NAIST-IS-DD0661211

Doctoral Dissertation

Studies on Power, Thermal & False-path Aware Test Techniques for Modern System-on-Chips

Thomas Edison Chua Yu

March 24, 2009

Department of Information Processing
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Thomas Edison Chua Yu

Thesis Committee:

Professor Hideo Fujiwara	(Supervisor)
Professor Yasuhiko Nakashima	(Co-supervisor)
Associate Professor Michiko Inoue	(Co-supervisor)
Assistant Professor Satoshi Ohtake	(Co-supervisor)
Assistant Professor Tomokazu Yoneda	(Co-supervisor)

Studies on Power, Thermal & False-path Aware Test Techniques for Modern System-on-Chips*

Thomas Edison Chua Yu

Abstract

Rapid advances in semiconductor manufacturing processes and design tools have led to a relentless increase in chip complexity. High power consumption and heat densities have become major concerns. These problems are greatly exacerbated for System-on-Chips (SoCs) which integrate several functional cores on one chip. SoCs operating at multiple clock domains and very low power requirements are being utilized in the latest mobile devices. Thus, the testing of SoCs under power and temperature constraints have been rapidly gaining importance. For this thesis, we first introduce a novel method for designing power-aware test wrappers for embedded cores with multiple clock domains. By effectively partitioning the various clock domains, making use of bandwidth conversion, multiple shift frequencies and clock-gating, we gain greater flexibility in determining an optimal test schedule under very tight power constraints.

For Socs, imposing power constraints does not always solve the problem of overheating due to the non-uniform distribution of power across the chip. We present two TAM/Wrapper co-design methodologies for SoCs that ensure thermal safety while still optimizing the test schedule. The methods combine simplified thermal-cost models with bin-partitioning and packing algorithms to minimize test time while satisfying temperature constraints.

Another problem is the difficulty in identifying untestable multi-cycle paths. Their rapid and accurate identification could result in significant reductions in

* Doctoral Dissertation, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0661211, March 24, 2009.

Automatic Test Pattern Generation (ATPG) time, tester memory, test cost and chip over-kill. For this, a novel method of identifying multi-cycle false paths at Register Transfer Level (RTL) is presented along with a case study to prove its effectiveness.

Keywords:

SoC, low-power, low-temperature, multi-cycle, design-for-testability, false-path identification

List of Publications

Journal Papers

1. Thomas Edison Yu, Tomokazu Yoneda, Danella Zhao and Hideo Fujiwara, "Effective domain partitioning for multi-clock domain IP core wrapper design under power constraints," *The IEICE Transactions on Information and Systems*, Vol.E91-D, No.3, pp.807-814, March 2008.

2. Thomas Edison Yu, Tomokazu Yoneda, Krishnendu Chakrabarty and Hideo Fujiwara, "Thermal-aware test access mechanism and wrapper design optimization for system-on-chips," *The IEICE Transactions on Information and Systems*, Vol.E91-D, No.10, pp.2440-2448, Oct. 2008.

International Conferences (Reviewed)

1. Thomas Edison Yu, Tomokazu Yoneda, Danella Zhao and Hideo Fujiwara, "Designing power-aware wrapper for multi-clock domain cores using clock domain partitioning," *Proceedings of the 7th IEEE Workshop on RTL and High Level Testing (WRTLT'06)*, pp.43-48, Nov. 2006.

2. Thomas Edison Yu, Tomokazu Yoneda, Danella Zhao and Hideo Fujiwara, "Using domain partitioning in wrapper design for IP cores under power constraints," *Proceedings of the 25th IEEE VLSI Test Symposium (VTS'07)*, pp.369-374, May 2007.

3. Thomas Edison Yu, Tomokazu Yoneda, Krishnendu Chakrabarty and Hideo Fujiwara, "Thermal-safe test access mechanism and wrapper co-optimization

for system-on-chip," *Proceedings of the 16th IEEE Asian Test Symposium (ATS'07)*, pp.187-192, Oct. 2007.

4. Thomas Edison Yu, Tomokazu Yoneda, Satoshi Ohtake and Hideo Fujiwara, "Identifying non-robust untestable RTL paths in circuits with multi-cycle paths," *Proceedings of the 17th IEEE Asian Test Symposium (ATS'08)*, pp.125-130, Nov. 2008.

5. Thomas Edison Yu, Tomokazu Yoneda, Krishnendu Chakrabarty and Hideo Fujiwara, "Test Infrastructure design for core-based System-on-Chip under cycle-accurate thermal constraint," *Proceedings of the 14th Asia and South Pacific Design Automation Conference (ASP-DAC'09)*, pp.793-798, Jan. 2009.

Technical Reports

1. Thomas Edison Yu, Tomokazu Yoneda, Danella Zhao and Hideo Fujiwara, "Power constrained IP core wrapper design with partitioned clock domains," *Technical Report of IEICE*, Vol. 107, No. 101, pp.37-42, June 2007. (In English)

2. Thomas Edison Yu, Tomokazu Yoneda, Krishnendu Chakrabarty and Hideo Fujiwara, "Thermal-aware test scheduling with cycle-accurate power profiles and test partitioning," *Technical Report of IEICE (VLD2007-84, DC2007-39(2007-11))*, Vol. 107, No. 335, pp.13-18, Nov. 2007. (In English)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The recent popularity of advanced technologies such as broadband internet, next-gen cellular phones and high-speed workstations are due to many factors, one of which is the rapid advancement in the design and production of VLSI (Very Large Scale Integration) chips. More importantly, it has now become possible to put entire systems onto a single chip which is commonly known as *System-on-Chip* (SoC). Currently, SoCs are widely used in devices intended for telecommunications, networking and digital signal processing. Moreover, they are increasingly being utilized in mobile on-the-field devices which increase the demand for highly-reliable, defect-free chips. In this chapter, the concept of core-based design, its advantages and disadvantages, as well as its impact on VLSI testing is discussed.

1. The System-on-Chip Paradigm

Traditional systems, called *System-on-Board* or SoB, involve using both pre-designed and user-defined *Integrated Circuit* (IC) chips which are laid-out on a *printed circuit board* (PCB). While SoBs generally cost higher and have lower performance than embedded systems, all the components are already pre-tested before use and it is just a matter of testing the interconnects and the operation of the whole system. Thus, faulty components can be easily replaced. As shown in Figure 1.1, SoCs are different from SoBs mainly in the fact that SoCs are designed from pre-designed logic modules called *Intellectual Property* (IP) cores which are embedded onto a single IC. Using IP-cores enables quicker time-to-

market because designers can re-use ready-made core designs as well as all the test data that pertains to it. This separates the parties involved in SoC design into two main groups: the *core providers* which provide the pre-designed cores, and the *system integrators* who choose which IP-core to utilize and how to integrate them in the SoC design. As shown in Figure 1.2, IP-cores can be provided in three formats. *Soft cores* are usually given in a high-level description language such as VHDL (VHSIC Hardware Description Language) and are thus technology independent and easily modified but sacrifice optimization, test generation and design time. On the other hand, *hard cores* are given as highly optimized synthesized layout files which cannot be modified, as well the corresponding test data. This means that hard cores sacrifice flexibility for faster design time and performance. Right in the middle are *firm cores* which are usually given as netlists with modifiable libraries. They present a compromise between the flexibility of the soft cores and the optimization of hard cores. No matter what form of cores are used, the resulting SoCs have the advantages of higher performance, lower cost, smaller size and lower power requirements. But, unlike SoBs, the testing of the IP-cores and interconnects must be considered at the same time which make SoC testing far more complicated.

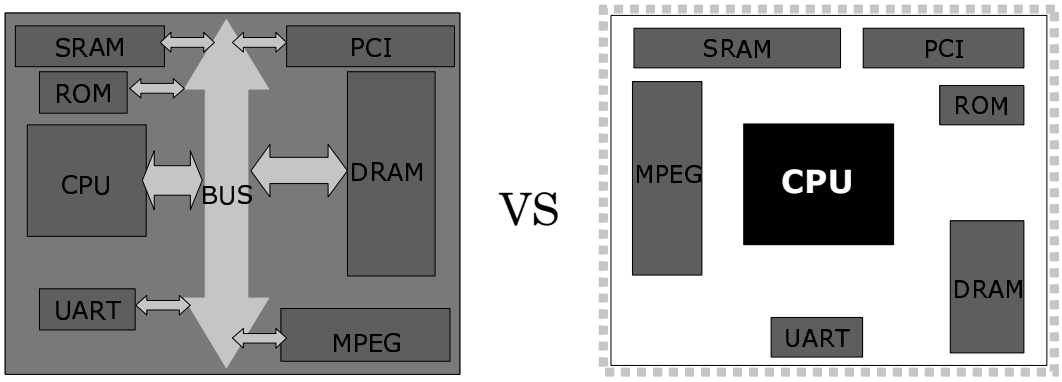


Figure 1.1. System-on-Board(left) vs. System-on-Chip(right)

In summary, SoCs give chip makers and designers the ability to combine predesigned IP cores into complex systems more cheaply and quickly. Furthermore, designs can be customized for various functions, such as for low-power applica-

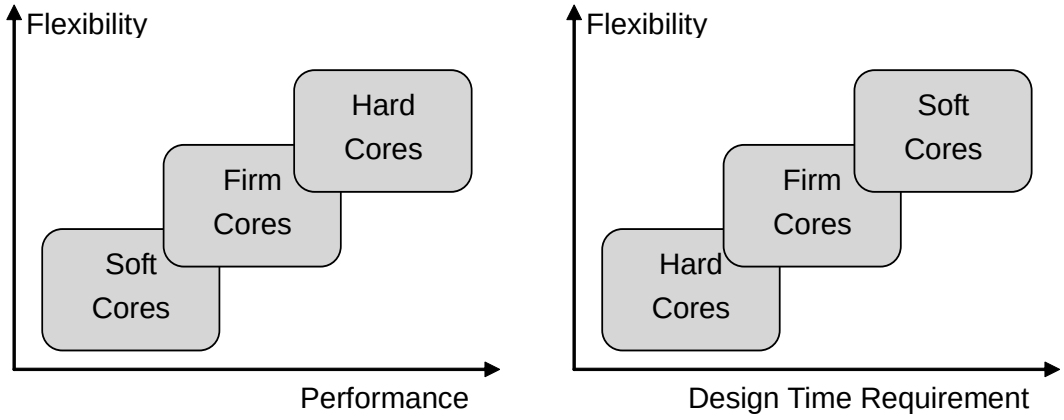


Figure 1.2. Comparison of trade-offs between core formats

tions. On the other hand, SoC testing presents the following problems:

- For complex designs, *Automatic Test Pattern Generation* (ATPG) can take a very long time. It might also be impossible to obtain acceptable test coverage. This can be somewhat alleviated if we find ways to re-use available test data that came with the pre-designed IP cores.
- Limited number of chip terminals means that direct control and observation of the cores during test is difficult.
- Minimizing test time usually means testing as many parts of the chip as possible, which usually leads to higher than normal power dissipation during test. This can have effects ranging from random errors to permanent chip damage which leads to yield loss.
- High power dissipation and uneven power distribution across the chip can lead to overheating and hot spots, which also affects yield.

2. Design for Testability (DFT)

To ensure that SoCs and other VLSI chips operate as intended, testing must be conducted per chip. As production capabilities improve, clock-rates rise expo-

nentially and transistor density increases dramatically, the testing of newer VLSI chips becomes a barrier to fully utilizing these newer technologies. More specifically, additional problems arise when testing SoCs. First of all, the increased complexity of the circuitry also means an increase in the amount of test data which usually results in longer test application time. Furthermore, test access becomes a problem since the cores cannot be directly accessed from the I/O pins of the chip. To solve these problems, various modifications, called *Design for Testability* (DFT) techniques, can be made to the design which would make testing faster and more efficient. The most common DFT for modern ICs is called *scan design*. The main difficulty in testing sequential circuits is the fact that we need to be able to set specific memory elements to a desired value as well as be able to observe test response values after test application. In scan-based designs, memory elements called *flip-flops* (FF) are connected serially into shift registers as seen in Figure 1.3. In this way, the values in the memory elements can be set by shifting-in the test stimulus followed by executing the test with the application of the system clock, and finally, test response can be captured by the FFs and shifted-out for observation. Furthermore, using a scan path virtually turns the circuit into a combinational design during testing and more conventional tools for combinational circuits can be used for test generation.

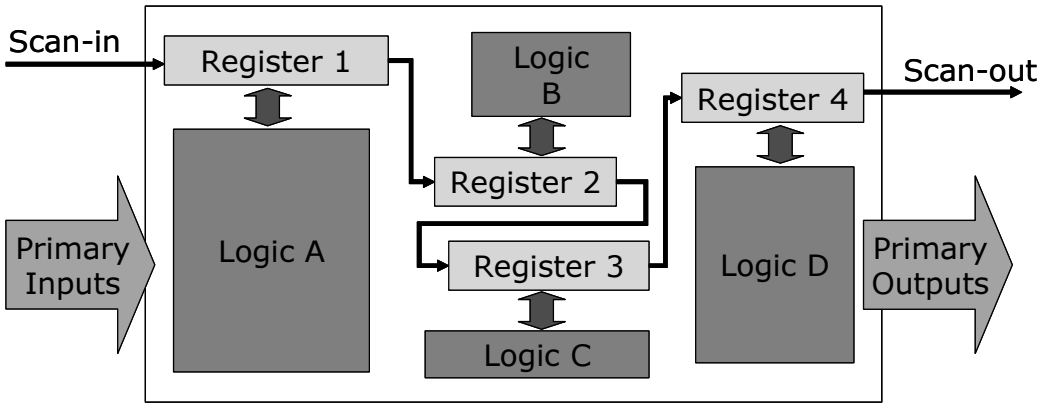


Figure 1.3. Scan design DFT technique applied on a core

2.1 DFT for SoCs

To better cope with the problems presented by SoCs with multiple cores, a test data delivery method, more commonly known as a *Test Access Mechanism* or TAM, and *wrappers* which isolate cores under test are commonly used. The wrapper isolates a core during test and provides an interface to apply and collect test data from the core under test. As shown in Figure 1.4, the TAM connects an external test source such as an *automatic test equipment* (ATE) to the *core under test* (CUT) as well as serving as a pathway for test response to be transmitted to a test sink such as devices for test data analysis.

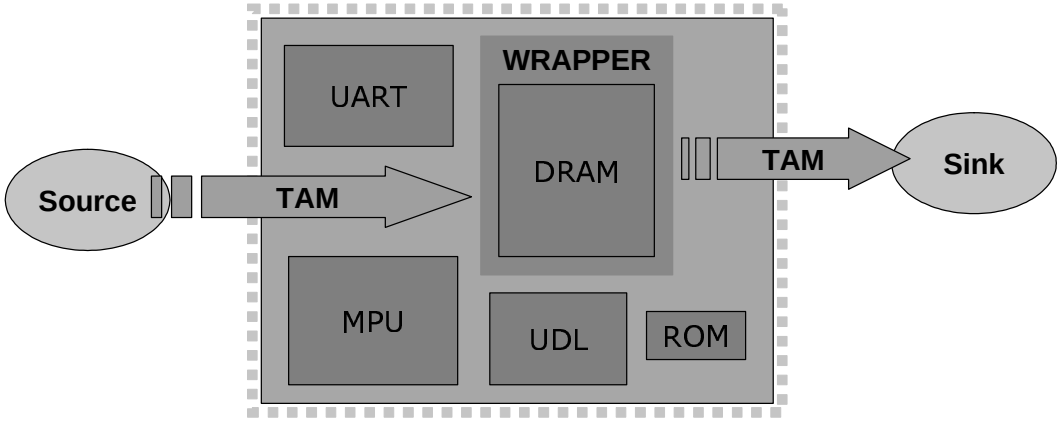


Figure 1.4. TAM-Wrapper combination for core-based test

Wrapper and TAM Test Architecture

More recently, the IEEE 1500 standard for embedded core test has been approved to provide guidelines for core wrapper design and interfacing to TAMs. Furthermore, several approaches to optimize wrapper designs for single frequency embedded core testing [1, 2] as well as wrapper and TAM co-optimization algorithms [3, 4, 12, 13] have already been suggested. Figure 1.5 illustrates a simple 1500 compatible wrapper. To realize a scan-based DFT, multiplexers are added to FFs to form wrapper cells to provide test access to the core terminals as shown in Figure 1.6. Note that the wrapper scan cells are connected to primary input

and output terminals and the internal FFs of the core are connected serially into scan chains. The boundary cells and the internal scan chains are then connected serially into several wrapper scan chains interfaced serially or in parallel with the external TAM. Dedicated circuitry controlling the wrapper as well as executing commands from the ATE are also present.

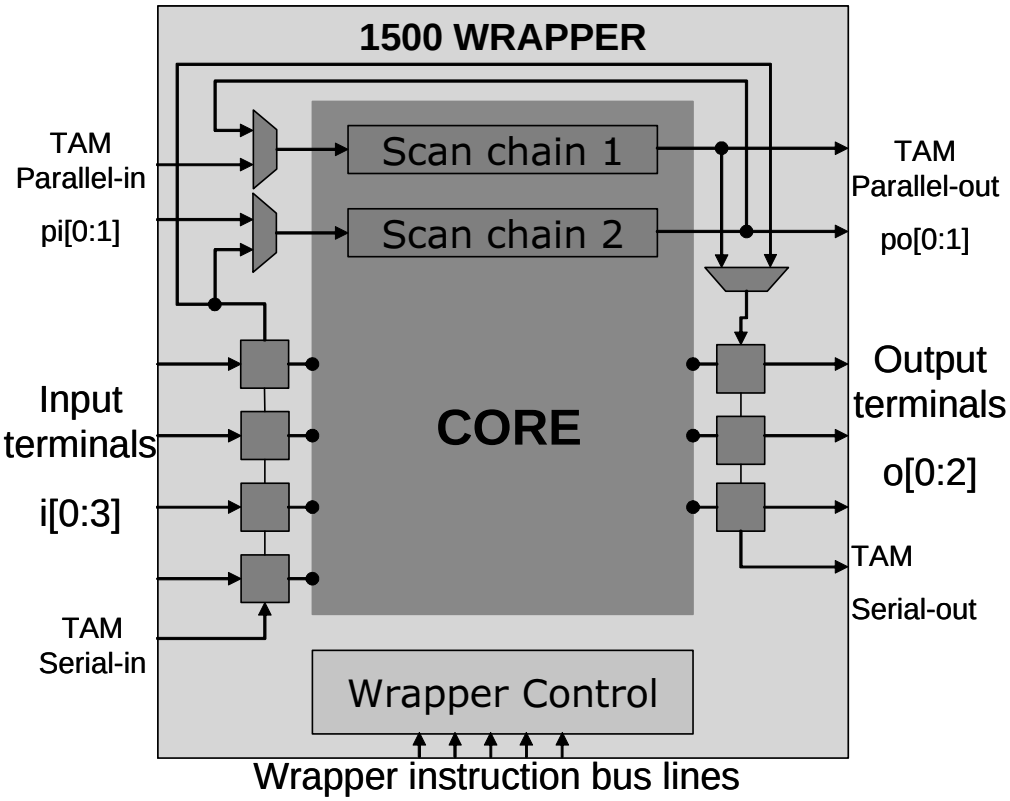


Figure 1.5. IEEE 1500 compatible wrapper

During wrapper design, the main aim is determining how scan-chains should be connected to wrapper scan cells to form wrapper scan chains given the allocated TAM width, such that the overall test application time is minimized. For example, given three scan chains of lengths 5, 8, and 4 with two input (pi) and three output (po) wrapper cells and TAM width of 2, various wrapper scan chains can be formed as shown in Figure 1.7. Normally, test data are scanned into all the wrapper scan chains at the same time, so for single frequency wrappers, the

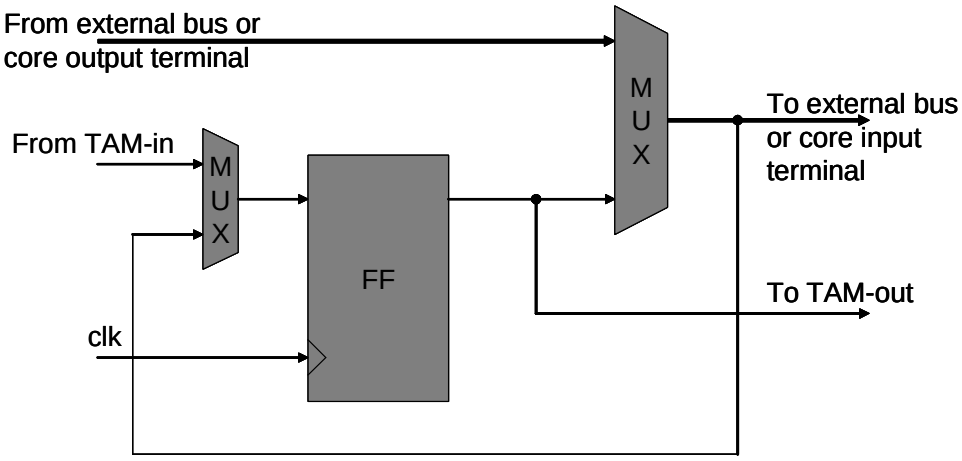


Figure 1.6. Typical wrapper scan cell

test application time is given by the equation below:

$$\tau(C) = 1 + max(sci, sco) \times p + min(sci, sco) \tag{1.1}$$

where *sci* and *sco* are the lengths of the maximum input and output wrapper scan-chains of core C respectively, and p is the number of test patterns for C.

Wrappers must generally be designed around specific TAM architectures. [2] described three basic scan-based TAM architectures depicted in Figure 1.8. For the *Multiplexing architecture* (Figure 1.8(a)), all the cores can access the full available TAM width, but they can only do so one at a time. This means that the total test time would be the sum of the individual test times of the cores, since they are tested in sequence. Furthermore, testing the external circuitry and wiring between the cores is difficult since each core can only be accessed one at a time. The *Daisychain architecture* (Figure 1.8(b)) has all the advantages of the Multiplexing architecture but overcomes the problem of external circuitry test and sequential-only test by adding bypass mechanisms into the architecture. The available TAM width is distributed among the cores in the *Distribution architecture* (Figure 1.8(c)), and allows concurrent tests of cores belonging to different TAM partitions. By optimizing the partitioning scheme in relation to the test data volume, the overall SoC test time can be minimized.

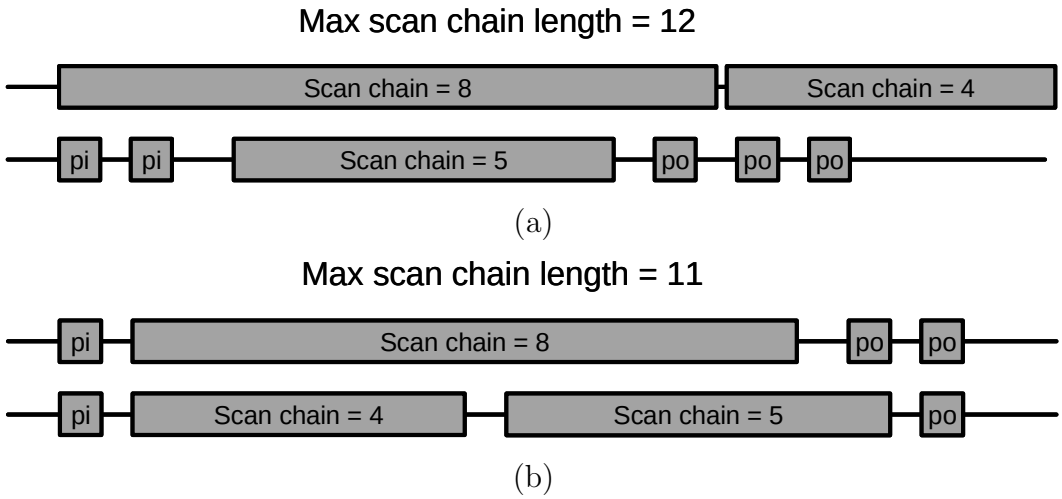


Figure 1.7. Example wrapper scan chain configurations

The basic TAM schemes can also be combined to form more complex and efficient TAM architectures. One such scheme is the *Test Bus architecture* [2], which is a combination of the Multiplexing and Distribution architectures, as shown in Figure 1.9. Cores connected to a single test bus (or TAM partition) can only be tested sequentially, as in a Multiplexing architecture. SoCs can have multiple test buses, each operating independently like in the Distribution architecture, and cores belonging to different buses can be tested in parallel to minimize test time. If we extend the capabilities of the Daisychain architecture such that each core can have a different TAM width assignment and test sequence is arbitrary, then we end up with what is referred to as a *Flexible TAM architecture* (Figure 1.10). In this scheme, the TAM distribution is decided with respect to the test schedule such that cores are tested with the maximum available TAM width, thus minimizing idle time and wasted test resources.

2.2 Test Scheduling

Test scheduling is basically deciding when to test certain parts of the chip. Conventional methods just treat SoCs as a flat design, performs ATPG without considering design hierarchy, and applies the test accordingly. Obviously, this does

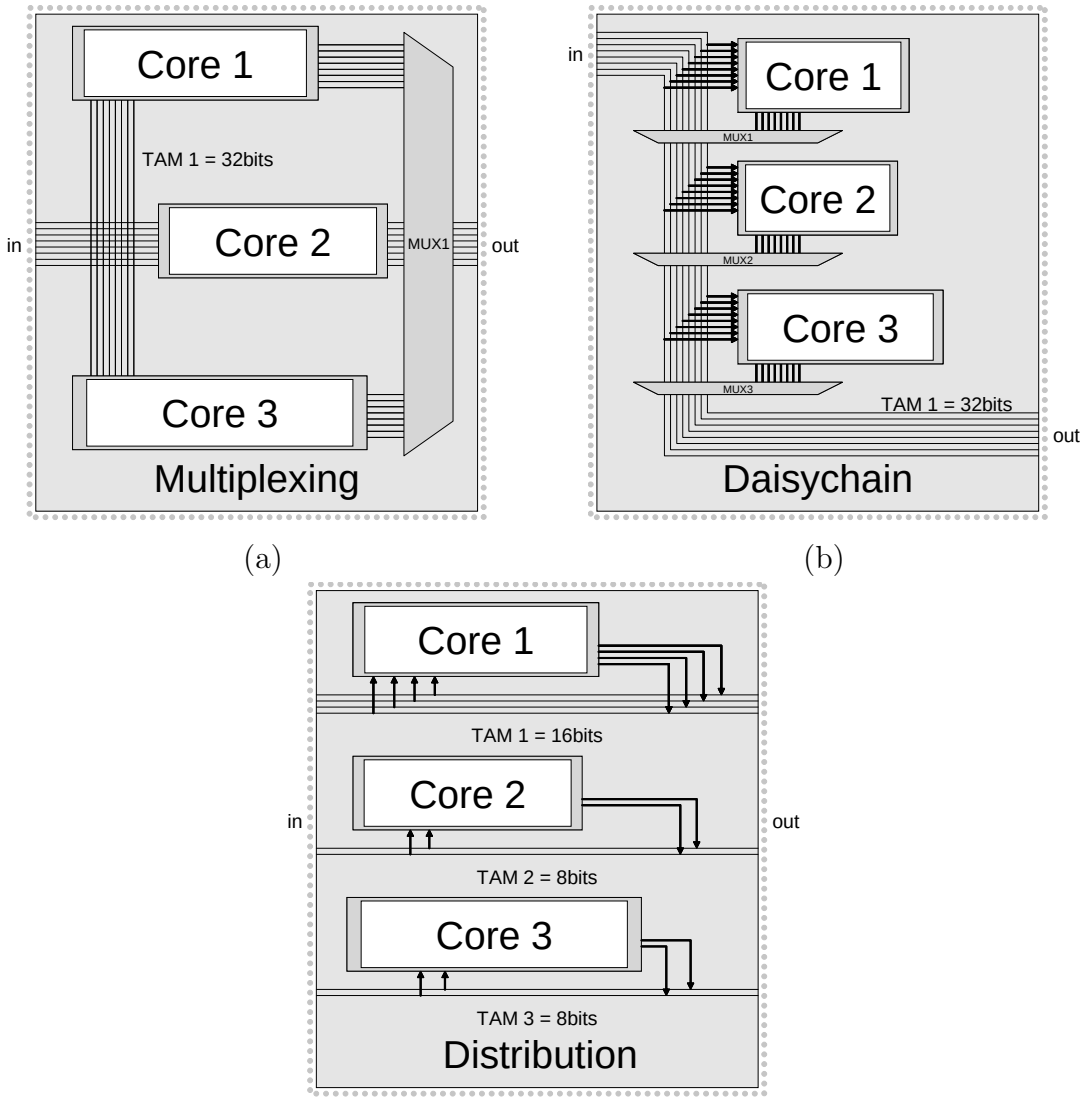


Figure 1.8. (a) Multiplexing, (b) Daisychain, and (c) Distribution TAM architectures.

