode and the TAM-to-chain connections.

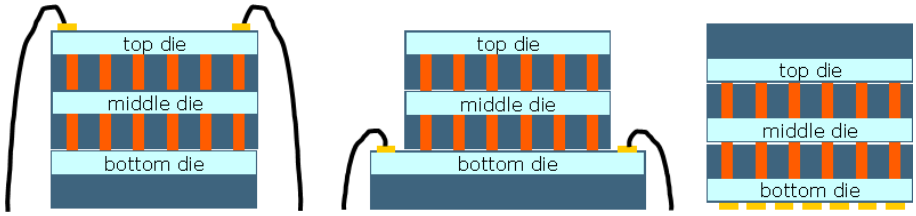
IEEE 1500 only standardizes the core-level test wrapper, and *not* the SOC-level test access architecture of the optional parallel TAMs. At the SOC-level, optimizations can be made w.r.t. TAM type [25, 26], TAM architecture [24], and corresponding test schedule. In a typical implementation, as shown in Figure 4.5, the SOC itself is equipped with an IEEE 1149.1 wrapper to facilitate board-level testing. The IEEE 1500 serial interface (WSC, WSI, and WSO) is multiplexed onto the IEEE 1149.1 Test Access Port [16] to save otherwise additional test pins. The IEEE 1500 parallel interface (WPI and WPO) can be multiplexed onto the functional external pins, as is common for regular scan chains; this saves otherwise additional test pins.

4.4 Requirements and Constraints

Testing 3D-SICs has much in common with conventional 2D IC testing; the same stuck-at, delay and other fault models apply. This is not unexpected, as the technology used to create 2D chips is still used to create the tiers of a 3D-SIC, leading to the same physical characteristics of the components leading to same requirements in regards to the fault models. However there are differences as well, especially in terms of Test Access; in applying the for mentioned tests to the tiers of the stack. In order to make a 2D die into a 3D tier hardware is added, such as the TSVs interconnects, any additional hardware will also need to be testable by the architecture In this section we highlight the differences between 2D dies and 3D tiers and derive requirements and constraints for a 3D-DfT Access Architecture, based on IEEE 1500.

We consider three types of 3D-SICs. Examples of these types are depicted in Figure 4.6; each stack consists of three tiers. The three types differ in their connections to the external world (‘*pins*’): (a) wire-bond from the top die, (b) wire-bond from the bottom die, and (c) flip-chip connections from the bottom die. All three types have in common that only one side of one of the extreme tiers (top or bottom) holds all external connections. In the remainder of this work we assume all external connections are in the *bottom* die. This assumption is without loss of generality, as we can always swap the references to top and bottom die.

We distinguish two specific test situations for each tier [12]; the first being the situation where the tier is not part of a stack yet, we call this situation *pre-bond*-testing. The other situation is



(a) (b) (c)

Figure 4.6: Three options for 3D-SIC external connections: (a) wire-bond from top die, (b) wire-bond from bottom die, and (c) flip-chip from bottom die.

called *post-bond*-testing, where the tier is part of a stack, the stack is not required to be fully completed.

Technically, pre-bond testing resembles the wafer-level tests performed on 2D chips before packaging. A singulated die is tested on its own without taking the rest of the stack into account. What does change is the manner the probe station needs to make contact with the die. In a 2D process the dies are outfitted pads on which later package pins are attached. These pads are large enough for the specialized wafer probers to connect to and perform the required tests. In contrast, a tier designed to be part of a 3D stack will have pads not designed to be fitted to off-chip pins, but designed to connect to the exposed TSV tips of another tier. These tips populate the surface of the tier in high densities with a minimum pitch of 10 μm [27], and because of the high density TSV tips, we also need high density low pitch TSV landing pads. To meet the low pitch requirements, the pads are very small. So small that even the most modern wafer probe stations are unable to probe these pads, as they are simply too small to reliably make contact [12]. The solution is to fit two types of pads. Small pads that target the TSV tips of a lower tier in the stack, and large dedicated Pre-Bond test pads that are big enough to be probed by the tester as to provide access to the chip [17, 12, 21].

Post-bond stack testing is different again, as for a non-bottom tier once part of a stack, no direct access to the tier is possible. The only tier to which the tester has access is the bottom tier. Every tier between the to-be-tested tier and the external IO therefore has to include a mechanism to transfer the test stimuli towards the tier that is to be tested, and transfer the responses from that tier back towards the external IO pins. In effect this means the test data is performing a 'u-turn' in the tier being tested; we refer to these as *TestTurns*. The hardware that provides this functionality and routes test signals up and down through the stack, includes the TSVs and is referred to as *TestElevators*. During stacking, new connections between the tiers are formed; we call these connections the interconnect between two tiers. Previously the TSVs and most of the landing pads were essentially dead ends and hard if not impossible to test, but after stacking they form direct connections between the tiers and are thus testable. A test of a stack may consist of (re-) tests of the various tiers, as well as tests of the TSV-based interconnects between the tiers. A 3D-SIC test access architecture should support all these tests. While testing unpackaged stacks, it should be possible not only to test the complete stack, but also to test partial stacks. Furthermore the test access architecture should also support the testing of the external interconnects, once the 3D-SIC is mounted on a board.

Furthermore, we want the 3D-SIC test to be a *modular* [24, 13], as opposed to a test in which the entire stack is tested as one monolithic entity. A modular test considers the various dies and TSV-based interconnect layers as separate test units. It is very well possible that complex dies are further sub-divided in multiple finer-grain test modules, e.g., embedded cores. Modular testing for 3D-SICs comes with the same benefits as it brings to 2D-SOCs [13]: (1) different tests for various modules of heterogeneous products, (2) test of black-boxed IP, (3) divide-and-conquer test generation and application, and (4) test reuse. Modular testing provides two more benefits specific to 3D-SICs: (5) flexibility in optimizing the test set per step of the test flow ("how often do we re-test a module?"), and (6) first-order diagnosis ("which module of the stack contains the fault?"). The latter is all the more important given the likelihood that multiple companies contribute to the manufacturing of a single 3D-SIC. Modular testing requires DfT in the form of *wrappers* that provide controllability and observability at the boundary of the module-under-test and *Test Access Mechanisms (TAMs)* that transport the test data from the chip's probe pads or pins to the module-under-test and vice versa [26, 13].

We assume a 3D-SIC of which the constituting tiers are *scan testable*; for example, this can include scan-testable digital logic, BIST-ed embedded memories, or even scan-enabled analog cores. Furthermore, we assume it is a requirement for board-level interconnect testing that the overall product is IEEE 1149.1 compliant on its external pins. We assume that additional external test pins beyond those required for IEEE 1149.1 are expensive and hence should be avoided. In contrast, we assume that additional TSV-based interconnects between tiers for the purpose of test are relatively affordable. We assume this because TSVs can be made small; e.g., IMEC's *via-middle* TSVs are made at a 10 μm minimum pitch [27, 28]. This type of TSV requires only a small amount of die space, and production costs of *additional* pins is negligible as the dies needs to be put through the production steps to manufacture the functional TSVs anyway.

The test access architecture should be able to provide a trade-off between additional area cost for DfT, test generation effort, and test length. To minimize silicon area, we want to re-use the existing intra-die DfT infrastructure as much as possible: internal scan chains, test control, test data compression circuitry, built-in self-test, etc. To minimize the test generation effort, we prefer to base our die-level wrapper on the existing IEEE Std. 1500, given its scalable TAM width and flexible WIR. The test access architecture should allow for flexible test scheduling to minimize the test length. It is also a requirement that the test access architecture itself is testable. It is desirable that this can be done without depending on the correct functionality of the existing DfT inside the local dies and embedded IP cores.

2D-SOCs allow us to design DfT features in the SOC circuitry around (i.e., outside) the embedded cores. This is not the case for 3D-SICs; all DfT needs to be in the various dies. The only thing that exists outside the dies are vertical interconnects, and even those need to be pre-designed in terms of die-level features, such as TSVs and TSV landing pads. This implies that wrappers, TAMs, and their control signaling all needs to be pre-designed in the die; not only for *that* die, but also for the dies above it in the stack. Hence, we assume that for all tiers, the DfT is designed in adherence to a pre-defined test access architecture, or that we have the freedom to modify the DfT circuitry; it cannot be added as an after-thought.

Finally we require the test access architecture to be *scalable*, in the sense that it works for an undetermined number of stack tiers. Also, the architecture should not predestine a die to a

certain tier level, such that dies that adhere to the architecture can function at any level in the stack hierarchy. Exceptions to this requirement are formed by the bottom die, which with its external connections is obviously predestined as bottom, and possibly the top die.

4.5 3D-SIC Test Access Architecture

This section covers a 3D-DfT test access architecture for 3D-SICs, that covers all the requirements and constraints presented in Section 4.4. First we will introduce the specific 3D-DfT features incorporated into the die level-wrapper of our architecture in Subsection 4.5.1. This is followed by Subsection 4.5.2, that covers the operating modes of the architecture and how the control mechanism of our architecture uses 3D-DfT features to enable 3D-SIC testing.

4.5.1 Die-Level Wrapper

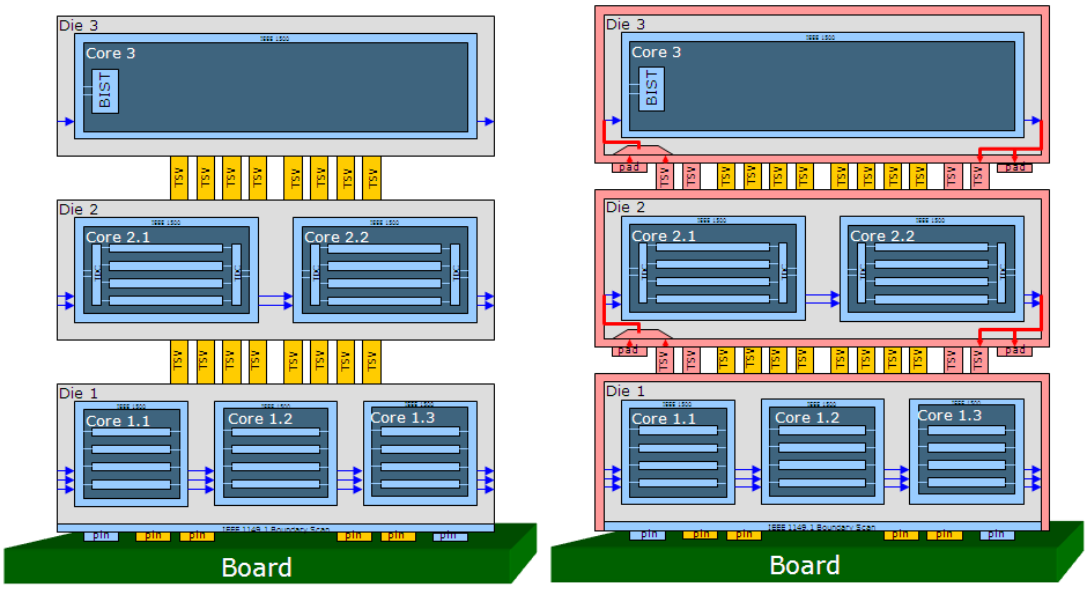
The test access architecture we propose for 3D-SICs is based on a die-level wrapper, which is an extended version of IEEE 1500. In Figure 4.7(a) a conceptual version of a stack is shown, with the existing DfT in light blue and only function connections made between the dies. In Figure 4.7(b) we show the same stack, with 3D-DfT die-level wrappers in pink, dedicated test TSVs and dedicated pads to enable *pre-bond*-testing. The 3D-DfT wrapper provides a consistent interface to other dies in the stack, while internally within the die, it connects up to the existing functional circuitry and regular intra-die DfT. The architecture uses a limited, scalable number of dedicated TSV-based interconnections between dies in addition to the already existing functional interconnects.

Die x in Figure 4.8(a) is equipped with an IEEE 1500-like wrapper that is normally encountered with embedded IP cores. The figure also shows the conventional IEEE 1500 features of that die-level wrapper: a seven-bit Wrapper Serial Control (WSC), a Wrapper Instruction Register (WIR), a Wrapper Boundary Register (WBR), a serial WSI-WSO interface for instructions and low-bandwidth test data, and the optional parallel WPI-WPO n-bits wide interface for test data.

Figure 4.8 shows the conceptual steps for upgrading a standard IEEE 1500 equipped Die x in Figure 4.8(a) to a die with the 3D-DfT architecture we propose fully implemented in Figure 4.8(d). In the figure we show a schematic overview of the DfT features and additional interconnects for an arbitrary Die x in the middle of a stack. The figure abstracts from the functional circuitry and interconnects. It shows two internal scan chains, which are representative for the possible die-internal DfT, such as any number of scan chains for a monolithic design, TAMs for a core-based SOC design, and/or BIST-ed logic or memory.

Our die-level wrapper has four 3D-SIC-specific features.

1. *TestTurns*. In a stack all I/O is perform via the bottom tier, necessitating all control and data signals of IEEE 1500 (WSC, WSI, WSO, WPI, and WPO) to enter and exit Tier x via the same side from and towards the bottom tier; see Figure 4.8(b). With this transformation, electrically few things change except for the additional pipeline delay elements in the



(a) (b)

Figure 4.7: (a) Concept stack of dies, functionally connected, (b) stack with three die-level 3D-DfT wrappers.

return data path. These pipeline delay elements provide a fixed delay of one clock pulse per die, eliminating unpredictable delay over long wiring and ensuring signal integrity.

2. *Dedicated probe pads*. These are required for pre-bond die testing, as shown in Figure 4.8(c). For easy figure layout, these probe pads are drawn on the bottom side; however, that does not imply that these probe pads need to be physically located at the bottom side of the die. Note that the width of the parallel interface WPI-WPO might be chosen differently for the TSV interconnects (n) and probe pads (m). As the parallel interface can be done away with entirely as it is only optional, it holds $n \geq m \geq 0$ for normal cases.

3. *TestElevators*. These are the mechanisms that raise all control and data signals of IEEE 1500 signals to a higher tier in the stack. The set of raised signals maintain their original names, post-fixed with the letter 's' (for 'stack'): WSCs, WSIs, WSOs, WPIs, and WPOs as shown in in Figure 4.8(d). The *TestElevators* are all situated on the top side of the die, and include dedicated test TSVs and routing hardware.

4. *Hierarchical WIR chains*. These are used to prevent unbridled lengthy WIR chains. This is further described later in this section and depicted in Figures 4.11 and 4.12.

Figure 4.9 depicts the test access architecture for an example 3D-SIC containing three tiers; Tiers 1, 2, and 3 are respectively the bottom, middle, and top tier of the stack. To show the similarities and differences with test access architectures for PCBs and 2D-SOCs, the tier are shown next to each other, instead of as a vertical stack.

As Figure 4.9(a) shows, the DfT in a bottom tier differs from a middle die in the following aspects.

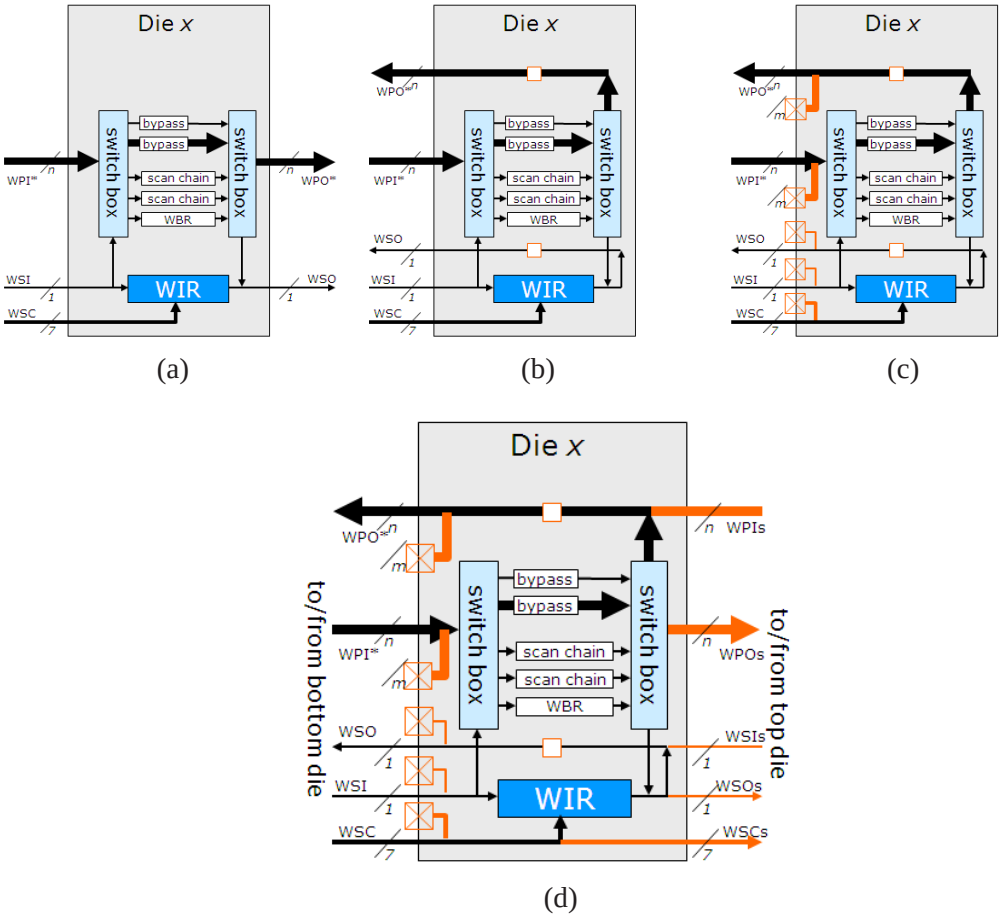


Figure 4.8: (a) IEEE 1500 die wrapper, (b) with 3D-DFT *TestTurns* and pipeline delay elements, (c) with *probeable pads*, (d) with *TestElevators*, fully 3D-DfT compatible.

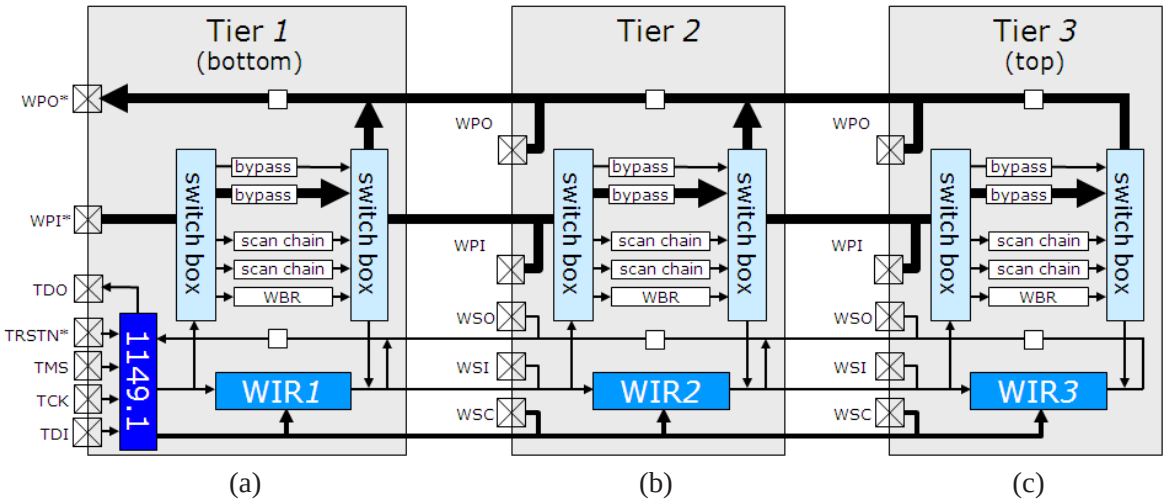


Figure 4.9: 3D-SIC test access architecture for tiers based on IEEE 1500.

- Dedicated pre-bond probe pads are not required. Instead, the functional external I/O pads can be used for probe access.
- The bottom die is equipped with IEEE 1149.1 to facilitate board-level testing and provide a board-level test and debug port. The JTAG boundary scan chain includes all external I/Os of the 3D-SIC product.
- The serial IEEE 1500 interface (WSC, WSI, and WSO) can be multiplexed onto the IEEE 1149.1 Test Access Port (TAP), similar to what is common in 2D-SOCs [16]. This saves otherwise dedicated pads, and makes the 3D test access architecture accessible even when the 3D-SIC is soldered onto a PCB.
- The parallel IEEE 1500 interface (WPI and WPO) is multiplexed onto the functional external I/O pads, similar to what is common for scan chains and parallel TAMs in 2D-SOCs. This saves otherwise dedicated pads, but restricts the *TestElevator* width to the available functional I/O.

The DfT in a top tier differs from a generic middle tier in the following aspect, as Figure 4.9(b) and Figure 4.9(c) show.

- The tier does not have TSV-based interconnects to an even higher-level tier, as it is the top tier. Hence, the top-side TestElevators WSCs, WSIs, WSOs, WPIs, and WPOs are absent.

IEEE 1500 allows various types of wrapper cells in its WBR. Embedded cores in 2D-SOCs commonly use the cell depicted in Figure 4.10(a); it consists of only a single flip-flop and hence occupies little silicon area. For the WBR chain of our proposed 3D-SIC die-level wrapper, we prefer to use the (also IEEE 1500-compliant) double flip-flop wrapper cell shown in Figure 4.10(b). At the expense of an extra flip-flop, this wrapper cell provides ripple-protection during shift mode, which seems appropriate especially if the various dies come from different sources, and ripple-during-shift might result in unwanted signal combinations at the inter-die interfaces.

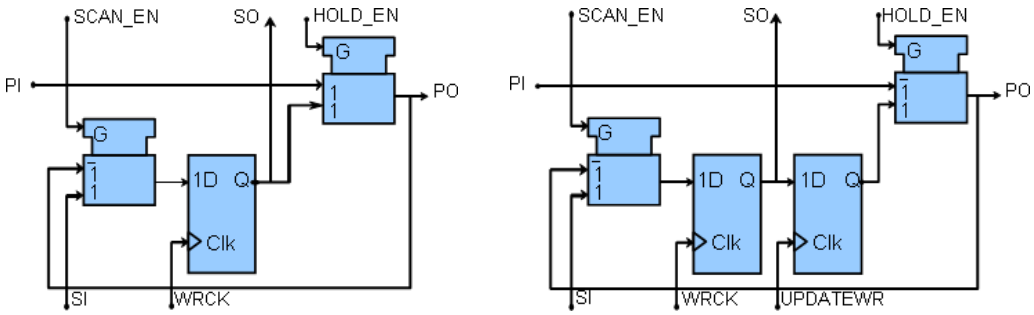


Figure 4.10: IEEE 1500-compliant WBR cells, (a) commonly used for embedded cores in 2D-SOCs, and (b) proposed for stacked tiers in 3D-SICs.

Loading instructions into a WIR of a die-level wrapper is similar to what is known from IEEE 1500-compliant cores in 2D-SOCs. While a new instruction is shifted into the WIR, the previous instruction remains valid; only once fully arrived in place, the new instruction is acti-

vated by pulsing the UPDATEWR signal. In IEEE 1500, the WIRs of multiple IP cores are to be concatenated in a single WIR chain, which allows different cores to be loaded with different instructions. For 3D-SICs, a single concatenated WIR chain might become very lengthy, especially in case the individual tiers are core-based SOCs [29] with their own concatenated WIR chain segments as shown in Figure 4.11(a). Hence, we propose an hierarchical WIR mechanism as in Figure 4.11(b), which opens up as needed, similar to a harmonica.

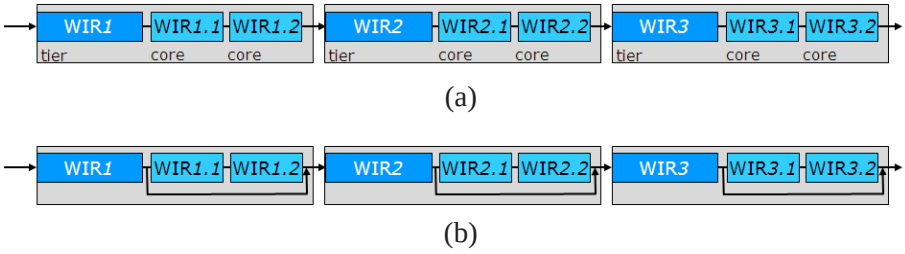


Figure 4.11: (a) WIR chain length subject the #tiers, #cores and opcode length, (b) *hierarchical* WIR chain, that allows the chain to be shorted, by bypassing core-lever WIRs.

Initially, the WIR chain only consists of the die-level WIRs. Once loaded with tier-level instructions, the core-level WIR chain segments are included in the overall WIR chain for only those tiers for which one of the *InTest* instructions was given; subsequently, further core-level WIR instructions can be loaded. Figure 4.12 schematically shows this concept by means of an example. The orange arrows highlight the active WIR chain. In this example, Dies 2 and 3 are in an *InTest* mode and hence, the WIR chain also includes the WIRs of their cores, resp. WIRC + WIRD and WIRE + WIRF. The benefit of this hierarchical WIR mechanism is that we prevent an unbridled growth of the WIR chain length; at any given moment, the WIR is only as long as required. The cost is the requirement for the user to keep track of the current WIR chain length and a more complex procedure for loading instructions.

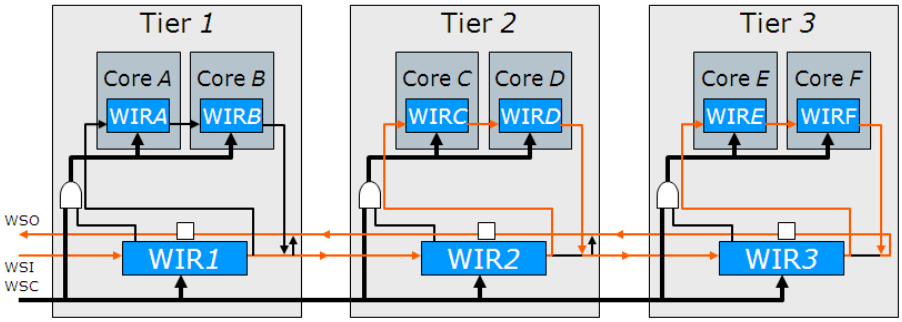


Figure 4.12: Hierarchical WIR chain, which has opened up for Tiers 2 and 3, which are in one of their *Intest* modes.

4.5.2 Operating Modes

The operating modes for the 3D-DfT test architecture are composed from four fields, with two or three settings each. Leading to 24 possible test modes. In Figure 4.13 we show these four

fields, and the options per field, the options and there effect are described below.

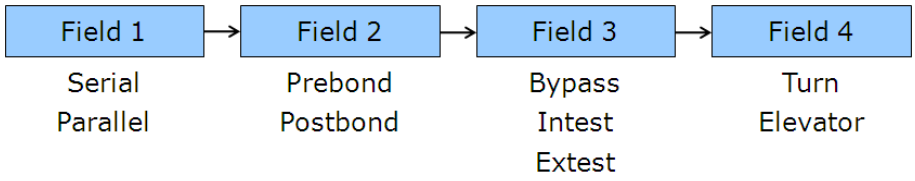


Figure 4.13: Composition of the operating modes, four fields with two or three options each.

- *Field one: Serial / Parallel* – The choice of interface for the test patterns to be administered to the tier, in a 'serial' mode, the test data path is routed through the single-bit WSI/WSO interface of the tier, while in 'parallel' mode it is routed to the n-bit wide WPI/WPO interface.
- *Field two: Prebond / Postbond* – This field specifies the use of the TSV landing pads or the larger dedicated probe pads to access the interface of field one. The TSV landing pads are used after bonding when the pads are connected to tiers lower in the stack, before bonding the dedicated pads are used to allow a probe station access to the tier directly.
- *Field three: Bypass / Intest / Extest* – Field three specifies the type of test administered to the tier. In 'Bypass' only the DfT access mechanism is tested, to verify it is functional before testing the die itself, 'Intest' is a test focused on the tiers itself, all scan chains as well as the WBR is incorporated into the test data path to maximize fault coverage. In 'Extest' the interconnect between two tiers is targeted by the test, necessitating the cooperation of the two tiers involved. Only the WBR in both tiers is used, isolating the TSV interconnect between those two WBRs.
- *Field four: Turn / Elevator* – Field four specifies which tiers of the stack are involved in the test. The 'Turn' setting directs the test data flow back towards the external I/O of the stack, isolating the tiers higher in the stack from the test. In 'Elevator' mode the tier above is involved in the test as the test data signals are propagated to it.

With two, two, three, and two options for the four fields of the operating mode, there are 24 possible operating modes. Not all of these operating modes seem currently useful, 'Elevator' mode for a tier in pre-bond testing would mean the test data is routed to a tier that has not been stacked unto the tier being tested yet. Similarly an 'Extest' is focused on testing the interconnect between two tiers, in pre-bond testing however, there is only one tier. This leads to the following operating modes:

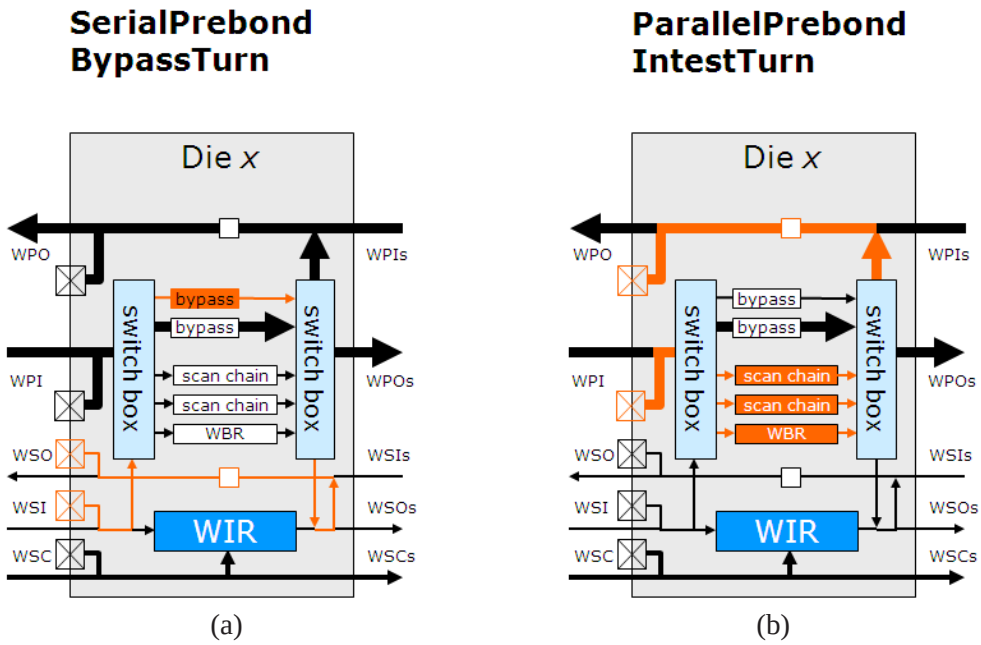
- *SerialPrebondBypassTurn*
- *SerialPrebondIntestTurn*
- *SerialPostbondBypassTurn*
- *SerialPostbondIntestTurn*
- *SerialPostbondExtestTurn*
- *SerialPostbondBypassElevator*
- *SerialPostbondIntestElevator*
- *SerialPostbondExtestElevator*
- *ParallelPrebondBypassTurn*
- *ParallelPrebondIntestTurn*

- *ParallelPostbondBypassTurn*
- *ParallelPostbondIntestTurn*
- *ParallelPostbondExtestTurn*
- *ParallelPostbondBypassElevator*
- *ParallelPostbondIntestElevator*
- *ParallelPostbondExtestElevator*

A bottom tier does not implement the pre-bond operating modes, as a bottom die does not have dedicated test pads. Similarly a top tier does not require Elevator modes, which therefore do not have to be implemented.

Combining instructions for the various tiers in a stack allows us to test one, multiple, or all tiers simultaneously, as well as test one, multiple, or all layers of TSV-based interconnects simultaneously. This gives the same test scheduling options as the single daisychain TAM for 2D-SOCs [30].

Figure 4.14 shows two of the four possible test modes for pre-bond testing, the *SerialPrebondBypassTurn* mode allows for the DfT architecture itself being tested. The internal scan chains, and WBR are not used, the test patterns are propagated through the WSI/WSO interface, switch boxes and the serial bypass, thereby testing those components. The *ParallelPrebondIntestTurn* mode is aimed at testing the die itself, in this mode the main aim is to propagate the signals through the die to its primary inputs and outputs by putting the WBR in the data path. Because large complex dies can be fitted with scan chains to aid testing of the core by introducing pseudo primary inputs and outputs, these scan chains are also part of the test data path.



(a) (b)

Figure 4.14: Two Prebond test modes, (a) testing of DfT, (b) testing the die.

Figures 4.15, 4.16, and 4.17 show examples of a 3D-SIC being tested in post-bond modes, in which neighboring tiers are in different operating modes. The modes in Figure 4.15 together

form a interconnect test between dies two and three only, with die one only there to transfer date to the other dies. The same depicted in Figure 4.16 but this time the parallel interface is used. In Figure 4.17, Die 1 is being tested. Die 3 is also in a *ParallelIntest* mode, but because Die 2 is a *turn* mode Die 3 is not tested. The responses from Die 3 are not propagated to the bottom die, thus no matter what mode Die 3 is in, it will not be included in the test until die two is in a *elevator* mode.

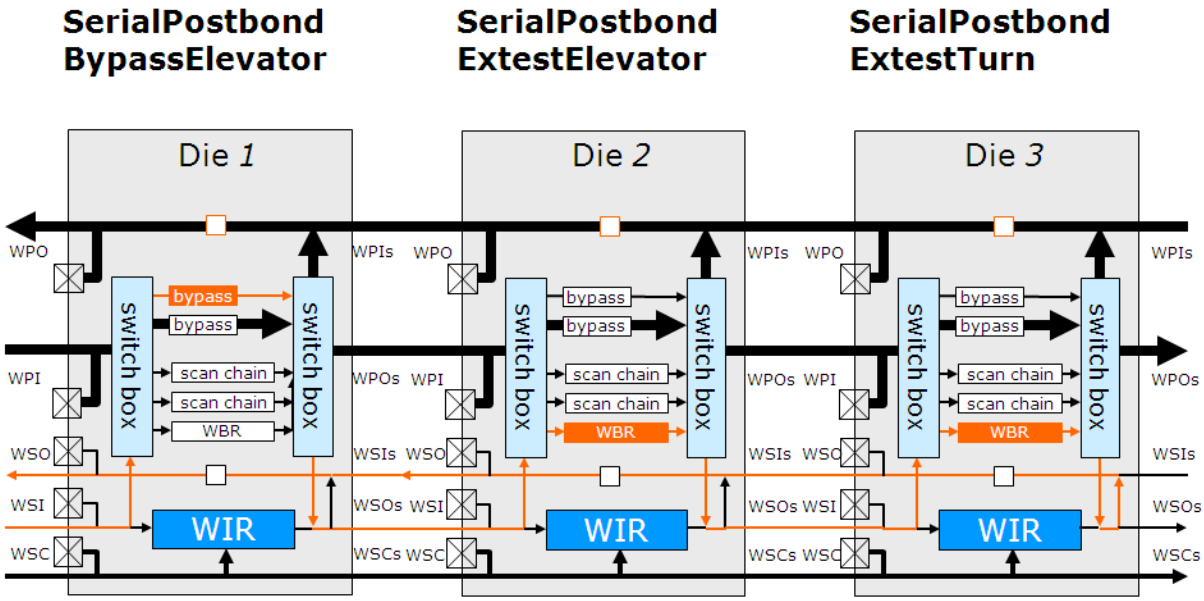


Figure 4.15: Example in which the interconnect between Dies 2 and 3 is being tested; Die 1 is in *BypassElevator* mode to enable test data propagated to the other dies.

