# 3D ICs Testing: Automated TAM Architecture

A Thesis

submitted to the designated
by the General Assembly
of the Department of Computer Science and Engineering
Examination Committee

by

**Leventos Konstantinos**

in partial fulfilment of the requirements for the degree of

MASTER OF SCIENCE IN DATA AND COMPUTER SYSTEMS ENGINEERING

WITH SPECIALIZATION
IN ADVANCED COMPUTER SYSTEMS

University of Ioannina

February 2021

Examination Committee:

- **Kavousianos Chrysovalantis,** Professor at the Department of Computer Science & Engineering in the School of Engineering of the University of Ioannina, and Supervisor to this Thesis.
- **Tsiatouhas Yiorgos,** Professor at the Department of Computer Science & Engineering in the School of Engineering of the University of Ioannina, and Head of the Department of Computer Science & Engineering in the School of Engineering of the University of Ioannina.
- **Efthymiou Aristides,** Assistant Professor at the Department of Computer Science & Engineering in the School of Engineering of the University of Ioannina.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF GRAPHS

# LIST OF TABLES

# LIST OF ALGORITHMS

# LIST OF TERMS

- 3D IC Three-dimensional Integrated Circuit
- 3DCR STAP Configuration Register
- ATGP Automatic Target Generation Process
- BIST Built-in Self-test
- BUS Cores connected in buses
- DAISY Cores connected in daisy chains
- DfT Design-for-Testing
- DWR Die Wrapper Register
- EN Enable
- Fmax Maximum Frequency
- FPGA Field Programmable Gate Array
- FPP Flexible Parallel Port
- FSM Finite State Machine
- GDS Graphical Design System
- I/O Input / Output
- IEEE Institute of Electrical and Electronics Engineers
- ILV Intra-Layer Via
- IP Intellectual Property
- IWLS International Workshop on Logic Synthesis
- K3 Karmarkar-Karp, Kruskal
- M3D IC Monolithic three-dimensional Integrated Circuit
- MST Minimum Spanning Tree
- PAT Python Aid Tool
- PRI Primary Interface
- PTAP Primary Interface Test Access Port
- REG Register
- RTI Run Test Idle State

- RTL Register Transfer Level
- SEC Secondary Interface
- SEL Select
- SO Serial Output
- SoC System on Chip
- STAP Secondary Interface Test Access Port
- Std Standard
- TAP Test Access Port
- TCK Test Clock
- TCL Tool Command Language
- TDI Test Data Input
- TDO Test Data Output
- TDR Test Data Registers
- TLR Test Logic Reset State
- TMS Test Mode Select
- TRST Test Reset
- TSV Through-Silicon Via
- UNCON Cores connected only functionally
- WBR Wrapper Boundary Register
- WBY Wrapper Bypass Register
- WIR Wrapper Instruction Register
- WPP Wrapper Parallel Port
- WRCK Wrapper Clock
- WRSTN Wrapper Reset, Inverted
- WSI Wrapper Serial Input
- WSO Wrapper Serial Output
- WSP Wrapper Serial Port

# ABSTRACT

The thesis studies the design, the optimization, and the evaluation of a testable 3D integrated circuit (IC), and more specifically its stacked layers. In all, it contains the descriptions of two different ICs, with three different ways of designing them. The first IC is the DAISY circuit, which is a layer of a 3D IC and consists of several cores which are connected sequentially to each other to enable testing. The second IC is the BUS circuit, which is a layer of a 3D IC and consists of several cores which are connected to a central bus through which testing is implemented. These two circuits are generated with a high degree of automation, in part with the Method-k3, and are compared after describing the development and ultimately the testing process of both. Finally, the thesis concludes with the superiority of the first circuit, and fully describes how it can simultaneously implement the IEEE 1149.1-2013 Standard, the IEEE 1500-2005 Standard, and the new IEEE 1838-2019 Standard.

**Keywords:** three-dimensional integrated circuits; 3D ICs, design for test; DfT, through-silicon vias; TSVs, IEEE standards

# Εκτεταμενη Περιληψη

Η διατριβή μελετά ένα τρισδιάστατο κύκλωμα. Στόχος της είναι αυτό το τρισδιάστατο κύκλωμα να είναι πλήρως ελέγξιμο. Αυτό είναι χρήσιμο γιατί η κατασκευή τρισδιάστατων κυκλωμάτων εμπεριέχει την χρήση ξεχωριστών επιπέδων τα οποία μπαίνουν σε μία στοίβα. Η διαδικασία τοποθέτησης τους στην στοίβα πολλαπλασιάζει την πιθανότητα να συμβεί κάποιο σφάλμα, και κομμάτι της στοίβας να υπολειτουργεί. Ο μόνος τρόπος για να βρεθεί αυτό το σφάλμα, ή για να είμαστε σίγουροι ότι η στοίβα λειτουργεί σωστά, είναι ο έλεγχος της. Για αυτόν τον λόγο, σε κάθε επίπεδο της στοίβας προστίθενται επιπλέον κυκλωματικά στοιχεία, που φτιάχνουν τον μηχανισμό ελέγχου του εκάστοτε επιπέδου. Παράλληλα, αυτοί οι ξεχωριστοί μηχανισμοί ελέγχου των επιπέδων της στοίβας είναι χρήσιμο να μπορούν να δημιουργούν ένα ενιαίο σύστημα ελέγχου για ολόκληρη την στοίβα.

Το τρισδιάστατο κύκλωμα περνάει από τις φάσεις της σχεδίασης, της βελτιστοποίησης, αλλά και της αξιολόγησης. Στην φάση της σχεδίασης, χρησιμοποιήθηκαν τεχνικές που είχαν αναπτυχθεί στο τμήμα μας και μετεξελίχθηκαν στην Μέθοδο-κ3, μαζί με το γνωστό πρότυπο 1500 της IEEE. Έπειτα, στην φάση της βελτιστοποίησης, οι τεχνικές άλλαξαν με τέτοιο τρόπο ώστε να μπορούν να ακολουθούν το νέο πρότυπο 1838 της IEEE. Αυτός ο συνδυασμός αξιολογήθηκε με επιτυχία χρησιμοποιώντας δύο διαφορετικές μεθόδους διασύνδεσης. Οι μέθοδοι διασύνδεσης συγκρίθηκαν και ως το ολικό κόστος των συνδέσεων τους, αλλά και τον ολικό χρόνο καθυστέρησης τους. Αυτή η σύγκριση συνέβη πάνω σε δύο επίπεδα τις τρισδιάστατης στοίβας, με τα οποία ασχολείται η παρούσα διατριβή.

Συνολικά, η διατριβή εμπεριέχει τις περιγραφές από δύο διαφορετικά ολοκληρωμένα κυκλώματα. Αυτά σχηματίζουν από ένα επίπεδο μίας τρισδιάστατης στοίβας. Πιο συγκεκριμένα, κάθε ολοκληρωμένο κύκλωμα έχει τρεις διαφορετικούς τρόπους σχεδίασης του, αφού η προσθήκη των μηχανισμών ελέγχου γίνεται τμηματικά. Αυτό συμβαίνει γιατί το πρότυπο 1500 αλλάζει τους πυρήνες των ολοκληρωμένων κυκλωμάτων, ενώ το πρότυπο 1838 αλλάζει τα επίπεδα της τρισ-

διάστατης στοίβας. Δηλαδή, τα δύο πρότυπα έχουν ως στόχο διαφορετικά σημεία του συνολικού ολοκληρωμένου κυκλώματος.

Το πρώτο ολοκληρωμένο κύκλωμα είναι το DAISY, του οποίου οι πυρήνες συνδέονται σε αλληλουχία μεταξύ τους για τον έλεγχο τους. Εσωτερικά, η σύνδεση αυτή γίνεται με δύο αλυσίδες διασύνδεσης, με τέτοιον τρόπο έτσι ώστε κάθε αλυσίδα του μηχανισμού ελέγχου να έχει τους μισούς πυρήνες του επιπέδου. Το δεύτερο ολοκληρωμένο κύκλωμα είναι το BUS, του οποίου οι πυρήνες συνδέονται σε ένα κεντρικό κανάλι που εκτελεί τον έλεγχο τους. Εσωτερικά, η σύνδεση αυτή γίνεται με δύο τέτοια κανάλια, με τέτοιον τρόπο έτσι ώστε το πρώτο να δίνει τα σήματα ελέγχου σε όλους τους πυρήνες, ενώ το δεύτερο να μπορεί να δεχθεί τα σήματα αυτά από έναν πυρήνα κάθε φορά.

Τα δύο αυτά κυκλώματα δημιουργούνται με υψηλό βαθμό αυτοματισμού, εν μέρει με την Μέθοδο-κ3. Έπειτα συγκρίνονται μεταξύ τους μέσα από τις τρεις φάσεις της παράλληλης βελτιστοποίησης τους. Πιο συγκεκριμένα, έχει χρησιμοποιηθεί και ένα τρίτο ολοκληρωμένο κύκλωμα, το UNCON, του οποίου οι πυρήνες συνδέονται μόνο με τις διασυνδέσεις της κανονικής λειτουργίας τους. Δηλαδή, οι πυρήνες του κυκλώματος αυτού δεν έχουν τον επιπλέον μηχανισμό ελέγχου. Με αυτόν τον τρόπο η πολυπλοκότητα αυτών των συνδέσεων κανονικής λειτουργίας μπορεί να αφαιρεθεί από τα άλλα δύο ολοκληρωμένα κυκλώματα.

Η σύγκριση αυτή συμβαίνει αφού περιγραφεί η ανάπτυξη αλλά και τελικά η διαδικασία ελέγχου και των δύο ολοκληρωμένων κυκλωμάτων. Μέσω αυτού η διατριβή καταλήγει στην υπεροχή του πρώτου κυκλώματος, του DAISY, υπέρ του κυκλώματος BUS. Πιο συγκεκριμένα, ο μηχανισμός ελέγχου του κυκλώματος DAISY έχει κατά μέσο όρο 62% μεγαλύτερη συχνότητα λειτουργίας από το κύκλωμα BUS. Είναι ένας μηχανισμός ελέγχου που καταφέρνει να συνδυάσει τρία πρότυπα της IEEE, με ένα ολοκληρωμένο κύκλωμα που περιγράφει μία τρισδιάστατη στοίβα που υλοποιεί ταυτόχρονα το πρότυπο IEEE 1149.1-2013, το πρότυπο IEEE 1500-2005, και το νέο πρότυπο IEEE 1838-2019.

# CHAPTER 1.

# INTRODUCTION



## 1.1 General Introduction

The emergence of 3D stacking technology offers high functionality at a reduced die footprint by enabling the integration of multiple silicon dies on a vertical stack. Separately manufactured dies are integrated onto the same package, and Through-Silicon Vias (TSVs) are used to connect the dies to each other [1]. 3D stacking removes the scalability barriers of nanometre technologies by offering reduced wire-length, reduced interconnect delays, lower power consumption, higher interconnect bandwidth and true heterogeneous integration [2, 3]. 3D stacked memory chips are on the verge of mainstream adoption [4]. Moreover, the semiconductor industry is expected to further exploit the benefits of 3D integration in a variety of products. Namely, 3D Network-on-Chips [5], 3D Memory-on-Processors [6], and 3D FPGAs [7].

TSVs constitute a key technological advantage for die connectivity but come at a cost; the significant area of silicon that gets occupied by them. The bonding process of the dies for the stack requires alignment of the dies with a precision of 0.5 μm, thus imposing strict limits on the minimum allowed TSV diameter [8, 9].

Moreover, surrounding every TSV there is a Keep-out-Zone of a minimum size equal to 3μm for ICs fabricated at 20nm, which forms a microcrack shield between the TSV and the active logic of the die [10]. As a result, the area occupied by every TSV is in the order of a few micrometres, which is equal to the area of several logic gates in nanometre technologies. Inevitably, the number of TSVs in a 3D stack is strictly limited by design constraints. Such constraints further limit the number of TSVs used for other purposes. For example, thermal-TSVs that relieve the thermal stress generated inside the dies, and test-TSVs used for testing purposes.

Besides their large area overhead, TSVs also suffer from several manufacturing defects. Specifically, like voids and cracks, incomplete fillings, pinholes on the insulator boundary, missing of landing pads, improper connections between pads and TSVs, and electromigration. All of these defects adversely affect the chip yield and further increase the manufacturing cost [11]. Even a single defective TSV in a stack leads to the disposal of the whole stack, hence wasting the good dies of the rest of the stack. In addition, several concerns have been raised about defects that may appear in the bottom layer when additional layers are processed on top of them [12]. Moreover, non-bottom layers are susceptible to process variations and electrostatic coupling, while the vias themselves are prone to shorts, opens, and delay defects. Therefore, effective defect screening and quality assurance are not only necessary, but a prerequisite for 3D ICs, especially for 3D processors. Even more important is the need for 3D-IC-oriented Design-for-Testability (DfT) solutions to enable defect isolation and yield enhancement. A powerful tool in this direction is the IEEE 1838-2019 Standard [13, 14], which mandates the insertion of a die wrapper register for TSV-based 3D ICs which provides controllability and observability even in the most deeply buried nodes of the stack.

Besides defect screening, TSV fault tolerance mechanisms are also of paramount importance in 3D stacks, because they constitute effective means to counterbalance some of the yield loss of 3D ICs [15]. TSV repair methods employ redundant TSVs to replace the defective ones, provided of course that effective defect screening techniques are available to pinpoint those TSVs. Since the number of

TSVs is strictly limited by design constraints, the trade-off between the number of redundant TSVs and their yield improvement must be carefully evaluated. At the same time, effective mechanisms must be devised to maximize yield recovery under limitations of TSV quantities.

An emerging technology that overcomes all limitations of die stacking is Monolithic 3D integration (M3D) [16]. In M3D, transistors are processed layer-by-layer on the same wafer in a sequential manner. This sequential integration of transistor layers enables high-density vertical interconnects, known as Inter-Layer Vias (ILVs), with size and pitch typically one or even two orders of magnitude smaller than those of TSVs. M3D integration can result in significantly reduced area and higher performance, which explains the growing interest towards adopting this technology.

Recently, most activities around this subject address the detection of performance variations due to high density integration, defect analysis and modelling, as well as defect isolation along with yield enhancement [17]. This includes the quantification of the electrostatic coupling impact and wafer bonding defects on the threshold voltage of the top layer transistors, all of which feed into path delay faults. It is evident from this work that effective delay test patterns are highly sought after, while a possible DfT solution could be an M3D oriented built-in-self-test approach. Furthermore, the IEEE 1838-2019 Standard, which is originally for 3D ICs, can potentially be extended for M3D ICs, through an M3D specific boundary register to enable modular testing by supporting inward facing and outward facing test modes. However, a major challenge in this case is the significant area overhead of the register at the boundary of every layer. That is because the number of ILVs in M3D ICs is expected to be an order of magnitude higher compared to the number of TSVs in stack-based 3D ICs [18]. Although the extension of IEEE 1838 Std. to M3D enables reuse of methods developed for TSV based 3D ICs, new test solutions are needed due to the significant differences between M3D and TSV based 3D in terms of design, fabrication, failure modes, and test constraints. Moreover, a die can be tested before bonding to reassure that a known-good die is used

in the 3D stack, but ILVs are absent in the uppermost layer during partial assembly testing. Due to these differences and the difficulty of extending IEEE 1838-2019 Standard to M3D ICs, there have been test solutions proposed that are based on dedicated test layers inserted between functional layers [19].

It is evident from the state-of-the-art in 3D technology that focus has already shifted to the DfT infrastructure these stacked layers ought to have for achieving a high-quality test. There is a lot of architectural knowledge from conventional 2D DfT structures, such as internal scan chains, test data compression circuitry, IEEE 1500 Std. wrappers around embedded cores, and built-in-self-test engines [20, 21], which will be re-engineered to adapt to the 3D technology. At the same time, novel 3D DfT structures ought to be engineered. The kind to provide modular test access from (and to) the external stack I/Os to (and from) the various dies and inter-die interconnect levels. Thus, being able to transport test stimuli and responses up and down through other dies on the way. Finally, novel mechanisms ought to be proposed for tolerating faulty TSVs, aiming at strengthening the faulty TSVs rather than replacing them. A major challenge in this researching activity is to ensure that all the 3D DfT architectures and the fault-tolerance mechanisms developed will interoperate together. Hence, there is a need for a per-die 3D DfT standard, such that if compliant dies are brought together in a die stack, a basic minimum of test access features are guaranteed to work across the stack. IEEE 1838-2019 has already become such a standard for stack-based 3D ICs, and is the focal point of this study, along with its merging to the well-known IEEE 1149.1 and IEEE 1500 standards.

In conclusion, it is evident that the future of circuit integration is three-dimensional and is quickly evolving further into a monolithic architecture. This research aims to provide a comprehensive solution when it comes to DfT for 3D ICs. Unless such a method is found, we risk shipping either untested, or exorbitantly expensive ICs, not only missing the chance for a "More than Moore" future, bringing the gains of each subsequent generation of ICs to a complete standstill.

## 1.2 Thesis Structure

The thesis is structured into 8 chapters, and then once more into sections. At the end of the thesis, the appendix follows suit, with 3 segments in total.

1. Chapter 1. Introduction
    - 1.1 General Introduction, where a few key terms of 3D ICs are introduced.
    - 1.2 Thesis Structure, where the sections are briefly described.
    - 1.3 Thesis Contributions, where the achievements of this thesis are presented.
2. Chapter 2. Background
    - 2.1 Basic Understanding, where a brief summary of ICs is presented.
    - 2.2 Testing of 2.5D ICs, where 2.5D ICs are described with more detail.
    - 2.3 Testing of 3D ICs, where 3D ICs are described with more detail.
    - 2.4 Testing of M3D ICs, where M3D ICs are described with more detail.
3. Chapter 3. Standards
    - 3.1 IEEE 1149.1-2013 Standard, where the 1149.1 Standard is described briefly, as it is considered well known.
    - 3.2 IEEE 1500-2005 Standard, where the 1500 Standard is described, as it is considered generally known.
    - 3.3 IEEE 1838-2019 Standard, where the new 1838 Standard is described extensively, as it is considered not known.
4. Chapter 4. Architecture
    - 4.1 Design Flow, where a theoretical die is presented which combines the standards presented in Chapter 3.
    - 4.2 TAM Architecture, where prior contributions that helped this thesis are presented.
5. Chapter 5. TAM-Design Automation
    - 5.1 The Method-k3, where a method used in the design is presented.

- 5.2 Categories of PATs, where all Python Aid Tools created for this thesis are presented.

6. Chapter 6. Detailed Design
   - 6.1 Floorplan, where a theoretical floorplan of the 3D stack is presented.
   - 6.2 Die Modules, where an in-depth description of the parts of the testing mechanism within each die is presented.
   - 6.3 Core Modules, where an in-depth description of the parts of the testing mechanism within each core is presented.
7. Chapter 7. Analysis
   - 7.1 Experimental Results, where the total routing of the BUS and DAISY circuits is presented.
   - 7.2 Timing Analysis, where the histograms of the BUS and DAISY circuits are presented and compared.
   - 7.3 Delay Paths, where the paths which cause delays for the BUS and DAISY circuits are presented and compared.
8. Chapter 8. Future Work, where possible continuation of the work of this thesis is presented.
9. Biblioraphy
10. Appendix
    - A. Design Process, where an in-depth record of the process of the Design Flow is kept.
    - B. Instructions – Operations, where the bitwise commands of the die and core registers are kept.
    - C. Large Floorplan TSVs – FPP "Towards", where a detail of the FPP within the design is described.

## 1.3 Thesis Contributions

This thesis proposes an implementation of the new IEEE 1838 Standard for a test-access-mechanism (TAM) architecture. The "$K^3$ TAM Optimization for Testing 3D-SoCs using Non-Regular Time-Division-Multiplexing" paper [22] is considered the precursor to this thesis. It proposed a 3D TAM architecture which was optimized by means of the $K^3$ design-automation process that combines the Kruskal algorithm with the Complete Karmarkar-Karp heuristic.

However, the $K^3$ TAM Optimization does not optimize the routing of the daisy chains as it does not consider the physical placement of the cores on the floorplan. Truly, the proposed architecture considerably reduces the intra-die connections as by the paper's experimental results, but this thesis asks questions about the inter-die ones.

- Should all cores in a die be connected via the daisy chain technique, and is their additional routing acceptable?
- Is it truly better than to simply have all cores connected onto a bus, which is a more classical approach?
- Finally, is it feasible to implement widely accepted standards with such routing, and even extend them in the direction of the new IEEE 1838 Standard?

In total, this thesis had 10 main achievements. They range from simply using existing standards, to automating parts of the combining of the standards. And from comparing simple routing lengths, to improving the way they were routed. In detail, these achievements were:

1. We used the following IWLS benchmark cores to design and simulate multiple floorplans: *des3_perf*, *eth_top*, *vga_enh_top*, and *wb_conmax*.
2. We wrapped these IWLS benchmark cores in IEEE 1500 Std. wrappers: *wrapper_des*, *wrapper_eth*, *wrapper_vga*, and *wrapper_con*.

3. We compared the total route length of floorplans in which cores were daisy chained or connected onto a bus. In particular, we designed three different floorplans.
    - The UNCON floorplan, where the cores are connected functionally only.
    - The BUS floorplan, where the cores are connected to a bus channel.
    - The DAISY floorplan, where the cores are connected using daisy chains.
4. We refined the "$K^3$" algorithm into the Method-k3, which now has two passes, and explored its limitations.
5. We utilized the Karmarkar-Karp Heuristic to split the cores into two groups, accounting for their shift path, and the Kruskal Algorithm to connect the cores of each of the two groups with the minimum amount of routing.
6. We created a total of 21 Python Aid Tools (PATs) in 7 broad categories, which are further analysed on Chapter 5.
    - Tools for Top-Level Design.
    - Tools for Bounding Boxes for the cores of the designs.
    - Tools for the Cell Locations of each core.
    - Tools for creating Top-Level Designs with specific cell locations.
    - Tools which execute the Method-k3.
    - Tools for the making of rudimentary TSV ports.
    - Tool for wrapping the benchmark cores into similar *façade core*s.
7. We implemented a combination of IEEE 1838 Std. and IEEE 1500 Std. in a final design, using two different floorplans, DAISY and BUS. We analysed them both and found that the DAISY one was 62% faster.
8. We studied their respective slack diagrams and found that their longest paths begin from Update cells and end in Capture cells.

9. We used the FPP in a fully configurable way. Specifically, we used the "TOWARDS" value in order to control if the FPP connects from Side to Side, from Pri to Side, from Sec to Side, or from Side to all three.
10. We fully synthesized and showcased the final design.
    - Showing all IEEE 1838 Std. parts on the die.
    - Showing all IEEE 1500 Std. parts on one of the cores.

# CHAPTER 2.

# BACKGROUND



In order to maximize the benefit of Circuit Integration, 3D ICs and their microscale Through-Silicon Vias were introduced as a new inter-die connection. Even though they suffer from both area and electrical coupling overhead, they are a key technological advancement for die connectivity. Currently, Through-Silicon Vias are not fully supported by commercial design software, especially when it comes to their effective testing. Testing, however, is of paramount importance when it comes to Through-Silicon Vias, as their high integration density, and their manufacturing process, makes them especially vulnerable to various defects. Therefore, effective defect screening and quality assurance are not only necessary, but a prerequisite for 3D ICs.

## 2.1 Basic Understanding

Before any further talk on 3D testing can take place, a basic understanding of the circuits themselves is required, especially when it comes to the way they are being used today.

1. 2D ICs.
    - These circuits constitute what is the mainstream technology.
    - They are two dimensional and consist of a layer on which transistors are etched, followed by multiple metal layers which connect them to achieve the required logic.
2. 2.5D ICs.
    - These circuits have been used as a precursor to full on 3D ICs. They are considered a safer alternative, which does not require significant changes in IC fabrication process.
    - They are 2D ICs placed on an interposer, a special metal layer which connects the normal I/Os of the 2D ICs together to achieve further integration.
    - They are the extrapolation of multiple cores being on the same package, as they are multiple processors acting as one SoC.
3. 3D ICs.
    - These circuits constitute the topic of this research, and they have been used in few commercially available processors so far.
    - The basic idea behind 3D ICs is a stack of dies. In a sense, 3D ICs are comprised by multiple 2D ICs placed on top of each other, connected vertically with TSVs.
    - They achieve even further integration as they use the third dimension in space. For example, two modules of a 2D IC placed at its edges require more routing than if those same modules were aligned vertically in a 3D stack.
4. M3D ICs.
    - Monolithic 3D processors are hailed as the future of ICs and are currently in a strictly research stage.
    - They are built completely differently from 2D ICs, as they consist of multiple transistor layers. That is to say that they can have a transis-

tor layer, followed by some metal layers, which is then followed by additional transistor and metal layers.

- Theoretically, they would achieve the maximum amount of integration in 3D space, as their entire volume consists of logic and its connections.

## 2.2 Testing of 2.5D ICs

With volume production and commercial exploitation of 3D ICs not being feasible before pressing concerns about heat dissipation and test cost are adequately addressed, interposer-based 2.5D ICs might be the only way currently for large-scale development. That is because they use a well-known technology, the interposer. An interposer is a passive device that allows dies to be mounted on it using micro-bumps.

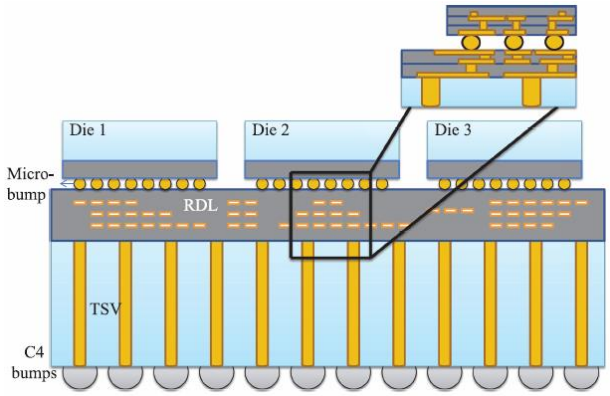



*Figure 2.2.1 Interposer-based 2.5D IC.*

Inside the interposer, there are two types of interconnects: the redistribution layer and the TSVs. The RDL is a structure of multiple metal layers that provides horizontal die-to-die interconnects (fig. 2.2.1). While the TSVs, which are connected to C4 bumps, are used for vertical die-to-package interconnects. However, since the interposer is a passive device, it cannot support any active logic. Therefore, possible BIST architectures and their associated BIST controllers must be integrated within each die, and no sharing of the BIST hardware is possible. Therefore, the area overhead of BIST for a 2.5D IC can potentially be several times larger than that for a corresponding 2D IC.

Such increased area overhead is undesirable since it leads to an increase in die area. Therefore, an increase in test-application time is inevitable, even with faster in-system and at-speed testing in BIST. Moreover, the faster test clock in the BIST architecture results in a higher test power consumption, which is also a significant challenge in 2.5D IC testing. To overcome the above limitations, it was deemed necessary to develop a new methodology for die testing in 2.5D ICs with reduced

test cost in terms of test-application time and hardware overhead, but with high fault coverage and less power consumption. To this end, an efficient test architecture that can enable the application of the same test patterns to multiple dies simultaneously with negligible area overhead was proposed [23], by using ATGP patterns.

Currently, high-density I/O ports are available for the dies in a 2.5D IC, and many die-to-die connections are available inside the interposer. Therefore, 2.5D ICs can provide enhanced system performance, reduced power consumption, and support for heterogeneous integration. High integration complexity gives rise to the likelihood of defects during the fabrication of 2.5D ICs. Since the structure of 2.5D ICs is different from traditional 2D ICs, new test challenges emerged. These challenges are the pre-bond interposer testing, the post-pond interposer testing, and the active die testing. Thus, solutions are required [24] for these challenges if we hope to use 2.5ICs as a precursor to full on 3D ICs.

And yet, with the high density of I\O ports and interconnects, testing a 2.5D IC or a 3D IC is far more challenging than testing a traditional 2D IC. For instance, there are 186k micro-bumps but only 25k C4 bumps in AMD Fiji [25]. While logic dies are typically equipped with full scan and boundary scan, the high density of interconnects typically leads to large test-data volume. If this large volume of test data has to be applied through a one-bit serial boundary-scan chain, the test will take a very long time to execute and hence become prohibitively expensive. Additionally, the high power consumption of 3D ICs, during testing various stagger values have to be used in order for the various blocks of the stack to avoid shifting concurrently, thus drawing too much power at the same time. To that end, a test-scheduling and optimization technique is also used for identifying groups of dies for multicast in order to reduce test-application time while satisfying constraints on the power budget and fault coverage. Simulation results have demonstrated that compared to previous built-in self-test techniques for 2.5D IC testing, there are techniques which reduce test-application time for benchmark designs with negligible area overhead and higher fault coverage [26].

Specifically for the testing of the interposer that makes 2.5D ICs possible, its structure has to be taken into account too. The vertical interconnects of the interposer are composed of microbumps, TSVs, and C4 bumps that connect the dies to the package substrate. The horizontal interconnects are composed of microbumps and a structure of multiple metal layers that connect various dies. The interconnects in the interposer are fabricated using the same processes as the interconnects in the silicon dies. As a result, an interposer can provide more than 10 000 die-to-die interconnects and approximately 1200 I/O pins [27]. Testing the interposer requires the targeting of both types of interconnects: horizontal and vertical. If both sides of the interposer can be probed at the same time, pre-bond interposer testing can be easily accomplished.

Conversely, the semiconductor industry continues to be faced with market demand for integrated circuits with increasing functionality and high performance. Its goals are to reduce chip footprint, to integrate more transistors in an IC, and to achieve higher performance. Thus, a specific kind of interconnect structures first received attention, ones that most easily led to multitiered ICs. Face-to-face bonded ICs, with TSVs embedded in the substrate of a silicon wafer, connecting the metal layers on the front side with another die or package on the same side [28]. This of course brought new opportunities for the design of dies and the interconnection between them, but also introduced new challenges for the testing of ICs.

However, double-sided probing of the interposer is not feasible today due to limitations related to wafer handling and probe-card design. In addition, it is difficult to probe the micro-bumps on the top side of the interposer due to their high density. Interconnect testing requires connecting the interconnects in a loop so that a logic value can be applied at one end and the propagated value can be observed at the other end. However, interconnects are separated and independent from each other at the pre-bond stage. Therefore, new and innovative solutions were needed for pre-bond testing, which culminated in a test architecture which uses e-fuses that can be programmed through voltage pulses outside the range of normal circuit operation.

In fact, increasing wire delay and higher interconnect power consumption are major concerns for nanoscale CMOS ICs. At first glance, 3D ICs based on Through Silicon Vias appear to be a promising solution to overcome this bottleneck in CMOS scaling. However, interposed based 2.5D ICs are being advocated [29] as a feasible precursor to full 3D ICs. Still, all the dies either in 2.5D or 3D ICs must be adequately tested for product qualification. Moreover, the introduction of TSVs for both signal routing across multiple dies as well as Power Delivery Network, imposes challenges in terms of manufacturing yield and resiliency issues which should be addressed in both design and test flows.

## 2.3 Testing of 3D ICs

Historically, the semiconductor industry has been able to meet the demand for high-performance integrated circuits with added functionality by relentlessly scaling device sizes. It has become clear, though, that it is increasingly difficult to sustain device scaling in an economically viable manner. Escalating costs which are mainly due to the challenges associated with the lithography of small features, interconnect scaling, and reducing and mitigating process variations. And that is exactly where 3D stacking comes into play, and the whole reason for trying to solve its issues. Additionally, 3D technologies enable the integration of heterogeneous fabrication processes, thus paving the way for complex systems such as memory-on-logic [30].

Although Through-Silicon Vias could be tested together with logic and memory, a Design-for-Test method is still required, especially for defect isolation and yield learning. Resistive Random Access Memory, which enables high bandwidth logic-memory integration has emerged as an attractive candidate for on-chip non-volatile memory technology. There have been significant efforts in developing these technologies as well as macros. However, same as before, there is a significant lack of electronic design automation methods and tools for their automatic macro generation.

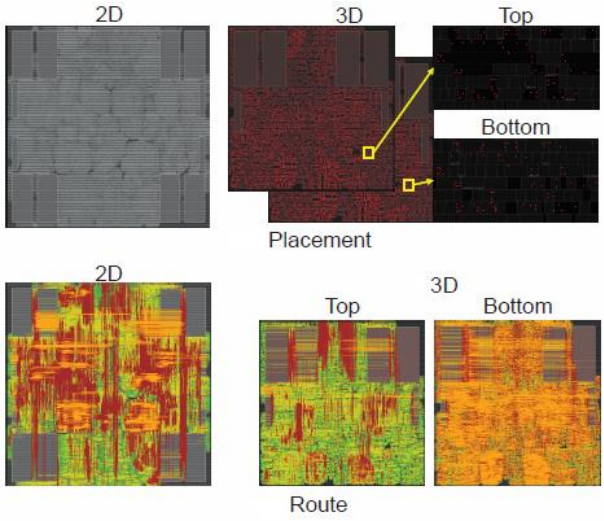



*Figure 2.3.1 GDS layout of 2D and 3D designs.*

There are papers [31] in which RTL-to-GDS design flows are discussed (fig. 2.3.1), along with DfT methods that detect faults, and a ReRAM module generator for design exploration.

Moreover, 3D integration is proving to be a promising way to achieve high-performance ICs with more functionality and a reduced die footprint. The basic idea lies on die or wafer stacking as it does not require substantial changes to the existing fabrication process. Thus, separately manufactured dies or wafers are integrated onto the same package, and Through-Silicon Vias are used to connect the dies to each other. Considerable research efforts have been directed towards the development of TSV-based 3D stacking technology. However, the keep-out-zone required for Through-Silicon Vias and limitation with the precision on die alignment impose limits on achievable device integration density.

Specifically, a minimum keep-out-zone of 3μm is required for ICs fabricated at 20nm, while the die alignment precision is currently limited to 0.5μm. For that reason, another emerging technology is Monolithic 3D integration, in which transistors are processed layer-by-layer on the same wafer. Sequential integration of transistor layers enables high-density vertical interconnects, known as Inter-Layer Vias. Their size and pitch are typically one or even two orders of magnitude smaller than those of a TSV. Therefore, M3D integration can result in significantly reduced area and higher performance, which explains the growing interest towards adopting this technology.

Nevertheless, various issues with this technology exist. Firstly, having to invent a low-temperature process to fabricate high-performance top transistor layers without damaging the bottom transistor layers. Additionally, finding design techniques to reduce interconnect length, along with critical path delay, and die area. Research exists [32] for the detection of performance variations due to high-density integration, defect analysis and modelling, and defect isolation along with yield enhancement. This includes the quantification of the electrostatic coupling impact and wafer-bonding defects on the threshold voltage of the top-layer transistors. All of

which feed into path delays, and the effectiveness of delay-test patterns, with a possible solution being built-in-self-testing.

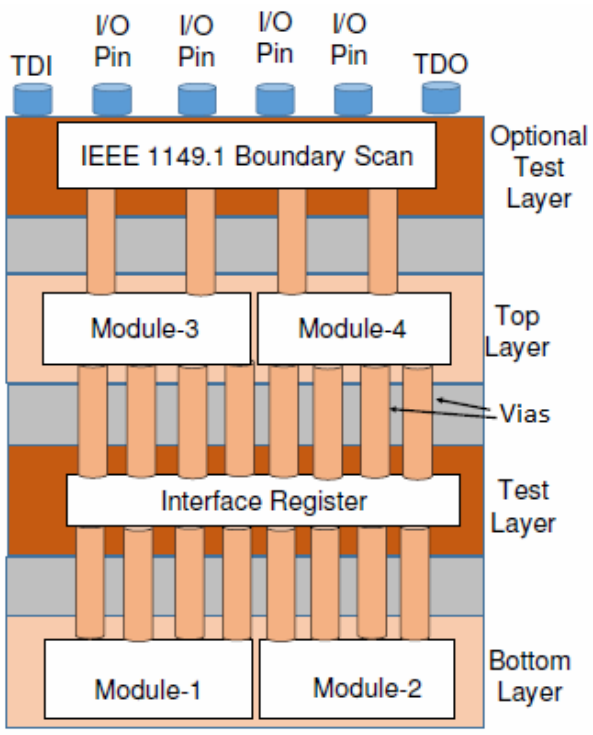



*Figure 2.3.2 Layered DfT solution.*

With all those issues, it is no wonder that research has to focus on finding test solutions for 3D ICs. The latest one seems to be based on dedicated test layers [33] which are inserted between the functional layers of the 3D stack (fig. 2.3.2). Meaning that one layer has the functional components, while the other layer has the test scan chains, and so on and so forth in an alternating fashion. More specifically, the test layer includes an interface register controlling signals from a testing module to one of the test scan chains, and an instruction register connected to the interface register. The instruction register processes testing instructions from the testing module, which is connected with Inter-Layer Vias to the functional components, and the test module throughout the test layer.

Granted that, 3D ICs promise to overcome interconnect bottlenecks in CMOS scaling while offering true heterogeneous SoC integration. The obstacle then for their widespread industry adoption truly are their low manufacturing yield. In general, the yield of 3D ICs can be reduced due to the defects in stacked dies or defects that occur during the assembly process. In the former case, it is critical to

conduct pre-bond testing to prevent the stacking of defective dies. For the latter case, the addition of spare TSVs to repair defective functional TSVs is an effective method for increasing yield and ensuring reliability. And yet, while several spare TSV allocation strategies have been proposed in literature, these methods only consider uniform TSV placement [34]. While such a layout offers advantages like lower heat dissipation and stronger package bonding, non-uniform TSV placement allows more design flexibility and leads to shorter wirelength.

As a result, non-uniform TSV placement provides two important benefits, namely lower latency, and power reduction. However, due to the added degree of freedom in the locations of functional TSVs associated with non-uniform placement, it is a challenge to enhance the yield for such designs, and advances in spare TSV allocation methods are needed to achieve the above performance benefits.

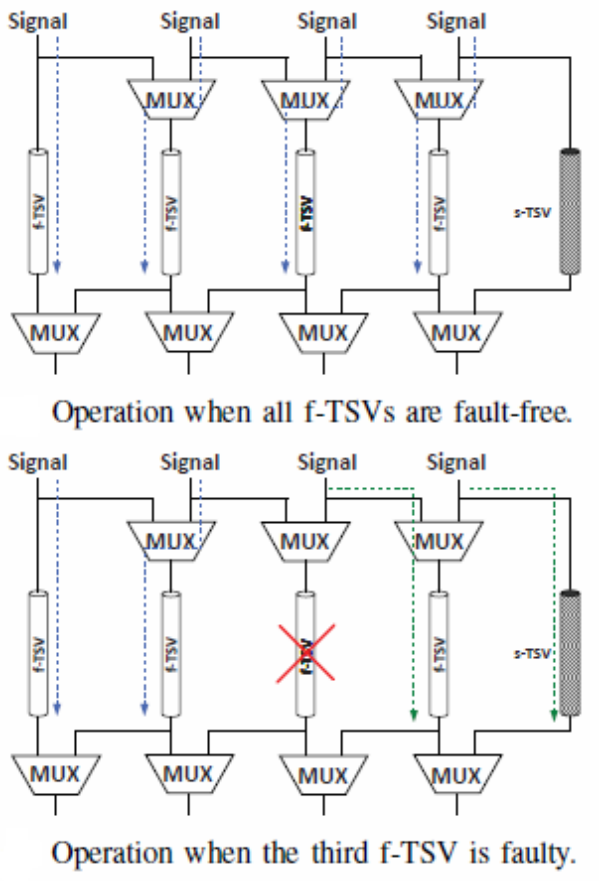



*Figure 2.3.3 Operation of f-TSVs for fault tolerance.*

In order to address this problem, s-TSV allocation techniques (fig. 2.3.3) have recently been proposed for non-uniform TSV placement even though there still

remain major challenges that limit the practicality of these techniques. Those challenges are namely that defects are not actually uniformly distributed, that TSVs cannot actually be placed anywhere on the chip, and that delay overhead doesn't originate only from signal re-routing.

On the other hand, since TSVs support high clock frequencies, and can thus be used as high-speed interfaces between the dies of the stack, very few of them ought to be used for testing purposes. This low number of test TSVs creates a serious bottleneck for the Test Access Mechanism of 3D ICs, which delivers high volume of test-data using a small number of horizontal and vertical interconnects. In addition, the scan-chains of the cores support low shift frequencies because they are not optimized for timing. Therefore, the highest rate at which test-data can be transferred through the TAM is very low. On top of that, the lower thermal conductivities of inter-tier and inter-metal dielectrics used in 3D ICs block the heat generated inside the stacks from reaching the heat sink. As a result, the scan shift frequencies are often further reduced to avoid violating power and thermal limitations of 3D ICs.

Hence the test time for a 3D IC is dominated by the time needed for transporting test data to various layers of the stack, the limited number of TSVs adversely affects the test time of 3D ICs. In order to overcome these limitations, TAM architectures for 3D ICs have been proposed [35] that exploit the high speed offered by TSVs in order to support fast transfer of test data using a small number of TSVs. That is achieved by the means of a 2D Time Division Multiplexing approach, which is applied at the vertical dimension of the 3D stack as well as on the horizontal dimension. An efficient test-scheduling approach must identify the appropriate shift-frequency for every core to maximize the number of tests that can be scheduled in parallel without violating the power and thermal constraints of a 3D IC.

On the other hand, there is always the thought of built-in self-test methods. 3D stacking involves many possible test insertions, due to multiple yield and test cost parameters corresponding to different dies and tests, such as for pre-bond, post-

bond, and partial stack. As an exponentially large number of test flows must be evaluated, analysis methods and tools are needed for test-cost optimization and automated test-flow selection. BIST is a promising solution because it simplifies test application. Especially in 3D ICs, since tests can be applied at many possible test stages or test insertions, there is a need for a distributed BIST framework. Such a framework can enable BIST-based testing at multiple test insertions [36]. Specifically, there are methods to locate defects in a passive interposer before and after stacking. Firstly, a technique for contactless pre-bond TSV testing and a DfT architecture for post-bond die access. Secondly, an optimization approach to select an effective test flow by systematically exploring an exponentially large number of candidate test flows. Lastly, an end-to-end design of a BIST infrastructure.

Furthermore, the very placing of the TSVs may turn known-good chips into faulty ones, which of course feeds back to the low yield of 3D integration. Such concerns have been highlighted especially about defects that may arise in the bottom layer when additional layers are processed. In addition, non-bottom layers are susceptible to process variations and electrostatic coupling, while the vias themselves are prone to shorts, opens, and delay defects. Therefore, there is a need for Design for Test solutions to enable defect isolation and yield enhancement. A strong candidate is the IEEE Std P1838 which mandates the insertion of a die wrapper register for TSV-based 3D ICs that provides controllability and observability.