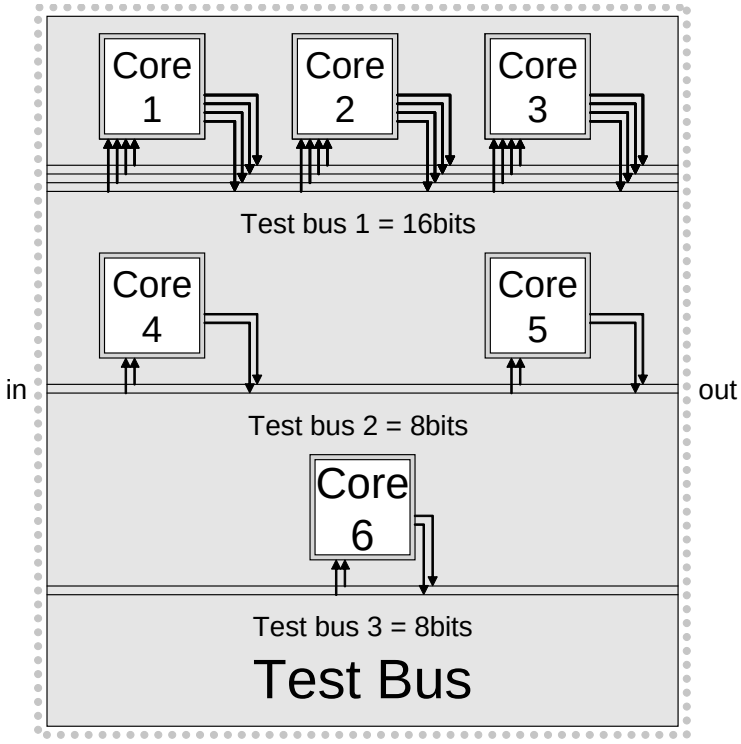


Figure 1.9. Example Test Bus architecture

not take advantage of the unique properties of SoCs, i.e. the ability to do modular testing and test data re-use. With the help of the DFT techniques presented before, we can obtain more efficient test schedules.

Typically, an SoC test is constrained by available TAM width. Thus, we can model a specific wrapper configuration of a core as a rectangle (or bin) whose height represents the required TAM width and width represents the required test application time. We can represent an empty schedule as an empty bin, whose height represents the overall TAM width and whose width represents overall test application time. Since the bin height is bounded by the overall TAM width, test scheduling is just a matter of filling the space with the proper rectangles (representing test of cores) such that the width (overall test time) is minimized. Figure 1.11 shows a possible schedule for the given Test Bus architecture in Figure 1.9. Here, Cores 1, 2 and 3 are tested sequentially, while cores 1, 4 and

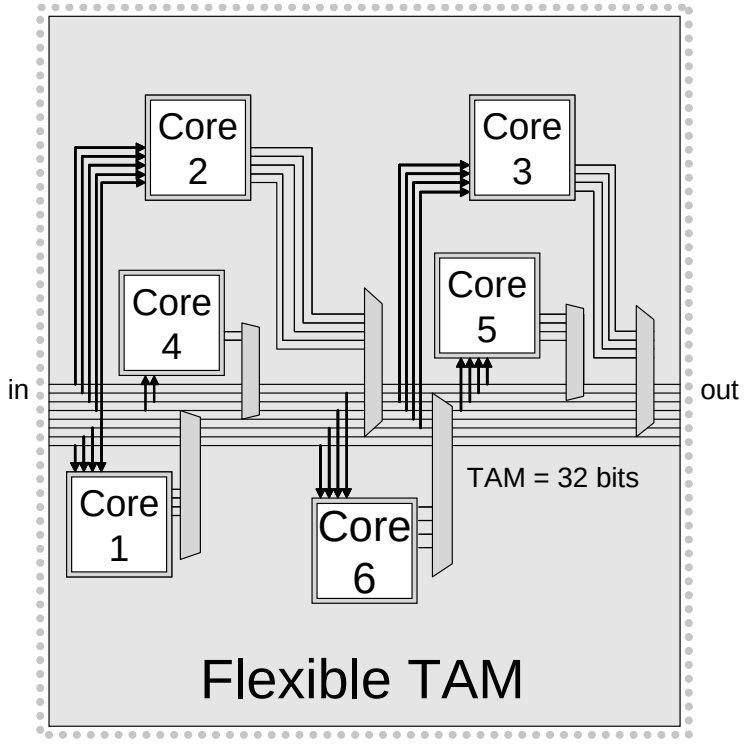


Figure 1.10. Example Flexible TAM architecture

6 are tested concurrently because they belong to different TAM partitions. For Flexible TAMs, there is no predetermined TAM partitioning scheme and cores can be tested as soon a TAM lines are available, as shown in Figure 1.12.

Besides TAM width constraints, other factors such as power dissipation can also be considered. In this case, the 2-dimensional Bin packing problem can be expanded into a 3-D Bin packing problem, where the height represents TAM width, width represents test time, and length represents power dissipation. Since total power during a certain time instance is just the cumulative sum of the power dissipations of the active components during that time, simple 3-D bin packing algorithms will suffice for scheduling. It gets more complicated when we consider things such as temperature, since temperature values cannot be super-positioned and we have to take into account factors such as cooling method, packaging, layout and power distribution.

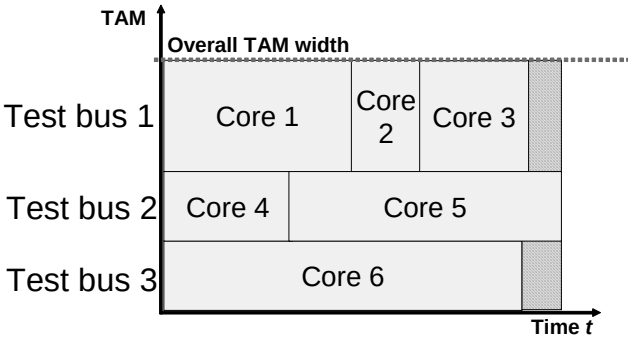


Figure 1.11. Example schedule for Test Bus architecture

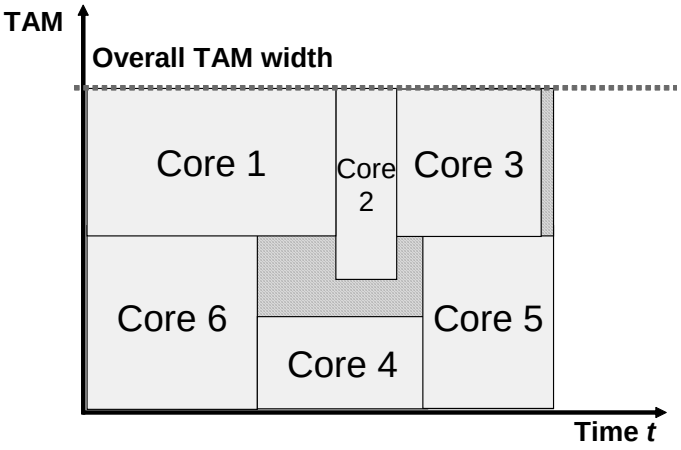


Figure 1.12. Example schedule for Flexible TAM architecture

3. Contributions of This Thesis

This thesis makes three main contributions to address the problems of SoC testing with regards to power-awareness, thermal safety and test efficiency. More specifically, the thesis introduces techniques that addresses the above test problems for current and future SoC designs. Furthermore, the following works can be combined during test planning to ensure proper, accurate and efficient test results.

The first contribution targets power-aware wrapper design for cores with multiple clock domains. We assume that as more designs are re-used, future SoCs would utilize cores operating at multiple frequencies. We propose a wrapper design methodology that alleviates the clock-skew problem presented by multi-clock designs while trying to minimize core test time under a power constraint. Furthermore, a simple heuristic 3-D bin packing test scheduling algorithm is also presented. The method improves upon previous works by Xu, et al. [9] and Zhao, et al.[10], especially under tight power constraints, by making use of domain partitioning and gated-clocks.

The second contribution is with regards to thermal aware SoC test architecture design. The thesis shows the limitations of power-constrained testing with regards to thermal control and proposes two wrapper/TAM architecture design methodologies that target flexible TAM and fixed-TAM architectures, respectively. To overcome the need for long thermal simulations, we make use of a simple thermal flow model and developed several thermal cost functions to aid our heuristic test scheduling algorithms. Furthermore, the effectiveness of test partitioning, test interleaving as well as bandwidth-matching in temperature control is also discussed. To the best of our knowledge, this is the first work which presented a complete thermal-safe TAM and wrapper design methodology.

Finally, the third contribution relates to the problem of long ATPG time as well as improving test efficiency. This thesis proposes a method to rapidly and correctly identify multi-cycle false paths, which are found in current and future multi-clock domain designs. For modern designs, false path identification at gate-level is no longer practical and this thesis proposes an identification methodology at RTL, which significantly reduces the number of paths to be examined. A case study shows that multi-cycle paths can indeed be identified at RTL, and that the

presence of these paths must be taken into account during test planning.

4. Thesis Organization

This thesis is organized as follows. This chapter (Chapter 1) introduces the concept of core-based system design as well as the problems facing SoC test engineers. It gives an overview of DFT for SoCs and discusses the problems pertaining to high power dissipation, overheating, and low test efficiency during test. It also summarizes the contributions of this thesis.

Chapter 2 targets wrapper design for multi-clock domain cores under a power constraint. For multi-clock domain cores, the possibility of clock skews during test must be considered. By using bandwidth-matching, clock-gating and domain partitioning, a method to design a power-aware wrapper architecture that solves the clock-skew problems while still minimizing the test time is presented. Experimental results shows that our proposed method is generally more effective than previous schemes, especially under tight power constraints.

Chapter 3 talks about thermal-aware TAM and wrapper design for SoCs with flexible TAM. It starts off with a discussion on the limitations of power-constrained testing as well as previous thermal-aware methodologies. It then discusses the simple thermal cost function developed to prevent unnecessary thermal simulations. It then proceeds to a detailed discussion of the test architecture design and test scheduling algorithm. The experimental results show that our method does indeed ensure that a test does not go beyond thermal-safety limits while still minimizing test application time.

Chapter 4 presents techniques to further improve the methodology presented in Chapter 3. More specifically, it studies the effectiveness of introducing test-reconfiguration, test set partitioning and interleaving, and bandwidth-matching to the test architecture design and test scheduling algorithm. To clearly show how these techniques can improve thermal-safety, the whole discussion is centered around SoCs with a fixed-TAM architecture. The new thermal cost function and expande4d scheduling algorithm is then discussed. Experimental results show that even under the restriction presented by the fixed TAM architecture, the new methodology enabled us to achieve greater decrease in test temperature compared

to previous methods.

Chapter 5 goes away from system-level test into high level test techniques. Specifically, it discusses the attractiveness of identifying multi-cycle false paths at RTL instead of at gate-level. It starts off with a discussion of the difficulties of multi-cycle path identification as well as the limitations of previous works. It then proceeds to give definitions of concepts such as RTL paths and multi-cycle RTL false paths. Next, the chapter presents two proposed transition propagation circuit models for which necessary conditions for false-path identification are developed and discussed. Finally, a case study is discussed to show how multi-cycle false paths can be identified at RTL as well as highlighting the fact that their existence must not be ignored and proper path classification is required to improve test efficiency.

Finally, Chapter 6 summarizes the whole thesis and concludes this work. Possible future works are also discussed.

Chapter 2

Power-Aware Wrapper Design for Multi-Clock Domain Cores

In this chapter, the proposed power-aware wrapper architecture for multi-clock domain cores is presented. Before delving further into the internals and control of the wrapper architecture, an introduction to multi-clock domain SoCs and IP-cores followed by a review of related works is presented. Next, the proposed architecture would be compared to several previously suggested works and the unique points are pointed out and explained further in this chapter. Then, the multi-clock-domain wrapper design problem, P_{MCDW}, is presented. This is followed by a discussion on the proposed 3-D bin packing algorithm used to determine the test schedule. Finally, this chapter ends with a discussion on experimental results as well as the conclusion.

1. Multi-clock Domain SoCs and IP-cores

Recent SoCs embed hundreds of memories, different types of logic, and other various IP's. Moreover, it has become possible to use multiple IP-cores with different operating frequencies to design multi-clock domain SoCs as shown in Figure 2.1. Aside from high test power, these designs can cause problems with synchronization with ATEs as well as inefficient use of test bandwidth which leads to wasted time and higher test costs. To solve these problems, [11] presented a wrapper and TAM design methodology using *Test Data Multiplexing* (TDM) to

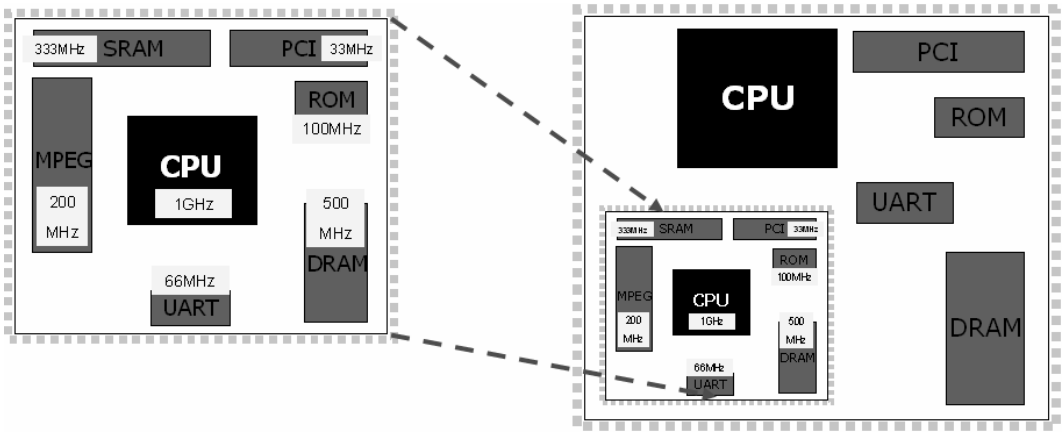


Figure 2.1. Multi-clock domain SoC reused as a multi-clock domain IP-core

enable bandwidth matching and its more efficient use during testing.

Modern IP cores, such as older generation multi-clock domain SoCs reused as cores of newer chips as seen in Figure 2.1, operate at various frequencies internally which have advantages such as reduced power and silicon area. Subsequently, these multi-clock domain cores present clock skew and at-speed testing (i.e. core operates at functional frequency during capture phase) problems which must be considered during testing. Clock skew problems arise from unsynchronized clock sources such as two signals of the same frequency but different clock trees that arrive out-of-sync to their destinations, thereby causing data corruption. Furthermore, power consumption and heat during test has become a big issue because of high switching activity during scan-shift operations as well as the possibility of more active components than expected during normal operation. To solve these problems, a 1500 compatible power-aware multi-clock domain core wrapper which partitions the IP core into smaller sub-groups and utilizes gated-clocks to control the start times of scan-shift operations and enable a more flexible and efficient use of the external bandwidth under a power constraint is proposed. A heuristic 3-D rectangular bin packing algorithm is also introduced which forms the basis of the proposed wrapper design method.

2. Review of Related Works

Most at-speed multi-clock domain testing techniques are based on *Built-in Self Test* (BIST) techniques [5, 6, 7] and utilize techniques such as programmable capture windows [5] and directly controlling separate launch and capture clocks [6] to solve the clock-skew problems while still allowing at-speed testing. An alternative to the BIST-based methods is inserting latches in-between clock domains but this is an invasive method which is not applicable to IP-protected cores.

The first non-BIST based multi-clock domain core wrapper design for IP-protected cores was proposed in [8]. In that approach, the core was divided into its clock domains, and each of the domains was referred to as a *virtual core*. Moreover, single frequency wrapper design was performed on each virtual core to assign a virtual wrapper to each of them. Finally, *virtual test bus* lines from each virtual core are connected to the external TAM via a de-multiplexing and multiplexing interface to synchronize the flow of the test data. The architecture used in [8] is shown in Figure 2.2. The method employs a single separate shift clock for all virtual cores, and it is multiplexed with the capture clock signals.

The test application process was divided into two phases, scan and capture phase. As shown in Figure 2.3, test data is scanned into the scan chains during the scan phase at a frequency which can be different from their functional frequencies. To avoid clock skew, the test enters the capture phase before or after the last bit of test data is scanned in. In the capture phase, all the scan chains are driven by their functional clocks to simulate normal operation. For multi-clock domain cores, a capture window can be designed to ensure proper capture of test response for all FFs in the various domains as shown in Figure 2.3.

In [9], the authors of [8] improved upon their design by allowing each virtual core to have a distinct shift frequency which allowed more flexible designs under power constraints. In both [8] and [9] all virtual cores are active at the same time and lowering shift frequencies which lead to large increases in test time might result under a very tight power constraint.

In [10], the use of gated-clocks to control the start and end times of the shift activity of each virtual core has been proposed. [10] used a 3-D bin packing algorithm which grouped virtual cores into *shelves* wherein all cores belonging to the same shelf become active at the same time and each shelf becoming active

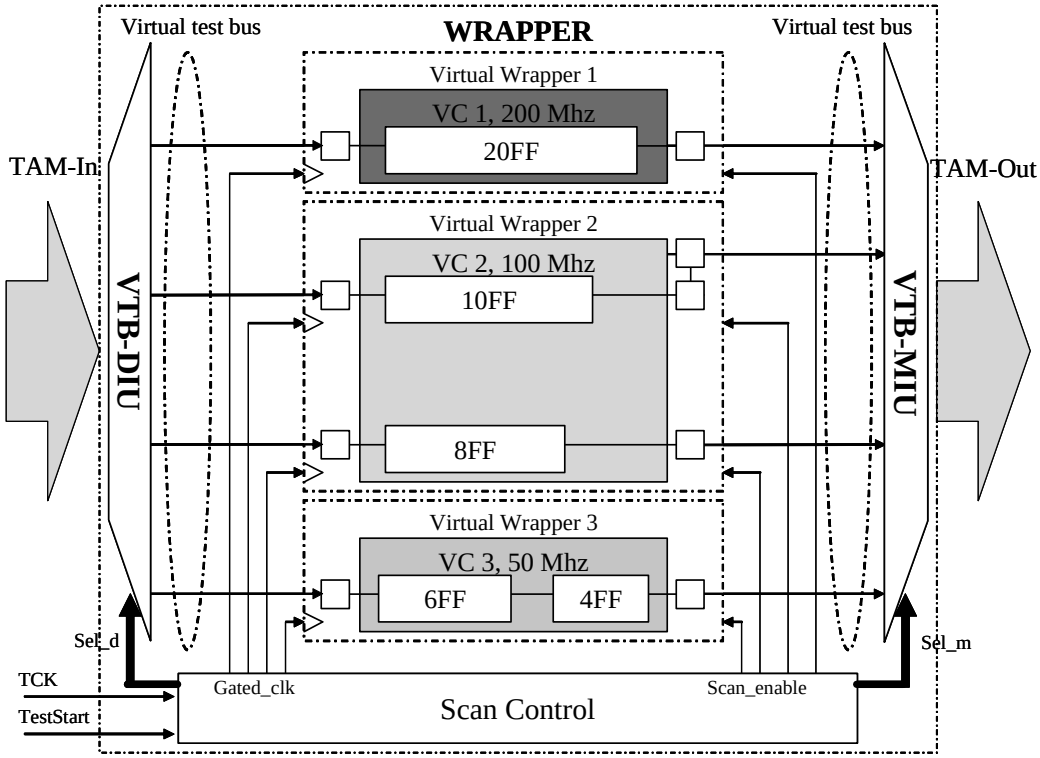


Figure 2.2. Multi-frequency wrapper architecture in [8]

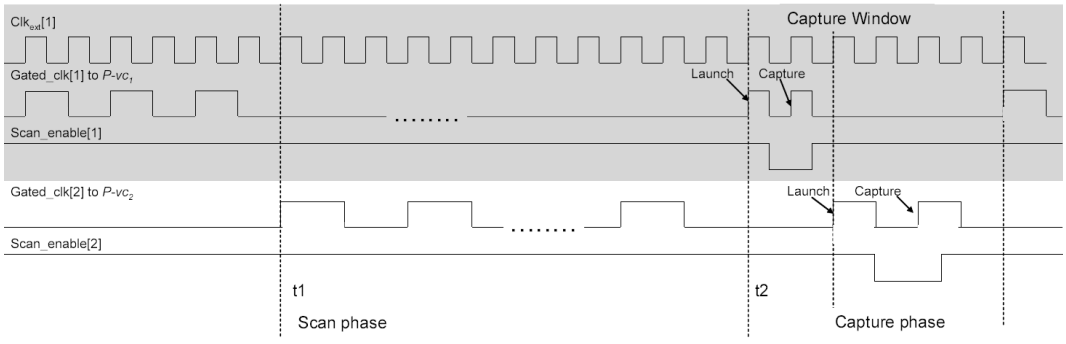


Figure 2.3. Timing diagram for the wrapper architecture in [8]

sequentially.

This research proposes a wrapper design method which uses a novel 3-D bin packing algorithm as its basis. The design allows for two things which improve upon the work in [10]: (i) each clock domain can be further partitioned into sub domains and (ii) a virtual core can be activated as soon as bandwidth is available. The proposed heuristic 3-D bin packing algorithm, unlike in [10], involves no post-processing and dynamically determines whether a clock domain should be divided or not during scheduling.

3. Multi-clock Domain Core Wrapper (MDCW) Architecture

As mentioned before, the proposed method allows the partitioning of clock domains into smaller groups, which would be referred to as *sub-domains*. Furthermore, unlike the previous works, *a virtual core can be made up of any combination of sub-domains from the same clock domain*. Furthermore, we assume that the following information is provided by the core designer:
P_{max}: Maximum allowed peak or average power dissipation (during scan)
N_C: Number of clock domains
N_{si}: Number of sub-domains for each clock domain $D_i(1 \leq i \leq N_C)$
For each sub-domain $S_{ij}(1 \leq j \leq N_{si})$ of clock domain D_i

- pi_{ij}: Number of primary input pins
- po_{ij}: Number of primary output pins
- bi_{ij}: Number of bidirectional I/O
- sc_{ij}: Number of internal scan chains and their lengths $l_{ijk}(1 \leq k \leq sc_{ij})$
- p_{ij}: Power dissipation at ATE frequency f_{ATE}

We now extend the definition of the virtual core from [8]. A group of one or more sub-domains from D_i is a virtual core $G_{ip}(1 \leq p \leq N_{gi})$. N_{gi} is the total number of possible combinations of the sub-domains of D_i. Furthermore, to

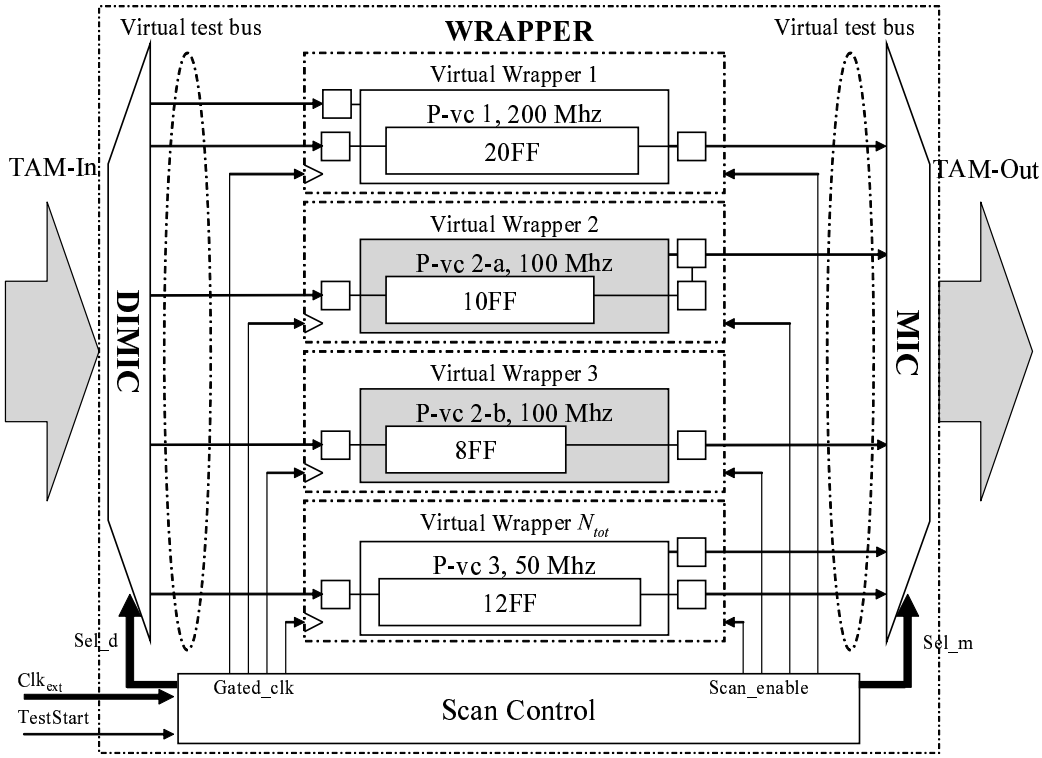


Figure 2.4. Proposed multi-clock domain wrapper

differentiate from the virtual core defined in [8, 9, 10], we would refer to virtual core in this paper as *P-vc*, short for *Partitioned virtual core*, and denotes the fact that it can represent one complete clock domain or just a part of it.