4.6 Verification of the proposed architecture

In this section we verify the 3D-DfT architecture presented in Section 4.5. There are four subsections in this section, in Subsection 4.6.1 we give an overview of the experimental environment and the tools we used. Subsection 4.6.2 covers an idealized tool flow to equip a die with the 3D-DfT architecture, followed by the tool flow of the experiment as we ran it, and test coverage reports for the 3D-DfT equipped dies. In Subsection 4.6.3 we will create a stack with our 3D-DfT equipped dies, and again have a look at an idealized flow next to the flow actually used in the experiment. We also provide test-coverage results for the stack in various operating modes, and show that high fault coverage of a 3D-SIC can be obtained by including the 3D-DfT architecture. The final Subsection 4.6.4 covers an analysis where we consider the costs of implementing the 3D-DfT architecture into the tiers of a 3D-SIC. We dissect the area overhead imposed by the 3D-DfT architecture into specific components and provide an method to accurately estimate the overhead of implementing the 3D-DfT architecture on a arbitrary chip.

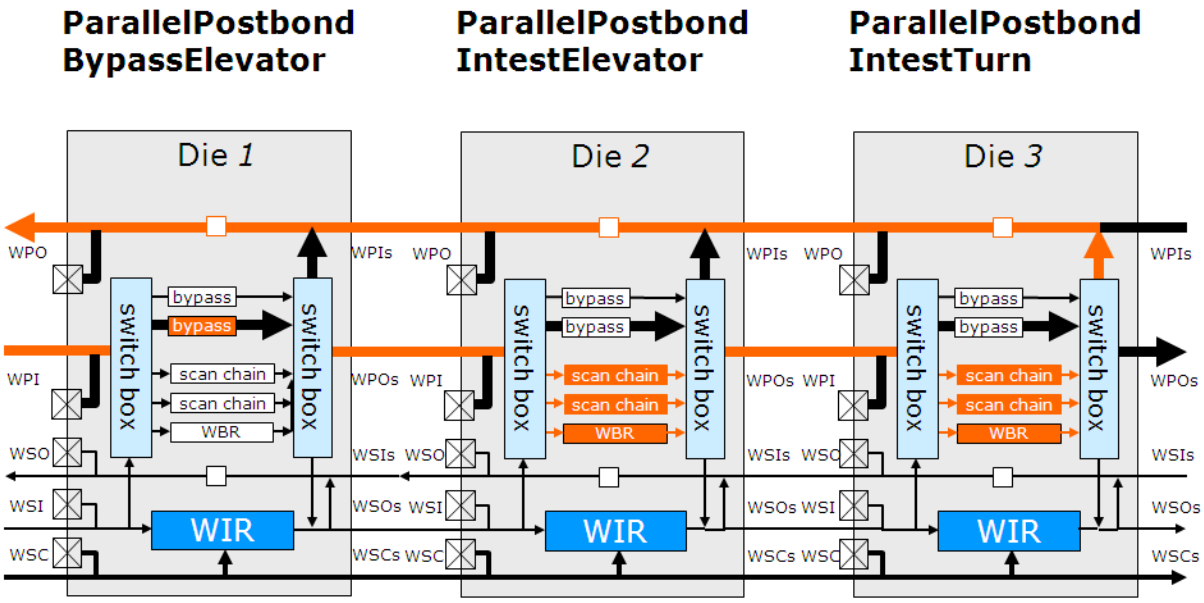


Figure 4.16: Example in which Dies 2 and 3 are being tested; Die 1 is in *BypassElevator* mode to enable test data propagated to the other dies.

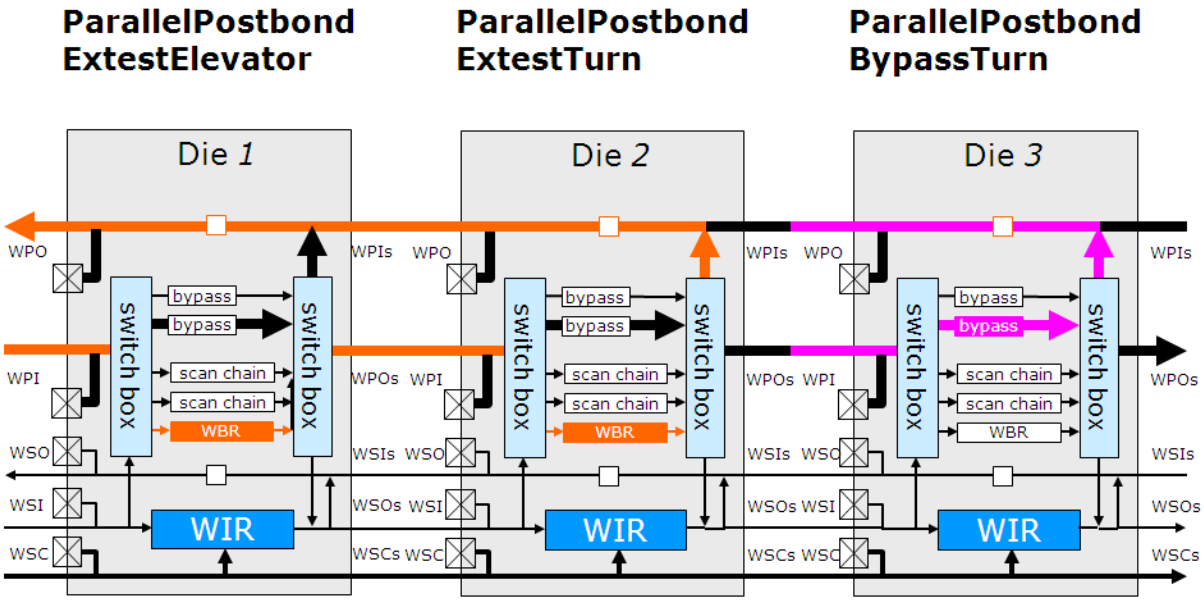


Figure 4.17: Example in which the interconnect between Dies 1 and 2 is being tested; Die 3 does not take part in the test, as Die 2 is in a *Turn* mode.

4.6.1 Experimental Vehicle

To verify our 3D-DfT architecture we required a minimum number of three digitally testable circuits to simulate bottom, middle, and top tiers in a stack. As the implementation of the 3D-DfT architecture would be performed manually, we wanted fairy small manageable netlists of these circuits. We elected three ISCAS'89 benchmark circuits, these are small monolithic designs of with a limited number of inputs, outputs, and flip-flops. We have taken these circuits because they are readily available to anyone and are well known in the field of IC design and testing. The three ISCAS circuits have been mapped to the UMC90 library, which we use in our design tools. The specifications of the selected s400, s1423, and s5378 ISCAS designs are shown in Table 4.1, the area shown in the table reflects designs with three scan chains enabled. These ISCAS designs are in size not comparable to large commercial circuits, but are still big enough to prove the concept of the 3D-DfT architecture. The high-level concepts of Figure 4.8(d) were adapted to the 2D design tools involved and finally lead to 3D-DfT components part of the ISCAS circuits.

tier	#Cells	FlipFlops	inputs	outputs	area(μm^2)
$s400$	186	21	3	6	1044
$s1423$	734	74	17	5	3748
$s5378$	2961	179	35	49	11751

Table 4.1: Functional input and output of three ISCAS'89 benchmark circuits.

We create a stack by taking the 3D-DfT enabled ISCAS netlists and adding them to a new top-level stack netlist, and link the functional I/O of the circuits to each other. During this procedure no additional hardware has been added, just like the stack maker would be unable to add hardware in the dies or between them while creating the stack. In Figure 4.18 we see how we linked the tiers, functional I/O is shown in yellow and the parallel (wpi/wpo) interface of the 3D-DfT architecture is shown in the pink arrows. We have selected $n = 3$ for our parallel interface for all tiers, for the bottom tiers this interface can be multiplexed with the functional I/O.

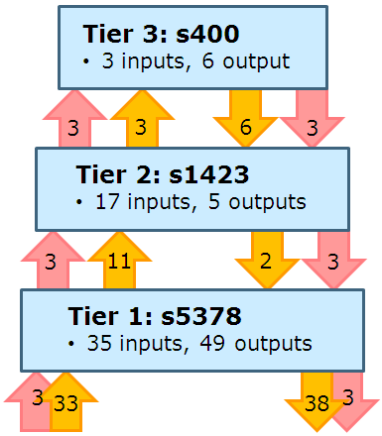


Figure 4.18: 3D stack comprised of three ISCAS designs.

We have chosen the ISCAS circuits so that lower tiers in the stack always have more available I/O then the tiers above. We have linked the 3 inputs and 6 outputs of the s400 to the 17 inputs and 5 outputs of the s1423, after completing this s1423 has 11 inputs and 2 output left unconnected. The remaining I/O of s1423 are now directed downward and connected to the I/O of s5378, the I/O of s5378 that is still left unconnected is directed downward as well, and can be considered as the inputs and outputs of the 3D-SIC. Functionally our stack does not make much sense, because the s400, s1423, and s5378 where never intended to work together, but it does reflect the way tiers would be connected in a 3D-SIC.

4.6.2 Die-level 3D-DfT verification

We envision a 3D-DfT tool flow as shown in Figure 4.19. Files like netlists and test patterns are shown as light blue, with software tools in a darker blue tone. A tier has been designed with scan chains in mind and the functional design has been completed. This design is fed to a 3D-DfT wrapper tool which adds all the necessary hardware to upgrade the die to full 3D-DfT compatible status. For the 3D-DfT enabled tier test patterns are generated by the ATPG tool. In the simulator stage the test patterns and test bench generated by the ATPG tool are simulated, and a pass/fail result is provided to the operator.

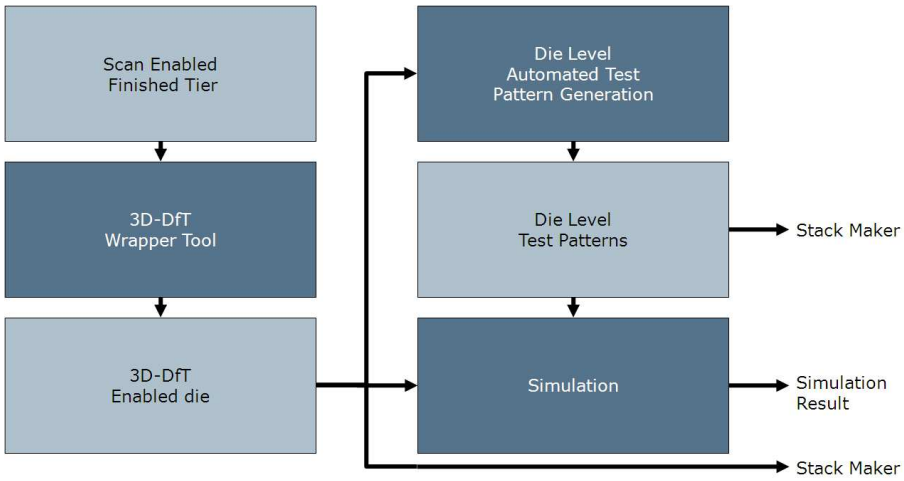


Figure 4.19: Ideal tool flow for 3D-DfT wrapping and verification of a finished tier design.

In Figure 4.19 the actual tool flow of the verification process of the 3D-DfT is depicted. As before files like netlists and test patterns are shown as light blue, with software tools in a darker blue tone, in black the manual intervention in the flow is shown. As our 3D-DfT architecture is based on the IEEE 1500 standard for core testing, the first step is to equip our benchmark circuits with a fully functioning IEEE 1500 DfT architecture. This also implies scan chains to be present, as it is common practice to equip any logic design with scan chains to facilitate testing of the product. We perform the task of scan insertion with *Cadence RTL Compiler*, we then provide the scan enabled tier to *Cadance Encounter Test*, which adds a basic IEEE 1500 wrapper to the tier. We take the 3D-DfT architecture into account while inserting IEEE 1500, by choosing an appropriate WIR opcode length, and some of the opcodes themselves.

In the 3D-wrapper section we add 3D-DfT elements for a tier directly to the netlist, this includes the addition of muxes, WIR control signals, modifications to the WIR itself and fixed delay elements. This does not however include simulated pre-bond probe able pads, we choose not to simulate these, as they would be nothing else but additional pins muxed with identical pins in the netlists, and would have created a problem in controlling the pre/post bond muxes. As there is little added benefit, we have chosen to abstract from this detail, and leave its implementation to a standardization group on 3D-SIC testing.

The 3D-DfT enabled tiers is then provided to *TetraMax* together with numerous ATPG scripts that put the die in various modes before executing the ATPG run. Multiple ATPG runs are necessary because when the tier is in one mode, it blocks access to various parts of the tier resulting in a low fault coverage. With multiple runs we can select all appropriate modes, and add the additional coverage of each mode to a total. To accurately add these coverage results, after each run we remove covered faults from the target fault list for subsequent ATPG runs. The ATPG runs together create a test bench which is there after used in a simulation together with the 3D-DfT netlist by *NCSim* that provides a pass/fail analysis on the functionality of architecture for *pre-bond* tier testing. The NCSim results are for the operator, when the results are satisfactory the 3D-DfT netlist and test bench can be provided to a stack maker.

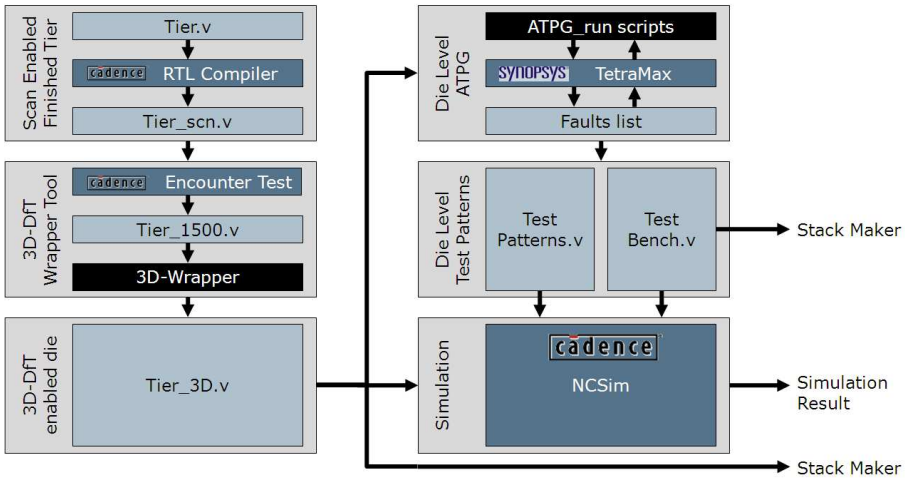


Figure 4.20: Actual tool flow for our 3D-DfT tier validation experiment.

The most fundamental change made in the flow is the addition of the 3D-Wrapper, which is based on IEEE 1500, and to insert the base IEEE 1500 implementation we use Cadence tools; And because the 1500 standard is not a complete specification, the resulting implementation of the 1500 standard is how Cadence chose to implement it. In Figure 4.21 Cadence' choices are shown.

On the top of Figure 4.21 we see the functional elements of the chip; inputs on the left, outputs on the right, and a core wrapped in wrapper cells on its functional I/O. These wrapper cells together form the Wrapper Boundary Register (WBR), the first of the added DfT components, in the rest of the figure we find more DfT components. The Wrapper Instruction Register (WIR) and Wrapper BYpass (WBY) register as stipulated by the IEEE 1500 standard. Instead of the

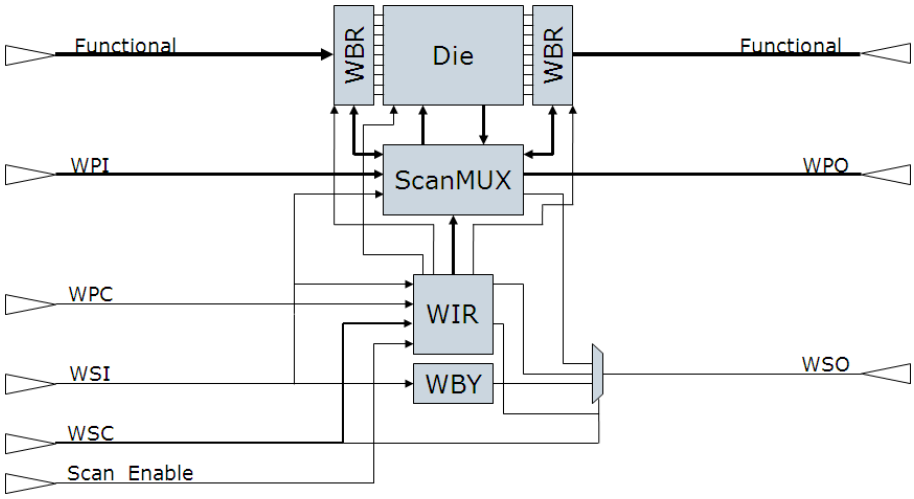


Figure 4.21: Conceptual overview of the Cadence IEEE 1500 solution.

dual Switch Boxes, there is a single *ScanMux*. *ScanMux* is responsible for concatenating the die's scan chains and WBR chains in response to *Serial* vs *Parallel* and *Intest* vs *Extest* modes loaded into the WIR.

In Intest mode a test is aimed at the core itself, enabling the core's internal scan chains as well as the WBR. In contrast, when the core is set in extest mode other parts of the chip are being targeted, requiring some input for the core. Enabling only the WBR surrounding the core means those inputs are directly controllable, and saves the core from having to be put into a specific state. In the Cadence approach of IEEE 1500 the ScanMux and the WIR work together to provide this functionality, the WIR interprets the instruction and enable a hot encoded instruction signal to ScanMux. ScanMux receives this signal and distributes the internal scan chains and WBR sections over the serial of parallel interface, according to which instruction signal was provided. The WIR also has direct access to the WBR and selects the appropriate mode for a WBR cell, for instance the 'Capture' mode.

Cadence also choose to add an additional set of control signals for the parallel interface: Wrapper Parallel Control (WPC), and a separate Scan Enable signal. We have implemented these signals as they were woven into the wrapper design, but we use the WSC to both derive our Scan Enable signal, and control the parallel interface. These are Cadence implementation details and have no consequences for the 3D-DfT architecture. Further more there are Wrapper Serial/Parallel Inputs (WSI/WPI) and Outputs (WSO/WPO) for test data.

One of the reasons we choose IEEE 1500 as a base for 3D testing is the flexibility the WIR gives us to add internal signals we need to control the flow of data through the 3D stack. When we add the hardware to simulate the TestElevators, TestTurns as well as the Pre-bond and Post-bond testing functionality, we end up with the situation of Figure 4.22. In green the newly added components, external pins, muxes, control signals, and the modified WIR which caters for those control signals. The I/Os with a 's' extension are connections to a tier higher up in the stack, and the ones without such an extension denote a connection to a tier lower in the stack or even

off-chip in case of the lowest tier in a stack (bottom tier).

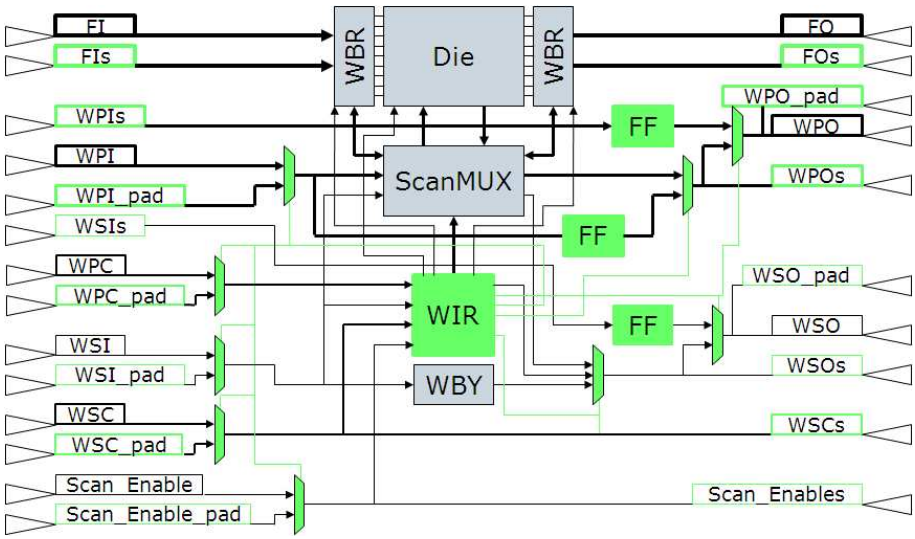


Figure 4.22: Our 3D-DfT architecture in the Cadence environment.

In Figure 4.23 we see the additional 3D-DfT components added to the s400 ISCAS benchmark circuit. On the bottom we see the WSC signals, above those the WBY and data paths of the WSI(s)/WSO(s) WPI(s)/WPO(s), and on top the WIR, ScanMux, WBR cells and the s400 itself. The other selected ISCAS benchmark designs, the s1423 and s5378 have been outfitted with the same 3D-DfT components, as shown in Figure 4.24 and Figure 4.25.

The next step is to verify that these tiers are testable, the flow for this is depicted in Figure 4.20. TetraMax is used multiple times, as seen in the flow. First it is given the Tier_3D and a setup script, in this script TetraMax is told what are the specifics of the design, such as clock information as well as a definition of the scan chains. The script does not stipulate how to set up the WIR before testing can commence. So multiple run scripts to cover each *Pre-bond* mode are created and fed into TetraMax, these run scripts specify a series of signals being sent to the tier before the actual test takes place. This puts the WIR in the correct mode; *Extest* or *Intest*, *Serial* or *Parallel Pre-bond* or *Post-bond*, *Turn* or *Elevate*, and finally *Bypass* the core or not. This enables a certain set of scan chains through the tier, and allows TetraMax to do a trace through those scan chains, and perform ATPG.

TetraMax will try to raise the fault coverage as high as possible, but this may still be very low when the chains are defined such that TetraMax has limited access to certain parts of the tier. To remedy this fact, we perform multiple TetraMax runs consequentially, with the WIR in a different mode each time. The fault coverage from previous runs is subtracted from the target fault list, thereby raising the overall coverage with each run. Finally the resulting overall test bench with patterns and including WIR setup sections is provided to NCSim for final verification of the tier.

With the verification by NCSim completed, the statistics for the test patterns given by TetraMax can be trusted. Figure 4.2 shows these statistics, it states the total number of fault sites in the circuit which excludes the 3D-DfT architecture itself, but includes the functional TSVs. The

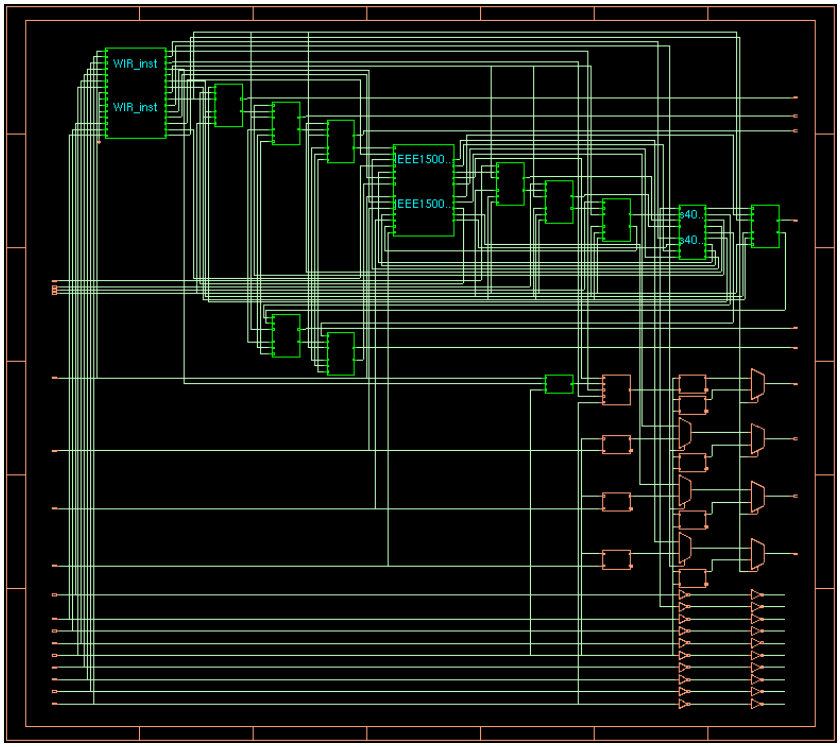


Figure 4.23: The ISCAS s400 outfitted with 3D-DfT components.

figure also shows the coverage as well as the number of patterns and the amount of test cycles these patterns require.

Pre-bond tests	Total Faults	Detected	Coverage	Patterns	Test Cycles
$s400$	1362	1239	91.0%	47	535
$s1423$	4712	4522	96.0%	107	3679
$s5378$	16226	15852	97.7%	370	33026
$total$	22300	21613	96.9%		

Table 4.2: Pre-bond ATPG results of the 3D-DfT enabled tiers.

4.6.3 Stack-level 3D-DfT verification

With the tiers individually wrapped in the 3D-DfT wrapper and verification of the wrapper enable tiers completed we now build a stack from those tiers. In Figure 4.26 an idealized tool flow is depicted for using 3D-DfT wrapped tiers to create a stack. It starts with the tiers, given to a stack creator tool, which has two outputs. The first is a stack build up from the provided tiers, the second is a spec list of the newly created stack. This spec list is provided to a 'Test Protocol Expansion' (TPE) tool, this tool used the description of the stack to modify the Die Level Test Patterns that came with the 3D-DfT tiers. And modified in such a way as to allow these test patterns to be reused as Stack Level Test Patterns (3D-SIC level). As the 3D-DfT architecture

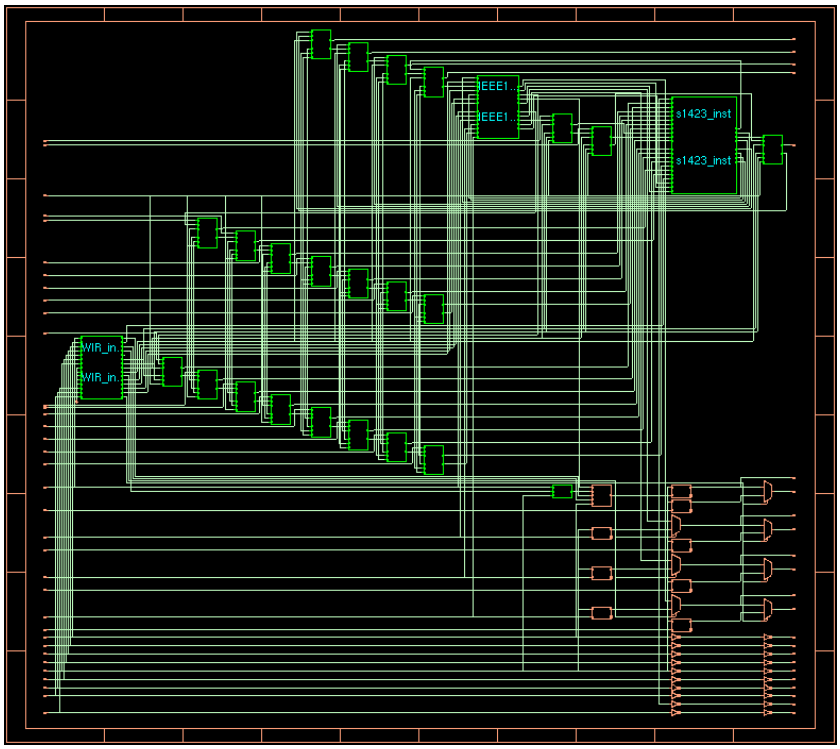


Figure 4.24: The ISCAS s1423 outfitted with 3D-DfT components.

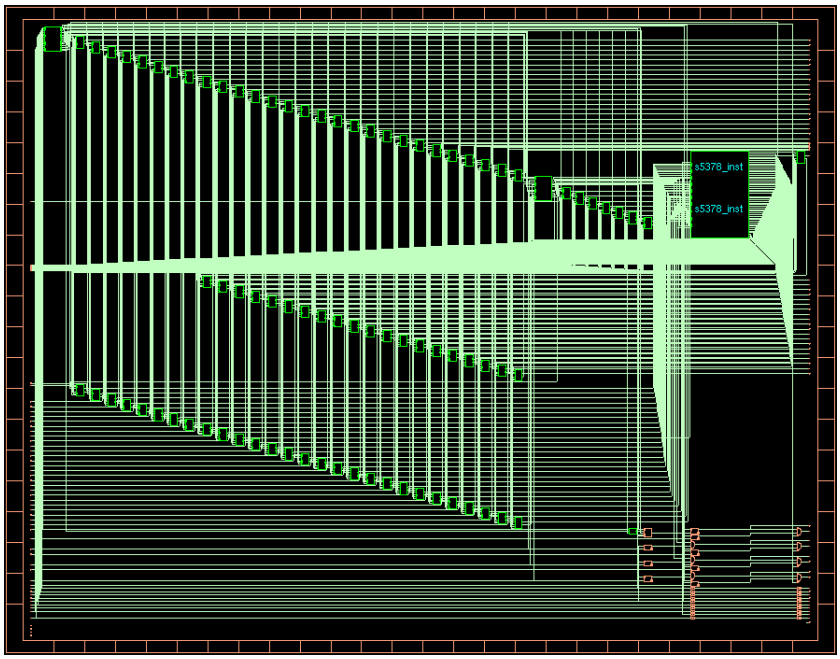


Figure 4.25: The ISCAS s5378 outfitted with 3D-DfT components.

allow for a pattern to be delivered to a tier at Tier-level, it is possible for the TPE tool to take a die-level pattern set the stack in a mode that opens up a scan chain to the appropriate tier, administer the pattern to the stack, propagate it to the tier, execute the test, and propagate the resulting bits back out of the stack for comparison. Once all the patterns have been expanded to 3D-SIC level, a simulation can be run to verify correctness. The test patterns can hereafter be used by ATE systems, and other testing equipment. While these patterns do cover the *Intest* of a tier, they do not cover the newly created interconnect between dies. The patterns for this interconnect will have to be generated by the stackmaker, this is included in the flow.

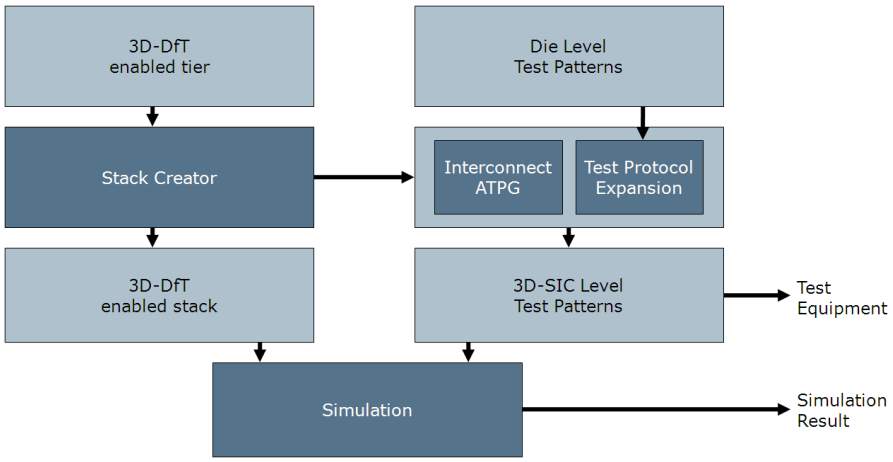


Figure 4.26: Ideal tool flow for building a stack out of 3D-DfT enabled tiers.

In our validation experiment there was no purpose build ‘stack creator’ tool, instead the stacking has been performed manually, as shown in Figure 4.27. A top level design has been created, and the three tiers have been included in that design. No additional hardware has been added, except for wiring between the tiers. This wiring includes all functional I/Os, test data pathways and control signals for the WIRs in each tier. At this stage RTL Compiler has been used as a viewer to verify the correct connections have been made. Once the 3D-SIC was completed, we have performed multiple ATPG runs on it, very similar to the tier-ATPG runs, to raise the fault coverage. The reason we went for this approach instead of the TPE route is the unavailability of such a tool at this time, they do exist as propitiatory tools within companies, but are not available commercially by the big EDA vendors. The approach we use now does work well though, but it does require the stack maker to perform ATPG instead of the actual manufacturer of the tier, and requires the whole netlist to be available to the stack maker. It does not seem likely that this is going to happen given the sensitive nature of the Intellectual Property (IP) used in the tiers. Figure 4.28 shows the stack as it have been created from our three ISCAS circuits.

We performed a series of post-bond tests, in these post-bond test the stack consists of all three tiers. The tiers are put in parallel post-bond mode, we vary the Turn/Elevate and Intest/Bypass/(Extest) settings to test the tier we want to test. In Figure 4.3 the Turn or Elevate status of the core is indicated by the name, for example the ‘s1423 ET’ indicates we are testing the s1423 tier, and the stack configured such that the bottom tier (s5378) is in Elevate mode and s1423 is in Turn mode. Because s1423 is in Turn mode, the mode of s400 is of no consequence

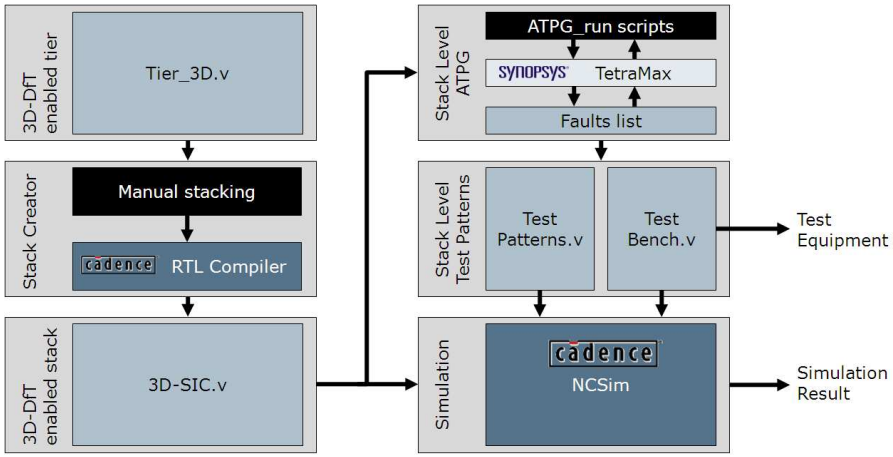


Figure 4.27: Actual tool flow for our 3D-DfT equipped 3D-SIC validation experiment.