## 2.4 Testing of M3D ICs

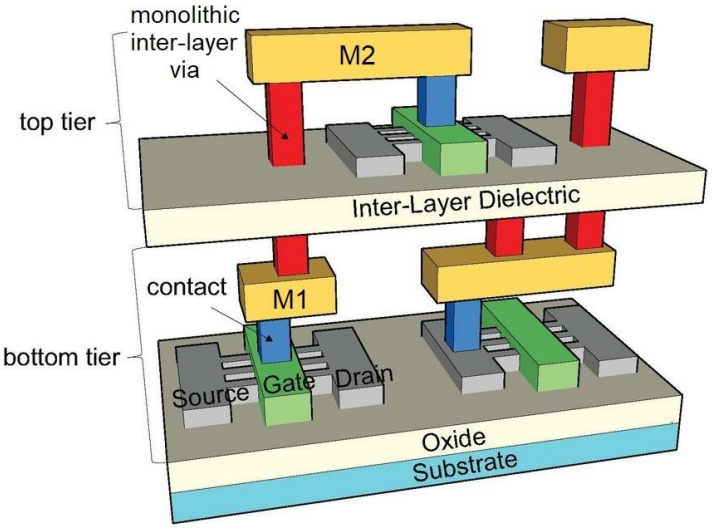



*Figure 2.4.1 An illustration of an M3D IC.*

Interestingly, P1838 can potentially be extended even for Monolithic 3D ICs (fig. 2.4.1), which are the next logical step of 3D Integration. The reason for this turn to Monolithic 3D ICs even without having solved all issues with TSV-based ones is that it is becoming increasingly difficult to sustain device scaling in an economically viable manner. This is due to challenges associated with interconnect scaling, lithography of small features, and process variations. In a M3D, the bottom-layer transistors and their associated interconnects are first processed using a standard high-temperature process. Next, a thin silicon layer is created over the bottom layer. The top-layer transistors are then processed under a strict thermal budget. Finally, ILVs are processed to connect the two layers. These steps are repeated for any additional layers. Sadly, the number of both functional and test layers can be limited by the likelihood of performance degradation due to the high temperature processing steps.

Therefore, it becomes clear that for this sequential integration to work, what is required is a low-temperature process to create a thin silicon film over the bottom layer. Furthermore, what is required is also a process to realize transistors on the top layer without damaging all the underlying interconnects or degrading the transistors on the bottom layer. One of the ways that is reported in literature [37] to succeed in achieving those two requirements is by shielding layers with a protec-

tion that keeps them from being damaged due to the high-temperature steps. Hence, with that issue solved, the fact remains that there must be a way to find the faults hiding in the ILVs. Clearly, the way to do that seems to be the BIST DfT solution of adding a test layer in-between every two functional layers.

Above all, it is the test solution based on dedicated test layers that are inserted between functional layers that seems to be the way forward, even when compared to test solutions inspired by the extension of the P1838 standard. That is because these layers provide controllability and observability to signals at the interfaces of functional layers. Their main features are a low-bandwidth serial interface, a higher-bandwidth parallel interface, dedicated probe pads on all layers except the top one to enable partial-assembly testing, and test structures to enable modular testing. And even though the addition of test layers to the M3D assembly can potentially lower chip yield because of more candidate defect locations, the improvement in test coverage and defect-isolation capability offsets this concern. Moreover, the dedicated test layers can be manufactured using a mature technology [38] and the number of back-end-of-the-line layers can be minimized to reduce the total impact on die yield.

As for Monolithic 3D integration, although it is receiving considerable interest, and while it can theoretically achieve higher device density compared to TSV-based 3D stacking, it is still considered an immature technology. Presently, there is a need to analyse the impact of wafer-bonding defects on path delays in a Monolithic 3D IC [39]. A need to understand the impact of bond defects on the threshold voltage of a top-layer transistor and on the ILVs. This impact of wafer-bonding defects on the threshold voltage of a top-layer transistor is significant, and cannot be ignored, especially for Monolithic 3D ICs integrated at the gate-level. Further on, it is known that the presence of defects at the bond interface can lead to a change in resistance of an ILV, and in some cases, lead to an open in the ILV or a short between two ILVs. These defects can significantly impact the slacks for paths through the top layer in a gate-level-integrated Monolithic 3D IC.

# CHAPTER 3.

# STANDARDS



Three IEEE standards have been used in this thesis, which are briefly presented in this chapter. Amongst them, one is considered new and relatively unknown, and thus it will be presented with more detail. Important from this brief presentation are the specific module and port names, as they will be used in the following chapters.

## 3.1 IEEE 1149.1-2013 Standard

The 1149.1 Std. [40] defines test logic that can be included in an integrated circuit to provide standardized test solutions. Firstly, it is used to test the interconnections between integrated circuits once they have been assembled onto a printed circuit board or other substrate. Secondly, it is used to test the integrated circuit itself. Lastly, it is used to observe or to modify circuit activity during the component's normal operation. The test logic consists of a boundary-scan register and other building blocks and is accessed through a test access port (TAP).

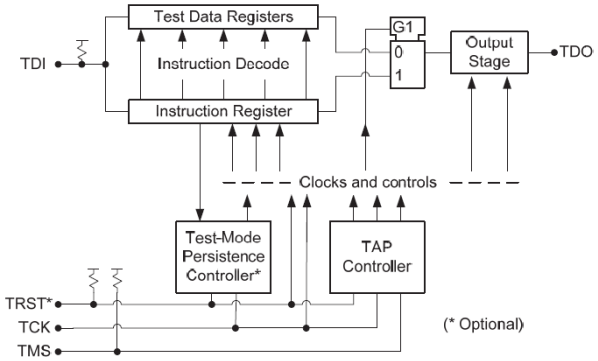



*Figure 3.1.1 On-chip test logic for 1149 Std.*

The TAP consists of five well-known ports, TCK, TMS, TDI, TDO, and TRST (fig. 3.1.1). Of those, TMS feeds into the TAP Controller, an FSM which has 16 states and creates the Update – Capture – Shift signals either for the Data Registers, or the Instruction Register. Especially for the Instruction Register, which is defined as having a width of at least 2 bits, the 1149.1 Std. defines some Instructions that always must be included, namely BYPASS, SAMPLE, PRELOAD, and EXTEST, while also allowing the merging of SAMPLE and PRELOAD into one Instruction.

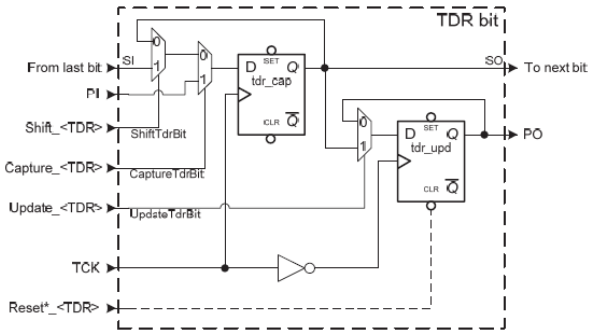



*Figure 3.1.2 Capture-update TDR cell with non-gated clock and optional reset.*

Finally, the Test Data Registers, of which there are at least two: the Bypass and the Boundary-Scan registers. They are composed from TDR cells (fig. 3.1.2), which are defined fully, complete with descriptions in hardware description languages. They are used as the very basis of the testing functionalities of the combined standards. Summarizing, the 1149.1 Std. is used in the final design of this thesis for its widely accepted testing ports, and its FSM, which can be used with no adjustments by the following two standards, as they were based on it as their precursor.

## 3.2 IEEE 1500-2005 Standard

The 1500 Std. [41] defines a scalable architecture for independent, modular test development and test application for embedded design blocks and enables testing of the external logic surrounding these cores. Modular testing is typically a requirement for embedded non-logic blocks, such as memories, and for embedded predesigned non-mergeable intellectual property (IP) cores. In addition, its architecture can also be used to partition large design blocks into smaller blocks of more manageable size and to facilitate test reuse for blocks that are reused from one system-on-chip (SoC) design to the next.

Considered well-known, it has developed a standard design-for-testability method for integrated circuits (ICs) containing embedded non-mergeable cores. Its method is independent of the underlying functionality of the IC or its individual embedded cores. The method creates the necessary requirements for the test of such ICs, while allowing for ease of interoperability of cores that may have originated from different sources. Its aim was to provide a consistent scalable solution to the test reuse challenges specific to the reuse of non-mergeable cores, while preserving the IP aspects that are often associated with these cores. This objective was achieved through provision of a core-centric methodology that enables successful integration of cores into SoCs.

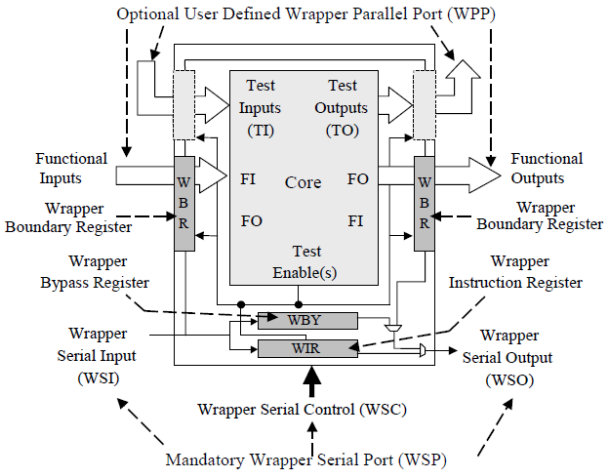



*Figure 3.2.1 Standard 1500 std. wrapper components.*

It is comprised by the Wrapper Serial Port (WSP), the Wrapper Instruction Register (WIR), the Wrapper Bypass Register (WBY), and finally the Wrapper

Boundary Register (WBR). It also allows for a user-defined set of wrapper terminals forming the Wrapper Parallel Port (WPP) which provide parallel access to the wrapper (fig. 3.2.1), which is important for the final standard presented in this chapter.

The WSP is compromised by the well-known ports WSI, WSO, WRCK, WRSTN, and the additional SelectWIR, CaptureWR, ShiftWR, and UpdateWR. When combined with the standard, the last three can be achieved with the additional of a specified "glue logic" module which converts the signals produced by the FSM to the ones required within the core.

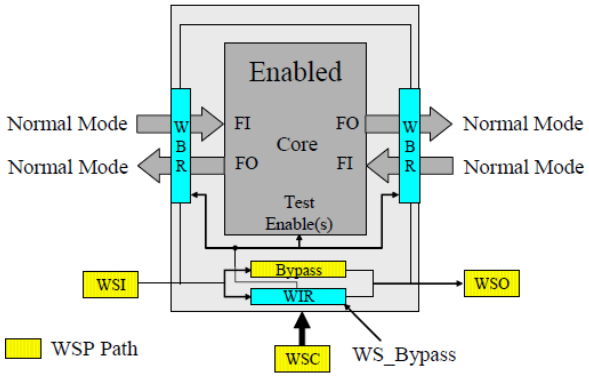



*Figure 3.2.2 An example core executing WS_BYPASS.*

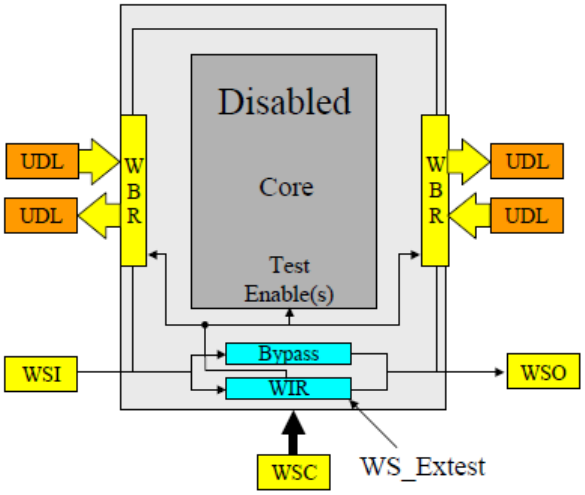



*Figure 3.2.3 An example core executing WS_EXTEST.*

For the WIR, it specifies that it must have a shift path of at least 2 bits, and that it must account for the two mandatory instructions WS_BYPASS (fig. 3.2.2) and WS_EXTEST (fig. 3.2.3). The former instruction is the one which uses the WBY, which allows for only some of the cores to be tested at a time. This will be used fully in the next standard and is the way the two standards can be used at the same time. It is what allows them to be tested serially by this standard, while also having a parallel function as allowed, which will conform to the specifications of the standard that follows.

## 3.3 IEEE 1838-2019 Standard

Advancements in interconnect, assembly, and packaging technology have led to a wide range of multi-die stack architectures. These die stacks need to be tested before they can be shipped with acceptable quality levels to customers. Consequently, three-dimensional design-for-test (3D DfT) structures that provide test access between the external stack I/Os and the various dies and inter-die interconnect are needed. Test access is needed for manufacturing phases that include both partially assembled and complete stacks. These are the issues that the 1838 Std. [42] addresses.

It is die-centric, applying to a die that is intended to be part of a multi-die stack. It defines die-level features that, when compliant dies are brought together in a stack, comprise a stack-level architecture. Initially, it enables transportation of control and data signals for the test of intra-die circuitry. Additionally, it enables inter-die interconnects in both pre-stacking and post-stacking situations. It supports testing for both partial and complete stacks in pre-packaging, post-packaging, and board-level situations. The primary focus of inter-die interconnect technology addressed by this standard is through-silicon vias (TSVs); however, this does not preclude its use with other interconnect technologies such as wire-bonding.

In particular, the 1838 std. standardizes mandatory and optional on-chip hardware components for 3D test access. Its aim is to define standardized and scalable 3D-DfT features based on and working with digital scan-based test access at die-level. Hence, when compliant dies are stacked, a stack-level 3D-DfT test access architecture emerges, an architecture with a minimum functionality and many optional extensions. The standard provides a modular test access architecture, in which dies and interconnect layers between adjacent stacked dies can be tested individually. The focus of the standard is testing the intra-die circuitry as well as the inter-die interconnects in pre-bond, mid-bond, and post-bond cases in pre-packaging, post-packaging, and board-level situations. The standard provides test access via a mandatory one-bit serial input/output test port and multi-bit parallel test ports.

Being die-centric, compliance to the standard pertains to a die (and not to a stack of dies). Standardized die-level design-for-test (DfT) features comprise a stack-level test access architecture. In this way, the standard enables interoperability between die makers and stack maker. The standard does not address stack-level challenges and solutions. The most prominent example of this is that the standard does not address compliance of the stack to IEEE 1149.1 Std. boundary scan for board-level interconnect testing (although the standard certainly does not prohibit application thereof). It also does not mandate specific defect or fault models, specific test generation methods, nor specific die-internal 2D-DfT features. However, the standard leverages existing 2D-DfT wherever applicable and appropriate, including test access ports, such as specified in IEEE 1149.1 Std., and on-chip DfT such as internal scan chains and wrappers of embedded cores, such as specified in IEEE 1500 Std. Similar to IEEE 1149.1 Std. and IEEE 1500 Std. it only defines a DfT architecture: the number, name, type, and function of test I/Os, the On-chip DfT hardware and corresponding description, and the clock-cycle accurate test operation protocol.

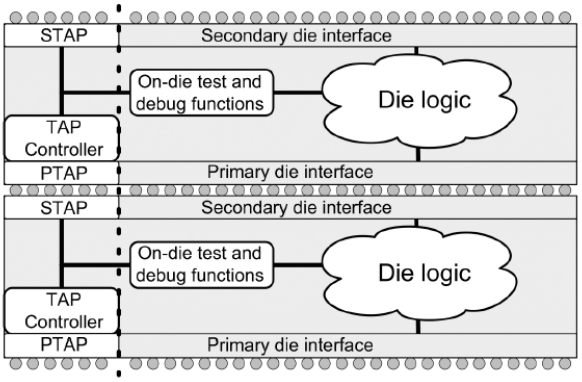



*Figure 3.3.1 Two layers of a 3D stack which implement the IEEE 1838 Standard.*

Its general architecture is based on its two interfaces, the Primary and Secondary one. They are made in such a way so that they can be brought together in a stack (fig. 3.3.1). Part of the interfaces is used for test data, while the rest is assumed to be used for functional data. That means that the standard can accommodate power TSVs, data elevators, and various other stack-based solutions without alterations.

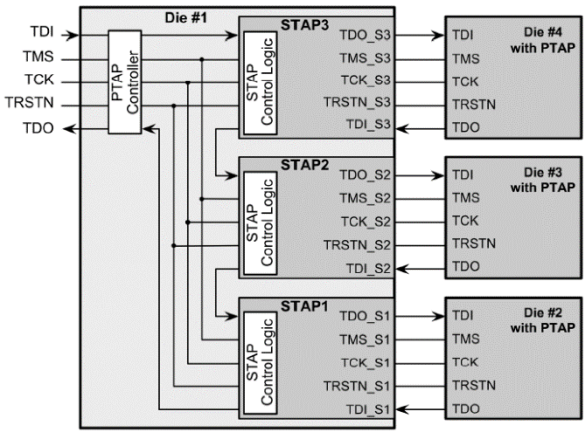



*Figure 3.3.2 A 3D stack with three dies on its second layer.*

The standard allows for various forms of stacking, accounting for layers that are formed by more than one dies. This is achieved by the Primary interface being able to drive any number of Secondary interfaces (fig. 3.3.2), as many as it is required from the specific stack architecture. This is one more reason why the standard can be easily implemented in all kinds of 3D stacks. That is especially true when it comes to its test signals, which are the well-known IEEE 1149.1 Std. ones, which further simplify the correct implementation of the standard.

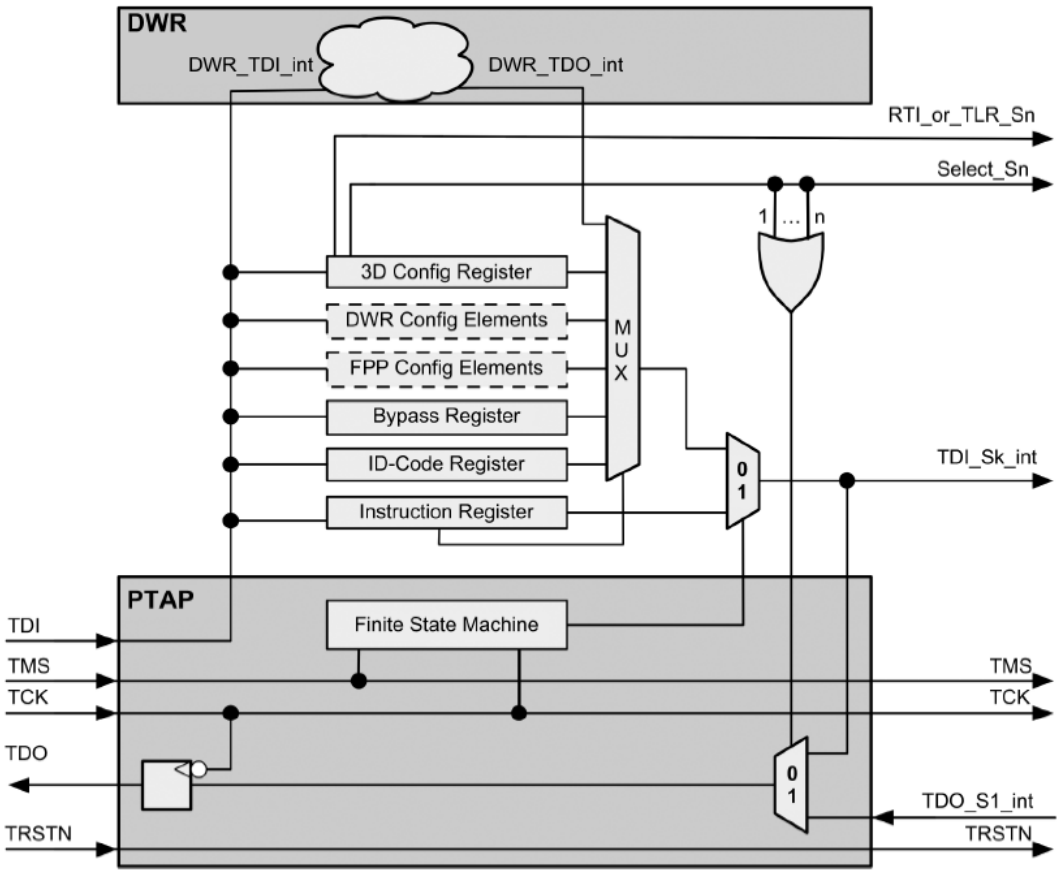



*Figure 3.3.3 Per-die Primary Test Access Port and register with signal connections for 3D extension units and feature Configuration Registers.*

Presented here (fig. 3.3.3) is the general architecture of the Primary Test Access Port (PTAP) and its surrounding registers, all being linked to TDI for input, and having the internal TDI signal as output. Of course, this also serves as the TDO signal, but only if the next Secondary Test Access Port (STAP) is not selected, which means it does not produce its own TDO signal. Other than the Die Wrapper Register (DWR), which will be explained separately, and the FSM within the PTAP, the rest of the architecture consists of registers. These are the heart of the standard, controlling its various functions, under the guidance of the FSM.

For a first understanding of its usage, it is sufficient to understand that the PTAP is only a bit of logic around the IEEE 1149.1 Std. FSM. In turn, it controls the Instruction Register, which can then decide the function of the entire wrapper. Its choices being the DWR, or the various registers, such as the Bypass Register, the 3D Configuration register, or the Identification Code Register. Simply put, the signals TMS and TCK control the FSM, which allows the TDI signal to reach the

Instruction Register. If it has already been set up, the TDI signal moves to one of the other choices, following the command within the Instruction Register. Further on, each piece of the IEEE 1838 Std. will be presented in more detail, for a more in-depth understanding of their functionality.

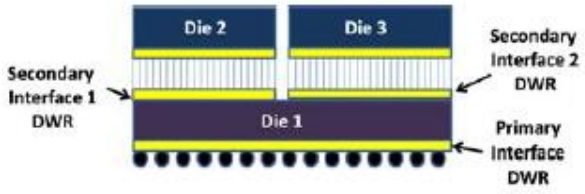



*Figure 3.3.4 Primary and Secondary Interfaces.*

Starting with the serial test access ports, which is one subset of the primary interface, first comes the primary test access port (PTAP). It contains the signals and internal die logic connections that are associated with the Primary Interface. Another subset of the secondary interface is one or more secondary test access ports (STAP). They contain the signals and internal die logic connections that are associated with the secondary interface (fig. 3.3.4). These may also include the flexible parallel port (FPP) which will be described further on.

The PTAP is associated with the surface closest to the board connection or package interface and is represented by five terminals: TCK, TMS, TDI, TDO, TRSTN. These signals drive the PTAP controller, which is an IEEE 1149.1 Std. compatible TAP controller. As for the STAP, each of them has five equivalent terminals: TCK_Sn, TMS_Sn, TDI_Sn, TDO_Sn, TRSTN_Sn. To illustrate, "n" is the number of the STAP starting from 1, with STAP-1 being connected closest in scan path order to the PTAP TDO terminal. As for which STAP is selected, this is where the 3DCR module comes in, else known as the STAP Configuration Register. It works in tandem with the STAP control logic, which is simple and described fully in the standard.

The 3DCR is defined to have three mandatory signals, the Config-Hold signal, the Select_Sn signal, and the RTI_or_TLR_Sn signal. The Config-Hold signal resets to a deasserted state and makes the STAP configuration bits and the STAPs persistent through the Test-Logic-Reset action of the PTAP controller's FSM. The Select_Sn signals reset to a deasserted state and select and activate the individual

numbered STAP_Sn. And the RTI_or_TLR_Sn signals, which are individual parking state definition signals, and reset to a logic 1 state, and define the parking state of the individually distributed TMS_Sn signals associated with the individual numbered STAP_Sn. Of course, the last die of the stack does not require to have a 3DCR, as it does not have any STAPs.

With each PTAP controller and register architecture including an Instruction Register, the instruction Select3DCR is the only way for the 3DCR to be accessed. As always, another mandatory instruction is the Bypass one, which selects the Bypass register. Another required module is the Device Identification register, a relic from the IEEE 1149.1 Std. which was included to the final design in this thesis only for the ability to fully implement the standard. Its instruction is Select IDCODE, while there are also three more recommended ones: Select DWR Extest, Intest, and Transparent.

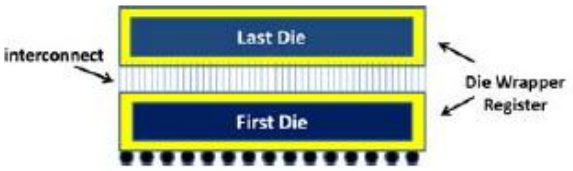



*Figure 3.3.5 The DWR on two dies.*

The Die Wrapper Register (DWR) (fig. 3.3.5) which exists for each die enables controllability and observability of the logic to and from die terminals for both INTEST and EXTEST modes. The standard allows multiple configurations of the DWR, namely that it can reuse one of more segments of an IEEE 1500 Std. WBR as the DWR. The DWR cell operation relies on the Shift, Capture, Update, and Apply events. Of which, only the later one is unknown, and is nothing more than the test clock pulse which connects to all registers. Lastly, the Test-Logic-Reset state can be used to force the DWR logic into a state that enables functional operation of the die.

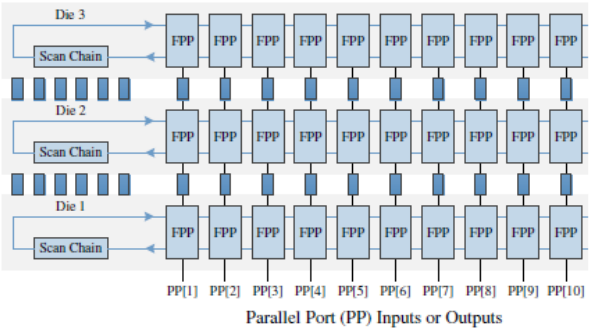



*Figure 3.3.6 The FPPs inside the dies.*

The last piece of the standard, and the cornerstone for using it, is the FPP module, the Flexible Parallel Port (fig. 3.3.6). By definition, the FPP can be used to establish a connection between the primary interface, the secondary interface, and the core of functional logic on the die. The optional FPP is intended for carrying arbitrary test data, clock, and control signals up and down the die stack independent of the die wrapper. The configuration of the FPP can vary depending on the application. The FPP is composed of a set of lanes. Each lane implements a one-bit wide path. Lanes with identical properties and control may be grouped into channels. Lanes can be unidirectional or bi-directional, registered (including pipe-lined pathways between terminals of the lane) or unregistered, and include several connection points between the bottom and top of the die. The collection of connection points is selectable according to the rules shown below but may include terminals from the lane to the core and back, from one lane to another lane, and between the primary and secondary interfaces. Multiplexing functions within the lane can select how these terminals interconnect within the lane. Controls for these multiplexing functions are derived from test data register bits sprinkled throughout the scan-accessible network.

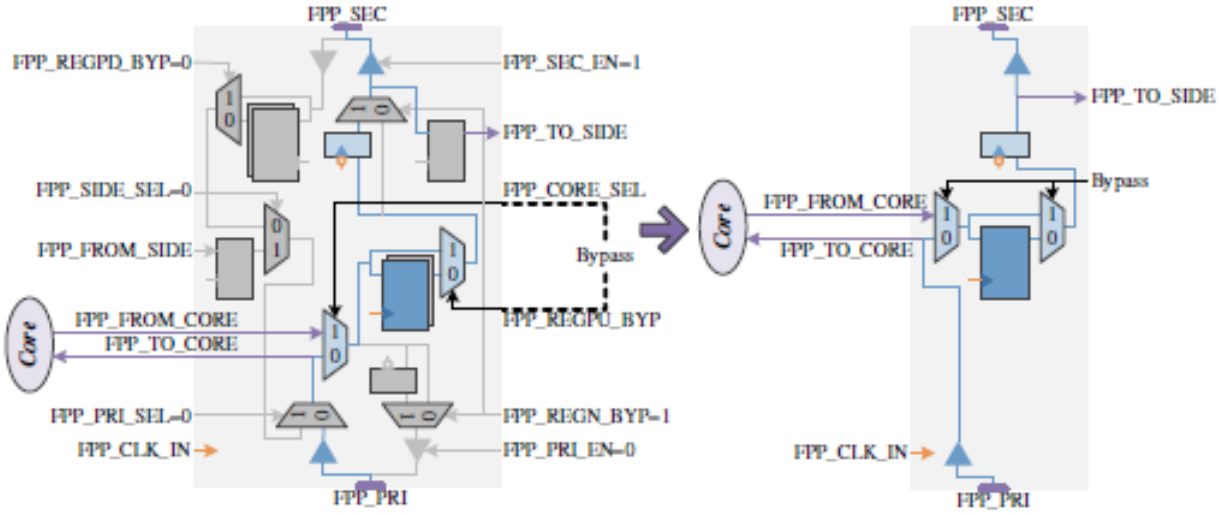



*Figure 3.3.7 Example of the flexibility of the FPP (PRI to Side and SEC).*

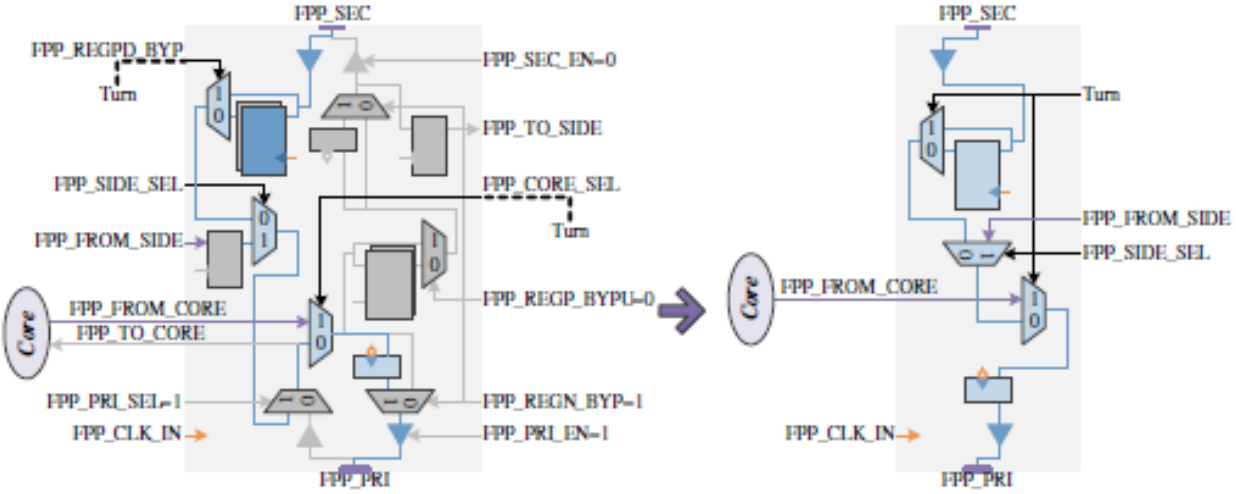



*Figure 3.3.8 Example of the flexibility of the FPP (SEC or Core to PRI).*

The purpose of the FPP is to enable parallel data flow into each die and between dies in the stack. In addition to the inter-die connection (fig. 3.3.7), it is also possible to connect the lanes with the core logic of the current die (fig. 3.3.8). This is done by its six terminals, in total. With FPP_PRI and FPP_SEC for connecting to the TSVs. With FPP_FROM_CORE and FPP_TO_CORE for connecting to the logic core within the die. And with FPP_FROM_SIDE and FPP_TO_SIDE for connecting to other FPP lanes. It may be composed of registered and non-registered lanes; the latter category can be further subdivided into clock lanes and non-clock lanes (fig. 3.3.9). The configuration of each of the lane elements is done with PTAP-accessible register bits (TDRs) that will hold their state as the FPP is being used to

apply tests. The lanes can each be controlled (if a pathway exists) to connect signals of the primary interface, secondary interface, and core to each other. These pathways might be registered to enable higher-frequency data communications across them. If registered, a clock can be provided from various sources. But it is envisioned that a clock might be easily connected through a non-registered lane, so special clock terminal names were selected, as its registration can be bypassed if so desired (FPP_CLK_IN and FPP_CLK_OUT).

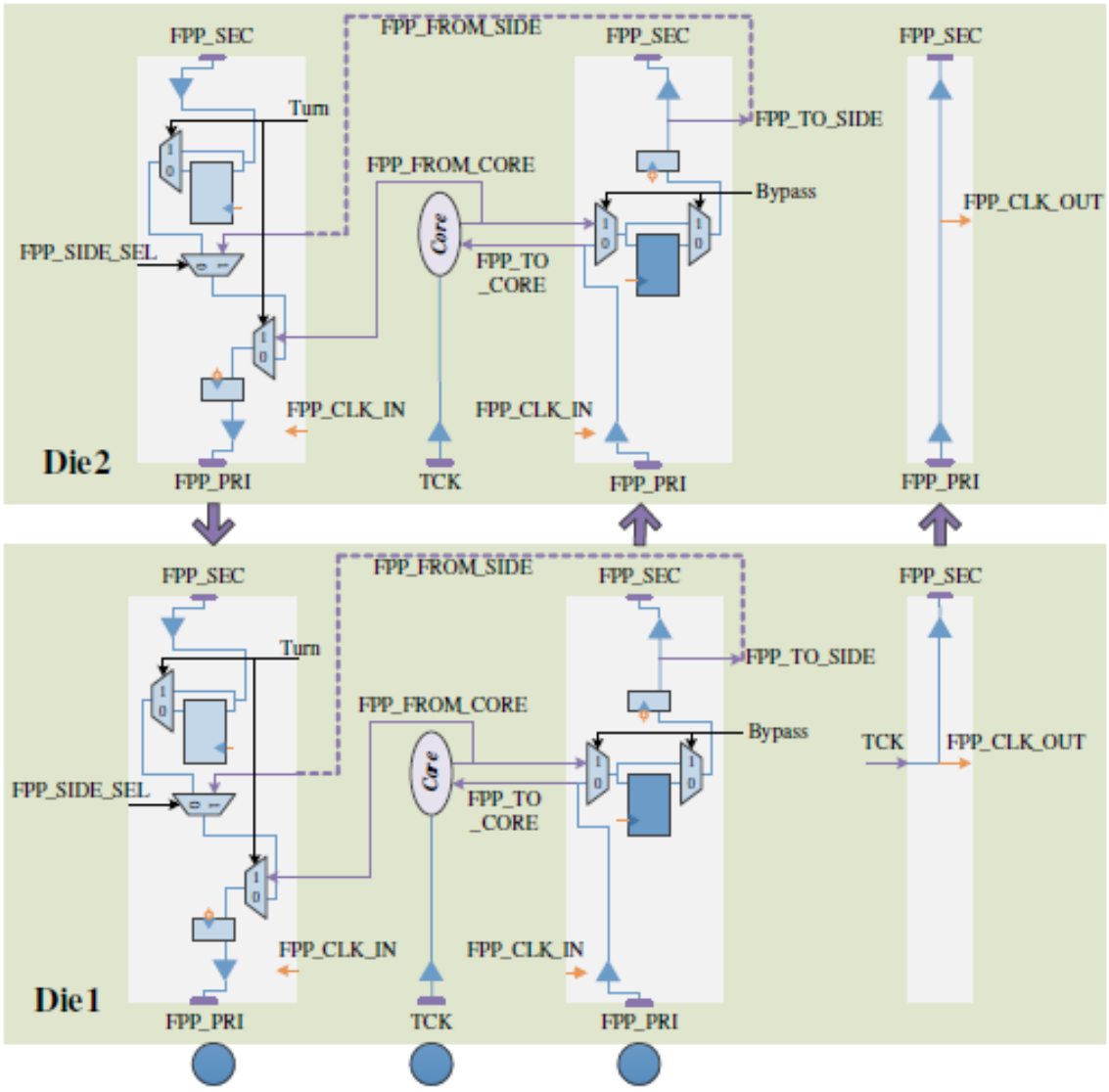



*Figure 3.3.9 Small example of FPPs on two dies.*

As mentioned throughout this presentation of the standard, IEEE 1149.1 Std. is leveraged for the serial control and data path. The stack-level and die-level interface is specified with respect to the 5 TAP interface signals. The PTAP controller

and register architecture is based off an IEEE 1149.1 Std. TAP controller and register architecture. The Instruction Register directs the device-level registers to be placed in the TDI-to-TDO pathway. Finally, the DWR is quite flexible in its construction. As such, an IEEE 1500 Std. core wrapper can become part of its content. That is how the standards can come together in such a way; because IEEE 1838 Std. allows it.

# CHAPTER 4.

# ARCHITECTURE



In Chapter 4 we present the general architecture of the 3D design used in this thesis. The design flow is presented first, and then the details on the TAM architectures examined.

## 4.1 Design Flow

Let us assume a theoretical die consisting of the 6 cores shown (fig. 4.1.1) for illustration purposes.

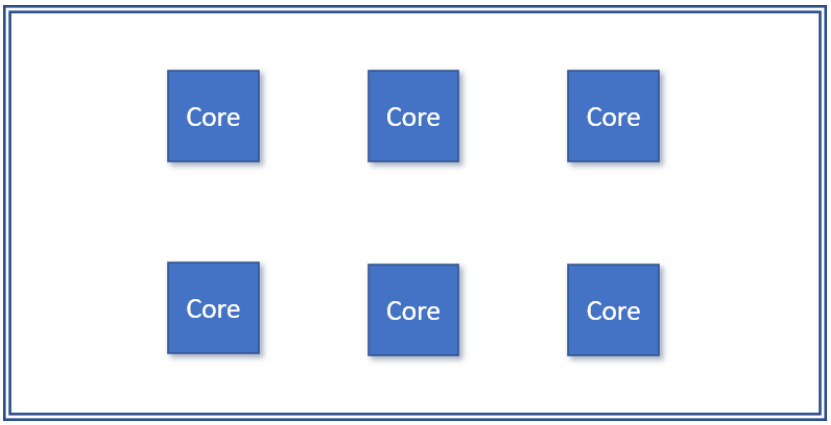



*Figure 4.1.1 A theoretical die which has 6 benchmark cores.*

Each of the benchmark cores has a different number of I/O ports, split into functional and testing ones. For ease of usage, they are wrapped with a façade, which splits the ports of the benchmark cores into functional and testing ports.

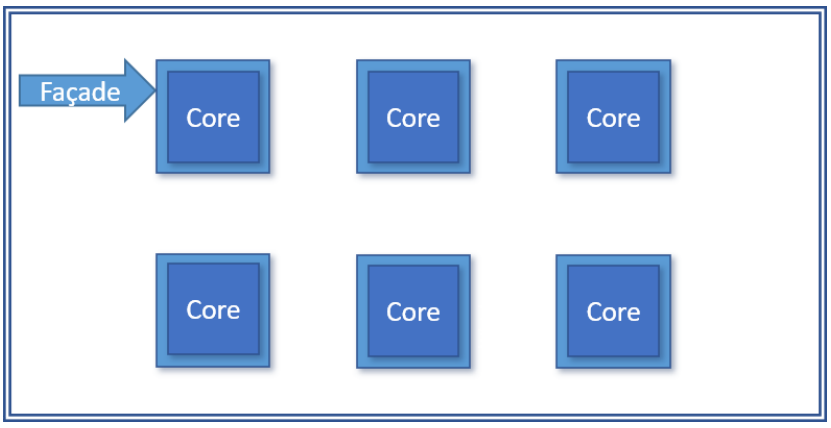



*Figure 4.1.2 Benchmark cores wrapped with the façade.*

The functional ports are used for the functional connections among the cores, while the testing ports are used solely for testing the die. The functional and testing ports were separated to enable the generation of the UNCON die. The UNCON has its test ports unconnected and serves as a reference die for measuring the exact amount of wiring used for the functional connections. By subtracting the wirelength of the UNCON die from the wirelength of any die with test connections, the exact wirelength of the test connections can be measured precisely.

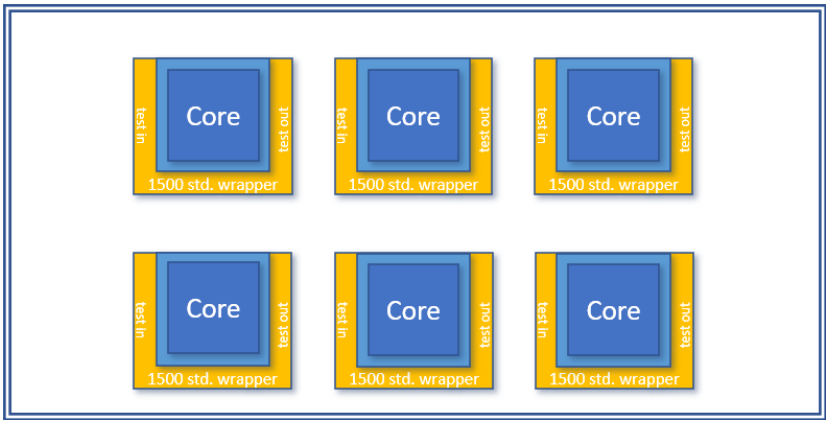



*Figure 4.1.3 Benchmark cores wrapped with the IEEE 1500 std. compatible wrapper.*

The next step is to wrap all the benchmark cores with IEEE 1500 std. compatible wrappers. These give them the ability to be tested serially with the connections between the cores, following the 1500 std. testing protocols.

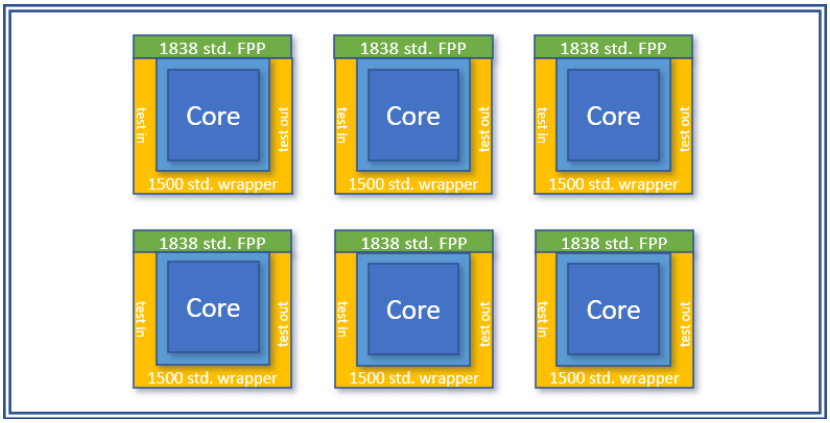



*Figure 4.1.4 Benchmark cores after the addition of the 1838 std. FPP.*

Additionally, the FPP switchboard is added to each of the cores, which works in tandem with each 1500 std. compatible wrapper. This enables in parallel testing, through the TSVs, adhering to IEEE 1838 testing protocols. However, this does not mean that the two testing schemes can only be used separately. For instance, all control signals can only be changed serially, through the connections of the 1500 std. Additionally, one can prepare all scan chains in parallel, but extract the results serially. Thus, the two testing schemes can seamlessly work together, because they use the same wrapper cells around each core.

Moreover, the two testing schemes can be used for different kinds of testing. For example, on the case of EXTEST, the connections between the cores can be accessed through the 1500 std. scheme, while the TSV connections between the clusters can be accessed through the 1838 std. scheme. The architecture is flexible in that regard, especially when it comes to the usage of the various bypass signals. One can skip over cores in the die, or even entire dies of the stack, allowing for the testing to happen in any way the testing process requires it to.