3.1 Wrapper Architecture

The basic architecture of the proposed multi-clock domain core wrapper is shown in Figure 2.4. The core is divided into smaller *P-vc*'s, each having its own virtual wrapper. These *P-vc*'s are connected to the external TAM via *virtual test bus* lines and a pair of *de-multiplexing* and *multiplexing interface circuits* (DMIC-MIC) that perform bandwidth matching and test data flow-control between the external TAM and the internal virtual test bus lines.

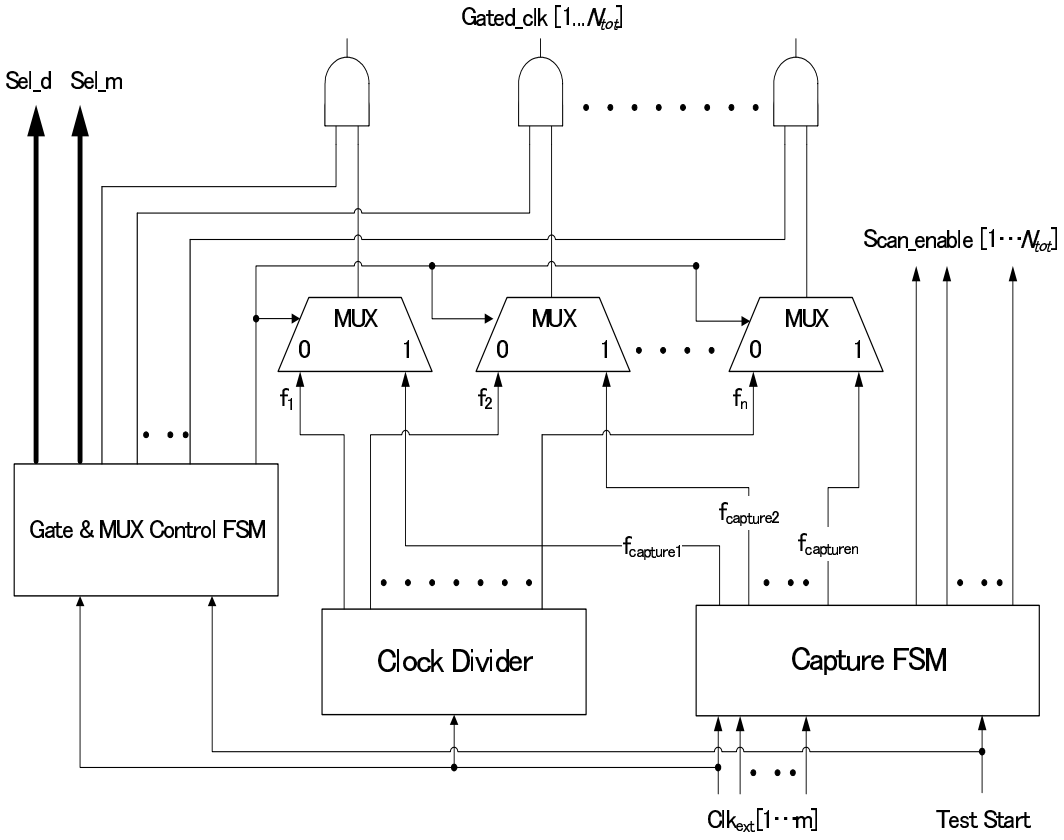


Figure 2.5. Proposed scan control block

3.2 Scan Control Block

The scan control circuitry for the proposed wrapper is shown in Figure 2.5. It is assumed that there are m external clock sources available, either from the ATE or on-chip PLL circuits. Furthermore, an external signal, *TestStart*, is triggered when the wrapper enters test mode. Normally, the test application process is divided into two phases, scan and capture phase. For this work, test data is scanned into the scan chains of each *P-vc* during the scan phase at a frequency that can be different from their functional frequencies. These scan clock signals are generated by the *Clock Divider* in Figure 2.5. We add a *gate and MUX control circuit* to control N_{tot}(total number of *P-vc*'s) gated clock signals during test as

well as switch to the functional clocks during normal operation. In the timing diagram shown in Figure 2.3, test data is scanned into $P\text{-}vc_1$ until time $t1$, when it becomes inactive and $P\text{-}vc_2$ starts the scan operation at a different frequency until time $t2$. At $t2$, the process begins the capture phase.

3.3 Comparison with Previous Works

Grouping the wrapper scan chains according to their clock domains prevents the effects of clock skew during this phase. To avoid clock skew at the capture phase, the test enters this phase before or after the last bit of test data is scanned in. In the capture phase, all the scan chains are driven by their functional clocks to simulate normal operation. For multi-clock domain cores, a capture window similar to that proposed in [8, 9], as shown in Figure 2.3, is used during this phase and the *P-vc*'s are activated in such a way that ensures inter-domain and intra-domain signals are properly captured for analysis. A comparison of schedules obtained in [9, 10] and by our method is shown in Figure 2.6. The use of gated clocks enables a more flexible and efficient test schedule compared to the concurrent shifting method in [9]. Unlike the shelf method in [10], our scheme allows partitioned testing with more flexibility and less wasted bandwidth during scheduling. Since we are using a capture window as proposed in [8, 9], this work will only focus on minimizing the shift time during the scan phase.

4. Problem Formulation

In this section, we formally present the multi-clock-domain wrapper design problem P_{MDCW}.

Problem P_{MDCW}: Given the test parameters for a multi-clock domain core C as described in Section 2 and the information below:
W_{ext}: TAM width allotted to the core
f_{ATE}: ATE frequency
$F = \{f_1, f_2, ..., f_m \mid f_j = 2 \times f_{j+1}, j \in 1, ..., m-1\}$: The set of allowed shift frequencies

determine the multi-clock domain wrapper design for C including:
N_{vi}: Number of *P-vc*'s $G_{ip} (1 \leq p \leq N_{vi})$ under domain D_i For each *P-vc* G_{ip} of

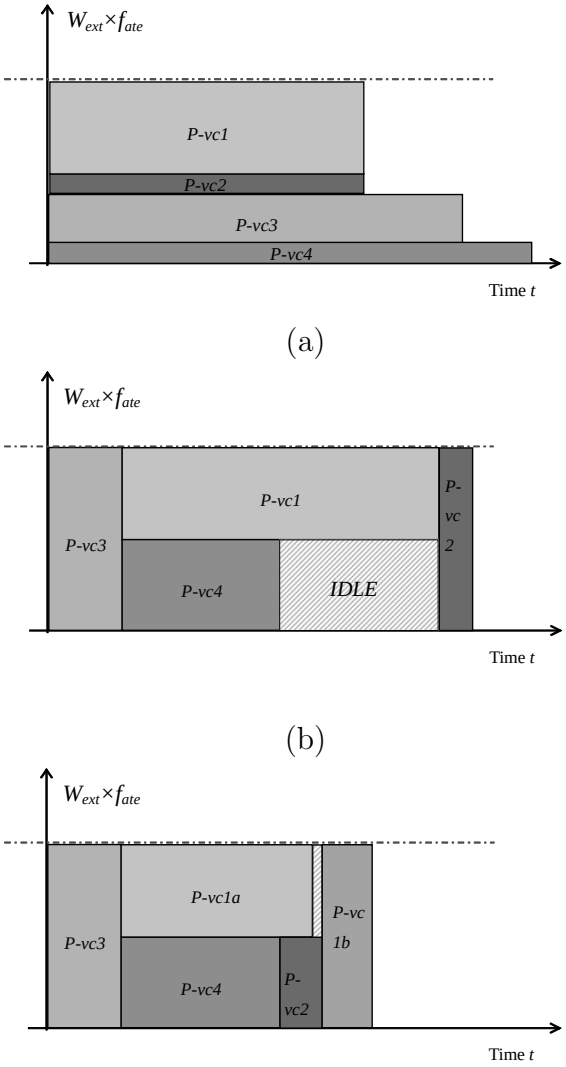


Figure 2.6. Comparison of (a) concurrent scan in [9] (b) shelf method from [10] (c) using proposed method

clock domain D_i

- $f_{ip} \in F$: Shift frequency
- w_{ip}: Number of virtual bus lines
- Its wrapper scan chain design and l_{ipmax}, which is the length of its longest wrapper scan chain
- t_{sip}: Scan-in start time

such that the following constraints are satisfied and the test application time is minimized:

1. The bandwidth used by all the active *P-vc*'s at any time t cannot exceed the total bandwidth coming from the ATE:

$$W_{ext} \times f_{ATE} \geq \max_{0 \leq t \leq T} \left(\sum_{i=1}^{N_C} \sum_{p=1}^{N_{vi}} a_{ipt} f_{ip} w_{ip} \right)$$
$$\text{where: } a_{ipt} = \begin{cases} 0, t < t_{sip} || t > (t_{sip} + l_{ipmax} \times \frac{f_{ATE}}{f_{ip}}) \\ 1, t_{sip} \leq t \leq (t_{sip} + l_{ipmax} \times \frac{f_{ATE}}{f_{ip}}) \end{cases} \tag{2.1}$$

2. The total power dissipation of all active *P-vc*'s at any time t cannot exceed the maximum allowed power dissipation P_{max}:

$$P_{max} \geq \max_{0 \leq t \leq T} \left(\sum_{i=1}^{N_C} \sum_{p=1}^{N_{vi}} a_{ipt} p_{ip} \times \frac{f_{ip}}{f_{ATE}} \right) \tag{2.2}$$

Since all the *P-vc*'s become active and inactive independently, the total scan-shift time for one test pattern can be computed from the *P-vc* with the latest end time as shown below:

$$T = \max_{1 \leq i \leq N_C} \left(\max_{1 \leq p \leq N_{vi}} \left(t_{sip} + l_{ipmax} \times \frac{f_{ATE}}{f_{ip}} \right) \right) \tag{2.3}$$

Each *P-vc* will have a distinct shift frequency and to simplify the clock generation circuitry, the ratio of usable frequencies is a two's exponent. Our initial experiments have shown that completely dividing cores into sub-domains would not always yield shorter test times so we developed a 3-D bin packing algorithm to determine how and when the domains should be partitioned to minimize the scan-shift time T while also optimizing the final number of *P-vc*'s.

5. Proposed Test Scheduling Algorithm

Similar to [4], rectangular 2-D bin packing have been extensively used to solve the test scheduling problem for embedded cores. For scheduling under a power constraint, it can be extended into a restricted 3-D bin packing problem where the length, width and height represent pin, peak power and total test time, respectively, of an SoC[13]. Specifically, a core (or in our case, *P-vc*), is represented as a cube whose length, width and height represents the alloted TAM width, power dissipation and test time, and when a cube overlaps in the time domain, they cannot overlap at any of the other two domains. In this research, instead of pin count or TAM width, the length represents the external bandwidth $BW_{ext} = W_{ext} \times f_{ATE}$ and each *P-vc* G_{ip} of an IP-core C can be represented by one cube among a set of permissible cubes. The cubes are packed into the 3-D bin until all the sub-domains S_{ij} of each domain D_i have been included in the packed *P-vc*'s while minimizing the total height.

In [13], which tackles scheduling for multiple cores in an SoC, the restricted 3-D bin packing algorithm for cubes representing allocated TAM width, power, and core test time has been shown to be *NP-Hard*. While similar, bandwidth, instead of TAM width, is represented in the bin packing problem for this research. The bandwidth used by each core is a function of allocated TAM and shift frequency. Therefore, if the shift frequency is kept constant on all the cores, the problem reduces to the problem in [13]. Consequently, this proves that our problem is also *NP-Hard* and this paper proposes a heuristic algorithm to solve the problem.

5.1 Initialization: Cube Creation and Ordering

To illustrate the various steps in the algorithm, the benchmark multi-clock domain IP core *hTCADT01*, first introduced in [9] and shown in Table 2.1, is used throughout the following sections. This IP core has seven clock domains and *sd#* denotes the sub-domain number, while P is the power dissipation of the sub-domains when shifting at 100MHz and is made equal to the sum of all the scan chain lengths Lsc_{ij} belonging to that sub-domain to simplify the setup.

P-vc's can be created to represent any combination of sub-domains belonging to the same clock domain. If a domain D_i has N_{si} sub-domains, then the total

Table 2.1. hCADT01 clock and sub-domain information

$sd\#$	f_{func} (MHz)	N_{in}	N_{out}	N_{bi}	P	N_{sc}	Lsc_{ij}
1.1	200	50	8	18	668	4	168, 168, 166, 166
1.2	200	25	8	18	652	4	163, 163, 163, 163
1.3	200	25	8	18	648	4	162, 162, 162, 162
1.4	200	9	8	18	604	4	151, 151, 151, 151
2.1	133	144	67	72	450	3	150, 150, 150
3.1	120	39	4	24	465	5	93, 93, 93 ,93 ,93
3.2	120	30	3	24	279	3	93, 93, 93
3.3	120	20	1	24	186	2	93, 93
4.1	75	61	10	36	657	3	219, 219, 219
4.2	75	50	21	36	657	3	219, 219, 219
5.1	50	87	200	36	1563	3	521, 521, 521
5.2	50	30	24	36	1042	2	521, 521
6.1	33	96	10	24	278	5	82, 81, 81, 17, 17
6.2	33	30	18	24	198	4	82, 81, 18, 17
6.3	33	20	40	24	100	2	82, 18
7.1	25	15	30	72	40	4	10, 10, 10, 10

number of possible *P-vc*'s from D_i is just the sum of all the possible combinations of S_{ij}.

$$N_{gi} = \sum_{j=1}^{N_{si}} {}_{N_{si}}C_j \tag{2.4}$$

Single frequency wrapper design such as in [3] is performed on all *P-vc*'s. In [3], given the alloted virtual test bus width, the I/O boundary scan cells and internal scan chains are connected into wrapper scan chains in such a way that the length of the longest wrapper scan chain, l_{ipmax}, is minimized. l_{ipmax} determines the test time for the core (Equation 2.7) and it can vary depending on the alloted test bus width. For example, the l_{ipmax} for domain 7 in Table 2.1 at $w_{ip} = 15$ and 3 is 10 and 48, respectively. At $f_{ip} = 100MHz$, this will give us test times of 0.10μsec.

and 0.48μsec., respectively. We denote the maximum number of virtual test bus lines that can be assigned to a *P-vc* as Vtb_{max}. Each *P-vc* $G_{ip}(1 \leq p \leq N_{gi})$ can have a maximum of Vtb_{max} possible wrapper designs and the same number of cubes are created. From the list of cubes of G_{ip}, the cube with the shortest test time regardless of power or bandwidth constraints is be selected as its ideal cube. It was proven in [10] that halving the shift frequency of the *P-vc* while doubling the virtual test bus width can result in either an increase in scan-shift time or no increase at all. We take advantage of this property to maintain the test time while still minimizing the power dissipation of the core. Each *P-vc* G_{ip} is expressed as a triplet $C_{ip} = \{BW_{ip}, p_{ip}, t_{ip}\}$, where $BW_{ip} = w_{ip} \times f_{ip}$ is the bandwidth of G_{ip}, w_{ip} is the virtual test bus width assigned to it, and $f_{ip} \in F$ represents its shift frequency. The power dissipation p_{ip} at f_{ip} can be expressed as:

$$p_{ip} = \frac{p_{ipmax} \times f_{ip}}{f_1} \tag{2.5}$$

where f_1 is the maximum allowed shift frequency. In this paper, we set $f_1 = f_{ATE}$ and p_{ipmax} is the power dissipation at f_1 as expressed below:

$$p_{ipmax} = \sum_{j=1}^{N_{si}} b_{ij} p_{ij} \tag{2.6}$$

where $b_{ij} = 1$ when sub-domain S_{ij} belongs to G_{ip} and $b_{ij} = 0$ if not, and p_{ij} is the power dissipation of S_{ij} at f_{ATE}. The minimum test time t_{ip} can be computed as follows:

$$t_{ip} = \frac{l_{ipmax}}{f_{ip}} \tag{2.7}$$

The ideal cubes of *P-vc*'s representing whole domains that satisfy the bandwidth and power restrictions are put into a master cube list L_M. The remaining ideal cubes not in L_M but satisfy the bandwidth and power constraints are then added to it. Finally, the ideal cubes of *P-vc*'s representing only one sub-domain is then added to L_M to ensure that all sub-domains are tested in the final schedule. Then, an area attribute $A_{ip} = BW_{ip} \times t_{ip}$ is also computed for each cube of *P-vc*'s representing single sub-domains. Since the bigger the area attribute, the harder

it is to pack, it follows that sub-domain groups which have big sub-domains must be prioritized by sorting L_M from the cube of the *P-vc* that has the member sub-domain with the biggest area attribute to the *P-vc* with the smallest one. If two cubes have the same sized sub-domains, then the overall area attribute of the cubes themselves are compared during sorting. After the list preparations, bin packing can be started from Step 1.

5.2 Step 1: Packing Domain Cubes

Before packing, the algorithm takes note of the current time in the schedule, denoted by a variable *curr_time*. Since the algorithm only divides domain *P-vc*'s when necessary, in this step the algorithm only looks at cubes representing whole domains in L_M until it finds a cube that has $p_{ip} \leq p_{avail}$ and $BW_{ip} \leq BW_{avail}$. p_{avail} is the available power and BW_{avail} is the available bandwidth, respectively. If a cube is found and packed into the bin, the algorithm checks what sub-domains belong to it and updates L_M by removing all cubes that has at least one member sub-domain equal to any of the sub-domains of the packed *P-vc*. It continues the above process until (a) $BW_{avail} = 0$ and/or $p_{avail} = 0$ or (b) if a proper cube cannot be found. Under condition (a), the algorithm looks among currently scheduled *P-vc*'s for the *P-vc* with the earliest test end time and sets *curr_time* equal to it and repeats Step 1. Under condition (b), the algorithm proceeds to Step 2. For the benchmark core *hTCAD01*[9] shown in Table 2.1 with external bandwidth $BW_{ext} = 1600$ and $P_{max} = 3000$, Step 1 packs the *P-vc* of domain no.5 ($P - vc$ that combines sub-domain 5.1 and 5.2 from Table 2.1) with power $p = 2605$, as shown in Fig. 2.7.

5.3 Step 2: Packing Sub-domain Group Cubes

Not finding a cube in Step 1 means that partitioning a domain is necessary. In Step 2 the algorithm only looks at the cubes not tried in Step 1, which represent sub-domain groups, until it finds a cube that satisfies $p_{ip} \leq p_{avail}$ and $BW_{ip} \leq BW_{avail}$. If a cube is found and packed into the bin, the algorithm again checks what sub-domains belong to it and updates L_M. Step 2 is repeated while there is available power and bandwidth or if a proper cube cannot be found. If BW_{avail}

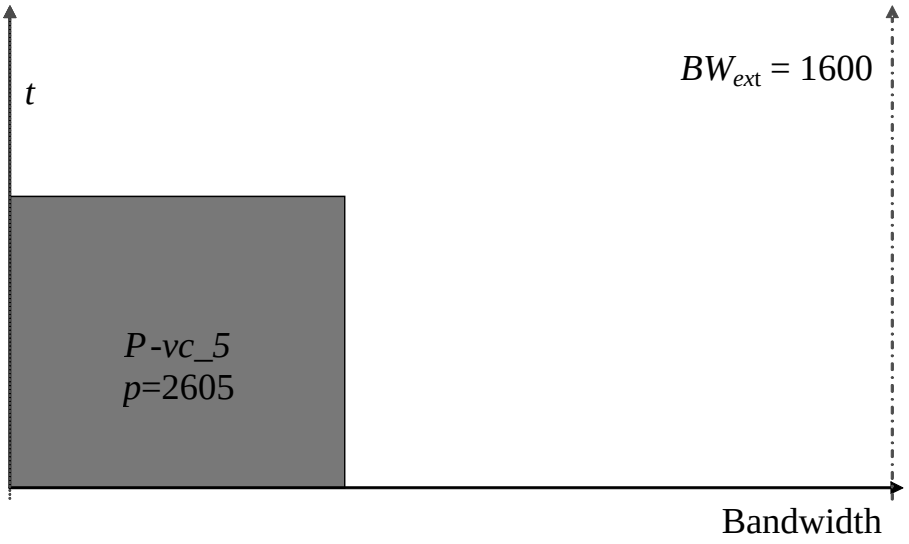


Figure 2.7. Virtual core of domain 5 packed in step 1

and/or p_{avail} become zero, *curr_time* is again updated and the algorithm goes back to Step 1. But if a proper cube cannot be found, the algorithm proceeds to Step 3. In Fig. 2.8, Step 2 is illustrated when the *P-vc* of sub-domain group $S_{61} \cup S_{63}$ of domain 6 is packed into the bin.

5.4 Step 3: Filling Idle Space by Decreasing Virtual Test Bus Lines

In Step 3, the algorithm searches for the packed cube using the biggest bandwidth and denotes its test end time as T_{endmax}. L_M is then traversed for a cube that satisfies p_{avail}. It then determines the new scan-shift time t_{ipnew} for the cube being tried given a new bandwidth $BW_{ipnew} = BW_{avail}$. Because of the limited selection of usable shift frequencies, choosing the next lowest f_k would automatically lead to a doubling of the scan-shift time. So for Step 3, the assigned virtual test bus width is decreased and t_{ipnew} is computed. The shift frequency is halved and the virtual test bus is doubled to minimize power as long as t_{ipnew} remains constant. If $t_{ipnew} \leq T_{endmax} \times (1 + dmax)$, then the cube is packed into the bin

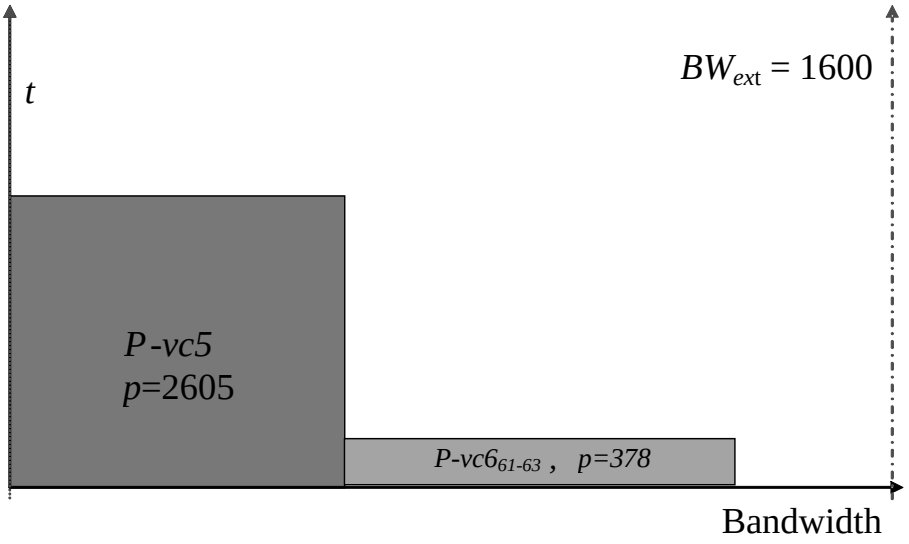


Figure 2.8. Virtual core of the combination of sub-domains S_{61} and S_{63} packed in step 2

and L_M is updated as before. $dmax$ is a heuristic value (in %) which expresses how much t_{ipnew} can go over T_{endmax}. The algorithm repeats Step 3 until there is no available bandwidth and/or power or no suitable cubes can be found. If BW_{avail} and/or p_{avail} become zero, $curr\_time$ is updated and the algorithm goes back to Step 1. If no suitable cubes were found, the algorithm proceeds to Step 4. For example, the ideal cube of domain no. 7 has f_{ip} = 100MHz, w_{ip} = 15, p_{ip} = 40 and t_{ip} = 0.10μsec. In Fig. 2.9, the idle bandwidth was 300MHz and for *P-vc7*, w_{ip} was first reduced to 3 at f_{ip} = 100MHz. Consequently t_{ip} increased to 0.48μsec. While keeping t_{ip} constant, we are able to increase the w_{ip} to 12, while decreasing the shift frequency f_{ip} to 25MHz and the power p_{ip} was lowered to 10 before packing the cube into the bin.