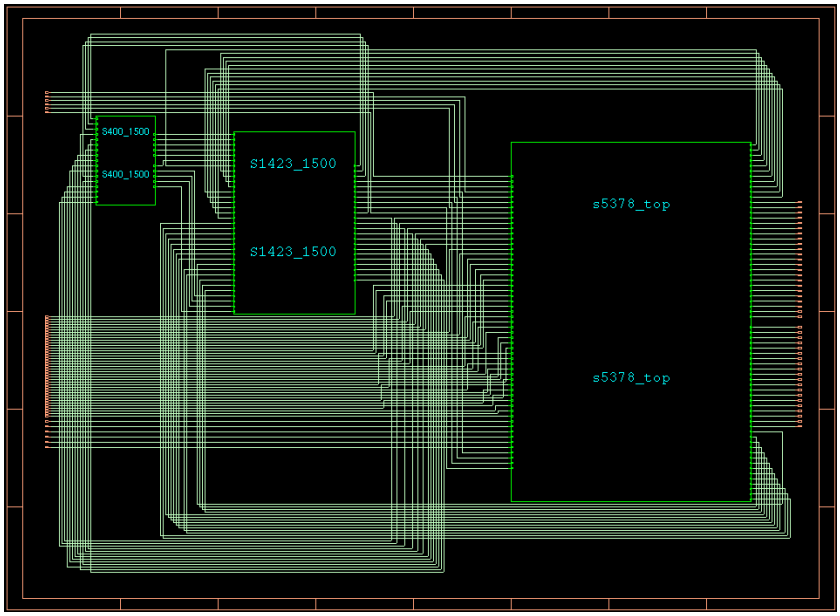


Figure 4.28: 3D stack viewed in RTL Compiler.

as the test flow of data does not go through s400. This differs from the ‘s1423 EET’ case where the same tier is actively tested, but as the flow of test data for this tier now also passes the s400 tier, the fault coverage increases as slightly more hardware is exposed to the test data. In this experiment a core in Turn mode is also in Intest mode, while one in Elevate mode is also in Bypass mode.

Finally we perform an ATPG run with all the tiers actively taking part in the test, the tiers are in Parallel Post-bond mode. The top wafer (s400) is in Turn mode to return the signals, while the others are in Elevate mode. In Figure 4.4 the results with all tiers in Extest as well as Intest have been given. The s7201 result is the stack result, 400+1423+5378=7201. In Extest we are

Post-bond tests	Total Faults	Detected	Coverage	Patterns	Test Cycles
$s400EET$	22068	1433	6.5%	50	784
$s1423ET$	22068	4571	20.7%	105	3835
$s1423EET$	22068	4599	20.8%	116	4465
$s5378T$	22068	15735	71.3%	368	32860
$s5378ET$	22068	15811	71.6%	364	33234
$s5378EET$	22068	15885	72.0%	351	32775

Table 4.3: Post-bond ATPG results, only one tier is actively being tested.

effectively performing the interconnect test, but even though the internal Scan chains are not active, TetraMax still manages to cover some of the faults inside the tiers. The s7201 total is therefore not just addition of the Extest and Intest results, but it is the joining of the covered faults sets.

Post-bond tests	Total Faults	Detected	Coverage	Patterns	Test Cycles
$s7201Extest$	22068	344	1.6%		
$s7201Intest$	22068	21449	97.2%		
$s7201Total$	22068	21515	97.5%	424	56969

Table 4.4: Post-bond ATPG results, all tiers are actively being tested.

4.6.4 3D-DfT cost analysis

The costs of implementing our 3D-DfT architecture can be subdivided into three categories, there are costs in the additional *pads* to enable Pre-bond testing, costs for the dedicated test *TSVs* between tiers, and in the *area* the 3D-DfT hardware occupies on a tier.

This test access architecture requires $7 + 2 + 2m$ dedicated probe pads at each (non-bottom) die in the stack. As Figure 4.8(d) makes clear seven pads for the WSC, two for the serial interface and $2m$ for the parallel interface. As the parallel TAM is optional in IEEE 1500, note that m can be zero. This number of dedicated probe pads needs to be extended by all required infrastructural pads for power, ground, clocks, etc.; these are not shown in our 3D-DfT equipped tier in Figure 4.8(d), although their presence is obviously essential for every tier in the stack.

Similar to additional pads the 3D-DfT architecture requires dedicated test TSVs to each non-bottom tier, see Figure 4.8(d). Again seven TSVs for the WSC, two more for the serial access mechanism and $2n$ for the optional parallel TAM.

The addition of our 3D-DfT architecture hardware to the s400, s1423, and s5378 obviously requires area on those tiers. Close inspection of the added hardware in our experiment has taught us the foremost contributers to the area overhead for our 3D-DfT architecture can be categorized into three groups.

- F: Fixed, elements that are needed in equal amounts no matter the size of the tier.
  Consists mostly of the WIR, WBY, ScanMux, delay elements in the WSC, a 3→1 Mux, a

2→1 Mux, two FFs. The WIR size may vary with the instruction set, but not with the size or complexity of the design.

- V_{sc}:Variable with the number of Scan Chains through the core.
  Consists of two muxes, two FFs, and the additional routing hardware in the ScanMux controller for every scan chain.
- V_{io}:Variable with the number of functional I/Os the core has.
  Consists of one WBR cell per I/O.

$$AreaOverhead = F + V_{sc} + V_{io} \tag{4.1}$$

Because the size of the used elements is known, see Table 4.5, we can derive Equation **??** that predicts the overhead of our 3D-DfT architecture for a given chip design. The area per component values in Table 4.5 are rounded and hold only for the UMC90 library, the results will differ for other libraries, but the ratio between the components should remain.

	Y							**V_{sc}**	**V_{io}**
	WIR	WBY	WBY	3→1 Mux	2→1 Mux	WSC	Flip-Flop	SC_cost	IO_cost
$size$	300	18	11	21	7	47	14	63	36
	432							**63**	**36**

Table 4.5: Area of various UMC 90 components.

$$AreaOverhead = Y\_cost + \#SC \cdot SC\_cost + \#IO \cdot IO\_cost$$
$$s400AreaOverhead = 432 + 3 \cdot 63 + 9 \cdot 36 = 945$$

Table 4.6 shows the area overhead the 3D-DfT architecture has on the ISCAS designs in our experiment. The overhead on the s400 especially is quite dramatic at 90%, however the s400 is a very small design, so even though the hardware required for the 3D-DfT architecture is quite small, it has a large impact. Using Equation (4.1) we also calculated the area overhead on a much bigger commercial chip design the PNX8550 [23], a Philips media chip for digital television. On this chip with an area of 40M μm^2 the area overhead of less the 15K μm^2 for the 3D-DfT hardware is very small at less then 0.05%

	specification			**overhead**		
tier	scan chains	I/O	2D area(μm^2)	predicted (μm^2)	actual (μm^2)	relative increase
$s400$	3	9	1044	945	942	90%
$s1423$	3	22	3748	1413	1411	38%
$s5378$	3	84	11751	3645	3645	31%
$pnx8550$	140	280	40M	19332		0.048%

Table 4.6: Area impact of our 3D-DfT Architecture.

4.7 Recommendation

During the validation experiments some elements of the Candence IEEE 1500 solution where not used, in Figure 4.29 these elements are marked in red, while in green the 3D-DfT components are marked. The Scan_Enable signal is already part of the WSC, and thus serves no additional purpose, also Cadence included a second set of control signals for the parallel interface. These Wrapper Parallel Control signals also bring no new functionality as the WIR already provides full control over the parallel interface via the instructions placed in its instruction register. We therefore see no need in either of those two signals to be implemented, and could be removed from the Cadence Test Encounter Tool.

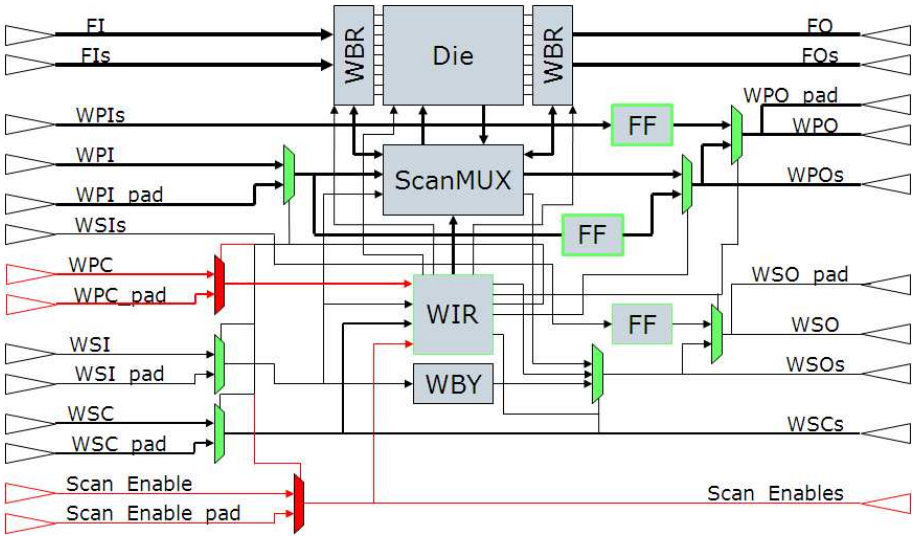


Figure 4.29: High level diagram of the Cadence IEEE1500 implementation, in *green* new 3D-DfT hardware, in *red* hardware that was not used.

The front end to most of the test related functionality provided by the Cadence tool set is *RTL Compiler*, yet for the insertion of IEEE 1500, *Test Encounter* is used. Most likely this work around is due to the relative short time IEEE 1500 has been available in the tool set. We recommend the functionality extended with the 3D-DfT architecture to be available in RTL Compiler directly, and in fact move the insertion of 3D-DfT hardware in front of scan chain insertion. The test coverage will be increased by including the 3D-DfT hardware itself, although caution must be applied not to impede on the functionality of the 3D-DfT hardware. Figure 4.30 reflects the proposed changes in tool flow and functionality.

Test pattern generation could also be improved if the software tool in question would support 3D-DfT and the multiple modes it is required to run in order to get a realistic fault coverage. This added functionality for Synopsys TetraMax is shown in Figure 4.31 and Figure 4.32. Where in case of the stack it would also be beneficial if the ATPG tool would understand, how the tiers would interact, without actually providing it with a netlist that includes a top-level design that holds these tiers. It is entirely possible to run a ATPG test on a tier, having only a netlist of that tier, and how to get access to that tier, without knowing the netlists of every other tier in the

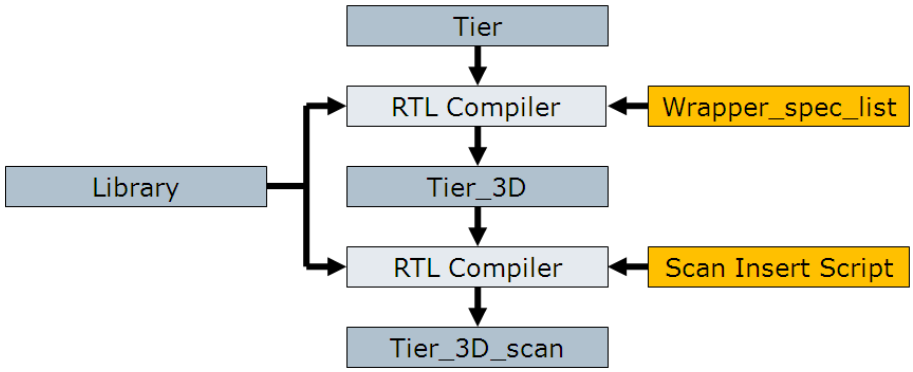


Figure 4.30: Proposed flow diagram for 3D-DfT insertion through Cadence tools stack.

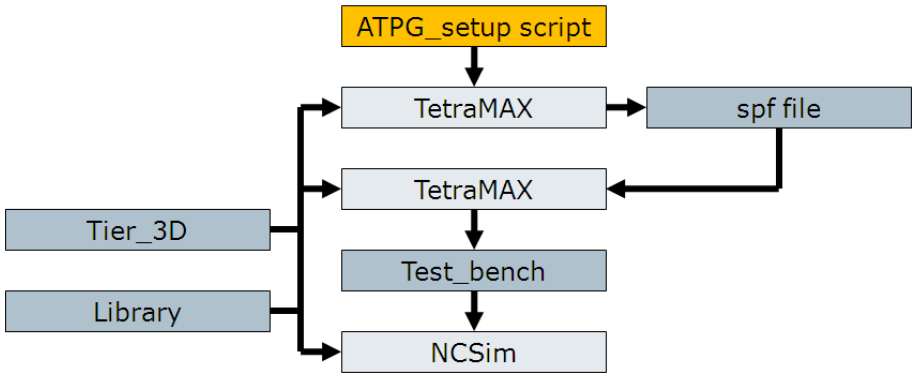


Figure 4.31: Proposed ATPG flow for a 3D-DfT equipped tier.

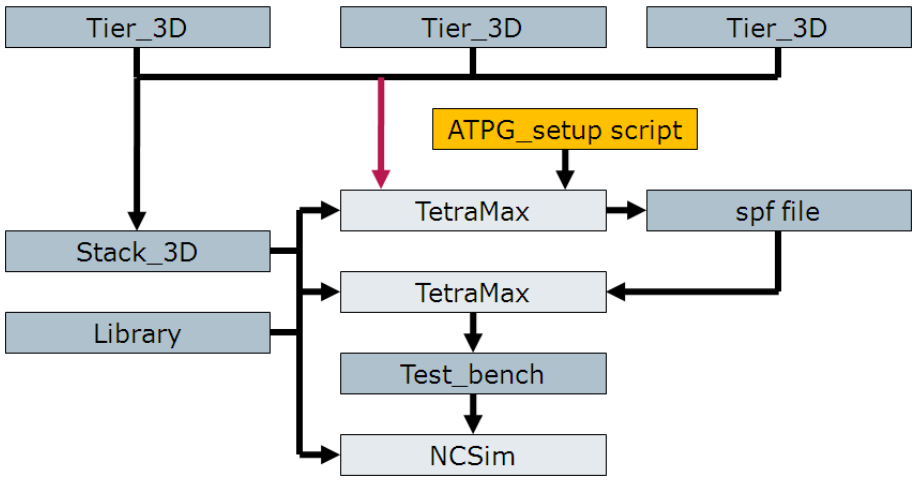


Figure 4.32: Proposed flow diagram for ATPG testing of a 3D-SIC with our 3D-DfT architecture.

We can now have a close look at what is actually required from the Cadence tool set to achieve

a full 3D-DfT implementation within the tool environment. Figure 4.33 shows the specifications and scripts provided to RTL Compiler in such an event. The WIR_spec_list provides the tool with requirements for WIR, we recommend to support not only the compulsory modes and instructions but to allow for user defined instructions. Why not have the user providing the information in Figure 4.34, a matrix of opcodes/instructions and the required behavior of the WIR. This would allow for much flexibility in supporting additional modes without the need for the tool set to anticipate the functionality before hand. An example of this functionality would be a designer using the 3D WIR to generate a signal to control an embedded instrument, providing JTAG like functionality without any additional cost.

The 1500.3D wrapper list describes RTL Compiler with information about the wrapper itself, this could involve TSV related requirements of describe the type of WBR cells to be used. The ScanMux_list gives information about what functionality ScanMux should provide, this is linked to the signals provided by the WIR. Modes like ‘Extest’ and ‘Intest’ are handled by Scanmux already, and this should continue to be the case. Most other 3D-DfT modes do not involve the ScanMux, and are focused on access to and from other tiers, instead of internal routing that involves ScanMux. However it is likely the user defined modes do, for example, a user may want to split the WBR into a top and a bottom section. Having a separate bottom WBR section would still provide Interconnect test functionality with the lower die, while shorting the chain by not involving WBR cells the do not have a connecting to that lower die.

The Scan_insert_script retains it’s current function, providing RTL compiler with scan chain specific information. The only addition is that because it is now performed after the insertion of the 3D-DfT components. It should not incorporate the WIR instruction register into a scan chain, doing so would change the mode of the tier while scanning, leading to a nearly untestable tier. The functionality to exclude specific registers of flip-flops is available in RTL Compiler, but it would be nice to have specific support to exclude WIR instruction registers.

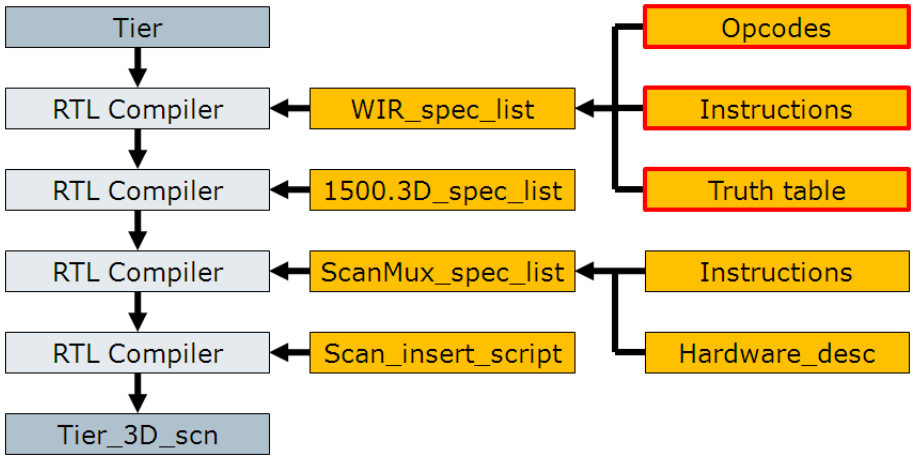
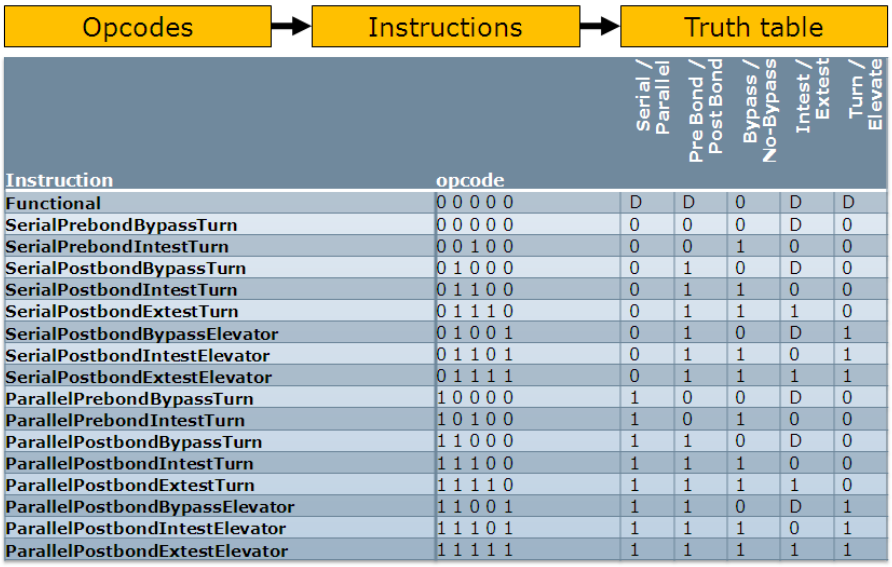


Figure 4.33: Streamlined approach for 3D-DfT insertion, with added functionality.



Instruction	opcode	Serial / Parallel	Pre Bond / Post Bond	Bypass / No-Bypass	Intest / Extest	Turn / Elevate
Functional	0 0 0 0 0	D	D	0	D	D
SerialPrebondBypassTurn	0 0 0 0 0	0	0	0	D	0
SerialPrebondIntestTurn	0 0 1 0 0	0	0	1	0	0
SerialPostbondBypassTurn	0 1 0 0 0	0	1	0	D	0
SerialPostbondIntestTurn	0 1 1 0 0	0	1	1	0	0
SerialPostbondExtestTurn	0 1 1 1 0	0	1	1	1	0
SerialPostbondBypassElevator	0 1 0 0 1	0	1	0	D	1
SerialPostbondIntestElevator	0 1 1 0 1	0	1	1	0	1
SerialPostbondExtestElevator	0 1 1 1 1	0	1	1	1	1
ParallelPrebondBypassTurn	1 0 0 0 0	1	0	0	D	0
ParallelPrebondIntestTurn	1 0 1 0 0	1	0	1	0	0
ParallelPostbondBypassTurn	1 1 0 0 0	1	1	0	D	0
ParallelPostbondIntestTurn	1 1 1 0 0	1	1	1	0	0
ParallelPostbondExtestTurn	1 1 1 1 0	1	1	1	1	0
ParallelPostbondBypassElevator	1 1 0 0 1	1	1	0	D	1
ParallelPostbondIntestElevator	1 1 1 0 1	1	1	1	0	1
ParallelPostbondExtestElevator	1 1 1 1 1	1	1	1	1	1

Figure 4.34: A closer look a the functionality provided by the 3D WIR.

4.8 Conclusion

In this chapter, we presented a generic test access architecture for TSV-based 3D-SICs. The architecture supports a modular test approach, in which dies and their embedded cores, as well as inter-die interconnects, can be tested separately. The architecture leverages (1) existing intra-die DfT features such as internal scan, test data compression, built-in self-test, and core-based wrappers and TAMs, as well as (2) boundary scan at the 3D-SIC's PCB interface.

A main component of the 3D-SIC test access architecture is a die-level wrapper. This wrapper is based on IEEE 1500, extended with four novel features:(1) TestTurns, the notion that for 3D-SICs test signals should eventually be routed to the same surface they originated from(2) dedicated probe pads for all IEEE 1500 test control and data signals on non-bottom dies, to facilitate pre-bond die testing; (3) TestElevators that transport the IEEE 1500 test control and data signals up and down during post-bond stack testing; and (4) a hierarchical WIR chain to prevent unbridled growth of its length.

The proposed architecture is *structured*, as it provides a common DfT template that meets all 3D-SIC test access requirements. The proposed architecture is also *scalable*, in the sense that it works for all stacks heights and provides user-defined test access bandwidth; the latter provides a trade-off opportunity between silicon area and test length. Future work is to automate the EDA tool flow for DfT insertion and test expansion, and to exploit the optimization opportunities that are offered, by careful parameter selection for the switch box in the die-level wrappers.

The architecture has been verified by simulating a 3D-Dft enabled 3D stack in current 2D EDA tools. By doing so proving the concepts behind the 3D-DfT architecture, their impact in terms of additional TSVs and probe pads, and other costs such as a reduction of the functional area on the chip. Current EDA tools did provide some of the features necessitated by the proposed

3D-DfT; The verification process also highlights those areas in which EDA tools need upgrading to automatically support 3D-DfT insertion into a design.

Given that it is likely that multiple companies contribute to the manufacturing of a single 3D-SIC, standardization of the die-level test access features makes sense. The proposed test access architecture serves as an excellent starting point for standardization: “IEEE 1500.3D”. It is based on the already existing standard IEEE 1500, and only requires a few specific extensions of that standard. Prove of concept of this new standard has been provided in this chapter, even without direct tool-support verification was possible, showing the benefit and effectiveness of leveraging excising DfT standards. It has also shown the fault coverage for a given tier is unaffected while, keeping the area overhead low, making it a viable option.

Standardization might require to fix the parameters n and m to standardized values, such as 16, 32, or 64. For real plug-and-play test interoperability between dies from different sources, it is required that the inter-die test interconnections are defined not only electrically, but also with respect to their x, y layout locations. If a single DfT standard is to serve 3D-SICs of different footprint size, it seems beneficial to concentrate the TestElevators in the center of their layout. A 3D-SIC DfT standard should also be accompanied by a description format for transfer of DfT and test ‘knowledge’ like the IEEE 1149.1 Boundary Scan Description Language (BSDL) [14, 15] and/or IEEE 1450.6 Core Test Language (CTL) [31, 32].

This chapter focused on a digital test access architecture for factory production tests. Future work includes extension of the architecture to support debug and diagnosis, making embedded test instruments available to system-level 3D-SIC users, and inclusion of analog tests.

5 Conclusion

In this chapter we conclude the master thesis. This chapter summarizes the contributions to 3D-SIC testing presented in Chapters 3 and 4.

This Conclusion is organized into two sections, Section 5.1 covers contributions in the area of 3D-SIC yield improvement for W2W stacking, and the 3D-DfT architecture to enable testing of 3D-SICs. Section 5.2 comprises of suggestions for the progression of the work presented in this thesis, further investigations in wafer matching strategies and repository setups. The 3D-DfT architecture can be extended further, and serve as a basis for a 3D-SIC testing standard, also included is a set of recommendation to include 3D-DfT functionality into commercially available EDA tools.

5.1 Contributions

In this thesis we have described our two main contributions to the field of computer chip testing, in particular the newly forming field of 3D Stacked IC testing. One contribution is a study into Wafer-to-Wafer matching a yield improvement study for W2W stacking, the other was the development of a IEEE 1500-based 3D-DfT access architecture and its verification.

In regards to field of wafer matching our contributions has been the following:

- We have captured the concept of wafer matching for the W2W stacking method into a close-loop mathematical yield model. This mathematical model showed that the yield improvement of wafer matching depends on (1) the number of stack tiers, (2) the number of dies per wafer, (3) the die yield, and (4) the wafer repository size. In addition, the model gave a fundamental insight into how W2W matching actually functions, as based on probability theory.
- A simulator has been created to further investigate wafer matching in a realistic production environment, to quantify trends and make it possible to accurately predict the yield benefit wafer matching would provide given a set of production details. We presented simulation results, focusing especially on the effects of a varying repository size. We have provided more accurate simulation results compared to the prior work [1, 2].
- Our simulation results showed that for a realistic production process and reasonable repository sizes of 25 to 50 wafers (one or two cassettes) that relative yield improvements of 0.5% to 10% can be achieved. Furthermore the simulation results indicate how the yield improvement is affected by the parameters identified in the mathematical model.

- A cost-benefit analysis of W2W matching has been worked out. The resulting cost model has shown that despite the costs involved, W2W matching can attribute to a reduction in overall cost per functioning 3D-SIC of 2-50% depending on the stacked yield and chosen W2W test flow.

This thesis also covered a 3D-DfT architecture for TSV-based 3D-SICs based on the IEEE 1500 /citeIEEE1500 wrapper for SOC testing. Primarily the architecture is focused on digital testability of 3D-SICs by providing scan access to every tier in the stack without having physical access to that particular tier. However with the 3D-DfT architecture we also want to provide a flexible standardizable template for testing 3D-SICs. A standard would allow for multiple companies to contribute tiers for a single 3D-SIC, without being intimately familiar with each others designs. In much the same way as PC memory standardization allows for a user to install any size of memory from any vendor, as long as it meets the memory standard stipulated by the motherboard manufacturer.

Our contribution to 3D-SIC testing is the following

- We have compiled a list of requirements and constraints for a 3D-DfT test access architecture to enable 3D-SIC testing during the various stages of its production process.
- We have presented a 3D-DfT architecture that supports a modular test approach, in which dies and their embedded cores, as well as inter-die interconnects, can be tested separately. The 3D-DfT architecture builds on the IEEE 1500 standard for core testing and adds four new features:(1) TestTurns, the notion that for 3D-SICs test signals should eventually be routed to the same surface they originated from(2) dedicated probe pads for all IEEE 1500 test control and data signals on non-bottom dies, to facilitate pre-bond die testing; (3) TestElevators that transport the IEEE 1500 test control and data signals up and down during post-bond stack testing; and (4) a hierarchical WIR chain to prevent unbridled growth of its length.
- A simulation of the 3D-DfT architecture has shown the validity and viability of our approach. Even in current day EDA tools is was found possible to implement some of the novel features into three ISCAS designs and simulated a 3D-SIC being tested in all defined modes of the 3D-DfT architecture. Where EDA support was unavailable, 3D-DfT features have been added to the designs by hand.
- The experiment has shown us the ability of the 3D-DfT architecture to provide the same fault coverage of a die regardless if it was ‘bare’ 2D chip, or a 3D-DfT enable chip in pre-bond, or in post-bond testing.
- The verification of the 3D-DfT architecture provided us with accurate estimations of the costs involved with implementing our 3D-DfT architecture. With only a few additional probe pads and TSVs, and an area overhead of less then 0.1% on large chips, the costs of implementing our 3D-DfT architecture is limited, making it’s implementation economical viable.
- The verification process has provides valuable information regarding the requirements for EDA tools to support the 3D-DfT and resulted into recommendations for the EDA vendors

5.2 Futurework

Currently our mathematical model for wafer matching does not support a repository of bottom wafers, and requires an natural value of faulty dies on each wafer, and an identical number of faulty dies for each wafer. Although our simulation covers these shortcomings, it too does not support clustering, the tendency of faults occurring near other faults. Wafer matching can be studied by extending the scoop further. One such extension has been a study into the effects of different matching criteria and replenishing repositories of wafers as in [33]. This may be further extended to include hybrid versions of replenishing and non-replenishing repositories to better suit the production process and machines.

Envisioned future work in regard to wafer matching:

- Extension of the mathematical model to include parameters mimic reality more closely
- Extension of simulator to include various matching strategies and replenshing repositories.

Our 3D-DfT architecture is focused purely on enabling 3D-SIC testing. In practice manufactures are reusing DfT hardware and features to support other purposes, such a reprogramming reconfigurable hardware, or real time access to embedded instruments inside the chip.

Envisioned future work in regard our 3D-DfT architecture:

- Extending the 3D-DfT architecture to also support debug and diagnosis purposes, and enabling support for embedded instruments, similar to how IEEE 1149.1 [14] is being used today.
- Working on support for a IEEE certified 3D-DfT standard, allowing for arbitrary dies to be stacked and tested without ever considering the other dies of the stack during the design process. Industry wide support would even allow the reuse of tiers into multiple stacks, even between vendors.
- Incorporation of 3D-DfT functionality into EDA tools, as indicated by our 3D-DfT verification process.

A Structured and Scalable Test Access Architecture for TSV-Based 3D Stacked ICs

A

Our paper presenting the IEEE 1500 based 3D-DfT architecture: *'A Structured and Scalable Test Access Architecture for TSV-Based 3D Stacked ICs'*

Has been accepted in the 2010 VLSI Test Symposium procedings, and has been presented during the event on May 19-22 2010.

A Structured and Scalable Test Access Architecture for TSV-Based 3D Stacked ICs

Erik Jan Marinissen[1] Jouke Verbree[2] Mario Konijnenburg[3]

[1] IMEC
SSET/DC
Kapeldreef 75
3001 Leuven, Belgium
erik.jan.marinissen@imec.be

[2] Delft University of Technology
Department of Computer Engineering
Mekelweg 4
2628CD Delft, The Netherlands
j.verbree@student.tudelft.nl

[3] Holst Centre/IMEC
SSET/WATS
High Tech Campus 31
5656AE Eindhoven, The Netherlands
mario.konijnenburg@imec-nl.nl

Abstract

New process technology developments enable the creation of three-dimensional stacked ICs (3D-SICs) interconnected by means of Through-Silicon Vias (TSVs). This paper presents a DfT test access architecture for such 3D-SICs that allows for both pre-bond die testing and post-bond stack testing. The DfT architecture is based on a modular test approach, in which the various dies, their embedded IP cores, the inter-die TSV-based interconnects, and the external I/Os can be tested as separate units to allow optimization of the 3D-SIC test flow. The architecture builds on and reuses existing DfT hardware at the core, die, and product level. It adds a die-level wrapper, which is based on IEEE 1500, with the following novel features: (1) *dedicated probe pads* on the non-bottom dies to facilitate pre-bond die testing, (2) *TestElevators* that transport test control and data signals up and down during post-bond stack testing, and (3) a *hierarchical* Wrapper Instruction Register (WIR) chain. The paper also hints at opportunities for optimization and standardization of this architecture.

1 Introduction

The semiconductor industry is preparing itself for 3D-SICs based on TSVs [1–3]. TSVs are conducting nails which extend out of the back-side of a thinned-down die and enable the vertical interconnect to another die [4, 5]. TSVs are high-density, low-capacity interconnects compared to traditional wire-bonds, and hence allow for many more interconnections between stacked dies, while operating at higher speeds and consuming less power [6]. TSV-based 3D technologies enable the creation of a new generation of 'super chips' by opening up new architectural opportunities [7, 8]. Combined with their smaller form factor and lower overall manufacturing cost, 3D-SICs have many compelling benefits, and hence their technology is quickly gaining ground.

Like all micro-electronics, TSV-based die stacks have a manufacturing process that is sensitive to defects, and hence 3D-SICs need to undergo electrical testing to ensure product quality. While the process and design technology is getting to maturity, testing 3D-SICs for manufacturing defects is considered by many as a major, still largely unresolved obstacle to make these devices a product reality. Next to all basic and most advanced test technology issues, 3D-SICs have some unique new test challenges of their own [9, 10]. These challenges include (1) development of new fault models and corresponding tests for TSV-based interconnects and new 3D-induced intra-die defects, (2) wafer probing on small and numerous micro-bumps and/or TSV tips and pads under stringent damage requirements, (3) handling of and probing on wafers with thinned-die stacks, (4) the design, partitioning, and optimization of DfT architectures that span across multiple dies, and (5) optimization of the test flow for maximum effectiveness and lowest cost.

In this paper, we focus on the design of a structured and scalable test access architecture. The architecture supports pre-bond die testing, post-bond stack testing, and final packaged-product testing. It allows for modular testing [11] of intra-die circuitry and inter-die TSV-based interconnects, and hence supports stacks of possibly heterogeneous and/or black-boxed dies, flexible test flow creation during the various maturity stages of a 3D-SIC product, and easy diagnosis. The architecture reuses commonly encountered design-for-test structures within the various dies as much as possible. It leverages two existing design-for-test standards, viz. IEEE 1149.1 [12, 13] for chips on a Printed Circuit Board (PCB) and IEEE 1500 [14, 15] for embedded cores within an System-On-Chip (SOC). The test access architecture initially targets 3D-SICs consisting of scan-testable digital logic and memory dies, but can be extended to include other types of dies as well.

The remainder of this paper is organized as follows. Section 2 describes related prior work in test access architectures for 3D-SICs. Section 3 provides an overview of test access architecture standards for PCBs and 2D-SOCs, which, like 3D-SICs, are also built from interconnected smaller components. The test requirements and constraints which are unique to 3D-SICs are discussed in Section 4. Our proposed 3D-SIC test access architecture is presented in Section 5. Section 6 concludes this paper.