At this stage of the design flow the cores can be considered test-ready. Therefore, the next step in the flow is to generate the Test-Access-Mechanism (TAM) of each die. The TAM makes every die test-ready and offers the means: a) to connect the in-test ports of all cores to the source of test-data at the die level, in order to transfer the test data into the cores, and b) the out-test ports to the test sink at the die level, in order to transfer the test responses out of the cores. Even though many different TAM architectures have been proposed in the literature, we study

in this thesis two different TAM architectures: the bus architecture and the daisy-chain architecture. In the bus architecture all the test-ports of the cores are connected on two shared buses: one bus for transferring the test-data into the cores through the in-test ports, and one bus for transferring the responses out of the cores through the out-test ports. In the daisy-chain architecture the cores are connected in a sequential manner forming a virtual daisy-chain: the out-test port of every core drives the in-test port of the next core in the chain, while the first one is driven by the test source of the die, and the last one drives the test sink of the die.

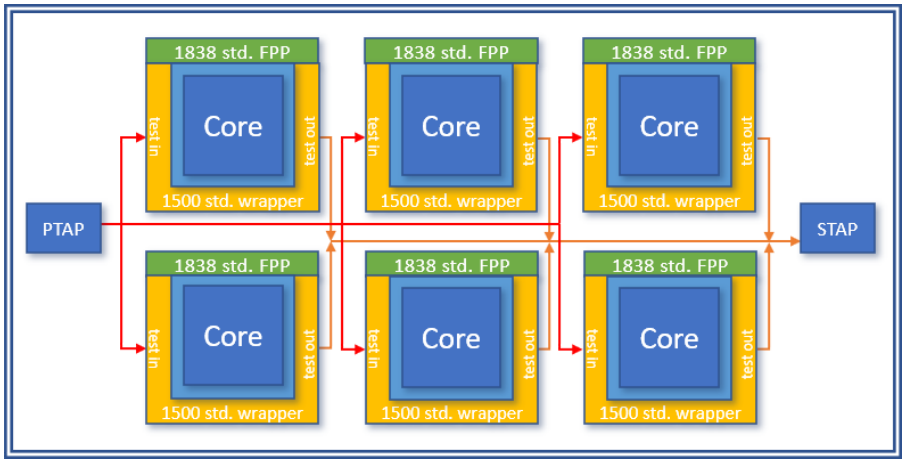



*Figure 4.1.5 An abstract rendering of the BUS die.*

In the first architecture shown (fig. 4.1.5), which will hereafter be called BUS, the TAM consists of two wide bus channels. The first is the input bus channel as it connects the input testing ports with the 1838 std. PTAP module, which serves as the TAM source at the die level. The second is the output bus channel as it connects the output testing ports with a bus ending with the 1838 std. STAP module, which serves as the TAM sink at the die level.

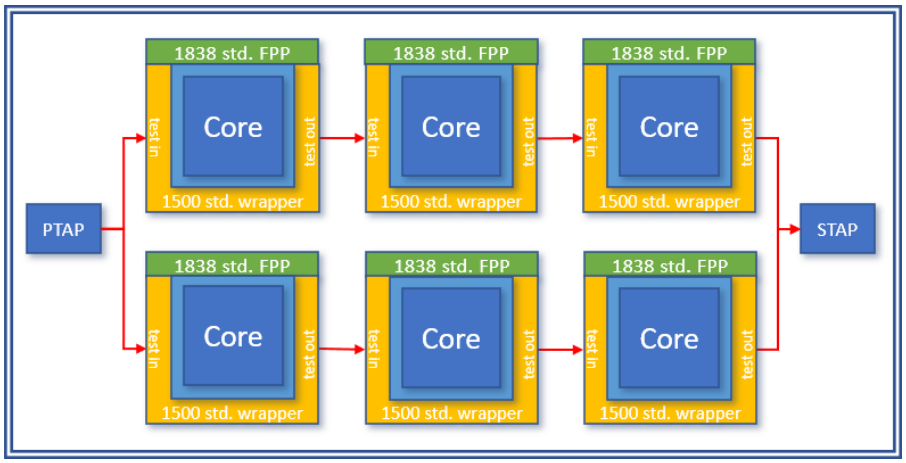



*Figure 4.1.6 An abstract rendering of the DAISY die.*

In the second architecture shown (fig. 4.1.6) which will hereafter be called DAISY, the TAM connects the testing ports of the cores in sequence, that is, the test-in port of each core is directly connected to the test-out port of the precursor core in the daisy chain. Since the daisy chain architecture requires longer test times than the BUS architecture, we use multiple daisy chains architectures in order to reduce the total test-time for shifting the test-data into the cores. In this thesis the TAM consists of two daisy chains at every die, but additional daisy chains can be used in a similar manner. The input of each of those chains is driven by the PTAP module, while their output drives the STAP module.

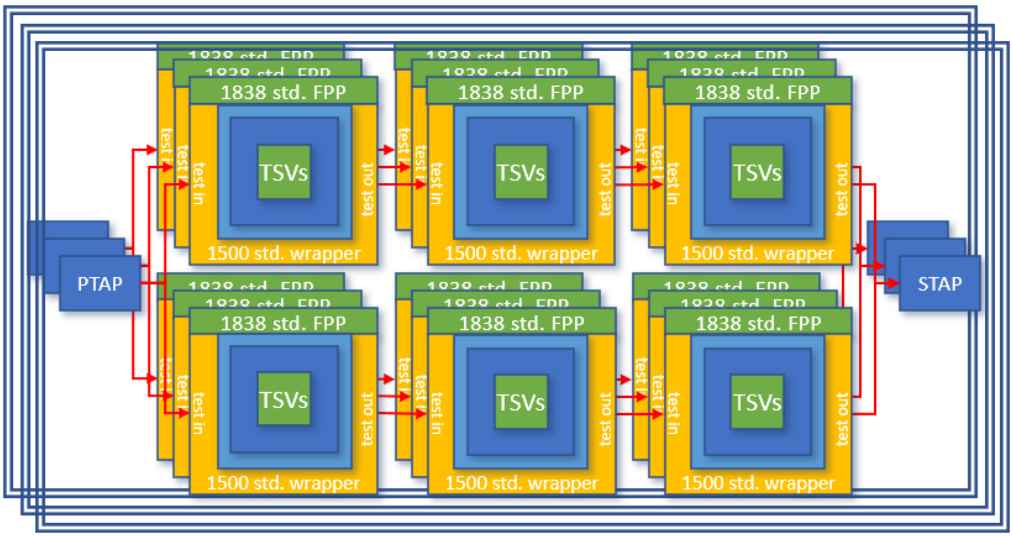



*Figure 4.1.7 An illustration of multiple DAISY dies in a 3D stack.*

In Fig. 4.1.7 we present the complete 3D test scheme developed in this thesis, which is a fully testable 3D stack, using the combination of the IEEE 1149.1-2013

Standard, the IEEE 1500-2005 Standard, and the new IEEE 1838-2019 Standard. A single die of this theoretical stack is presented in Chapter 6, along with the inner parts of a single wrapped core. Theoretically, and as it is illustrated, the testing connections of the BUS die should require additional routing than those of the DAISY die. This comparison is one of the main goals of this thesis, along with the design of this combination of IEEE standards.

We must note that the complete 3D scheme for serial testing implemented in this thesis assumes vertical connections among the PTAP and the STAP at the dies of the stack, which are realised by using TSVs. However, the TSVs through the IEEE 1838 std. FPPs accompanying every core can be used for in parallel testing. In Section 4.2 we present the implementation of the vertical connections as part of the TAM at the die level.

## 4.2 TAM Architecture

As mentioned in Section 4.1, we studied two different TAM architectures based on Time-Division-Multiplexing: the BUS architecture and the Daisy-Chain Architecture. Both architectures are very favourable in terms of test-times, as they explore the high bandwidth offered by the TSVs to transfer very fast the test-data to the entire stack and distribute these test data into the dies by using the local TAM at every die. This is enabled by the means of the Global Channels presented in "Testing 3D-SoCs" [43], which are very fast vertical connections based on TSVs. Hereafter we present the details of each part of the TAM architecture.

### 4.2.1 Global Channels

Global channels are fast vertical connections through the entire stack that use a small number of TSVs and TAM lines to transfer big volumes of test-data to the various dies at a high rate. The global test channels begin at the bottom die and they end at the top-most die. They consist of TSVs in the passive layers of the dies, and metal vias and buffers in the active layers of the dies. Test-data are time-multiplexed at each global channel at the bottom die, and they are transferred to the dies of the stack at the frequency supported by the TSVs. At each die the test-data are time-demultiplexed and distributed at every core using the slow shift-frequency permitted by the wrapper chain of the core and the TAM of the die.

Time-division multiplexing is applied in two dimensions, vertical and horizontal. In the vertical dimension, the test data are time-multiplexed in a round-robin fashion and they are transferred through the TSVs; at the first clock cycle, the test data of the first die are transferred; at the second clock cycle, the test-data of the second die are transferred, etc. In the horizontal dimension, the test data for different cores of each die are time-multiplexed at the specific clock cycles that the global channel transfers test data for the die, and they are transferred horizontally to reach the cores of that die. The vertical TDM depends on the frequency supported

by TSVs, while the horizontal TDM depends on the shift-frequency supported by the wrapper and the scan-chains of each core.

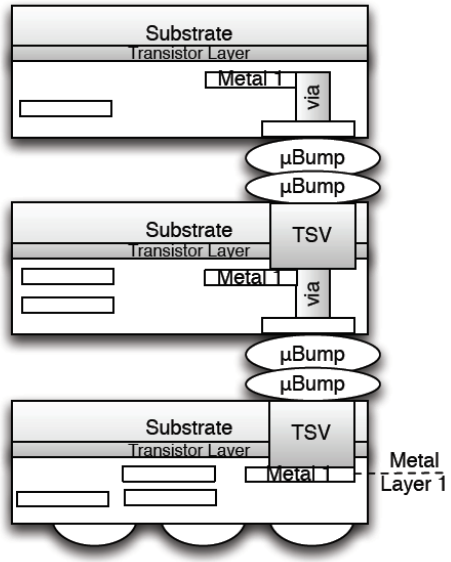



*Figure 4.2.1 Illustration of a Global Channel*

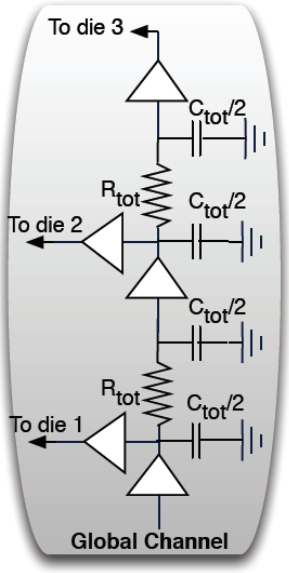



*Figure 4.2.2 Electrical model of a Global Channel*

Fig. 4.2.1 presents the structure of a global channel carrying one test-bit in the 3D stack. All dies face downwards, and they are connected face-to-back. The global channel begins from the first metal layer of the bottom die, and it goes through a TSV and two micro-bumps to reach the top metal layer of the middle die. Then, through successive metal vias it reaches the first metal layer and the transistor layer of this die, where it is connected to the local TAM structure. Then it is connected again to the first metal layer of the die, and through the next TSV and micro-bumps it is connected to the top metal layer of the third die. At each die, the signal of the global channel is transmitted to the local TAM using a buffer, and it is retransmitted upwards using a second buffer. The global channel ends at the first metal layer of the topmost die, where it is connected to the local TAM structure. The same structure in the reverse direction is used for global channels carrying test responses out of the IC.

In Fig. 4.2.2 the RC model used for three dies is presented. Rtot, Ctot represent the total resistance and capacitance of the TSV and the microbumps at each level of the stack. The inductance of the TSV and the micro-bump can be ignored because the global channels operate in the low-GHz frequency range. The maximum

shift-frequency for the IWLS benchmark cores was found using timing simulation to be in the range of 220MHz to 400MHz.

The proposed TAM architecture is modular, and it can be used for testing both partial and complete stacks. In the case of partial stacks, each global test channel ends at the die that is at the top of the stack. For pre-bond testing, the same mechanism can be used, provided that the circuit at the bottom die that multiplexes test-data from the ATE, is replicated at each die of the stack. Note that this circuit is very small, and it can be easily bypassed during post-bond testing. Finally, for 3D ICs with their clock network split across different tiers the redundant pre-bond clock tree is used [45] for connecting each core with the clock generated by the TDM scheme. If the pre-bond clock tree is not available, then the test clock inputs stemming from the other tiers must be bypassed and driven by the TDM clock signal.

### 4.2.2 Bus-based TAMs

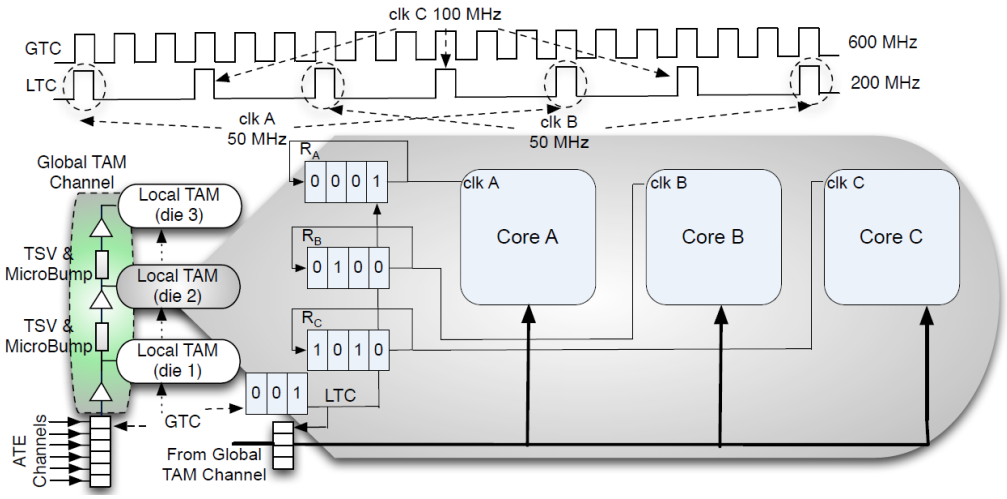



*Figure 4.2.3 An illustration of the Bus-based TAM architecture.*

The Bus-based TAM architecture [43] is shown in Fig. 4.2.3. Let $N_{Dies}$ be the number of dies in the stack. The Automatic-Test-Equipment (ATE) transfers test data with frequency $F_{ATE}$. The ATE channels are partitioned into groups, and each group loads one parallel to serial register of length $L$ at the bottom die of the stack. When it is loaded, this register transmits serially the test data through the

Global TAM line using Global Test Clock (GTC). W parallel global TAM lines form a global channel of width $W$. The global channel distributes the test data at the local TAMs, which are responsible for loading them into the cores. GTC is a periodic signal with frequency $F_{GTC} = F_{ATE} \times L$ generated either on-chip using a phase-locked loop, or it is provided directly by the ATE (note that $L = 1$ when the ATE can transfer test data with frequency $F_{GTC}$). First GTC is divided at each layer into a local test clock LTC (vertical TDM). The frequency of LTC, $F_{LTC}$, is lower or at most equal to the highest frequency that the TAM of the layer can transfer test data to the cores. Then, LTC is further divided at the local TAM at each layer (horizontal TDM) to shift test data into each core at different frequencies depending on the scan chain limitations and the power constraints of the core.

Fig. 4.2.3 shows one instance of a local TAM for the 3D SoC used in that work. Each layer is assigned one cyclical shift register of length $N_{Dies}$, which divides $F_{GTC}$ by the number $N_{Dies}$. Specifically, the pattern "001" rotates inside the register and it drives LTC with one active edge every $N_{Dies}$ active edges of GTC. Then, each core is assigned one cyclical shift register with length equal to $2N$, which divides $F_{LTC}$ by a value equal to $2^0$, $2^1$, $2^2$, . . . $2^N$. The length of the register is equal to the highest division required (it is equal to 4 in the example of Fig. 4.2.3 to provide division by 1, 2 and 4). The scan shift frequency for every core is set by loading appropriate non-overlapping patterns into each register before the testing of the cores begins. All shift registers are clocked using LTC (frequency $F_{LTC}$) and provide clock signals with frequencies equal to $F_{LTC}$, $F_{LTC}/2$, $F_{LTC}/4$ as it is shown in Fig. 4.2.3. At every cycle of LTC, $W$ test bits are available at the common bus ($W$ is the width of the global and the local bus). Since the patterns loaded into the registers are non-overlapping, only one layer is active at each GTC cycle and only one core loads the test data from the bus at each LTC cycle. In the example shown in Fig. 4.2.3 $F_{LTC} = F_{GTC}/3$ and the shift frequency for core A, B and C is set equal to $F_{LTC}/4$ (pattern 0001), $F_{LTC}/4$ (pattern 0100) and $F_{LTC}/2$ (pattern 1010) respectively, as shown in Fig. 4.2.3.

### 4.2.3 Daisy-Chain-based TAMs

The Daisy-Chain-based TAM architecture [44] is shown in Fig. 4.2.4 for a 3D-SoC consisting of three dies ($N_{Dies}$ = 3). At the lower level the test-data enter the stack, and they are transferred through a small number of TSVs to the various dies using high frequency. At every level of the stack the test-data are demultiplexed and they are shifted into the die using a division of the global-channel frequency. Then, they are distributed to multiple daisy chains using a second level of demultiplexing, and they are shifted into the cores by further dividing the shift frequency. During the shift/capture operations the daisy chains at every die are independently controlled using separate test clock and shift enable signals.

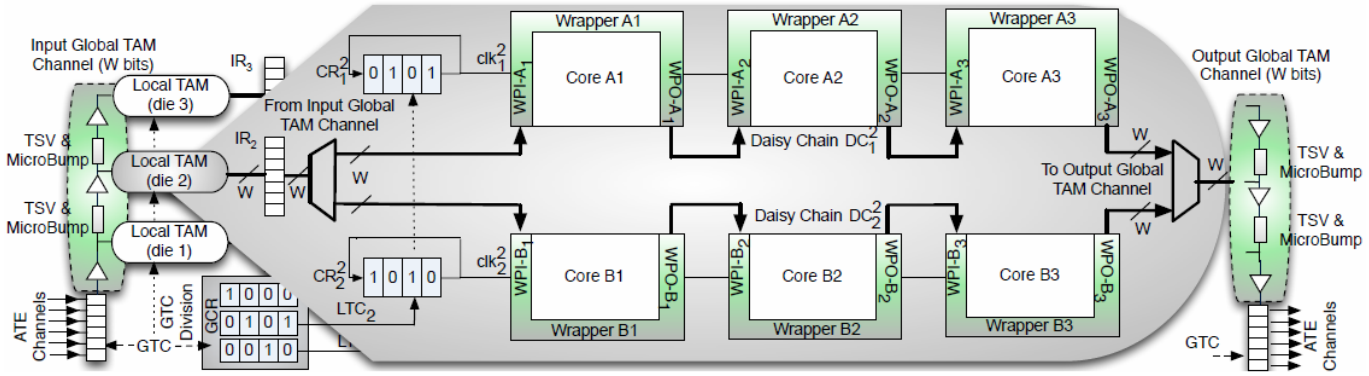



*Figure 4.2.4 An illustration of the Daisy-Chain-based TAM architecture.*

The Automatic-Test-Equipment (ATE) channels load one parallel-to-serial register at the bottom die of the stack with frequency $F_{ATE}$, which transmits serially the test-data through the Global TAM line synchronously with Global Test Clock, GTC (note that when $F_{ATE} \geq F_{GTC}$ the parallel-to-serial register can be omitted, and each ATE channel drives directly one global TAM line). $W$ parallel global TAM lines form a global channel of width $W$, which distributes the test data to the local TAMs using the local test-clocks $LTC_l$ generated for layers $l$ = 1,2,...,$N_{Dies}$. The local test clocks are divided versions of GTC generated by the global circular shift registers $GCR_1$,$GCR_2$,...,$GCR_N$. These registers circulate non-overlapping bit patterns that permit each clock cycle of GTC to be applied at a single die each time.

At every cycle of GTC $W$ test bits are transmitted over the global channel, and they are distributed among the interface registers IR by using the local test clocks. These $W$ test bits are stored into one selected interface register $IR_l$ upon the active

clock edge of $LTC_l$. Let $ND_l$ be the number of the daisy-chains $DC^l_1, DC^l_2, \ldots, DC^l_N$ Dl at layer *l*. The test-data stored into $IR_l$ are distributed among these daisy chains by further dividing the clock $LTC_l$ into clock signals $clk^l_1, clk^l_2, \ldots, clk^l_N$ ($clk^l_j$ is used to shift the test-data from $IR_l$ into $DC^l_j$ ). Like the division of GTC, the division of $LTC_l$ is achieved by using circular shift registers with nonoverlapping bit patterns, which enable $LTC_l$ to be applied on one daisy-chain at a time. For example, in Fig. 4.2.4 the test data are alternatively shifted into $DC^2_1$ , $DC^2_2$ by the means of two non-overlapping clocks $clk^2_1$ and $clk^2_2$ generated using circular registers $CR^2_1$ and $CR^2_2$ (similar register are used at every layer). Both of these registers shift non-overlapping patterns with the frequency of $LTC_2$ and generate two periodic signals with half the frequency of $LTC_2$. Each logic value equal to '1' that reaches the rightmost cell of every circular register $CR^l_j$ enables the test-data stored at $IR_l$ to be shifted into $DC^l_j$.

$DC^l_j$ connects several cores at layer *l* as follows: the WPI of the first core is driven by the local demultiplexing mechanism of the die, the WPO of the last core drives the output multiplexing mechanism of the die, and the WPIs (WPOs) of every intermediate core are connected to the WPOs (WPIs) of their predecessors (successors) cores. At every $DC^l_j$ one core is tested at each time instance, and the test-data are shifted into this core by configuring the wrappers of the rest of the cores (at the same chain) in bypass mode according to the IEEE 1500 standard (in that work both the parallel and serial ports of the wrappers are connected in daisy chains to support compatibility with the IEEE 1500 standard). The test-time $T(DC^l_j)$ for shifting all test-data into $DC^l_j$ is equal to the aggregate test-time of all the cores connected on chain $DC^l_j$.

The first step of the TAM optimization at the stack level ($K^3$) is to divide $F_{GTC}$ into local test frequencies $F_1, F_2, \ldots, F_N$ in a non-regular manner that depends on the specific test load and the test constraints of each layer. For example, let us assume that the three layers of the 3D SoC shown in Fig. 4.2.4 require (approximately) 4/7, 2/7 and 1/7 of the test data respectively (we assume that all TAM structures have the same bit-width). Then, $F_{GTC}$ is divided as follows: the circular shift regis-

ters $GCR_1$,$GCR_2$,$GCR_3$ are 7-bit wide (7 is the common denominator of the ratios 4/7, 2/7 and 1/7), and they are loaded with the non-overlapping patterns "0101011", "1000100" and "0010000" respectively (each pattern has as many logic '1' values as the number on the nominator of the respective ratio). However, even though $F_1 = 4/7 \times F_{GTC}$, two successive clock edges of GTC are applied at $LTC_1$ (the last two bits of the pattern "0101011" are both equal to '1'), therefore $F_1$ switches between $F_{GTC}$ (during the last clock cycle) and $F_{GTC}/2$ (during the rest of the clock cycles). In the cases that the higher value of $F_1$ ($F_{GTC}$ in the case at hand) cannot be supported by the local TAM design then another approximate division is selected, like for example 4/8, 3/8, 1/8 for the case at hand.

After dividing $F_{GTC}$ into $F_1,F_2,...,F_N$ the number $ND_l$ of the daisy-chains for each layer $l$ is set equal to $NDl \geq F_l(\max)/SF_{max}$, where $F_l(\max)$ is the maximum shift frequency supported by the daisy-chains at layer $l$ and $SF_{max}$ is the max shift frequency supported by the cores of layer $l$. Then, the frequency for each daisy-chain is set equal to $F_l/ND_l$, the cores of every layer $l$ are partitioned into $ND_l$ daisy-chains by applying the Karmarkar-Karp algorithm, and the cores are ordered by applying the Kruskal algorithm. Every ratio $F_l/F_{GTC}$ for layer $l$ must be nearly equal to the ratio of the test-data volume of layer $l$ to the test-data volume of the whole stack.

The selected ratios provide nearly equal test-times for the dies $TT(1) \approx TT(2) \approx ... \approx TT(N_{Dies})$ and minimize the test time of the stack $TT_{Stack}$ due to the parallel application of the tests using TDM. However, these ratios are very often not practical for frequency division, while there are also power constraints that prevent parallelization of certain tests. As a result, many values $TT(1), TT(2),..., TT(N_{Dies})$ may deviate a lot from $TT_{Stack}$ in practical applications. Nevertheless, this deviation provides a tolerance range for the lower test times to be increased without any impact on $TT_{Stack}$. This property is exploited to exchange cores among the $ND_l$ daisy-chains with aim to shorten the long interconnections.

Let $A \rightarrow B$ be the longest daisy-chain connection of layer $l$. We begin from this connection and we try to exchange one of the cores $A$, $B$ with a core $C$ of another

daisy-chain to replace this long connection with a short one. For all possible permutations A↔C and B↔C that do not violate the constraint $TT(l) \leq TT_{Stack}$ for $l = 1,2,\ldots,N_{Dies}$, the new daisy-chains are generated by applying the $K^2$. Among the permutations that reduce the wire-length of layer $l$, the permutation that offers either the highest wire-length reduction (criterion A) or the minimum increase of $TT(l)$ (criterion B) is selected, and the algorithm proceeds to the next permutation, until no further permutations are possible.

The complexity of the optimization process is very low even for large stacks that consist of thousands of cores (note that the optimization process is applied at every layer separately and thus the running time of the proposed method depends on the number of cores at each layer). The complexity is very low even for very large future 3D stacks, because technology limitations favour the integration scaling at the vertical instead of the horizontal dimension, limiting thus the potential number of cores integrated at every layer of the stack.

# CHAPTER 5.

# TAM-DESIGN AUTOMATION



As described, the testing flow starts with a rudimentary top-level design that contains some of the benchmark cores and ends with a fully synthesizable floorplan. In Chapter 5, we first describe a method used by the design, before we present the design automations themselves.

## 5.1 The Method-k3

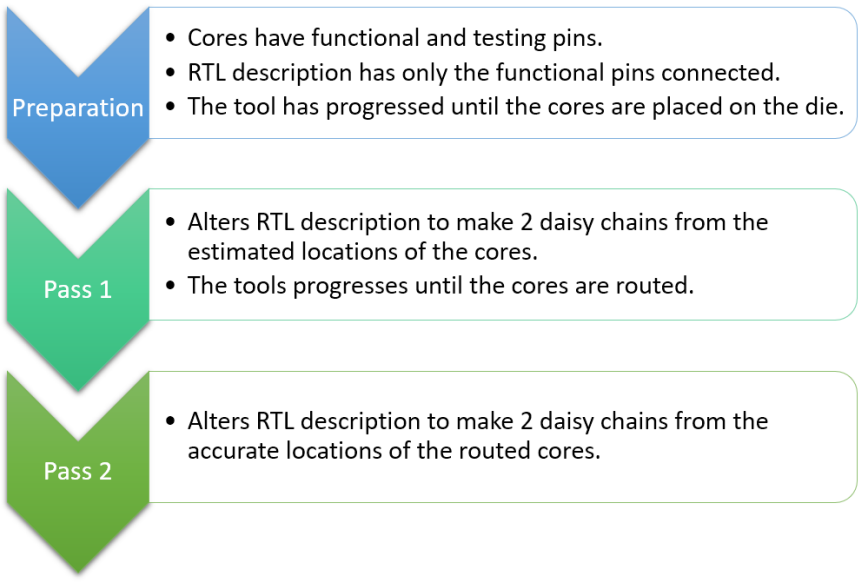



*Figure 5.1.1 Flowchart of the Method-k3.*

Named after Karmarkar, Karp, and Kruskal, the Method-k3 is a 2-pass algorithm. It was adopted in order to automate the successive experiments having to take place in search of better results and the progressive comparison between the daisy chain connection technique, versus the bus connected one.

1. *Preparation:*
    - Each core has functional and test pins.
    - An RTL description has been generated where the cores have only their functional pins connected. Their testing pins are left unconnected.
    - The synthesis has progressed until the *place_opt* phase of the tool, where the cores are placed in the die layout.
    - The location of each core is assigned to be the centre of their position, calculated from the results of the *get_bounding_box* command.
2. ***Pass 1:*** The TAM consisting of two Daisy Chains is generated using the estimated locations of the cores.
    - The preliminary TAM at the die level is generated by connecting the test pins of the cores in two Daisy Chains.
    - The synthesis is progressed until the *route_opt* phase of the tool, where the cores get routed automatically.
    - The location of each core is assigned to be the centre from its testing cells. Meaning, for each cell required out of *report_cells*, to find its location via *get_location*.
3. ***Pass 2:*** The TAM consisting of two Daisy Chains is generated using the accurate locations of the cores.
    - The final TAM at the die level is generated by connecting the test pins of the cores in two Daisy Chains.
    - The final RTL description of the layout has its functional pins connected as they were, and the testing pins connected in two Daisy Chains.

The Method-k3 is loosely based on the $K^3$ Algorithm described in Chapter 4 and used in the Bus-based and Daisy-Chain-based architectures described there. It was eventually altered for the reasons described in Section 5.2 of this chapter.

### 5.1.1 Karmarkar-Karp Heuristic

The Karmarkar-Karp Heuristic, formally "Complete Karmarkar-Karp: The differencing method of set partitioning" [46], takes a set of numbers, and splits them into a number of subsets which are as nearly equal as possible. For two subsets, it only has to take the two largest numbers, remove them from the set, and insert their difference back into the set. This represents the decision to put each of these numbers in a different subset. It proceeds this way until a single number remains. That single number is the difference between the two subsets. For reference, here is a Java implementation of the heuristic used within the Method-k3.

*Algorithm 5.1.1 The Karmarkar-Karp Heuristic used in the Method-k3.*

**Karmarkar-Karp Heuristic**

```
 1:  Int karmarkarKarpPartition(int[] baseArray) {
 2:      PriorityQueue<Integer> heap = new
 3:          PriorityQueue<Integer>(baseArray.length, REVERSE_INT_CMP);
 4:      for (int value : baseArray) {
 5:          heap.add(value);
 6:      }
 7:      while(heap.size() > 1) {
 8:          int val1 = heap.poll();
 9:          int val2 = heap.poll();
10:          heap.add(val1 - val2);
11:      }
12:      return heap.poll();
13:  }
```

The trick of this implementation is the "REVERSE" flag which changes the Priority Queue as to give priority to the highest values within it, and the "poll" method which returns the value with the highest priority. Thus, it can take the integer values of the base array and execute the heuristic. Of course, for the Method-k3 additional code has to be added in order to know which values were chosen for

each of the two subsets, the values themselves being the total length of the WBR of the cores.

### 5.1.2 Kruskal Algorithm

The Kruskal Algorithm, formally "Kruskal's algorithm on the shortest spanning subtree of a graph and the traveling salesman problem" [47], finds a minimum spanning forest of an undirected edge-weighted graph. Each time, it selects an edge of the graph with the minimum weight, that does not form a cycle, and adds it to the solution. For reference, here is the pseudocode of the algorithm used within the Method-k3.

*Algorithm 5.1.2 The Kruskal Algorithm used in the Method-k3.*

**Kruskal Algorithm**

```
1:  Algorithm Kruskal(G) is
2:      F := ∅
3:      for each v ∈ G.V do
4:          MAKE-SET(v)
5:      for each (u, v) in G.E ordered by weight (u, v), increasing do
6:          if FIND-SET(u) ≠ FIND-SET(v) then
7:              F := F ∪ {(u, v)}
8:              UNION(FIND-SET(u), FIND-SET(v))
9:      return F
```

Given a connected graph, such as the one for the Method-k3, the algorithm finds a Minimum Spanning Tree (MST). Using Euclidean distance, it can connect all cores in an MST, which this thesis uses for the connection of the daisy chains. However, it was discovered that not all MSTs produced could be translated into correct testing circuitry.

## 5.2 Categories of PATs

In order to create a floorplan that combines the three standards mentioned earlier on, parts of the design flow described in Chapter 4 were automated. The major design automations developed in this thesis are the following:

1. We automatically wrap the different benchmark cores in a way that groups their functional and testing pins separately. This helps the uniformity of the circuit description, and the testing connections between the cores.
2. We automatically tie the cores to specific locations in the floorplan of the dies. This helps the dies have a specific structure, and to form daisy chains on the estimated location of the cores. This is particularly important when IP cores comprise the SoC.
3. We automatically find the accurate location of every logic cell on the dies. This optimizes the design of the daisy chains and takes advantage of the tool's own placement optimizations.
4. We automatically lock the specific locations of the logic cells between the dies. This is important for the addition of the TSV ports between the dies.
5. We automatically identify the best daisy chain connections to minimize routing. This is done with the "Method-k3", which has been presented in Section 5.1.
6. We automatically generate a floorplan of bounded cores with specific logic cell locations and TSV ports for its 20 clusters, and a total of 60 benchmark cores. This is done in preparation for the addition of the 1838 std. compatible FPPs into the floorplan.
7. We automatically form TSV ports at the centre of each bounded cluster, complete with a customizable keep-out zone surrounding them. In that way the dies adhere both to the 1500 std. and the 1838 std. at the same time.

Finally, the floorplan comprises a 3D stack which adheres to the IEEE 1149.1-2013, IEEE 1500-2005, and IEEE 1838-2019 standards. examined. The automa-

tion in this thesis is achieved with using Python Aid Tools, which will hereafter be called PATs. PATs are divided into distinct categories, which are presented here.

### 5.2.1 Aid for Top-Level Design

The motivation for the PATs of this category is to effect changes on the RTL description of the floorplan. They serve as the starting point for all other PATs and are the only PATs which require input from the user. Category A PATs are the ones which use a large number of other PATs directly.

*Table 5.2.1 Category A.# PATs.*

| | Category A: Aid for Top-Level Design |
|---|---|
| **A.1** Die Reader | Identifies specific commented out tags and collects info from the die description directly. |
| **A.2** Layer Maker | Replaces the tags with useful code, to manage and form the functional connections for 30 cores, and the test connections by the daisy chain information. |
| **A.3** Small Layer Maker | Generates a bounded floorplan with 6 cores. |
| **A.4** Medium Layer Maker | Generates a bounded floorplan with specific cell locations with 12 cores. |
| **A.5** Large Floorplan Maker | Generates a bounded floorplan with specific cell locations and TSV ports with 20 clusters, 60 cores. |

"A.1 Die Reader" requires the usage of a notation system by placing Verilog comments in the description of the three dies, tags which the PAT can replace with functional connections. Initially, PAT A.1 seeks those specific tags, and by strategically placing these tags around the Verilog description, it identifies the specific cores, their input and output sizes, the shape of the two test chains, but also their closest neighbours on the die itself. Specifically, in 85% of the cases it connects the functional pins of a core with the functional pins of a neighbouring core, and in the rest 15% of the cases it opts for either a long connection, or a connection to one of the die's I/O pins. Nevertheless, it retains the same connections on all three dies, so that the functional connections remain the same amongst them for comparison purposes. Even though the functional connections among the cores on the

floorplan are selected randomly they resemble a realistic scenario where most of them are short connections between neighbouring cores, and a small number of them are long connections instead. With a couple prompts for the user, things like the width of the test chains, the randomness of the connections, and their threshold for long connections, are all controlled without having to rewrite the Verilog description of the dies.

"A.2 Layer Maker" is the PAT which alters the RTL description depending on the particular part of the Method-k3 applied. PAT A.2 is an evolution of PAT A.1, as it imports and exports Daisy Chains, and connects the test pins of the cores automatically depending on the Daisy Chains assigned to it. It achieves that by adding only one more prompt for the user, to separate which pass of the Method-k3 it is implementing.

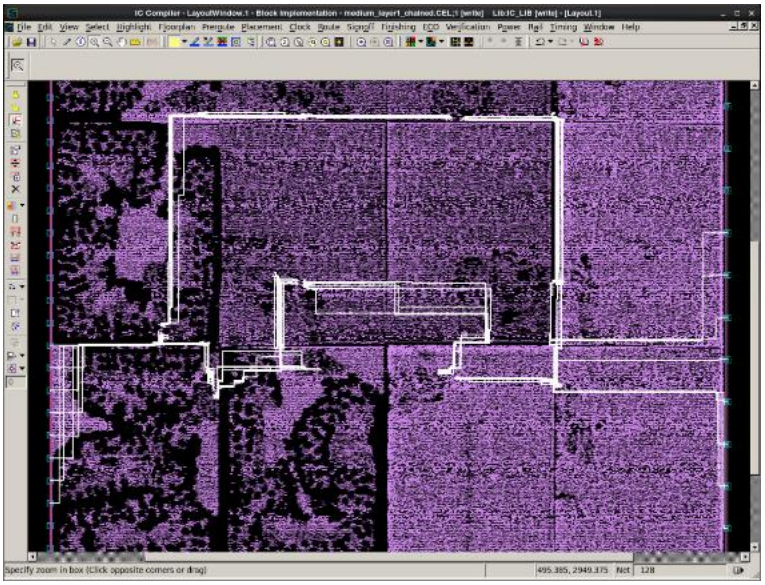


*Figure 5.2.1 Floorplan of 12-core DAISY circuit.*

In order to experiment on the total routing of dies with different amounts of cores, "A.3 Small Layer Maker" which is compatible with 6 cores, and "A.4 Medium Layer Maker" which is compatible with 12 cores (fig. 5.2.1) were developed. Hence, other than the neighbouring cores which change depending on their placement, and the different number of connections between the cores, these two PATs are very similar.

Lastly, "A.5 Large Floorplan Maker" creates a floorplan which contains 20 clusters, each of them containing 3 cores. These are *fpp_des*, *fpp_eth*, and *fpp_con*, which in turn contain the known *wrapper_des*, *wrapper_eth*, and *wrapper_con* cores. However, while before these used the generic RTL description, in this version they use a netlist of the cores that contain the embedded scan chains. This was the reason for the replacement of the *vga_enh_top* core with the *wb_conmax_top* core of the same benchmark. The tool could not embed the scan chains on the *vga* core, so it had to be replaced. At the same time, exactly half of the pins of the replacement core could be used, else the tool would announce an error message for the core exceeding an inner limitation for its number of pins.

### 5.2.2 Aid for Bounding Boxes

The motivation for the PATs of this category is to create scripts which force the tool used to tie the cores of the design to specific bounded locations. The scripts made from Category B PATs are called in the beginning of the process to create the finalized floorplan.

*Table 5.2.2 Category B.# PATs.*

| | Category B: Aid for Bounding Boxes |
|---|---|
| **B.1** Bounding Box and Report Cell Maker | Forms a .tcl script which bounds cores to specific parts of the dies, and requests the cell report for each of them after. |
| **B.2** Reader of Bounding Boxes and Maker of Crude Chains | Reads the data from the previous script and uses it to make the first general daisy chains, the ones the method will later improve. |

In the Preparation phase of the Method-k3, the location of each core has to be calculated. This is done with the help of PAT "B.1 Bounding Box and Report Cell Maker", which reports the cells of the cores.

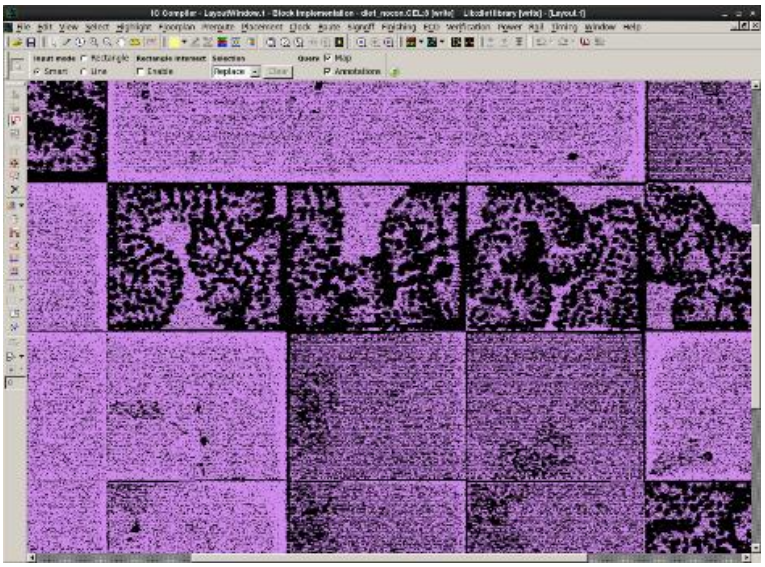


*Figure 5.2.2 Floorplan of UNCON circuit with the use of bounded cores.*

Similarly, after PATs A.3 and A.4, PAT "B.2 Reader of Bounding Boxes and Maker of Crude Chains" followed, which is required after the first pass of the Method-k3 on the two versions of the dies. Meaning that, this is the PAT that runs the first pass of the Method-k3 on the floorplans with 6 and 12 cores.

### 5.2.3 Aid for Cell Locations

The motivation for the PATs of this category is to find the cell locations of the cores. The idea of a "core" is quickly abandoned by the tool, replaced by the logic cells that comprise it. Category C PATs are the ones that shift through all cells, keeping a list of the ones that matter for the testing scheme.

*Table 5.2.3 Category C.# PATs.*

| | Category C: Aid for Cell Locations |
|---|---|
| **C.1** Find all Cells | Requests all cells on the die, per core. |
| **C.2** Make Get Locations | Using the previous data, forms a .tcl script requesting the specific locations of the cells. |
| **C.3** Finder of Cell Names | Analyses and reads only the leaf cells of the non-functional parts of the die. |
| **C.4** Maker of Detailed Get Locations | Using the previous data, forms a .tcl script requesting the specific locations of the chosen leaf cells. |

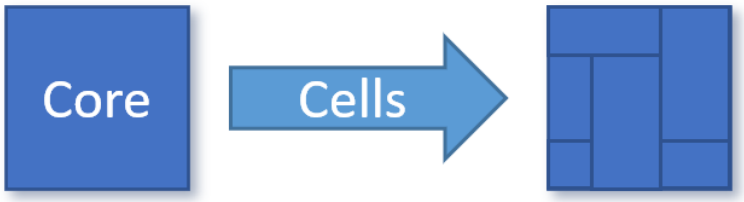



*Figure 5.2.3 An illustration for Category C.# PATs.*

With the information from PAT B.1, only the cells in relation to the test pins are useful, as the functional ones do not matter for the daisy chains. This is done with PAT "C.1 Find all Cells" which finds the cells needed by PAT "C.2 Make Get Locations". In tandem, those two PATs shift through the cell data, creating a smaller list of commands for the location of each cell in question.

Same as before but for the information from PAT B.2, and accounting for the different names, locations, and test data connections, PATs "C.3 Finder of Cell Names" and "C.4 Maker of Detailed Get Locations" were created. This was done to request from the tool the location of the cells that are needed to be kept at the same location between the two techniques.

### 5.2.4 Aid for Top-Level design with Cell Locations

The motivation for the PATs of this category is to utilize the locations of all logic cells which take part in the testing scheme in order to use their locations. That is to say, the data from Category C PATs. Category D PATs also change the RTL description of the floorplan, by using Category E PATs.