5.5 Step 4: Filling Idle Space by Decreasing Shift Frequency

Reaching Step 4 means that no cube satisfies p_{avail}. There is no choice but to lower the shift frequency of a *P-vc* to fit the available idle space in the bin. The

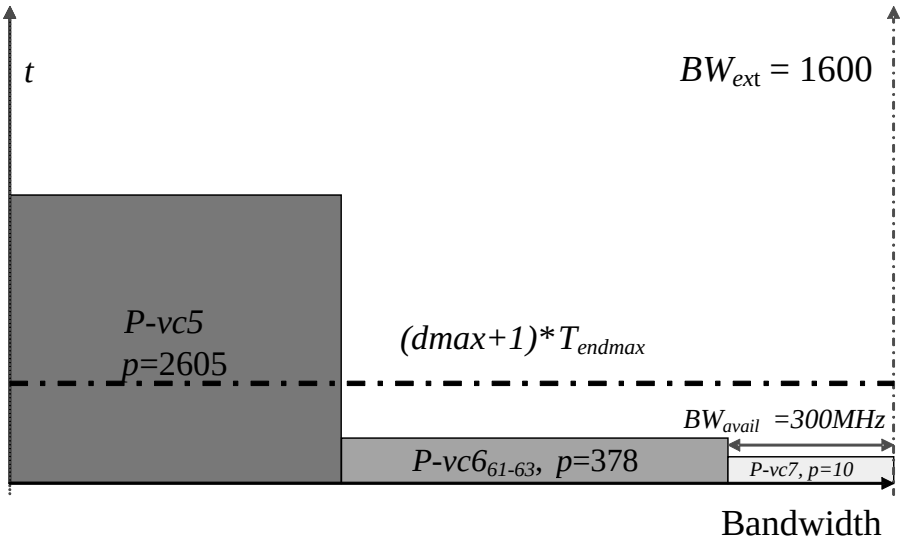


Figure 2.9. Virtual core of domain 7 with modified virtual test bus width and packed in step 3

algorithm determines T_{endmax} and makes a copy of L_M called L_{tmp}. Then, the shift frequencies of all cubes remaining in L_{tmp} are lowered until their power is less than or equal to p_{avail}. For each cube, the new scan-shift time t_{ipnew} is computed given a new bandwidth $BW_{ipnew} = BW_{avail}$. The shift frequency is halved and the virtual test bus is doubled to minimize power as long as t_{ipnew} remains constant. Similar to Step 3, the algorithm looks for a cube that satisfies $t_{ipnew} \leq T_{endmax} \times (1 + emax)$ and closest to T_{endmax}, as we have found during experimentation that this gives better results than simply packing the first cube that satisfies the first condition. The cube is packed into the bin and L_M is updated as before. Step 4 is repeated until no cubes are found or until BW_{avail} and/or p_{avail} becomes zero. The algorithm again updates *curr_time* and goes back to Step 1. Note that $emax$ is a heuristic value ,independent of $dmax$, which expresses how much t_{ipnew} can go over T_{endmax}. Steps 1 through 4 are repeated until L_M is empty. In Fig. 2.10, although there is a large BW_{avail}, p_{avail} is only 302 (see Fig. 2.11(b) so the f_{ip} of *P-vc2* is decreased to 50MHz, and this leads to a decrease in $p_{ip} = 225$ while w_{ip} remained the same and t_{ip} doubled to

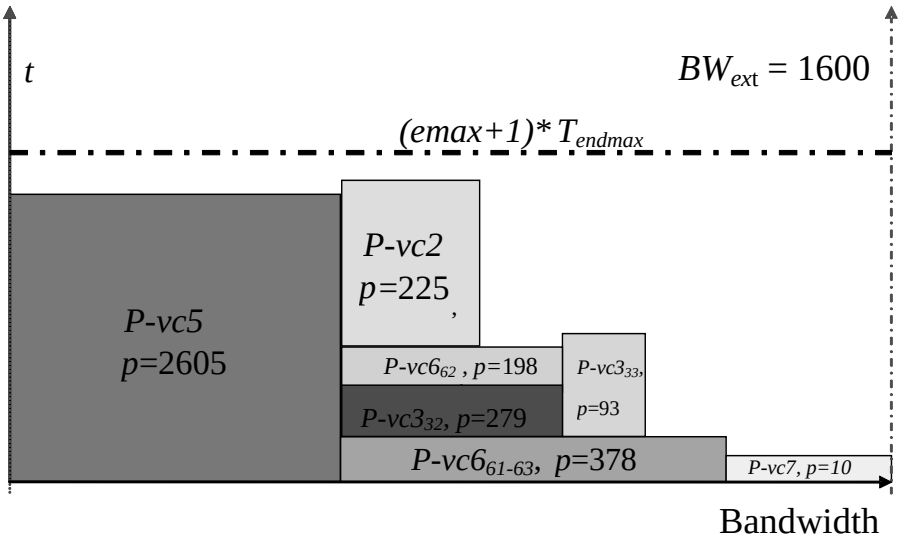
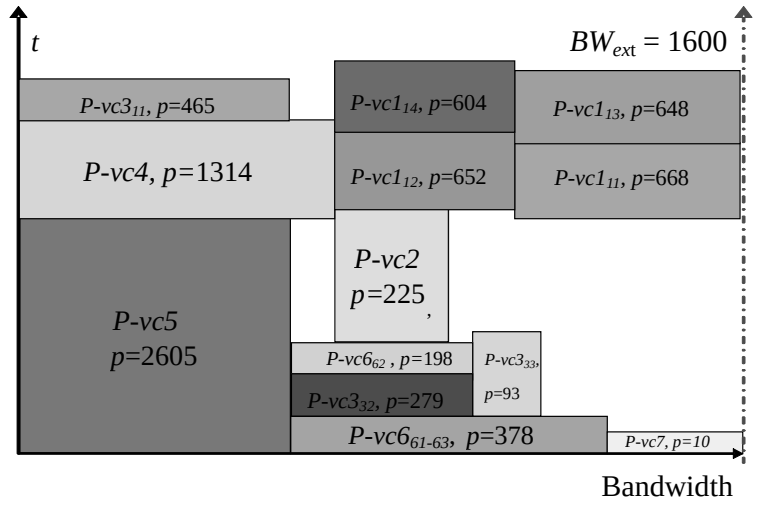


Figure 2.10. Packing result when domain 2 is packed in step 4

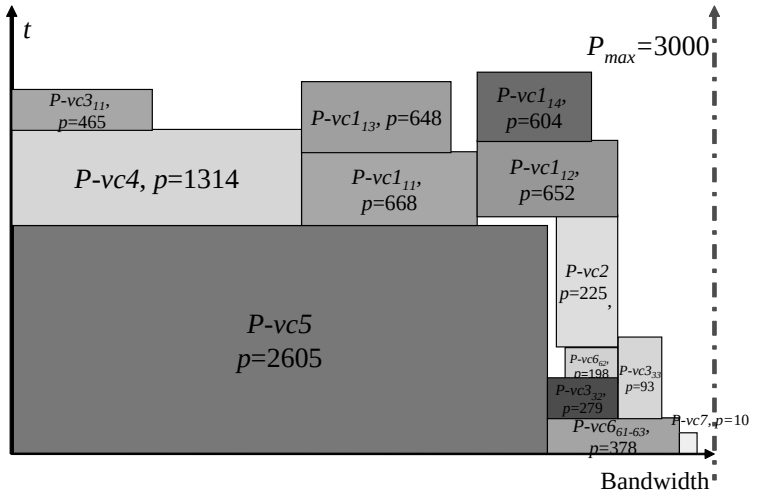
3.00μsec. Figure 2.11 shows the finished schedule separated into two 2-D graphs ((a)bandwidth vs. time and (b)power vs. time) to make it easier to see that there are times when available bandwidth could not be utilized due to very low available power. Also note that by partitioning the clock domains, we increase the chance that a *P-vc* can be scheduled (due to the presence of smaller cubes) during instances of low available power and/or bandwidth.

6. Experimental Result

The experiments were done using the benchmark multi-clock domain IP core *hTCADT01* which has seven clock domains, and we assumed that each domain can be further partitioned into sub-domains as shown in Table 2.1. In the table, *sd#* denotes the sub-domain number, f_{func} is the functional frequency, N_{in}, N_{out}, N_{bi} and N_{sc} are the number of inputs, outputs, bidirectional I/O and scan chains in the specific sub-domains respectively, Lsc_{ij} is the length of each scan chain and P is the power dissipation of the sub-domains when shifting at 100MHz and is made equal to the sum of all the scan chain lengths Lsc_{ij} belonging to that



(a)



(b)

Figure 2.11. Finished test schedule for hTCAD01 at $BW_{ext} = 1600$ (vs. time (a)) and $P_{max} = 3000$ (vs. time (b))

sub-domain to simplify the setup.

The experiment was conducted under four different power constraints P_{max}: 1500, 3000, 4500 and ∞. The maximum allowed frequency f_1 is 100MHz, which is equal to f_{ATE} to synchronize the internal shift frequencies with the ATE. To see the effectiveness of the proposed method, the resulting shift times denoted by T_{new} are compared to the results from [9], marked T[9], and from [10], marked T[10], in Tables 2.2-2.5. All times are in microseconds. $\%diff[9]$ and $\%diff[10]$ are the differences in percentage with respect to [9] and [10] respectively, given by the equation below.

$$\%diff[target] = \frac{T_{new} - T[target]}{T[target]} \times 100\% \tag{2.8}$$

During the experiments, $dmax$ and $emax$ were independently varied from 0% to 200% to find the optimal combination that yields a minimum scan-shift time. The experiments were done using a Sun Fire V490 1.35GHz UltraSPARC IV workstation with 32GB memory and the 40,000 looped reruns of the program didn't take more than 1sec. of CPU time.

At the tightest power constraint of $P_{max} = 1500$, our algorithm was able to decrease the shift time with a maximum of 24.42% compared to [9]. The average gain in test time, as P_{max} is decreased, increases dramatically from 8.69% at $P_{max} = \infty$ to 18.36% at $P_{max} = 1500$. This can be attributed to the fact that wider W_{ext} and lower P_{max} makes partitioning the domains more effective because of the extra freedom it gives during scheduling. Compared to [10], the trend is different and there is an almost constant average gain of around 4% across all power constraints and a maximum gain of 14.25% at $P_{max} = 3000$. Small differences in time (0-1%) are attributed to discrepancies in rounding-off among the programs used and makes them negligible, so our algorithm matches or exceeds [10] in 90% of the cases.

In [8, 9], the area increase due to the demultiplexing-multiplexing circuitry, scan-control modules and other necessary logic to implement the multi-clock domain wrapper was stated to be less than 10% the area size taken by the IEEE 1500 wrapper and other scan logic. Since our approach only requires a slight modification of the scan control circuitry in [8, 9], it is safe to assume that the added overhead would be minimal. Furthermore, as manufacturing processes be-

Table 2.2. Comparison of scan-shift time for hCADT01 under P_{max}=1500

$P_{max} = 1500$

W_{ext}	T[9]	T[10]	T_{new}	$\%diff$[9]	$\%diff$[10]
16	20.84	16.9	15.75	24.2	6.80
15	20.84	16.9	15.75	24.2	6.80
14	20.84	16.9	15.75	24.2	6.80
13	20.84	16.9	15.75	24.2	6.80
12	20.84	16.9	15.85	23.94	6.21
11	20.84	16.97	15.85	23.94	6.60
10	20.84	17.07	16.68	19.96	2.28
9	20.84	17.47	15.91	23.66	8.93
8	20.84	17.57	16.9	18.91	3.81
7	20.84	19.29	18.92	9.21	1.92
6	20.84	19.6	19.63	5.81	-0.15
5	25.04	23.38	23.19	7.39	0.81
4	29.76	26.16	25.34	14.85	3.13
3	41.68	34.06	34.08	18.23	-0.06
2	59.88	51.61	50.81	15.15	1.55
1	116.04	100.7	98.69	14.95	2.00

come smaller and transistor count becomes higher, the probable DFT overhead becomes more and more negligible in light of the possible gains in test application time. Also, the added flexibility of domain partitioning and partitioned test scheduling would be of greater benefit as designers start to re-use older generation multi-clock domain circuits as IP-cores in newer, more complex designs.

7. Concluding Remarks

This research has presented a novel method of designing a test wrapper for multi-clock domain IP-cores by effectively utilizing clock domain partitioning and gated-clocks. With minimal hardware overhead for gated-clock control, we

Table 2.3. Comparison of scan-shift time for hCADT01 under P_{max}=3000

$P_{max} = 3000$

W_{ext}	T[9]	T[10]	T_{new}	$\%diff$[9]	$\%diff$[10]
16	10.42	9.13	8.89	14.68	2.63
15	10.42	10.42	9.02	13.44	13.44
14	10.42	10.42	9.02	13.44	13.44
13	10.42	10.53	9.03	13.34	14.25
12	10.42	10.53	10.32	0.96	1.99
11	11.62	11.12	10.52	9.47	5.40
10	12.08	11.24	10.88	9.93	3.20
9	13.00	12.56	12.14	6.62	3.34
8	14.48	13.50	13.18	8.98	2.37
7	17.76	16.57	15.14	14.75	8.63
6	20.84	17.19	17.19	17.51	0.00
5	23.24	21.81	20.64	11.19	5.36
4	29.76	26.15	25.52	14.25	2.41
3	38.36	34.06	34.08	11.16	-0.06
2	58.02	50.36	50.48	13.00	-0.24
1	116.04	98.44	98.63	15.00	-0.19

have dramatically improved upon earlier methods which concurrently activate all the domains during test, especially under tight power constraints. Furthermore, the division of clock domains enabled us to give better results than the previous methods which also utilized gated-clocks. Note that while actual hardware overhead was not fully discussed in this thesis, the underlying circuitry that would be needed to realize the proposed wrapper architecture does not differ greatly from previous works. With regards to the DFT area overhead, it was stated in [8, 9] that the area of the scan control block, which is dependent on the capture window complexity and the number of virtual cores, is usually less than 10% the area size taken by the 1500 wrapper and other scan logic. Since our approach only requires a slight modification of the scan control circuitry in [8, 9], it is safe to

Table 2.4. Comparison of scan-shift time for hCADT01 under P_{max}=4500

$P_{max} = 4500$

W_{ext}	T[9]	T[10]	T_{new}	$\%diff$[9]	$\%diff$[10]
16	7.44	6.94	6.88	7.53	0.86
15	8.76	8.33	7.53	14.04	9.60
14	8.88	8.79	7.81	12.05	11.15
13	10.42	8.93	8.53	18.14	4.48
12	10.42	9.75	9.44	9.40	3.18
11	10.42	9.75	10.18	2.30	-4.41
10	11.62	10.58	11.2	3.61	-5.86
9	12.78	11.85	11.85	7.28	0.00
8	14.88	13.50	13.36	10.22	1.04
7	15.63	15.43	15.09	3.45	2.20
6	19.20	17.19	17.19	10.47	0.00
5	23.24	21.81	20.64	11.19	5.36
4	29.01	26.15	25.52	12.03	2.41
3	38.36	34.06	34.08	11.16	-0.06
2	58.02	50.36	50.48	13.00	-0.24
1	116.04	98.44	98.63	15.00	-0.19

assume that the added overhead would be minimal. Furthermore, as manufacturing processes become smaller and transistor count becomes higher, the probable DFT overhead becomes more and more negligible in light of the possible gains in test application time. Also, the added flexibility of domain partitioning and partitioned test scheduling would be of greater benefit as designers start to re-use older generation multi-clock domain circuits as IP-cores in newer, more complex designs.

Table 2.5. Comparison of scan-shift time for hCADT01 under $P_{max} = \infty$

$P_{max} = \infty$

W_{ext}	T[9]	T[10]	T_{new}	$\%diff$[9]	$\%diff$[10]
16	7.44	7.53	6.88	7.53	8.63
15	7.49	8.33	7.53	-0.53	9.60
14	8.88	8.79	7.81	12.05	11.15
13	9.59	8.79	8.53	11.05	2.96
12	10.42	9.15	9.44	9.40	-3.17
11	10.42	9.75	10.18	2.30	-4.41
10	11.62	10.58	11.2	3.61	-5.86
9	12.78	11.83	11.85	7.28	-0.17
8	14.88	13.5	13.36	10.22	1.04
7	15.63	15.43	15.09	3.45	2.20
6	19.18	17.19	17.19	10.38	0.00
5	23.24	21.81	20.64	11.19	5.36
4	29.01	24.84	25.52	12.03	-2.74
3	38.36	34.06	34.08	11.16	-0.06
2	58.02	50.36	50.48	13.00	-0.24
1	116.04	98.44	98.63	15.00	-0.19

Chapter 3

Thermal-Aware Test Access Mechanism and Wrapper Design Optimization for SoCs with Flexible TAM

As feature sizes and frequencies of newer System-on-Chips scale much faster than operating voltages, not only power densities but also heat densities will experience considerable increase. Furthermore, the problem of overheating becomes much larger during testing when beyond normal switching activities occur due to the need for concurrently testing cores to shorten test time. Overheating can lead to problems such as increased leakage power and even permanent chip damage. For every 20^oC rise in temperature, there is approximately a 5-6% increase in interconnect delay timing [36]. These timing uncertainties can result in further yield loss. Traditionally, simply using better packaging and cooling methods would suffice but this has become increasingly difficult and expensive. To reduce packaging cost, packages have increasingly been designed for the worst case *typical* application [22, 23] and the cost of cooling during test application has become very prohibitive.

For SoCs, test planning usually involves the design of a test data delivery method (TAM), and the use of *wrappers* which isolate cores under test. While several approaches to optimize wrapper designs for single frequency embedded

Table 3.1. Max. temperatures of p93791 under various power constraints

p93791	$TAM = 32$		$TAM = 64$	
P_{max}	$maxT(^oC)$	$TAT(cycles)$	$maxT(^oC)$	$TAT(cycles)$
13000	121.43	1105893	115.24	634685
17000	115.44	1033179	127.91	566076
21000	143.78	994803	110.66	538301
25000	127.33	975528	130.09	517541
∞	157.25	955989	123.49	523730

core test [1, 2] have been proposed, Iyengar et al. [3, 4] integrated the process into one wrapper and TAM co-optimization algorithm. Up to now, limiting power consumption during test has been the main method of temperature control, and test scheduling under power constraints have been considered in [4, 12, 13, 17].

Because of the non-uniform spatial power distribution across the chip, limiting the maximum chip-level power dissipation is not effective in reducing and avoiding localized heating (called *hot spots*) which occurs faster than chip-wide heating [19, 22, 23] as shown in Table 3.1. In Table 3.1, the maximum test temperatures, $maxT$, do not scale with power constraints P_{max} for the SoC p93791 using the power-constrained method in [4]. Furthermore, power-constrained test scheduling does not allow further exploration of schedule variations with the same test peak power. For the benchmark SoC d695 with a layout shown in Figure 3.1, two schedules having the same peak power values can have different peak temperatures, as shown in Figure 3.2. The peak temperatures for the three hottest cores, c5, c6, and c10 are indicated and the maximum temperature for c5 varies from 89.6^oC to 77.2^oC simply by changing its allocated TAM width from 32 to 31 bits.

In this chapter, we propose a design framework which integrates the TAM /wrapper co-optimization process with a thermal-aware test scheduling algorithm. The main contributions of this research are as follows:(1) consider a different cycle-accurate power profile per wrapper configuration for more realistic results, (2) resent a simplified thermal cost function and develop a test scheduling algo-

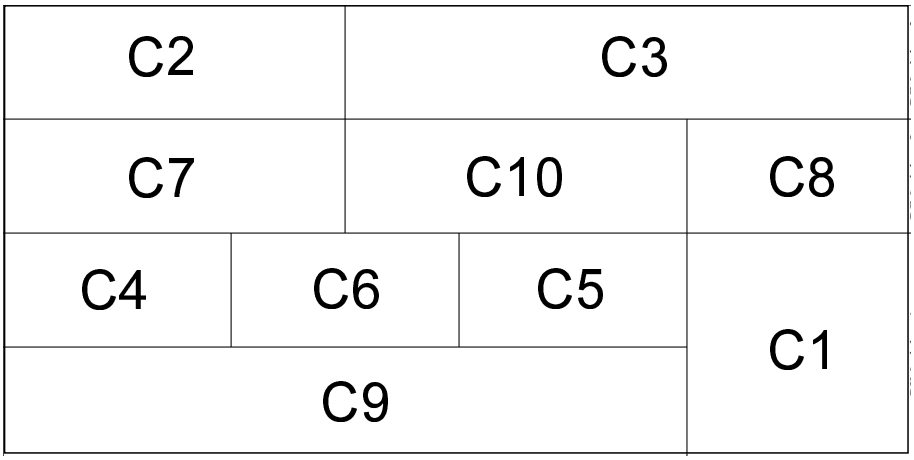


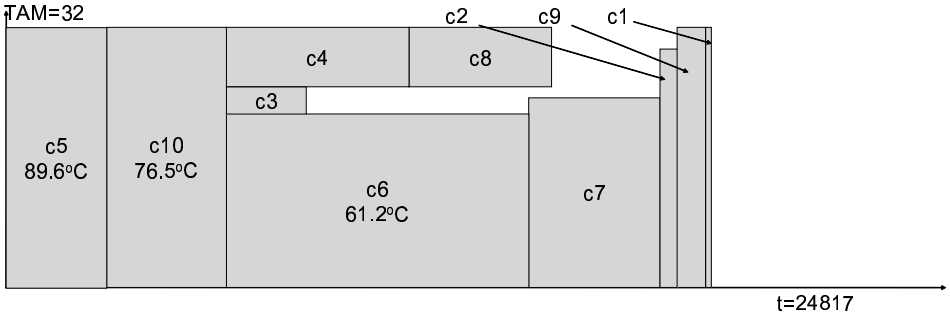
Figure 3.1. Hand-crafted layout of SoC d695

rithm to minimize the overall test time while satisfying temperature constraints, (3) show that for the ITC'02 SoC benchmarks [18], even a small increase in test time can yield a considerable decrease in test temperature as well as the possibility of further lowering temperatures beyond those achieved using traditional power-based test scheduling.

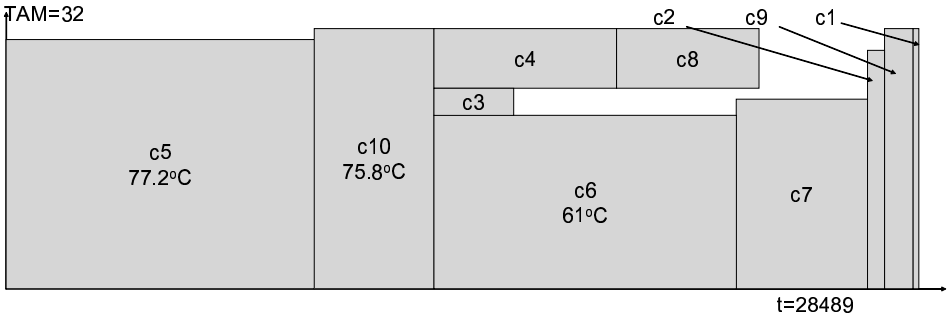
The rest of this chapter is organized as follows. A review of related works is given in the following section. The motivation for this work is then discussed. We then delve into the proposed TAM/wrapper co-optimization algorithm and the proposed test scheduling algorithm. The final two sections gives the experimental results and concluding remarks, respectively.

1. Review of Related Works and Motivation

Rosinger et al. [19] first proposed using a thermal model as a guide to test scheduling instead of a chip-level power constraint. They used the RC-equivalent micro-architecture thermal model from [22, 23, 24] which in turn makes use of the well-known duality between heat transfer and electrical phenomena: *heat can be described as a current passing through a thermal resistance and leading to a temperature difference analogous to a voltage* [22]. More specifically, [19] only



(a)



(b)

Figure 3.2. Two possible schedules at peak power=1600 switches, (a) $maxT = 89.6^oC$, (b) $maxT = 77.2^oC$

considered the lateral flow of heat away from an active core by reducing a chip into a network of thermal resistances as shown in Figure 3.3. The same thermal resistance network model is used in this work. The proposed test scheduling algorithm in [19] uses a test compatibility graph as its basis and cores are grouped into test sessions which are applied sequentially.

In [20], Liu et al. defines a "hot spot" as a core whose temperature is substantially higher than the average temperature over all cores. They proposed two algorithms which try to spread heat more evenly over a chip via layout information and a progressive weighting function, respectively. For this work, we define "hot spot" as any core which exceeds the thermal constraint during test. Thus, a core can be scheduled even if its temperature is much higher than its surrounding cores unlike in [20].

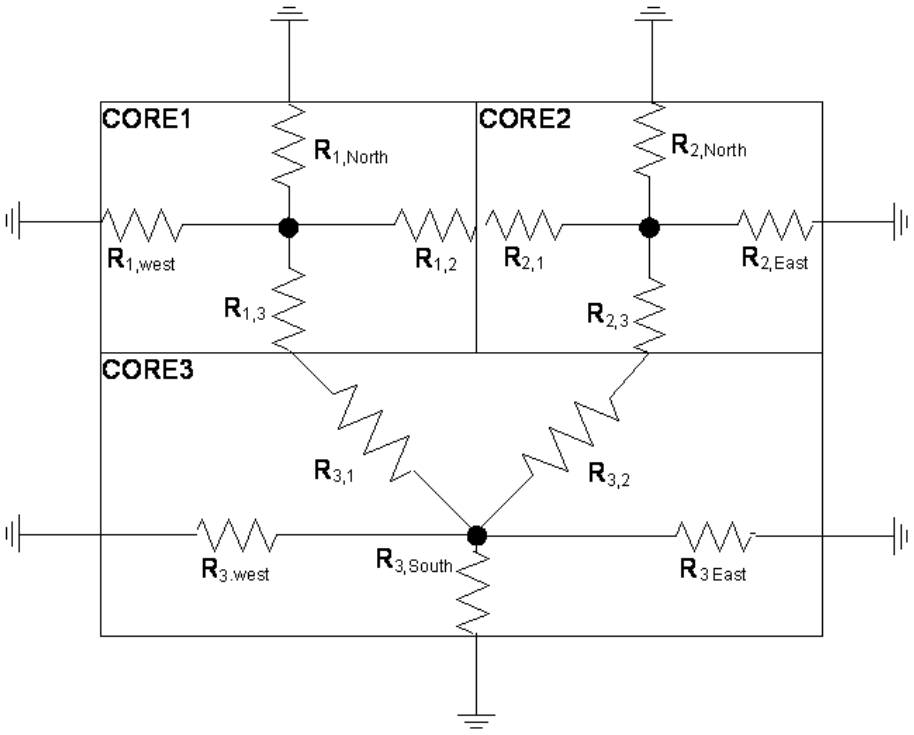


Figure 3.3. Lateral thermo-resistive model [19]

In [21], He et al. proposed using test partitioning and interleaving to allow hot cores to cool off while freeing the test resources to test other cores and avoid overheating.

For all previous methods, only a single fixed power value per core was considered and steady-state temperatures were used as temperature upper bounds [19]. However, this is not realistic, as shown in Table 3.2 where the peak temperature of test schedules using static average power T_{pavg} during thermal simulation are usually less than cycle-accurate values T_{real}, while maximum temperatures using peak power values T_{peak} are usually much higher and can be considered pessimistic.

Furthermore, the choice of using a single fixed power profile per core is also not realistic. From our experiments, we found that higher TAM widths (therefore, shorter test time) can yield lower maximum temperatures despite having higher

Table 3.2. Max. temperatures of d695 under various power constraints using different power models

d695	$TAM = 24$		
P_{max}	$T_{real}(^oC)$	$T_{pavg}(^oC)$	$T_{peak}(^oC)$
1600	99.64	90.14	345.68
1800	103.80	91.15	409.58
2000	106.84	93.52	424.86
2200	111.74	103.75	479.03
2400	104.94	93.60	421.48

peak power values. This is shown in Figure 3.4, where the temperature profile, as well as the peak temperature of core 5 of d695 varies with TAM width. This can be attributed the varying power profile per TAM configuration [17] as well as the RC characteristic of temperature rise: if a test can finish before the temperature curve reaches steady state, the capacitance can have a "filtering" effect on the maximum temperature values. Thus, test time must also be considered when deriving a thermal model or thermal cost function as discussed in the next section.

Finally, flexible TAM-width and partitioned testing were also outside the scope of [19] and [21]. To the best of our knowledge, this is the first work which attempts to integrate TAM/wrapper co-optimization and test scheduling under a thermal constraint using cycle-accurate power profile per wrapper configuration for more realistic temperature simulation.

2. Problem Formulation

In this section, we formally present the TAM/Wrapper co-optimization and test scheduling problem P_{TWOP}.

Problem P_{TWOP}: For an SoC S, given:

W_{ext}: external TAM width allocated to the SoC

N_C: Number of cores

$Temp_{max}$: maximum allowed temperature during test

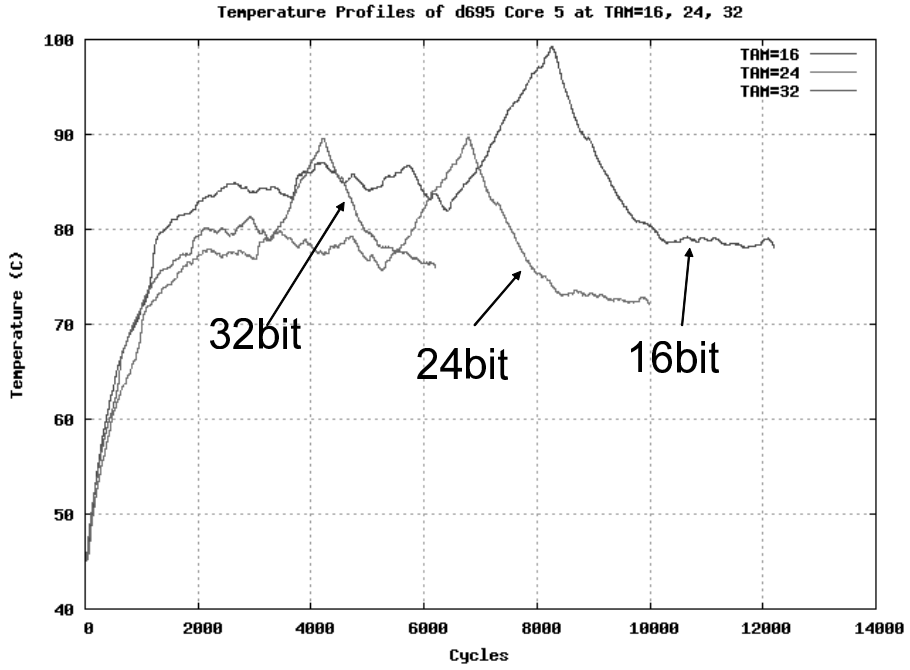


Figure 3.4. Varying temperature profiles per TAM configuration for core 5 of d695

For each core $C_i (1 \leq i \leq N_C)$ of SoC S
- $Wset_i$: number of usable wrapper configurations
- For each wrapper configuration $w_{ij} (1 \leq j \leq Wset_i)$

  - TAM_{ij}: alloted TAM width
  - P_{ij}: power profile
  - TAT_{ij}: test application time

Determine the following output:
For each core $C_i (1 \leq i \leq N_C)$ of SoC S
- w_{fi}: assigned final wrapper configuration

  - TAM_{fi}: final alloted TAM width

- t_{starti}: test start time
- t_{endi}: test end time

and minimize the overall test time of S such that the total number of TAM used at any given time does not exceed W_{ext} and temperatures do not exceed $Temp_{max}$.