2 Related Prior Work

The first paper dedicated to testability of 3D-SICs was [16]. It focuses on pre-bond die testing, required to achieve acceptable compound stack yields. Testing incomplete products as formed by the various stack tiers is identified as a potential problem. In our opinion, this does *not* have to pose a problem really, provided that (1) structural, not functional tests are applied, (2) a modular test strategy is followed, and (3) the infrastructure (power/ground, clocks) can be made operational per die. [16] proposes a 'scan island' approach, which is essentially the wrapper technology from IEEE 1149.1 [12, 13] and IEEE 1500 [11, 14, 15] under a different name.

Most other papers on 3D-SIC testing implicitly propose a test access architecture, while focusing on optimizing the design parameters of that ar-

chitecture to minimize the resulting test length and/or the associated wire length. Wu et al. [17] describe three scan chain optimization approaches for 3D-SICs. Implicitly, this paper assumes that a single logic test unit is partitioned over multiple tiers, which seems rather unrealistic. Therefore, in [18], Wu et al. propose a core-based design and test approach (as common for 2D-SOCs) in which each core resides on a single tier. The paper proposes an ILP-based Test Access Mechanism (TAM) optimization approach, which tries to minimize the resulting test length under a constraint for the number of additional 'test TSVs'. Both papers [17, 18] focus exclusively on post-bond stack testing, and ignore the requirements for pre-bond die testing.

Jiang et al. [19] describe a TAM optimization approach based on simulated annealing that minimizes test length and TAM wire length with a user-defined cost weight factor. They assume a modular core-based 3D-SIC test approach and take both pre-bond and post-bond test lengths into account. The paper lacks realistic constraints on wafer and packaged stack test access, due to which it unrealistically allows TAMs to start and end at any stack tier. Successor paper [20] remedies this partly, by working with pre-bond tests that are applied through dedicated probe pads at the die in question, for which a maximum count is assumed. The paper proposes heuristics that determine a post-bond stack test architecture, from which segments are reused as much as possible to build additional die-level test architectures for the pre-bond tests, while meeting the maximum probe pad count constraint and minimizing test length and TAM wire length.

In contrast to the prior work by others, our paper starts out by identifying realistic constraints and requirements set forward by, among others, wafer probe technology and test flow set-ups. Subsequently, we focus on the design of a generic and structured test access architecture. The architecture is scalable in the sense that its design parameters can be optimized for varying core, die, and stack parameters, but the focus of our paper is *not* on those optimization procedures. The prior work published until now has focused on testing the cores in the various dies constituting the 3D-SIC, but has ignored testing the circuitry within a die in between the cores, as well as it has ignored testing the (TSV-based) inter-die interconnects. The prior work also did not identify how existing DfT standards and test access architectures can be leveraged. Finally, test control and instructions were ignored in the prior work. We address all the above issues.

3 Related Test Access Standards

3.1 Test Access Architecture for PCBs

The commonly-used test access architecture for PCBs is based on IEEE Std. 1149.1, Boundary Scan (a.k.a. 'JTAG') [12, 13]. In order for chips to be compliant to IEEE 1149.1, a small hardware wrapper is added to them. IEEE 1149.1 works through a narrow single-bit interface, as every JTAG terminal requires an additional chip pin and these are considered expensive. Fortunately, the prime focus of IEEE 1149.1 is PCB interconnect testing, and that requires only a small number of test patterns [21]. The single-bit interface pins are called TDI and TDO, and they transport both instructions and test data. The control interface consists of the pins TCK, TMS (and optionally TRSTN). For an example PCB containing three chips, a common JTAG-based test access architecture is depicted in Figure 1. The control signals are broadcast to all chips, while the TDI-TDO pins are concatenated through the chips. The broadcast control signals can configure the TAP Controller finite state machine in a mode in which it is willing to receive instructions, which are subsequently scanned into the Instruction Register (IR) via the daisychained TDI-TDO interface. Note that this allows for different instructions for different chips; for example, Chip B can be configured in INTEST mode, while Chips A and C are configured in BYPASS mode. Then, the chips are brought into their instructed test modes via the broadcast control signals and test data is scanned in and out again via the daisychained TDI-TDO interface. The selected test data register (e.g., the bypass register, a Boundary Scan Register (BSR), or a chip-internal scan chain) depends on the instruction, and can be different for different chips; in any case, it is a single shift register, as shown in Figure 1.

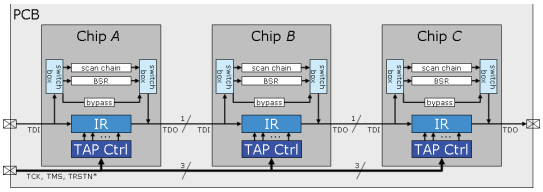


Figure 1: Board-level test access architecture for chips based on IEEE 1149.1.

3.2 Test Access Architecture for 2D-SOCs

The commonly-used test access architecture for (two-dimensional) SOCs containing embedded IP cores is based on IEEE Std. 1500 [11, 14, 15]. Like IEEE 1149.1, IEEE 1500 adds a small hardware wrapper around the module-under-test. As shown in Figure 2, the test access architecture for an IEEE 1500-based SOC shows similarities to IEEE 1149.1-based PCBs. Control signals are broadcast to all cores. Once configured in the appropriate mode, instructions are shifted into the Wrapper Instruction Register (WIR) via the daisychained WSI-WSO interface. That same instruction interface also doubles as single-bit test data interface. However, next to the similarities, there are also significant differences between IEEE 1149.1- and IEEE 1500-based test access architectures. Below, we list the most important ones.

- Unlike IEEE 1149.1, the focus of IEEE 1500 is not (only) on testing wiring interconnects between cores. First of all, the interconnect circuitry in between IP cores typically does not consist only of wires, but is often formed by deep sequential logic [22]. In addition, IEEE 1500 is meant to support also the testing of the cores themselves, and IP cores are often significantly-sized and complex design entities. Therefore, the test data volumes involved are typically quite large, and a single-bit test data interface would not suffice. Hence, IEEE 1500 has an optional n-bit ('parallel') test data interface (named WPI and WPO), where n can be scaled by the user to match the test data volume needs of the IP core in question.
- Adding wider interfaces to embedded IP cores does not add chip pins as in IEEE 1149.1, but only core terminals; and they are considered to be significantly less expensive than chip pins.
- IEEE 1149.1 has two (or three) standardized control pins, which are expanded within the chip by the TAP Controller. IEEE 1500 has no TAP Controller, but receives it control signals directly. These are six (or seven) signals: WRCK, WRSTN, SELECTWIR, SHIFTWR, CAPTUREWR, UPDATEWR (and optionally TRANSFERDR) [11, 14, 15].

Figure 2 also features a parallel wrapper bypass. This bypass is not mandated by IEEE 1500, but often implemented to shorten the test access path

to other cores in the same TAM [23]. It is the task of the *switch boxes* in Figure 2 to make an effective mapping between the active WIR instruction mode and the TAM-to-chain connections.

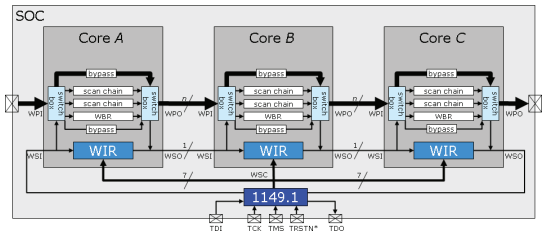


Figure 2: SOC-level test access architecture for cores based on IEEE 1500.

IEEE 1500 only standardizes the core-level test wrapper, and *not* the SOC-level test access architecture of the optional parallel TAMs. At the SOC-level, optimizations can be made w.r.t. TAM type [24, 25], TAM architecture [23], and corresponding test schedule. In a tyical implementation, as shown in Figure 2, the SOC itself is equipped with an IEEE 1149.1 wrapper to facilitate board-level testing. The IEEE 1500 serial interface (WSC, WSI, and WSO) is multiplexed onto the IEEE 1149.1 Test Access Port [15] to save otherwise additional test pins. The IEEE 1500 parallel interface (WPI and WPO) can be multiplexed onto the functional external pins, as is common for regular scan chains; this saves otherwise additional test pins.

4 Requirements and Constraints

We consider three types of 3D-SICs; examples of these types (in this case for stacks of three tiers) are depicted in Figure 3. The three types differ in their connections to the external world (*'pins'*): (a) wire-bond from the top die, (b) wire-bond from the bottom die, and (c) flip-chip connections from the bottom die. All three types have in common that only one side of one of the extreme tiers (top or bottom) holds all external connections. In the remainder of this paper we assume all external connections are in the *bottom* die. This assumption is without loss of generality, as we can always swap the references to top and bottom die.

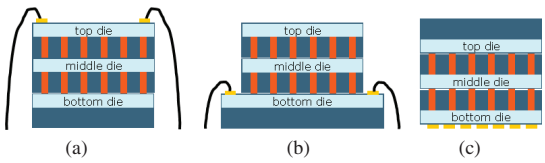


Figure 3: Three options for 3D-SIC external connections: (a) wire-bond from top die, (b) wire-bond from bottom die, and (c) flip-chip from bottom die.

A 3D-SIC test flow consists of (1) *pre-bond die tests* and (2) *post-bond stack tests* [10]. The pre-bond die tests are wafer tests; the post-bond stack tests can be carried out on both unpackaged as well as packaged stacks. A test of a stack might consist of (re-)tests of the various dies, as well as tests of the TSV-based interconnects between the dies. A 3D-SIC test access architecture should support all these tests. While testing unpackaged stacks, it should be possible not only to test the complete stack, but also to test partial stacks. Furthermore the test access architecture should also support external interconnect testing, once the 3D-SIC is mounted on a board.

We want the 3D-SIC test to be a *modular* test [11, 23], as opposed to a test in which the entire stack is tested as one monolithic entity. A modular test considers the various dies and TSV-based interconnect layers as separate test units; for complex dies, it is very well possible that they are further sub-divided in multiple finer-grain test modules, e.g., embedded cores. Modular testing for 3D-SICs comes with the same benefits as it brings to 2D-SOCs [11]: (1) different tests for various modules of heterogeneous products, (2) test of black-boxed IP, (3) divide-and-conquer test generation and application, and (4) test reuse. Modular testing provides two more benefits specific to 3D-SICs: (5) flexibility in optimizing the test set per step of the test flow ("how often do we re-test a module?"), and (6) first-order diagnosis ("which module of the stack contains the fault?"). The latter is all the more important given the likelihood that multiple companies contribute to the manufacturing of a single 3D-SIC. Modular testing requires DfT in the form of *wrappers* that provide controllability and observability at the boundary of the module-under-test and *Test Access Mechanisms (TAMs)* that transport the test data from the chip's probe pads or pins to the module-under-test and vice versa [11, 25].

We assume a 3D-SIC of which the constituting dies are *scan testable*; for example, this can include scan-tested digital logic, BIST-ed embedded memories, or even scan-enabled analog cores. Furthermore, we assume it is a requirement for board-level interconnect testing that the overall product is IEEE 1149.1 compliant on its external pins. We assume that additional external test pins beyond IEEE 1149.1 are expensive and hence should be avoided. In contrast, we assume that some additional TSV-based interconnects between tiers for the purpose of test are relatively affordable; e.g., IMEC's *via-middle* TSVs are made at a 10μm minimum pitch [4, 5].

The test access architecture should be able to provide a trade-off between additional area cost for DfT, test generation effort, and test length. To minimize silicon area, we want to re-use the existing intra-die DfT infrastructure as much as possible: internal scan chains, test control, test data compression circuitry, built-in self-test, etc. To minimize the test generation effort, we prefer to base our die-level wrapper on the existing IEEE Std. 1500, given its scalable TAM width and flexible WIR. The test access architecture should allow for flexible test scheduling to minimize the test length. It is also a requirement that the test access architecture itself is testable. It is desirable that this can be done without depending on the correct functionality of the existing DfT inside the local dies and embedded IP cores.

The bottom die can be probed on the wire-bond or flip-chip pads for its external connections; that is 'business-as-usual'. For their pre-bond die test, the other (non-bottom) dies need to be probed as well. Today's probe technology is insufficient to provide probe access on TSV tips and TSV landing pads [10]; they are too small and too fragile. Hence, for all non-bottom dies, it is a requirement to provide dedicated *probe pads* for pre-bond wafer test access [10, 16, 20].

For the post-bond stack tests, test access is only possible via the bottom die. This implies that signals for test control and test data exclusively come from and go to the bottom die, and hence have a 'u-turn' type of shape; we refer to these as *TestTurns*. Also, in order to reach dies higher up in the stack, all test signals have to be transported up and down through a new type of DfT hardware that includes TSVs and which we refer to as *TestElevators*.

2D-SOCs allow us to design DfT features in the SOC circuitry around (i.e., outside) the embedded cores. This is not the case for 3D-SICs; all DfT needs to be in the various dies. The only thing that exists outside the dies are vertical interconnects, and even those need to be pre-designed in terms of die-level features, such as TSVs and TSV landing pads. This im-

plies that wrappers, TAMs, and their control signaling all needs to be predesigned in the die; not only for *that* die, but also for the dies above it in the stack. Hence, we assume that for all tiers, the DfT is designed in adherence to a pre-defined test access architecture, or that we have the freedom to modify the DfT circuitry; it cannot be added as an after-thought.

We require the test access architecture to be *scalable*, in the sense that it works for an undetermined number of stack tiers. Also, the architecture should not predestine a die to a certain tier level, such that dies that adhere to the architecture can function at any level in the stack hierarchy. Exceptions to this requirement are formed by the bottom die, which with its external connections is obviously predestined as bottom, and possibly the top die.

5 3D-SIC Test Access Architecture

5.1 Die-Level Wrapper

The test access architecture we propose for 3D-SICs is based on a die-level wrapper, which is an extended version of IEEE 1500. The die-level wrapper provides a consistent external interface to other dies in the stack, while internally within the die, it connects up to the existing functional circuitry and regular intra-die DfT. The architecture uses a limited, scalable number of dedicated TSV-based interconnections between dies in addition to the already existing functional interconnects.

Figure 4 shows a schematic overview of the DfT features and additional interconnects for an arbitrary Die x in the middle of a stack. The figure abstracts from the functional circuitry and interconnects. It shows two internal scan chains, which are representative for the possible die-internal DfT, such as any number of scan chains for a monolithic design, TAMs for a core-based SOC design, and/or BIST-ed logic or memory. Die x is equipped with an IEEE 1500-like wrapper that is normally encountered with embedded IP cores. The figure shows the conventional IEEE 1500 features of that die-level wrapper: a seven-bit Wrapper Serial Control (WSC), a Wrapper Instruction Register (WIR), a Wrapper Boundary Register (WBR), a serial WSI-WSO interface for instructions and low-bandwidth test data, and parallel WPI-WPO interface for test data. Note that it is mandatory that the entire IEEE 1500 interface is situated at the bottom side of the die.

Our die-level wrapper has three 3D-SIC-specific features (Items 1 and 2 are highlighted in orange in Figure 4):

1. All control and data signals of IEEE 1500 (WSC, WSI, WSO, WPI, and WPO) not only enter or exit Die x via TSV-based interconnects from/to the die below Die x for post-bond stack testing, but are also equipped with dedicated probe pads for pre-bond die testing. In Figure 4, for easy figure layout, these probe pads are drawn on the bottom side; however, that does not imply that these probe pads need to be physically located at the bottom side of the die. Note that the width of the parallel interface WPI-WPO might be chosen differently for the TSV interconnects (n) and probe pads (m).
2. All control and data signals of IEEE 1500 can be transferred to a die above Die x via a set of signals with identical names, post-fixed with the letter 's' (for 'stack'): WSCs, WSIs, WSOs, WPIs, and WPOs. These signals are called TestElevators, and are all situated on the top side of the die.
3. To prevent unbridled lengthy WIR chains, we propose hierarchical WIR chains. This is further described in Section 5.2 and depicted in Figure 7.

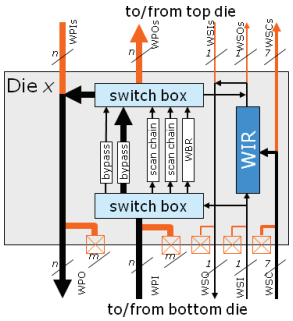


Figure 4: Schematic view of the die-level wrapper based on IEEE 1500.

The DfT in a bottom die differs from a middle die in the following aspects.

- Dedicated pre-bond probe pads are not required. Instead, the functional external I/O pads can be used for probe access.
- The bottom die is equipped with IEEE 1149.1 to facilitate board-level testing and provide a board-level test and debug port. The JTAG boundary scan chain includes all external I/Os of the 3D-SIC product.
- The serial IEEE 1500 interface (WSC, WSI, and WSO) can be multiplexed onto the IEEE 1149.1 Test Access Port (TAP), similar to what is common in 2D-SOCs [15]. This saves otherwise dedicated pads, and makes the 3D test access architecture accessible even when the 3D-SIC is soldered onto a PCB.
- The parallel IEEE 1500 interface (WPI and WPO) is multiplexed onto the functional external I/O pads, similar to what is common for scan chains and parallel TAMs in 2D-SOCs. This saves otherwise dedicated pads, but restricts the TestElevator width to the available functional I/O.

The DfT in a top die differs from a middle die in the following aspect.

- The die does not have TSV-based interconnects to an even higher-level die, as it is the top die. Hence, the top-side TestElevators WSCs, WSIs, WSOs, WPIs, and WPOs are absent.

Figure 5 depicts the test access architecture for an example 3D-SIC containing three dies; Dies 1, 2, and 3 are respectively the bottom, middle, and top die of the stack. To show the similarities and differences with test access architectures for PCBs and 2D-SOCs, the dies are shown next to each other, instead of as a vertical stack.

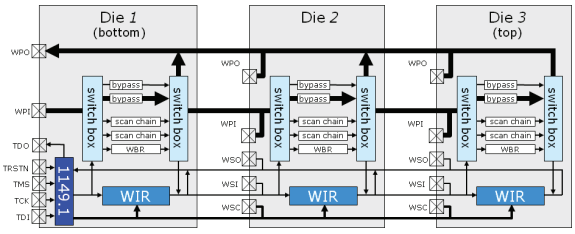


Figure 5: 3D-SIC test access architecture for dies based on IEEE 1500.

This test access architecture requires $7 + 2 + 2m$ dedicated probe pads at each (non-bottom) die in the stack. As the parallel TAM is optional in IEEE 1500, note that m can be zero. This number of dedicated probe pads needs to be extended by all required infrastructural pads for power, ground, clocks, etc.; these are not shown in Figure 4, although their presence is obviously essential.

IEEE 1500 allows various types of wrapper cells in its WBR. Embedded cores in 2D-SOCs commonly use the cell depicted in Figure 6(a); it consists of only a single flip-flop and hence occupies little silicon area. For the WBR chain of our proposed 3D-SIC die-level wrapper, we prefer to use the (also IEEE 1500-compliant) double flip-flop wrapper cell shown in Figure 6(b). At the expense of an extra flip-flop, this wrapper cell provides ripple-protection during shift mode, which seems appropriate especially if the various dies come from different sources, and ripple-during-shift might result in unwanted signal combinations at the inter-die interfaces.

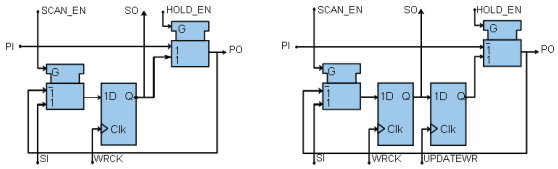


Figure 6: IEEE 1500-compliant WBR cells, (a) commonly used for embedded cores in 2D-SOCs, and (b) proposed for stacked dies in 3D-SICs.

5.2 Operating Modes

The test architecture has a number of test modes, which define the following settings.

- *Functional* / *Serial* / *Parallel* – non-test vs. test modes, resp. via serial or parallel test interface
- *Prebond* / *Postbond* – usage of dedicated test pads or TestElevators
- *Bypass* / *Intest* / *Extest* – selected test data register: bypass, all chains, or only the WBR chain
- *Turn* / *Elevator* – test responses from this die are fed via the TestTurn directly towards the bottom die or, via TestElevators, test responses from this die are transported up and responses from a higher-level die are transported down.

This leads to the following operating modes: *Functional*; *SerialPrebondBypassTurn*, *SerialPrebondIntestTurn*, *SerialPostbondBypassTurn*, *SerialPostbondIntestTurn*, *SerialPostbondExtestTurn*, *SerialPostbondBypassElevator*, *SerialPostbondIntestElevator*, *SerialPostbondExtestElevator*; *ParallelPrebondBypassTurn*, *ParallelPrebondIntestTurn*, *ParallelPostbondBypassTurn*, *ParallelPostbondIntestTurn*, *ParallelPostbondExtestTurn*, *ParallelPostbondBypassElevator*, *ParallelPostbondIntestElevator*, *ParallelPostbondExtestElevator*. A bottom die does not implement the pre-bond operating modes, as a bottom die does not have dedicated test pads.

Combining instructions for the various dies in a stack allows us to test one, multiple, or all dies simultaneously, as well as test one, multiple, or all layers of TSV-based interconnects simultaneously. This gives the same test scheduling options as the single daisychain TAM for 2D-SOCs [26].

Loading instructions into a WIR of a die-level wrapper is similar to what is known from IEEE 1500-compliant cores in 2D-SOCs. While a new instruction is shifted into the WIR, the previous instruction remains valid; only once fully arrived in place, the new instruction is activated by pulsing the UPDATEWR signal. In IEEE 1500, the WIRs of multiple IP cores are to be concatenated in a single WIR chain, which allows different cores to be loaded with different instructions. For 3D-SICs, a single concatenated WIR chain might become very lengthy, especially in case the individual dies are core-based SOCs [27] with their own concatenated WIR chain segments. Hence, we propose an hierarchical WIR mechanism, which opens up as needed, similar to a harmonica. Initially, the WIR chain only consists of the die-level WIRs. Once loaded with die-level instructions, the core-level WIR chain segments are included in the overall WIR chain for only those dies for which one of the *InTest* instructions was given; subsequently, further core-level WIR instructions can be loaded. Figure 7 schematically shows this concept by means of an example. The orange arrows highlight the active WIR chain. In this example, Dies 2 and 3 are in an *InTest* mode and hence, the WIR chain also includes the WIRs of their cores, resp. WIRC + WIRD and WIRE + WIRF. The benefit of this hierarchical WIR mechanism is that we prevent an unbridled growth of the WIR chain length; at any moment, the WIR is as long as needed. The cost is the requirement for the user to keep track of the current WIR chain length and a more complex procedure for loading instructions.

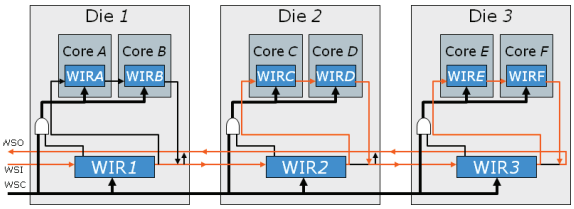


Figure 7: Hierarchical WIR chain, which has opened up for Dies 2 and 3, which are in one of their *Intest* modes.

Figures 8 and 9 show two examples of a 3D-SIC in which neighboring dies are in different operating modes. In Figure 8, Die $(x-1)$ is in its *ParallelPostbondBypassElevator* mode, while Die x is in its *ParallelPostbondIntestTurn* mode. This means that Die x is currently being tested, while the test data passes up and down in the stack through Die $(x-1)$. The orange arrows in the figure highlight the test data flow.

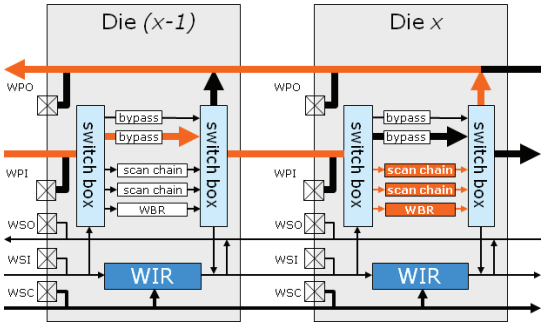


Figure 8: Example in which Die x is being tested; Die $(x-1)$ is in *ParallelPostbondBypassElevator* mode and Die x is in *ParallelPostbondIntestTurn* mode.

In Figure 9, Die $(x-1)$ is in its *ParallelPostbondExtestElevator* mode, while Die x is in its *ParallelPostbondExtestTurn* mode. This means that the TSV-based interconnects between Dies $(x-1)$ and x are currently

being tested. The orange arrows in the figure highlight the test data flow.

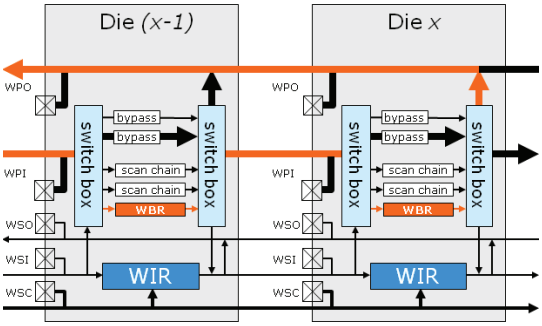


Figure 9: Example in which the TSV-based interconnects between Dies $(x-1)$ and x are being tested; Die $(x-1)$ is in *ParallelPostbondExtestElevator* mode and Die x is in *ParallelPostbondExtestTurn* mode.

6 Conclusion

In this paper, we presented a generic test access architecture for TSV-based 3D-SICs. The architecture supports a modular test approach, in which dies and their embedded cores, as well as inter-die interconnects, can be tested separately. The architecture leverages (1) existing intra-die DfT features such as internal scan, test data compression, built-in self-test, and core-based wrappers and TAMs, as well as (2) boundary scan at the 3D-SIC's PCB interface.

A main component of the 3D-SIC test access architecture is a die-level wrapper. This wrapper is based on IEEE 1500, extended with three novel features: (1) dedicated probe pads for all IEEE 1500 test control and data signals on non-bottom dies, to facilitate pre-bond die testing; (2) TestElevators that transport the IEEE 1500 test control and data signals up and down during post-bond stack testing; and (3) a hierarchical WIR chain to prevent unbridled growth of its length.

The proposed architecture is *structured*, as it provides a common DfT template that meets all 3D-SIC test access requirements. The proposed architecture is also *scalable*, in the sense that it works for all stacks heights and provides user-defined test access bandwidth; the latter provides a trade-off opportunity between silicon area and test length. Future work is to automate the EDA tool flow for DfT insertion and test expansion, and to exploit the optimization opportunities that are offered, by careful parameter selection for the switch box in the die-level wrappers.

Given that it is likely that multiple companies contribute to the manufacturing of a single 3D-SIC, standardization of the die-level test access features makes sense. The proposed test access architecture serves as an excellent starting point for standardization: "IEEE 1500.3D". It is based on the already existing standard IEEE 1500, and only requires a few specific extensions of that standard. Standardization might require to fix the parameters n and m to standardized values, such as 16, 32, or 64. For real plug-and-play test interoperability between dies from different sources, it is required that the inter-die test interconnections are defined not only electrically, but also with respect to their x, y layout locations. If a single DfT standard is to serve 3D-SICs of different footprint size, it seems beneficial to concentrate the TestElevators in the center of their layout. A 3D-SIC DfT standard should also be accompanied by a description format for transfer of DfT and test 'knowledge' like the IEEE 1149.1 Boundary Scan Description Language (BSDL) [12, 13] and/or IEEE 1450.6 Core Test Language (CTL) [28, 29].

This paper focused on a digital test access architecture for factory production tests. Future work includes extension of the architecture to support debug and diagnosis, making embedded test instruments available to system-level 3D-SIC users, and inclusion of analog tests.

References

[1] Robert S. Patti. Three-Dimensional Integrated Circuits and the Future of System-on-Chip Designs. *Proceedings of the IEEE*, 94(6):1214–1224, June 2006.

[2] Eric Beyne and Bart Swinnen. 3D System Integration Technologies. In *Proceedings of IEEE International Conference on Integrated Circuit Design and Technology (ICICDT)*, pages 1–3, June 2007.

[3] Philip Garrou, Christopher Bower, and Peter Ramm, editors. *Handbook of 3D Integration – Technology and Applications of 3D Integrated Circuits*. Wiley-VCH, Weinheim, Germany, August 2008.

[4] Bart Swinnen et al. 3D Integration by Cu-Cu Thermo-Compression Bonding of Extremely Thinned Bulk-Si Die Containing 10μm Pitch Through-Si Vias. In *Proceedings IEEE International Electron Devices Meeting (IEDM)*, pages 1–4, May 2006.

[5] Jan Van Olmen et al. 3D Stacked IC Demonstration using a Through Silicon Via First Approach. In *Proceedings IEEE International Electron Devices Meeting (IEDM)*, pages 1–4, December 2008.

[6] Kaustav Banerjee et al. 3-D ICs: A Novel Chip Design for Improving Deep-Submicrometer Interconnect Performance and Systems-on-Chip Integration. *Proceedings of the IEEE*, 89(5):602–633, May 2001.

[7] Gabriel H. Loh, Yuan Xie, and Bryan Black. Processor Design in 3D Die-Stacking Technologies. *IEEE Micro*, 27(3):31–48, May/June 2007.

[8] Roshan Weerasekera et al. Extending Systems-on-Chip to the Third Dimension: Performance, Cost and Technological Tradeoffs. In *Proceedings International Conference on Computer-Aided Design (ICCAD)*, pages 212–219, November 2007.

[9] Hsien-Hsin S. Lee and Krishnendu Chakrabarty. Test Challenges for 3D Integrated Circuits. *IEEE Design & Test of Computers*, 26(5):26–35, September/October 2009.

[10] Erik Jan Marinissen and Yervant Zorian. Testing 3D Chips Containing Through-Silicon Vias. In *Proceedings IEEE International Test Conference (ITC)*, November 2009. Paper ET1.1.

[11] Erik Jan Marinissen and Yervant Zorian. IEEE 1500 Enables Modular SOC Testing. *IEEE Design & Test of Computers*, 26(1):8–16, January/February 2009.

[12] IEEE Computer Society. *IEEE Std 1149.1*$^{\text{TM}}$*-2001, IEEE Standard Test Access Port and Boundary-Scan Architecture*. IEEE, New York, NY, USA, June 2001.

[13] Kenneth P. Parker. *The Boundary-Scan Handbook*. Springer-Verlag, third edition, June 2003.

[14] IEEE Computer Society. *IEEE Std 1500*$^{\text{TM}}$*-2005, IEEE Standard Testability Method for Embedded Core-based Integrated Circuits*. IEEE, New York, NY, USA, August 2005.

[15] Francisco da Silva, Teresa McLaurin, and Tom Waayers. *The Core Test Wrapper Handbook – Rationale and Application of IEEE Std. 1500*$^{\text{TM}}$, volume 35 of *Frontiers in Electronics Testing*. Springer-Verlag, Boston, MA, USA, 2006.

[16] Dean L. Lewis and Hsien-Hsin S. Lee. A Scan-Island Based Design Enabling Prebond Testability in Die-Stacked Microprocessors. In *Proceedings IEEE International Test Conference (ITC)*, October 2007. Paper 21.2.

[17] Xiaoxia Wu, Paul Falkenstern, and Yuan Xie. Scan Chain Design for Three-dimensional Integrated Circuits (3D ICs). In *Proceedings International Conference on Computer Design (ICCD)*, pages 208–214, October 2007.

[18] Xiaoxia Wu et al. Test-Access Mechanism Optimization for Core-Based Three-Dimensional SOCs. In *Proceedings International Conference on Computer Design (ICCD)*, pages 212–218, October 2008.

[19] Li Jiang, Lin Huang, and Qiang Xu. Test Architecture Design and Optimization for Three-Dimensional SoCs. In *Proceedings Design, Automation, and Test in Europe (DATE)*, pages 220–225, April 2009.

[20] Li Jiang et al. Layout-Driven Test-Architecture Design and Optimization for 3D SoCs under Pre-Bond Test-Pin-Count Constraint. In *Proceedings International Conference on Computer-Aided Design (ICCAD)*, pages 191–196, November 2009.

[21] Erik Jan Marinissen et al. Minimizing Pattern Count for Interconnect Tests Under A Ground-Bounce Constraint. *IEEE Design & Test of Computers*, 20(2):8–18, March/April 2003.

[22] Sandeep Kumar Goel et al. Test Infrastructure Design for the Nexperia$^{\text{TM}}$ Home Platform PNX8550 System Chip. In *Proceedings Design, Automation, and Test in Europe (DATE) Designers Forum*, pages 108–113, Paris, France, February 2004.

[23] 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.

[24] 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, USA, October 1998.

[25] Erik Jan Marinissen et al. A Structured And Scalable Mechanism for Test Access to Embedded Reusable Cores. In *Proceedings IEEE International Test Conference (ITC)*, pages 284–293, Washington, DC, USA, October 1998.

[26] Tom Waayers, Richard Morren, and Roberto Grandi. Definition of a Robust Modular SOC Test Architecture; Resurrection of the Single TAM Daisy-Chain. In *Proceedings IEEE International Test Conference (ITC)*, Austin, TX, USA, November 2005.

[27] Sandeep Kumar Goel, Erik Jan Marinissen, Anuja Sehgal, and Krishnendu Chakrabarty. Testing of SOCs with Hierarchical Cores: Common Fallacies, Test-Access Optimization, and Test Scheduling. *IEEE Transactions on Computers*, 58(3):409–422, March 2009.

[28] IEEE Computer Society. *IEEE Std 1450.6*$^{\text{TM}}$*-2005, IEEE Standard Test Interface Language (STIL) for Digital Test Vector Data – Core Test Language (CTL)*. IEEE, New York, NY, USA, 2005.

[29] Rohit Kapur. *CTL for Test Information of Digital ICs*. Kluwer Academic Publishers, Boston, MA, USA, 2003.

On the Cost-Effectiveness of Matching Repositories of Pre-Tested Wafers for Wafer-to-Wafer 3D Chip Stacking

B

Our paper regarding W2W-Matching: '*On the Cost-Effectiveness of Matching Repositories of Pre-Tested Wafers for Wafer-to-Wafer 3D Chip Stacking*'

Has been accepted in the 2010 European Test Symposium procedings, and has been presented during the event on May 24-28 2010.