*Table 5.2.4 Category D.# PATs.*

| | **Category D: Aid for Top-Level design with Cell Loca-** |
|---|---|
| **D.1** Floorplan Reader | Finds the bounding boxes of each of the cores and collect their locations. |
| **D.2** Compare Locations | Using the previous data, compares the locations of old and new floorplans, to test their similarity. |
| **D.3** Reader of Detailed Locations and Maker of Set Cell Locations and Detailed Chains | Collects all specific location data, using it to create a .tcl script which forces the tool to use the specific locations on subsequent time. Forms chains from the specific locations. |

| D.4<br>Maker of Set Cell Locations | A subset of D.3, it executes the Pass 2 of the Method-k3, to the new cell locations. |
|---|---|

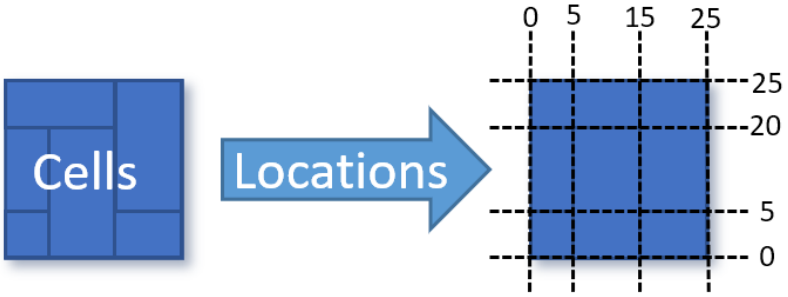



*Figure 5.2.4 An illustration for Category D.# PATs.*

PATs "D.1 Floorplan Reader" and its evolution PAT "D.2 Compare Locations" require the data from PATs C.1 and C.2. Originally made to see if the average location of the cells of the cores are in similar places between the three dies – which they were not, and thus the need for bounds to be created became clear. With the Method-k3, the original idea was that multiple passes might have been needed before the positions of the cells settled in particular locations. However, PAT D.2 revealed that two passes suffice: one with the estimated chains, and one with the accurate ones.

Similarly, after PATs C.3 and C.4, two more PATs are needed: PATs "D.3 Reader of Detailed Locations and Maker of Set Cell Locations and Detailed Chains" and "D.4 Maker of Set Cell Locations". They execute the second pass of the Method-k3, as it was described in Section 5.1, on the floorplans which have been made with PATs A.3 and A.4, the 6-core and 12-core floorplans respectively.

### 5.2.5 Aid for Method-k3

The motivation for the PATs of this category is to implement the Method-k3. Category E PATs cannot be used on their own, as they are made to help Category D PATs in order to add the daisy chains to the RTL description of the floorplan.

*Table 5.2.5 Category E.# PATs.*

| | Category E: Aid for Method-k3 |
|---|---|
| **E.1** Function K3 | Executes the Methdod-k3 as it has been described earlier. |
| **E.2** Route Length Adder | Estimates distances of lengths between connections. |
| **E.3** Exhaust Graph | Forces the testing connections to reach all cores on the die for the Method-k3, avoiding the bad case of MST. |

The Karmakar-Karp Heuristic and the Kruskal Algorithm, which have been presented in Section 5.1, are implemented in PAT "E.1 Function K3" for the Daisy Chains to be decided on, from the respective locations of each Pass of the Method-k3. When it comes to the first of the two algorithms used, timing data was taken from earlier experiments of the same benchmark cores, so as for the two chains to have as close of a maximum frequency as possible.

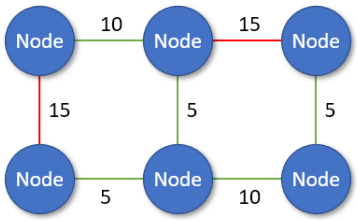



*Figure 5.2.5 Example weighted graph of what causes a bad MST.*

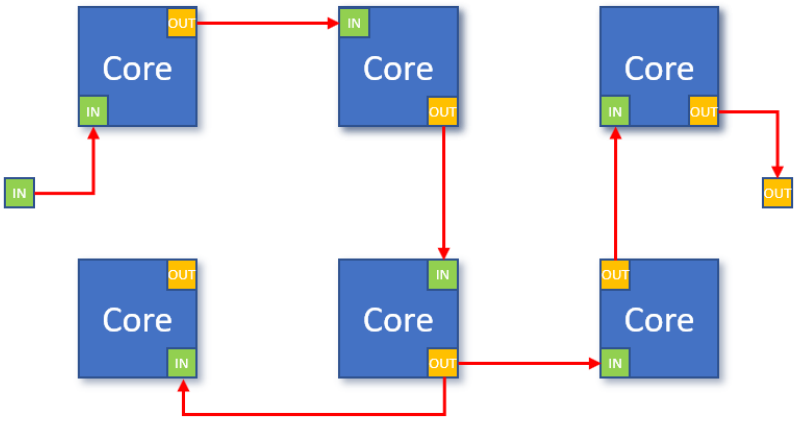



*Figure 5.2.6 Example of bad MST for Method-k3.*

The Method-k3, and especially its part about finding the Minimum Spanning Tree (MST) of the location of the pins after the placement of the cores, has a weakness. That is that any method which tries to achieve low routing by using any sort of MST algorithm can create circuits that cannot be routed. The more cores exist on a die, the more are the chances the MST produced cannot be translated into one coherent chain of connections (fig. 5.2.6). So much so, that in the DAISY die with the 12 cores, a manual connection was required. The reason can be described best by this short theoretical example die with 6 cores, where the ports of its cores move a little bit, and yet produce wildly different MSTs.

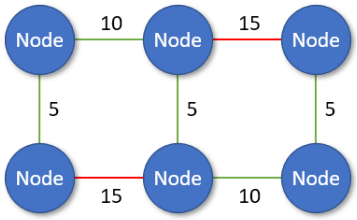



*Figure 5.2.7 Example weighted graph of what causes a good MST.*

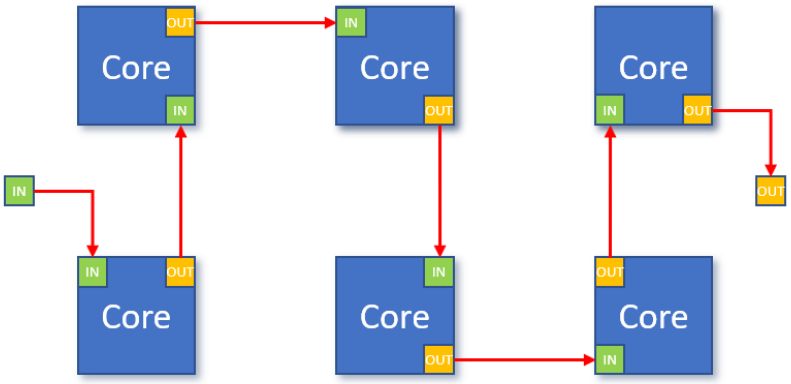



*Figure 5.2.8 Example of good MST for Method-k3.*

Both examples are MSTs, and yet only one of them can reliably be made into chained connections for the cores (fig. 5.2.8). Of course, this can be solved be requesting not for the Minimum Spanning Tree, but for the contiguous MST instead, as it was eventually done for the Method-k3.

Thus, two more PATs were created, "E.2 Route Length Adder" and "E.3 Exhaust Graph". PAT E.2 is simple and has to do with the way the tool represents its results, splitting them among metal layers, and signals. However, PAT E.3 signals the end of the Method-k3, replacing its intelligent algorithms with a brute force calculation of all combinations of paths among the cores, seeking the smaller one that is contiguous. Of course, its price is that as the number of cores approaches the first hundred, its running time skyrockets exponentially, even if it will always come to the right conclusion. This was acceptable, as there is no experimental need to exceed such an amount.

## 5.2.6 Aid for TSVs

The motivation for the PATs of this category is to use inner-die ports to simulate the openings for the TSVs. Category F PATs replace Category B PATs, as they create new scripts which both tie the cores to specific locations, but also place them away from the keep-out margin around the TSV ports.

*Table 5.2.6 Category F.# PATs.*

| | Category F: Aid for Vias |
|---|---|
| **F.1**<br>Via Maker | Able to form TSV ports at specific locations on the die, complete with a keep-out zone. |

| F.2<br>Bounds and Vias Maker | Able to form TSV ports at the centre of each bounded cluster, complete with a customizable keep-out zone surrounding them. |
|---|---|

Inspired by IEEE 1838 Standard, PAT "F.1 Via Maker" was created to overcome a fundamental issue with the CAD tool used in all the experiments. That is the lack of any modelling for TSVs. Therefore, they had to be approximated as much as possible. To this end, two commands were identified to simulate TSVs on the die, *set_pin* and *keepout_margin*.

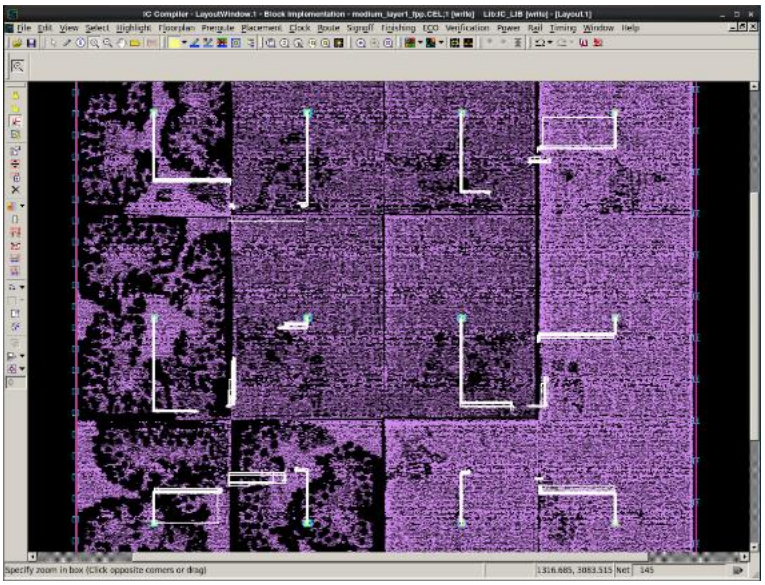


*Figure 5.2.9 Floorplan of DAISY circuit with TSV ports.*

Finally, PAT "F.2 Bounds and Vias Maker" places each cluster in place, and attaches a TSV up until Metal Layer 1, and a TSV down until Metal Layer 6, with alternating directions.

## 5.2.7 Aid for Benchmark Cores

The motivation for the PAT of this category is to equate the functional and testing port names of the benchmark cores, which greatly helps the uniformity of the circuit description. The Category G PAT is used only so that the RTL description used by Category A PATs is simplified, no matter the specific benchmark core used.

*Table 5.2.7 Category G.# PATs.*

| | Category G: Aid for Benchmark Cores |
|---|---|
| **G.1**<br>Façade Maker | Wraps the different benchmark cores in a way that groups their functional and testing pins separately. |


*Figure 5.2.10 An illustration for Category G.# PATs.*

Finally, the three cores are made similar by PAT "G.1 Façade Maker" and can become compatible with the IEEE 1500-2005 Standard by being wrapped. They can then become compatible with the IEEE 1838-2019 Standard with the addition its FPP functionalities, which in turn require the total floorplan of the die to adhere to the IEEE 1149.1-2013 Standard.

# CHAPTER 6.

# DETAILED DESIGN



In Chapter 6 we present the Large Floorplan in detail, with the help of an RTL Viewer. The chapter is split into three sections, with a top-down approach. Section 6.1 offers a general understanding of the entire floorplan of the theoretical 3D stack. Section 6.2 offers an exploration of the modules of a specific die of the stack, while Section 6.3 of the modules of a specific core of the stack implemented.

## 6.1 Floorplan

The Large Floorplan is what PAT A.5 creates with the help of PATs G.1 and F.2, and what will be presented in this chapter. It consists of a layer of a 3D stack that can be used multiple times, as is guaranteed by IEEE 1838 Std. Each layer has 20 clusters, and each of them has 3 benchmark cores inside it. These are *des3_perf*, *eth_top*, and *wb_conmax*. As it has been described in Chapter 4, each of these cores is wrapped with an IEEE 1500 Std.-compatible wrapper and paired with an IEEE 1838 Std. FPP Switchboard. The testing pins of the clusters are connected with the daisy chain technique, because of the analysis which will be presented in Chapter 7.

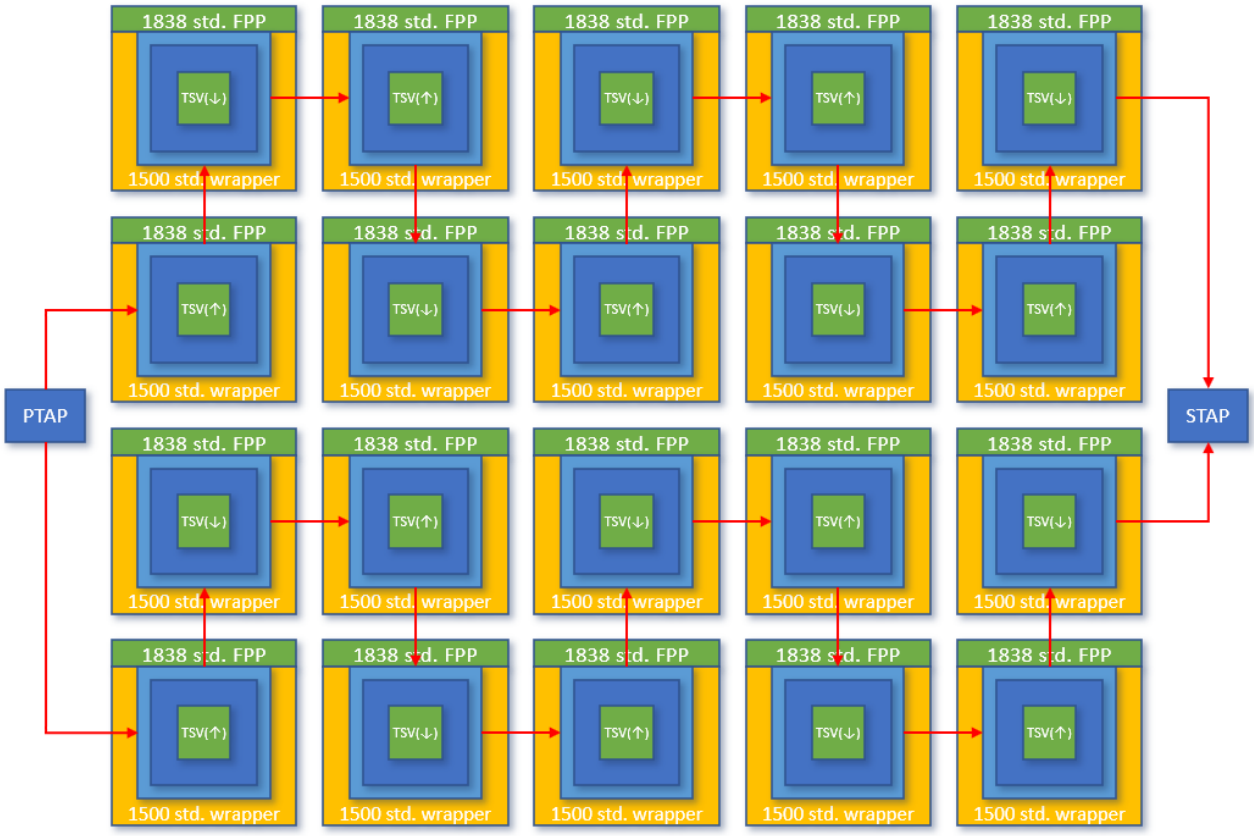



*Figure 6.1.1 An illustration of the Large Floorplan.*

In Fig. 6.1.1 the Large Floorplan is presented. The PTAP and STAP modules are linked serially with the clusters. The red connections show the two daisy chains. Each cluster has TSVs for the parallel connections through the FPP, with a direction either towards or away the stack's I/Os. This is how every non-terminal layer of the Large Floorplan looks like.

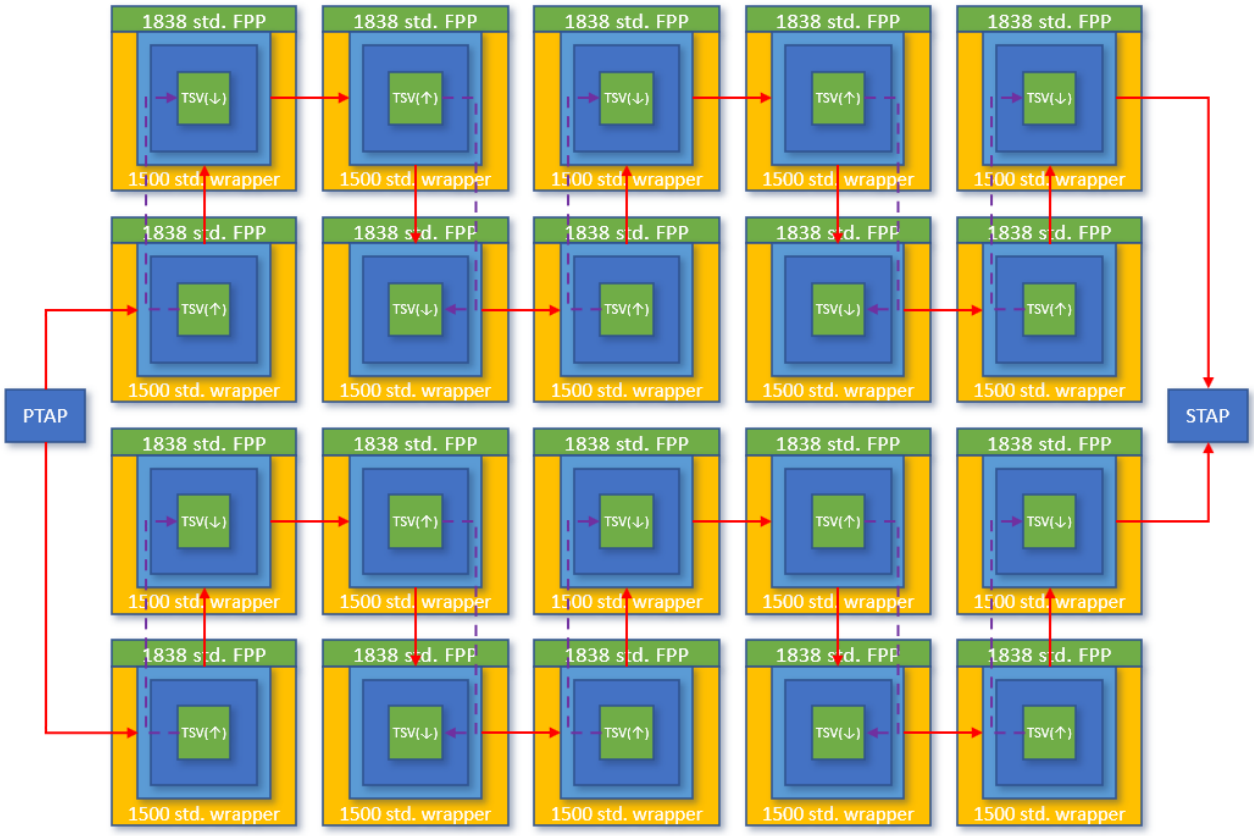



*Figure 6.1.2 An illustration of the terminal layer of the Large Floorplan.*

In Fig. 6.1.2. the terminal layer of the Large Floorplan is presented, the layer furthest away from the stack's I/Os. Its only addition are the purple connections which show the parallel connections between the clusters. In all other layers, the only parallel connections are the TSVs themselves, which move data either from or towards the stack's I/Os. The difference between the non-terminal layers of the 3D stack and its terminal layer is that the TSVs moving data away from the stack's I/Os have to be connected to the TSVs moving data towards the stack's I/Os, in order to complete the circuit.

## 6.2 Die Modules

In order to implement the three different standards, all of their specifications had to be meticulously combined in order to be able to follow the right composition of the thousands of rules, recommendations, and permissions.

As for the standards themselves, they offered almost no help as to how they can be brought together, assuming it is possible, but skirting around the issue with an abstract high-level theoretical approach. This is made clear from the following extract, at section 5.5.6.2. (page 41) of the IEEE 1838-2019 Standard, which is the description of the Instruction Register: "A more complex DWR may be composed of boundary-scan cells, wrapper elements from IEEE 1500 compliant cores, and individual IEEE 1838 compliant wrapper cells. Control of this complex DWR might come from the instruction loaded into the Instruction Register, SIBs, scannable TDR bits, WIRs, or other serially accessible elements. These elements may, in turn, not be available in the fully composed DWR. So, accessing the DWR might take several scan operations and various configuration settings before the proper content and test mode is established. But, in some cases, a single instruction such as SelectDWR - EXTEST, SelectDWR - INTEST, and SelectDWR - TRANSPARENT could be used."

Which is also evident in the permissions of the specification of the register design of the DWR (at section 6.1.1. at page 45) as the standard mentions that "An IEEE 1500 wrapper cell may be shared as a DWR cell and used in the DWR scan chain(s) if it is compliant to all IEEE Std 1838 DWR cell rules."

Thus, an exploration of the true shape of the Large Floorplan is required, along with all minor choices made in its creation, as the combination of the three standards can create wildly different results, simply by the chosen set of permissions and recommendations to be implemented.

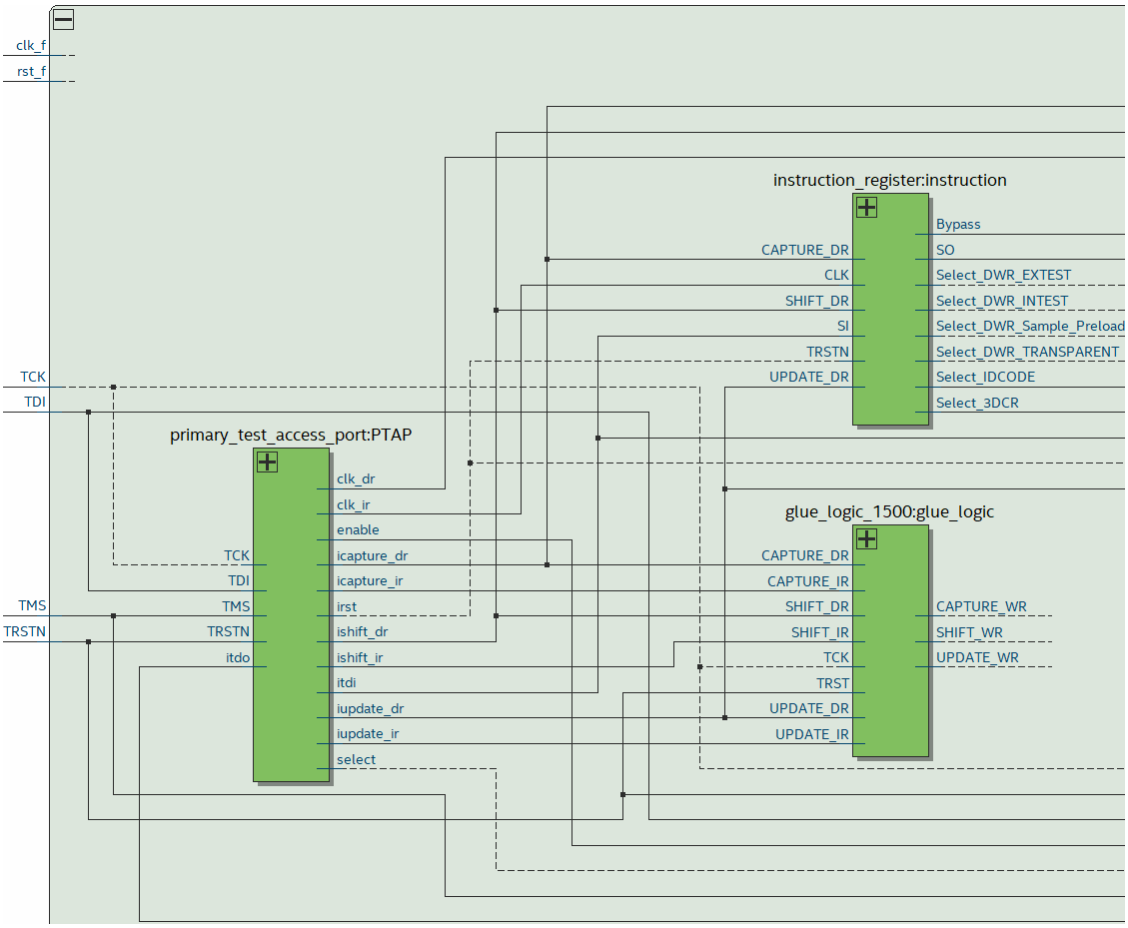



*Figure 6.2.1 Left part of Large Floorplan.*

In Fig. 6.2.1 the part of the Large Floorplan close to the PTAP can be seen, along with the testing inputs of the die. The modules shown are:

- The Primary Test Access Port (PTAP)
- The Instruction Register
- The Glue Logic

This is the left side of the RTL Viewer, meaning that these are the modules used by almost all others following them. The PTAP is extensively described in the IEEE 1838 Std. as is the Instruction Register. As for the Glue Logic, it is mentioned in the IEEE 1500 Std. as a way to convert the signals made from the FSM of the IEEE 1149.1 Std to the signals required by the WBRs of the IEEE 1500 Std.; that is the FSM the PTAP contains, and that is why each die of the total stack is compatible with IEEE 1149.1 Std. on its own too.

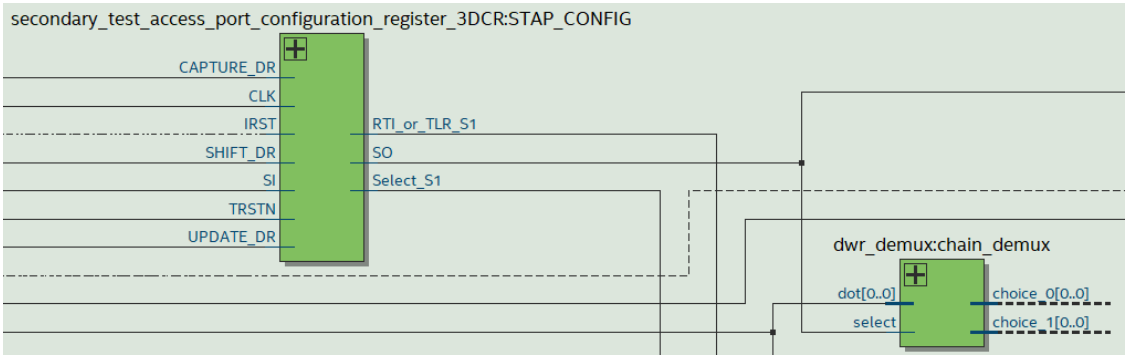



*Figure 6.2.2 Middle part of the Large Floorplan.*

In Fig. 6.2.2 the part of the Large Floorplan around the 3DCR can be seen. The modules shown are:

- The Secondary Test Access Port Configuration Register (3DCR)
- A demultiplexer

For this viewing, the 20 clusters with their 60 cores in total were hidden, so that every other part of the die would be more easily visible. Thus, at the middle of the remaining die are the 3DCR and the demultiplexer that splits the test serial input into two chains, which coil through 10 clusters each. The signal controlling this will be presented when the "STAP_CONFIG" module is described further on.

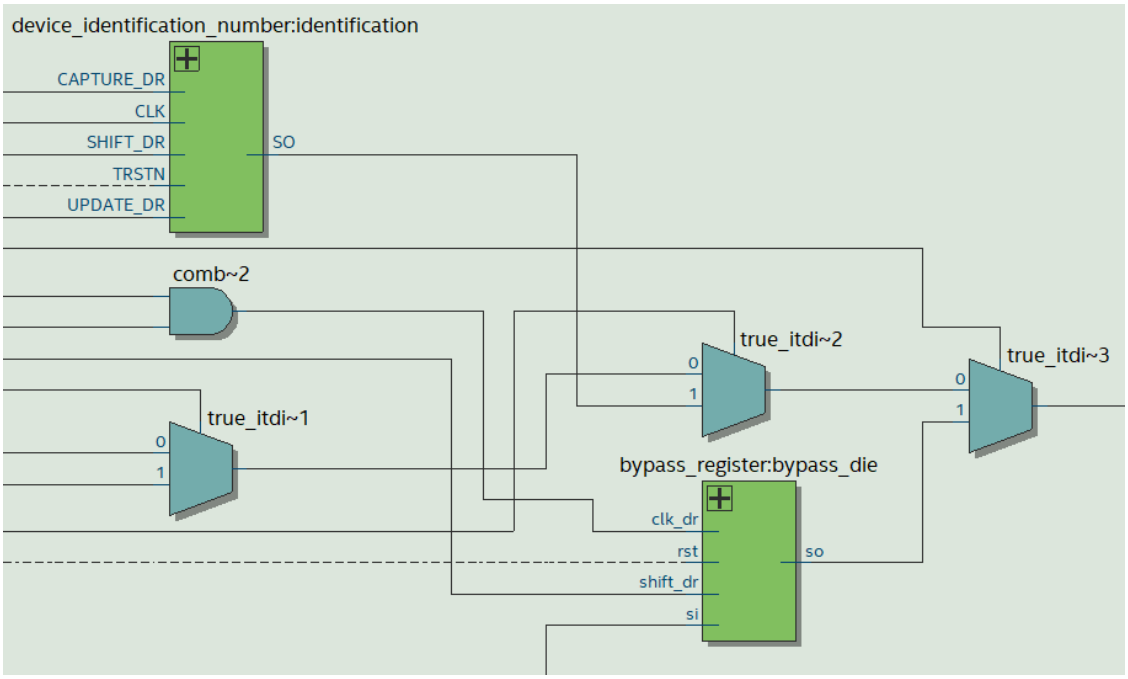



*Figure 6.2.3 Signal choice of the Large Floorplan.*

In Fig. 6.2.3 the part of the Large Floorplan nearby the outputs can be seen, which selects the testing signal from all options of the IEEE 1838 Std. The modules shown are:

- The Device Identification Register
- The Bypass Register

At this part of the circuit, the various signals that can leave the die from the serial output are weeded out. Starting with the Device Identification Number register. Followed by the Bypass Register of the entire die for when a part of the 3D stack does not need to be tested. And finishing with the two chains mentioned before, as only half of the die can be serially tested. That limitation does not exist when the cores are tested in parallel, as will become evident when one of the cores will be presented completely.

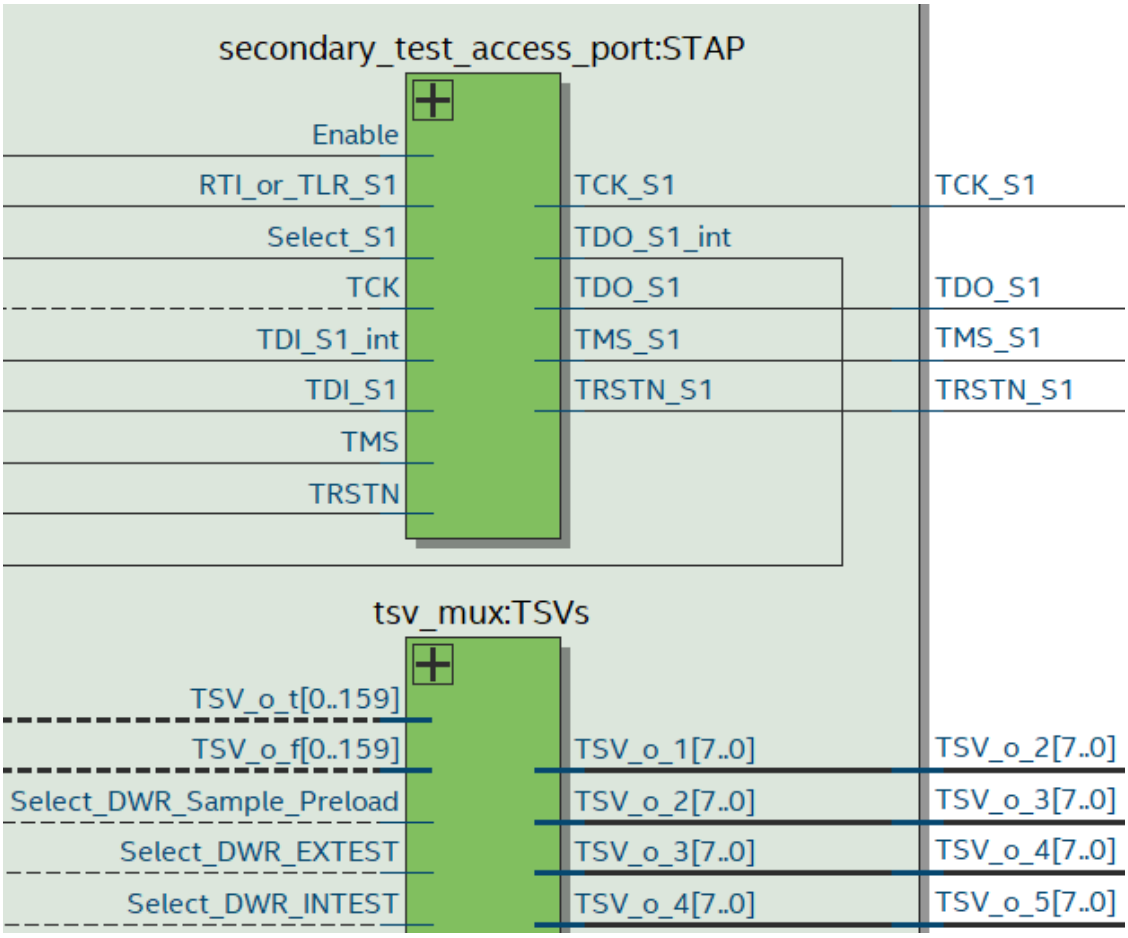



*Figure 6.2.4 Right part of the Large Floorplan.*

In Fig. 6.2.4 the part of the Large Floorplan close to the STAP can be seen, along with the testing outputs of the die. The modules shown are:

- The Secondary Test Access Port (STAP)
- A wide multiplexer

Finally, the rightmost part of the circuit has, as it is expected, the STAP as defined by the IEEE 1838 Std. and also a large multiplexer which allows for the TSVs to be used both functionally when the circuit is not under test, but also as parallel testing channels through the usage of the FPP as shown further on.

The outputs of the Large Floorplan, other than the alternating input and output TSVs, are what is required for the next die to also be IEEE 1149.1 Std. compatible, meaning the signals the next PTAP down the line requires.

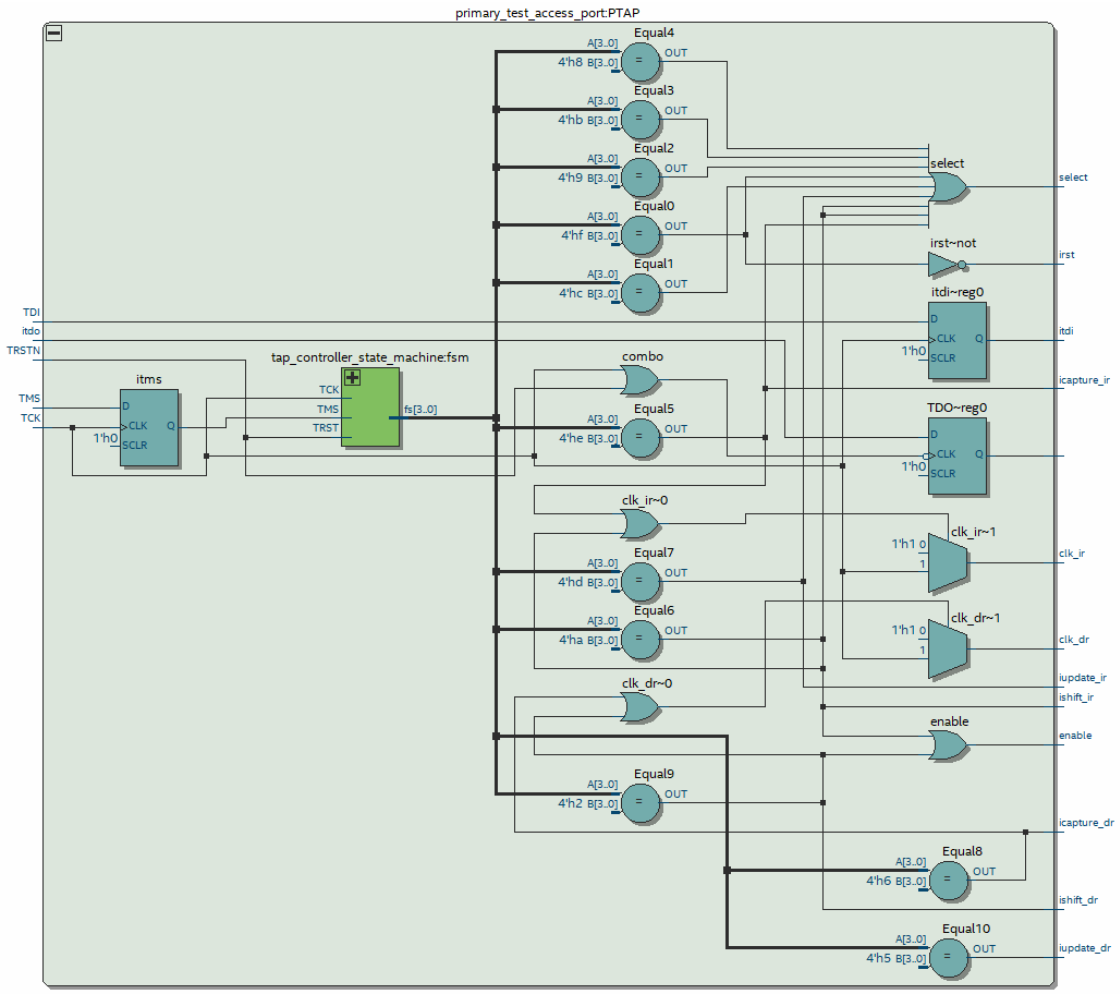



*Figure 6.2.5 The PTAP module.*

The PTAP (fig. 6.2.5) is what converts the simple *TDI*, *TRSTN*, *TMS*, and *TCK* signals into useful control signals for the rest of the circuit, through the states of the IEEE 1149.1 Std. compatible FSM it contains. The rest of its circuitry is rudimentary, made out of simple gates that correctly combine the 16 encoded states of the FSM in order to create the total of the control signals.

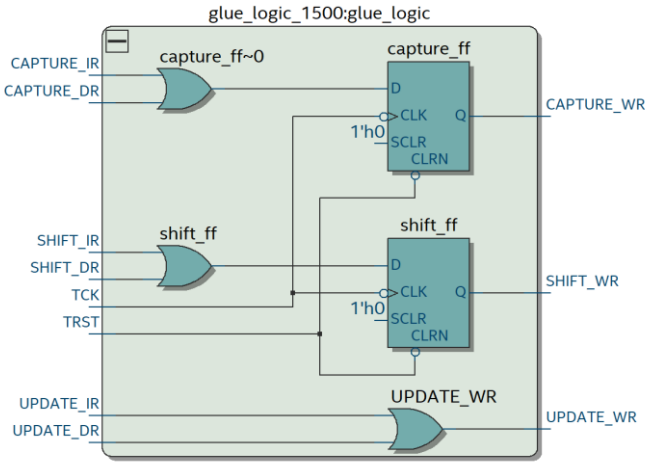



*Figure 6.2.6 The glue logic module.*

As defined in IEEE 1500 Std. the glue logic module (fig. 6.2.6) is a simple addition right after the FSM, in order to combine its DR and IR signals into unified WR signals.

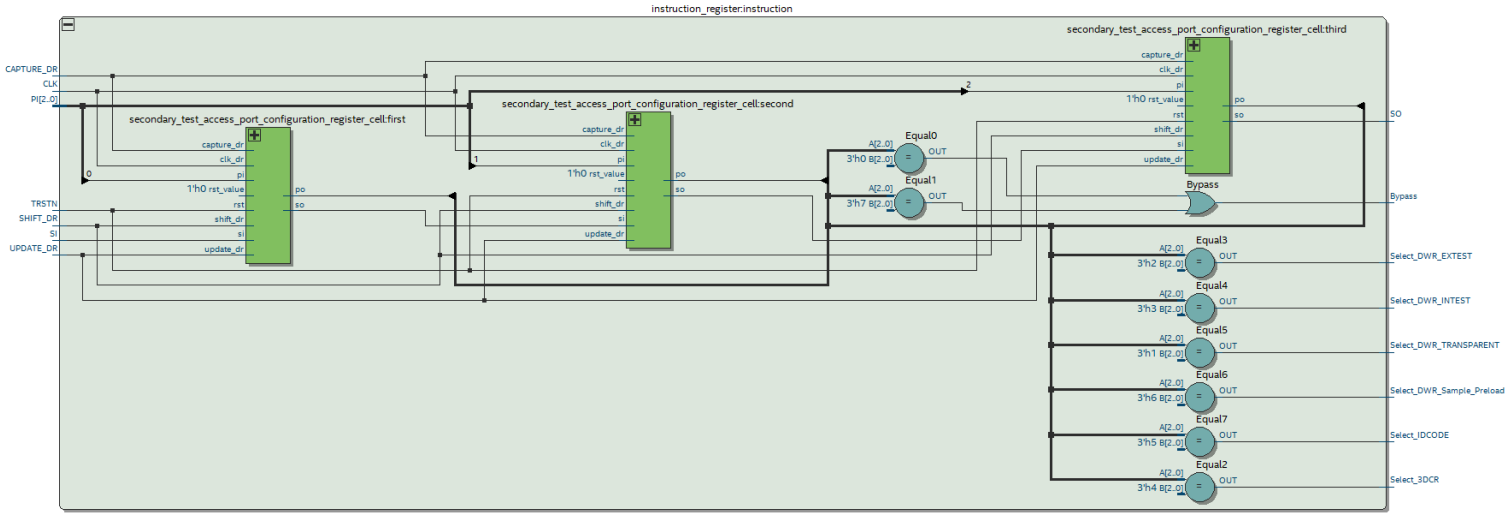



*Figure 6.2.7 The Instruction Register module.*

Made out of simple register cells, linked serially to hold the bits of the instructions, the Instruction Register (fig. 6.2.7) creates the rest of the control signals the circuit requires. As mentioned earlier, it also controls which of the two chains is

deemed active, and that is done by the *SO* output of the third register cell. Normally such a value would be useless, as the instructions appear on the *PO* outputs of the register cells, and this is why it was chosen for this task.