Since we cannot ignore per-cycle power values and their effects on temperature, each wrapper configuration is given a different power profile as explained in the previous section.

As explained in previous chapters, rectangular 2-D bin packing has been extensively used to solve the test scheduling problem for embedded cores. Each wrapper configuration of a core is represented by a rectangle whose width and height represents test application time and TAM width, respectively. The rectangles are packed into a bin with unbounded width, representing overall test time, and bounded height representing external TAM width. The aim is to find the optimal way of packing the rectangles such that overall test time (e.g. bin width) is minimized. For scheduling under a power constraint, it can be extended into a restricted 3-D bin packing problem where the length, width and height represent total test time, peak power and TAM width, respectively, for an SoC core. For this paper, previous bin-packing algorithms cannot be directly applied since we cannot simply add the various temperatures of the cores to obtain the overall temperature of the SoC. Furthermore, since it has been shown that the bin packing problem is *NP-Hard*, this paper proposes a heuristic algorithm to solve the problem.

3. Thermal Cost Function

Since temperature cannot be handled in the same simplistic and direct way as power(i.e. simple superposition is inapplicable), we need a thermal model and cost function which can effectively and simply express the heating phenomena without the need for data from thermal simulations.

The results in [19] prove that there exists a positive correlation between heat and heat dissipation paths represented by lateral thermal resistances. Thus, we have chosen to use lateral thermal resistance as one of the basis for our model and

cost function, with necessary modifications of assumptions from previous works so the model can better approximate heating patterns during testing. From Equation 3.1, the thermal resistance R_{TH} between two adjacent bodies is directly proportional to the thickness of the heat source t and inversely proportional to the cross-sectional area A of the destination across which the heat is being transferred and the thermal conductivity k of the material per volume unit.

$$R_{TH} = t/kA \tag{3.1}$$

First, similar to [19], it is assumed that heat transfer between two cores tested concurrently is negligible and thermal resistances between these cores are removed as shown in Figure 3.5, where we are left with lateral resistances in parallel for core 1 and core 2. Since the thermal resistances of a core C_i are in parallel to each other, the total thermal resistance $Rth_{TOT,i}$ can be computed as follows:

$$Rth_{TOT,i} = \frac{1}{\sum_{j=1}^{N} 1/R_{i,j}} \tag{3.2}$$

where N is the total number of thermal resistances $R_{i,j}$ of C_i. Note that removing a thermal resitance from the network increases the total thermal resitance, which reflects the fact that there are fewer paths for heat to escape to and this reflects a higher maximum temperature for the core.

The assumption made in [19] that inactive cores are thermally grounded and do not heat up is not realistic unless ample time is given for tested cores to cool down before the next test session, as shown in Figure 3.6, where c5 can increase the temperatures of its inactive peripheral cores by as much as 10^oC for c10. Obviously this is not practical because of the required increase in idle time. Furthermore, our experiments show that the temporal dimension, more specifically, the test length as well as the order in which cores are tested can greatly affect the maximum temperature of the next core to be tested as shown in Figure 3.7 where the peak temperature of core 5 increases by 7^oC when core 10 is tested right before it (Fig. 3.7b) compared to the opposite sequence (Fig. 3.7a). Thus, when a core is about to be tested, the lateral resistances to cores whose test has just ended are also removed from the total lateral resistance. For example, if core 2 is tested right after core 1 in Figure 3.3, then $R_{2,1}$ is removed.

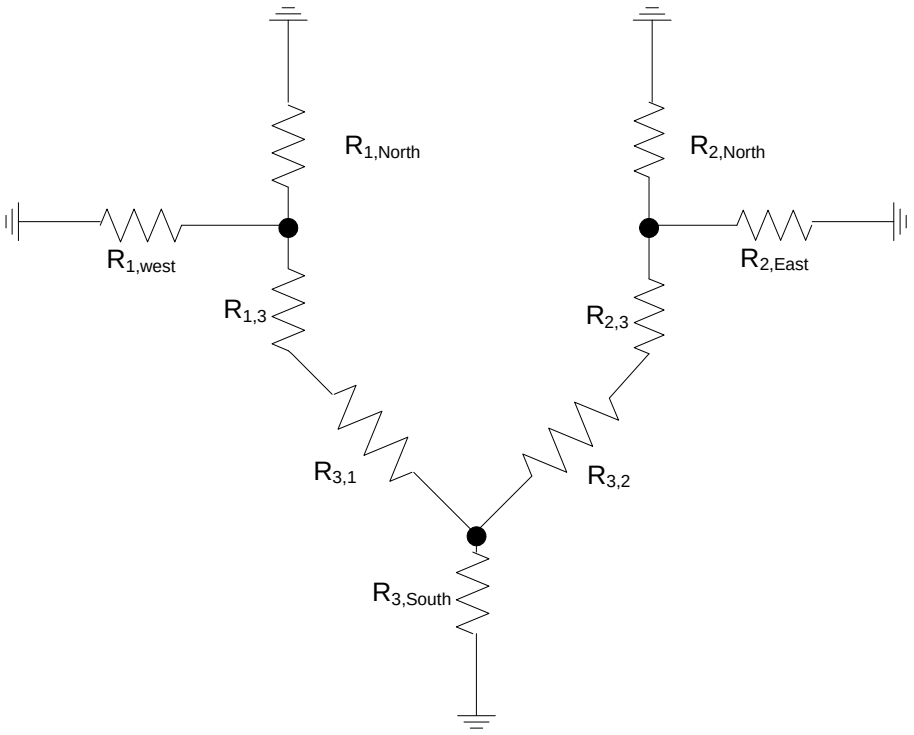


Figure 3.5. Thermal resistance network when cores 1 and 2 are concurrently tested

Furthermore, the time dependence of temperature and the power consumption must also be considered. As a rule, we want to test hot cores with large power densities as short as possible and minimize their effects on other cores (avoid concurrency and immediate precedence with cores in immediate physical periphery of the hot spot core).

Due to the localized nature of hot spots as well as the effects of layout and varying thermal resistance configurations, the core with the highest thermal cost does not always mean that it is hotter than cores with lower thermal costs. Thus, we define the following thermal cost for each core C_i with respect to its wrapper configuration w_{ij} and time t as shown below:

$$Cost_i(w_{ij}, t) = p_{ij} \times (R_{THi}(t) + TAT_{ij}) \tag{3.3}$$

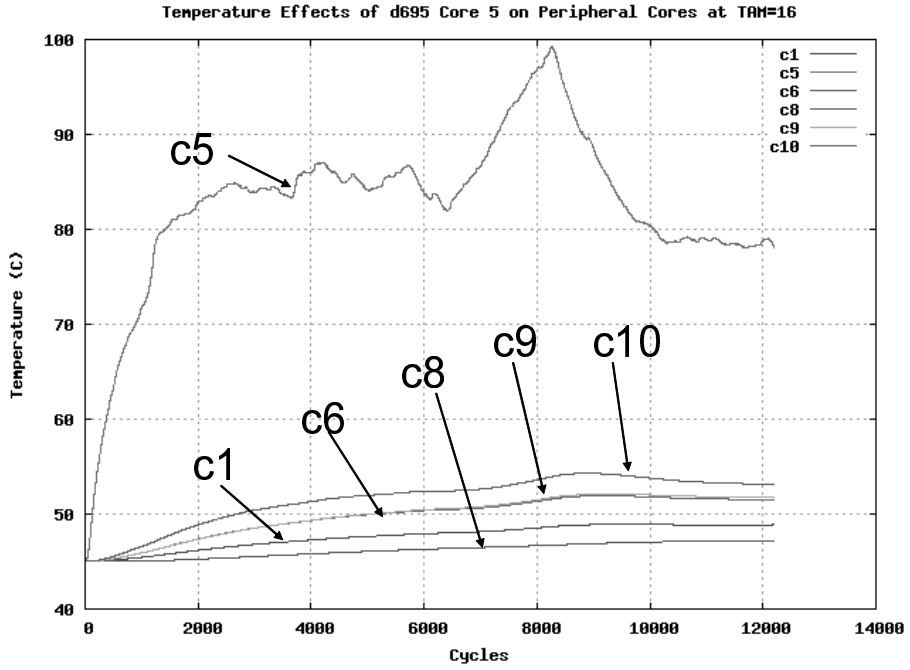
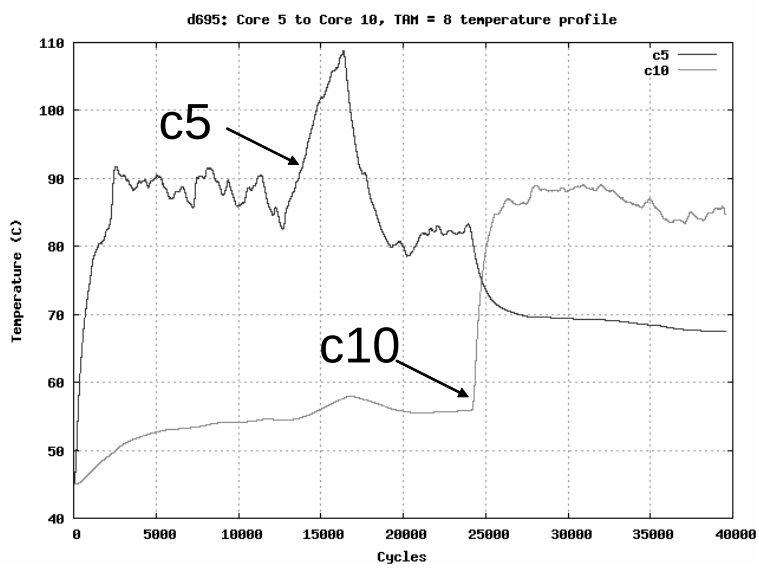
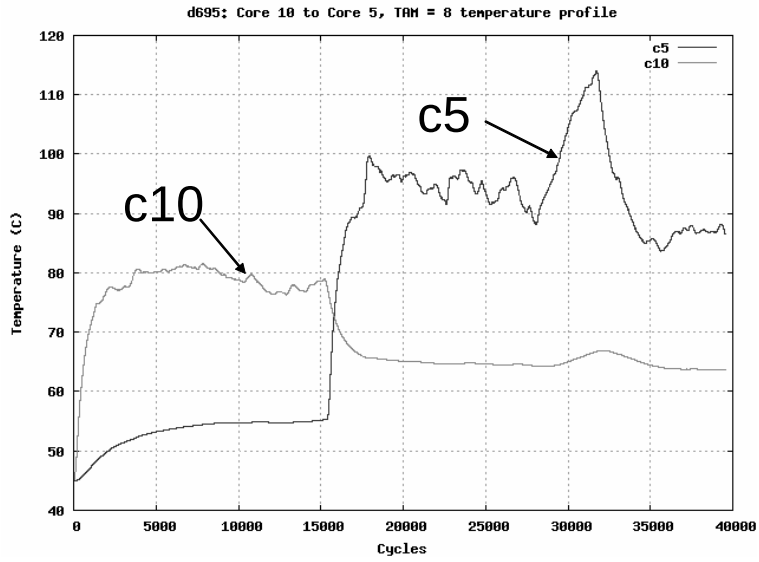


Figure 3.6. Thermal effect of d695 core 5 on peripheral cores

TAT_{ij} is the test application time and p_{ij} is the average power computed from power profile P_{ij} for wrapper configuration w_{ij}. The lateral resistance R_{THi} is expressed as a function of time because it changes according to when core C_i is scheduled and what cores are tested before as well concurrently with it. In our experiments, the average power dissipation was found to give a closer thermal profile curve to the actual thermal profile derived from cycle-accurate values compared to peak power values. Thus, instead of considering cycle accurate power, we chose to use average power values which vary with respect to w_{ij} to greatly simplify cost calculations. The main idea is to pick out hot spot cores, determine an upper limit to their thermal cost, $cost\_max_i$, and gradually decrease this limit until the thermal constraint is satisfied. Furthermore, a thermal cost minimum is computed which represents the worst case configuration of a core to be packed. It inevitably leads to the core being tested alone regardless of time frame, and not preceded by any immediate peripheral cores as given by the



(a)



(b)

Figure 3.7. Effects of test order on peak temperature, (a) core 5 before core 10, (b) core 10 before core 5

equation below:

$$cost\_min_i = \min_{1 \leq j \leq W_{ext}} (Cost_i(w_{ij}, NULL)) \tag{3.4}$$

where $Cost_i(w_{ij}, NULL)$ denotes the cost of unscheduled core C_i with wrapper configuration w_{ij} and no thermal resistance is removed in equation 3.3, denoted by $NULL$ time.

4. Test Scheduling Algorithm

The pseudo-code for our proposed algorithm is shown in Figure 3.8.

4.1 Initialization: Optimal Wrapper Configuration Creation

The initialization steps (lines 1-5 of Figure 3.8) first makes sure that a configuration for each core can be found which satisfies the thermal constraint $Temp_{max}$. Initially, the highest cost *cost_max* is set to infinity, and the minimum cost *cost_min* is computed for each core (line 4). It then uses a selection process introduced in [4] where Pareto-optimal points of the TAM vs. Test time graph are chosen as optimal wrapper configurations (w_{iopt}) in line 5. When choosing optimal wrapper configurations, the thermal cost must always satisfy both cost constraints.

4.2 Priority 1: Packing Rectangles with Optimal Wrapper Configuration

Before packing, the algorithm takes note of the current time in the schedule, denoted by the variable *current_t*. In line 8, we try to pack as many cores using optimal TAM widths while $available\_TAM \neq 0$. Each core C_i is examined in order of decreasing thermal cost when using their optimal wrapper configurations, denoted by $Cost_i(w_{iopt}, NULL)$, since potential hot spot cores should be sched-

	Function *Schedule*(**S**, W_{ext}, $Temp_{max}$)
1	**Do** thermal simulation for each w_{ij} configuration of core $C_i \in$ **S**
2	**If** no configuration that satisfies $Temp_{max}$, terminate scheduling;
3	Set $available\_TAM = W_{ext}$, $current\_t = 0$, $maxT = \infty$;
4	**For** each $C_i \in$ **S,** compute $cost\_min_i$, set $cost\_max_i = \infty$,
5	Find w_{iopt} (from[4]) such that $Cost_i(w_{iopt}, NULL) \leqq cost\_max_i$, then **end For**
6	**While S** $\neq \emptyset$
7	**If** $available\_TAM > 0$
8	**(Priority 1)** **If** there exist a core $C_i \in$ **S** such that $TAM_{iopt} \leqq available\_TAM$ **AND** $Cost_i(w_{iopt}, NULL)$ is maximum **AND** $Cost_j \leqq cost\_max_j$ for all scheduled cores C_j when C_i is scheduled at $current\_t$ with TAM_{iopt} **Then, Assign**(C_i, TAM_{iopt}) and go to line 6;
9	**(Priority 2)** **Else If** there exist a core $C_i \in$ **S** such that $TAM_{iopt} \leqq (available\_TAM + \alpha)$ **AND** TAM_{iopt} is minimum **AND** $Cost_j \leqq cost\_max_j$ for all scheduled cores C_j when C_i is scheduled at $current\_t$ with $available\_TAM$ **Then, Assign**(C_i, $available\_TAM$) and go to line 6;
10	**(Priority 3)** **Else If** there exist a scheduled C_i with assigned wrapper w_{fi} such that $tstart_i = current\_t$ **AND** has maximum decrease in test application time if $TAM_{fi} = TAM_{fi} + available\_TAM$ **AND** $Cost_j \leqq cost\_max_j$ for all scheduled cores C_j when C_i is scheduled at $current\_t$ with $TAM_{fi} + available\_TAM$ **Then, Assign**(C_i, $TAM_{fi} + available\_TAM$) and go to line 6;
11	**Else**, update $current\_t$ to the earliest test end time among currently scheduled cores, reset $available\_TAM$, return to line 6;
12	**End While**
13	**(Updating and cost adjustment)** **Do** thermal simulation of finished schedule and compute $maxT$ **If** $maxT \leqq Temp_{max}$, **Then** terminate scheduling
14	**Else**, Find the hottest core C_{hot} **If** $cost\_max_{hot} = \infty$ **Then** compute $cost\_max_{hot}$
15	**If** $(cost\_max_{hot} * adjust\_factor) \geqq cost\_min_{hot}$, **Then** $cost\_max_{hot} = (cost\_max_{hot} * adjust\_factor)$ and determine a new w_{hotopt} as done in line 5 and go to line 6;
16	**Else If** next hottest core exists, let it be C_{hot}, go to line 15;
17	**Else** terminate scheduling (no adjustable cores exists);

Figure 3.8. Pseudo code of proposed test scheduling algorithm

	Function *Assign*(C_i, TAM_{ij})
1	$TAM_{fi} = TAM_{ij}$;
2	$t_{starti} = current\_t$;
3	$t_{endi} = current\_t + TAT_{ij}$;
4	Update thermal cost $Cost_k$ for all scheduled cores C_k
5	$\mathbf{S} = \mathbf{S} - \{C_i\}$;
6	$available\_TAM = available\_TAM - TAM_{ij}$;

Figure 3.9. Pseudo code for the core Assign() function

uled as early and as quickly as possible to minimize their effects on subsequent cores.

Here and in all subsequent steps, the thermal costs for all active cores are computed and checked with their upper and lower limits before packing since they change whenever a new core is scheduled. The *Assign()* function in Figure 3.9 invoked after every priority step updates the parameters of the core to be scheduled (lines 1-3), recomputes the thermal costs of all the scheduled cores (line 4), updates the remaining core list (line 5) and available TAM (line 6). As the algorithm iterates further, hot spot cores are gradually separated from each other during scheduling due to the imposition of cost limits.

4.3 Priority 2: Insertion of Rectangles into Idle Space

If no rectangle can be packed in their optimal configuration, the algorithm looks for a core C_i whose optimal tam-width TAM_{iopt} is less than or equal to $available\_TAM+\alpha$ where $(1 \leq \alpha \leq 4)$ in line 9. In Figure 3.10, the wrapper configuration of core 4 was changed to a non-optimal configuration ($c4_{old}$ to $c4_{new}$) and inserted into the idle space of the schedule.

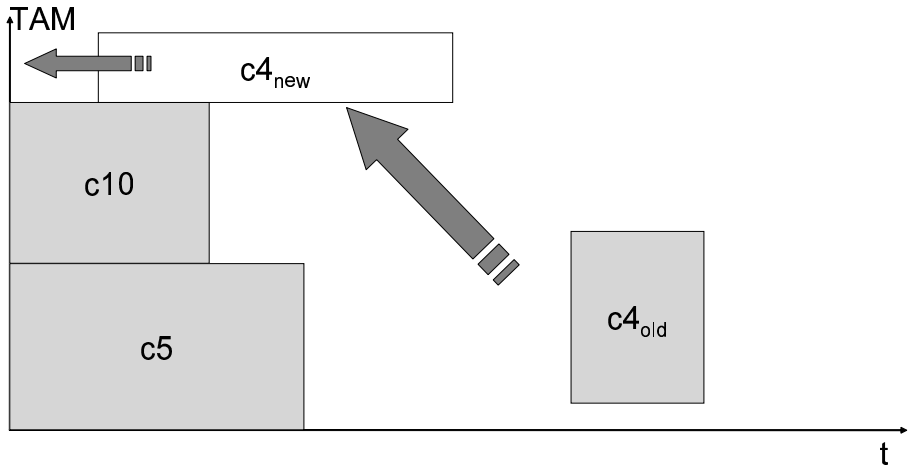


Figure 3.10. Inserting core 4 into idle TAM space by reducing assigned TAM width

4.4 Priority 3: Filling Idle Space by Increasing TAM width

The algorithm checks among the currently scheduled cores whose start times *tstart* equal *current_t* and determines which core would have the largest gain in test time if given the unused TAM lines and packs this core in line 10. In Figure 3.11, the alloted TAM width to the scheduled core c6 is changed to minimize wasted TAM wires. In line 11, *current_t* is updated when *available_TAM* becomes zero or when no cores can be scheduled in lines 8-10.

4.5 Updating and Cost Adjustment

When all cores have been scheduled, thermal simulation using HotSpot tool is performed using cycle-accurate power profiles in line 13. The peak chip-wide temperature $maxT$ is then compared to the thermal constraint. If it is satisfied, then the program ends. If not, then cost adjustment is performed on the hottest core C_{hot} in lines 14-15 and $cost\_max_{hot}$ is updated. Line 16 looks for the next hottest core to adjust when the current hot spot core's cost can no longer be adjusted. The program ends when the thermal constraint is satisfied or no more

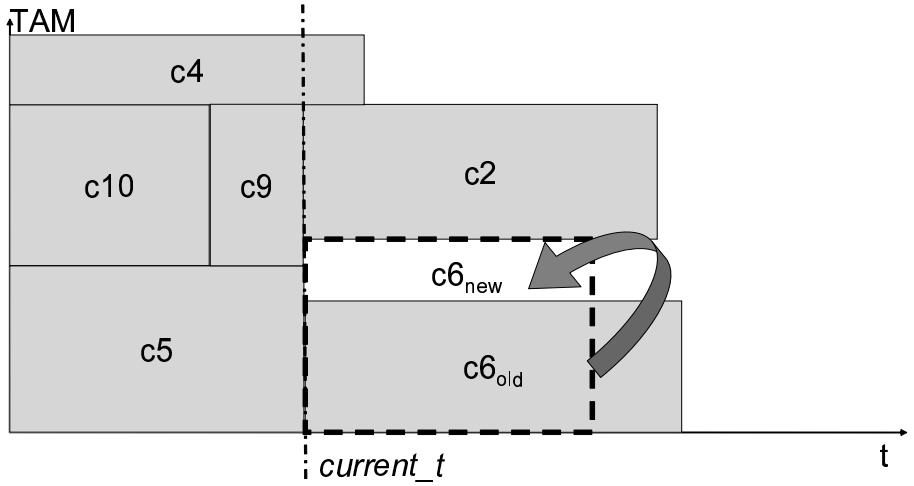


Figure 3.11. Allotting more TAM wires to core 6

cores can be adjusted. The adjustment factor, $adjust\_factor$, can be any value from 0-1. For this work, a constant factor of 0.90 is used.

5. Experimental Result

The experiments were done using three SoCs from the ITC'02 SoC Benchmark suite [18], d695, p22810, and p93791. For thermal simulation, cycle-accurate power profiles provided by the authors of [17] were used. Note that the actual power profiles were originally expressed as number of transitions per clock cycle. We converted the values into Watts by simply dividing them by 20, 200, and 500 for d695, p22810, and p93791, respectively, to reflect power dissipation during test. The test data for d695, upon thermal simulation, reveals that the total test time under TAM configurations used for this experiment (16, 24, 32, 64) are too short to show any significant heating of the chip. Therefore, when necessary, we have increased the length of the sampling interval during thermal simulation to allow the effects of heat to show. This is reasonable if we consider that tests for delay faults are normally 2-4 times larger than stuck-at-fault test sets. Since the test application time per core is normally much larger in magnitude compared

to lateral resistance, we scaled the test time values for each SoC such that their magnitudes are within acceptable range of each other (in this work, both total lateral resistance and test time was adjusted to not exceed 100) when computing for the thermal costs. Experiments were done using an HP ProLiant Workstation with 4 Opteron CPU's operating at 2.4GHz with 32GB of memory.

Since the original SoC benchmarks did not include layout information, we handcrafted the layout of each SoC. The scheduling and thermal simulation results for d695, p22810 and p93791 are shown in Tables 3.3 to 3.7. Before applying any thermal constraints, we used our scheduling algorithm to create a base schedule without any constraints. From the non-constrained schedule, we determine its maximum temperature, $maxT$, and use it as the thermal constraint, $Temp_{max}$. We gradually decreased the constraint by 5 degree steps, each time recording the actual maximum temperature ($maxT$), the test application time (TAT), and peak power value (P_{max}) given as number of switches. We also computed the gains in temperature (dT) with respect to the base temperature as well as the differences in TAT ($dTAT$).

In Table 3.3 for d695, a maximum temperature gain of 26.64% was achieved with a modest 24.75% increase in TAT (TAM = 32, $Temp_{max} = 80.16^oC$). For as little as 5.30% increase in TAT, we can get a relatively large gain of 20.86% in temperature reduction (TAM = 24, $Temp_{max} = 107.42^oC$). The limitations of global peak-power based approaches becomes apparent when we consider the results for TAM = 32 in Table 3.3. For most of the temperature variations, the peak power value remained constant at 1598. When such a power constraint is applied, the temperatures of the generated schedule can vary within the range of $89.58^oC - 77.15^oC$ and our algorithm makes sure that the thermal constraint is indeed satisfied.