On the Cost-Effectiveness of Matching Repositories of Pre-Tested Wafers for Wafer-to-Wafer 3D Chip Stacking

Jouke Verbree[2] Erik Jan Marinissen[1] Philippe Roussel[1] Dimitrios Velenis[1]

[1] IMEC vzw
3D Integration Program
Kapeldreef 75
3001 Leuven, Belgium
erik.jan.marinissen@imec.be

[2] Delft University of Technology
Department of Computer Engineering
Mekelweg 4
2628CD Delft, The Netherlands
j.verbree@student.tudelft.nl

Abstract

Three-dimensional stacked ICs (3D-SICs) based on Through-Silicon Vias (TSV) promise high-performance low-power functionality in a smaller form factor at lower cost. Stacking entire wafers has attractive benefits, but unfortunately suffers from low compound stack yield, as one cannot prevent to stack a bad die to a good die or vice versa. Matching individual wafers from repositories of pre-tested wafers to each other is a simple yet effective method to significantly increase the compound stack yield. In this paper, we present a mathematical model, which shows that the yield increase depends on (1) the number of stack tiers, (2) the number of dies per wafer, (3) the die yield, and (4) the repository size. Simulation results demonstrate that, for realistic cases, relative yield increases of 0.5% to 10% can be achieved. We also show that the required investment, in terms of a limited increase in either test or package costs, is typically well justified.

1 Introduction

The semiconductor industry is preparing itself for 3D-SICs based on TSVs [1–3]. TSVs are conducting nails which extend out of the back-side of a thinned-down die and enable the vertical interconnection to another die [4, 5]. TSVs are high-density, low-capacity interconnects compared to traditional wire-bonds, and hence allow for many more interconnections between stacked dies, while operating at higher speeds and consuming less power [6]. TSV-based 3D technologies enable the creation of a new generation of 'super chips' by opening up new architectural opportunities [7, 8]. Combined with their smaller form factor and lower overall manufacturing cost [9, 10], 3D-SICs have many compelling benefits, and hence their technology is quickly gaining ground.

In the production of 3D-SICs, we distinguish three different stacking approaches: Wafer-to-Wafer (W2W), Die-to-Wafer (D2W), and Die-to-Die (D2D). W2W requires identical wafer and die sizes in each stack tier. Compared to the other two stacking approaches, W2W offers the highest manufacturing throughput and allows for the smallest die sizes, thinnest wafers, and highest TSV densities [11, 12].

One of the most attractive benefits of 3D-SICs is the cost reduction obtained from the fact that the total product is manufactured out of individual dies that are significantly smaller than their single-die (two-dimensional) equivalent [10, 11, 13]. This cost reduction has two components. The first component is that smaller dies lead to smaller losses of area at the edge of the finite-area wafer [14]. The second component is that, for a given defect density, the yield of smaller dies is significantly higher. In order to benefit from the latter, it is necessary to test the individual dies before stacking and stack only those dies that passed the pre-bond test [15]. D2W and D2D stacking approaches allow for this selective stacking, but W2W stacking does not, as it stacks entire wafers. Hence, in W2W stacking, we cannot avoid that a bad die might be stacked to a good die or vice versa. Typically, this does not affect final product quality, as these bad stacks are filtered out by the final (packaged-product) test, but the bad stacks negatively affect the compound stack yield and hence the product cost.

However, also in W2W stacking, pre-bond test results can be exploited. Instead of blindly stacking wafers, we can perform *matching* on repositories of pre-tested wafers, in order to find out which wafer combinations would result in the highest stack yield [11, 12]. This requires a software tool to analyze and match the wafer maps of repositories of to-be-stacked wafers. A wafer map contains the per-die pass/fail results of pre-bond tests [16, 17]. Subsequently, based on the outcome of the matching algorithm, the wafers need to be sorted before being stacked, e.g., with one of many wafer sorter machines that are available on the market [18–23]. This flow is depicted in Figure 1.

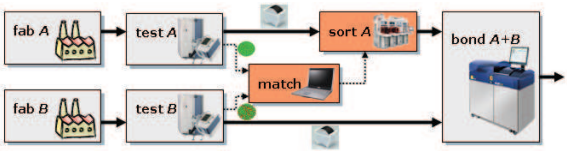


Figure 1: W2W stacking flow including wafer matching, with matching and sorting as new flow steps.

In this paper, we evaluate the cost-benefit trade-off of such a wafer matching approach. We analyze the increase of the expected stack yield, as a function of (1) the number of stack tiers, (2) the number of dies per wafer, (3) the yield per wafer, and (4) the wafer repository size. This analysis is performed both through mathematical modeling, as well as through simulation. Furthermore, we assess whether the yield benefit covers the investment of performing the pre-bond tests required for wafer matching.

The remainder of this paper is organized as follows. Section 2 describes related prior work in this field. Our mathematical model to approximate the expected yield obtained through wafer matching is described in Sec-

tion 3. Section 4 presents yield results for 10,000 simulations of repositories of pseudo-randomly generated wafer maps with varying die yield, die size, and number of stack tiers. Section 5 presents a cost analysis of two alternative production flows based on wafer matching, and shows that the costs of the additional pre-bond tests can be off-set by savings in other tests. Section 6 concludes this paper.

2 Related Prior Work

Increasing the compound yield of W2W stacking by matching of pre-tested wafers has been suggested in [11]. For a single simulation experiment for a two-tier stack of 300 mm wafers with 1320 dies of 50 mm^2 and a repository size of 25 pre-tested wafers, the paper reports stack yield increases between 0.2% and 1.2%. The authors of the paper wonder whether these small yield increases are enough to justify the additional costs of pre-bond testing. The paper lacks details on its matching approach, while the reported yield increases suffer from the incidental variation of its single-simulation experiment.

A more elaborate study on wafer matching in the context of W2W stacking was presented in [12]. The paper formulates the wafer matching problem and shows it to be $\mathcal{NP}$-hard [24]. To solve the problem, the paper subsequently presents two heuristic algorithms and an optimal algorithm based on Integer-Linear Programming (ILP). Experimental results are presented that compare the three algorithms on the basis of pseudo-randomly generated wafer maps and consider the impact of defect density, die area, number of tiers in the stack, and wafer lot size. The paper reports results based on only five simulation runs with pseudo-random wafer maps, while, in our experience, this number is too small to warrant statistically stable results.

Our contributions in this paper over the prior work are the following.

- We formulate a closed-form mathematical equation that approximates the expected stack yield due to wafer matching. The derivation of this equation provides insight in how the expected yield is influenced by parameters as the number of stack tiers, the number of dies per wafer, the yield per wafer, and the wafer repository size.
- We refine the simulation experiments in [11, 12] by presenting results from no less than 10,000 pseudo-randomly generated wafer maps.
- We analyze the cost impact of pre-bond testing and wafer matching on the overall product creation flow. We demonstrate that the additional costs of pre-bond testing are not as high as initially expected, and are in many cases more than compensated by the obtained yield benefit.

3 Yield Modeling

In this section, we derive a closed-form mathematical equation that approximates the expected stack yield that can be obtained through wafer matching as a function of the number of stack tiers, the number of dies per wafer, the yield per wafer, and the wafer repository size.

3.1 Stacking Two Individual Wafers

First, we model the expected yield of stacking two individual wafers, each containing d dies. The *top* and *bottom* wafer have respectively f_t and f_b faulty dies, as determined by their pre-bond die tests. We assume $0 \leq f_t \leq f_b \leq d$. This assumption is without loss of generality; in case $f_b < f_t$, we simply relabel top and bottom.

Let us consider the case with $d = 6$, $f_b = 3$, and $f_t = 2$ as a simplified but illustrative example running throughout this section. Two example wafers that fulfill these settings are depicted in Figure 2.

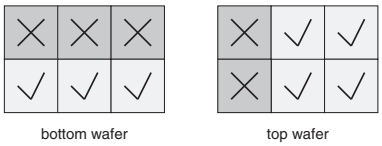


Figure 2: Simplified running example: $d = 6$, $f_b = 3$, and $f_t = 2$.

Function $y(i)$ denotes the compound yield of the stacked wafers[1] for exactly i matching faulty dies (with $0 \leq i \leq f_t$):

$$y(0) = \frac{1}{6} \quad , \qquad y(1) = \frac{2}{6} \quad , \qquad y(2) = \frac{3}{6}$$

If we generalize this, we get

$$y(i) = \frac{\max(d - f_b - f_t + i, 0)}{d} \tag{3.1}$$

Function $p(i)$ denotes the probability of occurrence of exactly i matching faulty dies. $p(i)$ can be expressed as (1) the number of possibilities to match i faulty top dies to f_b faulty bottom dies, multiplied by (2) the number of possibilities to match the remaining $f_t - i$ faulty top dies to $d - f_b$ good bottom dies, divided by (3) the number of possibilities to match all f_t faulty top dies to all d bottom dies:

$$p(i) = \binom{f_b}{i} \cdot \binom{d - f_b}{f_t - i} / \binom{d}{f_t} \tag{3.2}$$

For our running example, the complete function $p(i)$ is defined as follows:

$$p(0) = \binom{3}{0} \cdot \binom{6-3}{2-0} / \binom{6}{2} = \frac{6}{30}$$
$$p(1) = \binom{3}{1} \cdot \binom{6-3}{2-1} / \binom{6}{2} = \frac{18}{30}$$
$$p(2) = \binom{3}{2} \cdot \binom{6-3}{2-2} / \binom{6}{2} = \frac{6}{30}$$

Note that it must always hold that $\sum_{i=0}^{f_t} p(i) = 1$.

The expected yield Y can now be expressed as

$$Y = \sum_{i=0}^{f_t} (y(i) \cdot p(i)) \tag{3.3}$$

[1] In Sections 3 and 4, the stack yield explicitly disregards yield loss due to intra-die defects due to the stacking process itself and stack interconnect faults. Only in Section 5, this yield loss is again taken into account.

For our running example, this equates to

$$Y = \sum_{i=0}^{2} (y(i) \cdot p(i)) = \frac{1}{6} \cdot \frac{6}{30} + \frac{2}{6} \cdot \frac{18}{30} + \frac{3}{6} \cdot \frac{6}{30} \approx 33.33\%.$$

3.2 Matching to a Repository of Wafers

In this section, we consider the case in which a top wafer selected out of a repository of k top wafers is stacked onto a given bottom wafer with f_b faulty dies. These top wafers are all assumed to have an equal number f_t of faulty dies per wafer, but the wafer maps of the individual top wafers can be different. This setting allows to improve the yield, by selecting *that* top wafer out of the repository for which its wafer map best matches the wafer map of the bottom wafer.

We generalize the probability function $p(i)$ from Section 3.1 into a function $p(i,k)$ which denotes the probability of occurrence of exactly i matching faulty dies, given a repository of k top wafers. Note that $p(i) \equiv p(i,1)$. Let $q(i) = \sum_{j=0}^{i} (p(j))$ denote the probability of at most i matching faulty dies. Then, the generalized probability function can be expressed as follows.

$$p(i,k) = q(i)^k - q(i-1)^k \qquad \text{for } i > 0 \tag{3.4}$$

$$p(i,k) = q(0)^k \qquad \text{for } i = 0 \tag{3.5}$$

For our running example, for a repository size $k = 5$:

$$p(0,5) = q(0)^5 = 0.2^5 = 0.00032$$
$$p(1,5) = q(1)^5 - q(0)^5 = 0.8^5 - 0.2^5 = 0.32736$$
$$p(2,5) = q(2)^5 - q(1)^5 = 1^5 - 0.8^5 = 0.67232$$

Table 1 shows how an increasing repository size k increases the probability of matching a larger number of faulty dies in both wafers.

$p(i,k)$	$k=1$	$k=3$	$k=5$	$k=8$
$i=0$	0.2	0.008	0.00032	0.00000256
$i=1$	0.6	0.504	0.32736	0.16776960
$i=2$	0.2	0.488	0.67232	0.83222784

Table 1: Larger repository size k improves the probability of more faulty die matches i.

We define the expected yield $Y(k)$ for a repository of k top wafers as follows:

$$Y(k) = \sum_{i=0}^{f_t} (y(i) \cdot p(i,k)) \tag{3.6}$$

For the running example, the expected yield increases from 33.33% for a single-wafer repository to the following value for $k = 5$:

$$Y(5) = \sum_{i=0}^{2} (y(i) \cdot p(i,5)) = \frac{1}{6} \cdot p(0,5) + \frac{2}{6} \cdot p(1,5) + \frac{3}{6} \cdot p(2,5) \approx 44.73\%$$

3.3 Wafer Stacks With More Than Two Tiers

We now consider stacks of n tiers. We assume that the stacking is done in an iterative fashion. The first two tiers are stacked first. Each stack result serves again as the bottom in the stacking process with the next tier. Every wafer that is stacked onto a bottom wafer or bottom stack is selected from a repository of k wafers which have been pre-tested and for which the wafer map with good and faulty dies is known. Each wafer contains d dies. A wafer at stack level j ($1 \le j \le n$) has f_j faulty dies.

We approximate the expected yield $Y(n,k)$ for a stack of n tiers ($n \ge 2$) and a repository size of k wafers in a recursive way as follows:

- Base : $Y(2,k) = \sum_{i=0}^{f_t} (y(i) \cdot p(i,k))$ (3.7)

  where $f_t = f_2$ and $f_b = f_1$ (3.8)

- Step : $Y(n,k) = \sum_{i=0}^{f_t} (y(i) \cdot p(i,k))$ (3.9)

  where $f_t = f_n$ and $f_b = \lfloor (1 - Y(n-1,k)) \cdot d + 0.5 \rfloor$ (3.10)

3.4 Calculated Results

For the experimental results throughout this paper, we consider a *reference process* with 300 mm wafers with 3 mm edge clearance, a defect density $d_0 = 0.5$ defects/cm^2 and a defect clustering parameter $\alpha = 0.5$ [25]. Our *reference design* consists of a two-tier stack of equal-sized square dies with area $A = 50$ mm^2. These inputs give $d = 1278$ dies per wafer [14], and a die yield $y = (1 + A \cdot d_0/\alpha)^{-\alpha} = 81.65\%$ [26], which roughly corresponds to $f_j = 235$ for all tiers j.

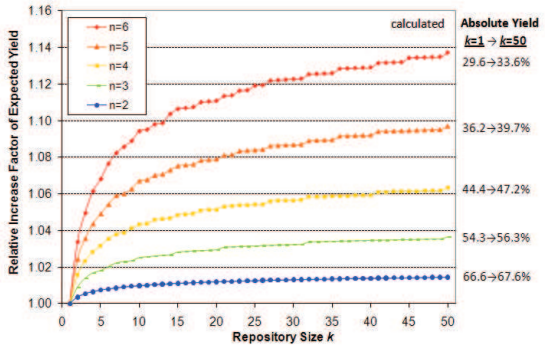


Figure 3: Relative increase factor of calculated expected stack yield for varying wafer repository size k for various numbers of stack tiers.

In Figure 3, the curves show the relative increase factor of the calculated expected stack yield, based on the model presented earlier in this section, for our reference process and reference design, for varying repository sizes $k \in \{1, \ldots, 50\}$ and various number of stack tiers $n \in \{2, \ldots, 6\}$. At the right-hand side of the figure, the absolute stack yields for $k = 1$ (minimum) and $k = 50$ (maximum) are given. The figure shows that significant yield increases of several percent points can be obtained for sufficiently large repository sizes. The more tiers in a stack, the lower the compound

stack yield is, but the larger the relative yield benefit due to wafer matching for a given repository size. The figure also shows that the yield increment due to wafer matching diminishes for larger repository sizes. Hence, the wafer repositories do not need to be large to harvest the major share of yield improvement offered by wafer matching.

The theoretical model presented in this section has a number of shortcomings. One limitation is that the model assumes a fixed number of faulty dies f_j per stack tier; in reality, there will be variations in that number. In the case of stacks of more than two tiers, the compound number of faulty dies in the bottom stack had to be calculated from the intermediate expected yield, which involves rounding to the nearest integer (see Eq. (3.10)); this assumption is responsible for the clearly visible step behavior in Figure 3, which becomes more pronounced for larger values of n. Finally, the model assumes that a single bottom wafer is stacked to the best of k top wafers; in reality, entire cassettes of wafers need to be used and matched to each other. These shortcomings are addressed in our simulation model presented in the next section.

4 Simulation Results

To assess the yield benefit of wafer matching through simulation, we have built two software components: (1) a wafer map generator, and (2) a wafer matching algorithm.

In our system, wafer maps are represented as binary arrays of length the number of dies d per wafer; a value of zero (resp. one) represents that the corresponding die on the wafer passed (resp. failed) its pre-bond test. In an actual industrial application, the wafer maps would be based on real-life test results [16, 17]. The wafer map generator mimics real life for simulation purposes, by filling the wafer maps using the internal C++ pseudo-random generator function and the fraction $(d - f)/d$ to determine whether the die was assumed to pass or fail its pre-bond test. This set-up guarantees that over a long run of generated wafer maps, the average die yield is indeed $(d - f)/d$, but allows for random variations per wafer, just as is the case for real-life wafer maps. To eliminate the bias from individual wafer maps, experiments are executed 10,000 times and averaged.

The wafer matching algorithm we have implemented matches the entire content of n wafer repositories of size k to each other; in other words, no wafers are scrapped. The algorithm iteratively matches wafers, tier by tier. The algorithm first fully matches the Tier 1 and Tier 2 wafer repositories; subsequently, the Tier 3 repository is matched to the repository of Tiers 1+2 stacks; and so on, until Tier n is also matched and stacked. In matching a repository of k new top wafers to a repository of k bottom wafers or wafer stacks, we use the following algorithm. All k^2 combinations are evaluated and the highest-yielding match is selected. Subsequently, we consider the remaining repositories of size $k - 1$, repeat the procedure, and so on. Our matching algorithm is an iterative version of the 'greedy' heuristic for only two tiers as described in [12]. Note that this software component can also be used to match real-life wafer maps, instead of the pseudo-randomly generated ones. Also, our matching algorithm can be replaced by another matching algorithm, for example the heuristic or exhaustive algorithms described in [12].

The relative increase factor of the expected yield depends on four input parameters: (1) the number n of stack tiers, (2) the number d of dies per wafer, (3) the number f_j of faulty dies per wafer (for $1 \leq j \leq n$), and (4) the wafer repository size k. We present three simulation experiments, which are based on our reference process and reference design (see Section 3.4); in each experiment we vary k and one of the other three input parameters.

In the first simulation experiment, we vary f while maintaining the die size A constant; this leads to variations in the die yield $y = (d - f)/d$. This experiment simulates the effect of varying process maturity of the wafer fab. Our reference design in our reference process has a yield of ∼82%. We have experimented with approximate die yields of 50% ($f = 639$), 60% ($f = 511$), 70% ($f = 383$), 80% ($f = 256$), and 90% ($f = 128$). The results are depicted in Figure 4.

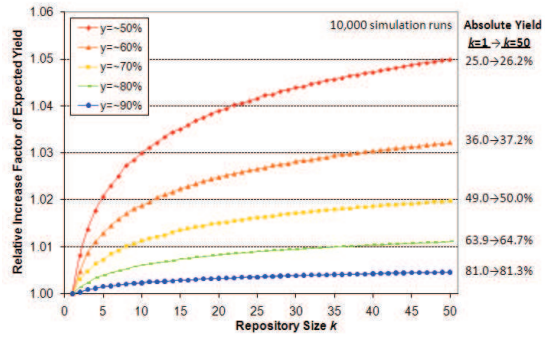


Figure 4: Relative increase factor of simulated expected stack yield for varying wafer repository size k for various die yields.

In the second simulation experiment, we vary the die size A. We keep the process maturity of the wafer fab constant, which implies that the yield per die decreases for larger dies. Table 2 shows, for the selected die sizes, the resulting d and f values which served as inputs for our simulations. The simulation results are depicted in Figure 5.

A	y [26]	d [14]	f	$(d - f)/d$
25 mm^2	89.44%	2597	274	89.45%
50 mm^2	81.65%	1278	235	81.61%
75 mm^2	75.59%	838	205	75.54%
100 mm^2	70.71%	622	182	70.74%
125 mm^2	66.67%	491	164	66.60%

Table 2: Calculation of d and f parameters, that served as inputs for our second simulation experiment.

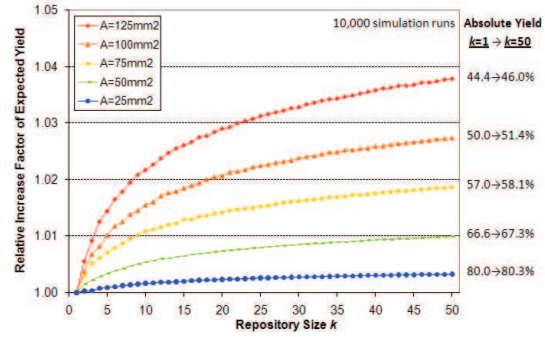


Figure 5: Relative increase factor of simulated expected stack yield for varying wafer repository size k for various die areas.

In the third simulation experiment, we use the reference process and reference design and vary the number of tiers $n \in \{2, \ldots, 6\}$. The simulation results are shown in Figure 6.

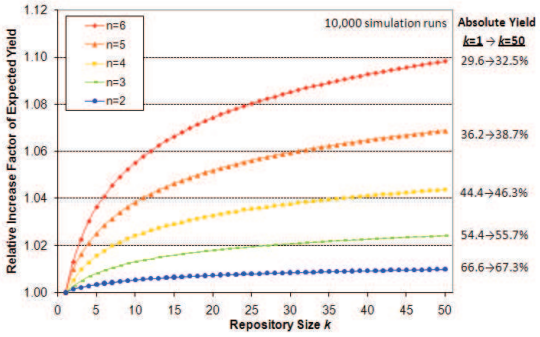


Figure 6: Relative increase factor of simulated expected stack yield for varying wafer repository size k for various numbers of stack tiers.

All three experiments lead to similar observations. The absolute compound stack yield of W2W stacking decreases for increasing number of tiers, decreasing die yield, and increasing die area; in some cases drastically. Hence, if the number of tiers is large, the die yield is low, and/or the die area is large, it is rather unattractive to do W2W stacking. However, the lower the compound stack yield, the higher the relative yield increase that can be obtained through wafer matching. In our experiments, the observed relative yield increase varies between 0.5% and 10%.

5 Wafer Matching and Test Costs

The previous sections have demonstrated the benefit of the wafer matching approach in W2W stacking in terms of yield increase. However, wafer matching is only possible if pre-bond test results are available for all dies. In case the wafer fabrication and stacking is performed by different companies, and the wafers are delivered to the stacking company under a *Known-Good Die* (KGD) agreement, die testing will be necessary to guarantee the outgoing product quality of the wafer fab. In other cases, the question is whether the additional costs of performing pre-bond die tests are compensated by the obtained yield increases [11, 12].

We consider the three production flows [15] in Figure 7. All three flows have a final test which re-tests dies and interconnects, and hence result in products with the same final test quality. The differences are in the earlier parts of the flows. Flow (a) is the base flow for W2W stacking: dies and interconnect are tested for the first time only *after* stacking. Flows (b) and (c) both make use of wafer matching, enjoy the corresponding increased yield benefit, but require pre-bond die testing. In Flow (b), the stack is fully tested after stacking but before assembly and packaging, i.e., the newly-formed interconnects between the stacked dies are tested, but also the dies themselves are re-tested. The die re-tests might show limited fall-out, e.g., due to intra-die defects induced by wafer thinning [15]. Flow (c) is an optimized variant of (b), in which during the stack test, only the newly-formed die interconnects are tested. If present, intra-die defects will slip through this stack test, but are assumed to be caught by the final test, hence maintaining a constant quality of the final product but increasing the packaging cost.

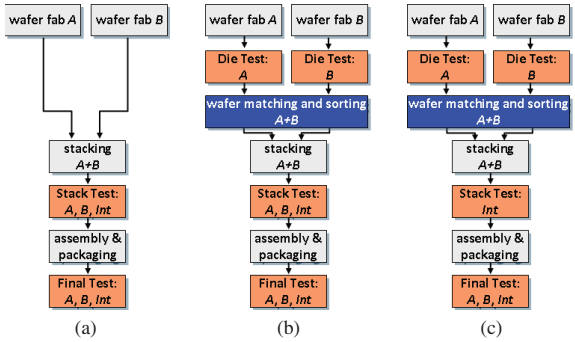


Figure 7: Production flows: (a) base flow (without wafer matching), (b) wafer matching flow, (c) optimized wafer matching flow.

Function $t(x)$ denotes the test costs for Flow (x), with $x \in \{a, b, c\}$. $t(x)$ can be expressed as a sum of products. The products express how many items are tested and what the test costs per item are. The sum iterates over the various test stages that are executed: pre-bond die test, post-bond stack test, and final test. In addition to the variables from Section 3, we introduce four new variables: t_{die} and t_{int} denote respectively the test costs (in arbitrary units (a.u.), we use this only for relative comparison) for a die test and an interconnect test; and ys_{die} and ys_{int} denote respectively the die and interconnect pass yield during the stack test. We can express $t(x)$ now as follows:

$$
\begin{aligned}
& & \text{number of items tested} & & \text{test cost per item} & \\
t(a) = \;& & d & & \cdot (n \cdot t_{\text{die}} + (n-1) \cdot t_{\text{int}}) & \\
& + & d \cdot Y(n,1) \cdot ys_{\text{die}}^{n} \cdot ys_{\text{int}}^{n-1} & & \cdot (n \cdot t_{\text{die}} + (n-1) \cdot t_{\text{int}}) & (5.1) \\
t(b) = \;& & d & & \cdot n \cdot t_{\text{die}} & \\
& + & d \cdot Y(n,k) & & \cdot (n \cdot t_{\text{die}} + (n-1) \cdot t_{\text{int}}) & \\
& + & d \cdot Y(n,k) \cdot ys_{\text{die}}^{n} \cdot ys_{\text{int}}^{n-1} & & \cdot (n \cdot t_{\text{die}} + (n-1) \cdot t_{\text{int}}) & (5.2) \\
t(c) = \;& & d & & \cdot n \cdot t_{\text{die}} & \\
& + & d \cdot Y(n,k) & & \cdot (n-1) \cdot t_{\text{int}} & \\
& + & d \cdot Y(n,k) \cdot ys_{\text{int}}^{n-1} & & \cdot (n \cdot t_{\text{die}} + (n-1) \cdot t_{\text{int}}) & (5.3)
\end{aligned}
$$

For a fair comparison of the test cost per flow, we attribute the total test costs only to the functional (passing) stacks. For Flow (a), the number of functional stacks is $d \cdot Y(n,1) \cdot ys_{\text{die}}^{n} \cdot ys_{\text{int}}^{n-1}$; for Flows (b) and (c), the number of functional stacks is $d \cdot Y(n,k) \cdot ys_{\text{die}}^{n} \cdot ys_{\text{int}}^{n-1}$.

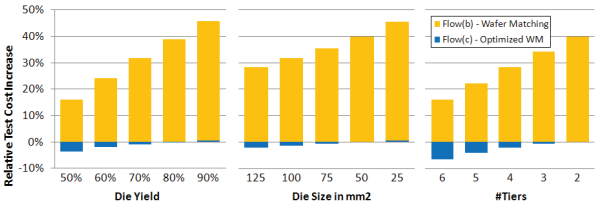


Figure 8: Relative test cost per functional stack of wafer-matching flows in comparison to the base W2W flow without wafer matching.

In Figure 8 we compare the test costs increase (or decrease) per functional stack for the two wafer-matching flows (Flows (b) and (c)), relative to the base flow (Flow (a)). For these cost calculations, we made the following

assumptions: $k = 50$, $t_{\rm die} = 5,000$ a.u., $t_{\rm int} = 50$ a.u.[2], $ys_{\rm die} = 99\%$, $ys_{\rm int} = 97\%$. The set-up of other parameters for the experiments in Figure 8 are equal to the three experiments of Section 4, in which we vary respectively yield, die size, and number of stack tiers.

At first glance, one would think that adding a pre-bond die test as a third test to a flow that already contains a post-bond stack test and a final test adds roughly 50% test costs. However, if the wafer maps resulting from the die test are kept, the stack test can simply skip stacks which contain one or more known bad dies, and hence reduce its test costs. This is confirmed by Figure 8, which shows that the test cost increase of Flow (b) varies between 15% and 50%. The lower the compound stack yield (due to low die yield, large die size, and/or a large number of tiers), the lower the additional test costs. If the compound stack yield is low, it is beneficial to find that out as early as possible through a pre-bond die test, as further (test) costs downstream can be prevented. Figure 8 also shows that the optimized wafer-matching flow (Flow (c)) more or less keeps the test costs per die constant, and in some cases even achieves a cost reduction up to 6.5%. The price to be paid for that is that due to the lower-quality stack test, some faulty stacks get packaged and are only detected as faulty during the final test. This applies to a fraction $(1 - ys_{\rm die}^n)$ of the functional stacks; in our example, for $n = 2$, this amounts to roughly 2% of the functional stacks.

6 Conclusion

The W2W stacking approach for 3D-SICs offers the highest manufacturing throughput and allows for the smallest die sizes, thinnest wafers, and highest TSV densities. The drawback of W2W stacking is that one cannot avoid that a bad die is stacked onto a good die or vice versa, leading to low compound yields. This drawback is exacerbated by a large number of stack tiers, a small number of (large) dies per wafer, and/or low die yield.

Nevertheless, if one chooses for a W2W approach, wafer matching on the basis of pre-tested wafer repositories can increase the compound yield significantly. Our mathematical yield model shows that the yield increase depends on (1) the number of stack tiers, (2) the number of dies per wafer, (3) the die yield, and (4) the wafer repository size. For reasonable repository sizes of 25 to 50 wafers (one or two cassettes), our simulations demonstrate that relative yield improvements of 0.5% to 10% can be achieved; the benefits are larger when the absolute compound yield is lower. The additional investment required to achieve this yield increase is either an relative test cost increase of 15-50% (for the wafer matching Flow (b)), or an additional package cost of 2-5% (for the optimized wafer matching Flow (c)). This implies that wafer matching is an economically viable approach in the context of W2W stacking.

Acknowledgments

We gratefully acknowledge the constructive comments from Said Hamdioui (Delft University of Technology) and Henk Hollmann (Philips Research) on an earlier draft of this paper.

[2]For our comparative study, the actual values of $t_{\rm die}$ and $t_{\rm int}$ are far less important than their ratio.

References

[1] Robert S. Patti. Three-Dimensional Integrated Circuits and the Future of System-on-Chip Designs. *Proceedings of the IEEE*, 94(6):1214–1224, June 2006.

[2] Eric Beyne and Bart Swinnen. 3D System Integration Technologies. In *Proceedings of IEEE International Conference on Integrated Circuit Design and Technology (ICICDT)*, pages 1–3, June 2007.

[3] Philip Garrou, Christopher Bower, and Peter Ramm, editors. *Handbook of 3D Integration – Technology and Applications of 3D Integrated Circuits*. Wiley-VCH, Weinheim, Germany, August 2008.

[4] Bart Swinnen et al. 3D Integration by Cu-Cu Thermo-Compression Bonding of Extremely Thinned Bulk-Si Die Containing 10μm Pitch Through-Si Vias. In *Proceedings IEEE International Electron Devices Meeting (IEDM)*, pages 1–4, May 2006.

[5] Jan Van Olmen et al. 3D Stacked IC Demonstration using a Through Silicon Via First Approach. In *Proceedings IEEE International Electron Devices Meeting (IEDM)*, pages 1–4, December 2008.

[6] Kaustav Banerjee et al. 3-D ICs: A Novel Chip Design for Improving Deep-Submicrometer Interconnect Performance and Systems-on-Chip Integration. *Proceedings of the IEEE*, 89(5):602–633, May 2001.

[7] Gabriel H. Loh, Yuan Xie, and Bryan Black. Processor Design in 3D Die-Stacking Technologies. *IEEE Micro*, 27(3):31–48, May/June 2007.

[8] Roshan Weerasekera et al. Extending Systems-on-Chip to the Third Dimension: Performance, Cost and Technological Tradeoffs. In *Proceedings International Conference on Computer-Aided Design (ICCAD)*, pages 212–219, November 2007.

[9] Xiangyu Dong and Yuan Xie. System-Level Cost Analysis and Design Exploration for Three-Dimensional Integrated Circuits (3D ICs). In *Proceedings IEEE Asia South Pacific Design Automation Conference (ASP-DAC)*, pages 234–241, January 2009.

[10] Dimitrios Velenis et al. Impact of 3D Design Choices on Manufacturing Cost. In *Proceedings IEEE International Conference on 3D System Integration (3DIC)*, pages 1–5, September 2009.

[11] Greg Smith et al. Yield Considerations in the Choice of 3D Technology. In *International Symposium on Semiconductor Manufacturing (ISSM)*, pages 1–3, October 2007.

[12] Sherief Reda, Gregory Smith, and Larry Smith. Maximizing the Functional Yield of Wafer-to-Wafer 3-D Integration. *IEEE Transactions on VLSI Systems*, 17:1357–1362, September 2009.

[13] Larry Smith et al. 3D: It All Comes Down to Cost. In *3-D Architectures for Semiconductor Integration and Packaging*, October 2007.

[14] Dirk K. de Vries. Investigation of Gross Die Per Wafer Formulas. *IEEE Transactions on Semiconductor Manufacturing*, 18(1):136–139, February 2005.

[15] Erik Jan Marinissen and Yervant Zorian. Testing 3D Chips Containing Through-Silicon Vias. In *Proceedings IEEE International Test Conference (ITC)*, November 2009.

[16] SEMI. SEMI G81-0703, Specification for Map Data Items. http://www.semi.org.

[17] SEMI. SEMI G85-0703, Specification for Map Data Format. http://www.semi.org.

[18] C&D Semiconductor Services, Inc. http://www.cdsemi.com/.

[19] Dynamic Micro Systems Semiconductor Equipment GmbH. http://www.dms-semi.de/.

[20] Mitsubishi Materials Techno. http://www.mmtec.co.jp/english/010513.html.

[21] Renesas Eastern Japan Semiconductor, Inc. http://www.tosemi.renesas.com/en/index.html.

[22] Ricmar Group. http://www.ricmar.com/.

[23] Rorze Corporation. http://www.rorze.com/english/Tech/wafer/frame3.htm.

[24] Michael R. Garey and David S. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Francisco, 1979.

[25] Jan Schat. Fault Clustering in Deep-Submicron CMOS Processes. In *Proceedings Design, Automation, and Test in Europe (DATE)*, pages 511–514, March 2008.

[26] Michael Bushnell and Vishwani Agrawal. *Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits*. Wiley-VCH, Weinheim, Germany, August 2000.

Test-Architecture Optimization for TSV-Based 3D Stacked ICs

C

A paper regarding 3D-DfT to which we have been contributors: '*Test-Architecture Optimization for TSV-Based 3D Stacked ICs*'

Has been accepted in the 2010 European Test Symposium procedings, and has been presented during the event on May 24-28 2010.