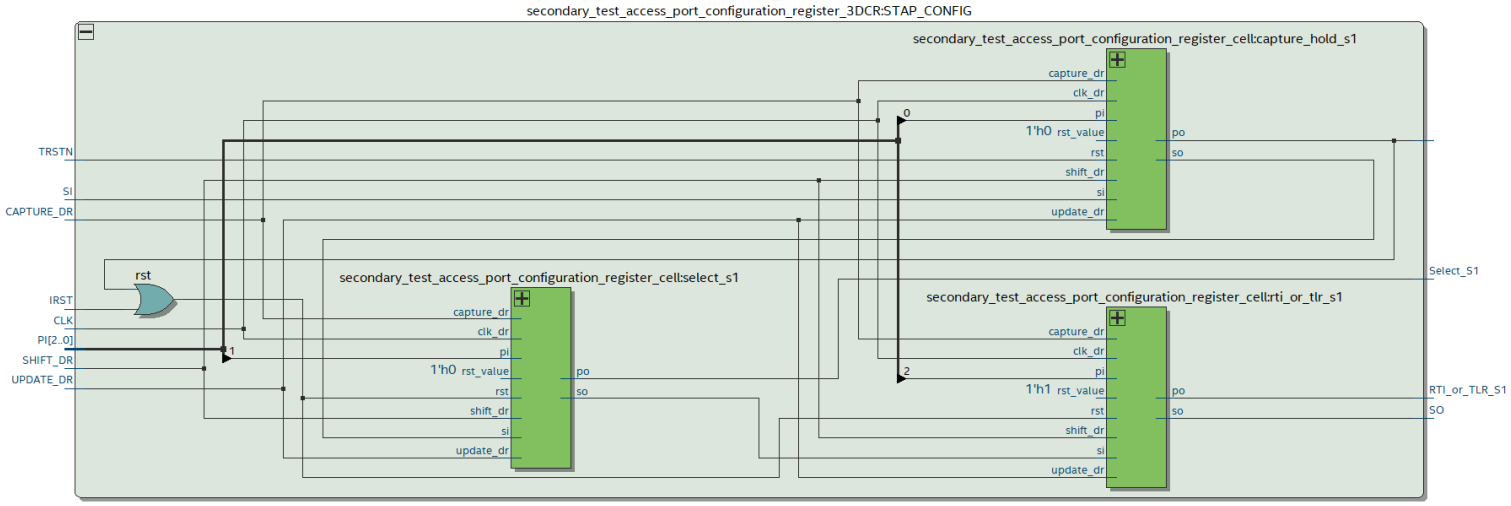



*Figure 6.2.8 The 3DCR module.*

Defined in IEEE 1838 Std. the 3DCR module (fig 6.2.8) is very simple in the circuit, because the Large Floorplan (DAISY) has only one STAP, so selecting it to be active is a certainty. Still, it creates two more signals, one of which is for resetting to a de-asserted state. A state which makes the STAP configuration bits and all the STAPs persistent through the Test-Logic-Reset action of the PTAP controller's FSM. And one for defining the parking state of the individually distributed *TMS_Sn* signals associated with the individual numbered STAP_Sn.

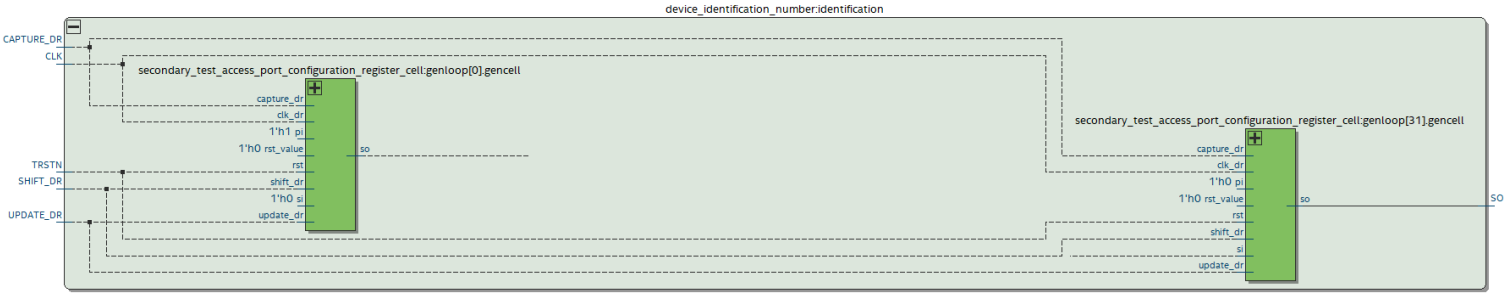



*Figure 6.2.9 The device identification number module.*

Made out of 32 register cells, the device identification number module (fig. 6.2.9) shifts out the device identification number of the circuit. It has to be includ-

ed in any IEEE 1838 Std. compatible circuit, even though it plays no role in the functional or testing modes of the circuit.

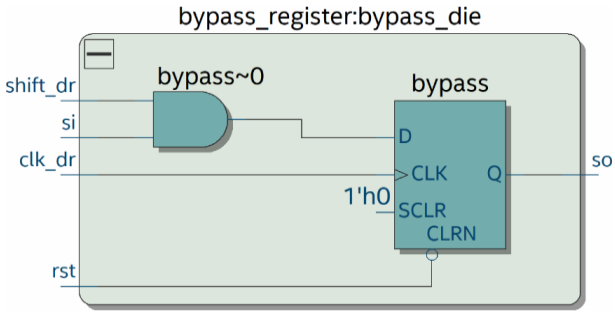



*Figure 6.2.10 The bypass register module.*

Including a single register, it should not be confused with the bypass registers of each of the cores. This is the main bypass register (fig. 6.2.10), the one for the entire die, which renders the entire level of the stack transparent to testing.

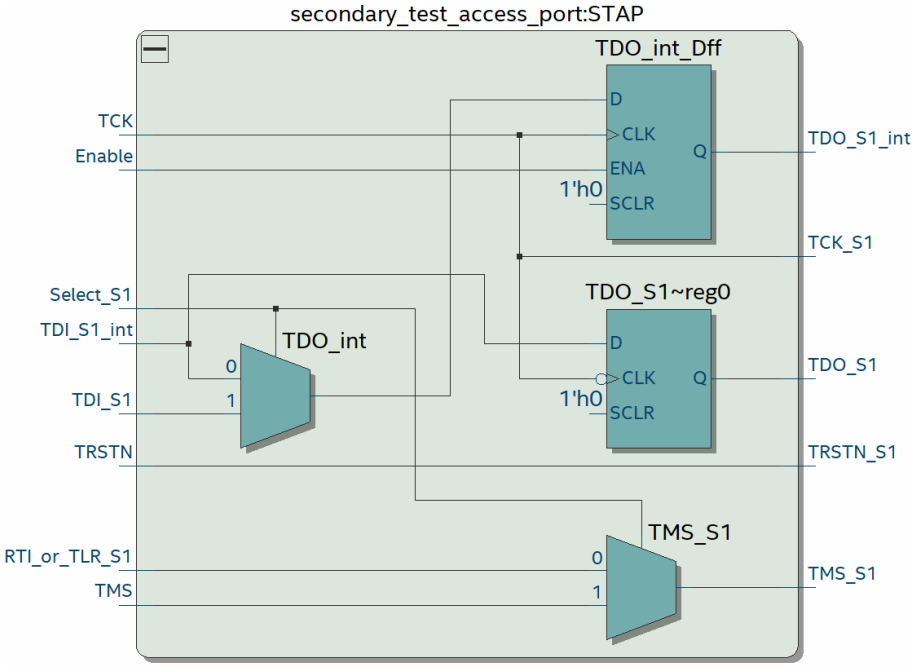



*Figure 6.2.11 The STAP module.*

Sitting on the other end of the circuit, away from the PTAP module, the STAP module (fig. 6.2.11) exists to collect and register the signals required by the PTAP on the next level of the stack. It is controlled by the 3DCR, and it is where all possible *SO* signals converge into one true *TDO* signal.

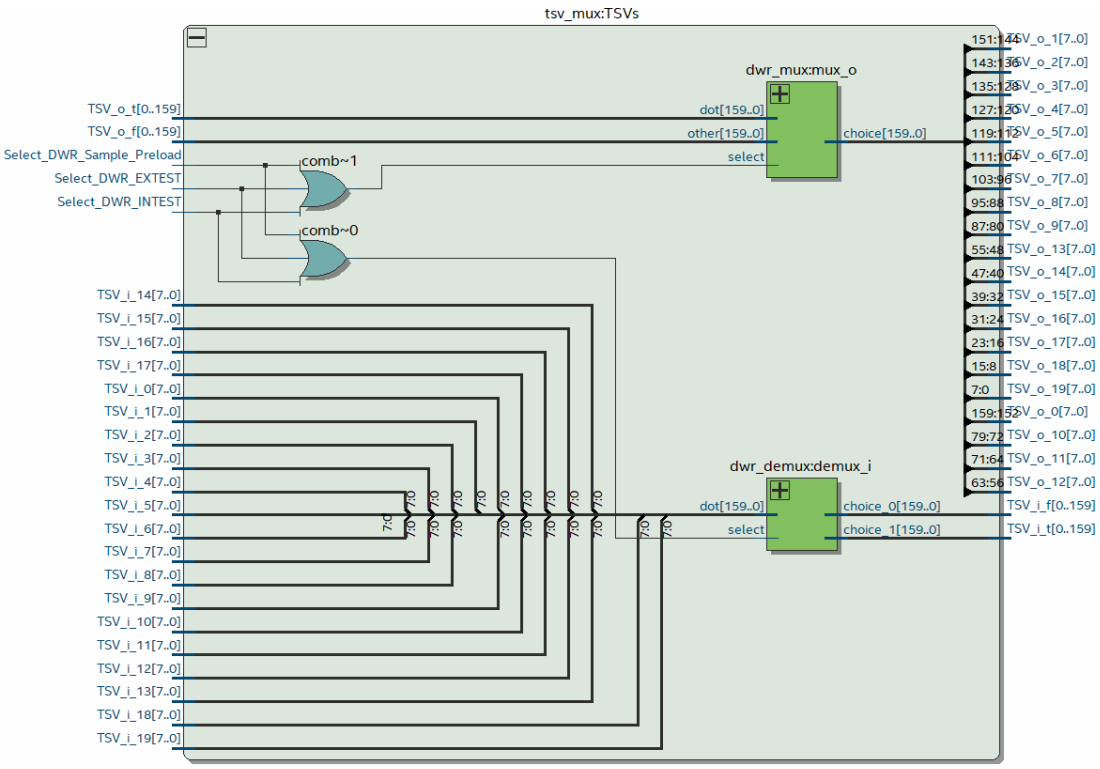



*Figure 6.2.12 The TSV mux module.*

As the testing mode of the circuit shares the same TSVs as its functional mode, multiple mux modules border the TSVs (fig. 6.2.12), in order to exchange the two sets of signals. This switch is controlled by some of the control signals, namely the three DWR ones that require the TSVs for their execution.

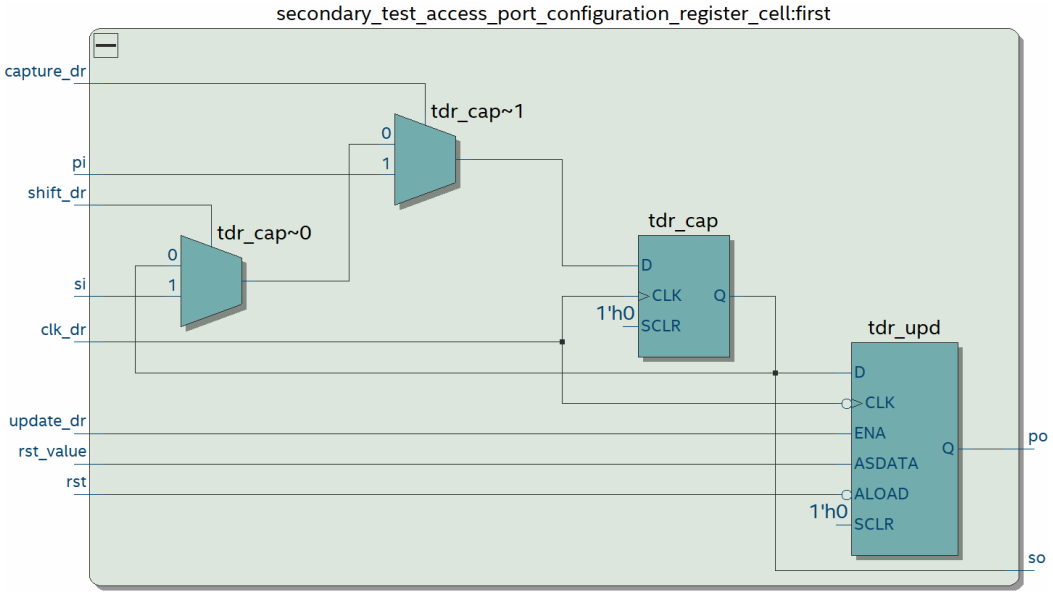



*Figure 6.2.13 The register cell module.*

A simple construct, the register cell (fig. 6.2.13) is made from the Capture and the Update register, controlled by the DR signals Capture, Shift, and Update.

*Table 6.2.1 States of the FSM.*

| | Source State | Destination State | Condition |
|---|---|---|---|
| 1 | capture_dr | shift_dr | (!TMS) |
| 2 | capture_dr | exit_1_dr | (TMS) |
| 3 | capture_ir | shift_ir | (!TMS) |
| 4 | capture_ir | exit_1_ir | (TMS) |
| 5 | exit_1_dr | update_dr | (TMS) |
| 6 | exit_1_dr | pause_dr | (!TMS) |
| 7 | exit_1_ir | pause_ir | (!TMS) |
| 8 | exit_1_ir | update_ir | (TMS) |
| 9 | exit_2_dr | update_dr | (TMS) |
| 10 | exit_2_dr | shift_dr | (!TMS) |
| 11 | exit_2_ir | update_ir | (TMS) |
| 12 | exit_2_ir | shift_ir | (!TMS) |
| 13 | pause_dr | exit_2_dr | (TMS) |
| 14 | pause_dr | pause_dr | (!TMS) |
| 15 | pause_ir | pause_ir | (!TMS) |
| 16 | pause_ir | exit_2_ir | (TMS) |
| 17 | run_test_idle | select_dr_scan | (TMS) |
| 18 | run_test_idle | run_test_idle | (!TMS) |
| 19 | select_dr_s... | select_ir_scan | (TMS) |
| 20 | select_dr_s... | capture_dr | (!TMS) |
| 21 | select_ir_sc... | test_logic_reset | (TMS) |
| 22 | select_ir_sc... | capture_ir | (!TMS) |
| 23 | shift_dr | shift_dr | (!TMS) |
| 24 | shift_dr | exit_1_dr | (TMS) |
| 25 | shift_ir | shift_ir | (!TMS) |
| 26 | shift_ir | exit_1_ir | (TMS) |
| 27 | test_logic_r... | test_logic_reset | (TMS) |
| 28 | test_logic_r... | run_test_idle | (!TMS) |
| 29 | update_dr | select_dr_scan | (TMS) |
| 30 | update_dr | run_test_idle | (!TMS) |
| 31 | update_ir | select_ir_scan | (TMS) |
| 32 | update_ir | run_test_idle | (!TMS) |

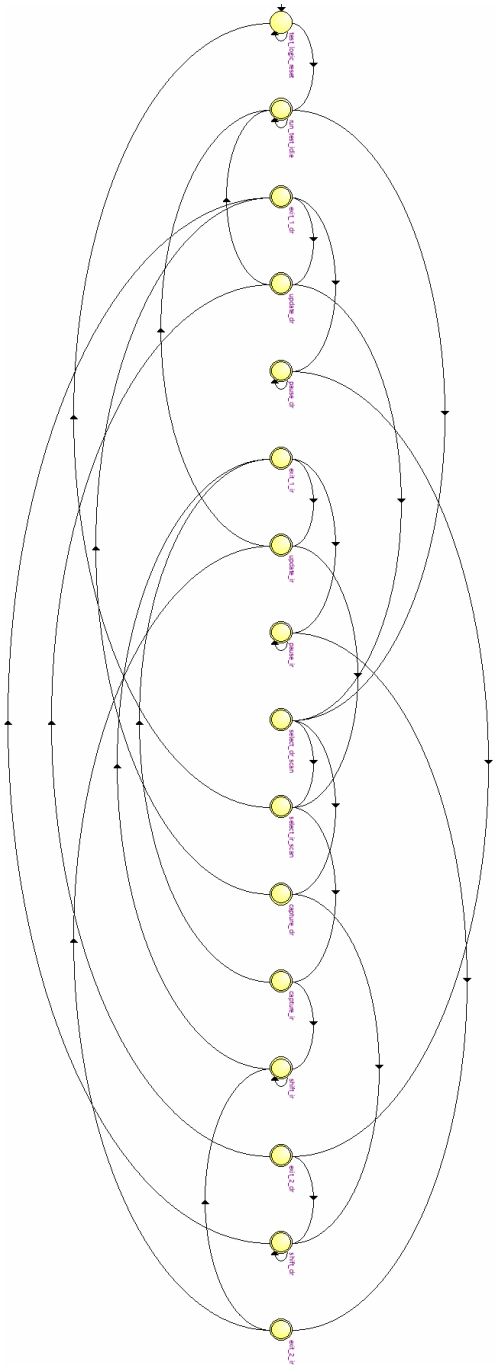


*Figure 6.2.14 States of the FSM.*

As defined in 1149.1 Std. the FSM has 16 states (fig. 6.2.14), split into two tracks, the DR one and the IR one. All state changes happen with the values of the TMS signal.

These were all the modules that exist outside the cores of the die, as defined by the die-centric IEEE 1838 Std. standard. Following on, it is important to understand what takes place in each core, especially with all of the control signals of the various states of the FSM.

## 6.3 Core Modules

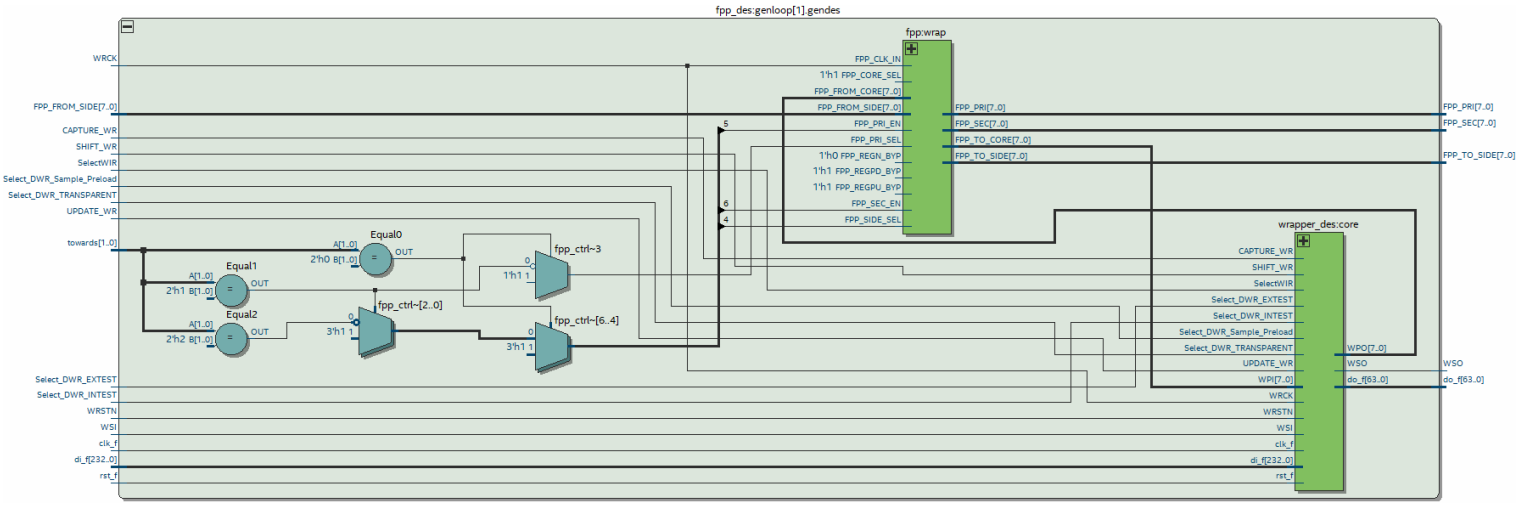



*Figure 6.3.1 A des core.*

In Fig. 6.3.1 one of the *des* cores can be seen, along with the testing outputs of the die. The modules shown are:

- The Flexible Parallel Port (FPP)
- The *wrapper_des*

This is one of the many *des* cores within the die. They all look similar to this, thanks to the *façade cores* which split functional and testing inputs and outputs in a specific way, allowing for the *wrapper cores* to be the same. The *des* core is made out of the FPP module, which is the last piece of the IEEE 1838 Std. and the *wrapper core* which adheres to the IEEE 1500 Std. as well.

As for the few logic gates on the left of the circuit, these are a few control signals. They change the functionality of the FPP module, depending on if it accepts values from another core of the TSVs, and if it gives values to another core, or the TSVs. They are what constitutes the "Towards" value, which is an 8-bit collection of control signals as they are described in the FPP.

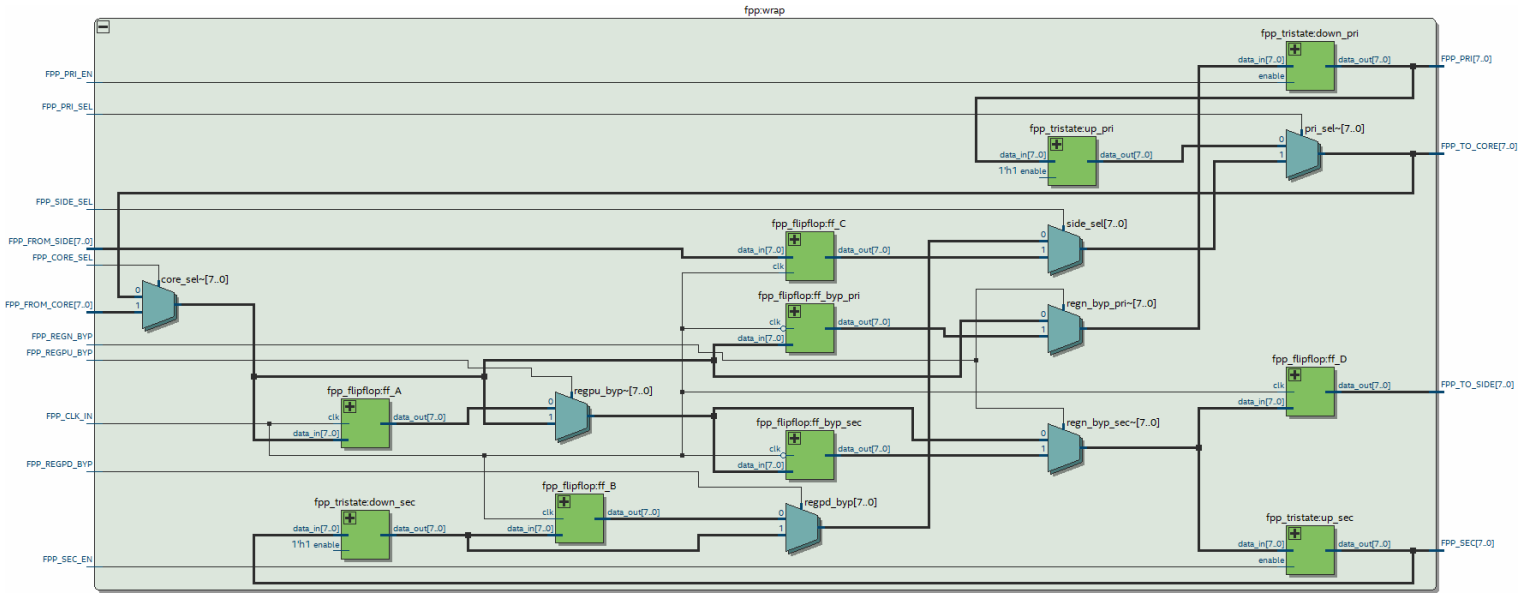



*Figure 6.3.2 The FPP module.*

As defined in the IEEE 1838 Std. the FPP module (fig. 6.3.2) is a particular type of registered switchboard, which has 6 connections. *From* and *To* the *Core*, *From* and *To* the *Side*, and *Pri* and *Sec*, which could have easily been named *From* and *To* the *TSVs*. The reason they are not is because they are defined as bi-directional, but as they are not used as such by the current design, that functionality is beside the current scope of the circuit. At this stage, it is sufficient to say that they are connected to the *wrapper core*, and they can be connected either to the TSVs, or their surrounding cores, in order to create vertical chains across the entirety of the stack.

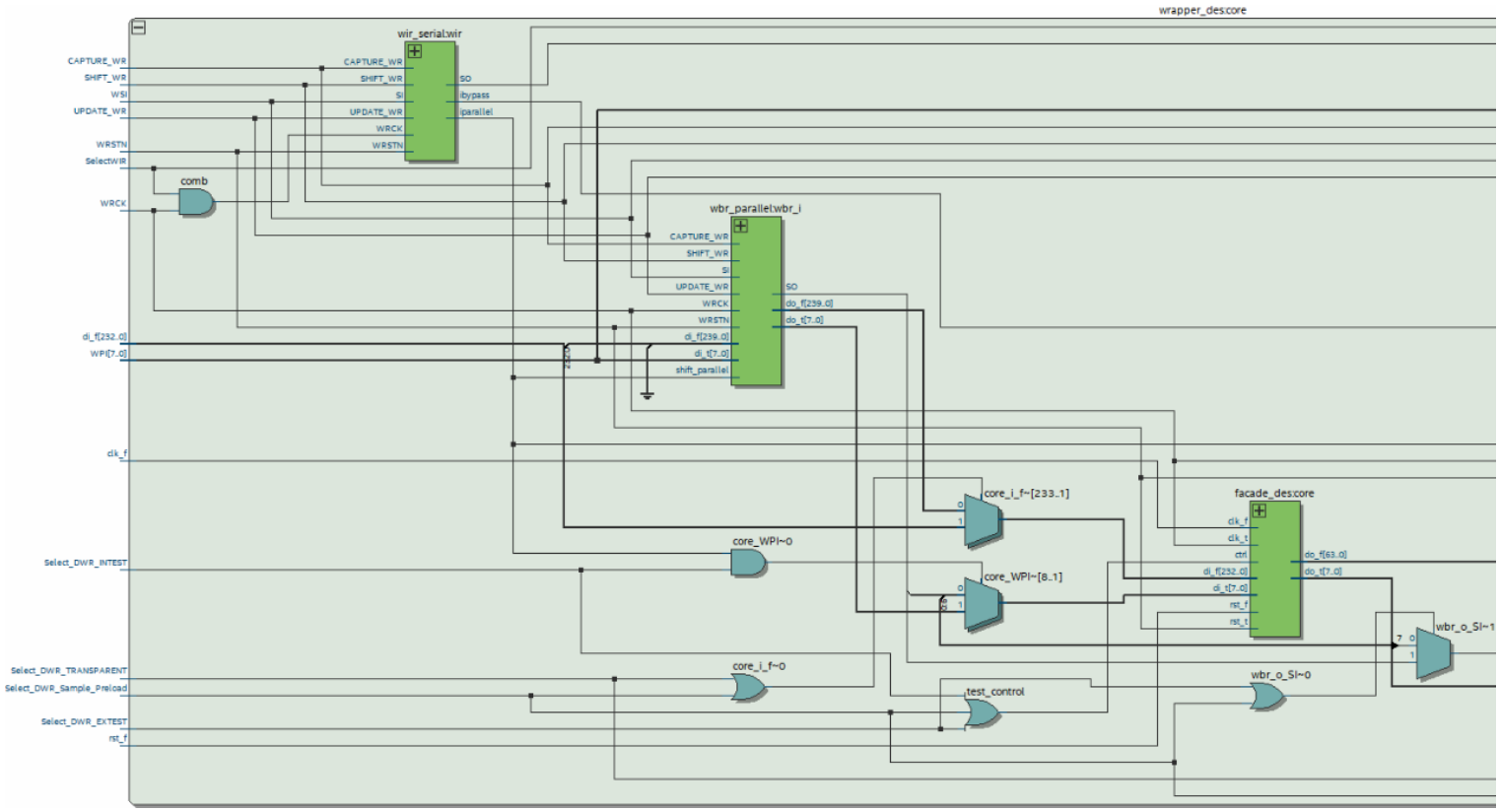



*Figure 6.3.3 Left part of the wrapper core.*

In Fig. 6.3.3 one of the *wrapper cores* can be seen, in particular the part of the *wrapper_des* which is closer to its inputs. The modules shown are:

- The Wrapper Instruction Register (WIR)
- A Wrapper Boundary Register (WBR)
- The *façade_des*

Defined by IEEE 1500 Std. the first thing on the top left of the circuit is the WIR which has to be a part of any core implementing the standard. However, with most of the testing functionality decided by the Instruction Register of the die, little is left for the WIR. For instance, to bypass the core or to make its testing serially (from ports WSI and WSO) or even in parallel though the From and To Core ports of the FPP.

Lower than it, is the WBR for the input ports of the *façade core*, which in turn holds the actual benchmark core *des3*. As for the surrounding logic, it is for the simplification of the control signals, as in, making the WBR transparent on both

the Transparent state, but also the Sample-Preload one of the IEEE 1500 standard.

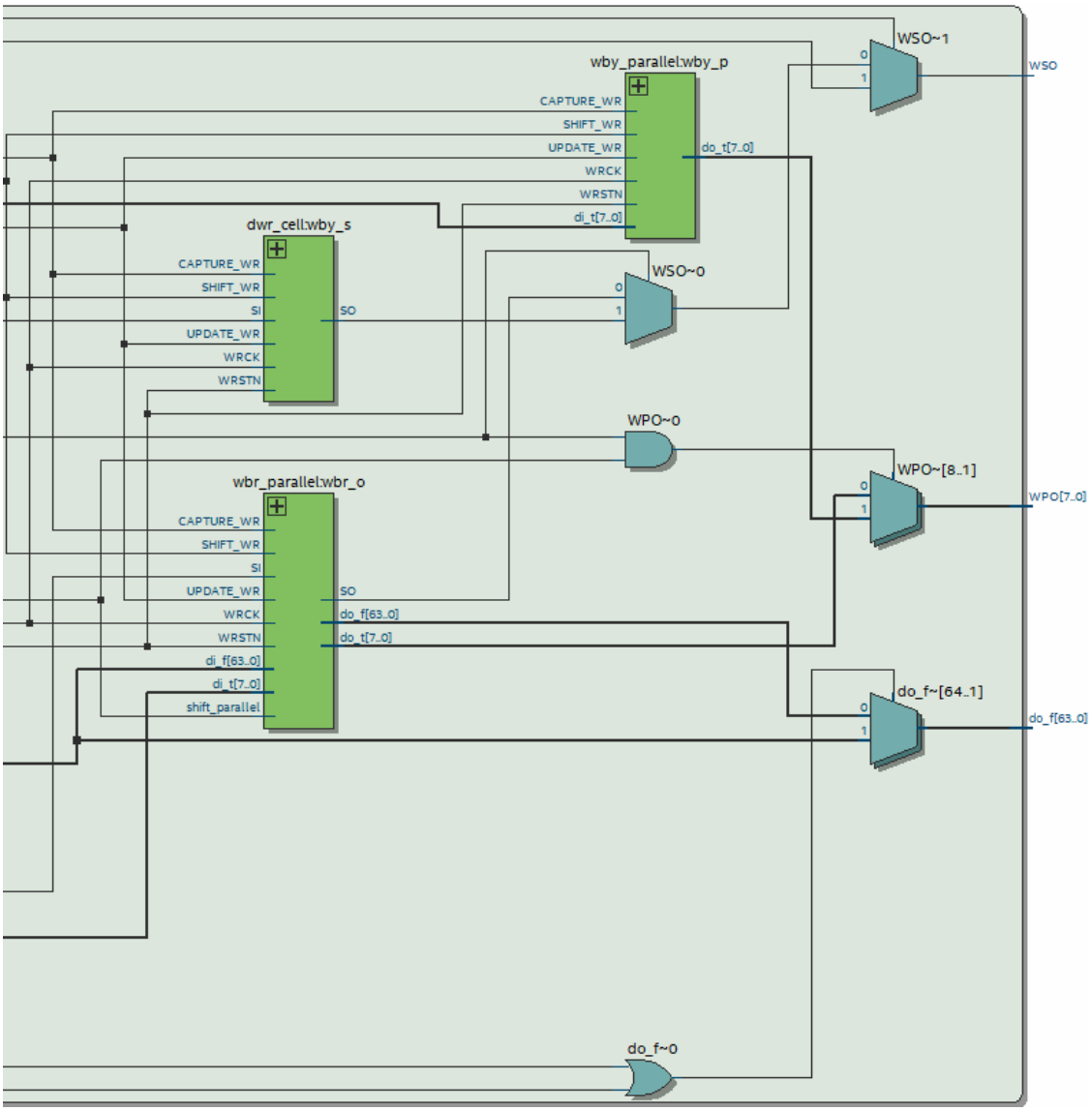



*Figure 6.3.4 Right part of wrapper core.*

In Fig. 6.3.4 the part of the *wrapper_des* which is closer to its outputs can be seen. The modules shown are:

- Two Wrapper Bypass Registers (WBY)
- Another Wrapper Boundary Register

On the second part of the *wrapper core*, there are only a few more WBR modules. One for the output ports of the *façade core*, and one for the parallel bypass of the core. That is important for when there are only some cores that need to be tested through the FPP. Lastly, there is also a bypass for the serial signal,

which is made from a single register cell, as is required by the IEEE 1500 Std. and which is practically the same as the IEEE 1838 Std. register cell.

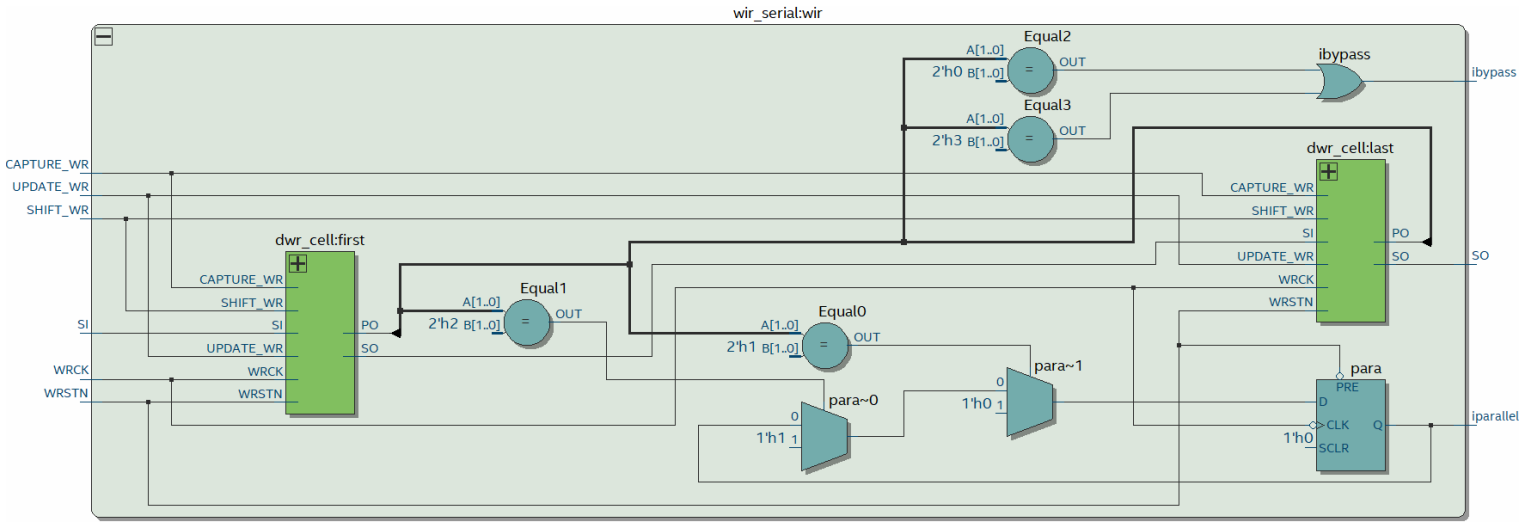



*Figure 6.3.5 The WIR module.*

Much like the Instruction Register of the die, the WIR module (fig. 6.3.5) is made from a minimum of 2 register cells to be IEEE 1500 Std. compatible, and controls only if the core is bypassed during the testing, and if it is tested through the FPP. Meaning that what is defined in the IEEE 1500 Std. as optional in parallel testing is covered in the circuit by the functionality of the IEEE 1838 Std. FPP switchboard.

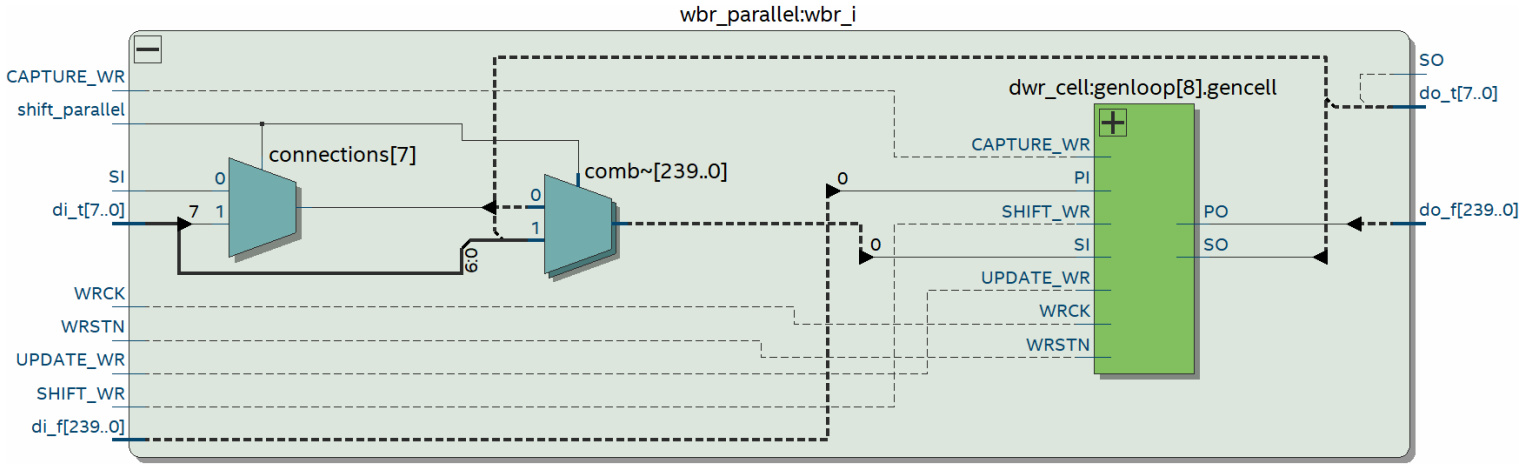



*Figure 6.3.6 The WBR module.*

Made out of IEEE 1500 Std. compatible register cells, the WBR is shown here (fig. 6.3.6) after hiding all DWR cells apart from one, which hides one part of its functionality. The WBR can be shifted 1 value at a time, for when the test is serial,

and its ends are connected to the WSI and WSO ports. However, it can also be shifted 8 values at a time, for when it is connected to the FPP. Additionally, having 8 scan chains within the *façade core* means that, during parallel loading, the scan chains are loaded 1 bit at a time each, instead of linking end-to-front and filling them all in one long sequence.

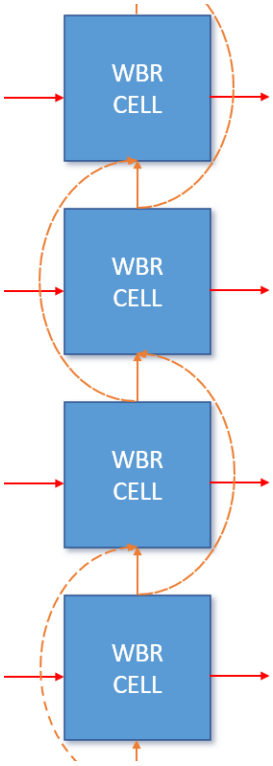



*Figure 6.3.7 Schematic of a WBR with a shift ability of 1 or 2 bits.*

Provided here (fig. 6.3.7) is an example of a WBR which can shift as normal (one bit at a time) but can also skip the following cell to be able to shift two bits at a time. In the circuit, these cells do this for 8 bits at a time, and they are made to be multiples of 8 as well.

However, those up-to-7 cells that get added in order to achieve those multiples of 8 get deactivated when the circuit is in a serial testing mode, so as to not add additional cycles to the filling of the WBR.

Thus, in the end, the cells are able to pass signals from the outside of the *wrapper core* to the *façade core* within, to pass signals in a chain around the *façade core*, but to also shift in intervals of 8.

It is also worth mentioning that when the core is in a functional mode, all this added complexity is done away, as all of the cells around the *façade core* become transparent.

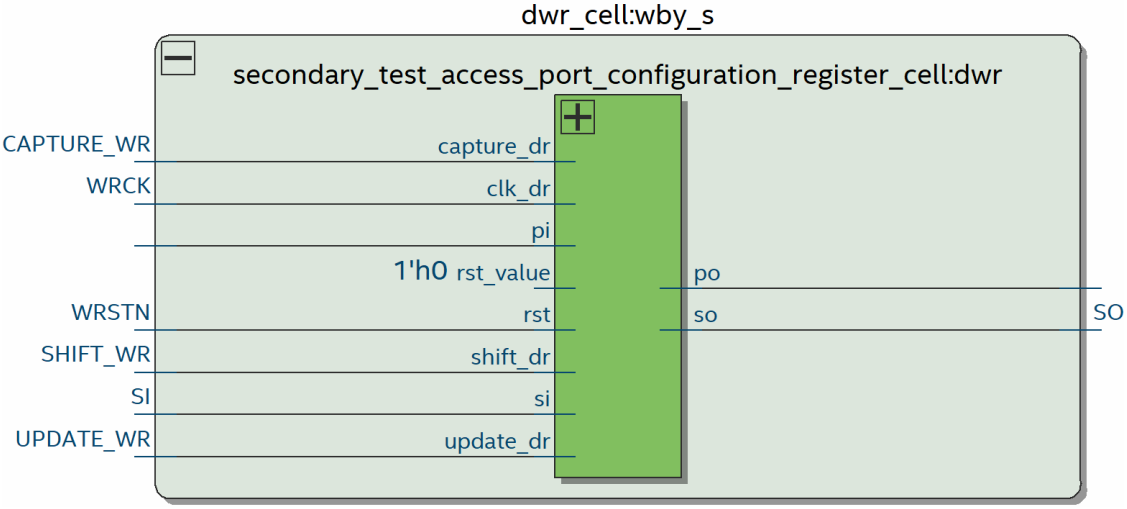



*Figure 6.3.8 The DWR cell module.*

Finally, it is shown (fig. 6.3.8) that the only difference between the register cell defined in IEEE 1838 Std. and the one defined in IEEE 1500 Std. is that the latter one has a set value of the logical zero for its reset value, while the former one allows for the cell to be reset to the logical one. For example, in the third cell of the 3DCR, the standard requires exactly that, for the cell to be reset to the logical value of one.

With this, the entire circuit has been explored with the RTL Viewer, providing a deeper understanding as to what exactly has been achieved; a successful merging of three testing standards, while using the daisy chain technique, for each die in a 3D stack.