For p22810 in Tables 3.4 and 3.5, a maximum temperature reduction of 33.82% can be had for a 20.38% increase in TAT (TAM = 24, $Temp_{max} = 111.37^oC$). At TAM = 32, the algorithm was able to decrease the temperature from 155.5^oC to a manageable 109.36^oC with just a 9.33% sacrifice in TAT. Similar results were obtained for p93791 in Tables 3.6 and 3.7, where there is a maximum temperature reduction of 35.61% with a 35.64% increase in TAT at TAM = 64. Note that at TAM=24, there was no further gain in temperature from temperature at

Table 3.3. Experimental results for d695

TAM = 16					
$Temp_{max}$	$maxT$	TAT	P_{max}	dT	$dTAT$
(^oC)	(^oC)	$(cycles)$	$(switches)$	(%)	(%)
∞	101.54	43504	1598	N/A	N/A
96.54	92.79	46873	1598	8.62	-7.74
91.54	N/A	N/A	N/A	N/A	N/A
TAM = 24					
$Temp_{max}$	$maxT$	TAT	P_{max}	dT	$dTAT$
(^oC)	(^oC)	$(cycles)$	$(switches)$	(%)	(%)
∞	122.42	30879	1713	N/A	N/A
117.42	109.53	31490	1624	10.53	-1.98
112.42	109.53	31490	1624	10.53	-1.98
107.42	96.88	32516	1598	20.86	-5.30
:	:	:	:	:	:
92.42	91.49	34250	1630	25.27	-10.92
87.42	N/A	N/A	N/A	N/A	N/A
TAM = 32					
$Temp_{max}$	$maxT$	TAT	P_{max}	dT	$dTAT$
(^oC)	(^oC)	$(cycles)$	$(switches)$	(%)	(%)
∞	105.16	22837	1650	N/A	N/A
100.16	89.58	24817	1598	14.82	-8.67
:	:	:	:	:	:
85.16	81.41	28489	1598	22.58	-24.75
80.16	77.15	28489	1598	26.64	-24.75
75.16	N/A	N/A	N/A	N/A	N/A
TAM = 64					
$Temp_{max}$	$maxT$	TAT	P_{max}	dT	$dTAT$
(^oC)	(^oC)	$(cycles)$	$(switches)$	(%)	(%)
∞	92.76	12696	1689	N/A	N/A
87.76	84.71	15343	1620	8.68	-20.85
82.76	N/A	N/A	N/A	N/A	N/A

Table 3.4. Experimental results for p22810 (TAM width = 16, 24bits)

TAM width = 16					
$Temp_{max}$	$maxT$	TAT	P_{max}	dT	$dTAT$
(^oC)	(^oC)	($cycles$)	($switches$)	(%)	(%)
∞	170.94	467362	8488	N/A	N/A
165.94	152.47	472762	7226	10.80	-1.16
:	:	:	:	:	:
150.94	133.02	511441	6006	22.18	-9.43
:	:	:	:	:	:
130.94	N/A	N/A	N/A	N/A	N/A
TAM width = 24					
$Temp_{max}$	$maxT$	TAT	P_{max}	dT	$dTAT$
(^oC)	(^oC)	($cycles$)	($switches$)	(%)	(%)
∞	166.37	324723	8104	N/A	N/A
161.37	154.07	338267	8054	7.39	-4.17
156.37	154.07	338267	8054	7.39	-4.17
151.37	148.98	345661	8048	10.45	-6.45
146.37	145.06	357802	7258	12.81	-10.19
141.37	135.45	359907	6986	18.59	-10.84
136.37	135.45	359907	6986	18.59	-10.84
131.37	113.89	396397	6166	31.54	-22.07
:	:	:	:	:	:
111.37	110.1	390905	6006	33.82	-20.38
106.37	N/A	N/A	N/A	N/A	N/A

$Temp_{max} = \infty$ and is mainly a result of the lack of flexibility due to the limited usable TAM width. Overall, the algorithm works well for designs with many cores and exploits the availabilty of wider TAMs.

Table 3.5. Experimental results for p22810 (TAM width = 32, 64bits)

TAM width = 32					
$Temp_{max}$	$maxT$	TAT	P_{max}	dT	$dTAT$
(^oC)	(^oC)	$(cycles)$	$(switches)$	(%)	(%)
∞	155.5	241403	9222	N/A	N/A
150.5	149.25	254660	7898	4.02	-5.49
145.5	109.36	263916	6184	29.67	-9.33
:	:	:	:	:	:
105.5	N/A	N/A	N/A	N/A	N/A
TAM width = 64					
$Temp_{max}$	$maxT$	TAT	P_{max}	dT	$dTAT$
(^oC)	(^oC)	$(cycles)$	$(switches)$	(%)	(%)
∞	138.81	149604	9936	N/A	N/A
133.81	129	145417	9974	7.07	2.80
128.81	113.79	153146	8542	18.02	-2.37
:	:	:	:	:	:
108.81	107.25	185614	6010	22.74	-24.07
103.81	N/A	N/A	N/A	N/A	N/A

6. Concluding Remarks

In this chapter, we have presented a TAM/Wrapper co-optimization framework for system-on-chips that ensures thermal safety while still optimizing the test schedule. The proposed method allows us to further explore, beyond the limits of peak-power based test scheduling, possible variations of a schedule which can lead to further reductions in temperature while limiting increases in test application time. Using cycle-accurate power profiles per wrapper configuration and considering both the spatial and temporal dimensions of heat transfer, overall, allows us to more closely approximate real world thermal phenomena.

Table 3.6. Experimental results for p93791 (TAM width = 16, 24bits)

TAM width = 16					
$Temp_{max}$	$maxT$	TAT	P_{max}	dT	$dTAT$
(^oC)	(^oC)	$(cycles)$	$(switches)$	(%)	(%)
∞	137.59	1842004	19690	N/A	N/A
132.59	119.74	1868602	12540	12.97	-1.44
:	:	:	:	:	:
117.59	115.03	1875576	12540	16.40	-1.82
112.59	N/A	N/A	N/A	N/A	N/A
TAM width = 24					
$Temp_{max}$	$maxT$	TAT	P_{max}	dT	$dTAT$
(^oC)	(^oC)	$(cycles)$	$(switches)$	(%)	(%)
∞	107.02	1261748	12540	N/A	N/A
102.02	N/A	N/A	N/A	N/A	N/A

Table 3.7. Experimental results for p93791 (TAM width = 32, 64 bits)

TAM width = 32					
$Temp_{max}$	$maxT$	TAT	P_{max}	dT	$dTAT$
(^oC)	(^oC)	($cycles$)	($switches$)	(%)	(%)
∞	139.82	946416	27890	N/A	N/A
134.82	126.9	969552	20675	9.24	-2.44
129.82	126.9	969552	20675	9.24	-2.44
124.82	115.37	1030210	16350	17.49	-8.85
119.82	115.37	1030210	16350	17.49	-8.85
114.82	107.93	1141742	12930	22.81	-20.64
109.82	107.93	1141742	12930	22.81	-20.64
104.82	103.78	1153424	12930	25.78	-21.87
99.82	96.96	1207921	12545	30.65	-27.63
94.82	94.63	1157587	12540	32.32	-22.31
89.82	N/A	N/A	N/A	N/A	N/A
TAM = 64					
$Temp_{max}$	$maxT$	TAT	P_{max}	dT	$dTAT$
(^oC)	(^oC)	($cycles$)	($switches$)	(%)	(%)
∞	142.25	483680	36930	N/A	N/A
137.25	113.38	527141	20935	20.30	-8.99
:	:	:	:	:	:
112.25	100.3	585385	19545	29.49	-21.03
:	:	:	:	:	:
97.25	92.53	631314	12885	34.95	-30.52
92.25	91.59	656079	12885	35.61	-35.64
87.25	N/A	N/A	N/A	N/A	N/A

Chapter 4

Thermal-Aware Test Infrastructure Design for SoCs with Fixed TAM

In this chapter, we present a technique for TAM optimization and test scheduling for core-based SoCs under thermal constraints that introduces new techniques to further improve the methodology in the previous chapter. Here, we assume a fixed-width TAM architecture, as in [2], and we consider test-set partitioning [21] and bandwidth matching [11] to derive more effective solutions. The main contributions of this chapter are as follows:

1. We study the impact of test-set partitioning and bandwidth matching on thermal-aware TAM optimization and test scheduling. Cycle-accurate power profiles are used for each wrapper configuration of an embedded core.

2. A computationally tractable thermal-cost model is used as the basis for an optimization algorithm for SoC TAM design and test scheduling. We minimize the SoC test time under thermal constraints.

3. Detailed simulation results are presented for two ITC'02 Benchmark SoCs.

The results show that (i) the test application time obtained using the proposed method is in most cases less than that using the method in the previous

chapter; (ii) the proposed method provides solutions even under tight temperature constraints, including situations where the previous method fails to find a solution.

The rest of this chapter is organized as follows. Additional motivation for this work, an overview of related prior work, and some key aspects of the proposed method are presented in the next section. Then, the proposed TAM optimization and test scheduling method is discussed in detail. We then present simulation results, including a detailed comparison with prior work. Finally, we end with some concluding remarks.

1. Limitations of Related Prior Works

As was discussed in the previous chapter, some of the previous works share certain limitations which make them unrealistic and impractical. To summarize, Rosinger et al. [19] first proposed the use of an RC-network based thermal model (based on [22]) for SoC test scheduling. This work draws upon the analogy between heat transfer and electrical current flow, which serves as the basis for test scheduling under thermal constraints. In [20], Liu et al. proposed scheduling algorithms that attempt to evenly spread heat over a chip using layout information and a progressive weighting function. In [21], He et al. proposed the use of test-set partitioning and test interleaving to allow hot cores to cool (while test resources are used to exercise other cores) and thereby avoid overheating. A drawback of all the above methods is that they consider fixed average power values per core and steady-state temperatures. Such an assumption is too restrictive in practice due to the temporal and spatial variation of hot spots and chip temperatures [22]. Furthermore, it was assumed that temperature influences between cores are negligible, which was shown in the previous chapter to be too optimistic and unrealistic.

Thus, a thermal-aware TAM/wrapper co-optimization and test scheduling method for SoCs with a flexible-width TAM architecture was presented in the preceding chapter. This approach uses cycle-accurate power profiles for accurate thermal simulation. The computation time for test scheduling is reduced by the use of a computationally tractable thermal-cost model which considers

the thermal effects between cores, and a heuristic bin-packing algorithm for test scheduling. Simulation results showed that, while the proposed solution is useful in many situations, especially for wide SOC-level TAMs, it is relatively ineffective under tight thermal constraints and narrow TAM widths. Therefore, we introduce techniques to overcome some of the limitations of the previous methodology in the following sections.

2. Test-Schedule Reshaping, Test-Set Partitioning, Test-Interleaving, and Bandwidth Matching

In this section, we incorporate test-schedule reshaping, test-set partitioning, test interleaving, and bandwidth matching techniques with respect to cycle-accurate power and temperature data. As shown in Figure 4.1, we assume that a Test Bus architecture, as in [2], is used for the target SoC. This architecture assumes that the TAM is partitioned into several fixed width test buses and each core is assigned to one of these partitions, as illustrated in Figure 4.1 for the d695 benchmark SoC.

To show the effects of test-schedule reshaping and the importance of considering temperature effects between cores, consider the example floor plan for the d695 SoC with the ten cores laid out as shown in Figure 3.1. Given a TAM architecture and core assignment shown in Figure 4.1, the test schedule in Figure 4.2(a) yields a maximum temperature of 110^oC using the HotSpot temperature simulation tool presented in [22]. During test re-shaping, by reordering the test of cores c2, c6, and c8 on TAM2, and c7, c4, c10, and c9 on TAM3, as shown in Figure 4.2(b), we are able to decrease the temperature to 100^oC. This is because the new schedule avoids the concurrent testing of c5 with c6 and c10, which are placed next to each other and are the hottest, 2nd-hottest, and 3rd-hottest cores, respectively. Furthermore, partitioning c5 into c5a and c5b and interleaving them with c3 in Figure 4.2(c) leads to an additional 5^oC drop in temperature. Note that in [21], temperature simulations were done for each test per core to determine the partitioning and cooling periods prior to actual scheduling. Sim-

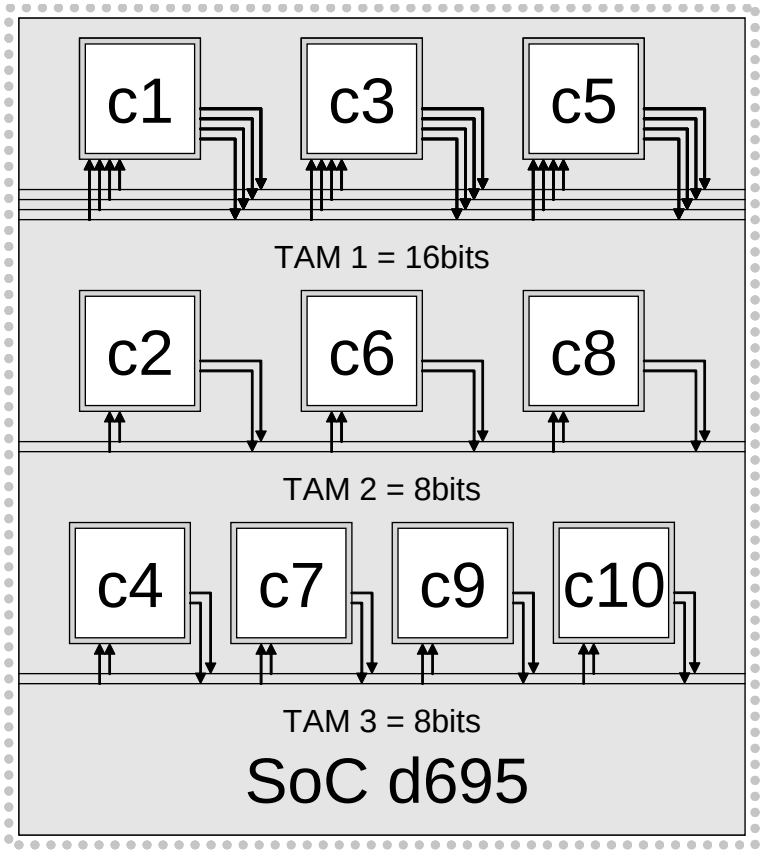
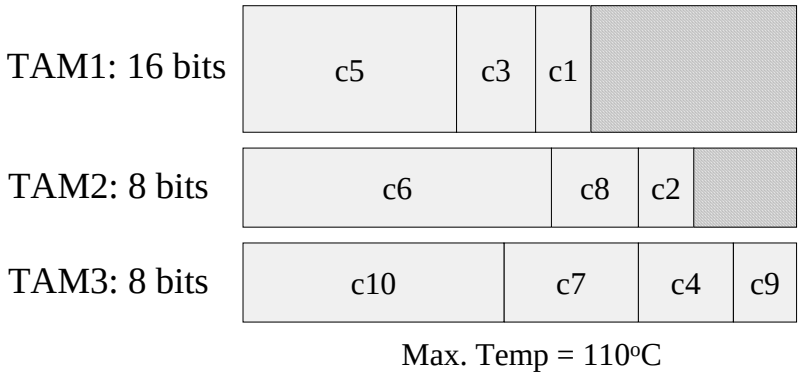


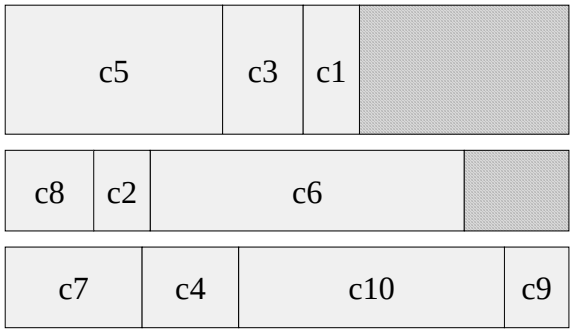
Figure 4.1. Example Test Bus architecture for d695 SoC

ulating the interleaved test of core 5 and 10 (Figure 4.3(a)), the thermal profile for core 5 (Figure 4.3(b)) shows that ignoring inter-core effects [21] and/or using fixed power profiles is too optimistic and only cycle-accurate thermal simulation will yield realistic results. In our method, partitioning and interleaving are done during scheduling, which ensures more realistic thermal profiles.

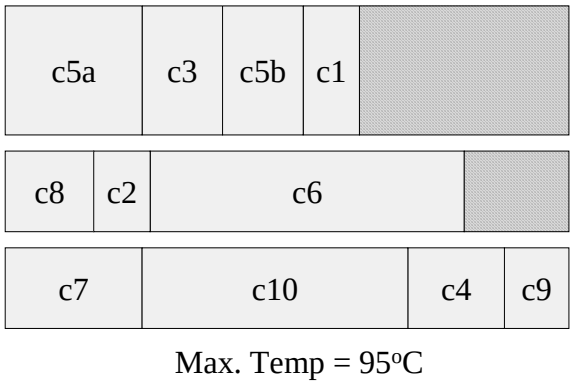
Under very tight temperature constraints, we propose using bandwidth-matching circuitry to significantly reduce test temperature. Frequency throttling has been combined with bandwidth matching circuitry and virtual TAM techniques in [11] to reduce dynamic power while minimizing the increase in test application time. Given an ATE frequency f_{ATE} with n TAM wires and a target virtual TAM frequency $freq(bi)$, by inserting a pair of *demultiplexing* (DeMUX) and *multiplexing*



(a)

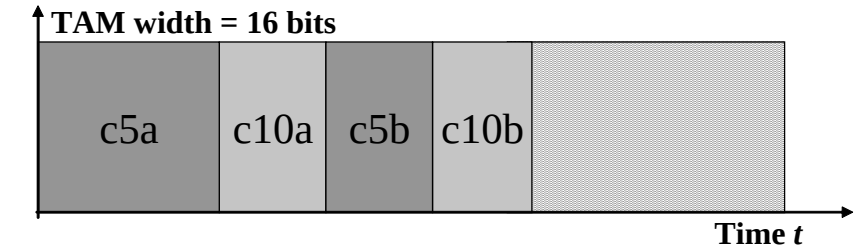


(b)

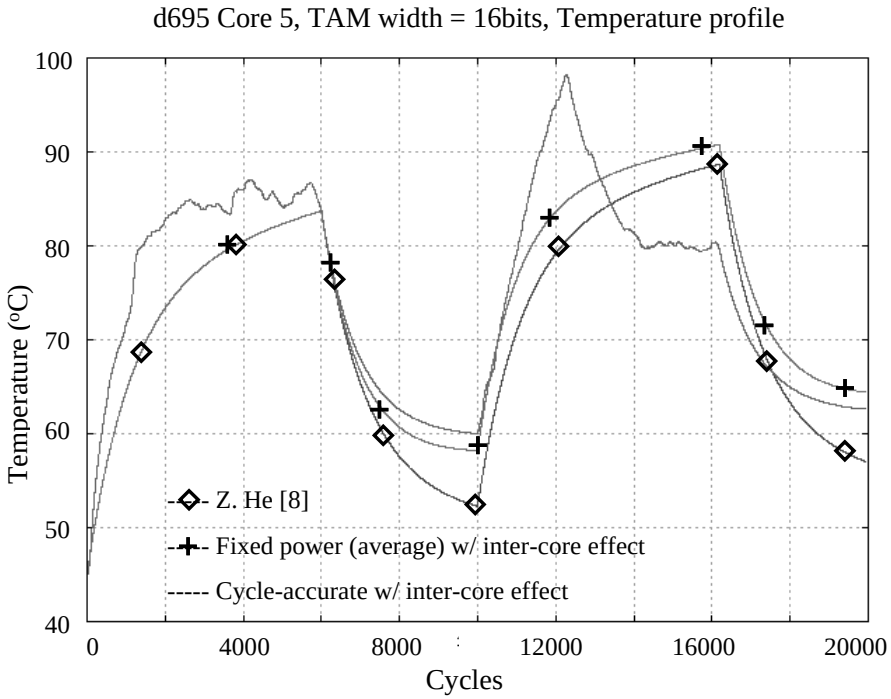


(c)

Figure 4.2. (a) Example test schedule, (b) after reshaping, (c) after test partitioning and interleaving



(a)



(b)

Figure 4.3. (a) Interleaved schedule for core 5 and 10, (b) thermal profile of core 5

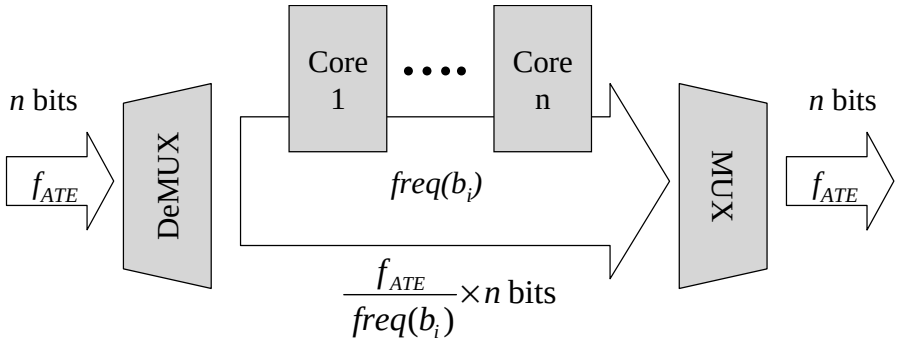


Figure 4.4. Bandwidth matching technique for core-based SoCs

(MUX) circuitry between the ATE and the internal TAM bi and increasing the number of virtual TAM wires to $f_{ATE}/freq(bi) * n$ bits, we can reduce the virtual TAM frequency (and therefore, power consumption) while minimizing the increase in test time, as shown in Figure 4.4. To simplify the clock generation circuitry, we assume that only repeated halving of the virtual test bus frequency (thereby doubling the virtual-bus wire count) is allowed. Since increasing the virtual TAM allotted to a core does not always results in a test time reduction, repeatedly halving the frequency has a best case scenario of 50% power reduction without sacrificing test time. The overall power reduction can lead to a significant drop in temperature during test.

3. Problem Formulation

In this section, we formally present the TAM design and test scheduling problem P_{SKED}.

Problem P_{SKED}: For an SoC S, given:

W_{ext}: external TAM width alloted to the SoC

N_C: a set of cores belonging to S

$Temp_{max}$: maximum allowed temperature during test

For each core $c_i (1 \leq i \leq |N_C|)$ of SoC S

- $Wset_i$: number of usable wrapper configurations
- NP_{maxi}: maximum number of test partitions allowed
- For each wrapper configuration $w_{ij}(1 \leq j \leq Wset_i)$

  - TAM_{ij}: alloted TAM width
  - P_{ij}: power profile
  - TAT_{ij}: test application time

Our goal is to determine the following output:
TAM_{cfg}: TAM and core configuration of the SoC, which includes
- B: a set of TAMs
- For each TAM $b_i \in B$ of S,

  - W_i: alloted TAM width
  - C_i: a set of cores belonging to b_i
  - For each core $c_j \in C_i$

    - NP_j: set of partitions of the test for c_j
    - For each test partition $p_k \in NP_j$

      * $tstart_k$: test start time
      * $tend_k$: test end time

such that the temperature does not exceed $Temp_{max}$ while the test application time is minimized.

4. TAM Design and Test Scheduling Algorithm

Our basic strategy for TAM design and test scheduling involves four main steps.

1. During the initialization step, the algorithm determines an initial TAM design and optimal test schedule for the SoC under no thermal constraint and determines the hottest possible core.

2. During test reshaping, the schedule is rearranged to minimize the temperature of the hot spot core. The new schedule undergoes another thermal simulation.

3. If the temperature constraint is not satisfied in Step 2, test partitioning is performed on the hottest core. Steps 2 and 3 are repeated until the test for the hottest core can no longer be partitioned. Note that interleaving is done during the reshaping stage.

4. If the previous step fails, the partitions of test for the hottest core are recombined and bandwidth matching circuitry is inserted on the TAM where the hot spot core belongs. Steps 2 to 4 are repeated until the constraint is satisfied or the virtual TAM width limit is reached.

Note that cycle-accurate thermal simulation is performed to check the test temperature every time the schedule is reshaped. In reality, this accounts for almost all the processing time. Note also that exploration of all possible schedule arrangements, partitioning and virtual TAM configurations is virtually impossible. Thus, we propose a simplified thermal cost function which will give us an idea of the heating phenomena during test without resorting to thermal simulation. This also serves as the basis for the heuristic test scheduling algorithm to minimize the thermal simulation effort and overall computation time.

4.1 Thermal Cost Function

Since we are dealing with SoCs with a fixed TAM configuration (i.e. fixed partitioning and width) as well as fixed core distribution among the TAM partitions, the wrapper configuration and power profile for each core are already fixed during the scheduling step. The problem of minimizing the hot spot temperature, therefore, becomes a problem of limiting the thermal contributions of the peripheral cores on the hot spot core. We assume that the thermal contribution of core c_j on core c_i for a given schedule depends on the following three parameters:

1. the average power consumption of the core c_j

2. the thermal resistance between c_j and c_i

3. the relative test times between c_j and core c_i.

It was established in [19] that there exists a positive correlation between heat and heat dissipation paths represented by lateral thermal resistances, shown in Figure 3.3. Thermal resistance is directly proportional to the thickness of the material and inversely proportional to the cross-sectional area across which the heat is being transferred [22]. For this work, we express the thermal contribution of core c_j on core c_i for a test schedule as the thermal cost function below:

$$Tcont_j(c_i) = \frac{R_{ji}}{R_{TOT,j}} \times Pavg_j \times \frac{Trel_{ji}}{TAT_i} \tag{4.1}$$

where R_{ji} is the lateral thermal resistance from core c_j to c_i (R_{ii}=0), $R_{TOT,j}$ is the total lateral resistance from core c_j, and $Pavg_j$ is the average power dissipation of c_j. Moreover, the parameter $Trel_{ji}$ is defined as follows:

$$Trel_{ji} = \left\{ \begin{array}{rl} TAT_j - (Tstart_i - Tend_j) & \text{, if } \quad Tend_j < Tstart_i \\ TAT_j & \text{, if } \quad Tstart_i \leq Tend_j \leq Tend_i \\ TAT_j - (Tend_j - Tend_i) & \text{, if } \quad Tstart_j < Tend_i < Tend_j \end{array} \right\} \tag{4.2}$$

where TAT_i is the test application time of c_i, $Tstart_i$ is the test start time of c_i, and $Tend_i$ is the test end time of c_i. In Equation 4.1, we assume that the heat flowing from a core c_j to core c_i is proportional to the lateral resistance R_{ji} from the source to the destination core as well as the source's power dissipation, $Pavg$. Moreover, the more heat-dissipation paths a source core has, the less heat flowing through each lateral resistance. Therefore, we divide the cost by $R_{TOT,j}$. The parameter $Trel_{ji}$ expresses the weight we give on how the relative test times between the two cores c_i and c_j affect their thermal contributions to each other and models the fact that the greater the time they have to affect each other, the greater the heat contribution of the cores to each other, but it is set to zero when the value becomes negative. From the previous chapter, it has been shown that using fixed average power values, instead of peak power values, for thermal simulation gives a closer approximate of the thermal profile curve derived from cycle-accurate values. Thus, instead of considering cycle accurate power, we chose to use average power values to greatly simplify cost calculations. The total

thermal contribution of other cores to c_i for a certain schedule is computed as in Equation 4.3, where N is the total number of cores of the SoC. The main idea is to use this information to reconfigure the test schedule such that the overall thermal contribution to the hot spot core is minimized to the point that the constraint is satisfied.

$$Tcont_{TOT}(c_i) = \sum_{j=1}^{N} Tcont_j(c_i) \tag{4.3}$$

4.2 Heuristic TAM Design and Test Scheduling Algorithm

Each step of the proposed TAM design and test scheduling algorithm is explained in detail in this subsections.