Test-Architecture Optimization and Test Scheduling for TSV-Based 3D Stacked ICs*

Brandon Noia[1], Sandeep Kumar Goel[1], Krishnendu Chakrabarty[1], Erik Jan Marinissen[2], and Jouke Verbree[3]

[1]Dept. Electrical and Computer Engineering
Duke University
Durham, NC 27708, USA
{brn2, skgoel, krish}@ee.duke.edu

[2] IMEC
Kapeldreef 75
B-3001 Leuven, Belgium
erik.jan.marinissen@imec.be

[3]Dept. of Computer Engineering
Delft University of Technology
Mekelweg 4, 2628CD Delft, The Netherlands
j.verbree@student.tudelft.nl

Abstract

Testing of 3D stacked ICs (SICs) is becoming increasingly important in the semiconductor industry. In this paper, we address the problem of test architecture optimization for 3D stacked ICs implemented using Through-Silicon Vias (TSVs) technology. We consider 3D-SICs with die-level test architectures that are either fixed or available for SIC test-access design with various levels of flexibility and show that these problems are $\mathcal{NP}$-hard. We next present mathematical programming techniques to derive optimal solutions for these problems. Experimental results for three handcrafted 3D-SICs of various SOCs from the ITC'02 SOC test benchmarks show that compared to the baseline method of sequentially testing all dies in a stack, the proposed solutions can achieve significant reduction in test time. We also show that increasing the number of test pins provides a greater reduction in test time compared to an increase in the number of TSVs. Furthermore, it is shown that 3D stacks with large and complex dies at lower layers require less test time than stacks with complex dies at higher layers.

1 Introduction

The semiconductor industry is pushing relentlessly for high-performance and low-power chips. Recent advances in semiconductor manufacturing technology have enabled the creation of complete systems with direct stacking and bonding of die-on-die. These system chips are commonly referred to as three-dimensional (3D) stacked ICs (SICs). Although a number of 3D integration methods have been proposed in the literature, in this work we focus on through-silicon via (TSV) vertical interconnects, as it offers the promise of the highest vertical interconnect density among the proposed technologies. Using TSV technology, 3D ICs are created by placing multiple device layers together through wafer or die stacking, and these are then connected using vertical TSVs [5].

The promise of 3D IC technology lies in the numerous benefits it can potentially provide over traditional 2D ICs [1, 2]. Due to the relentless increase in chip complexity, interconnects have become longer in 2D ICs, leading to increased circuit delay and power consumption. 3D ICs will lead to a reduction in the average interconnect length and help obviate the problems caused by long global interconnects [3, 4, 5]. Since die can be stacked in a 3D environment, on-chip data bandwidth can be increased as well. Furthermore, since 3D ICs can scale "up" instead of "out", higher packing density and smaller footprint can be achieved.

The manufacture of functional 3D-SICs has been demonstrated. Memories in particular are easier to stack due to high initial yields and simplified testing and design, and as such 3D memory stacks have already been manufactured [6]. Stacks that include memory stacked on logic die [7], or multiple logic die [8], are likely to be seen in the near future. Though 3D design-automation and testing tools are not yet fully mature for commercial exploitation, they are well on their way [9]. These tools need to be able to exploit the benefits of 3D technologies, while taking into account the various design-related tradeoffs. For example, in a TSV-based 3D-SIC, the number of available TSVs for test access is limited because of available chip area. Most TSVs are likely to be dedicated for functional access, power/ground, and clock routing.

Testing core-based dies in 3D-SICs brings forward new challenges [10, 11]. In order to test the dies and associated cores, a Test Access Mechanism (TAM) must be included on the dies to transport test data to the cores, and a 3D TAM is needed to transfer test data to the dies from the stack input/output pins. TAM design in 3D-SICs involves additional challenges compared to TAM design for 2D SOCs. In a 3D-SIC, a test architecture must be able to support testing of individual dies as well as testing of partial and complete stacks [11]. Furthermore, test architecture optimization must not only minimize the test time (test length), but it also needs to minimize the number of TSVs used to route the 3D TAM; as each TSV has area costs associated with it and is a potential source of defects in an 3D-SIC.

In this paper, we address the problem of test-architecture optimization of 3D-SICs with (1) *hard dies*, in which a test architecture already exists, (2) *soft dies*, for which we also design

*The work of Brandon Noia was supported in part by a Master's Scholarship from the Semiconductor Research Corporation (SRC). A preliminary version of this paper will be presented at the IEEE European Test Symposium, 2010.

the test architecture for each die, and (3) *firm dies*, in which a test architecture already exists but serial/parallel conversion hardware may be added to the die in order to reduce test-pin use and achieve better test resource allocation for stack testing. In addition to minimizing the test time for each soft die, we minimize the test time for the complete stack in both problem instances. While it is theoretically possible to have multiple dies on a given layer in a stack, we only consider one die per layer in a stack. Also, a core is considered to be part of a single die only, i.e., we do not consider "3D cores."

The rest of the paper is organized as follows. Section 2 provides an overview of the related prior work. Section 3 uses a simple example to motivate this work and formally describes the three problems addressed in this paper. Section 4 presents integer linear programming (ILP) models to solve the test-architecture optimization problems described in Section 3. Section 5 presents experimental results for various 3D-SICs constructed using SOCs from the ITC'02 SOC test benchmarks [12]. Finally, Section 6 concludes the paper.

2 Related Prior Work

Testing of 2D SOCs and the optimization of related test-access architectures have been well studied [13, 14, 15, 16]. Optimization methods have included integer linear programming (ILP) [13], rectangle packing [13, 17], iterative refinement [15], and other heuristics [16, 18]. However, all these methods were originally developed for 2D SOCs, and the added test complexities related to 3D technology were not considered.

Recently, some early work has been reported on testing of 3D-SICs. Heuristic methods for designing core wrappers in 3D-SICs were developed in [19]. ILP models for test architecture design for each die in a stack is presented in [20]. While these ILP models take into account some of the constraints related to 3D-SIC testing such as a TSV limit, this approach does not consider the reuse of die-level TAMs. A TAM wire-length minimization technique based on simulated annealing is presented in [21]. Heuristic methods for reducing weighted test cost while taking into account the constraints on test pin widths in pre-bond and post-bond tests are described in [22]. An unrealistic assumption made in [22] is that TAMs can start and terminate in any layer.

In most prior work on 3D-SIC testing, TAM optimization is performed at die-level only, which leads to inefficient TAMs and non-optimal test schedules for partial/complete stack test. Furthermore, all previous methods assume that the designer can create TAM architectures on each die during optimization, which may not be possible in all cases. In [23] a die-level wrapper and associated 3D architecture is presented to allow for all pre-bond and post-bond tests. This approack relies on die-level wrappers and it leverages current standards, IEEE 1149.1 and IEEE 1500. In addition to functional and test modes, die-level wrappers allow bypass of test data to and from higher die in the stack and reduced test bandwidth during pre-bond tests. This is a realistic and practical look at test architectures in 3D-SICs, but it offers no insight into optimization and test scheduling. The optimization methods presented in this paper are compatible with the test architecture of [23], and they do not make any unrealistic assumptions on die wrappers or the 3D TAM. This paper considers test-architecture optimization for the entire stack and considers 3D stacks with hard, soft, and firm dies. We also explore the effect of available test pins, test TSVs used, and die ordering on TAM design and test scheduling.

3 Problem Definition

In a 3D-SIC, the lowest die is usually directly connected to chip I/O pins, therefore it can be tested using test pins. To test the non-bottom dies in the stack, test data must enter through the test pins on the lowest die. Therefore, to test other dies in the stack, the test access mechanism (TAM) must be extended to all die in the stack through the test pins at the lowest die. To transport test data up and down the stack, "TestElevators" [23] need to be included on each die except for the highest die in the stack [11]. The number of test pins and TestElevators as well as the number of TSVs used affect the total test time for the stack. Currently, stacks consist of anywhere from two to eight dies.

Consider a simple 3D-SIC with three dies with given test access

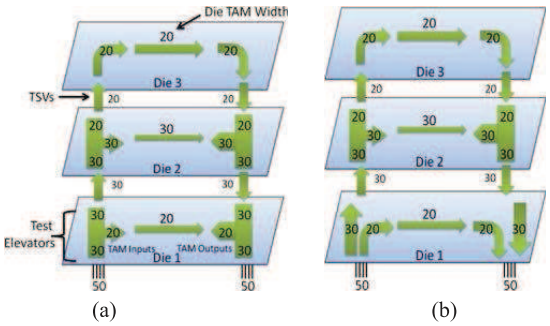


Figure 1: Example 3D-SIC with three hard dies.

architectures as shown in Figure 1. Suppose the test times for Die 1, Die 2, and Die 3 are 300, 800, and 600 clock cycles respectively. The total number of available test pins at the bottom die is 100. Die 1 requires 40 test pins (TAM width of 20), and dies 2 and 3 require 60 TestElevators and 40 TestElevators, respectively. The test time for each die is determined by its test architecture.

Figure 1(a) shows the TestElevator widths and the number of TSVs used if all dies are tested serially. In this case, a total of 100 TSVs are used and only 60 out of the available 100 test pins are necessary. The total test time for the stack is the sum

of the test times of the individual dies, i.e., 1700 cycles. Figure 1(b) shows the test architecture required if Die 1 and Die 2 are tested in parallel. In this case, the number of TSVs used is the same as in Figure 1(a). However, all 100 test pins are required to test Die 1 and Die 2 in parallel. Also, 60 TestElevators must pass between Die 1 and Die 2 in order to pass a separate 30-bit wide TAM to Die 2 for parallel testing. For this case, the total test time for the stack is 600 + max{800, 300} = 1400 cycles (we assume session-based test scheduling [24]). This example clearly shows that there is a trade-off between test time and the number of test pins and TSVs used. Therefore, a test-architecture optimization algorithm for 3D-SICs has to minimize the test time while taking into account upper limits on the number of test pins and TSVs used.

Test-architecture optimization for 3D-SICs with hard dies is illustrated in Figure 2. The optimization problem can be defined as follows.

Problem 1 [3D-SIC with Hard Dies (PSHD)]
Given a stack with a set M of dies, total number of test pins W_{max} available for test, and a maximum number of TSVs (TSV_{max}) that can be used for TAM design. For each die $m \in M$, the die's number corresponds to its teir in the stack (die 1 is the bottom die, die 2 is next, and so forth), and we are given the number of test pins on the bottom die w_m ($w_m \leq W_{max}$) required to test the die, and the associated test time t_m (since the test architecture per die is given, t_m is also given). Determine an optimal TAM design and test schedule for the stack such that the total test time T for the stack is minimized and the number of TSVs used does not exceed TSV_{max}. □

The problem statement is different for a 3D-SIC with soft dies. In the case of soft dies, the test architecture for each die is not pre-defined, but it is determined during the test-architecture design for the stack. This scenario provides greater flexibility in terms of test time optimization. Test-architecture optimization for 3D-SICs with soft dies is illustrated in Figure 3. The optimization problem can be formally

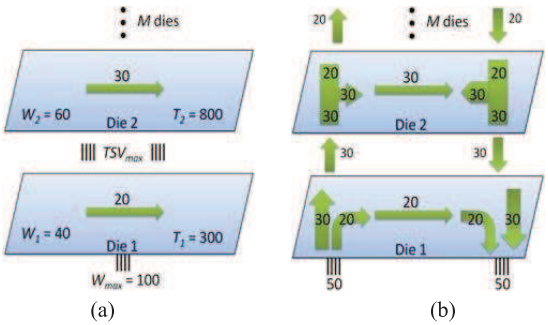


Figure 2: Illustration of PSHD: (a) a problem instance; (b) optimized architecture.

defined as follows.

Problem 2 [3D-SIC with Soft Dies (PSSD)]
Given a stack with a set M of dies, the total number of test pins W_{max} available for test at the lowest die, and a maximum number of TSVs (TSV_{max}) that can be used for TAM design. For each die $m \in M$, we are given the total number of cores c_m. Furthermore, for each core n, the number of inputs i_n, outputs o_n, total number of test patterns p_n, total number of scan chains s_n, and for each scan chain k, the length of the scan chain in flip flops $l_{n,k}$ are given. Determine an optimal TAM design and test schedule for the stack, as well as for each die, such that the total test time T for the stack is minimized and the number of TSVs used does not exceed TSV_{max}. □

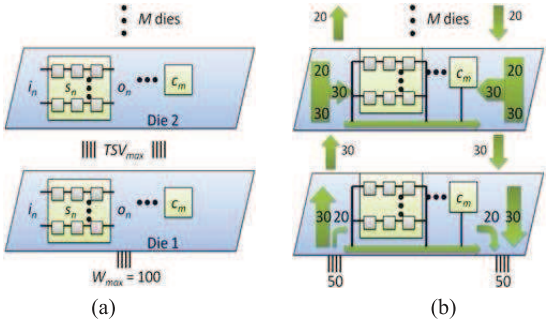


Figure 3: Illustration of PSSD: (a) a problem instance; (b) optimized architecture.

Finally, we present the problem statement for a 3D-SIC with firm die. In the case of firm dies, the test architecture for each die is pre-defined, but additional serial/parallel hardware may be added to the die to allow for fewer test elevators (or test pins in the case of the lowest die) to be used than the fixed 2D TAM width of the die. Compared to the case of SICs with hard dies, this scenario not only allows the use of fewer test pins at the expense of higher test times, but it also allows additional flexibility in test scheduling and test-time optimization. The problem of test-architecture optimization for 3D-SICs with firm dies is shown in Figure 2 can be formally defined as follows.

Problem 3 [3D-SIC with Firm Dies (PSFD)]
Given a stack with a set M of dies, the total number of test pins W_{max} available for test at the lowest die, and a maximum number of TSVs (TSV_{max}) that can be used for TAM design. For each die $m \in M$, we are given a fixed 2D TAM architecture with the total number of cores c_m, which TAM partitions they utilize, and their TAM widths. Furthermore, for each core n, the total number of test patterns p_n is given and we are given the number of test pins on the bottom die $wlim_m$ required to test the die. Determine an optimal TAM design and test schedule for the stack, as well as possible serial/parallel conversion for each die, such that the total test time T for the stack is minimized and the number of TSVs used does not exceed TSV_{max}.

□

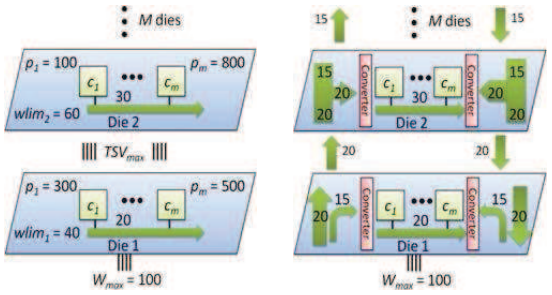


Figure 4: Illustration of PSFD: (a) a problem instance; (b) optimized architecture.

The problems in 1, 2, and 3 are $\mathcal{NP}$-hard ("proof by restriction"), as they can be reduced using standard techniques to the rectangle packing problem, which is known to be $\mathcal{NP}$-hard [25]. For example, for PSHD, if we remove the constraints related to maximum number of TSVs, each die can be represented as a rectangle with width equal to its test time and height equal to the number of required test pins. Now we need to pack all these rectangles (dies) into a bin with width equal to the total number of test pins and height equal to the total test time for the stack, which needs to be minimized. Similarly, for PSSD, a rectangle must also be selected for each die from a set of rectangles with different widths and heights, but a special case of the scenario is identical to PSHD. Despite the $\mathcal{NP}$-hard nature of these problems, they can be solved optimally since the number of layers in a 3D-SIC is expected to be limited, e.g., up to four layers have been predicted for logic stacks [26].

4 Test-Architecture Optimization

In this paper, we use integer linear programming (ILP) to solve the problems defined in the previous section. Although ILP methods do not scale well with problem instance size, the problem instance sizes for P_{SHD} and P_{SSD} are relatively small for realistic stacks, therefore ILP methods are good candidates for solving them.

4.1 ILP Formulation for PSHD

To create an ILP model for this problem, we need to define the set of variables and constraints. We first define a binary variable x_{ij}, which is equal to 1 if die i is tested in parallel with die j, and 0 otherwise. Constraints on variable x_{ij} can be defined as follows:

$$x_{ii} = 1 \qquad \forall i \tag{4.1}$$

$$x_{ij} = x_{ji} \qquad \forall i, j \tag{4.2}$$

$$1 - x_{ij} \geq x_{ik} - x_{jk} \geq x_{ij} - 1 \qquad \forall i \neq j \neq k \tag{4.3}$$

The first constraint indicates that every die is always considered to be tested with itself. The second constraint states that if die i is tested in parallel with die j, then die j is also tested in parallel with die i. The last constraint ensures that if die i is tested in parallel with die j, then it must also be tested in parallel with all other dies that are tested in parallel with die j.

Next, we define a second binary variable y_i, which is equal to 0 if die i is tested in parallel with die j on a lower layer ($l_i > l_j$), and 1 otherwise. The total test time T for the stack is the sum of test times of all dies that are tested in series plus the maximum of the test times for each of the sets of parallel tested dies. Using variables x_{ij} and y_i, the total test time T for a stack with set of dies M can be defined as follows.

$$T = \sum_{i=1}^{|M|} y_i \cdot \max_{j=i..|M|} \{x_{ij} \cdot t_j\} \tag{4.4}$$

An inductive proof of correctness of (4.4) is presented in the appendix. It should be noted that Equation (4.4) has two nonlinear elements, the max function, and the product of variable y_i and the max function. We linearize this by introducing two new variables. The variable c_i takes the value of the max function for each die i and the variable u_i represents the product $y_i \cdot c_i$. The variables u_i and c_i are defined using standard linearization techniques as shown in Figure 5. The linearized function for total test time can be written as follows.

$$T = \sum_{i=1}^{|M|} u_i \tag{4.5}$$

As the number of test pins used for parallel testing of dies should not exceed the given test pins W_{max}, a constraint on the total number of pins used to test all dies in a parallel set can be defined as follows.

$$\sum_{j=1}^{|M|} x_{ij} \cdot w_j \leq W_{max} \qquad \forall i \tag{4.6}$$

Similarly, the total number of used TSVs should not exceed the given TSV limit TSV_{max}. The number of TSVs used to connect layer i to layer $i-1$ is the maximum of the number of pins required by the layer at or above layer i that takes the most test pin connections, and the sum of parallel tested die at or above layer i in the same parallel tested set. Based on this, we can define the constraint on the total number of TSVs used in a test architecture as follows.

$$\sum_{i=2}^{|M|} \{\max_{k=i}^{|M|} \{w_k, \sum_{j=k}^{|M|} w_j \cdot x_{kj}\}\} \leq TSV_{max} \tag{4.7}$$

We can linearize the above set of constraints by representing the max function by a variable d_i. Finally, to complete the ILP model for PSHD, we must define constraints on binary variable y_i and the relationship between binary variable y_i and x_{ij}. For

this purpose, we first define a constant C that approaches but is less than 1. We then define y_i as follows:

$$y_1 = 1 \tag{4.8}$$

$$y_i \geq \frac{1}{1-i} \sum_{j=1}^{i-1} (x_{ij} - 1) - C \qquad \forall i > 1 \tag{4.9}$$

The first equation forces y_1 to 1, since the lowest layer cannot be tested in parallel with any layer lower than itself. Constraint 4.9 defines y_i for the other layers. To understand this constraint, we first make the observation that the objective function (as shown in Equation (4.4)) would be minimized if each y_i is zero. This would make the objective function value equal to 0, which is an absolute minimum test time. Thus, we only need to restrict y_i to 1 where it is absolutely necessary, and then we can rely on the objective function to assign a value 0 to all unrestricted y_i variables. This equation considers the range of values that the sum of x_{ij} can take. The fraction in the equation normalizes the sum to a value between 0 and 1 inclusive, while the summation considers all possible cases for a die being tested in parallel with a die below it. A formal proof of correctness of this constraint is given in the appendix. The complete ILP model is shown in Figure 5.

Objective:
Minimize $\sum_{i=1}^{|M|} u_i$
Subject to:
$t_{max} = \max_{i=1}^{|M|} t_i$
$c_i \geq x_{ij} \cdot t_j \qquad \forall i, j = i..|M|$
$u_i \geq 0 \qquad \forall i$
$u_i - t_{max} \cdot y_i \leq 0 \qquad \forall i$
$u_i - c_i \leq 0 \qquad \forall i$
$c_i - u_i + t_{max} \cdot y_i \leq t_{max} \qquad \forall i$
$\sum_{j=1}^{|M|} x_{ij} \cdot w_j \leq W_{max} \qquad \forall i$
$x_{ii} = 1 \qquad \forall i$
$x_{ij} = x_{ji} \qquad \forall i, j$
$1 - x_{ij} \geq x_{ik} - x_{jk} \geq x_{ij} - 1 \qquad \forall i \neq j \neq k$
$\sum_{i=2}^{|M|} d_i \leq TSV_{max} \qquad \forall i$
$d_i \geq \sum_{j=k}^{|M|} w_j \cdot x_{kj} \qquad \forall i, k = i..|M|$
$d_i \geq w_j \qquad \forall i, j = i..|M|$
$y_1 = 1$
$y_i \geq \frac{1}{1-i} \sum_{j=1}^{i-1} (x_{ij} - 1) - C \qquad \forall i > 1$

Figure 5: ILP model for 3D TAM optimization PSHD.

4.2 ILP Formulation for PSSD

The ILP formulation for 3D-SICs with soft cores is derived in a similar manner as that for 3D-SICs with hard cores. In this case, the test time t_i for die i is a function of the TAM width w_i assigned to it. Using the variables x_{ij} and y_i as defined in Section 4.1, the total test time T for the stack with the set of soft dies M can be defined as follows.

$$T = \sum_{i=1}^{|M|} y_i \cdot \max_{j=i..|M|} \{x_{ij} \cdot t_j(w_j)\} \tag{4.10}$$

It should be noted that Equation (4.10) has several non-linear elements. To linearize this equation, first we must define the test time function. For this purpose, we introduce a binary variable $g_{in} = 1$ if $w_i = n$, and 0 otherwise. We then linearize this expression using the variable v_{ij} for $x_{ij} \cdot \sum_{n=1}^{k_i} (g_{jn} \cdot t_j(n))$. Similar to Equation (4.5), the variable c_i takes the value of the max function for each die i and the

Objective:
Minimize $\sum_{i=1}^{N} u_i$
Subject to:
$t_{max} = \max_{i=1}^{|M|} t_i$
$c_i \geq v_{ij} \qquad \forall i, j = i..|M|$
$v_{ij} \geq 0 \qquad \forall i, j$
$v_{ij} - t_{max} \cdot x_{ij} \leq 0 \qquad \forall i, j = 1..|M|$
$-\sum_{n=1}^{k_i} (g_{jn} \cdot t_j(n)) + v_{ij} \leq 0 \qquad \forall i, j$
$\sum_{n=1}^{k_i} (g_{jn} \cdot t_j(n)) - v_{ij} + t_{max} \cdot x_{ij} \leq t_{max} \qquad \forall i, j$
$u_i \geq 0 \qquad \forall i$
$u_i - t_{max} \cdot y_i \leq 0 \qquad \forall i$
$u_i - c_i \leq 0 \qquad \forall i$
$c_i - u_i + t_{max} \cdot y_i \leq t_{max} \qquad \forall i$
$z_{ijk} \geq 0 \qquad \forall i, j, k$
$z_{ijk} - t_{max} \cdot x_{jk} \leq 0 \qquad \forall i, j, k$
$-wi + z_{ijk} \leq 0 \qquad \forall i, j, k$
$w_i - z_{ijk} + t_{max} \cdot x_{jk} \leq t_{max} \qquad \forall i, j, k$
$\sum_{i=2}^{|M|} d_i \leq TSV_{max}$
$d_i \geq \sum_{j=k}^{|M|} z_{jkj} \qquad \forall i, k = i..|M|$
$d_i \geq w_j \qquad \forall i, j = i..|M|$
$\sum_{j=1}^{|M|} z_{jij} \leq W_{max} \qquad \forall i$
$x_{ii} = 1 \qquad \forall i$
$x_{ij} = x_{ji} \qquad \forall i, j$
$1 - x_{ij} \geq x_{ik} - x_{jk} \geq x_{ij} - 1 \qquad \forall i \neq j \neq k$
$y_1 = 1$
$y_i \geq \frac{1}{1-i} \sum_{j=1}^{i-1} (x_{ij} - 1) - M \qquad \forall i > 1$

Figure 6: ILP model for 3D TAM optimization PSSD.

variable u_i represents the product $y_i \cdot c_i$. Since w_j is now a decision variable, we linearize $x_{ij} \cdot w_j$ using a new variable z_{ijk} defined for all i, j, k. We represent the max function by the variable d_i as before. By using the variable z_{ijk}, the TAM width that can be given to each die can be constrained by an upper limit, which is the number of available test pins. We represent this with the following set of inequalities. The complete ILP model for PSSD is shown in Figure 6.

$$\sum_{j=1}^{|M|} z_{jij} \leq W_{max} \qquad \forall i \tag{4.11}$$

4.3 ILP Formulation for PSFD

The ILP formulation for 3D-SICs with firm dies is an extension to the model for soft dies. We add a constraint to indicate that the number of test pins used for a die cannot exceed the number of test pins required by the fixed 2D TAM for that die. This

constraint is expressed as:

$$w_i \leq wlim_i \qquad \forall i \tag{4.12}$$

where $wlim_i$ is the number of test pins required by the 2D TAM on each die i prior to any serial/parallel conversion. In order to accurately determine test times for the dies using serial/parallel conversion, we modify the control-aware TAM design method of [27] to allow the architecture to be fixed in terms of assignment of modules to TAM partitions for a die. We then iteratively reduce the effective widths of the TAM partitions and reoptimize, thereby determing the optimal serial/parallel conversion to use depending on the bandwidth given to that die, as shown in Figure 7.

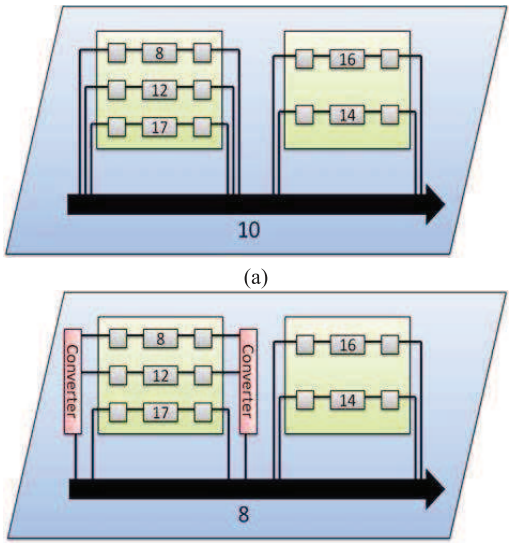


Figure 7: Illustration of TAM width reduction using die-level serial/parallel conversion.

Figure 7 (a) shows a die prior to TAM width reduction, where ten pins are required to test the die. There are two cores, one with three wrapper chains consisting of the given number of scan flops, and another with two wrapper chains. The ammount of time needed to test each core depends on the length of the longest wrapper chain and the number of test patterns required by the core. In this example, we assume that both cores require the same number of test patterns. Therefore, when we reduce the TAM width by two, it is best to combine the wrapper chains of length eight and twelve in the first core, resulting in a longest wrapper chain of length 20 bits as seen in Figure 7 (b).

5 Experimental Results

In this section, we present experimental results for the ILP models presented in the previous section. As benchmarks, we have handcrafted three 3D-SICs (as shown in Figure 8) out of several SOCs from the ITC'02 SOC Test Benchmarks as dies inside the 3D-SICs. The SOCs used are d695, f2126, p22810, p34292, and p93791. In SIC 1, the die are ordered such that the lowest die is the lease complex (d695), with die increasing in complexity as one moves higher in the stack. The order is reversed in SIC 2, while for SIC 3, the most complex die is placed in the middle of the stack, with die decreasing in complexity moving out from that die.

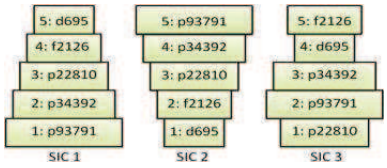


Figure 8: Three 3D-SIC benchmarks.

To determine the test architecture and test time for a given die (SOC) with a given TAM width, we have used the control-aware TAM design method in [27]. Control-aware TAM design takes into account the number of scan enable signals required for independent testing of TAMs in the architecture. For PSHD (3D-SIC with hard dies), the test times (cycles) and TAM widths for different dies are listed in Table 1. Note that test pins were assigned to dies based on their sizes in order to avoid very large test times for any individual die.

For a fixed TSV_{max} and range of W_{max}, Table 2 presents results for PSHD for the three benchmark 3D-SICs using hybrid TestRail architectures [15]. For PSHD and its comparison to PSHD, optimizations were done using hybrid TestBus [15] architecture for variety. The ILP models were solved using the XPRESS-MP tool [28].

Die	d695	f2126	p22810	p34392	p93791
Test Length	96297	669329	651281	1384949	1947063
Test Pins	15	20	25	25	30

Table 1: Test lengths and number of test pins for dies as required in PSHD.

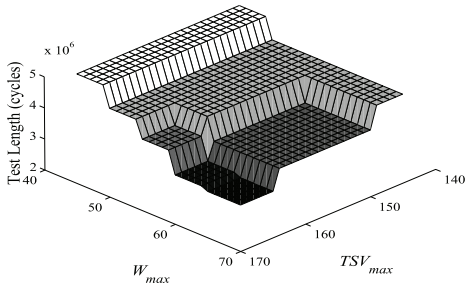


Figure 9: Variation in test time with W_{max} and TSV_{max} for SIC 2 with hard dies.

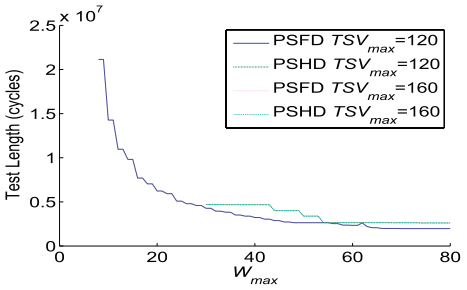


Figure 10: Variation in test time with W_{max} for SIC 1 with firm dies.

In this table, Column 1 shows the maximum number of TSVs allowed (TSV_{max}), while Column 2 represents the number of available test pins W_{max}. Columns 3, 6 and 9 represent the total test length (cycles) for the stack for 3D-SIC 1, 2 and 3 respectively. Columns 4, 7, and 10 show the resulting test schedule for the 3D-SICs, where the symbol "||" indicates parallel testing of dies, and a "," represents serial testing. Finally, Columns 5, 8, and 11 show the percent decrease in test time over the serial testing case for the three 3D-SICs. From Table 2 we can see that compared to serial testing of all dies (first row in the table), the proposed method obtains up to 57% reduction in test time. Note that although identical test times were obtained for SIC 2 and SIC 3 for $TSV_{max} = 160$, different TAM architectures and test schedules are obtained from the optimization algorithm (see Columns 4 and 10).

For a different number of TSVs (TSV_{max}), Figure 12(a) and Figure 12(b) show the variation in test time T with an increase in number of test pins W_{max} for SIC 1 and SIC 2. From the figures we can see that both TSV_{max} and W_{max} determine which dies should be tested in parallel, and thus the total test time for the stack. For a given value of TSV_{max}, increasing W_{max} does not always decreases the test time. Similarly, increasing TSV_{max} for a given W_{max} does not always decreases the test time. These Pareto-optimal points are shown in Figure 12(c) for SIC 2.

TSV_{max}	W_{pin}	SIC 1			SIC 2			SIC 3		
		Test Length (cycles)	Test Schedule	Reduction (%)	Test Length (cycles)	Test Schedule	Reduction (%)	Test Length (cycles)	Test Schedule	Reduction (%)
160	30	4748920	1,2,3,4,5	0.00	4748920	1,2,3,4,5	0.00	4748920	1,2,3,4,5	0.00
160	35	4652620	1,2,3,4‖5	2.03	4652620	1‖2,3,4,5	2.03	4652620	1‖5,2,3,4	2.03
160	40	4652620	1,2,3,4‖5	2.03	4652620	1‖3,2,4,5	2.03	4652620	1‖5,2,3,4	2.03
160	45	3983290	1‖5,2‖4,3	16.12	3983290	1‖3,2‖4,5	16.12	3983290	1‖4,2‖5,3	16.12
160	50	3428310	1‖4,2‖3,5	27.81	3428310	1,2‖5,3‖4	27.81	3428310	1,2‖4,3‖5	27.81
160	55	2712690	1‖2,3‖4,5	42.88	2712690	1,2‖3,4‖5	42.88	2712690	1,2‖3,4‖5	42.88
160	60	2616390	1‖2,3‖4‖5	44.91	2616390	1‖2‖3,4‖5	44.91	2616390	1‖4‖5,2‖3	44.91
160	65	2616390	1‖2,3‖4‖5	44.91	2616390	1‖2‖3,4‖5	44.91	2616390	1‖4‖5,2‖3	44.91
160	70	2616390	1‖2‖5,3‖4	44.91	2616390	1‖2‖3,4‖5	44.91	2616390	1‖2‖3,4‖5	44.91
160	75	2598340	1‖2‖4,3‖5	45.29	2616390	1‖2‖3,4‖5	44.91	2616390	1‖2‖3,4‖5	44.91
160	80	2598340	1‖2‖4,3‖5	45.29	2616390	1‖2‖3,4‖5	44.91	2616390	1‖2‖3,4‖5	44.91
160	85	2598340	1‖2‖4,3‖5	45.29	2616390	1‖2‖3,4‖5	44.91	2616390	1‖2‖3,4‖5	44.91
160	90	2598340	1‖2‖4,3‖5	45.29	2616390	1‖2‖3,4‖5	44.91	2616390	1‖2‖3,4‖5	44.91
160	95	2598340	1‖2‖4,3‖5	45.29	2616390	1‖2‖3,4‖5	44.91	2616390	1‖2‖3,4‖5	44.91
160	100	2043360	1‖2‖3‖4,5	56.97	2616390	1‖2‖3,4‖5	44.91	2616390	1‖2‖3,4‖5	44.91
160	105	2043360	1‖2‖3‖4,5	56.97	2616390	1‖2‖3,4‖5	44.91	2616390	1‖2‖3,4‖5	44.91

Table 2: Experimental results for PSHD.

Optimization Problem	W_{max}	Test Length (cycles)	Reduction (%)	Test Schedule	No. of Test Pins used per Die
$PSHD$	35	4678670	0	1,2,3,4‖5	30,24,24,20,14
	44	4009340	0	1‖4,2,3‖5	30,24,24,20,14
	50	3381720	0	1‖3,2‖5,4,	30,24,24,20,14
	60	2658750	0	1‖5,2‖3,4,	30,24,24,20,14
	80	2658750	0	1‖5,2‖3,4,	30,24,24,20,14
$PSFD$	35	3828490	18.17	1‖4,2‖3,5	28,24,10,7,14
	44	2875900	28.27	1‖2,3‖4,5	28,16,24,18,14
	50	2641060	21.90	1‖2,3‖4‖5	30,18,24,18,4
	60	2335780	12.15	1‖2‖3‖4,5	28,16,10,6,14
	80	1971400	25.85	1‖2‖3‖4‖5	30,24,10,8,8

Table 3: Comparisons between PSHD and PSFD.