# CHAPTER 7.

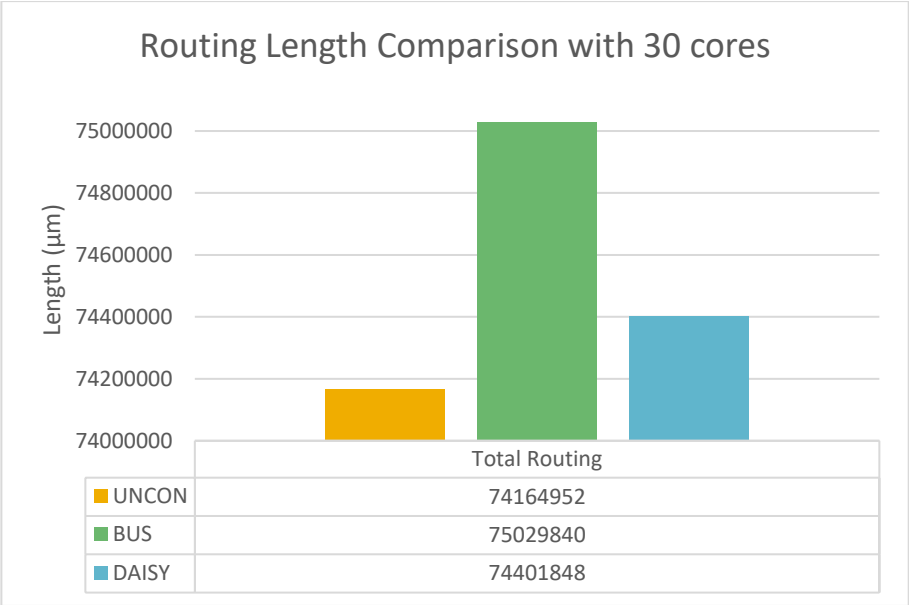
# ANALYSIS



In Chapter 7 we analyse the design presented in prior chapters in respect with its routing, slack histograms, and delay paths.

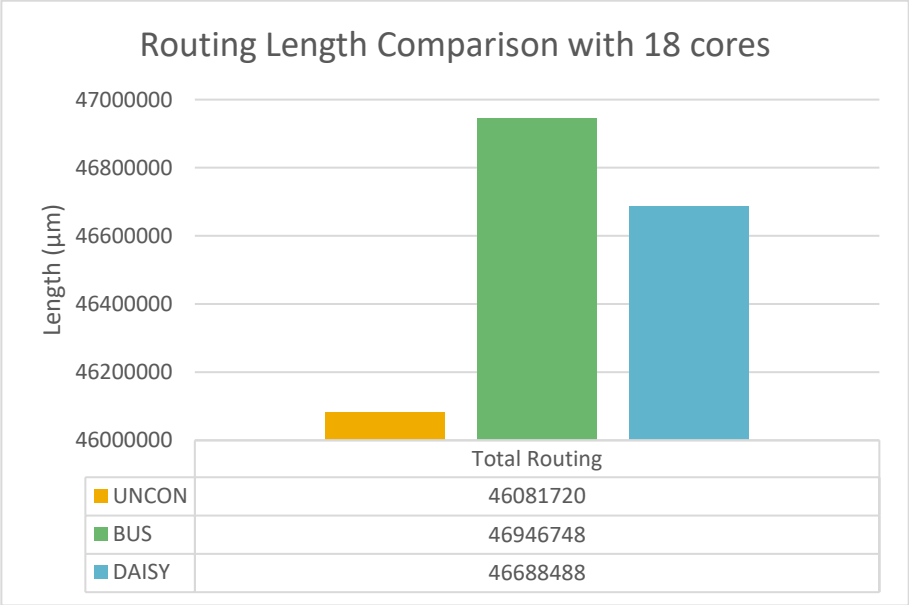
## 7.1 Experimental Results

In the first part of the experiments, we applied PAT A.1 to bound the placement of the 30 cores without the usage of the Method-k3. The cores were placed in a 6×5 grid as it was presented in "Figure 5.2.2 Floorplan of UNCON circuit with the use of bounded cores" and three dies were generated in total: the UNCON die, which has only functional connections between the cores, the DAISY die, which has a TAM architecture consisting of two Daisy Chains, and the BUS die, which has a TAM architecture consisting of two Bus Channels, one connecting the TAM source with the test-inputs of the cores, and one connecting the TAM sink with the test-outputs of the cores. By subtracting the total routing of the UNCON die from the total routing overhead of the DAISY and BUS dies we get an accurate estimation of the respective TAM routing-overhead.



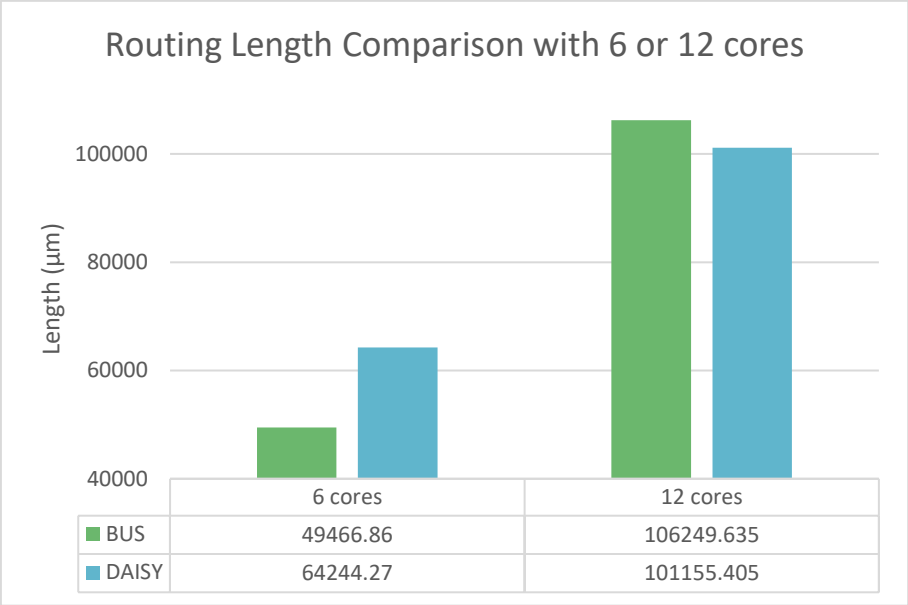

*Graph 7.1.1 Comparison of 30-core circuits UNCON, DAISY, and BUS regarding their respective total routing length.*

The routing length of the first three dies generated is shown in Graph 7.1.1. It is obvious that the routing overhead of the BUS architecture is considerably larger than the routing overhead of the DAISY architecture. This is obviously attributed to the TAM routing-overhead, which is considerably higher of the BUS die than the TAM routing overhead of the DAISY die. Specifically, the BUS scheme requires +265% TAM routing-overhead than the DAISY scheme.




*Graph 7.1.2 Comparison of 18-core circuits UNCON, DAISY, and BUS regarding their respective total routing length.*

In the second experiment we applied Method-k3 by the means of PAT A.2. It generates the same three dies as the previous experiment, but with 18 bounded cores in a 6×3 grid. The routing overhead of the three dies in this case is shown in Graph 7.1.2. In this case, the BUS requires +42.50% additional routing than DAISY for the TAM, even though it was anticipated to be double as much, especially after the application of the Method-k3. The reason for the small gap between the two techniques is the small number of cores in a die. In particular, we found that the more cores exist in the die, the worse the gap between the two techniques becomes.




*Graph 7.1.3 Comparison between the DAISY and BUS circuits, and their 6-core and 12-core variants.*

To support this observation, we performed a third experiment using PATs A.3 and A.4, with 12 and 6 cores respectively. In this particular experiment, instead of measuring the total TAM routing overhead using as baseline the UNCON die, we measured only the routing overhead of the test-data connections of the daisy chain and the bus. The results presented in Graph 7.1.3 show that the DAISY die requires more additional routing for its TAM in the 6-core version of the dies, while requiring slightly less routing in the 12-core version. This is explained by the fact that in its effort to lower congestion, the router produces two very similar results for the 6-core version of the dies. Because of the small number of cores, and the

fact that they can easily be split into the top ones and the bottom ones, the gap between the two techniques not only closed, but inverted.

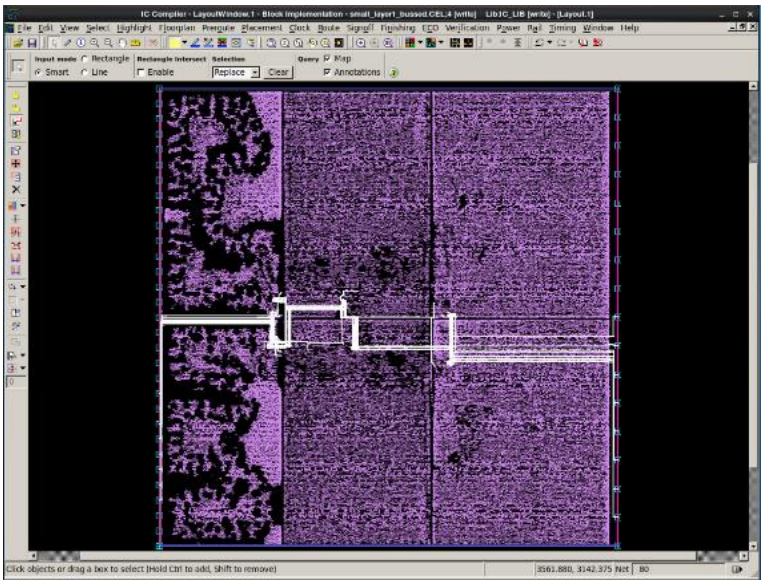


*Figure 7.1.1 Floorplan of 6-core BUS circuit.*

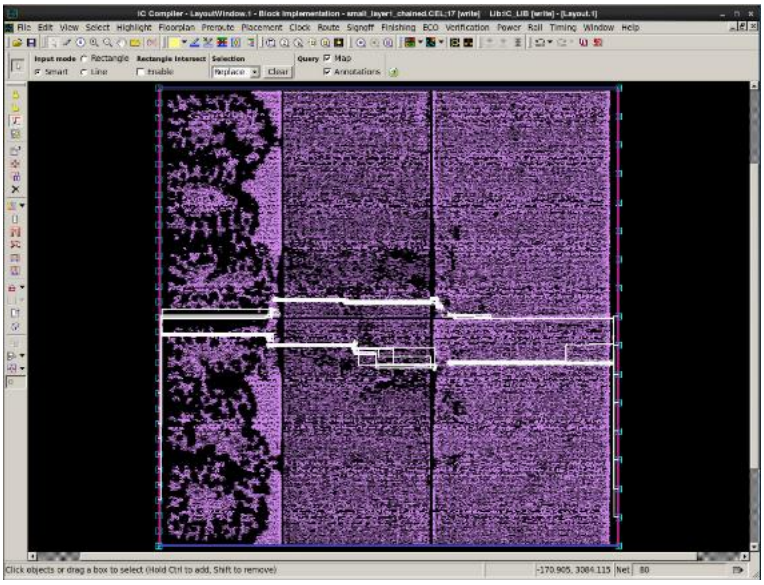


*Figure 7.1.2 Floorplan of 6-core DAISY circuit.*

In Fig. 7.1.1 and Fig. 7.1.2 we present the 6-core BUS die and the 6-core DAISY die respectively, with their TAM signals highlighted. It is obvious that the actual image of the two techniques is very different from the theoretical shapes explored in Chapter 4. They look closer to how they are presented here, with the pins of the cores leaning towards the centre of the die. That is the reason why the bus connected technique (fig. 7.1.3) produces better results than the daisy chain connected one (fig. 7.1.4).

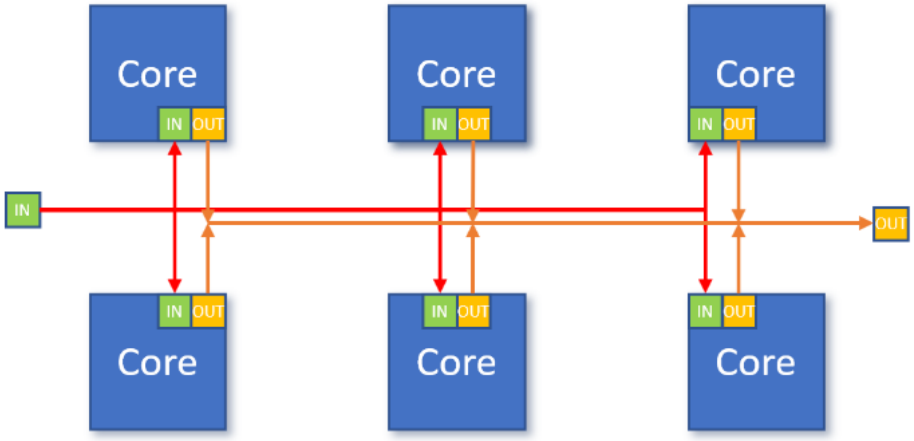



*Figure 7.1.3 Schematic of actual 6-core BUS shape.*

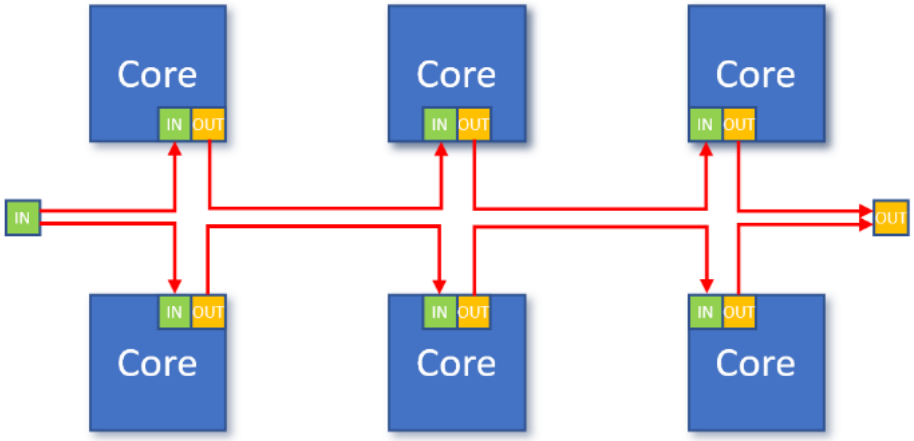



*Figure 7.1.4 Schematic of actual 6-core DAISY shape.*

However, in the general case the cores are not expected to be few and aligned as in Figs. 7.1.1 and 7.1.2, and so the results produced by the two techniques return to normal once the cores are doubled. It happens as soon as the cores can no longer be split by one main axis, to the top and the bottom cores. Additionally, as soon as the bus has to split repeatedly to reach all of the cores in BUS, it can no longer achieve a route length shorter than the DAISY die.

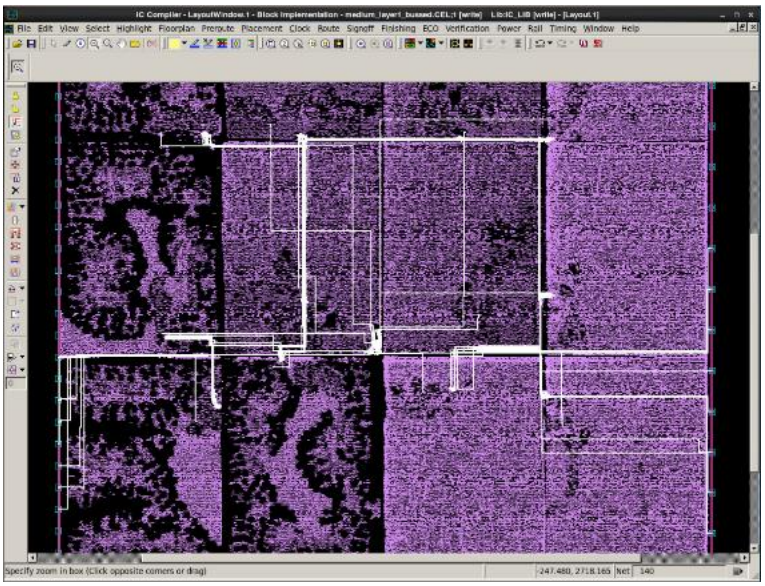


*Figure 7.1.5 Floorplan of 12-core BUS circuit.*

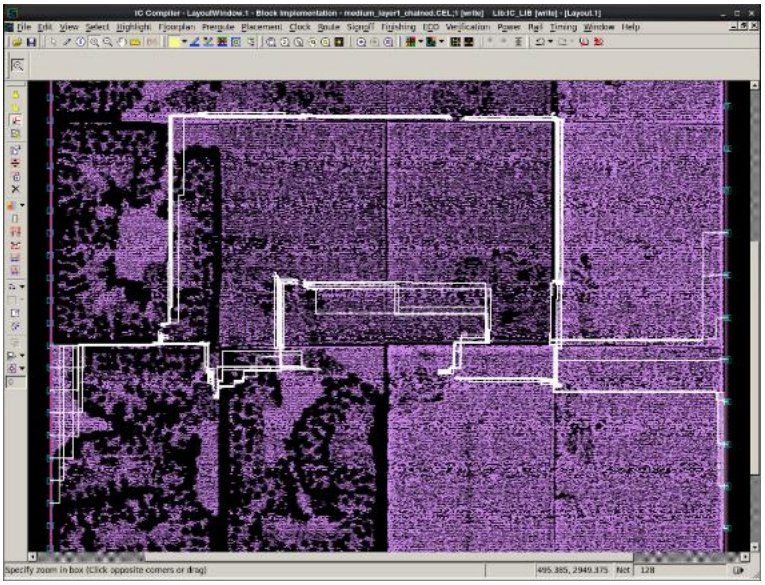


*Figure 7.1.6 Floorplan of 12-core DAISY circuit.*

This is shown in Fig. 7.1.5 for BUS and Fig. 7.1.6 for DAISY. The two chains can easily be seen in the DAISY die with 12 cores, something that cannot be said for the BUS die, which no longer looks like it has one main bus. Instead, it presents as congested wires with no clear structure other than spreading from the left and right edges of the die, at its centre, and outwards to all of the 12 cores on the die.

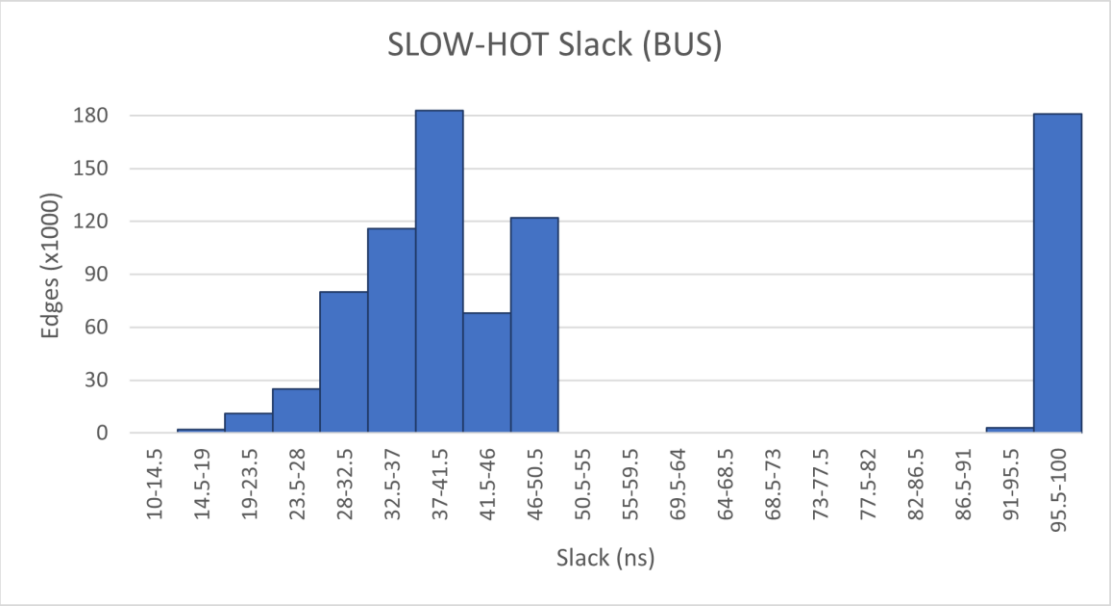
## 7.2 Timing Analysis

For the timing analysis of the Large Floorplan, the cores were simplified. The *façade cores* connected their functional and test inputs to their corresponding outputs. That was done because the functional timing of the benchmark cores is not under consideration. Thus, only the testing mechanism of the die remained, in order for it to be measured.

The TimeQuest Timing Analyzer tool creates a slack histogram for one of four functioning corners of the circuit. These are a combination of the temperature of the circuit, and the speed of its inner silicon because of process variations. More specifically, the four corners are Slow-Hot, Slow-Cold, Fast-Hot, and Fast-Cold, where *hot* is 80 degrees Celsius, and *cold* is 0 degrees Celsius.

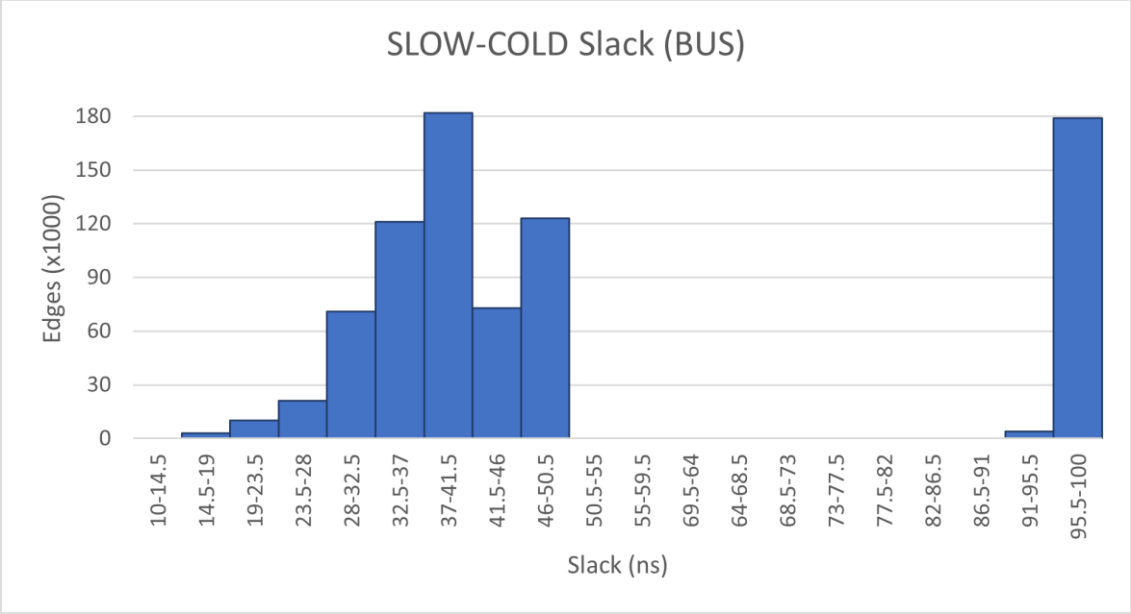
### 7.2.1 Timing Analysis - BUS

When it comes to the slack histograms produced, a uniform value of 100ns was used for the easier comparison of the graphs, which of course does not change the shape of the histograms. Meaning that, should the value 200ns had been used, all slack values would simply grow by +100ns. All histograms show the amount of routing edges vertically, and the buckets of the timing slack horizontally.



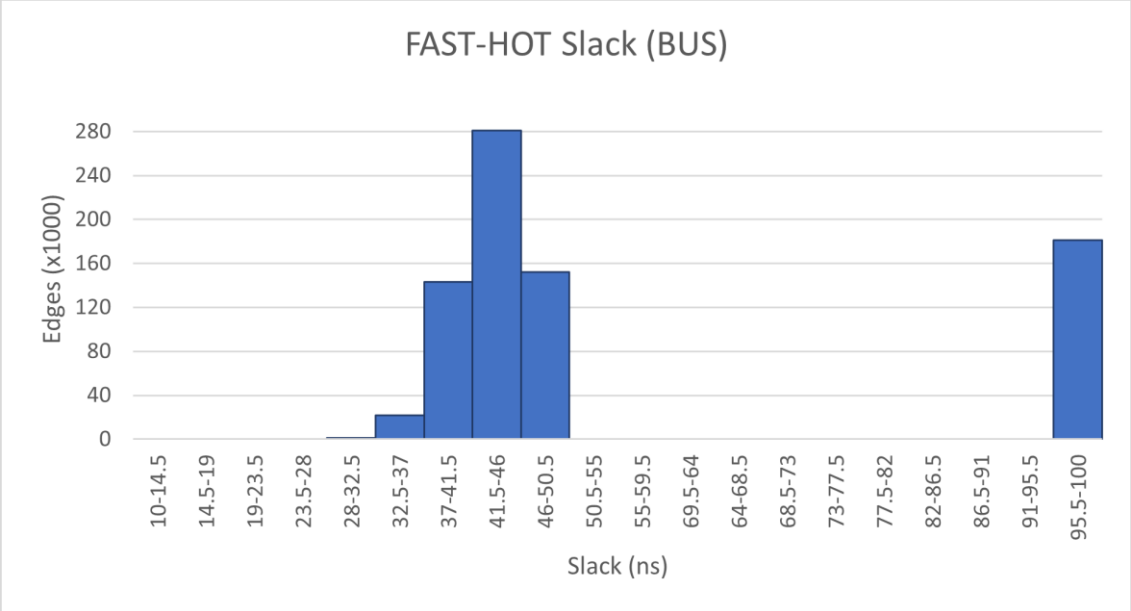

*Graph 7.2.1 Histogram of SLOW – HOT corner slack for the BUS die.*

The histogram clearly shows the inferiority of the BUS architecture as compared to the DAISY one. The slack of the slowest path drops to 14.5ns, which is justified by the slow bus connections.



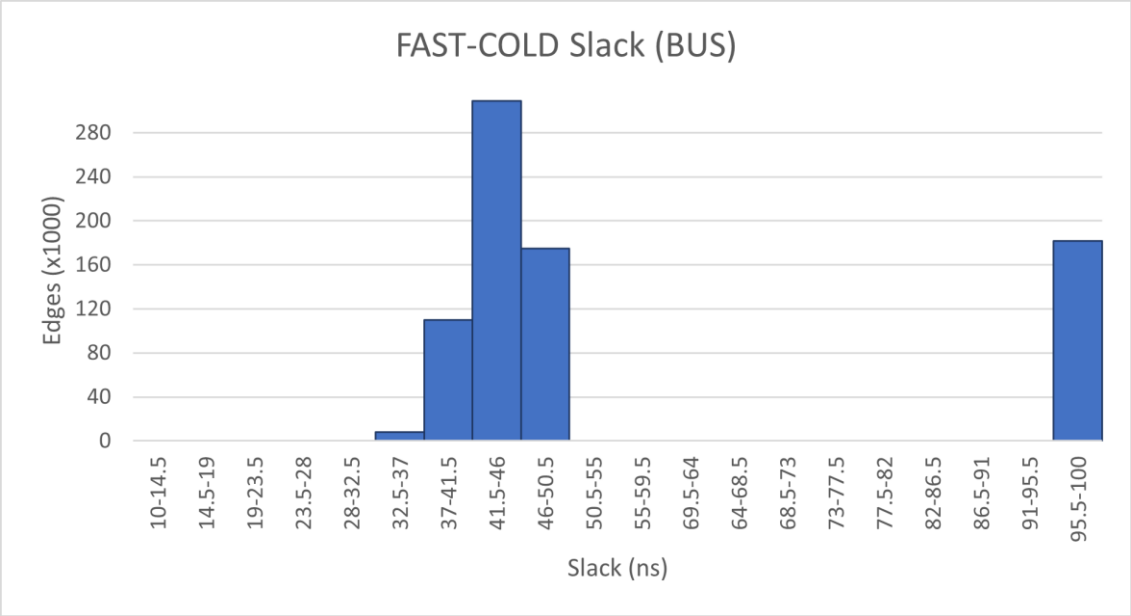

*Graph 7.2.2 Histogram of SLOW – COLD corner slack for the BUS die.*

Even when the circuit is cooled down, its slow version is very similar. Where there were more paths between 28ns and 32.5ns than 41.5ns and 46ns, now this is inverted. However, the same number of paths remain after 14.5ns, meaning that once again it is the buses who are holding the circuit at a slow shift frequency.



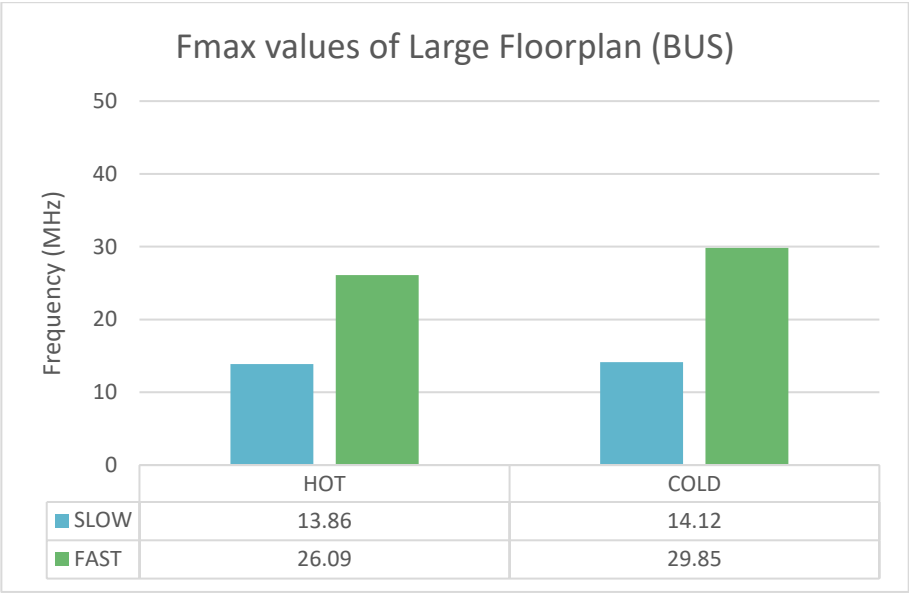

*Graph 7.2.3 Histogram of FAST – HOT corner slack for the BUS die.*

When the circuit achieves the best possible speed with the FAST timing model, the histograms are still not good enough when compared to the DAISY architecture. The histogram shows a small number of connections, at 20000 edges, needing between 32.5ns and 37ns of slack.




*Graph 7.2.4 Histogram of FAST – COLD corner slack for the BUS die.*

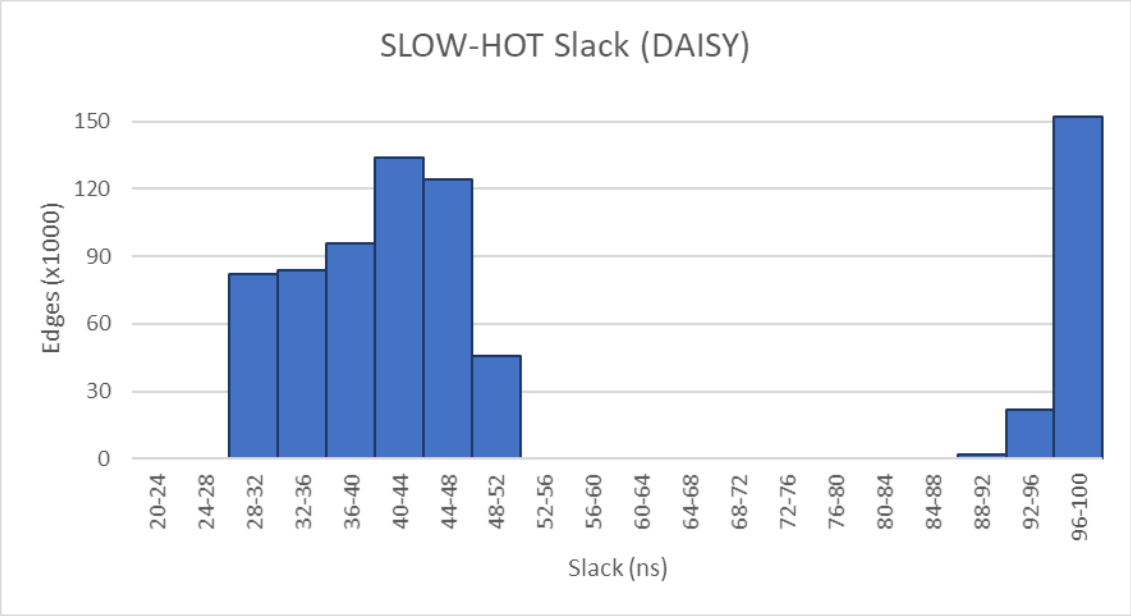
Finally, with the circuit functioning at its absolute best, there still remain edges in the 32.5ns – 37ns bucket. Which means that, once again, it is the buses that do not allow the circuit to surpass the speed of the daisy chained version of itself, which can be seen more easily with summarizing the "Fmax" values achieved.




*Graph 7.2.5 Comparison between Fmax values among all BUS corners.*

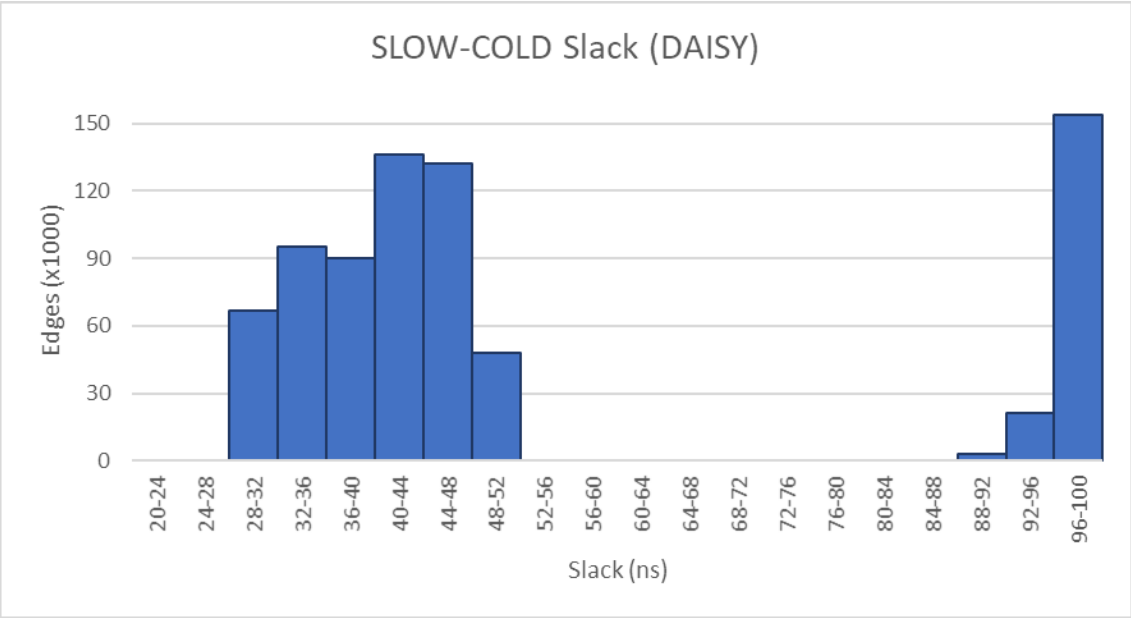
### 7.2.2 Timing Analysis - DAISY

After the results of Section 7.1, it is known that the BUS die achieves worse routing than the DAISY die. But it must still be shown that the DAISY architecture is faster than the BUS one. To this end, the timing analysis of the same design corners is provided.



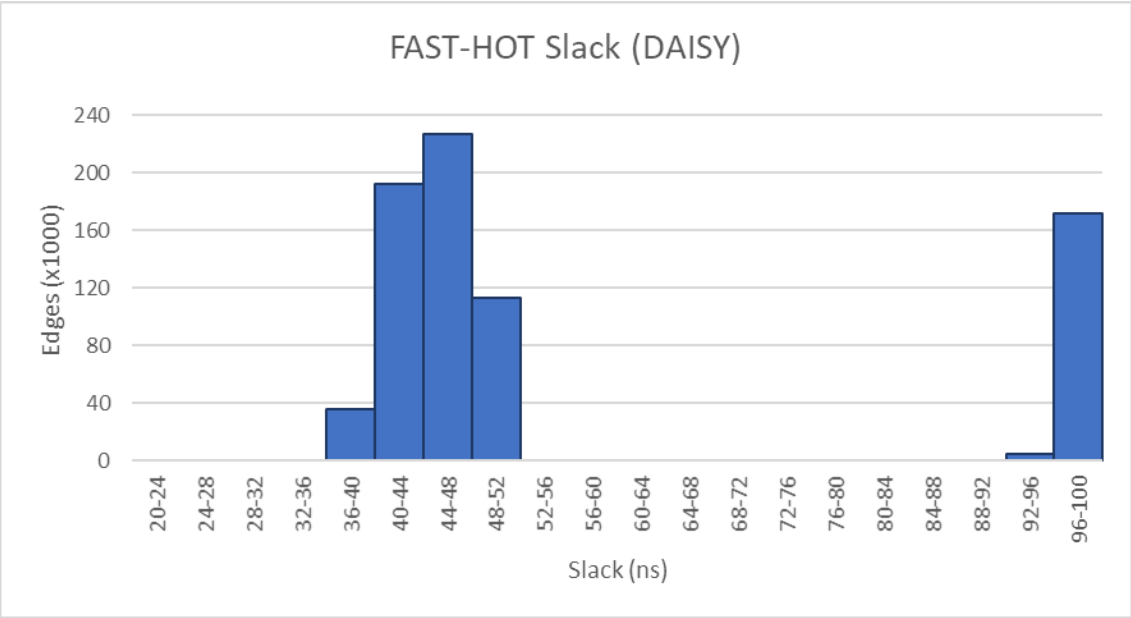

*Graph 7.2.6 Histogram of SLOW - HOT corner slack for the DAISY die.*

This histogram shows that the lowest slack of about 28ns is provided by nearly 85000 connections. This is an expected result, meaning that there are not only a couple of paths that cause this minimum slack, but a wide array of it. On the other end of the histogram, there are the fast paths that belong to the WBR cells surrounding the cores, which are placed right next to each other, with no logic in between them.




*Graph 7.2.7 Histogram of SLOW - COLD corner slack for the DAISY die.*

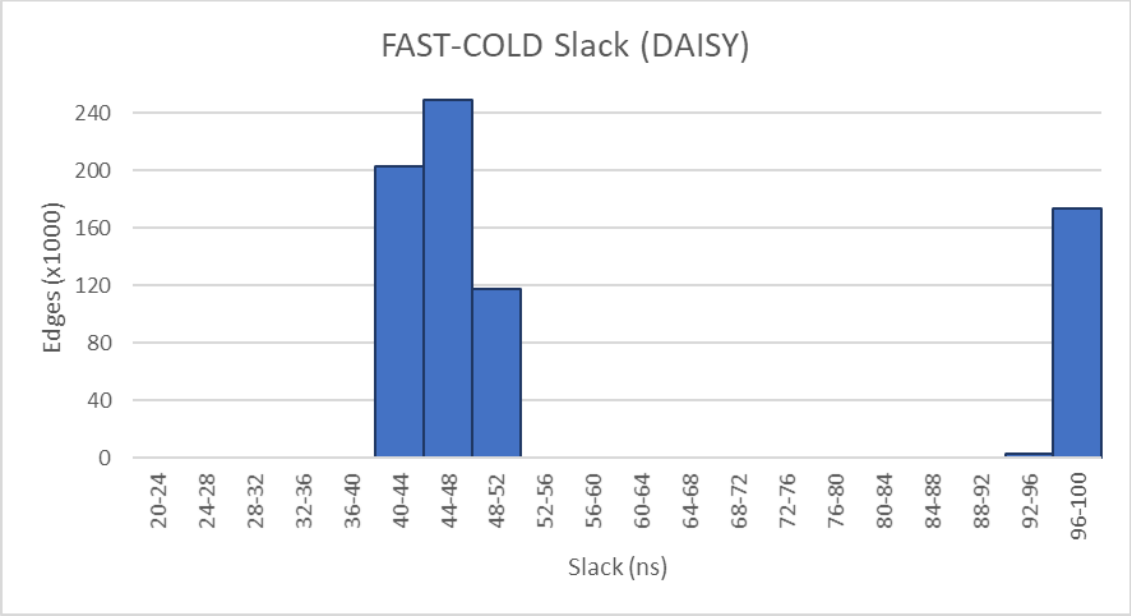
Similarly, with the same variations that cause the slowest silicon, but working under the better, cooler temperature, the minimum slack is near 28ns as well. Once again, the histogram is split into two clumps, showing that the temperature of the circuit only has minor effects in its working speed.




*Graph 7.2.8 Histogram of FAST – HOT corner slack for the DAISY die.*

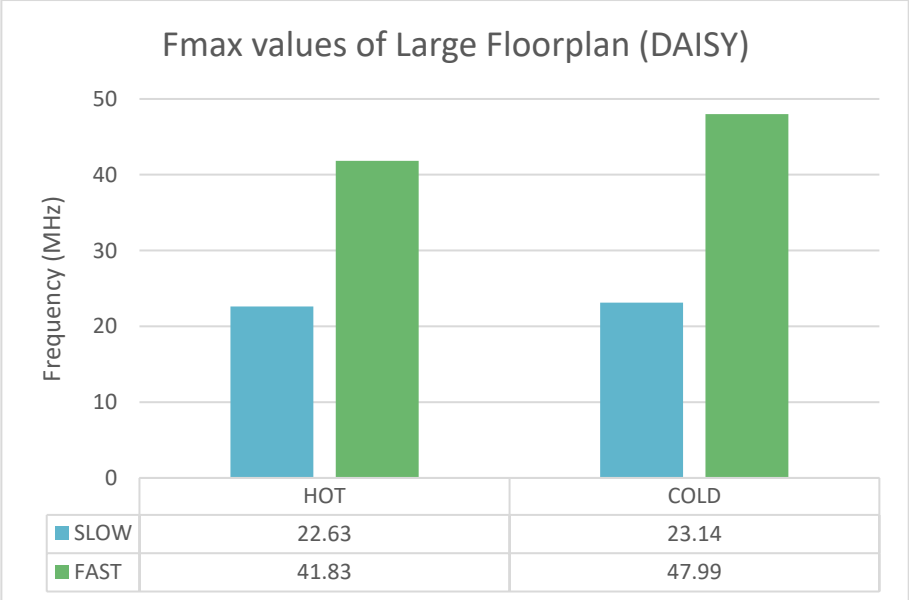
Things change drastically when the process variations have the silicon working at its best possible performance, with 35000 connections have a slack of at least 36ns. However, the two groups of values remain, split at the same point, at 52ns of

slack. Also, it is useful to notice that most connections, 225000 in number fall between 44ns and 48ns, an amount that for the first time surpasses the 170000 connections that are instant.