Step 1: Initial TAM Design, Bin Sorting, and Initial Scheduling

Among TAM design algorithms, *TR-Architect* [2] has been shown to be one of the most efficient algorithms for determining TAM partition and core assignments. For our methodology, we utilize this algorithm to determine an initial TAM design and core assignment that minimizes the test application time without any power or thermal constraints during the initialization step. Then, the algorithm makes sure that each core wrapper configuration satisfies the thermal constraint $Temp_{max}$. Each core c_i has a minimum thermal cost $cost\_max_i$ (initially set to 0) and a temporary cost to determine potential hot spot cores, $cost\_tmp_i$, computed for each core using Equation 4.4, where $Area_i$ is the surface area of c_i. It is assumed in Equation 4.4 that the core with the highest power density and/or longest test time has the potential to be the hottest core during test. Each core is represented as a rectangle, where the height represents allotted TAM width and the length represents test application time. The rectangles are then sorted in descending order from the core with the highest *cost_tmp*.

$$cost\_tmp_i = \frac{Pavg_i}{Area_i} \times TAT_i \tag{4.4}$$

During initial scheduling, an empty bin, whose height and width represent total external TAM width and overall test time, respectively, is first divided into

$|B|$ sub-bins representing each TAM partition, $b_i(1 \leq i \leq |B|)$. The rectangles are packed into their respective pre-assigned sub-bins (e.g. TAM partitions) according to their *cost_tmp*. Thermal simulation is then performed, after all rectangles have been packed, on the finished schedule to determine the hottest core, c_{HOT}, and the time when the temperature is equal to $Temp_{max}$, t_{HOT}, using the HotSpot simulator developed in [22]. The algorithm ends if the hottest core does not exceed $Temp_{max}$. Otherwise, it proceeds to Step 2.

Step 2: Cost-constrained Schedule Reshaping

In this step, the algorithm rearranges the current test schedule to minimize the cost of the current hottest core, c_{HOT}, without increasing the costs of previous hot spot cores (called *reference cores*) belonging to the set C_{ref}, which is initially empty.

For each TAM partition designed in Step 1, the cores are sorted in descending order of $cost\_tmp_i$ to determine potential hot spot cores. It then looks for the core c_{target} with highest *cost_tmp* value but with the smallest thermal contribution to c_HOT. For each core c in C_{ref}, if the new cost of c, $Tcont_{TOT}(c)$ after packing c_{target} does not exceed its maximum cost constraint, we pack c_{target} into its assigned TAM partition. If no core can be found, revert to the schedule at the beginning of this step and go to Step 3. Otherwise, continue this step until all cores have been packed. In Figure 4.5, when packing the new target core, $c2$, the new costs of previous hot spot cores, $c3$, $c4$, and $c7$ must be checked and they must not exceed their *cost_max* values.

After packing all cores, thermal simulation is performed to determine the new hottest core c_{new}. The algorithm finishes if the temperature of c_{new} satisfies $Temp_{max}$. If the temperature of c_{new} does not satisfy the thermal constraint and c_{new} already belongs to $C_{ref} \cup c_{HOT}$, then we proceed to Step 3. Otherwise, add c_{HOT} to C_{ref}, and set $c_{HOT} = c_{new}$, and we repeat Step 2.

Step3: Test Partitioning and Interleaving

The algorithm takes note of the time t_{HOT} when the temperature of the hottest core c_{HOT} is equal to $Temp_{max}$. Then, the test of c_{HOT} is partitioned at t_{HOT} into two tests (creating two new virtual cores c_{HOT1}, c_{HOT2}) as long as the number

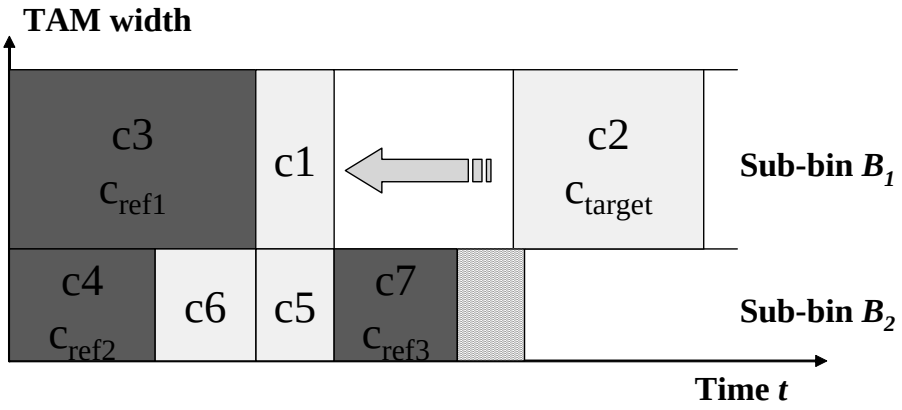


Figure 4.5. Core $c2$ is scheduled if reference cores $c3$, $c4$ and $c7$ satisfy their cost constraints

of test partitions for c_{HOT} does not exceed NP_{max}. The algorithm updates the core list and returns to Step 2, but this time with an added precedence constraint that the partition c_{HOT2} can only be scheduled after finishing the test of c_{HOT1}. Furthermore, the schedules of all other cores that were active on or before t_{HOT} remain unchanged so that the temperature profile up to this time is preserved. If the test of c_{HOT} can no longer be partitioned, the algorithm proceeds to Step 4. In Figure 4.6, core $c3$ is partitioned and $c1$ and $c2$ are inserted between the two partitions during scheduling during reshaping. Also, the schedule of $c4$ remains unchanged since it was active at time t_{HOT}.

Step 4: Bandwidth Matching Circuitry Insertion

In this step, bandwidth matching circuitry is added to the TAM partition where the target hot spot core found in Step 1, c_{HOT}, is assigned. Before doing so, all the cores are reset to their unpartitioned configuration, and all their cost values are reset to their initial values. For this step, the algorithm tries to reduce the power consumption of the target core by half by halving the TAM partition frequency. Increase in total test application time for the target TAM partition is minimized by doubling the assigned virtual TAM width. The algorithm then re-computes *cost_tmp* for all cores and repeats Steps 1 to 4.

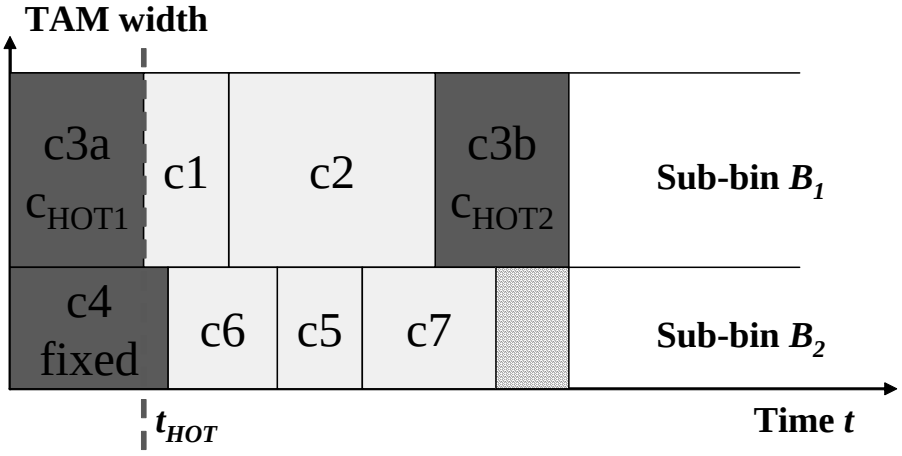


Figure 4.6. Core $c3$ is partitioned during step 3

5. Experimental Results

The experiments were carried out using d695 and p22810 SoCs from the ITC'02 SoC Benchmark suite [18]. For thermal simulation, cycle-accurate power profiles provided by the authors of [17] were used. Note that the actual power profiles were originally expressed as number of transitions per clock cycle. We converted the values into Watts by scaling them to reflect realistic power dissipation during test. Experiments were done using an HP ProLiant Workstation with 4 Opteron CPU's operating at 2.4GHz with 32GB of memory. All temperature values were obtained using the HotSpot temperature simulator from [22].

Since the original SoC benchmarks did not include layout information, we handcrafted the layout of the SoCs. Experiments were conducted for TAM widths 16, 24, 32 and 64 bits. Furthermore, each core can only be partitioned up to 3 times and the maximum virtual TAM width for each TAM partition is set to 64 bits.

The experimental results for d695 and p22810 are shown in Tables 4.1 to 4.4, respectively. We set the thermal constraint, $Temp_{max}$, at the initial value of the actual maximum temperature of the schedule, $maxT$, when the constraint is at infinity and decrease it by 5^oC and 10^oC intervals for d695 and p22810, respec-

tively, each time recording the test application time (TAT), and peak power value (P_{max}) given as number of switches. We also computed the gains in temperature (dT) with respect to the original base temperature as well as the differences in TAT ($dTAT$) compared to the unconstrained TAT. Grayed-out values indicate results achieved using a combination of reshaping, partitioning, and bandwidth matching while unmarked values were obtained using reshaping alone. The effectiveness of the reshaping and partitioning steps can be seen when temperature drops were achieved without any increase in TAT and/or drastic decrease in power dissipation, as can be seen from $Temp_{max}$=104.3^oC to 80.59^oC for TAM width of 64 bits for d965 in Table 4.2, and $Temp_{max}$=167.71^oC to 147.55^oC for TAM width of 16 bits for p22810 in Table 4.3. Note that as TAM width increases, more TAM partitions can be formed and fewer cores are placed in each partition, resulting in a higher probability of hot cores being tested concurrently and reducing the ability of the algorithm to separate their test instances via interleaving, which is indicated by overall higher minimum temperatures for larger TAM widths.

To further show the effectiveness of the proposed algorithm, the results obtained using the method in the previous chapter is compared with the results using the proposed algorithm for d695 under the same thermal constraints in Tables 4.5 and 4.6, where $diffTAT$ represents the difference in TAT. Before applying any thermal constraints, we used the scheduling algorithm in the previous chapter to create a base schedule under no constraints. The results show that for TAM widths of 24, 32 and 64 bits, the proposed algorithm yields shorter overall test application time under the same thermal constraints, with a maximum difference of 26% at a TAM width of 64 bits. Furthermore, this new method allows us to generate results at lower thermal constraints that exceed those in the previous chapter.

The minimum temperatures and the respective test times for each SoC achieved using the algorithm in the previous chapter and the one proposed here are shown in Table 4.7 for TAM widths of 16, 24, 32 and 64 bits. $diffT$ shows the difference in minimum temperatures while $diffTAT$ represents the difference in TAT. For d695, the new proposed algorithm enabled us to lower the test temperature much further compared to before, with a minimum temperature approx. 40% lower at

TAM width of 16 bits (92.79^oC compared to 55.8^oC), albeit with a 152% increase in TAT. While this increase might seem large, the algorithm at least offers the option to trade-off TAT for further decrease in temperature when needed. On average, the minimum temperatures for d695 were 22% lower using the proposed algorithm. The results are similar for p22810, were the biggest temperature difference was 41% at TAM width of 16 bits and the average temperature difference is 24%. Note that the algorithm proposed here is especially effective at very narrow TAM widths, as shown by generally lower minimum temperatures at narrower $TAMwidths$. This is the situation where we can benefit most from reshaping, partitioning, interleaving and frequency throttling, as whene we have fewer TAM partitions, more cores are assigned to each of them.

6. Concluding Remarks

In this chapter, we have presented a thermal-aware TAM design and test scheduling algorithm for system-on-chips with fixed-width TAMs that ensures thermal safety while minimizing the test application time. The proposed method allows us to further explore, beyond the limits of peak-power based test scheduling, possible variations of a schedule which can lead to further reductions in temperature using test reconfiguration, partitioning, interleaving and bandwidth matching techniques. Such techniques enabled us to overcome the limitations of a fixed-TAM configuration and achieve better results compared to method presented in the previous chapter. Using cycle-accurate power profiles per wrapper configuration and considering both the spatial and temporal dimensions of heat transfer, overall, allows us to more closely approximate real world thermal phenomena.

Table 4.1. Results using proposed algorithm for d695 (TAM width = 16, 24bits)

d695					
TAM width: 16 bits					
Temp $_{max}$	***maxT***	***TAT***	***P*** $_{max}$	***dT***	***dTAT***
(°C)	***(°C)***	***(cycles)***		***(%)***	***(%)***
∞	111.28	45363	1646	N/A	N/A
106.28	106.19	45363	1625	4.57	0.00
101.28	80.73	51485	1647	27.45	-13.50
:	:	:	:	:	:
76.28	76.25	51485	1624	31.48	-13.50
71.28	71.28	51485	1647	35.95	-13.50
66.28	64.16	63863	1633	42.34	-40.78
61.28	61.28	63863	810	44.93	-40.78
56.28	55.80	118375	820	49.86	-160.95
51.28	N/A	N/A	N/A	N/A	N/A
TAM width: 24 bits					
Temp $_{max}$	***maxT***	***TAT***	***P*** $_{max}$	***dT***	***dTAT***
(°C)	***(°C)***	***(cycles)***		***(%)***	***(%)***
∞	103.47	29122	1855	N/A	N/A
98.47	98.46	29122	1863	4.84	0.00
93.47	78.27	34189	1779	24.35	-17.40
:	:	:	:	:	:
73.47	73.46	34189	1771	29.00	-17.40
68.47	67.15	35871	1724	35.10	-23.17
63.47	63.08	35871	1667	39.04	-23.17
58.47	58.46	49995	888	43.50	-71.67
53.47	N/A	N/A	N/A	N/A	N/A

Table 4.2. Results using proposed algorithm for d695 (TAM width = 32, 64bits)

d695					
TAM width: 32 bits					
Temp $_{max}$	***maxT***	***TAT***	***P*** $_{max}$	***dT***	***dTAT***
(°C)	***(°C)***	***(cycles)***		***(%)***	***(%)***
∞	118.02	22543	2023	N/A	N/A
113.02	89.09	22543	1840	24.51	0.00
:	:	:	:	:	:
88.02	86.13	22619	1764	27.02	-0.34
83.02	69.91	23851	1130	40.76	-5.80
:	:	:	:	:	:
68.02	N/A	N/A	N/A	N/A	N/A
TAM width: 64 bits					
Temp $_{max}$	***maxT***	***TAT***	***P*** $_{max}$	***dT***	***dTAT***
(°C)	***(°C)***	***(cycles)***		***(%)***	***(%)***
∞	104.30	11358	1811	N/A	N/A
99.30	80.59	11358	1769	22.73	0.00
:	:	:	:	:	:
79.30	N/A	N/A	N/A	N/A	N/A

Table 4.3. Results using proposed algorithm for p22810 (TAM width = 16, 24bits)

p22810					
TAM width: 16 bits					
Temp $_{max}$	***maxT***	***TAT***	***P*** $_{max}$	***dT***	***dTAT***
(°C)	***(°C)***	***(cycles)***		***(%)***	***(%)***
∞	167.71	482480	7902	N/A	N/A
157.71	157.18	482480	7869	6.28	0.00
147.71	147.55	482480	7876	12.02	0.00
137.71	121.64	568077	4934	27.47	-17.74
:	:	:	:	:	:
117.71	112.34	568077	4864.5	33.02	-17.74
107.71	104.58	568077	4575.5	37.64	-17.74
97.71	97.71	783823	3105.25	41.74	-62.46
87.71	82.69	783823	2464.25	50.69	-62.46
77.71	77.71	783823	2453	53.66	-62.46
67.71	N/A	N/A	N/A	N/A	N/A
TAM width: 24 bits					
Temp $_{max}$	***maxT***	***TAT***	***P*** $_{max}$	***dT***	***dTAT***
(°C)	***(°C)***	***(cycles)***		***(%)***	***(%)***
∞	193.67	307039	8557	N/A	N/A
183.67	167.89	307039	9055	13.31	0.00
:	:	:	:	:	:
163.67	148.66	332611	6110.5	23.24	-8.33
:	:	:	:	:	:
143.67	143.67	332611	6110.5	25.82	-8.33
133.67	115.66	355157	5152.5	40.28	-15.67
:	:	:	:	:	:
113.67	107.347	355157	5407.5	44.57	-15.67
103.67	102.29	363043	3967.5	47.18	-18.24
93.67	N/A	N/A	N/A	N/A	N/A

Table 4.4. Results using proposed algorithm for p22810 (TAM width = 32, 64bits)

p22810					
TAM width: 32 bits					
Temp $_{max}$	***maxT***	***TAT***	***P*** $_{max}$	***dT***	***dTAT***
(°C)	***(°C)***	***(cycles)***		***(%)***	***(%)***
∞	166.17	230136	9744	N/A	N/A
156.17	151.46	293501	6863	8.85	-27.53
146.17	146.17	293501	6734	12.04	-27.53
136.17	136.17	293501	7271	18.05	-27.53
126.17	122	295015	6228.5	26.58	-28.19
116.17	115.78	295015	6802.5	30.32	-28.19
106.17	106.17	474807	5688	36.11	-106.32
96.17	94.64	474807	4846.75	43.05	-106.32
86.17	86.17	474807	4650.5	48.14	-106.32
76.17	76.08	474807	3343.75	54.22	-106.32
66.17	N/A	N/A	N/A	N/A	N/A
TAM width: 64 bits					
Temp $_{max}$	***maxT***	***TAT***	***P*** $_{max}$	***dT***	***dTAT***
(°C)	***(°C)***	***(cycles)***		***(%)***	***(%)***
∞	130.44	133404	10300	N/A	N/A
120.44	118.89	165909	9619.5	8.85	-24.37
110.44	105.11	165909	6652.5	19.42	-24.37
100.44	100.23	223385	5811	23.16	-67.45
90.44	N/A	N/A	N/A	N/A	N/A

Table 4.5. Comparison of results using method in previous chapter and proposed algorithm for d695 (TAM width = 16, 24bits)

d695					
TAM width: 16 bits					
	Using [10]		**Proposed algorithm**		
Temp $_{max}$	***maxT***	***TAT***	***maxT***	***TAT***	***diffTAT***
(°C)	***(°C)***	***(cycles)***	***(°C)***	***(cycles)***	***(%)***
∞	101.54	43504	111.28	45363	-4.27
96.54	92.79	46873	80.73	51485	-9.84
:	N/A	N/A	:	:	N/A
76.54	:	:	76.53	51485	N/A
71.54	:	:	71.54	51485	N/A
66.54	:	:	64.17	63863	N/A
61.54	:	:	61.54	63863	N/A
56.54	:	:	55.80	118375	N/A
51.54	:	:	N/A	N/A	N/A
TAM width: 24 bits					
Temp $_{max}$	***maxT***	***TAT***	***maxT***	***TAT***	***diffTAT***
(°C)	***(°C)***	***(cycles)***	***(°C)***	***(cycles)***	***(%)***
∞	122.42	30879	103.47	29122	5.69
117.42	109.53	31490	103.47	29122	7.52
112.42	109.53	31490	103.47	29122	7.52
107.42	96.88	32516	103.47	29122	10.44
102.42	96.88	32516	102.41	29122	10.44
97.42	96.88	32516	97.41	29122	10.44
92.42	91.49	34250	78.27	34189	0.18
:	N/A	N/A	:	:	N/A
77.42	:	:	77.40	34189	N/A
72.42	:	:	72.41	34189	N/A
67.42	:	:	67.15	35871	N/A
62.42	:	:	62.41	35871	N/A
57.42	:	:	N/A	N/A	N/A

Table 4.6. Comparison of results using method in previous chapter and proposed algorithm for d695 (TAM width = 32, 64bits)

d695					
TAM width: 32 bits					
Temp $_{max}$	***maxT***	***TAT***	***maxT***	***TAT***	***diffTAT***
(°C)	***(°C)***	***(cycles)***	***(°C)***	***(cycles)***	***(%)***
∞	105.16	22837	118.02	22543	1.29
100.16	89.58	24817	89.09	22543	9.16
:	:	:	:	:	:
85.16	81.41	28489	85.16	22619	20.60
80.16	77.15	28489	69.91	23851	16.28
:	N/A	N/A	:	:	N/A
65.16	:	:	N/A	N/A	N/A
TAM width: 64 bits					
Temp $_{max}$	***maxT***	***TAT***	***maxT***	***TAT***	***diffTAT***
(°C)	***(°C)***	***(cycles)***	***(°C)***	***(cycles)***	***(%)***
∞	92.76	12696	104.30	11358	10.54
87.76	84.71	15343	80.59	11358	25.97
82.76	N/A	N/A	80.59	11358	N/A
77.76	:	:	N/A	N/A	N/A

Table 4.7. Comparison of minimum temperature using method in previous chapter and proposed algorithm

	TAM width: 16 bits					
	[10]		**Proposed Method**			
SoC	***maxT (°C)***	***TAT (cycles)***	***maxT (°C)***	***TAT (cycles)***	***diffT (%)***	***diffTAT (%)***
d695	92.79	46873	55.80	118375	39.86	-152.54
p22810	133.02	511441	77.71	783823	41.58	-53.26
	TAM width: 24 bits					
	[10]		**Proposed Method**			
SoC	***maxT (°C)***	***TAT (cycles)***	***maxT (°C)***	***TAT (cycles)***	***diffT (%)***	***diffTAT (%)***
d695	91.49	34250	58.46	49995	36.10	-45.97
p22810	110.1	390905	102.29	363043	7.09	7.13
	TAM width: 32 bits					
	[10]		**Proposed Method**			
SoC	***maxT (°C)***	***TAT (cycles)***	***maxT (°C)***	***TAT (cycles)***	***diffT (%)***	***diffTAT (%)***
d695	77.15	28489	69.91	23851	9.38	16.28
p22810	109.36	263916	71.17	523139	34.92	-98.22
	TAM width: 64 bits					
	[10]		**Proposed Method**			
SoC	***maxT (°C)***	***TAT (cycles)***	***maxT (°C)***	***TAT (cycles)***	***diffT (%)***	***diffTAT (%)***
d695	84.71	15343	80.59	11358	4.86	25.97
p22810	107.25	185614	92.79	223385	13.48	-20.35

Chapter 5

Non-Robust Untestable RTL Path Identification in Circuits with Multi-Cycle Paths

As LSI manufacturing technology improves and the time-to-market for products becomes stricter, more and more circuit designs have multiple clock domains due to concerns such as design re-use, power reduction and temperature control. It is not uncommon for these designs to have multi-cycle paths which are untestable. The rapid identification of these untestable paths reduces test generation time as well as over-testing due to DFT. For current and future designs, this has already become impractical at the gate-level. This chapter presents a method to identify non-robust untestable multi-cycle paths at the register transfer level and the details in a case study of a benchmark circuit.

The contributions of this work are:

1. introduce the concept of multi-cycle false paths at RTL
2. derive sufficient conditions for multi-cycle RTL false paths for two transition propagation models which can be modified to apply to various circuit designs
3. give a case study of a benchmark circuit which proves the necessity of identifying multi-cycle false paths as well as show the effectiveness of our method.

The rest of this chapter is organized as follows. A review of related works is given in the following section. Preliminary ideas and assumptions are given then discussed. Next, multi-cycle RTL false path identification is explained and a case study about applying the necessary conditions on a benchmark circuit is discussed. Finally, it ends with some concluding remarks.

1. Review of Related Works

Efficient identification of path delay faults (PDF) has become an important topic as chip timing requirements become stricter and delay testing and analysis become more important. Generally, DFT techniques are used to reduce test cost. However, they introduce over-testing problems since DFT techniques can make untestable paths testable. Therefore, rapid and correct untestable path identification has become a requirement to reduce ATPG time as well as yield loss from over-testing when applying DFT to chip designs [33]. Several efficient methods have already been proposed for untestable path identification for combinational circuits at the gate-level [27] - [29]. Untestable PDF identification for sequential circuits have been addressed in [30], also at gate-level.

However, with today's complex chip designs, complete untestable path identification at the gate-level has become unfeasible, especially for sequential circuits [30]. Considering the great number of gate-level paths, [33] presented a way to identify untestable paths at RTL for sequential circuits. At RTL, the number of paths to be analyzed is greatly reduced and if the correspondence of RTL paths to gate-level paths can be established, a great reduction to ATPG time can be gained.

Chip designs can have multi-cycle paths which can be due to resource sharing, multiple clock domains for low-power designs and IP re-use as well. Thus, the identification of multi-cycle untestable paths has increasingly become an important problem for reducing ATPG time and test size [35]. [31, 32, 34, 35] have presented several techniques for multi-cycle path identification. More specifically, [35] studied the effects of multi-cycle false path removal on testability while [32] aimed to derive the valid clock period of a circuit using gate and segment delay information. [34] specifically addressed the problem of reducing test generation and

fault grading time. However, all of these approaches for multi-cycle untestable path identification only considered gate-level circuits.

In this chapter, we present sufficient conditions for rapid and efficient identification of non-robust untestable multi-cycle paths at RTL. Although, untestable path identification at RTL was first introduced in [33], the authors only considered single-cycle RTL untestable paths and ignored the possibility that some paths may not be untestable if tested at more than one clock cycle. Thus, for this research, we increase the quality of the identified false paths compared to [33]. For the rest of this chapter, non-robust untestable paths will be referred to as false paths as in [33].

2. Preliminaries

2.1 RTL Circuits

The method presented in this work uses RTL information to quickly identify false paths. The RTL circuits under consideration are represented as a structural design consisting of a controller (finite-state machine), a datapath (made up of multiplexers (MUXs), registers, and other RTL combinational modules), and the corresponding RTL signal lines (control and status signals) connecting them as shown in Figure 5.1.

Here, we only consider designs with a single clock domain, and RTL paths in datapaths. We assume that, for controllers, there exists no multi-cycle paths, and state transitions are completely specified for each possible state and input vector pair. We also assume that delay information for the modules and segments are not available at RTL.

2.2 RTL Path

An RTL path is a path which starts at a primary input (PI) or a register (start register Rs) and ends at a primary output (PO) or register (end register Re). Furthermore, it must only pass through combinational modules and can represent one or a bundle of single-bit paths. For example, a path from register R1 passing through multiplexer M1, the ALU, multiplexer M2 and ending at reg-

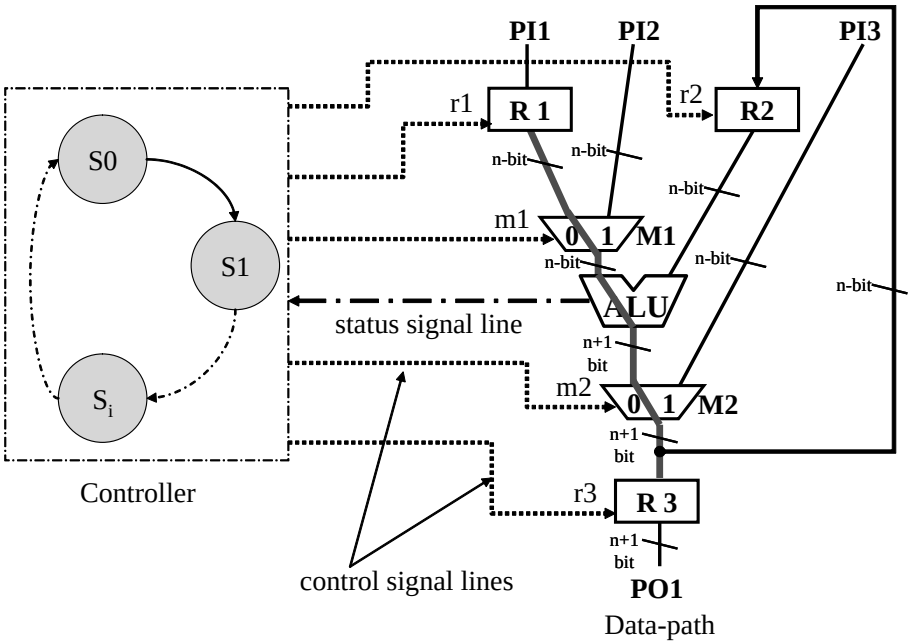


Figure 5.1. Example RTL circuit

ister R3 in Figure 5.1 is considered an RTL path. Since each RTL path can represent a bundle of gate-level paths, after logic synthesis, these RTL paths are transformed into a large number of gate-level paths. However, the final gate-level circuit configuration greatly depends on the type of logic synthesis used. In order to achieve RTL path to gate-level path correspondence, we restrict logic synthesis to a *module interface preserving-logic synthesis* (MIP-LS) as defined in [33]. In MIP-LS, each RTL module is transformed into individual gate-level net-lists. Since optimizations are only performed within each module and each RTL signal line connecting the modules are split into 1-bit signal lines, the connection configuration in-between the modules are guaranteed to be preserved and propagated to the gate-level synthesized circuit.