Figure 9 shows the variation in test time for SIC 2 when both TSV_{max} and W_{max} are varied. From the figure, we can see that a small increase in the number of test pins W_{max} for a given TSV_{max} reduces the test time significantly, while to achieve the same reduction in test time with a fixed number of test pins W_{max}, a large increase in TSV_{max} is required.

Table 3 shows comparative results for both PSHD and PSFD for hyrbrid TestBus architectures. Column 1 gives the optimization problem. Column 2 shows the number of test pins available for use. Column 3 presents the test length for the 3D-SIC in cycles, with Column 4 presenting the reduction in test time versus the hard die optimization case. Column 5 gives the test schedule and Column 6 gives the number of test pins needed to test each die for the optimization.

We see from Table 3 that by adding serial/parallel conversion of TAM inputs to hard die, we can obtain as much as 28% reductions in test time. This is because the conversion allows for an increase in the test time of individual die in order to minimize the overall SIC test time during test schedule optimization. We also note that the test schedules and the number of test pins utilized for each die differ considerably between the hard die and firm die problem instances. Compared to a hard die, a firm die requires a small amount of extra hardware to convert a narrow TAM at the die input to a wider die-internal TAM, and vice versa. The area cost of this additional hardware is negligible compared to the die area and the hardware required for the core and die wrappers.

Figures 10 and 11 show comparative test times between PSHD and PSFD when W_{max} is varied, for two values of TSV_{max} and for two SICs. It is impossible to test the hard dies in these cases using fewer than 30 test pins without using serial/parallel conversion. As fewer test pins are used, the test times for individual die greatly increase, resulting in a sharp increase in overall test time below certain values of W_{max}. It is important to note that the test time for a SIC with hard dies can never be shorter than the test time for the same SIC with firm dies; at best it can be equal. This is because, in the worst case with respect to test time, the optimization for firm dies is equivalent to the optimization for hard dies, i.e., no serial/parallel conversion is carried out.

For PSSD (3D-SIC with soft dies), Pareto-optimality is almost non-existent when W_{max} is varied; see Figure 13 and Figure 14. This is due to the fact that as dies in the stack are soft, it is always possible to find one die for which adding an extra test pin reduces the overall test time. In Figure 14, some Pareto-optimal points can be identified for SIC 2. This is because the most complex die in a stack tends to be the bottleneck in reducing test time. Since these dies are stacked toward the top of the stack in SIC 2, TSV constraints are more restrictive; the

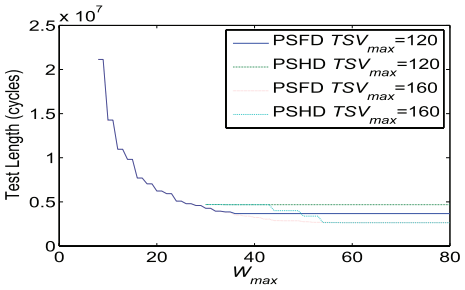


Figure 11: Variation in test time with W_{max} for SIC 2 with firm dies.

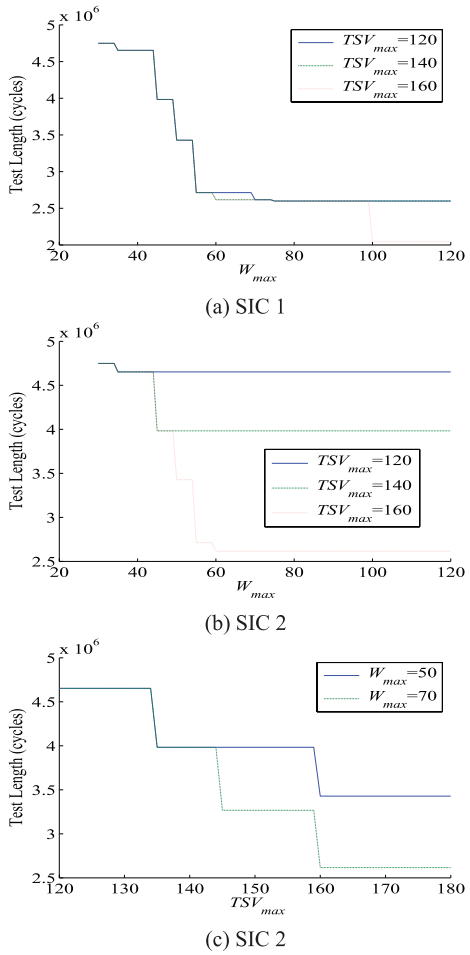


Figure 12: The test time with respect to TSV_{max} and W_{max} for SIC 1 and SIC 2 with hard dies.

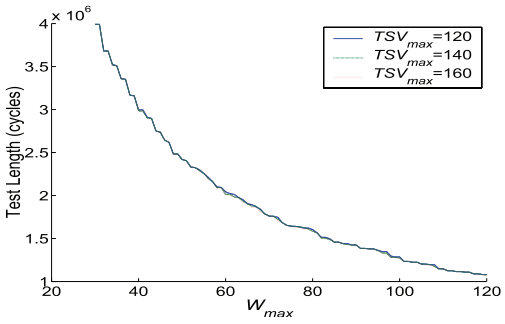


Figure 13: Variation in test time with W_{max} for SIC 1 with soft dies.

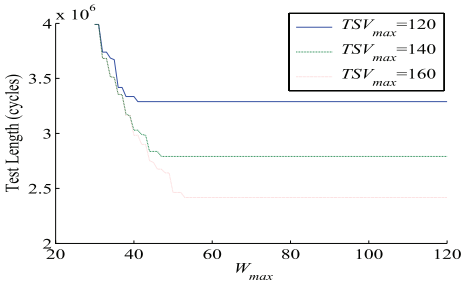


Figure 14: Variation in test time with W_{max} for SIC 2 with soft dies.

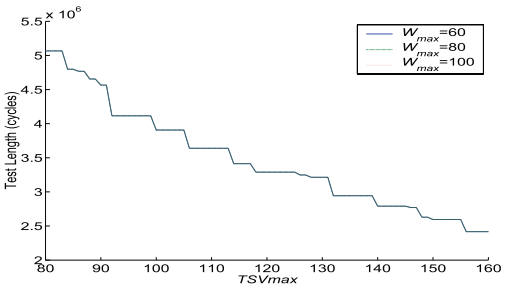


Figure 15: Variation in test time with TSV_{max} for SIC 2 with soft dies.

addition of test pins to these dies requires more TSVs and TestElevators throughout the stack. However, for PSSD, although varying W_{max} does not create Pareto-optimal points, varying TSV_{max} results in various Pareto-optimal points as shown in Figure 15.

Note that this effect is more pronounced in SIC 2 than in the other 3D-SICs. This is because the addition of test pins to the bottleneck die (at the highest layer) introduces a larger TSV overhead than in the other 3D-SICs. Furthermore, as long as W_{max} is sufficient, TSV_{max} is the limiter on test time. For PSHD, PSSD, and PSFD, the stack configuration (SIC 1) with

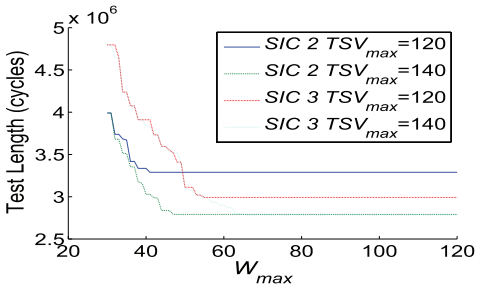


Figure 16: Comparison of test times for PSSD for SIC 2 and SIC 3

the largest die at the lowest layer and the smallest die at the highest layer is the best for reducing test time while using the minimum number of TSVs. Figure 16 shows a comparison of optimized test times for soft die between SIC 2 and SIC 3. As shown, SIC 3 leads to test times lower than or equivalent to SIC 2 at higher values for W_{max}. However, under tight test pin constraints SIC 2 results in better test times.

6 Conclusions

We have introduced the problem of test-architecture optimization for 3D stacked ICs with hard, soft, and firm dies. In the case of hard dies, the test architecture for each die is fixed and given, for firm dies serial/parallel conversion is allowed, while for soft dies the test architecture has to be determined while designing the test architecture for the entire stack. We have used ILP techniques to solve the above problems. We have considered constraints on the number of available test pins and the number of TSVs used. Results for three different stack configuration made up of five SOCs taken from the ITC'02 SOC Test Benchmarks show that Pareto-optimal points are present for stacks with hard dies, while for stacks with soft dies and firm dies, test time is, in most cases, monotonically reduced with an increase in the number of test pins. Moreover, increasing the number of test pins provides a greater reduction in test time compared to an increase in the number of TSVs. Finally, stacks with large and complex dies at the lowest layers lead to lower test times compared to stacks with complex dies at the highest layers.

References

[1] K. Banerjee et al., "3-D ICs: a Novel Chip Design for Improving Deep-submicrometer Interconnect Performance and Systems-on-chip Integration", *Proc. IEEE*, vol. 89, no. 5, pp. 602-633, 2001.

[2] R. Weerasekera et al., "Extending Systems-on-chip to the Third Dimension: Performance, Cost and Technological Tradeoffs", in *Int. Conference on Computer-Aided Design*, pp.212-219, 2007.

[3] G. Loh, Y. Xie, and B. Black, "Processor Design in 3D Die Stacking Technologies", *IEEE Micro* Vol 27, No. 3, 2007, pp.31-48

[4] W.R. Davis et al., "Demystifying 3D ICs: the Pros and Cons of Going Vertical", *IEEE Design and Test of Computers*, vol. 22, no. 6, pp. 498-510, 2005.

[5] Y. Xie, G. H. Loh, and K. Bernstein, "Design Space Exploration for 3D Architectures", J. *Emerg. Technol. Comput. Syst.*, 2(2):65-103, 2006.

[6] Uksong Kang et al. "8Gb 3D DDR3 DRAM Using Through-Silicon-Via Technology", In *Proceedings International Solid State Circuits Conference (ISSCC)*, pp. 130-132, February 2009.

[7] C.C. 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, November/December 2005.

[8] B. Black, D.W. Nelson, C. Webb, and N. Samra, "3D Processing Technology and its Impact on iA32 Microprocessors", *In Proceedings International Conference on Computer Design (ICCD)*, pp. 316-318, October 2004.

[9] Shamik Das, Anantha Chandrakasan, and Rafael Reif. "Design Tools for 3-D Integrated Circuits," In *Proceedings IEEE Asia South Pacific Design Automation Conference (ASP-DAC)*, pp. 5356, January 2003.

[10] H.-H.S. Lee and K. Chakrabarty, "Test challenges for 3D integrated circuits", *IEEE Design & Test of Computers*, vol. 26, pp. 26-35, September/October 2009

[11] E.J. Marinissen and Y. Zorian, "Testing 3D Chips Containing Through-Silicon Vias", *International Test Conference*, E 1.1, 2009.

[12] E.J. Marinissen, V. Iyengar, and K. Chakrabarty. "A Set of Benchmarks for Modular Testing of SOCs", In *International Test Conference*, pp. 519-528, Oct. 2002.

[13] V. Iyengar, K. Chakrabarty, and E.J. Marinissen, "Test Wrapper and Test Access Mechanism Co-optimization for System-on-chip", *Journal of Electronic Testing, Theory, and Applications*, vol. 18, pp. 213-230, 2002.

[14] E.J. Marinissen, S.K. Goel, and M. Lousberg, "Wrapper Design for Embedded Core Test", In *International Test Conference*, pp. 911-920, 2000.

[15] S.K. Goel and E.J. Marinissen. "SOC Test Architecture Design for Efficient Utilization of Test Bandwidth", *ACM Transactions on Design Automation of Electronic Systems*, 8(4):399429, 2003.

[16] E. Larsson, K. Arvidsson, H. Fujiwara, and Z. Peng, "Efficient Test Solutions for Core-based Designs", *IEEE Transactions on Computer Aided Design*, vol. 23, no. 5, pp. 758-775, 2004.

[17] Y. Huang et al., "Optimal Core Wrapper width Selection and SOC Test Scheduling based on 3-D Bin Packing Algorithm", In *IEEE International Conference on Computer Design*, pp. 74-82, 2009.

[18] Q. Xu and N. Nicolici, "Resource-Constrained System-on-a-Chip Test: A Survey", *IEE Proceedings: Computers and Digital Techniques*. vol. 152, pp. 67-81, Jan. 2005.

[19] B. Noia, K. Chakrabarty, and Y. Xie, "Test-Wrapper Optimization for Embedded Cores in TSV-Based Three-Dimensional SOCs", *Proc. IEEE International Conference on Computer Design*, pp. 70-77, 2009.

[20] X. Wu, Y. Chen K. Chakrabarty, and Y. Xie, "Test-access Mechanism Optimization for Core-based Three-dimensional SOCs", *IEEE International Conference on Computer Design*, pp.212-218, 2008.

[21] L. Jiang, L. Huang, and Q. Xu, "Test Architecture Design and Optimization for Three-dimensional SoCs", *Design, Automation, and Test in Europe*, pp. 220-225, 2009.

[22] L. Jiang, Q. Xu, K. Chakrabarty and T.M. Mak, "Layout-driven test-architecture design and optimization for 3D SoCs under pre-bond test-pin-count constraint", *Proc. IEEE International Conference on Computer-Aided Design*, pp.191-196, 2009.

[23] E.J. Marinissen, J. Verbree, and M. Konijnenburg, "A Structured and Scalable Test Access Architecture for TSV-Based 3D Stacked ICs", *IEEE VLSI Test Symposium*, Santa Cruz, California, April 2010

[24] M.L. Flottes, J. Pouget, and B. Rouzeyre, "Sessionless Test Scheme: Power-constrained Test Scheduling for System-on-a- Chip", *Proceedings of the 11th IFIP on VLSI-SoC*, pp. 105110, 2001.

[25] E.G. Coffman, Jr., M.R. Garey, D.S. Johnson and R.E. Tarjan. "Performance bounds for level-oriented two-dimensional packing algorithms", *SIAM J. Computing*, vol. 9, pp. 809826, 1980.

[26] X. Dong and Y. Xie. "System-level Cost Analysis and Design Exploration for 3D ICs", *Proceedings of Asia-South Pacific Design Automation Conference (ASP-DAC)*, pp.234-241, Jan. 2009.

[27] S.K. Goel, E.J. Marinissen, "Control-Aware Test Architecture Design for Modular SOC Testing", *European Test Workshop*, pp. 57-62, 2003.

[28] FICO. Xpress-MP.
http://www.fico.com/en/Products/DMTools/Pages/FICO-Xpress-Optimization-Suite.aspx

Appendix

We present proofs of correctness of Equations (4.4) and (4.9).

Inductive Proof Correctness for (4.4)

Base Case:
For the base case, we consider two layers, and there are two possible outcomes. Either both die are tested in series, or both die are tested in parallel. In the case of series testing, then $y_1 = 1, x_{11} = 1, x_{12} = 0$ and $y_2 = 1, x_{21} = 0, x_{22} = 1$. Using this information and equation 4.4, we determine the test time as $y_1 \max\{x_{11} \cdot t_1\} + y_2 \max\{x_{22} \cdot t_2\} = \max\{t_1, 0\} + \max\{t_2\} = t_1 + t_2$. For parallel testing, the variables become $y_1 = 1, x_{11} = 1, x_{12} = 1$ and $y_2 = 0, x_{21} = 1, x_{22} = 1$. The equation becomes $1 \cdot \max\{t_1, t_2\} + 0 \cdot \max\{t_2\} = \max\{t_1, t_2\}$. These can both be seen to be correct.

Induction Hypothesis:
We assume that (4.4) holds for M die.

Recursive Step:
We wish to prove that the test time for die $M + 1$ is properly considered in the overall test time. Either die $M + 1$ is tested in serial with regards to the die in the stack, or it is tested in parallel with some die on a lower layer of the stack. When die $M + 1$ is tested in series, $y_{M+1} = 1, x_{M+1,M+1} = 1$, and $x_{n,M+1}$ and $x_{M+1,n}$ are zero for all $n \neq M + 1$. The test time

On Maximizing the Compound Yield for 3D Wafer-to-Wafer based ICs

D

A paper regarding W2W-Matching to which we have been contributors: '*On Maximizing the Compound Yield for 3D Wafer-to-Wafer based ICs*'

Has been accepted in the 2010 International Test Symposium procedings, and will be presented during the event on October 31st untill November 5th 2010.

On Maximizing the Compound Yield for 3D Wafer-to-Wafer Stacked ICs

Mottaqiallah Taouil[1] Said Hamdioui[1] Jouke Verbree[1,2] Erik Jan Marinissen[2]

[1] Delft University of Technology
Faculty of EE, Mathematics and CS
Mekelweg 4, 2628 CD Delft, The Netherlands
{M.Taouil , S.Hamdioui}@tudelft.nl

[2]IMEC vzw
3D Integration Program
Kapeldreef 75, 3001 Leuven, Belgium
{jouke.verbree, erik.jan.marinissen}@imec.be

Abstract

Three-dimensional stacked IC (3D-SIC) technology is an emerging technology that provides heterogeneous integration, higher performance, and lower power consumption compared to planar ICs. Fabricating these 3D-SICs using wafer-to-wafer (W2W) stacking has on one hand several advantages, which include: high throughput, thin wafer and small die handling, and high TSV density. However, on the other hand, W2W stacking suffers from low compound yield, especially with increased number of stacked layers. This paper investigates various matching processes using different wafer matching criteria in order to maximize the compound yield. It first establishes a framework covering different matching processes and wafer matching criteria for both replenished and non-replenished wafer repositories. Thereafter, a subset of the framework is analyzed. The simulation results show that the compound yield not only depends on the number of stacked dies, die yield, and repository size, but also strongly depends on the used match process and the wafer matching criteria. Moreover, by choosing appropriate wafer matching processes and criteria, the compound yield can be improved up to 13.4% relative to random W2W stacking.

Keywords: *3D integration, wafer matching, matching criteria, compound yield, wafer-to-wafer stacking*

1 Introduction

The ability to create three-dimensional Stacked Integrated Circuits (3D-SICs) alleviates or eliminates various existing problems in planar ICs. A 3D-SIC consists of multiple stacked planar dies, fabricated in a conventional process augmented by new Through Silicon Via (TSV) proces steps, which electrically connect the planar wafers in the vertical direction. This third (vertical) dimension offers relaxation of the routing complexity experienced in planar circuits. An efficient partitioning of IP cores among the stacked dies reduce the need for long wires and is thus able to positively effect the wire delay, as well as the power dissipation [2]. Preliminary analyses in [3] demonstrate the huge advantages of using a smaller wire length distribution for 3D circuits.

Heterogeneous integration is a promising concept for 3D-SICs, since each stratum can be manufactured with different technology and optimized for e.g., speed or area. This affects the yield, performance, and lithography cost positively. Furthermore, miniaturization of the physical sizes of stacked dies reduces the footprint size and volume area, and increases the package density. Examples of 3D-SICs include 3D CMOS sensors [4], 3D FPGAs [4], 3D processors [5], 3D cache and memory [6, 7], and combined stacks of memories and processors [4, 8, 9].

Tiers are stacked at the die or wafer level and can be stacked based on Wafer-to-Wafer (W2W), Die-to-Wafer (D2W) or Die-to-Die (D2D) bonding. In W2W bonding, complete wafers are stacked and bonded together. One of the benefits of W2W stacking is the high manufacturing throughput due to single wafer alignment [10]. High alignment accuracy can also be applied to D2W and D2D, but it negatively affects the throughput due to many dies that have to be aligned [10]. However, the yield loss for 3D-SICs is one of the major bottlenecks that must be overcome for 3D technology to make it a lucrative business [11]. The major limitations of W2W stacking is the rapid compound yield decrease as the number of layers in the stack increases. The compound yield can be improved by wafer matching, initially introduced by Smith et al. [12]. In wafer matching, a software algorithm keeps track of the fault map of each wafer, and matches wafer pairs that contain the same or similar fault distribution. This increases the 3D compound yield over randomly stacked wafers. More elaborate studies of wafer matching are performed in [14, 15]. Nevertheless, all the published work considered wafer matching with *static* repositories, i.e., after wafer selection the repositories are not replenished, unless they are empty. In addition, these papers focused on matching of the good dies from the bottom layer with the good dies from the top layer. Besides this matching criterion, the matching algorithm could for example match the faulty top dies that land on the faulty bottom dies.

In this paper, the impact of *replenished* repositories on the compound yield by using different wafer matching criteria is investigated. In this case, when a wafer is selected from a repository, its empty spot is directly replenished with a new one. This keeps the size of this *running* repository constant over time. The main contributions of this paper are as follows.

- A new framework is presented that covers all matching processes and wafer matching criteria for both *static* and *running* repositories.
- The impact of several matching processes and wafer matching criteria on the compound yield of 3D-SICs is illustrated.
- The impact of running repositories on the 3D-SIC compound yield is demonstrated.
- A comparison between the yield benefits gained from static and running repositories over random stacking is presented.

The remainder of the paper is organized as follows. Section 2 provides an overview of the prior work in the area of wafer matching. Section 3 introduces the framework for wafer matching and defines the focus of this paper; the framework includes the matching processes, wafer matching criteria and is applicable for both static and running repositories. Section 4 describes the wafer matching scenarios used in the yield experiments. Section 5 presents the simulation results. Section 5.4 compares the obtained results with the related work. Section 6 concludes the paper.

2 Related Prior Work

Improving the yield for 3D circuits based on wafer matching was initially introduced by Smith et al. [12], where the authors compare the yield improvement of a single die SoC, by mapping it into a 3D-SICs with two equal sized layers. The yield improvement is both simulated for D2W and W2W stacking. In the W2W stacking case, a software matching algorithm is used to select pair-wise the best wafers from two repositories with a size of 25 each. The wafer fault map is based on a random generation.

In [13], Ferri et al. used wafer matching to increase the *parametric yield* of a two layered D2W stacked 3D-SIC. Only functional dies are considered in this case to produce an optimal binning; i.e., maximize the fastest speed bins and minimize the slowest ones. Wafer matching is then used to combine and improve the 3D parametric yield by including the process variation of both layers in a D2W stacking approach. The authors were able to increase the number of 3D-SICs in the fastest speed bins as well as simultaneously reducing the number of slow 3D-SICs.

The concept of W2W matching introduced by Smith [12] is further generalized by Reda et al. [14]. The paper formulates the W2W matching problem and proves it to be $\mathcal{NP}$-hard. Several matching processes and wafer matching algorithms are investigated, including the optimal hard one. In [15], Verbree et al. define a mathematical model for wafer matching. The model has some practical limitations, but nevertheless it gives a good indication of the yield improvements. The authors include wafer matching simulations for a greedy algorithm that address the limitations. In addition, the authors justify their pre-bond test cost required for wafer matching.

All the previous work considered *static* repositories and used a *single* wafer matching criterion.

3 Wafer Matching Framework

As it has already been mentioned, W2W where the entire wafers are stacked, provides the highest manufacturing throughput and is suitable for wafers with identical die sizes and/or small die sizes. However, it suffers from lower compound yield, as one can not avoid the stacking of bad dies on good dies and vice versa. *Wafer matching* can be performed on repositories of wafers in order to find out the best wafer combinations that would result in the highest yield, given that the wafers were tested before the bonding. This section define a framework for all possible *wafer matching scenarios* for 3D W2W stacked ICs; a wafer matching scenario combines different aspects at a time e.g., wafer matching criterion, static versus running repositories etc. In the rest of the section, first the problem of W2W 3D-SICs is defined, then the aspects of wafer matching scenarios are addressed, thereafter the wafer matching framework is given.

3.1 W2W 3D-SIC Problem

The problem of W2W 3D-SICs can be defined as follows: Given, (a) n number of repositories each with k wafers, (b) a fault map of all the wafers (based on pre-bond testing), and (c) a production size of m 3D-SICs, the purpose is to maximize the overall compound yield for all m 3D-SICs, by selecting appropriate wafers for the n-layer 3D-SICs from the repositories. Figure 1 shows two freedom degrees to create 3D stacks. The vertical direction considers the wafers and the selection freedom here is the number of wafers that are selected to be stacked simultaneously. This can be either one wafer at a time (*Wafer-by-Wafer*) or k wafers at a time (*All-Wafers*). The horizontal direction shows the freedom selection from the number of layers that are considered simultaneously for stacking. This can be for either two layers at a time (*Layer-by-Layer*) or n layers at a time (*All-Layers*).

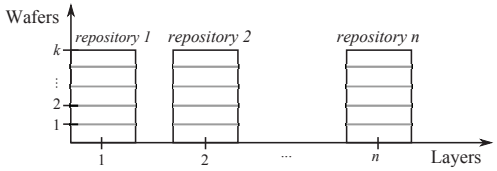


Figure 1. Wafer vs layers in 3D stacking

A wafer matching scenario can be partitioned in the following subspaces:

1. *Static versus running repositories*
2. *Wafer matching process*; e.g., how many wafer and/or layers are considered at each step
3. *Wafer matching criterion*; e.g., select the matching based on the good matched dies

By combining the subspaces several matching scenarios can be defined. Each subspace is addressed.

3.2 Static Versus Running Repositories

Wafer matching can be considered as a time consuming problem, if the objective is to obtain the global compound yield for a production size m. The production size can be in the order of thousands or millions of 3D-SICs. To split up and divide the problem, a fixed number of k (usually $k \ll m$) wafers can be considered at a time and matched. Depending on either a repository is replenished immediately after a wafer is removed from it for matching and stacking, two classes can be defined:

- *Static repositories*: From each repository k wafers are selected and processed before considering the next group of k wafers. The procedure stops after m/k steps.
- *Running repositories*. The repositories are immediately replenished with a new wafer each time a wafer is selected. The procedure stops after m wafers are processed.

The freedom to select wafers from static repositories reduces over time, since the cassettes become more and more empty. In the beginning, wafers can be selected and matched with much freedom, but when the cassette get emptier, the selection freedom diminishes. For running repositories, the repository provides always the full repository (of size k) to select from. This improves the effectiveness of the algorithm over the case where the repositories are static. The downside of running repositories is that unattractive wafers may remain in the repository for many iterations, occupying space, and in effect reducing the size of the repository in long run. We call this effect, the repository *pollution*.

A second choice which differs between static and running repositories is its actual implementation. While static repositories map

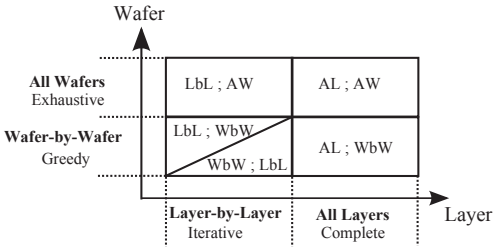


Figure 2. Framework of matching processes

fairly well onto a production line, where basically the repositories are the wafer containers that move from one machine in the production line to the next. With running repositories, a container would need to go back and forth between the bonding machine and the wafer production line to be replenished, before a new selection can be made. Clearly this is highly impractical, and therefore we suggest using two containers. One to do the selection from, and one that is used as a wafer source to replenish the first one at the bonding machine. This, however, reduces the effective capacity of the bonding machine, as both containers are in the machine, yet only one is used to select a wafer from.

3.3 Matching Process

The matching process defines the step-by-step process to be followed in order to realize the matching for each wafer from each repository with wafers from other repositories. The matching process, therefore determines the *number of repositories* and the *number of wafers* that are considered at a time.

Depending on the number of involved repositories, two cases are distinguished:

- Layer-by-Layer (LbL): In this case, initially, the first two repositories are selected for wafer matching. In each additional step, only one additional repository is used during matching. Hence, this is an *iterative* process in terms of the number of layers involved.
- All-Layers (AL): In this case, in each step of the wafer matching process, all repositories are used at once. As every wafer in every repository is taken into account, this process is labeled *complete*.

In a similar way, depending on the k wafers involved in each step of the matching process, two cases can be distinguished:

- Wafer-by-Wafer (WbW): In this case, in each step of the wafer matching process, the best wafers contributing to the possible match are selected. One from each repository involved in the process. With no regard to the remaining wafers in each of those repositories. Thus this process is regarded as *greedy*.
- All-Wafers (AW): In this case, in each step of the wafer matching process, all wafers from all involved repositories are matched. The best overall solution is selected for that set, and as the process considers all possible outcomes, for all k wafers to be matched, this process is considered *exhaustive*.

The above combinations result in five possible wafer matching processes, as shown in Figure 2.

- WbW;LbL: The matching process steps are iterative over the repositories. In each step, only two repositories are used. In addition, only a single wafer pair selection is performed at a time (one from each repository). After n iterations a compound is created and the process repeats itself by starting again at the first two repositories.
- LbL;WbW: The matching process steps are iterative over the repositories. Again, in each step, only two repositories are considered. The difference with the WbW;LbL is that, in this process k wafers are iteratively matched first before the next repository is used.
- LbL;AW: Similar to LbL;WbW, the matching process iteratively considers two repositories at a time, but in this case, all wafers from the two repositories under consideration are matched. Note that, this matching process is only applicable to static repositories, since running repositories are replenished each time a wafer is selected out of them. The difference between LbL;AW is that, in this process an optimal solution is considered within the LbL process, while LbL;WbW selects the wafers one by one in a greedy way.
- AL;WbW: The matching process considers all repositories simultaneously in each matching step, and select the best matching combination of n wafers along the repositories. The same step is repeated over time. In the case of static repositories, the selection of best matching n wafers is performed, first along the n repositories each with k wafers, in the second step with $k-1$ wafers, etc. In case of running repositories, the matching considers always k wafers from each repository.
- AL;AW: This is similar to AL;WbW, but here all k wafers from each repository are matched simultaneously. Note that this matching process is only applicable to static repositories.

The iterative LbL and iterative WbW have two variants. The differences between the variants are a result of exchanging the inner and outer loops of the processes. For the LbL processes, an additional freedom can be defined for the traversal order for the repositories. The number of freedom possibilities to step over the repositories equals to $\binom{n}{2} \cdot (n-2)! = \frac{n!}{2}$. In the left hand side of the equation, the first term represents the number of possibilities to select the first two repositories out of n repositories. The second term $(n-2)!$ presents the number of combinations of the remaining repositories.

3.4 Matching Criteria

The matching processes select wafers based on certain criteria; e.g., best good dies. Each criterion is orthogonal with respect to the process. Using the fact that each wafer consists of both good dies and bad dies and that the purpose of the matching process is to maximize the compound yield, one can define three possible criteria: (a) maximize the matching good dies, (b) maximize the matching faulty dies, and (c) minimize the matching between good dies and bad dies. The criteria are defined as follows:

- *Max(MG)*. This criterion selects the best wafer pair combinations based on the maximum *matched good* dies. All the published work so far regarding wafer matching considers only this criterion.
- *Max(MF)*. This criterion selects the best wafer pair combinations based on maximum *matched faulty* dies.
- *Min(UF)*. This criterion selects the best wafer pair combinations based on minimum *unmatched faulty* dies. The objective is to increase the compound yield by minimizing faulty dies that land on good dies and vice versa.

All the above criteria produce the same result in terms of compound yield, in case the wafer decision selection is exhaustive (AW process)

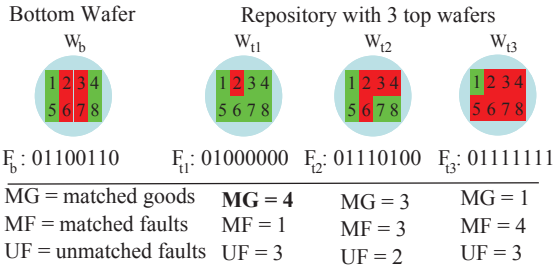


Figure 3. Wafer matching criteria

Table 1. W2W Matching Framework

Matching Process	Static repository	Running repository
WbW;LbL	yes	yes
LbL;WbW	yes (greedy [15])	yes
LbL;AW	yes (IMH [14])	n.a
AL;WbW	yes (Greedy [14])	yes
AL;AW	yes (ILP/UB [14])	n.a

n.a denotes not applicable

for static repositories. For the greedy wafer decision selection processes (WbW), it is evident that different criteria lead to different results due to the greediness of the algorithm. For running repositories, the criteria lead to different compound yields as will be explained next.

In order to provide more insight into the impact of the above criteria on wafer selection, refer to the example shown in Figure 3, which considers a bottom wafer W_b and three potential top wafers (W_{t1}, W_{t2}, W_{t2}), each with its own fault map. The fault map of each wafer is denoted by F and contains a sequence of 0s (good dies) and 1s (bad dies) in the same order of the indices of the dies on the wafer, e.g., the bottom wafer has $F_b = 01100110$, since the dies 2, 3, 6 and 7 are faulty. The bottom table in the figure lists the value of the different criteria for the three matching possibilities, e.g., for matching W_b-W_{t1}, the number of matched good dies is MG = 4 (which are 1, 4, 5, 8). The figure clearly shows that depending on the criterion, different top wafers will be selected, e.g., if max(MG) is considered, then W_{t1} will be selected. However, if the max(MF) is the criterion, then W_{t2} is the best match.

The criteria will be mathematically formulated. Let the function $G(F_i)$ be the number of faulty dies in the wafer with fault map F_i, then,

$$max(MG) = max(\forall_{i,j}, G(\bar{F}_i \& \bar{F}_j)) \quad (1)$$

$$max(UF) = min(\forall_{i,j}, G(F_i \oplus F_j)) \quad (2)$$

$$max(MF) = max(\forall_{i,j}, G(F_i \& F_j)) \quad (3)$$

Here, $0 \leq i, j \leq k$, where k the repository size.

3.5 The W2W matching framework

The wafer matching scenario aspects discussed in the previous section can be integrated into a complete framework that covers all wafer matching scenarios, shown in Table 1. The table shows the possible combinations of matching process and repository types (e.g., static and running repositories). Each combination results in a wafer matching scenario, when combined with a matching criteria. The matching scenarios, considered in the previous published work are included through their references in the table. The criteria are left

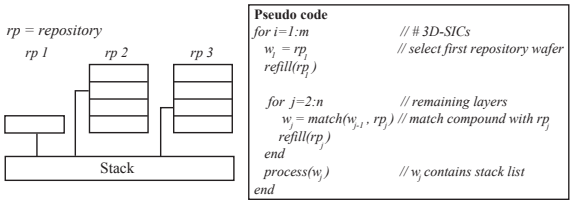


Figure 4. Matching scenario FIFO1

out, since they are independent of the matching processes. The table shows, whether for each combination between the processes and the repository types, a valid combination exists or not. Going vertically down the entries of the table, more advanced algorithms are used which in general lead to a higher compound yield, at the cost of higher computational effort. Putting the previous work in the context of the framework defined in Section 3, the following can be concluded.