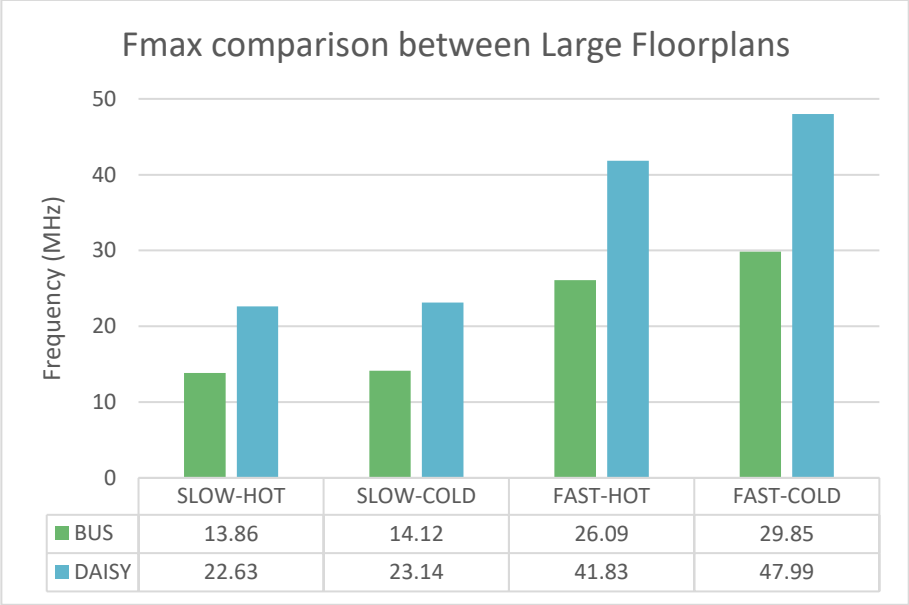

*Graph 7.2.9. Histogram of FAST – COLD corner slack for the DAISY die.*

Lastly, when the circuit is allowed to function at its absolute best, cooled at 0 degrees and with the fastest possible silicon, it achieves a slack of over 40ns. Almost 250000 connections lie between 44ns and 48ns, and the two groups of values have a lot less variation that before.




*Graph 7.2.10 Comparison between Fmax values among all DAISY corners.*

All together, these are the "Fmax" values TimeQuest places on the four functional corners of the circuit, which are the highest frequency each of them can achieve, when factoring in these slack values. For this calculation, the original value of 100ns is ignored by the tool, as it tries to incrementally approach the maximum value for each of the corners of the circuit.




*Graph 7.2.11 Comparison between DAISY and BUS Fmax values on all corners.*

Finally, this is the graph which shows how much better the values of the daisy chain technique are from the bus connected one. At its low end, when the circuit is slow, the DAISY die is better by a 63% - 64%. And at its high end, when the circuit is fast, it is better by 60% - 61%. Which means that, on average, Large Floorplan (DAISY) is 62% faster than Large Floorplan (BUS), settling the comparison between the two techniques.

## 7.3 Delay Paths

Having a general idea of how the Large Floorplan circuit behaves from its corresponding histograms, it is necessary to further investigate the paths that cause the least amount of slack to be available. This is paramount in order to further understand the shape of the circuit and provides a deeper look on why the BUS circuit is inferior to the DAISY one.

### 7.3.1 Delay Paths – BUS

Once again for comparison purposes, it is necessary to examine the BUS version of the Large Floorplan. After its histograms, it is known that it is the worst choice, but there still remains a question as to why this is the case.

*Table 7.3.1 Hierarchy of the beginning of paths of the SLOW (COLD) corner for the BUS die.*

| From Node Hierarchy | Slack | Paths |
|---|---|---|
| large_floorplan:inst | 14.581 | 10 |
| instruction_register:instruction | 14.581 | 10 |
| secondary_test_access_port_configuration_register_cell:first | 14.581 | 6 |
| tdr_upd | 14.581 | 6 |
| secondary_test_access_port_configuration_register_cell:second | 14.756 | 3 |
| tdr_upd | 14.756 | 3 |
| secondary_test_access_port_configuration_register_cell:third | 15.104 | 1 |
| tdr_upd | 15.104 | 1 |

It makes sense that the worst paths of the circuit once again begin from the Instruction Register, as it activates and deactivates the tristate buffers surrounding the two buses, the input and the output one. This is evident by the way the first bit of the instruction has the least amount of slack, unlike the DAISY circuit in which it is absent from the hierarchy.

*Table 7.3.2 Hierarchy of the ending of paths of the SLOW (COLD) corner for the BUS die.*

| To Node Hierarchy | Slack | Paths |
|---|---|---|
| large_floorplan:inst | 14.581 | 10 |
| fpp_des:genloop[*].gendes | 14.581 | 2 |
| fpp_des:genloop[5].gendes | 14.581 | 2 |
| wrapper_des:core | 14.581 | 2 |
| wbr_parallel:wbr_i | 14.581 | 2 |
| fpp_con:genloop[*].gencon | 14.671 | 8 |
| fpp_con:genloop[5].gencon | 14.671 | 4 |
| wrapper_con:core | 14.671 | 4 |
| wbr_parallel:wbr_i | 14.671 | 3 |
| wbr_parallel:wbr_o | 14.820 | 1 |
| fpp_con:genloop[11].gencon | 14.756 | 4 |
| wrapper_con:core | 14.756 | 4 |
| wbr_parallel:wbr_i | 14.756 | 4 |

On the other hand, there are fewer clusters at the end of those worst paths, which belong to the very end of the bused channel, and thus have to go through multiple buffers in order to reach their destination.

*Table 7.3.3 The 10 worst paths of the SLOW (COLD) corner for the BUS die.*

| | Slack | Commands | From | To |
|---|---|---|---|---|
| 1 | 14.581 | report timing | large_floorplani...ll:first\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 2 | 14.671 | report timing | large_floorplani...ll:first\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 3 | 14.756 | report timing | large_floorplani...l:second\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 4 | 14.817 | report timing | large_floorplani...ll:first\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 5 | 14.820 | report timing | large_floorplani...ll:first\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 6 | 14.912 | report timing | large_floorplani...ll:first\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 7 | 14.985 | report timing | large_floorplani...ll:first\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 8 | 15.038 | report timing | large_floorplani...l:second\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 9 | 15.104 | report timing | large_floorplani...ll:third\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 10 | 15.128 | report timing | large_floorplani...l:second\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |

Of course, the bus itself does not change the functionality of the WBR, implemented inside the *wrapper cores* to adhere to the IEEE 1500 Std. Thus, the worst paths of the BUS circuit begin and end with the well-known Update and Capture cells.

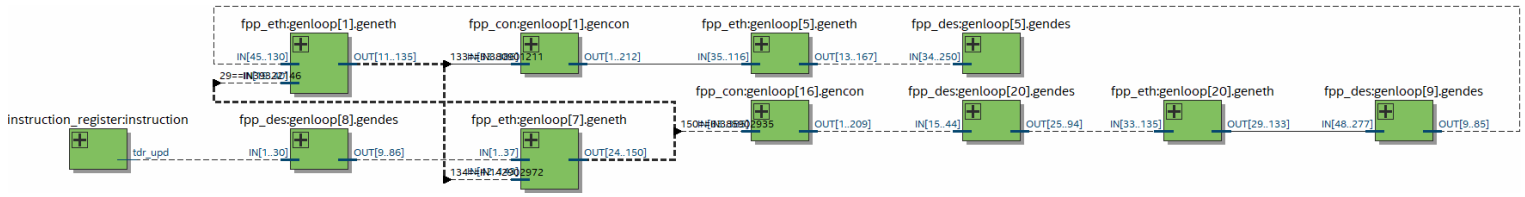



*Figure 7.3.1 Technology map of the worst path of the SLOW (COLD) corner for the BUS die.*

The complexity of the worst path, as shown (fig. 7.3.1), is increased, and for that the sole guilty party is the bus itself. In total, after leaving the Instruction Register, the path goes through 7 different clusters, and 10 different cores. Nothing else must be pointed out here, as it important to see if the same complexity will remain even with the fast variation of the circuit.

*Table 7.3.4 Hierarchy of the beginning of paths of the FAST (COLD) corner for the BUS die.*

| From Node Hierarchy | Slack | Paths |
|---|---|---|
| large_floorplan:inst | 33.247 | 10 |
| instruction_register:instruction | 33.247 | 10 |
| secondary_test_access_port_configuration_register_cell:first | 33.247 | 4 |
| tdr_upd | 33.247 | 4 |
| secondary_test_access_port_configuration_register_cell:second | 33.368 | 3 |
| tdr_upd | 33.368 | 3 |
| secondary_test_access_port_configuration_register_cell:third | 33.414 | 3 |
| tdr_upd | 33.414 | 3 |

Without a change from the SLOW corner, all three bits of the instruction appear in the hierarchy, in the same order as before. It is expected then that the ends of the paths will be similar too.

*Table 7.3.5 Hierarchy of the ending of paths of the FAST (COLD) corner for the BUS die.*

| To Node Hierarchy | Slack | Paths |
|---|---|---|
| large_floorplan:inst | 33.247 | 10 |
| fpp_con:genloop[*].gencon | 33.247 | 10 |
| fpp_con:genloop[11].gencon | 33.247 | 9 |
| wrapper_con:core | 33.247 | 9 |
| wbr_parallel:wbr_i | 33.247 | 7 |
| wbr_parallel:wbr_o | 33.501 | 2 |
| fpp_con:genloop[15].gencon | 33.628 | 1 |
| wrapper_con:core | 33.628 | 1 |
| wbr_parallel:wbr_i | 33.628 | 1 |

Half right, it is easy to notice the return of the 11$^{th}$ cluster, taking nine of the ten worst path endings. Situated around the middle of the bus, this cluster becomes a bottleneck of sorts when process variations push the circuit to be able to work faster.

*Table 7.3.6 The 10 worst paths of the FAST (COLD) corner for the BUS die.*

| | Slack | Commands | From | To |
|---|---|---|---|---|
| 1 | 33.247 | report timing | large_floorplani...ll:first\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 2 | 33.368 | report timing | large_floorplani...l:second\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 3 | 33.401 | report timing | large_floorplani...ll:first\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 4 | 33.414 | report timing | large_floorplani...ll:third\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 5 | 33.485 | report timing | large_floorplani...ll:third\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 6 | 33.501 | report timing | large_floorplani...ll:first\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 7 | 33.600 | report timing | large_floorplani...l:second\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 8 | 33.622 | report timing | large_floorplani...l:second\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 9 | 33.628 | report timing | large_floorplani...ll:third\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 10 | 33.652 | report timing | large_floorplani...ll:first\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |

Like all before it, so too does the FAST corner of the BUS circuit have its worse paths beginning and ending on the same Update and Capture cells. This

makes sense as the bus exists outside the wrapped cores, and thus the endpoints of the paths do not change.

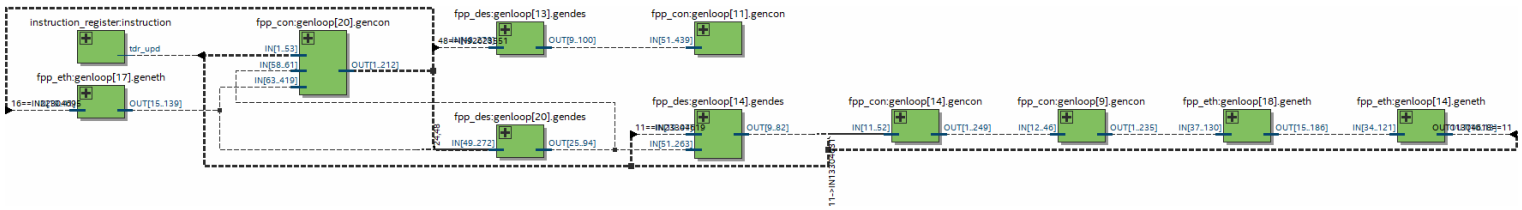



*Figure 7.3.2 Technology map of the worst path of the FAST (COLD) corner for the BUS die.*

In conclusion (fig. 7.3.2), the worst path of the FAST corner of the BUS circuit starts from the Instruction Register, but through the bus it ends up visiting a large part of the total circuit. In total, it goes through 7 clusters, and 10 cores, the same amount as before. This shows that there is no significant change from the process variations of the circuit, but that it is the addition of the bus that lowers the Fmax of the BUS circuit so drastically.

## 7.3.2 Delay Paths – DAISY

Finally, the analysis of the delay paths of the DAISY version of the Large Floorplan is presented here. Through it, the BUS circuit is shown to be lesser than the DAISY circuit in every way possible.

*Table 7.3.7 Hierarchy of the beginning of paths of the SLOW (COLD) corner for the DAISY die.*

| From Node Hierarchy | Slack | Paths |
|---|---|---|
| large_floorplan:inst | 28.389 | 10 |
| instruction_register:instruction | 28.389 | 10 |
| secondary_test_access_port_configuration_register_cell:second | 28.389 | 6 |
| tdr_upd | 28.389 | 6 |
| secondary_test_access_port_configuration_register_cell:third | 28.520 | 4 |
| tdr_upd | 28.520 | 4 |

This shows what is expected from the testing functionality of the circuit, everything beginning by each instruction in the Instruction Register.

*Table 7.3.8 Hierarchy of the ending of paths of the SLOW (COLD) corner for the DAISY die.*

| To Node Hierarchy | Slack | Paths |
|---|---|---|
| large_floorplan:inst | 28.389 | 10 |
| fpp_con:genloop[*].gencon | 28.389 | 7 |
| fpp_con:genloop[17].gencon | 28.389 | 3 |
| wrapper_con:core | 28.389 | 3 |
| wbr_parallel:wbr_i | 28.389 | 2 |
| wbr_parallel:wbr_o | 28.716 | 1 |
| fpp_con:genloop[15].gencon | 28.520 | 1 |
| wrapper_con:core | 28.520 | 1 |
| wbr_parallel:wbr_o | 28.520 | 1 |
| fpp_con:genloop[18].gencon | 28.675 | 1 |
| wrapper_con:core | 28.675 | 1 |
| wbr_parallel:wbr_o | 28.675 | 1 |
| fpp_con:genloop[13].gencon | 28.711 | 2 |
| wrapper_con:core | 28.711 | 2 |
| wbr_parallel:wbr_i | 28.711 | 1 |
| wbr_parallel:wbr_o | 28.712 | 1 |
| fpp_des:genloop[*].gendes | 28.697 | 1 |
| fpp_des:genloop[13].gendes | 28.697 | 1 |
| wrapper_des:core | 28.697 | 1 |
| wbr_parallel:wbr_i | 28.697 | 1 |
| fpp_eth:genloop[*].geneth | 28.743 | 2 |
| fpp_eth:genloop[18].geneth | 28.743 | 1 |
| wrapper_eth:core | 28.743 | 1 |
| wbr_parallel:wbr_o | 28.743 | 1 |
| fpp_eth:genloop[20].geneth | 28.755 | 1 |
| wrapper_eth:core | 28.755 | 1 |
| wbr_parallel:wbr_o | 28.755 | 1 |

On the other hand, the ones requiring the signals in time are the wrappers of the cores, which contain the testing capabilities of each of the cores, including both the IEEE 1868 Std. and IEEE 1500 Std. functionality.

*Table 7.3.9 The 10 worst paths of the SLOW (COLD) corner for the DAISY die.*

| | Slack | Commands | From | To |
|---|---|---|---|---|
| 1 | 28.389 | report timing | large_floorplani...l:second\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 2 | 28.520 | report timing | large_floorplani...ll:third\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 3 | 28.623 | report timing | large_floorplani...ll:third\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 4 | 28.675 | report timing | large_floorplani...ll:third\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 5 | 28.697 | report timing | large_floorplani...l:second\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 6 | 28.711 | report timing | large_floorplani...l:second\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 7 | 28.712 | report timing | large_floorplani...l:second\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 8 | 28.716 | report timing | large_floorplani...l:second\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 9 | 28.743 | report timing | large_floorplani...ll:third\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 10 | 28.755 | report timing | large_floorplani...l:second\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |

More specifically, it can easily be seen that the worst paths begin from the Update cells, and end up in the Capture cells, something that is to be expected from core wrappers that implement IEEE 1500 Std.

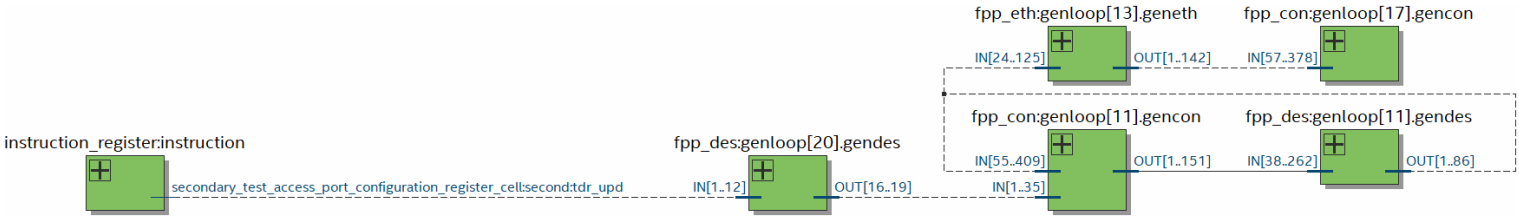



*Figure 7.3.3 Technology map of the worst path of the SLOW (COLD) corner for the DAISY die.*

As expected (fig. 7.3.3), the worst path begins from the Instruction Register, and passes from the des core of the 20th cluster. It goes onto the con and des cores of the 11th cluster, and finally goes through the eth core of the 13th cluster. It then ends up in the con core of the 17th cluster.

*Table 7.3.10 Hierarchy of the beginning of paths of the FAST (COLD) corner for the DAISY die.*

| From Node Hierarchy | Slack | Paths |
|---|---|---|
| large_floorplan:inst | 39.582 | 10 |
| instruction_register:instruction | 39.582 | 10 |
| secondary_test_access_port_configuration_register_cell:third | 39.582 | 5 |
| tdr_upd | 39.582 | 5 |
| secondary_test_access_port_configuration_register_cell:second | 39.605 | 2 |
| tdr_upd | 39.605 | 2 |
| secondary_test_access_port_configuration_register_cell:first | 39.694 | 3 |
| tdr_upd | 39.694 | 3 |

Similar to the SLOW corner, the FAST corner also begins its paths from the Instruction Register, the faster silicon meaning that even the first bit of the instruction creates three paths that have the worst slack.

*Table 7.3.11 Hierarchy of the ending of paths of the FAST (COLD) corner for the DAISY die.*

| To Node Hierarchy | Slack | Paths |
|---|---|---|
| large_floorplan:inst | 39.582 | 10 |
| fpp_con:genloop[*].gencon | 39.582 | 4 |
| fpp_con:genloop[14].gencon | 39.582 | 3 |
| wrapper_con:core | 39.582 | 3 |
| wbr_parallel:wbr_o | 39.582 | 3 |
| fpp_con:genloop[18].gencon | 39.713 | 1 |
| wrapper_con:core | 39.713 | 1 |
| wbr_parallel:wbr_o | 39.713 | 1 |
| fpp_des:genloop[*].gendes | 39.626 | 5 |
| fpp_des:genloop[14].gendes | 39.626 | 2 |
| wrapper_des:core | 39.626 | 2 |
| wbr_parallel:wbr_i | 39.626 | 2 |
| fpp_des:genloop[5].gendes | 39.688 | 2 |
| wrapper_des:core | 39.688 | 2 |
| wbr_parallel:wbr_i | 39.688 | 2 |
| fpp_des:genloop[4].gendes | 39.710 | 1 |
| wrapper_des:core | 39.710 | 1 |
| wbr_parallel:wbr_i | 39.710 | 1 |
| fpp_eth:genloop[*].geneth | 39.703 | 1 |
| fpp_eth:genloop[18].geneth | 39.703 | 1 |
| wrapper_eth:core | 39.703 | 1 |
| wbr_parallel:wbr_o | 39.703 | 1 |

Showing different wrappers than its SLOW counterpart, it is obvious that the FAST corner and all of its variations change the very timing paths within the circuit. The only common cluster is the 18th one, showing up in both hierarchies.

*Table 7.3.12 The 10 worst paths of the FAST (COLD) corner for the DAISY die.*

| | Slack | Commands | From | To |
|---|---|---|---|---|
| 1 | 39.582 | report timing | large_floorplani...ll:third\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 2 | 39.605 | report timing | large_floorplani...l:second\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 3 | 39.626 | report timing | large_floorplani...ll:third\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 4 | 39.649 | report timing | large_floorplani...l:second\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 5 | 39.688 | report timing | large_floorplani...ll:third\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 6 | 39.694 | report timing | large_floorplani...ll:first\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 7 | 39.703 | report timing | large_floorplani...ll:third\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 8 | 39.707 | report timing | large_floorplani...ll:first\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 9 | 39.710 | report timing | large_floorplani...ll:first\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |
| 10 | 39.713 | report timing | large_floorplani...ll:third\|tdr_upd | large_floorplani...cell:dwr\|tdr_cap |

Much like before, the worst paths begin from the Update cells of the wrapper boundary register and end up in the Capture cells of other cores in different clusters.

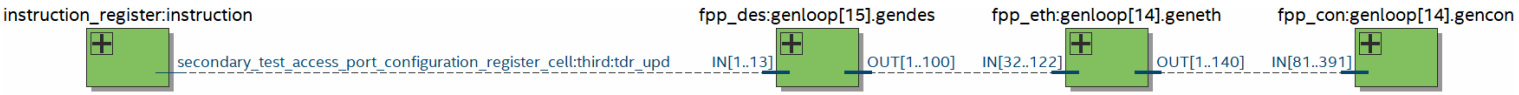



*Figure 7.3.4 Technology map of the worst path of the FAST (COLD) corner for the DAISY die.*

Simpler than before (fig. 7.3.4), the worst path for the FAST corner begins at the Instruction Register, passing from the des core of the 15$^{th}$ cluster, ending up at the con core of the 14$^{th}$ cluster after passing by the nearby eth core. This technology map is further proof that no additional delay has been added to the total circuit, as even the worst path is streamlined to a single channel.

# CHAPTER 8.

## FUTURE WORK

---

With the end of this thesis, some opportunities for further research are born, in order for the continuation of this research, and the usage of its findings into the future:

1. Using the combined standards fully.
    - In order to make an entire stack testable.
    - To simulate and to route it.
    - To analyse its timing, and to measure the effect of the TSVs on it.
2. Finding a better way to chain the cores together.
    - Either with a better algorithm that could work with any number of cores.
    - Or with a heuristic to split the cores in more chains than two.
3. Comparing the routing with other connectivity methods.
    - To find if the daisy chain technique is truly superior.
4. Expanding the use of the combined standards.
    - To find if it is possible to use them with ILV-based solutions.
    - Additionally, to find if they can be used with DfT solutions which have separate testing and functional layers.

The world of 3D ICs is still new, and the future M3D ICs are still being researched. The IEEE 1838 Std. will play a valuable part in this future, especially if

a DfT flow is found which includes it. Finally, such a design flow ought to be flexible enough to work both with TSV-based and ILV-based solutions, and that is the true additional research this thesis suggests.

# BIBLIOGRAPHY


[1] J. Lau, "Evolution, challenge, and outlook of TSV, 3D IC integration and 3D silicon integration," in *Proc. Int. Symp. Adv. Packag. Mater. (APM)*, Xiamen, China, Oct. 2011, pp. 462–488.

[2] R. S. Patti, "Three-dimensional integrated circuits and the future of system-on-chip designs," Proceedings of the IEEE, vol. 94, no. 6, pp. 1214–1224, 2006.

[3] M. Motoyoshi, "Through-silicon via (TSV)," Proceedings of the IEEE, vol. 97, no. 1, pp. 43–48, 2009.

[4] U. Kang *et al.*, "8 Gb 3-D DDR3 DRAM using through-silicon-via technology," *IEEE J. Solid-State Circuits*, vol. 45, no. 1, pp. 111–119, Jan. 2010.

[5] B. S. Feero and P. P. Pande, "Networks-on-chip in a three-dimensional environment: A performance evaluation," *IEEE Trans. Comput.*, vol. 58, no. 1, pp. 32–45, Jan. 2009.

[6] G. H. Loh, Y. Xie, and B. Black, "Processor design in 3D die-stacking technologies," *IEEE Micro*, vol. 27, no. 3, pp. 31–48, May/Jun. 2007.

[7] C. Ababei, P. Maidee, and K. Bazargan, "Exploring potential benefits of 3D FPGA integration," in *Proc. Field Program. Logic Appl.*, Leuven, Belgium, Aug. 2004, pp. 874–880.

[8] Jae-Seok Yang et al. TSV Stress Aware Timing Analysis with Applications to 3D-IC Layout Optimization. In Proceedings ACM/IEEE Design Automation Conference (DAC), pages 803–806, June 2010.

[9] Jae-Seok Yang. Nanometer VLSI Design-Manufacturing Interface for Large Scale Integration. PhD thesis, The University of Texas at Austin, 2011.

[10] Koneru, A., Kannan, S., & Chakrabarty, K. (2017). Impact of electrostatic coupling and wafer-bonding defects on delay testing of monolithic 3D integrated circuits. ACM Journal on Emerging Technologies in Computing Systems (JETC), 13(4), 1-23.

[11] Jung, D. H., Kim, Y., Kim, J. J., Kim, H., Choi, S., Song, Y. H., ... & Orlandi,

A. (2016). Through silicon via (TSV) defect modeling, measurement, and analysis. IEEE transactions on components, packaging and manufacturing technology, 7(1), 138-152.

[12] Gyujei Lee et al. Mechanical characterization of residual stress around TSV through instrumented indentation algorithm. In Proceedings IEEE International Conference on 3D System Integration (3DIC), 2012.

[13] E.J. Marinissen, T. McLaurin and H. Jiao, "IEEE Std P1838: DfT standard-under-development for 2.5 D-, 3D-, and 5.5 D-SICs" In *2016 21st IEEE European Test Symposium (ETS)* (pp. 1-10).

[14] IEEE Standard for Test Access Architecture for Three-Dimensional Stacked Integrated Circuits, in IEEE Std 1838-2019, vol., no., pp.1-73, 13 March 2020.

[15] T. Ni, et. All., "A Novel TDMA-Based Fault Tolerance Technique for the TSVs in 3D-Ics Using Honeycomb Topology", *IEEE Transactions on Emerging Topics in Computing*, 2020

[16] P. Batude et al. "3-D Sequential Integration: A Key Enabling Technology for Heterogeneous Co-Integration of New Function With CMOS.". *IEEE J. Emerging and Selected Topics in Circuits and Sys*. 2, 4 (Dec 2012), 714-722.

[17] A. Koneru, and K. Chakrabarty, "Test and Design-for-Testability Solutions for Monolithic 3D Integrated Circuits" in *ACM Proc. Of the 2019 on Great Lakes Symposium on VLSI*, pp. 457-462.

[18] P. Batude et al. 3-D Sequential Integration: A Key Enabling Technology for Heterogeneous Co-Integration of New Function With CMOS.IEEE JETCAS, 2(4):714–722, 2012.

[19] A. Koneru, S. Kannan, and K. Chakrabarty, "A Design-for-Test Solution Based on Dedicated Test Layers and Test Scheduling for Monolithic 3D Integrated Circuits" *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2018*.

[20] L.-T. Wang, C.-W. Wu, X. Wen, "VLSI Test Principles and Architectures, Design for Testability", Morgan Kaufman, 2006

[21] L.-T. Wang, C. E. Stroud, N. A. Touba, "System-on-Chip Test Architectures

Nanometer Design for Testability", Morgan Kaufman, 2008.

[22] Georgiou Panagiotis, Iakovos Theodosopoulos, and Xrysovalantis Kavousianos. "K3 TAM Optimization for Testing 3D-SoCs using Non-Regular Time-Division-Multiplexing." *2019 IEEE European Test Symposium (ETS)*. IEEE, 2019.

[23] Wang, S., Wang, R., Chakrabarty, K. And Tahoori, M.B., 2018. Multicast Testing of Interposer-Based 2.5 D ICs: Test-Architecture Design and Test Scheduling. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, *23*(3), p.35.

[24] Wang, R. And Chakrabarty, K., 2017. Tackling Test Challenges for Interposer-Based 2.5-D Integrated Circuits. *IEEE Design & Test*, *34*(5), pp.72-79.

[25] Wang R, Chakrabarty K. Testing of Interposer-Based 2.5D Integrated Circuits: Challenges and Solutions. In 2016 IEEE 25th Asian Test Symposium (ATS) 2016 Nov 21 (pp. 74-79). IEEE.

[26] Wang S, Wang R, Chakrabarty K, Tahoori MB. Multicast test architecture and test scheduling for interposer-based 2.5D ICs. In 2016 IEEE 25th Asian Test Symposium (ATS) 2016 Nov 21 (pp. 86-91). IEEE.

[27] Wang R, Li Z, Kannan S, Chakrabarty K. Prebond testing and test-path design for the silicon interposer in 2.5D ICs. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 36, no. 8 (2016), pp. 1406-1419.

[28] Wang R, Chakrabarty K. Testing of interposer-based 2.5D integrated circuits. In 2016 IEEE International Test Conference (ITC) 2016 Nov 15 (pp.1-10). IEEE.

[29] Tahoori, M. And Chakrabarty, K., 2016, November. Test and Reliability Issues in 2.5 D and 3D Integration. In *2016 IEEE 25th Asian Test Symposium (ATS)* (pp. 73-73). IEEE.

[30] Koneru, A., Kannan, S. And Chakrabarty, K., 2017. Impact of Electrostatic Coupling and Wafer-Bonding Defects on Delay Testing of Monolithic 3D Integrated Circuits. *ACM Journal on Emerging Technologies in Computing Sys-*

*tems (JETC)*, *13*(4), p.54.

[31] Park, H., Chang, K., Ku, B.W., Kim, J., Lee, E., Kim, D., Chaudhuri, A., Banerjee, S., Mukhopadhyay, S., Chakrabarty, K. And Lim, S.K., 2019, June. RTL-to-GDS Tool Flow and Design-for-Test Solutions for Monolithic 3D ICs. In *Proceedings of the 56th Annual Design Automation Conference 2019* (p. 101). ACM.

[32] Koneru, A. And Chakrabarty, K., 2019, May. Test and Design-for-Testability Solutions for Monolithic 3D Integrated Circuits. In *Proceedings of the 2019 on Great Lakes Symposium on VLSI* (pp. 457-462). ACM.

[33] Kannan, S., Koneru, A. And Chakrabarty, K., GlobalFoundries Inc and Duke University, 2019. *Testing monolithic three-dimensional integrated circuits*. U.S. Patent Application 15/801,380.

[34] Wang, S., Chakrabarty, K. And Tahoori, M.B., 2018. Defect clustering-aware spare-TSV allocation in 3D ICs for yield enhancement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.

[35] Georgiou, P., Vartziotis, F., Kavousianos, X. And Chakrabarty, K., 2017. Testing 3D-SoCs Using 2-D Time-Division Multiplexing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *37*(12), pp.3177-3185.

[36] Wang, R., Deutsch, S., Agrawal, M. And Chakrabarty, K., 2016, November. The hype, myths, and realities of testing 3D integrated circuits. In *Proceedings of the 35th International Conference on Computer-Aided Design* (p. 58). ACM.

[37] Koneru, A. And Chakrabarty, K., 2018, April. An inter-layer interconnect BIST solution for monolithic 3D ICs. In *2018 IEEE 36th VLSI Test Symposium (VTS)* (pp. 1-6). IEEE.

[38] Koneru, A., Kannan, S. And Chakrabarty, K., 2017, November. A Design-for-Test Solution for Monolithic 3D Integrated Circuits. In *2017 IEEE International Conference on Computer Design (ICCD)* (pp. 685-688). IEEE.

[39] Koneru, A., Kannan, S. And Chakrabarty, K., 2016, October. Impact of wafer-

bonding defects on Monolithic 3D integrated circuits. In *2016 IEEE 25th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)* (pp. 91-94). IEEE.

[40] IEEE Standard for Test Access Port and Boundary-Scan Architecture," in IEEE Std 1149.1-2013 (Revision of IEEE Std 1149.1-2001), vol., no., pp.1-444, 13 May 2013, doi: 10.1109/IEEESTD.2013.6515989.

[41] IEEE Standard Testability Method for Embedded Core-based Integrated Circuits," in IEEE Std 1500-2005, vol., no., pp.1-136, 29 Aug. 2005, doi: 10.1109/ IEEESTD.2005.96465.

[42] IEEE Standard for Test Access Architecture for Three-Dimensional Stacked Integrated Circuits," in IEEE Std 1838-2019, vol., no., pp.1-73, 13 March 2020, doi: 10.1109/IEEESTD.2020. 9036129.

[43] Georgiou, P., Vartziotis, F., Kavousianos, X. and Chakrabarty, K., 2017. Testing 3d-socs using 2-d time-division multiplexing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *37*(12), pp.3177-3185.

[44] F. Vartziotis, X. Kavousianos, K. Chakrabarty, A. Jain, and R. Parekhji, "Time-Division Multiplexing for Testing DVFS-Based SoCs," IEEE Transactions on CAD of Integrated Circuits and Systems, vol. 34, no. 4, pp. 668–681, April 2015.

[45] X. Zhao, D. L. Lewis, H. H. S. Lee, and S. K. Lim, "Low-power clock tree design for pre-bond testing of 3-d stacked ICs," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 30, no. 5, pp. 732–745, May 2011.

[46] Karmarkar, N. and Karp, R.M., 1982. *The differencing method of set partitioning*. Berkeley: Computer Science Division (EECS), University of California.

[47] Kruskal, J.B., 1956. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, *7*(1), pp.48-50.

# APPENDIX A

## DESIGN PROCESS

For the 3D design used in this thesis, particular cores from the IWLS 2005 Benchmarks suit are used: the *des3* core, the *eth_top* core, and the *vga_enh_top* core. Those are wrapped in a custom implementation of the IEEE 1500 Standard, aptly named *wrapper_des*, *wrapper_eth*, and *wrapped_vga* in turn.

*Table A.1 Functional pins of each of the three wrapped cores.*

| | Functional Inputs | Functional Outputs |
|---|---|---|
| wrapper_des | 234 | 64 |
| wrapper_eth | 96 | 115 |
| wrapper_vga | 89 | 109 |

In total, there are 12570 functional connections between the I/O ports of the benchmark cores at each die. These connections originate from the 8610 output ports provided in the same die, as well as a few hundred external die inputs. Despite the large number of functional connections over test data ones, synthesis provides dies that are completely incomparable in terms of their test cost. Apart from their difference in total width and length of the dies, the location of the logic cells comprising the cores shifts so much in the final layout, it is impossible to compare the route lengths of the DAISY and BUS dies. Even the elimination of the functional connections from the wirelength of each die, by subtracting the total routing length of the UNCON die, provides no meaningful results.

To alleviate this problem, the dies were simplified, removing from them the ability to change test data width, along with moving the logic of the test states outside the die. Meaning that the die requires all signals on the die level, even if on the embedded core level it still manages to implement the IEEE 1500 Standard. Thus, the new *simple8_wrapper_[des, eth, vga]* cores were used, along with a customized TMS signal which maps signals and not states (fig. A.1).

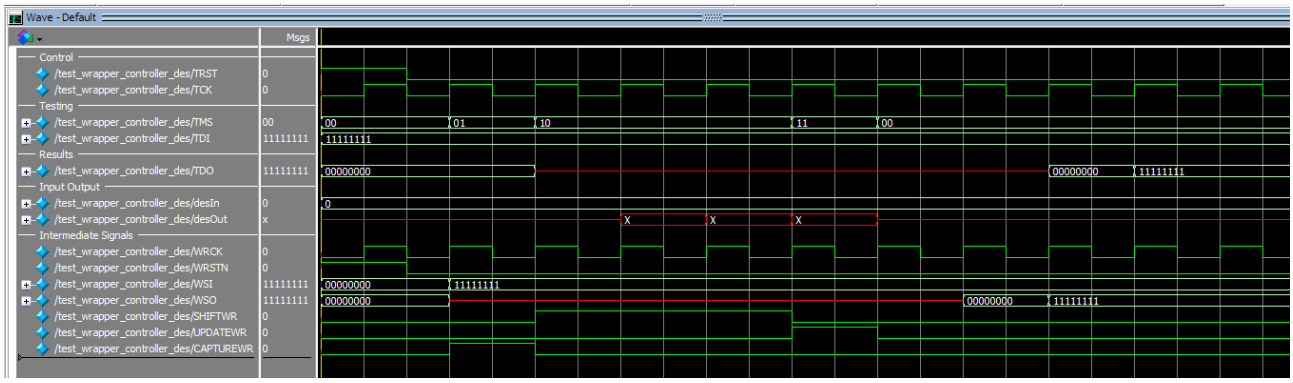



*Figure A.1 Waveform of circuit simple8_wrapper_des.*

In order to provide more comparable layouts, placement constraints were imposed on the cores of the dies through the command named *create_bounds* which ties down the cores in specific locations across the die, as it was presented in "Figure 5.2.2 Floorplan of UNCON circuit with the use of bounded cores".

This was done to limit the relocation of the logic cells during their placement in the three dies as much as possible. Even though this relocation is often exploited by the synthesis process to optimize routing, the modern SoC design style imposes the placement of the cores in the floorplan before any optimization is applied inside each core, which is completely compatible with this process. The cores are initially placed in a manner that fit the two Daisy Chains, by hand. The total routing of the dies increased slightly, echoing the warning of the tool itself: the more bounds a design contains, more congestion is the result of the effort to meet them. This is the main reason, along with the older library used and the age of the tool in question, for the results in "Graph 7.1.1 Comparison of 30-core circuits UNCON, DAISY, and BUS regarding their respective total routing length".

A new method was devised, one that does not require human input for the generation of the TAM. A method able to automate not only those steps, but even the decision of which chain each core must belong to. Meaning that, this is no longer a problem of design alone, but one of optimization too. That is where the K3 TAM optimization method comes into play, by inspiring what was named the Method-k3. However, because of the added bounds, and the complexity of the method, each experiment took an increasing amount of time to complete, and thus the dies were shrunk from 30 cores into 18 cores.

After the unexpected results of "Graph 7.1.2 Comparison of 18-core circuits UNCON, DAISY, and BUS regarding their respective total routing length", further experimentation was required.

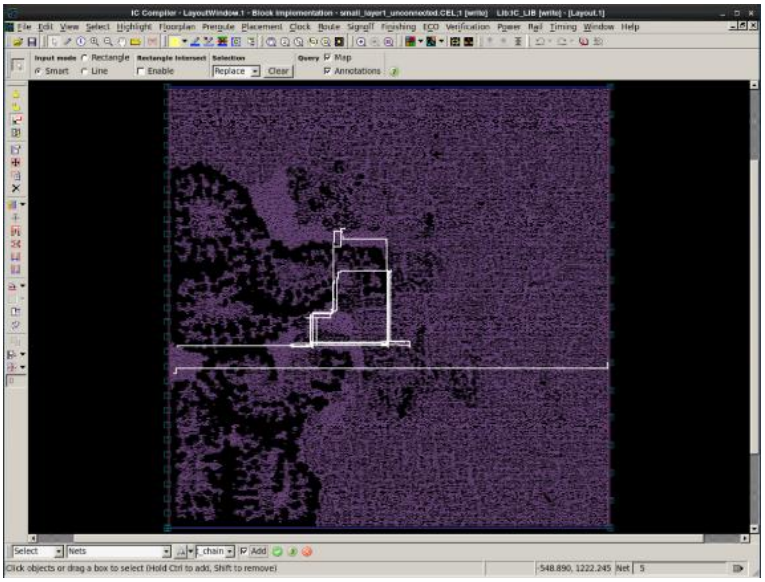


*Figure A.2 Floorplan of UNCON circuit.*

Presented here is the unconnected die (fig. A.2), with only 6 cores, with the signals *Select WIR*, *Shift WR*, *Capture WR*, *Update WR*, and *Select Chain* shown. The first five belong to the IEEE 1500 Standard, while the latter one is the implementation of splitting the controller of the die into two parts. Between the part that serves as the beginning of the two chains, and a small part in the end which allows only one of them as the die's output. That was done so that the chains did not have to return to the leftmost part of the die only to then travel to the rightmost part again, which was part of the issue when compared to the bused die.

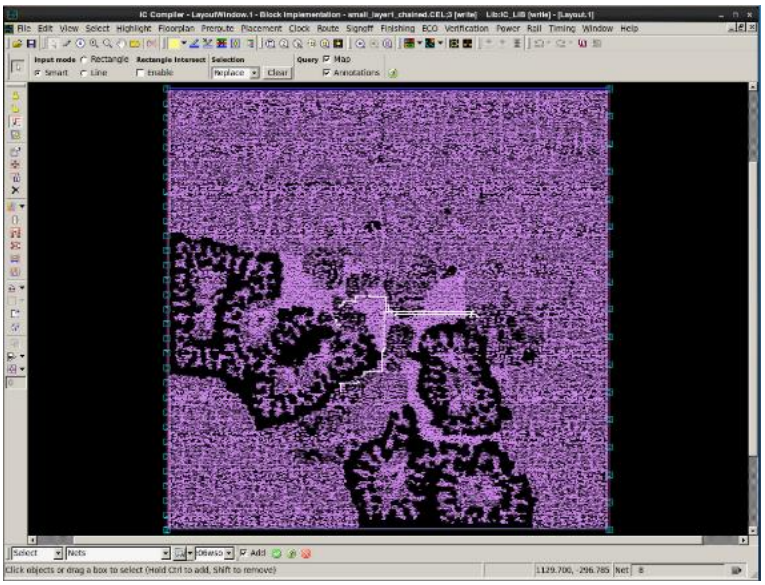


*Figure A.3 Floorplan of DAISY circuit, Pass 1 of Method-k3, serial signals.*

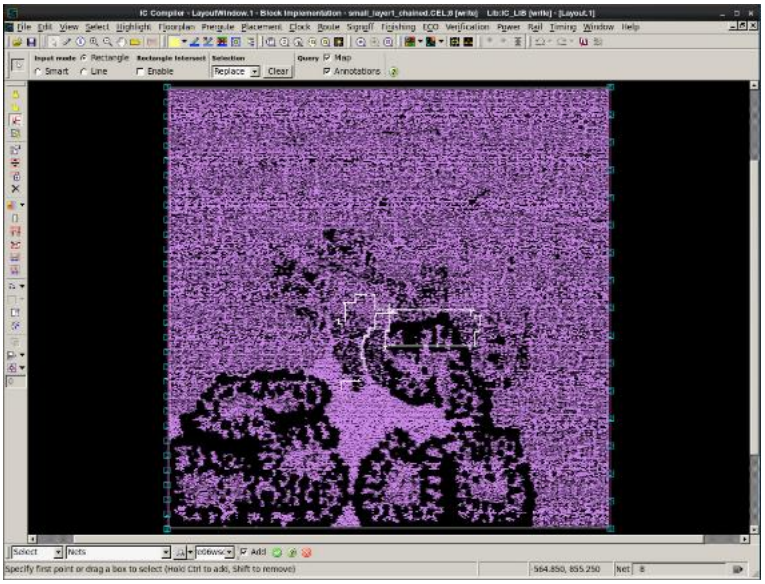


*Figure A.4 Floorplan of DAISY circuit, Pass 2 of Method-k3, serial signals.*

Following, on the die connected with daisy chains after the first pass of the Method-k3, those very chains are shown (fig. A.3). They connect the 6 cores, three at a time, but as there are no bounds in place, they gather around the centre of the die, which further explains the smaller difference in routing.