In the first scenario of [14], a greedy algorithm is used to create a sorted list of all k^n possibilities. The arranged list is sorted based on the compound yield in a descending order. From this list, valid compounds are selected starting from the highest yields. A combination is considered invalid, when at least one of the wafers of the current compound has already been taken in a previous cycle. We classify this as an AL;WbW process. In the second method, the Iterative Heuristic Matching (IHM) algorithm is used, which considers two repositories at a time and optimally matches them by the Hungarian algorithm. These steps are iteratively repeated by including one additional repository in each iteration. The IHM algorithm in [14] is an LbL;AW process. In their last method, a global optimal algorithm based on Inter Linear Programming (ILP), which explores the exhaustive search space and obtains the global maximum yield, is investigated. They further reduce the execution time of the ILP method (UB) by relaxing the ILP, nd allowing the program variables to take fractional values. The ILP/UB scenarios are AL;AW processes. The greedy algorithm [15] is a LbL;WbW process.

From Table 1, we conclude that several scenarios are not explored yet, mainly the ones for running repositories. The focus in this paper is the WbW;LbL matching process for running repositories, which belongs to the class of scenarios with smallest runtime complexity. The considered matching scenarios are explained in the next section. The remaining scenarios for running repositories are not considered, since these scenarios require a longer simulation time.

4 Wafer Matching Scenarios for Running Repositories

In this section, we describe a subset of wafer matching scenarios for running repositories that are considered in the rest of the paper. For these scenarios, the following aspects will be considered.

- Running repositories.
- WbW;LbL matching process.
- All matching wafer criteria.

The following different strategies are considered: FIFO1, FIFOn and Best Pair (BP). Each strategy is a collection of three scenarios, which differ from their criterion (max(MG), max(MF) or min(UF)). For example, the strategy FIFO1 consists of three different scenarios in which each scenario differs in the wafer matching criterion. The same applies to the other two strategies. The three strategies are explained in Sections 4.1, 4.2 and 4.3 respectively. All the strategies end, when the production size m is reached.

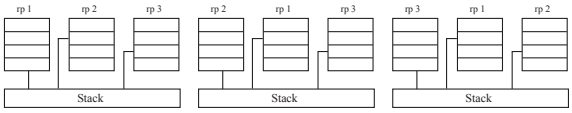


Figure 5. Matching scenario FIFOn

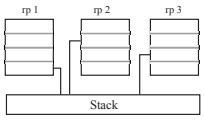


Figure 6. Matching scenario BP

4.1 FIFO1

In FIFO1, the first repository is used as a FIFO, as depicted for an example with $n = 3$ in Figure 4. The wafers from repository 1 (rp 1) are selected without any freedom and matched with the best wafer from the second repository. The algorithm iterates over all the repositories and selects a single wafer out of them. The order in which the repositories are traversed is linear from bottom layer 1 till top layer n. The worst case memory complexity equals $O(k)$. This is the memory required to store k compound yield numbers to be able to select 1 out of k wafers. The runtime complexity is $O(m \cdot k \cdot (n-1)) = O(m \cdot k \cdot n)$.

4.2 FIFOn

In FIFOn, we generalize this concept by moving the FIFO repository in a round robin fashion among all repositories. This is shown in Figure 5 for $n = 3$. At the left side of the figure, repository one (rp 1) is used as a FIFO. After an n-compound stack is created, the repository belonging to the next layer is considered to be the FIFO, as shown in the middle of the picture. Here, the algorithm starts from repository 2 (rp 2) and proceeds next with repositories 1 and 3. For the next compound, repository 3 (rp 3) is used as a FIFO. These steps are repeated until the production size is reached. The first traversed repository is the repository that is considered as the FIFO, the remaining repositories are traversed in monolithic increasing order from layer 1 to layer n. The traverse order is written in the top part of Figure 5. The FIFOn forces wafers to stay maximally $n \cdot k$ cycles in a repository. The purpose of FIFOn is to control the pollution of the repositories. The repositories could get contaminated with bad wafers that stay for a long time in the repositories without being selected. The memory and runtime complexity for this scenario are the same as in the case for FIFO1.

4.3 Best Pair

The last scenario we present, is the Best Pair (BP) matching strategy. In this strategy, the best wafer pair between the first two repositories are selected, as seen in Figure 6. The process iteratively proceeds along the repositories until a single n-compound match is determined. Then, this process is repeated until the production size m is met. On one side, best pair has more freedom in wafer selection (due to absence of the FIFO mechanism), but on the other side, it does not guarantee a higher compound yield as compared to FIFOn. The reason for this is, that it lacks controlling the repository pollution. For this scenario, the memory complexity equals $O(k^2)$, required to store all compound yield combinations between the first repositories. The runtime complexity equals $O(m \cdot k \cdot n + k^2) = O(mnk)$. Here, we initially calculate k^2 compound yield combinations for the first two repositories and update each time $2 * k - 1$ numbers for these repositories, when new wafers enter the repositories. To complexity to select $n - 2$ single wafers from the remaining repositories equals to $(n-2) * k$.

5 Simulation Results

In this section, several experiments are considered for each matching scenario to be able to measure their quality in terms of compound yield. To achieve this, different planar wafer yields and a different number of stacked layers are simulated. The process parameters of the experiments are described in Section 5.1. We investigate the aspects of each strategy by considering its repository type, matching process and criterion. We investigate the impact of running repositories in Section 5.2, subsequently followed by the impact of the wafer matching process and criterion in Section 5.3. Finally, we select our best wafer matching scenario and compare it with the related work in section 5.4.

5.1 Experimental Setup

In this section, we describe our simulation experiments that resemble real case scenarios. The experiments are based on the reference process in [15]. A standard 300 mm diameter wafer is selected with an edge clearance of 3 mm. The defect density is considered to be d_0 = 0.5 defects/cm^2 and the defect clustering parameter $\alpha = 0.5$. For the reference design, the die area is assumed to be $A = 50\ mm^2$. For this die area and wafer size, the number of Gross Dies per Wafer (GDW) approximately equals 1278 [17]. The expected yield of the wafers can be estimated by the negative binomial formula for yield, $y = (1 + \frac{A \cdot d_0}{\alpha})^{-\alpha}$= 81.65% [18].

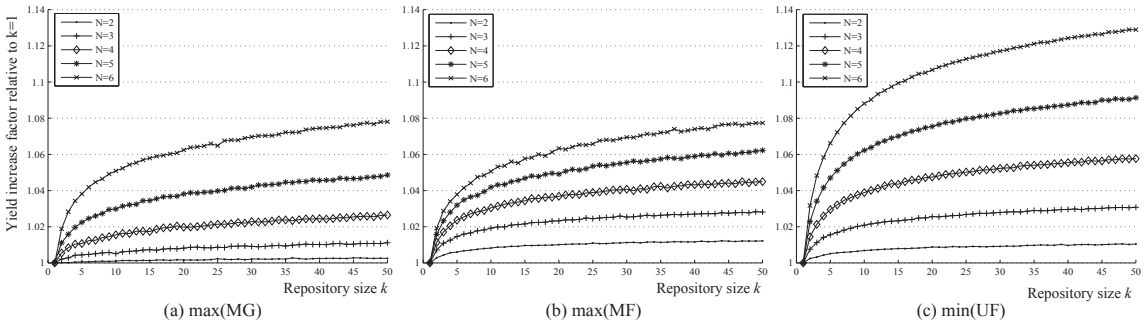
In our experiments, we simulate a production size $m = 25000$. Here, m is the number of produced 3D-SICs. Initially, each repository is filled up with k wafers and after selecting and stacking m-compound wafers, the wafers that are left in the repository are discarded and not included in the simulation results for two reasons.

- First, we want to observe the impact of the running repository only.
- Second, even if the wafers would be thrown away, their impact on the compound yield is minimal (k/m) due to a high production volume m. Actually, the matching scenarios presented in [14] and [15] for static repositories could be used to match these last k unconsidered wafers.

5.2 Impact of Running Repositories

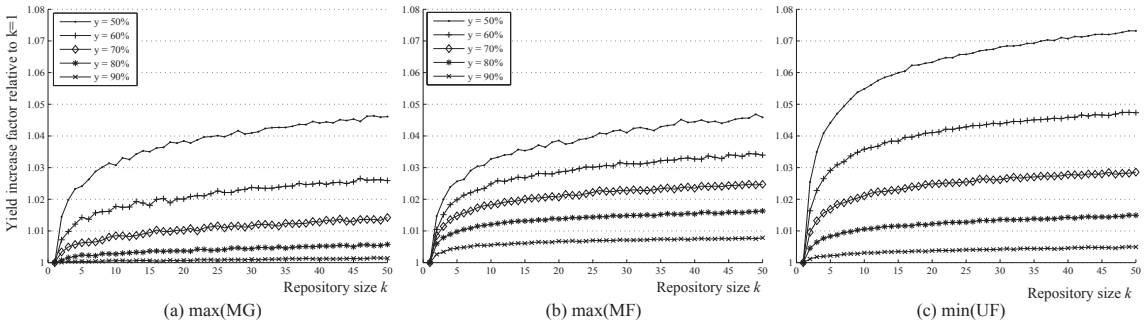
To measure the impact of running on the compound yield, repository size and repository pollution three experiments will be considered:

- In experiment 1, the impact on the compound yield for different stacked number of layers n ($2 \leq n \leq 6$) for various repository sizes are examined. Here, the reference process is considered and all criteria are simulated for each scenario.
- In experiment 2, we adjust the yield of the reference process over a wide range to simulate the impact of the yield learning curve on stacked 3D-SICs. We consider a stack of two layers and vary the repository size. For experiments 1 and 2, we consider the FIFO1 strategy for all criteria.
- The last experiment consists of indirect measurement of the repository pollution. By plotting the compound yield for different stack sizes versus different production sizes m, we can indirectly measure the pollution that takes place and observe



(a) max(MG) (b) max(MF) (c) min(UF)

Figure 7. Impact of n and k on compound yield for FIFO1 using the reference process



(a) max(MG) (b) max(MF) (c) min(UF)

Figure 8. Impact of wafer yield and k on the compound yield for FIFO1

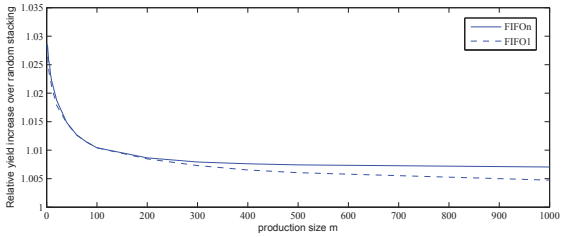


Figure 9. Yield decrease as a result of wafer pollution for larger production sizes.

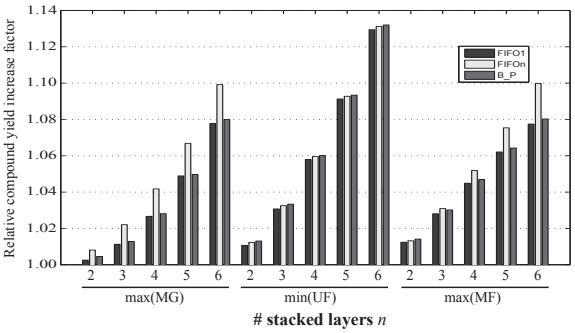


Figure 10. Yield increase by varying n

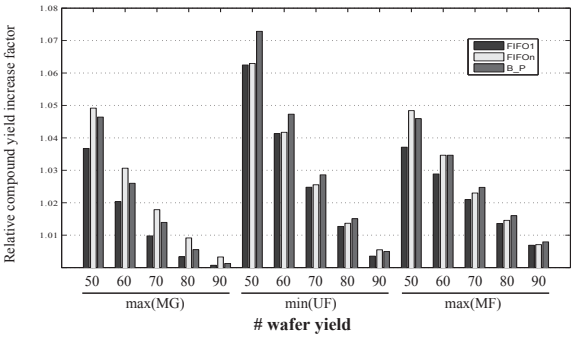


Figure 11. Yield increase by wafer yield variation

the effect on the compound yield. Moreover, we look at the compound yield differences between FIFO1 and FIFOn.

Figure 7 plots the relative compound yield increase for FIFO1 with respect to random stacking ($k = 1$). The figure shows the compound yield increase for different stacked layers n and repository sizes k for each criteria. The results of the second experiment, varying the wafer yield and repository sizes, is shown in Figure 8.

The following is concluded from both figures: The relative compound yield increases with larger repositories, but the obtained gain stabilizes for each criteria. The compound yield improvement versus random stacking improves with (a) bigger stack sizes, and (b) lower wafer yield.

To show that there is indeed wafer pollution, Figure 9 plots the relative compound yield for FIFOn and FIFO1 over random stacking for different production wafer sizes m. Here, the reference process is used with $n = 2$, $k = 25$ and the matching criterion max(MG). Three observation can be seen from the graph:

- The relative yield for both FIFO1 as well as FIFOn decreases with increasing production size. For low m, many good wafer choices can be selected and a high yield is obtained. However, as more wafers are selected, the pollution is taking place and the yield drops.
- With increasing m, the compound yield of FIFO1 decreases faster than FIFO1, while the scenarios only differ in the repository which is used as a FIFO. FIFOn forces wafers to leave the repository after $k \cdot n$ cycles and it has a positive affect on the yield.
- The pollution is stabilizing for larger m.

5.3 Impact of Wafer Matching Criteria

In this section we consider the impact of the matching criteria. We first consider the different matching criteria in Figures 7 and 8. For the given wafer yield of 81.65%, the criteria min(UF) outperforms the other two criteria in Figure 7 for $n \geq 3$. For $n = 2$, the criterion max(MF) performs the best. From the curves in Figure 8, we conclude that min(UF) performs the best for wafer yields in the range of 50%-70%. For higher wafer yields (80% and above), the criteria max(MF) performs the best.

To obtain a complete picture of the matching strategy and its criteria for all the strategies, FIFO1, FIFOn and BP, the same experiments 1 and 2 are replicated, for all these strategies with a fixed repository size of $k = 50$. Figure 10 and 11 show the results.

From both the graphs we conclude the following:

- In general, a higher improvement can be gained for larger stack sizes and lower wafer yields. Note that, when the stack size increases, the compound yield decreases.
- FIFOn always performs better than FIFO1 for the same conditions, especially for the criteria that relatively perform poor (i.e, the ones from Figure 7). For the min(UF) criterion, little improvement is gained over FIFO1, which means that small pollution is taking place for this criterion.
- The BP scenario always performs better than FIFO1 for the same experiment conditions, but not always better than FIFOn. BP is unable to control repository pollution and therefore it is not the best option to select, in case the criterion performs poor. Nevertheless, the overall highest compound yield improvements are a result from this scenario. For the stacked layers, i.e., for $n = 2$ BP max(MF) and for $3 \leq n \leq 6$ BP min(UF) perform the best. For the wafer yields y, i.e., for $y \in \{50\%, 60\%, 70\%\}$ BP min(UF) and for $y \in \{80\%, 90\%\}$ BP max(MF) perform the best.

The events in terms of different yields for different criteria that take place here and in the Figures 7 and 8, are explained as follows. In case, the wafer yield is low, the majority of the dies are faulty. The max(MG) tries to match the good stacked dies only and since these are in minority, the choice to select is relatively easy. The max(MF) however, has much more difficulties to stack the wafers, since the majority of dies is faulty anyway. This affects the wafer lifetime and yield negatively. In contrast, max(MF) performs better with higher wafer yields and max(MG) worse for that case. The min(UF) performs the best in midrange wafer yields, where the probability of the presence of good and bad dies is similar.

The question rises that which combination of the three wafer scenarios and matching criteria must be selected for a certain process to maximize the compound yield. Table 2 simplifies this decision. The table shows the criteria for different top wafer yield y_t and bottom wafer yield y_b that must be selected to achieve this highest compound yield.

Table 3 shows the criteria for different combinations of wafer yield and the number of stacked layers that must be selected to attain the highest compound yield. Only a single criterion is considered in the stack.

From both tables, the following can be concluded.

- When the wafer yield decreases or the stack size increases, the max(MG) criterion should be selected.

Table 2. Optimal criterion selection for given top and bottom wafer yield in (%)

$y_t \backslash y_b$	10	20	30	40	50	60	70	80	90
10	MG	MG	MG	UF	UF	UF	MG	MG	MG
20	MG	MG	UF	UF	UF	UF	UF	MF	MF
30	MG	UF	UF	UF	UF	UF	UF	UF	MF
40	UF	UF	UF	UF	UF	UF	UF	UF	UF
50	UF	UF	UF	UF	UF	UF	UF	UF	UF
60	UF	UF	UF	UF	UF	UF	UF	UF	UF
70	MG	UF	UF	UF	UF	UF	UF	UF	MF
80	MG	MG	UF	UF	UF	UF	UF	MF	MF
90	MF	MF	MF	UF	UF	UF	MF	MF	MF

Table 3. Single criterion that leads to maximal compound yield

Yield	n = 2	n = 3	n = 4	n = 5	n = 6
10%	max(MG)	max(MG)	max(MG)	max(MG)	max(MG)
20%	max(MG)	max(MG)	max(MG)	max(MG)	max(MG)
30%	min(UF)	max(MG)	max(MG)	max(MG)	max(MG)
40%	min(UF)	max(MG)	max(MG)	max(MG)	max(MG)
50%	min(UF)	min(UF)	max(MG)	max(MG)	max(MG)
60%	min(UF)	min(UF)	min(UF)	max(MG)	max(MG)
70%	min(UF)	min(UF)	min(UF)	min(UF)	min(UF)
80%	max(MF)	min(UF)	min(UF)	min(UF)	min(UF)
90%	max(MF)	max(MF)	max(MF)	min(UF)	min(UF)

Table 4. Yield comparison with [15] for $n = 2$, $k = 50$ and $d = 1278$

yield	Greedy [15] (%)	BP (%)	$\frac{\text{BP}}{\text{Greedy [15]}}$ (%)	$\frac{\text{BP(k=50)}}{\text{random}}$ (%)
0.5	26.2	26.82	2.4	7.28
0.6	37.2	37.71	1.4	4.75
0.7	50.0	50.40	0.8	2.86
0.8	64.7	64.96	0.4	1.50
0.9	81.3	81.64	0.4	0.79

- When the wafer yield is in the mid-range values or when moderate stack sizes are considered, the min(UF) criterion is selected.
- For very high yields, it is more advantageous to select the max(MF) criterion up to a certain stack size.

5.4 Comparison of Wafer Matching Scenarios

In this section, we select our best matching strategy in combination with the best criteria and use this scenario to compare our algorithm with the scenarios of static repositories mentioned in [14, 15]. As already mentioned in the previous section, the strategy BP performs best if the correct criterion is considered. The same pattern was observed for each entry of Table 2. An adaptive scenario with respect to the wafer selection, is created when Table 2 is implemented for the BP strategy. The new scenario always selects highest yield for all simulation parameters. From now on, we refer to the BP process as the process that adapts itself with respect to the criterion selection. In our simulations, Table 2 is used iteratively over the layers. During the iterations over the repositories to select n wafer, we replace y_b with the compound yield of the so far selected wafers. This enables us to change the criterion on the fly. This adaptive scenario is used for the comparison experiments.

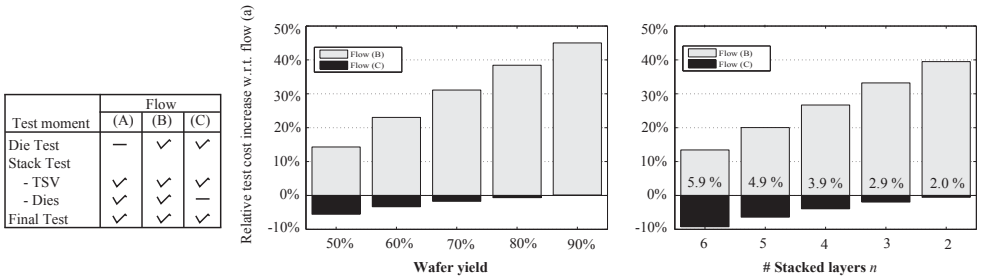
To consider the impact of running repositories versus static repositories, we repeat the same experiments as in [14, 15]. We consider two published extremes for static repositories, on one side the greedy scenario in [15] and on the other side the optimal $\mathcal{NP}$-hard scenario

Table 5. Yield comparison with [15] for $k = 50$, $y = 81.61\%$ and $d = 1278$

n	Greedy [15] (%)	BP (%)	$\frac{\text{BP}}{\text{Greedy [15]}}$ (%)	$\frac{\text{BP(k=50)}}{\text{random}}$ (%)
2	67.3	67.56	0.4	1.43
3	55.7	56.24	1.0	3.47
4	46.3	47.10	1.7	6.18
5	38.7	39.65	2.5	9.53
6	32.5	33.50	3.1	13.39

Table 6. Yield comparison with [14] for $n = 3$, $k = 25$, $d = 590$

yield



(a) Test moments (b) Test impact on yield (c) Test impact on # layers

Figure 12. Normalized test cost for wafer matching test flows (B) and (C) relative to the random W2W stack flow (A)

Table 8. Wafer matching scenarios complexities

Ref	Scenario	Memory complexity	Runtime complexity
[14]	Greedy	$O((n+1)\cdot k^n)$	$O(m\cdot k^{n-1}\cdot log(k))$
[14]	IMH	$O(k^2)$	$O(m\cdot n^2\cdot k^2)$
[14]	ILP/UB	$O((n+1)\cdot k^n)$	$O(\frac{m}{k}\cdot (k!)^{n-1})^*$
[15]	Greedy	$O(k^2)$	$O(m\cdot k^2\cdot n)$
This	Fifo1	$O(k)$	$O(m\cdot k\cdot n)$
This	Fifon	$O(k)$	$O(m\cdot k\cdot n)$
This	Best Pair	$O(k^2)$	$O(m\cdot k\cdot n)$

* denotes the complexity of the search space

an instance for $n = 3$ and 40.64 seconds for $n = 4$ and runs out of memory for more stacked layers [14]. For the same parameters, our wafer adaptive BP scenario implemented in Matlab only required 0.0028 seconds to match a single compound for $n = 7$. For each scenario, Table 8 contains the memory and runtime complexity cost of each algorithm. The table shows the memory and time complexity for each discussed algorithm.

To justify the additional costs required for pre-bond testing, we compare three test flows in [15] depicted in Figure 12(a). Pre-bond die tests are required in W2W matching and these are performed before stacking takes place. The stack tests verify the stacked wafers before they are packaged and bonded. The final tests ensure overall chip functionality.

The three test flows in [15] are:

- Flow (A) includes a stacking test and a final test, but has no pre-bond die tests. This flow is applicable for random wafer-to-wafer stacking.
- Flow (B) consists of pre-bond die tests (required for wafer matching), a stacking test that tests both dies and interconnects, and a final test.
- Flow (C) consists of pre-bond die tests, a stacking test for the interconnects only and a final test. The idea in this flow is to optimize the wafer test flow (B) by not replicating the die test in the stacking test, which already has been tested in the pre-bond phase. As a consequence of faults introduced into the dies during stacking, a small percentage of faulty dies is still packed.

The test cost per functional good die in terms of test time for the test flows (B) and (C) relative to flow(A) are shown in Figure 12(b) and (c). The absolute number variation from 2.0-5.9% in Figure 12(c) presents the package waste cost. For Figure 12(b) this package waste is equal to 2.0%, for all different yields [15]. The following is concluded.

- Flow (C) has a positive affect on the test cost, except when the wafer yield is high. For this case, a small increase in test cost is expected, as for the case of a wafer yield of 90% in Figure 12(b).
- There is a small additional packaging cost depending on the number of layers in the stack.

Except reducing the test cost, the yield impact is also improved as shown in Tables 4 and 5. For example, in Figure 12(c), for a two-stacked 3D-SIC, the test time reduction is 0.55 %, the yield is increased with 1.41 % (Table 5), while the packaging cost is increased with 2 %. For a six-layered stack, a test cost reduction of 9.23 % is expected with a yield increase of 13.19 %, but with a package cost increase of only 5.9 %. Since the compound yield for running repositories is higher than in the case of static repositories with equal cost in pre-bond die tests, we can safely conclude that the test time per functional working die is lower than in the case of static repositories. The equivalent graph of Figure 12 in [15] confirms this.

6 Conclusion

In this paper, a framework for 3D Wafer-to-Wafer matching for different wafer matching scenarios is presented. Each scenario is a combination of a matching processes, wafer matching criterion, and the repository type (e.g., running or static repositories). Several scenarios with running repositories were investigated with realistic wafer yields and various stack sizes. The results show that a compound yield improvement up to 13.4% can be obtained relative to random stacking. This was possible by the integration of the matching criteria into an adaptive scenario with respect to the criterion selection. With this adaptive scenario we are able to outperform optimal scenarios for static repositories in terms of compound yield given the same simulation parameters. In addition, our adaptive scenario reduces test cost, memory requirement and runtime complexity compared to the previous work for static repositories.

References

[1] D. Sylvester and C. Hu, "Analytical modeling and characterization of deep-submicrometer interconnect", *Proc IEEE* ,Volume: 89, pp. 634-664, May 2001.

[2] W. Rhett Davis et al., "Demystifying 3D ICs: The Pros and Cons of Going Vertical", *IEEE Design Test on Computers*, 22(6):498-510, Nov 2005.

[3] J. A. Davis et al. "Interconnect Limits on Gigascale Integration (GSI) in the 21st Century", *Proc. IEEE*, vol 89, Issue 3, Pages:305-224, 2001.

[4] Robert S. Patti, "Three-Dimensional Integrated Circuits and the Future of System-on-Chip Designs", *Proceedings of the IEEE*, Vol 94, Issue 6, June 2006.

[5] G. Loh et al. "Processor design in 3D die-stacking technologies", *IEEE Micro*, Volume 27, Issue 3, pp. 31-48, August 2007.
[6] K. Puttaswamy et al. "3D-Integrated SRAM Components for High-Performance Microprocessors", *IEEE Transactions on Computers*, Vol 58, Issue 10, pp. 1369-1381, August 2009.
[7] Yuh-Fang Tsai et al. "Design Space Exploration for 3-D Cache", *IEEE Transactions on Very Large Scale Integration Systems*, Vol 16, Issue 4, pp. 444-455, 2008.
[8] Thorlindur Thorolfsson et al. "Comparative analysis of two 3D integration implementations of a SAR processor", *IEEE International Conference on 3D System Integration*, October 2009.
[9] F. Li et al. "Design and Management of 3D Chip Multiprocessors Using Network-in-Memory", *International Symposium on Computer Architecture*, pp. 130-141, July 2006.
[10] Philip Garrou, Christopher Bower and Pater Ramm, "Handbook of 3D Integration", Wiley-VCH, 2008.
[11] J. Baliga, "Chips go vertical", *IEEE Spectrum*, Vol 41, Issue 3, pp. 43-47, March 2004.
[12] L. Smith, G. Smith, S. Hosali, and S. Arkalgud, "Yield considerations in the choice of 3D technology", *In Proc. IEEE Int. Symp. Semiconductor Manufacturing*, pp. 535-537, 2007.
[13] C. Ferri, S. Reda, and R. Iris Bahar, "Parametric yield management for 3D ICs: Models and strategies for improvement", *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, Vol 4, Issue 4, October 2008.
[14] 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*, Vol 17 , Issue: 9, pp. 1357 - 1362, 2010.
[15] Jouke Verbree et al. "On the Cost-Effectiveness of Matching Repositories of Pre-Tested Wafers for Wafer-to-wafer 3D Chip Stacking", Paper accepted in: *IEEE European Test Symposyum*, May 2010.
[16] Erik Jan Marinissen and Yervant Zorian, "Testing 3D Chips Containing Through-Silicon Vias", *International Test Conference, 2009*, Nov 2009.
[17] Dirk K. de Vries, "Investigation of Gross Die Per Wafer Formulas", *IEEE Transactions on Semiconductor Manufacturing*, Vol 18, Issue 1, pp. 136-139, Feb 2005.
[18] M. Bushnell and Vishwani Agrawal, "Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits", Wiley-VCH, Weinheim, Germany, August 2000.

Bibliography

[1] G. Smith *et al.*, "Yield Considerations in the Choice of 3D Technology," in *International Symposium on Semiconductor Manufacturing (ISSM)*, Oct. 2007, pp. 1–3.

[2] S. Reda, G. Smith, and L. Smith, "Maximizing the Functional Yield of Wafer-to-Wafer 3-D Integration," *IEEE Transactions on VLSI Systems*, vol. 17, pp. 1357–1362, Sep. 2009.

[3] IEEE Computer Society, *IEEE Std 1500™-2005, IEEE Standard Testability Method for Embedded Core-based Integrated Circuits*. New York, NY, USA: IEEE, Aug. 2005.

[4] M. Bushnell and V. Agrawal, *Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits*. Weinheim, Germany: Wiley-VCH, Aug. 2000.

[5] A. Milenkovic and V. Milutinovica, "A quantitative analysis of wiring lengths in 2D and 3D VLSI implementation of 2D systolic arrays," *Microelectronics Journal*, vol. 29, no. 6, pp. 313–321, Aug. 1998.

[6] F. N. M. B. X. E. T. L. V. B.-M. C. C. J. A. B. S. N. Henry, D. Jacquet, "Through silicon vias technology for CMOS image sensors packaging," in *Proceedings IEEE International Test Conference (ITC)*, May 2008.

[7] Michael R. Garey and David S. Johnson, *Computers and Intractability – A Guide to the Theory of NP-Completeness*. San Francisco: W.H. Freeman and Company, 1979.

[8] J. Schat, "Fault Clustering in Deep-Submicron CMOS Processes," in *Proceedings Design, Automation, and Test in Europe (DATE)*, Mar. 2008, pp. 511–514.

[9] D. K. de Vries, "Investigation of Gross Die Per Wafer Formulas," *IEEE Transactions on Semiconductor Manufacturing*, vol. 18, no. 1, pp. 136–139, Feb. 2005.

[10] SEMI, "SEMI G81-0703, Specification for Map Data Items," http://www.semi.org.

[11] ——, "SEMI G85-0703, Specification for Map Data Format," http://www.semi.org.

[12] E. J. Marinissen and Y. Zorian, "Testing 3D Chips Containing Through-Silicon Vias," in *Proceedings IEEE International Test Conference (ITC)*, Nov. 2009.

[13] ——, "IEEE 1500 Enables Modular SOC Testing," *IEEE Design & Test of Computers*, vol. 26, no. 1, pp. 8–16, January/February 2009.

[14] IEEE Computer Society, *IEEE Std 1149.1™-2001, IEEE Standard Test Access Port and Boundary-Scan Architecture*. New York, NY, USA: IEEE, Jun. 2001.

[15] K. P. Parker, *The Boundary-Scan Handbook*, 3rd ed. Springer-Verlag, Jun. 2003.

[16] F. da Silva, T. McLaurin, and T. Waayers, *The Core Test Wrapper Handbook – Rationale and Application of IEEE Std. 1500™*, ser. Frontiers in Electronics Testing. Boston, MA, USA: Springer-Verlag, 2006, vol. 35.

[17] D. L. Lewis and H.-H. S. Lee, "A Scan-Island Based Design Enabling Prebond Testability in Die-Stacked Microprocessors," in *Proceedings IEEE International Test Conference (ITC)*, Oct. 2007.

[18] X. Wu, P. Falkenstern, and Y. Xie, "Scan Chain Design for Three-dimensional Integrated Circuits (3D ICs)," in *Proceedings International Conference on Computer Design (ICCD)*, Oct. 2007, pp. 208–214.

[19] X. Wu *et al.*, "Test-Access Mechanism Optimization for Core-Based Three-Dimensional SOCs," in *Proceedings International Conference on Computer Design (ICCD)*, Oct. 2008, pp. 212–218.

[20] L. Jiang, L. Huang, and Q. Xu, "Test Architecture Design and Optimization for Three-Dimensional SoCs," in *Proceedings Design, Automation, and Test in Europe (DATE)*, Apr. 2009, pp. 220–225.

[21] L. Jiang *et al.*, "Layout-Driven Test-Architecture Design and Optimization for 3D SoCs under Pre-Bond Test-Pin-Count Constraint," in *Proceedings International Conference on Computer-Aided Design (ICCAD)*, Nov. 2009, pp. 191–196.

[22] E. J. Marinissen *et al.*, "Minimizing Pattern Count for Interconnect Tests Under A Ground-Bounce Constraint," *IEEE Design & Test of Computers*, vol. 20, no. 2, pp. 8–18, March/April 2003.

[23] S. K. Goel *et al.*, "Test Infrastructure Design for the Nexperia$^{\text{TM}}$ Home Platform PNX8550 System Chip," in *Proceedings Design, Automation, and Test in Europe (DATE) Designers Forum*, Paris, France, Feb. 2004, pp. 108–113.

[24] S. K. Goel and E. J. Marinissen, "SOC Test Architecture Design for Efficient Utilization of Test Bandwidth," *ACM Transactions on Design Automation of Electronic Systems*, vol. 8, no. 4, pp. 399–429, Oct. 2003.

[25] P. Varma and S. Bhatia, "A Structured Test Re-Use Methodology for Core-Based System Chips," in *Proceedings IEEE International Test Conference (ITC)*, Washington, DC, USA, Oct. 1998, pp. 294–302.

[26] E. J. Marinissen *et al.*, "A Structured And Scalable Mechanism for Test Access to Embedded Reusable Cores," in *Proceedings IEEE International Test Conference (ITC)*, Washington, DC, USA, Oct. 1998, pp. 284–293.

[27] B. Swinnen *et al.*, "3D Integration by Cu-Cu Thermo-Compression Bonding of Extremely Thinned Bulk-Si Die Containing 10μm Pitch Through-Si Vias," in *Proceedings IEEE International Electron Devices Meeting (IEDM)*, May 2006, pp. 1–4.

[28] J. V. Olmen *et al.*, "3D Stacked IC Demonstration using a Through Silicon Via First Approach," in *Proceedings IEEE International Electron Devices Meeting (IEDM)*, Dec. 2008, pp. 1–4.

[29] S. K. Goel, E. J. Marinissen, A. Sehgal, and K. Chakrabarty, "Testing of SOCs with Hierarchical Cores: Common Fallacies, Test-Access Optimization, and Test Scheduling," *IEEE Transactions on Computers*, vol. 58, no. 3, pp. 409–422, Mar. 2009.

[30] T. Waayers, R. Morren, and R. Grandi, "Definition of a Robust Modular SOC Test Architecture; Resurrection of the Single TAM Daisy-Chain," in *Proceedings IEEE International Test Conference (ITC)*, Austin, TX, USA, Nov. 2005.

[31] I. C. Society, *IEEE Std 1450.6™-2005, IEEE Standard Test Interface Language (STIL) for Digital Test Vector Data – Core Test Language (CTL)*. New York, NY, USA: IEEE, 2005.

[32] R. Kapur, *CTL for Test Information of Digital ICs*. Boston, MA, USA: Kluwer Academic Publishers, 2003.

[33] J. V. Mottaqiallah Taouil, Said Hamdioui and E. J. Marinissen, "On Maximizing the compound Yield for 3D Wafer-to-Wafer Stacked ICs," in *Proceedings IEEE International Test Conference (ITC)*, 2010.