The same is shown on the die produced after the second phase of the Method-k3 (fig. A.4), the same signals connecting the 6 cores in a chain. Their difference is infinitesimal, because once again they go around the centre, the cores of the die arranged around like petals, which is not particularly good for the checking of the two techniques.

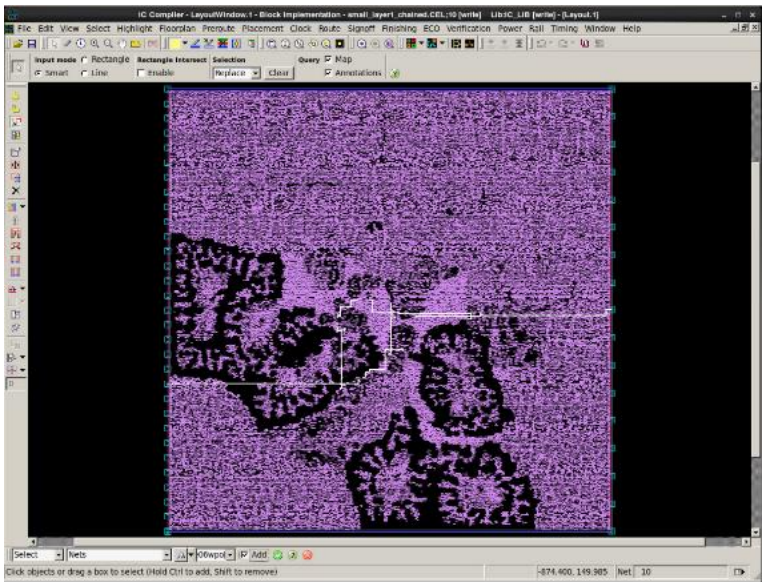


*Figure A.5 Floorplan of DAISY circuit, Pass 1 of Method-k3, parallel signals.*

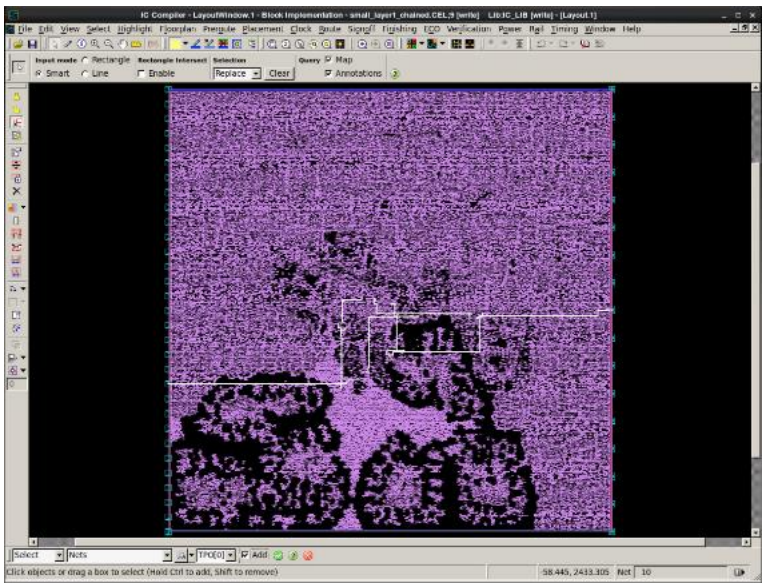


*Figure A.6 Floorplan of DAISY circuit, Pass 2 of Method-k3, parallel signals.*

To show the connections better, it was deemed necessary to change the signals presented. Instead of the serial test inputs of the IEEE 1500 Standard, the parallel ones were measured separately, or more specifically, the zeroth bit of them – as they were eight in total, and would over complicate both the images produced, and their measurement. Thus, the signals *TPI[0]*, *WPI[0]*, *TPO[0]*, *WPO[0]*, and the *Connection[0]* are shown (fig. A.5). Of those, the first four are defined in the IEEE 1500 Standard, while the fifth one is the name of the inner connections of the parallel test data linking the cores into the two chains.

Of course, it stands to reason that two measurements are produced, one for each of the passes of the Method-k3 (fig. A.6). Shown here is the same floorplan, of the DAISY circuit, after the second pass of the Method-k3.

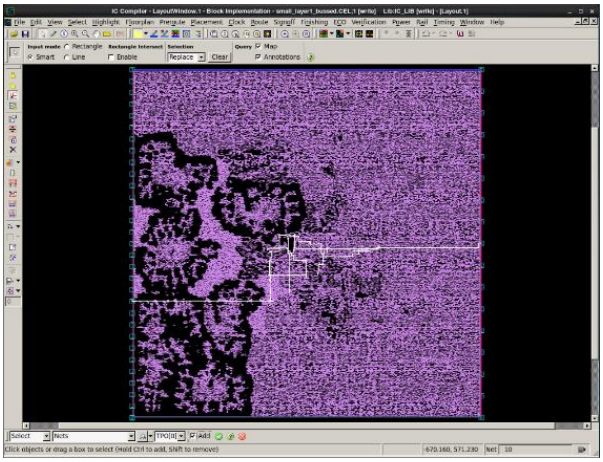


*Figure A.7 Floorplan of BUS circuit.*

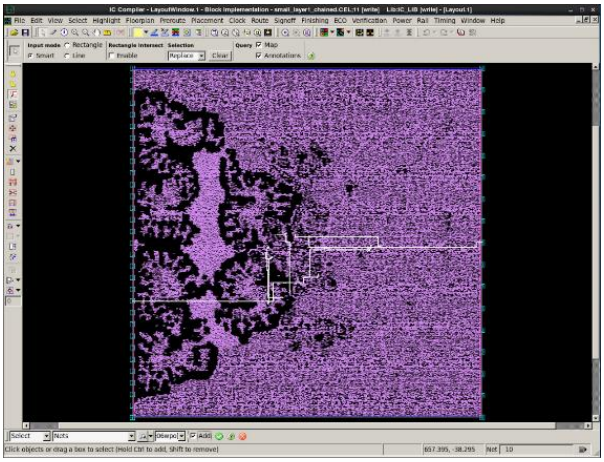
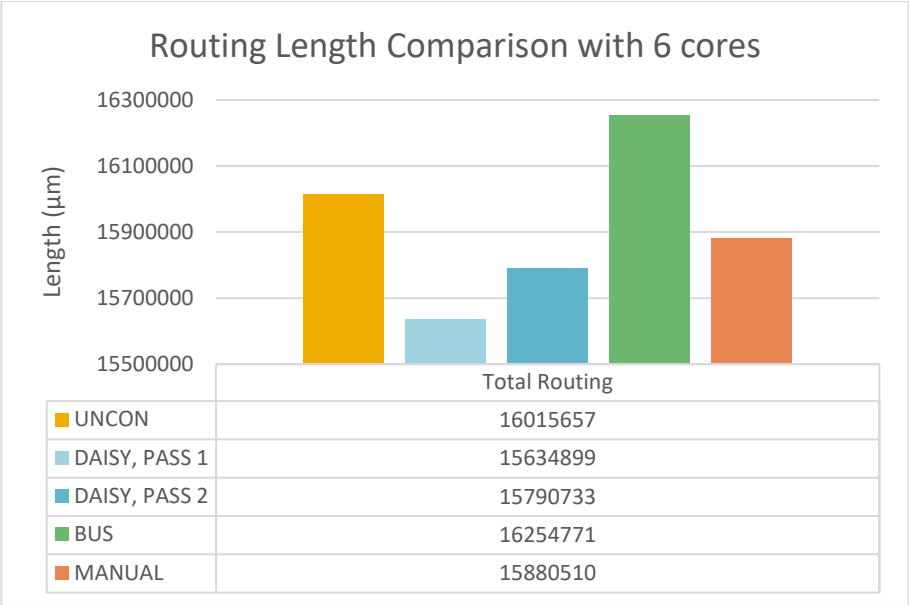


*Figure A.8 Floorplan of DAISY circuit, with its cores connected manually.*

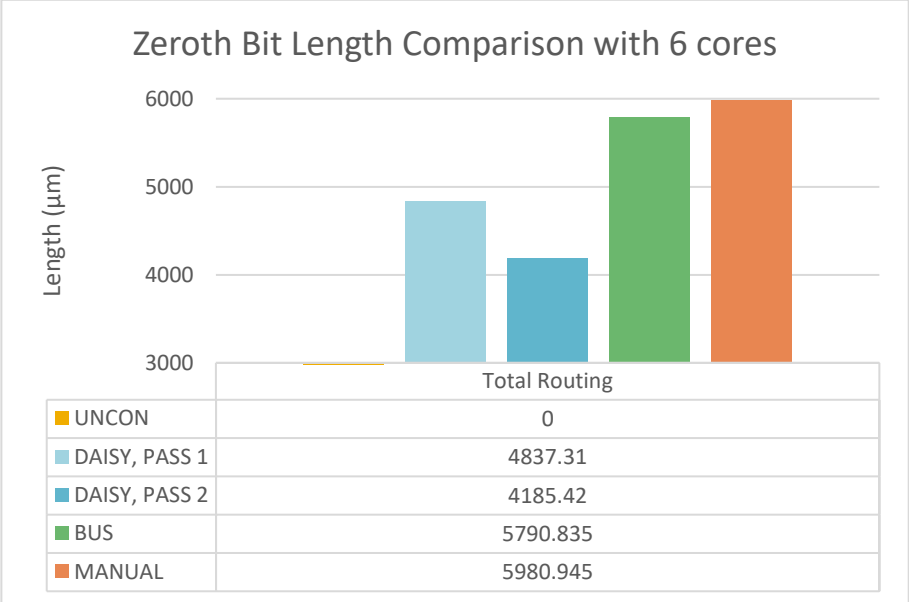
Finally, two more measurements of the same signals are made, one for the bused technique (fig. A.7), and one by linking the cores without using the Method-k3 (fig. A.8). As in, using the theoretical ratio of the sizes of the core, and linking them together in a way that made sense to the best humanly possible degree. The measurements are taken same as before, the whole routing length, and the specific length of the zeroth bit of the parallel connections between the dies. This distinction is the cornerstone of the explanation to the results this series of experiments have.




| | Total Routing |
|---|---|
| UNCON | 16015657 |
| DAISY, PASS 1 | 15634899 |
| DAISY, PASS 2 | 15790733 |
| BUS | 16254771 |
| MANUAL | 15880510 |

*Graph A.1 Comparison of total routing length between the circuits UNCON, DAISY after both Passes of the Method-k3, BUS, and the manually connected version of DAISY.*

Which are results that have to be combined with the measurements of the zeroth bit of the parallel connections between the cores in order for them to be understood.




| | Total Routing |
|---|---|
| UNCON | 0 |
| DAISY, PASS 1 | 4837.31 |
| DAISY, PASS 2 | 4185.42 |
| BUS | 5790.835 |
| MANUAL | 5980.945 |

*Graph A.2 Comparison of routing length for the zeroth bit of test signals between the circuits UNCON, DAISY after both Passes of the Method-k3, BUS, and the manually connected version of DAISY.*

The results shown above reveal that the total routing length is not a good measure to understand the length of the test data connections between the cores that change between the two techniques, nor the two passes of the Method-k3.

Combined, the two sets of measurements can explain all questions raised by this series of experiments.

- Why is the total routing of the three daisy chained dies lower than the total routing of the unconnected dies?
    - The answer lies with the fundamental difference between the bused and unconnected dies, with the chained ones. Their difference is, of course, the chains themselves. Without them, the tool has too much freedom with cell placement, creating unneeded congestion. The chains provide the die with a structure, which separates the cores better on the die, and allows for better routing.
- How can it be that the parallel length of pass 2 is less than the parallel length of pass 1, while the total length of pass 2 is a lot more than the total length of pass 1?
    - The answer lies with the focus of the Method-k3. To achieve better values on the test connections between the cores, it sets core cells in specific locations. As a result, the tool has less freedom with all other cells, which creates an uptick in total routing.

However, when comparing the ratio of the routing of the best chained die to the bused die, against the earlier ratio produced by the dies which had more than five times as many cores, the gap between the two techniques seems to lessen. Additionally, the results were once again far away from the theoretical values, which dictated that the bused dies should have twice as much routing as the daisy chained ones. The answer to that last question could only come after the re-instating of the bounded cores, and the comparison of the values between a die with 6 cores, and one with 12 cores.

Theoretically, the daisy chained die with the 6 cores ought to look as it has been presented in "Figure 4.1.6 An abstract rendering of the DAISY die". Two 8-bit lines that split early and go to their respective cores. In contrast, the bus connected die ought to be more complicated, with the two 8-bit buses having to go through the same pathways twice, which produces results double of that of the

daisy chain technique, as shown in “Figure 4.1.5 An abstract rendering of the BUS die”.

After the results presented in “Graph 7.1.3 Comparison between the DAISY and BUS circuits, and their 6-core and 12-core variants.”, and the usage of PAT A.5, a new design was made which served as a precursor to the Large Floorplan. In this phase of the design, IEEE 1838 std. was still a generalized proposal (P1838), and thus only served as an inspiration for what was named the Stacked Design. Its usage was simple but requires an example to be understood completely.


*Figure A.9 Waveform of initial combination of IEEE 1500 std and IEEE 1838 std.*

Combined with the well-known IEEE 1500 Standard *wrapper_des* core, the new states were encoded in the mandated WIR as “110”, meaning *WP_INTEST*. Meaning an in-test of the core using the parallel test data pins, which were connected to the FPP Switchbox of the P1838. As a first stage (fig. A.9), the inner scan chains of the core have to be filled, so that the state of the core can be set.

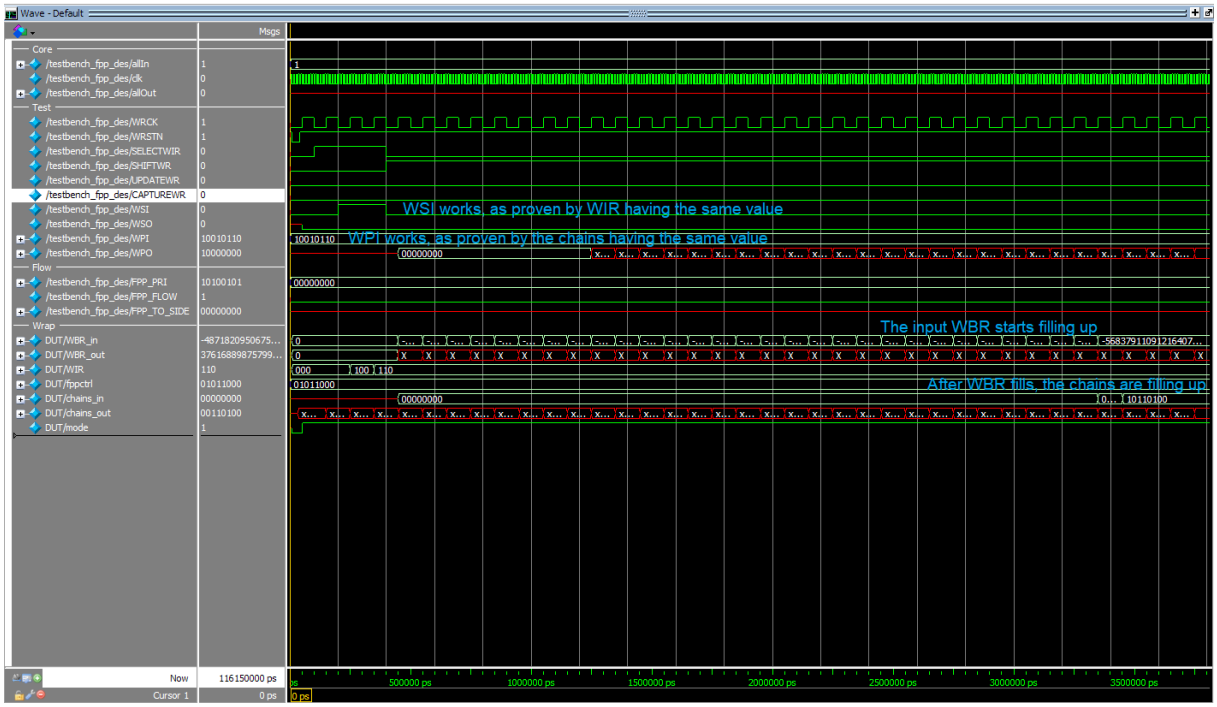



*Figure A.10 Beginning of waveform, zoomed in.*

More specifically, on the first half (fig. A.10), WSI activates the WIR, which allows the WPI to pass its value to the WBR, which in turn fills the inner scan chains of the core after it fills up.

Finally, the utility of the FPP Switchboard could be shown (fig. A.11), passing the values from the core to the hypothetical TSV, after being allowed to by the signal *FPP_FLOW*, which simulates what will later on be a signal coming from the PTAP. But that could only happen when the P1838 became the "IEEE 1838-2019 Standard", at the beginning of Spring 2020.

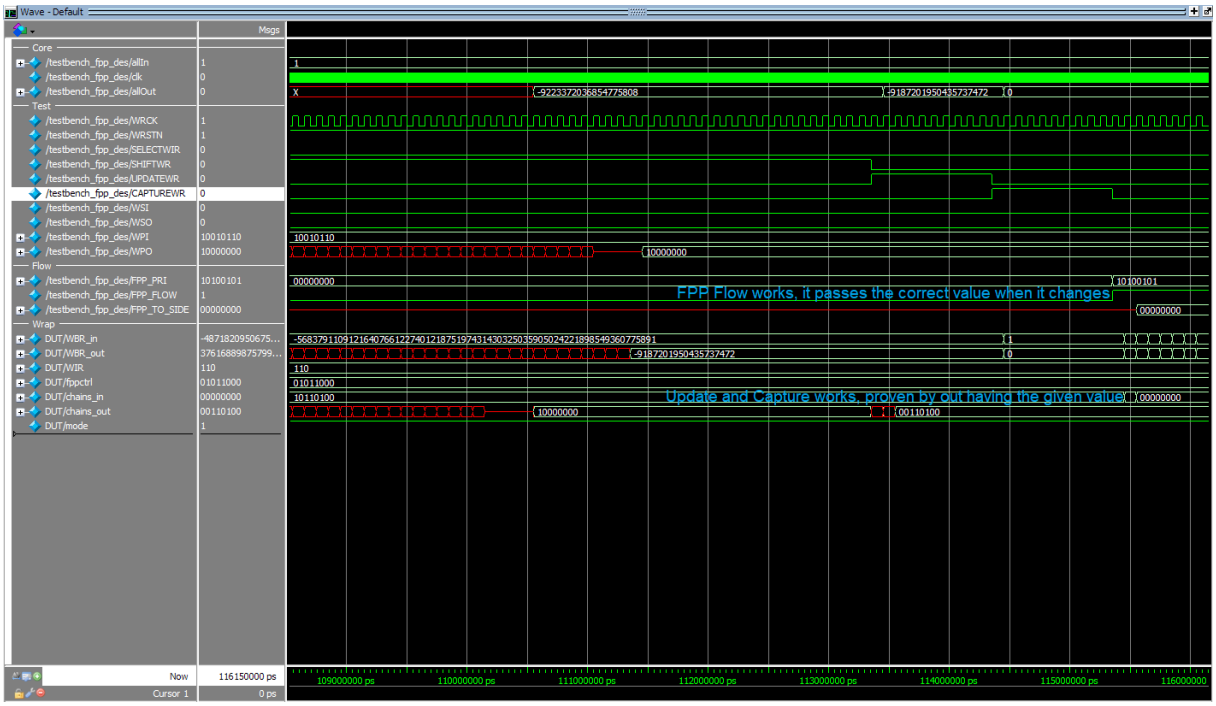



*Figure A.11 Ending of waveform, zoomed in.*

For the final design, the Large Floorplan, it is important to first look at the scanned cores after PAT G.1 makes them uniform.

*Table A.2. Number of pins per wrapper, with test and functional pins split.*

| | façade_des | façade_eth | façade_con |
|---|---|---|---|
| Functional Clock | 1 | 1 | 1 |
| Functional Reset | 1 | 1 | 1 |
| Test Enabling Control | 1 | 1 | 1 |
| Test Clock | 1 | 1 | 1 |
| Test Reset | 1 | 1 | 1 |
| Functional Inputs | 233 | 94 | 1128 |
| Functional Outputs | 64 | 115 | 1416 |
| Test Inputs | 8 | 8 | 8 |
| Test Outputs | 8 | 8 | 8 |

It is PAT A.5 which links together all those functional inputs and outputs, with a 5% chance of using the cluster's TSVs, followed by a 2/3rds chance of being

satisfied by elsewhere within the cluster. The remainder is split among the close neighbourhood of the cluster, or a 15% of long connections elsewhere on the die. The TSVs are reused during testing, through the FPP capabilities of the design.

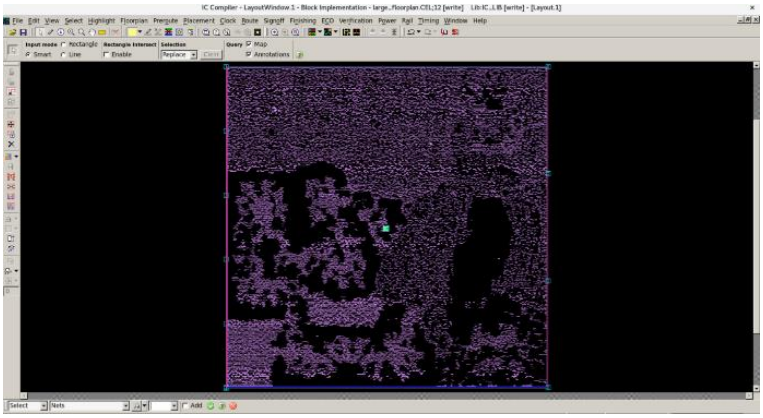


*Figure A.12 Floorplan of the single cluster.*

In particular, because of the complexity of the design, it would only be understood as a fully synthesized single cluster (fig. A.12). Or by hollowing out the insides of the sixty *façade cores*, so that only the connections above that level had to be synthesized, for when the functionality of the die was unimportant.

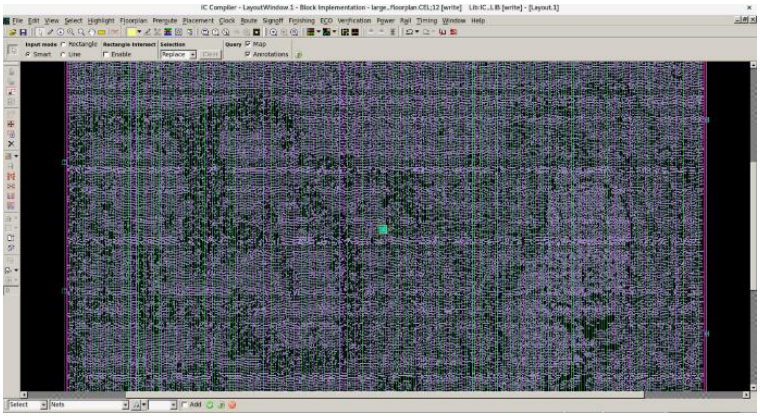


*Figure A.13 Floorplan of single cluster, zoomed.*

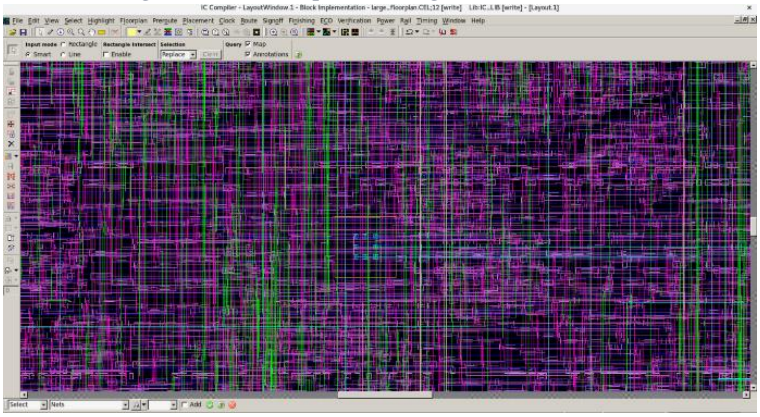


*Figure A.14 Floorplan of TSVs of single cluster.*

Thus, in this 1-cluster die, visible are the ports for the functional clock and reset, the ports required for the PTAP and the STAP of the IEEE 1838-2019 Standard which are for the entire die, and the 8 TSVs per cluster (fig. A.13). Zooming in, the TSVs are visible in a square formation, accompanied by the empty 5μm transistor gap around them (fig. A.14).

To begin understanding the signals and how they route around the die, it is important to understand a few parts of it. First and foremost, it is important to learn of the relative locations of the three cells contained within the cluster.

On paper, the *fpp_des* core ought to take most of the space, followed by *fpp_eth*, and lastly *fpp_con*. Inversely, it is expected for more congestion to exist around *fpp_con*, as it has by far the most connections, more than the other two cores have combined.

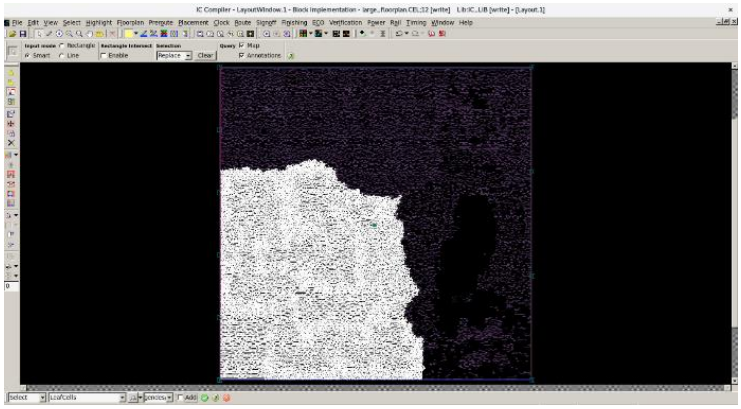


*Figure A.15 Des core leaf cells area in a single cluster.*

This is what happens on the synthesized die too, with *fpp_des* taking all the bottom-left of the die (fig. A.15), extending further on that the midpoint of both the bottom edge and the left side of the die.

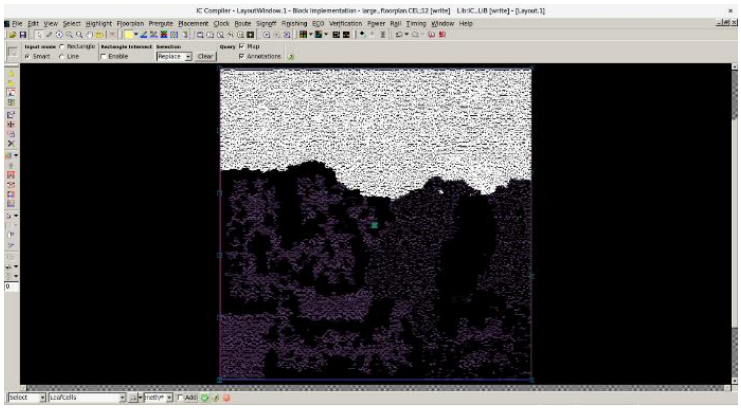


*Figure A.16 Eth core leaf cells area in a single cluster.*

Next, *fpp_eth* takes all of the upper part of the die (fig. A.16), spreading from the left edge to the die all the way to the right edge, apart from a few pieces at the right that will be presented later on.

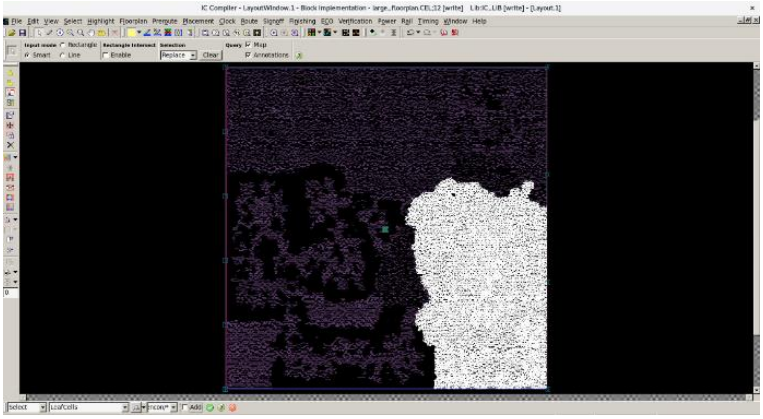


*Figure A.17 Con core leaf cells area in a single cluster.*

Lastly, *fpp_con* takes almost all of the remaining space (fig. A.17). Namely the right side of the die, after the midpoint of the bottom edge, but extending above the midpoint of the right edge, but to a smaller degree that *fpp_des* does.

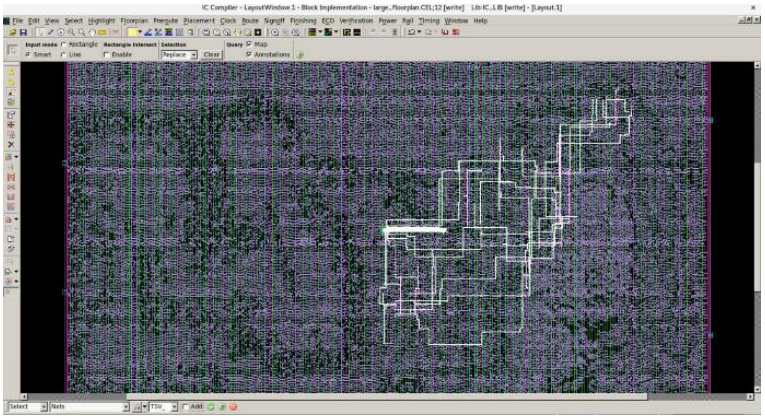


*Figure A.18 Connections of TSVs in a single cluster.*

The midpoint of the edges can be easily seen because of the placement of the TSVs, which is at the dead centre of the die. From that point, and extending outwards, are the signals from those TSVs (fig. A.18). However, as they are shared by both the functional and the test component of the dies, there are two components to them. For the functional component, they spread through the die, but for the test component of the die, they create a bus of signals that can be seen at the right of the centre of the die.

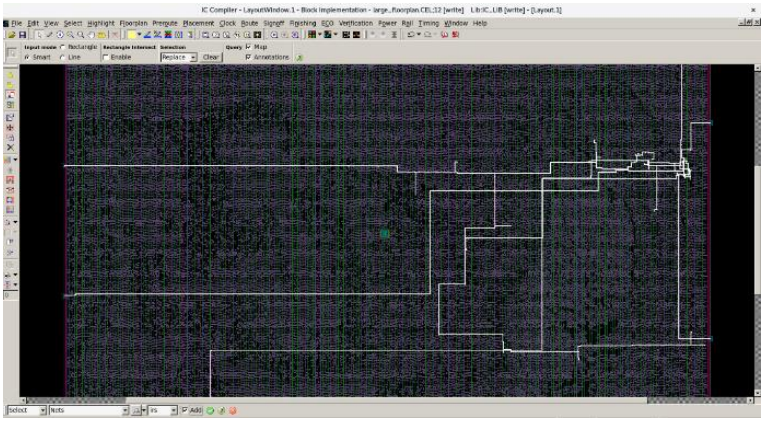


*Figure A.19 Shared testing signals in a single cluster.*

Following are the three signals which form the backbone of the testing capabilities of the die, known as *TCK*, *TMS*, and *TRST* (fig. A.19). They are signals known and well defined in the IEEE 1149.1 Standard, which are reused by the IEEE 1838 Standard, for a part of the test cells known as the PTAP.

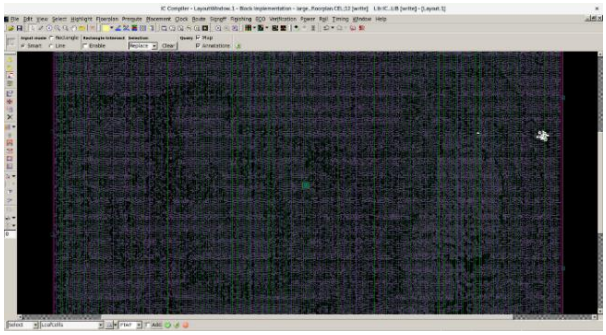


*Figure A.20 PTAP leaf cells area taken.*

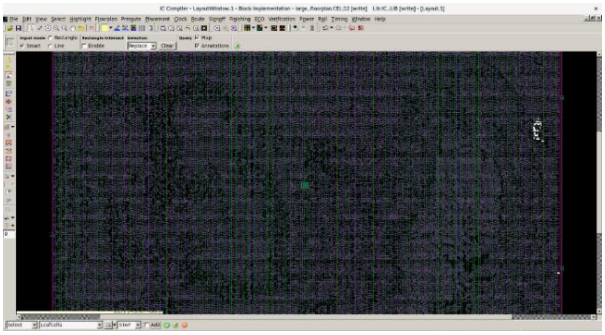


*Figure A.21 STAP leaf cells area taken.*

The reason these signals go through the entire die, from the left side of the die to the right one, is that they become the outputs of the STAP. They serve as the inputs of the next PTAP in line, hence creating the uniformity of the IEEE 1838 Standard. As for the PTAP (fig. A.20) and STAP (fig. A.21) cells themselves, they barely take any space on the die. Especially when, in a die with multiple clusters, they would still only exist once each, belonging to the design of the die, and not each specific cluster.

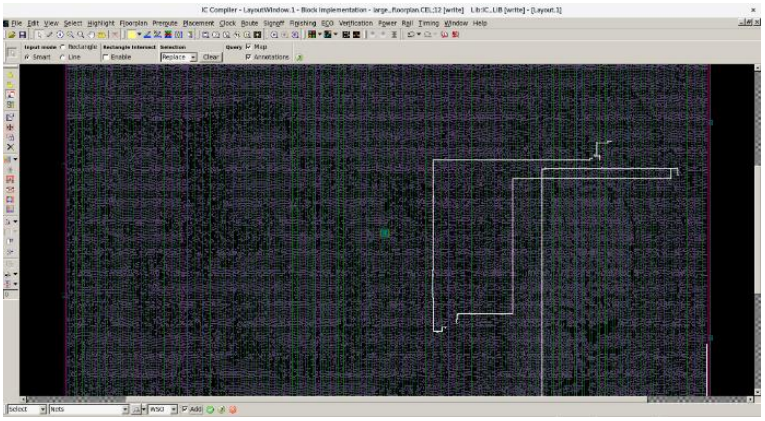


*Figure A.22 Chain connection of cores in a single cluster.*

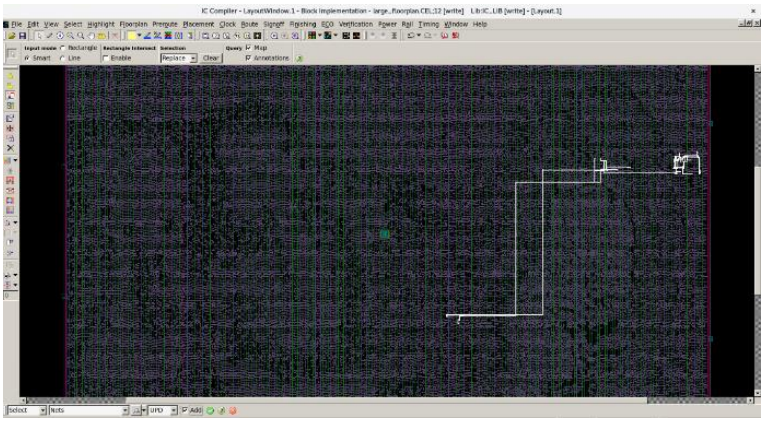


*Figure A.23 Capture, Shift, and Update signals in a single cluster.*

Of course, it is expected for the three cores within the clusters to be connected in a chain for their serial test data connections (fig. A.22). Along with their *CAPTURE*, *SHIFT*, and *UPDATE* signals to follow closely in the surrounding pins of the cores (fig. A.23), which are of course in turn created by the PTAP.

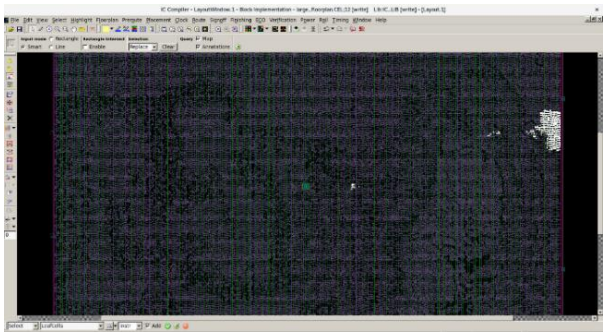


*Figure A.24 Leaf cells of Single Instruction.*

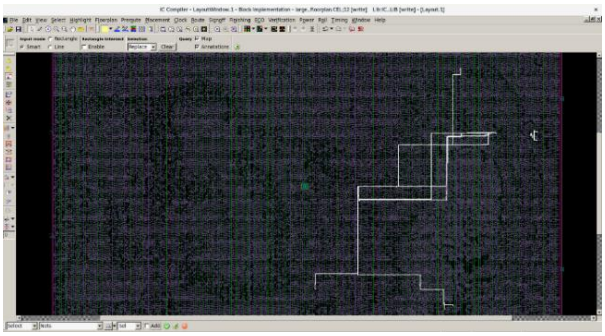


*Figure A.25 Select signal in a single cluster.*

Lastly, the rest of the cells of the IEEE 1838 Standard are presented, namely the Select Instruction part (fig. A.24), along with the *Select* signal (fig. A.25), which actually commands the cores to execute one of the test functions that they are able to carry out.

# APPENDIX B

## INSTRUCTIONS – OPERATIONS

Presented here are the instructions accepted by the Instruction Register on the total die, and the Wrapper Instruction Register in each of the cores.

*Table B.1 Instructions for the IR.*

| | Instruction Register (IEEE 1838 Std.) |
|---|---|
| 3'b000 | Bypass – The entire die is bypassed from testing. |
| 3'b001 | Select DWR Transparent – All testing apparatus is bypassed, and the entire die works as it is functionally intended. |
| 3'b010 | Select DWR Extest –The connections between the cores and their WBRs are tested. |
| 3'b011 | Select DWR Intest – The connections among the WBRs to the cores are tested. |
| 3'b100 | Select 3DCR – Able to change STAP(s) and active chain. |
| 3'b101 | Select IDCODE – Shifts out a specific 32bit identification number. |
| 3'b110 | Select DWR Sample Preload – Allows for the filling of the WBRs. |
| 3'b111 | Bypass – The entire die is bypassed from testing. |

*Table B.2 Instructions for the WIR.*

| | WIR Serial (IEEE 1500 Std.) |
|---|---|
| 2'b00 | Bypass – The specific core is bypassed from testing. |
| 2'b01 | Serial Testing Mode – The Scan Chains of the core are filled by the WBR, which shifts 1 bit at a time. |
| 2'b10 | Parallel Testing Mode – The Scan Chains of the core are filled by the WBR, which shifts 8 bits at a time, through the FPP. |
| 2'b11 | Bypass – The specific core is bypassed from testing. |

# APPENDIX C

## LARGE FLOORPLAN TSVS – FPP "TOWARDS"

While the chained connections between the clusters of the die have been mentioned extensively, little has been shown for the connections among the dies, through their TSVs. This is where the "Towards" value outside the FPP comes into play and is the reason why while the PRI and SEC pins of the FPP are bidirectional, they are not used as such in the Large Floorplan.

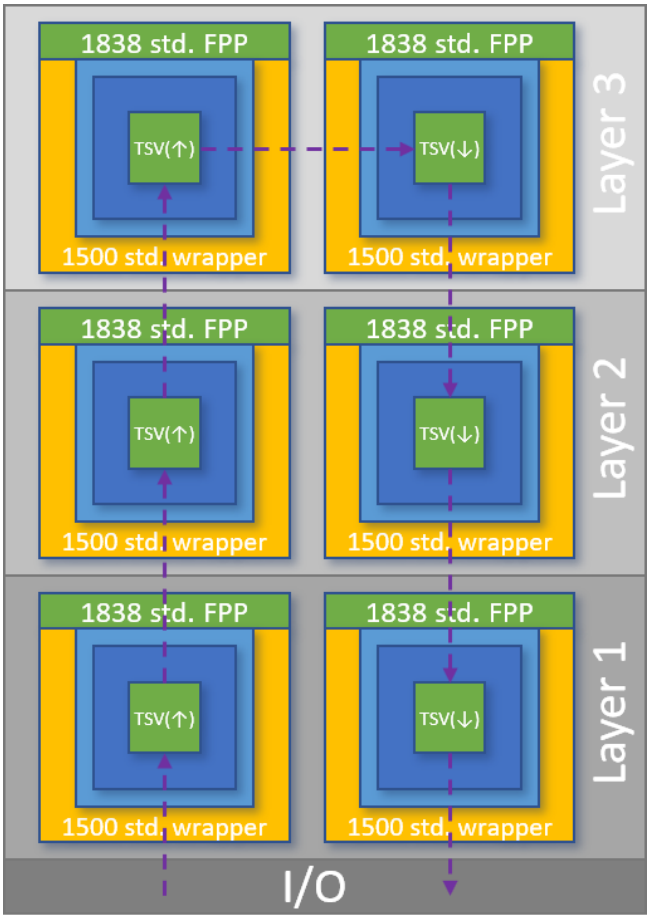



*Figure C.1 The shape of the FPP connections among the dies.*

The main idea is that the cores are connected in many "Π" shaped connections. Meaning that clusters with an odd number within the daisy chain use their TSVs to shift data away from the I/O of the stack. While clusters with an even number use theirs to shift data towards them. This is done with the help of the 8

control signals of the FPP, namely *PRI_SEL*, *CORE_SEL*, *SIDE_SEL*, *REGPD_BYP*, *REGPU_BYP*, *REGN_BYP*, *PRI_EN*, and *SEC_EN*.

*Table C.1 Values for the TOWARDS value.*

| | Towards – FPP Control Signal |
|---|---|
| 2'b00 | 8'b11111000 – Connection from Side to Side. |
| 2'b01 | 8'b01111000 – Connection from Pri to Side. |
| 2'b10 | 8'b11011000 – Connection from Sec to Side. |
| 2'b11 | 8'b11111011 – Connection from Side to All. |

For example, in a cluster with an odd number within the daisy chain, the first core has a "towards" value of 2'b01, the second core of 2'b00, and the third core of 2'b11 but with only its SEC port connected.

# CURRICULUM VITAE

**Konstantinos Leventos**

PhD Student at the Department of Computer Science & Engineering in the School of Engineering of the University of Ioannina

**Tel.** 6949971566

**Email** kon.leventos@gmail.com & k.leventos@uoi.gr

**Address** Dexamenis 4-10 Ioannina, Greece

## Education

October 2018 – January 2021, MSc in Advanced Computing Systems, Department of Computer Science and Engineering of the University of Ioannina.

> October 2019 – January 2021, MSc Thesis "3D IC Testing: Automated TAM Architecture".

September 2013 – September 2018, BSc & Integrated MSc Diploma in Computer Science & Engineering, University of Ioannina.

> February 2018 – June 2018, BSc Thesis "RISC Microprocessor Design and Implementation: Design and Evaluation of Techniques for the Limitation of Load-Use Delays in a Pipelined Processor".

## Additional Degrees

2011 – English, Cambridge ESOL Level 3, Certificate of Proficiency

2011 – French, Hellenic State Language Certificate