2.3 Single-cycle RTL False Path

At gate-level, false paths are paths which cannot be sensitized by any input vector.

Definition 1: RTL False Path

RTL false paths are RTL paths whose corresponding gate-level paths are all false for any logic synthesis [33].

□

Although RTL false paths were first defined in [33], the authors only presented sufficient conditions for single-cycle RTL false paths.

Theorem 1: An RTL path is single-cycle RTL false w.r.t. MIP-LS if it satisfies one of the following conditions at any time t:

1. No transition occurs at the starting point of the path at time t.
2. Any transition at the starting point at time t is not propagated along the path to the end point within one clock cycle.
3. No value is captured at the ending point at time *t+1*.
4. No value captured at the ending point of the path at time *t+1* is ever propagated to any PO.

□

Proof: The proof of Theorem 1 given in [33] is restated here.

Let D and D' be an RTL circuit and its corresponding gate-level circuit synthesized by an MIP-LS, respectively. Let $F = \{F_j | 1 \leq j \leq m\}$ be a set of flip-flops in D' corresponding to an m-bit register R in D and τ(R) denotes a mapping of R to F. Given a path p, let Rs and Re denote the starting and ending registers o p, respectively. Furthermore, let M_1,M_2,...,M_l be the RTL modules on p and suppose that p passes through the input ports M_{1in}, M_{2in},..., M_{lin} and output ports M_{1out}, M_{2out},..., M_{lout} of the RTL modules, respectively. Finally, let Q be a set of all gate-level paths between τ(Rs) and τ(Re) passing through M_{1in}, M_{1out}, M_{2in}, ..., M_{lin}, M_{lout} in order, and δ(p) denotes a mapping of p to Q.

If we let g denote a gate-level path in D', we must show that $\forall g \in \delta(p)$ are gate-level false for any MIP-LS. A combinational RTL module M_a in D and its synthesized logic block M'_a in D' are guaranteed to have completely the same

functionality if the outputs of M_a are completely specified for the input domain 2^n, where n is the number of inputs of M'_a. Thus, we can completely determine the capability of propagating transitions through M'_a by analyzing Ma at RTL. Otherwise, if the outputs of M_a are incompletely specified, M_a and M'_a are not guaranteed to have the same functionality since the functionality of M_a is a proper subset of that of M'_a. For an RTL path p, none of the conditions of Theorem 1 can be satisfied if there exists an incompletely specified M_a whose unknown/unspecified outputs affect the values captured by Rs, the propagation of values through p, the capturing of values in Re, or the propagated value from Re. Thus, we further assume that no incompletely specified RTL module that affects p, as above, exists.

Then, we let I_1, I_2, I_3, and I_4 be a set of the input sequences that satisfy conditions 1, 2, 3, and 4 of Theorem 1, respectively, and $I_1 \cup I_2 \cup I_3 \cup I_4$ contains all the possible input sequences.

Condition 1: Any input sequence $a \in I_1$ does not launch any transition at the starting register Rs on p. Let f be a flip-flop in D', since there is no incompletely specified combinational module that affects Rs, a does not launch any transition at $\forall f \in \tau(Rs)$, which is the starting point of $\forall g \in \delta(p)$.

Condition 2: Any input sequence $b \in I_2$ prevents the propagation of any transition launched at the starting point of p at some module M_c on p. Since there is no incompletely specified combinational module module affecting the off-inputs of p in the sub-circuit that drives the ending point of p, propagation of any transition on $\forall g \in \delta(p)$ is also prevented at the logic block corresponding to M_c.

Condition 3: Any input sequence $c \in I_3$ does not capture any transition at the end register Re on p. Let f be a flip-flop in D', since there is no incompletely specified combinational module that affects Re, c does not capture any transition at $\forall f \in \tau(Re)$, which is the ending point of $\forall g \in \delta(p)$.

Condition 4: Any input sequence $d \in I_4$ prevents the propagation of any transition captured at the ending point of p at some module M_d or some register Rd on every sequential path from Re to every primary output. Since there is no incompletely specified combinational module that affects propagation of the captured value at Re to the primary outputs, propagation of the value captured on $\forall f \in \tau(Re)$ is also prevented at the logic block corresponding to M_c or at

$\forall f \in \tau(Rd)$.

Thus, for any input sequence, no transition is launched at the starting point of $\forall g \in \delta(p)$, or no transition is propagated to the ending point along $\forall g \in \delta(p)$, or no value is captured at the ending point of $\forall g \in \delta(p)$, or no value valued at the ending point of $\forall g \in \delta(p)$ is propagated to any primary output. Therefore, $\forall g \in \delta(p)$ are gate-level false and p is single-cycle RTL false if one of the three conditions of Theorem 1 is satisfied for any input sequence at any time.

3. Multi-Cycle RTL False Path Identification

3.1 Multi-Cycle RTL Path

Multi-cycle RTL paths are essentially RTL paths which don't need to finish propagating a signal from a starting point to its end point within one clock cycle.

Definition 2: *k*-cycle RTL Path

A multi-cycle RTL path which has up to k clock cycles to propagate a transition from source to destination.

□

Assume that the RTL circuit shown in Figure 5.2 has a 3-cycle RTL path from R1 passing through multiplexer M1, the ALU, multiplexer M2 and ending at register R3. At time t, the control signal of R1, r1, is set to L and R1 loads a value from primary input PI1. At *t+1*, this transition is propagated through the path and reaches the ALU within 1 clock cycle. At *t+2*, the transition continues to propagate from the ALU until it reaches the input of R3. While not shown, the control signal of R3, r3, becomes L at *t+3* and the transition is captured.

3.2 *k*-Cycle RTL False Path

At gate-level, k-cycle false paths are paths which cannot be sensitized by any input vector within k clock cycles.

Definition 3: *k*-Cycle RTL False Path

A k-cycle RTL false path is an RTL path whose corresponding gate-level paths are all k-cycle false for any logic synthesis. □

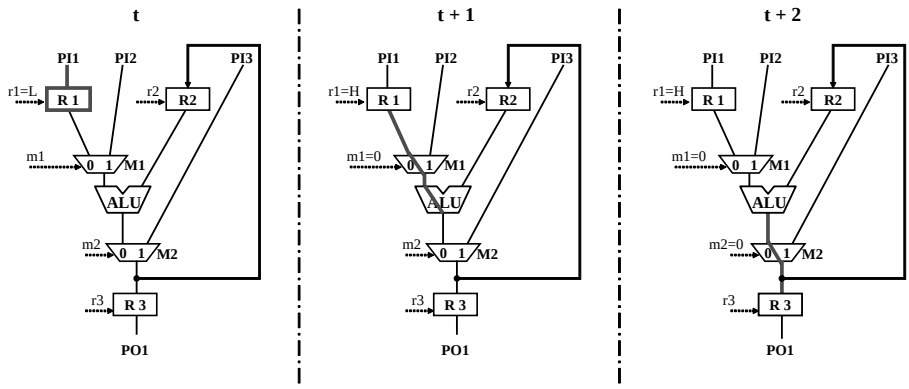


Figure 5.2. Example 3-cycle RTL path

Theorem 2: An RTL path is k-cycle false w.r.t MIP-LS if it satisfies one of the following conditions at any time t:

1. No transition ever occurs at the start register, Rs, or PI of the RTL path at time t.
2. Any transition at the starting point of the path at time t is never propagated along the path to the ending point within k clock cycles.
3. No value is captured at the ending point at time $t+k$.
4. No value captured at the ending register, Re, at time $t+k$ is ever propagated to any PO.

□

Proof: The proof of Theorem 2 is just an extension of the proof of Theorem 1. Specifically, we consider input sequences of length $k+1$ instead of just 2 for k-cycle paths.

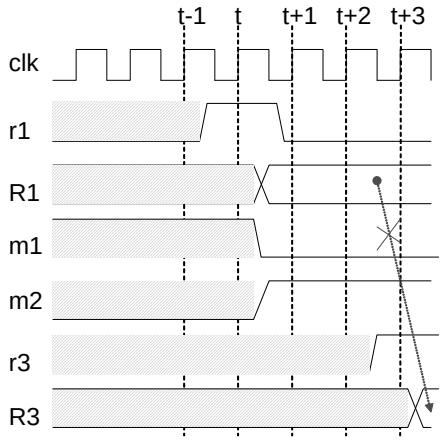
RTL paths which are k-cycle RTL false for $k \geq 2$ are multi-cycle RTL false paths. Furthermore, if a path is k-cycle RTL false for any value of k, it is only then that we can consider it as an RTL false path.

3.3 Control-Dependent k-Cycle RTL False Path

At RTL, we can gather information regarding state transitions and control signals (i.e. Load / Hold signals, MUX select signals, etc.) given a state and input vector pair. We can therefore derive sufficient conditions for identifying k-cycle RTL false paths using the control signal information. k-cycle false paths identified this way are called control-dependent k-cycle false (CDkF) paths. For this work, we will only consider paths starting at datapath registers (DR) or PIs and ending at datapath registers (DR) or POs.

An example of a CDkF RTL path is shown in Figure 5.3. Assuming that the timing diagram and control signal table of the path specified in Figure 5.2 is the only possible control sequence at any time t, the path starting from R1 to R3 passing through MUX M1, an ALU, and MUX M2 is CDkF at k=3 if we consider the timing diagram for the control signal sequences for R1 (r1), M1 (m1), M2 (m2), and R3 (r3) from time t to $t+3$ (the variables in parentheses are the respective control signals per module). The table shows the effective control signal at each clock edge, where L means a LOAD signal, H represents a HOLD signal, numbers represent the selected input of a multiplexer, and x's represent signal values which can be ignored at each time frame. At time t, r1=L and R1 loads a new value. This value is then propagated through M1 (m1 selects the on-path) within 3 clock cycles. Unfortunately, while R3 has a load signal at $t+3$, M2 never selects the on-path (m2 must be 0 in order to propagate the transition) from $t+1$ to $t+3$ and blocks the propagation of the signal from M1. Therefore, judging from the control signals from t to $t+3$, the path is 3-cycle CDkF.

The specific definition of a multi-cycle path can vary according to the design rules being followed by the circuit designers. For example, an RTL path will be classified differently depending on whether it belongs to a non-pipelined, pipelined or wave-pipelined circuit. For this work, we present solutions for a simple and strict single-transition circuit model and a less-strict, generalized model which are applicable to many circuit designs. Each transition propagation model and their corresponding sufficient conditions for CDkF RTL path identification are discussed in the following subsections. Note that these two models represent extreme cases and can be easily modified to reflect other circuit designs.



	t	t+1	t+2	t+3
r1	L	H	H	x
m1	x	0	0	0
m2	x	1	1	1
r3	x	x	x	L

Figure 5.3. Example 3-cycle CDkF path timing

3.4 Single-transition Circuit Model

In the single-transition model, the k-cycle paths of a circuit must exhibit the following characteristics at any time t:

1. When the start register, Rs, loads at t, it must hold the value from $t+1$ to $t+k$.

2. All multiplexers on the path must continuously select the on-path from $t+1$ to $t+k$.

3. The end register Re must load at $t+k$.

In short, given a k-cycle path, only the transition from Rs at time t is ever propagated to Re within k clock cycles. An example control sequence table of a 3-cycle path that can be a true path is shown in Figure 5.4(a) for the path given in Figure 5.2. For the transition to propagate from R1 to R3, it is imperative that all the multiplexers select the on-path from $t+1$ to $t+3$. In contrast, in Figure 5.4(b), R1 loads a new value at $t+2$ while in Figure 5.4(c), the multiplexer M1 selects an off-path at $t+2$. Both of these situations violate the requirements for single-transition propagation through the target path.

	t	t+1	t+2	t+3
r1	L	H	H	x
m1	x	0	0	0
m2	x	0	0	0
r3	x	x	x	L

(a)

	t	t+1	t+2	t+3
r1	L	H	L	x
m1	x	0	0	0
m2	x	0	0	0
r3	x	x	x	L

(b)

	t	t+1	t+2	t+3
r1	L	H	H	x
m1	x	0	1	0
m2	x	0	0	0
r3	x	x	x	L

(c)

Figure 5.4. (a) Possible 3-cycle testable path, (b)(c) 3-cycle CDkF w.r.t. the single-transition model

To derive sufficient conditions for false path identification under the single-transition model, we first introduce the concept of register *controllability* and *observability*. Register controllability denotes the capability of triggering a transition on a register while register observability denotes the capability of propagating the transition from a register to a PO.

Definition 4: Register Controllability for Single-transition Model

Let P be the set of paths ending at register R. R is uncontrollable at time t if it satisfies one of the conditions below for every path $q \in P$ for any $v \geq 1$:

1. No control sequence for the starting register, Rs, of q starting from $t-v$ to t is of the form $\{LH^{v-1}x\}$ or Rs is uncontrollable at time $t-v$.
2. R does not load at time t.
3. Let o_{Mi} be the control signal which selects the on-path for each multiplexer M_i in $q(1 \leq i \leq n$ and n is the number of multiplexers in $q)$. No control sequence for each M_i is of the form $\{x^v o_{Mi}\}$ from $t-v$ to t.

□

Definition 5: Register Observability for Single-transition Model

Let P be the set of paths starting at R. R is unobservable at time t if it satisfies one of the conditions below for every path $q \in P$ for any $v \geq 1$:

1. No control sequence for R starting from t to $t+v$ is of the form $\{LH^{v-1}x\}$.
2. The end register, Re, does not load at time $t+v$ or Re is unobservable at time $t+v$.
3. Let o_{Mi} be the control signal which selects the on-path for each multiplexer M_i in $q(1 \leq i \leq n$ and n is the number of multiplexers in $q)$. No control sequence for each M_i is of the form $\{x^v o_{Mi}\}$ from t to $t+v$.

□

Lemma 1: RTL CDkF Paths for Single-transition Model

An RTL path p is RTL CDkF with respect to the single-transition model and MIP-LS if one of the following 3 conditions is satisfied for state transition sequences at any time t to $t+k$:

1. Given the set of all possible control signal sequences of the start register Rs for k clock-cycles, (a) a control sequence of the form $\{LH^{k-1}x\}$ starting at time t cannot be found or (b) Rs is uncontrollable at t.

2. Let o_{Mi} be the control signal which selects the on-path for each multiplexer M_i in $p(1 \leq i \leq n$ and n is the number of multiplexers in $p)$. No control sequence for each M_i is of the form $\{xo^k_{Mi}\}$ from t to $t+k$.

3. (a) The transition is not captured by Re at $t+k$ or (b) Re is unobservable at $t+k$.

□

Proof: We show that Lemma 1 is properly included in Theorem 2 with respect to the single-transition model. For any input sequence of length k+1 with any initial state, the input sequence causes, at most, all the state transition sequences considered in Lemma 1. If any state transition sequence satisfies at least one of the conditions of Lemma 1, then the input sequence also satisfies at least one of the conditions of Lemma 1.

If condition 1(a) is satisfied for a state transition sequence, Rs holds a value. If Condition 1(b) is satisfied for the state transition sequence, the value in Rs is not changed even though it loads a new value. For both sub-conditions, no transition is ever launched at Rs. If condition 1 of Lemma 1 is satisfied for the state transition sequence, condition 1 of Theorem 2 is also satisfied for the state transition sequence.

If condition 2 of Lemma 1 is satisfied for the state transition sequence, M_i prevents the propagation of the transition and condition 2 of Theorem 2 is also satisfied.

If condition 3(a) is satisfied for the state transition sequence, no transition is captured by Re and condition 3 of Theorem 2 is satisfied. If Condition 3(b) is

satisfied for the state transition sequence, no captured value at Re is propagated to any primary output. Thus, for the state transition sequence, condition 4 of Theorem 2 is satisfied. One of the three conditions of Theorem 2 is satisfied for any input sequence if one of the three conditions of Lemma 1 is satisfied for any state transition sequence, and therefore, Lemma 1 is properly included in Theorem 2.

3.5 Generalized Transition Model

We further extend our methodology to consider the "possibility" of a transition starting from the Rs of a k-cycle RTL path to be propagated to its Re within k-clock cycles regardless of whether or not Rs continuously holds the value and/or all the MUXs continuously select the on-path. Here, as long as there is a possibility for a transition to be propagated within k-clock cycles, we cannot consider it to be CDkF. As an example, consider the control sequence table in Figure 5.5(b) for the RTL path in Figure 5.2, where a transition propagation path can be found from M1 to M2 because the transition could have passed through M1 at $t+1$ even though m1 selected the off-path during the subsequent cycles. The transition could also have passed through M2 at $t+3$ and finally captured by R3 at $t+3$. In contrast, M2 completely blocks the propagation of the transition from M1 to R3 at $t+2$ to $t+3$ in Figure 5.5(c). Note that successful transition propagation w.r.t. the single-transition model (Figure 5.5(a)) also means possible success w.r.t. the generalized model but not vice versa. Also, a path that violates the conditions for the generalized model automatically violates the single-transition model but not vice versa.

Definition 6: Register Controllability for Generalized Model

Let P be the set of paths ending at register R. R is uncontrollable at time t if it satisfies one of the conditions below for every path $q \in P$ for any $v \geq 1$:

1. No control sequence for the starting register, Rs, of q starting from $t-v$ to t is of the form $\{Lx^v\}$ or Rs is un-controllable at time $t-v$.
2. R does not load at time t.
3. Let o_{Mi} be the control signal which selects the on-path for each multiplexer M_i in $q(1 \leq i \leq n$ and n is the number of multiplexers in $q)$. No control

	t	t+1	t+2	t+3
r1	L	H	H	x
m1	x	0	0	0
m2	x	0	0	0
r3	x	x	x	L

(a)

	t	t+1	t+2	t+3
r1	L	x	x	x
m1	x	0	1	1
m2	x	1	1	0
r3	x	x	x	L

(b)

	t	t+1	t+2	t+3
r1	L	x	x	x
m1	x	1	0	1
m2	x	0	1	1
r3	x	x	x	L

(c)

Figure 5.5. (a)(b) Possible 3-cycle testable path, (c) 3-cycle CDkF w.r.t. the generalized transition model

sequence for each M_i is of the form $\{x^{W_i} o_{Mi} x^{v-W_i}\}$ from $t-v$ to t, where $1 \leq W_i \leq v, W_{i-1} \leq W_i$ and $2 \leq i \leq n$.

□

Definition 7: Register Observability for Generalized Model

Let P be the set of paths starting at register R. R is unobservable at time t if it satisfies one of the conditions below for every path $q \in P$ for any $v \geq 1$:

1. No control sequence for R starting from t to $t+v$ is of the form $\{Lx^v\}$.
2. The end register, Re, of q does not load at time $t+v$ or Re is unobservable at time $t+v$.
3. Let o_{Mi} be the control signal which selects the on-path for each multiplexer M_i in $q(1 \leq i \leq n$ and n is the number of multiplexers in $q)$. No control sequence for each M_i is of the form $\{x_{W_i} o_{Mi} x^{v-W_i}\}$ from t to $t+v$, where $1 \leq W_i \leq v, W_{i-1} \leq W_i$ and $2 \leq i \leq n$.

□

Lemma 2: RTL CDkF paths for Generalized Model

An RTL path p is RTL CDkF with respect to the generalized model and MIP-LS if one of the following 3 conditions is satisfied for state transition sequences at any time t to $t+k$:

1. Given the set of all possible control signal sequences of the start register, Rs, for k clock-cycles, (a) a control sequence of the form $\{Lx^k\}$ cannot be found or (b) Rs is uncontrollable at t.
2. Let o_{Mi} be the control signal which selects the on-path for each multiplexer M_i in $p(1 \leq i \leq n$ and n is the number of multiplexers in $p)$. No control sequence for each M_i is of the form $\{x^{W_i} o_{Mi} x^{k-W_i}\}$ from t to $t+k$, where $1 \leq W_i \leq k, W_{i-1} \leq W_i$ and $2 \leq i \leq n$.
3. (a) The transition is not captured by Re at $t+k$ or (b) Re is unobservable at $t+k$.

□

Proof: Since we can show that Lemma 2 is properly included in Theorem 2 with respect to the generalized transition model by following the same steps made for the proof of Lemma 1, the details are omitted in this work.

4. Case Study

In this section, we present a case study using a simple benchmark circuit to illustrate how the derived necessary conditions can be used to identify CDkF paths as well show the importance of properly classifying these paths according to the value of k.

4.1 Problem Definition

In this subsection, we formally define the RTL CDkF path identification problem P_{CDKF}:

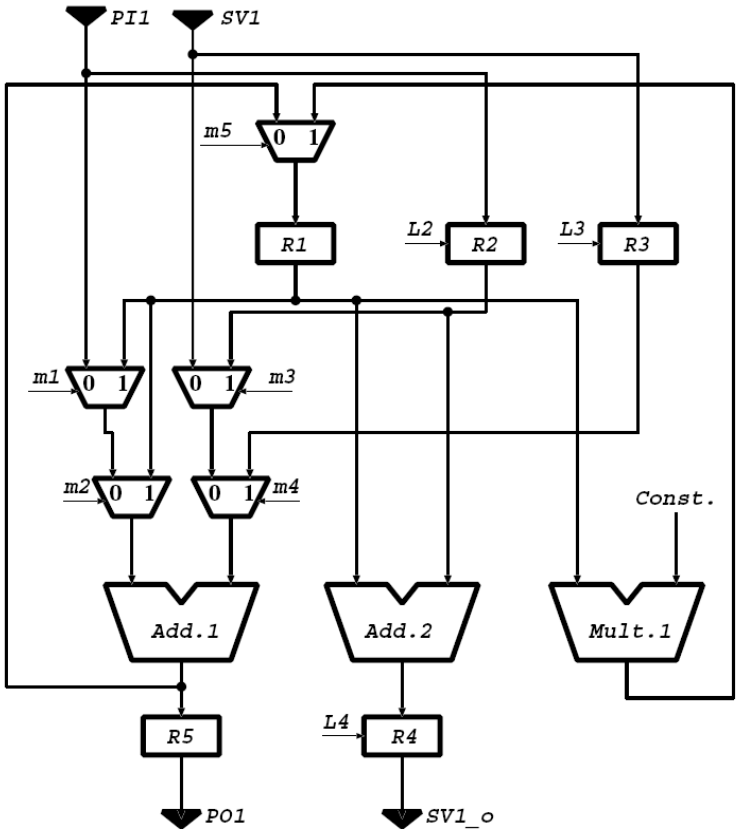
Given an RTL circuit C_{RTL}, its corresponding transition propagation model and k, determine the RTL CDkF paths in the datapath of C_{RTL}.

□

4.2 Analysis of the LWF Benchmark Circuit

As a case study, we have opted to apply the conditions for RTL CDkF path identification for both transition propagation models on a Lattice Wave Filter (LWF) [33] benchmark circuit in Figure 5.6(a). The characteristics (bit width, number of PIs, POs, Registers, States and RTL paths) of the LWF circuit are shown in Table 5.1. Note that for this case study, we ignored the presence of the RESET input as well as the reset function of the circuit. Thus, the state transition of the circuit is simplified into the various sequences formed by the four possible circuit states, as shown in Figure 5.6(b).

Applying the sufficient conditions for RTL CDkF identification w.r.t. the single-transition model, we get the results shown on Table 5.7(a). In Table 5.7(a), the classification of each of the 19 RTL paths is shown with respect to the value



(a)

Present state	Next state	Control Signals							
		m1	m2	m3	m4	m5	L2	L3	L4
s0	s1	0	0	0	0	0	L	L	H
s1	s2	0	0	0	0	1	H	H	H
s2	s3	0	1	0	1	0	L	H	L
s3	s0	1	0	1	0	0	H	H	H

(b)

Figure 5.6. (a)LWF block diagram and (b)state transition table

Table 5.1. LWF circuit characteristics

$Bitwidth$	$\sharp PIs$	$\sharp POs$	$\sharp REGs$	$\sharp States$	$\sharp RTLpaths$
8	3	2	6	4	19

of k, where those marked **F** are CDkF at that value of k. An asterisk means there is no definite conclusion, and thus the path might be testable at that value of k. As we have expected, there are paths whose testability will vary according to the value of k. For example, path $\sharp$10 is single cycle untestable, but it might be testable at k=2. If we only use the method in [33], this possibility would have been ignored and the path would be immediately considered false. Paths $\sharp$3 and $\sharp$6, on the other hand, might be testable at both k=1, 2. It is reasonable to conclude that the paths must be tested at both cycle counts unless the designer specifies other wise. Another interesting result is that for paths $\sharp$1, $\sharp$4, $\sharp$7 which show that these paths are testable up to k=4. Finally, paths $\sharp$8, $\sharp$9, $\sharp$11 and $\sharp$15 were all false for all values of k. If the circuit designer states the maximum value of k to be 4, then we can consider these paths as RTL false w.r.t. the design constraints of the circuit.

The CDkF identification results w.r.t. the generalized model are shown in Table 5.7(b). As expected, the number of identified false paths is much less compared to the single-transition model. It must be noted that the single-transition model can be considered as a special case of the generalized model, and thus paths that are not considered false in Table 5.7(a) are also classified in the same way in Table 5.7(b). Note that while paths $\sharp$8 and $\sharp$9 are considered untestable for all the allowed values of k under the single-transition model, the generalized model considers the possibility that they are testable at values of $k > 1$. Furthermore, paths that are false in Table 5.7(a) for all k, such as paths $\sharp$11 and $\sharp$15, can possibly be testable for all k under the generalized model.

Comparing the results using the method in [33] for single-cycle RTL paths shown in Table 5.7(c) to Table 5.7(a), we can see that the results are identical for k=1. Furthermore, the possibility of path $\sharp$10 being testable at k=2 under the single-transition model and testable at k=4 under the generalized model is completely ignored in [33]. This shows that an increase in the resolution of

Path #	k			
	1	2	3	4
1	*	*	*	*
2	*	F	F	F
3	*	*	F	F
4	*	*	*	*
5	*	F	F	F
6	*	*	F	F
7	*	*	*	*
8	F	F	F	F
9	F	F	F	F
10	F	*	F	F
11	F	F	F	F
12	*	F	F	F
13	*	F	F	F
14	*	F	F	F
15	F	F	F	F
16	*	F	F	F
17	*	F	F	F
18	*	F	F	F
19	*	F	F	F

(a)

Path #	k			
	1	2	3	4
1	*	*	*	*
2	*	*	*	*
3	*	*	*	*
4	*	*	*	*
5	*	*	*	*
6	*	*	*	*
7	*	*	*	*
8	F	*	*	*
9	F	*	*	*
10	F	*	F	*
11	*	*	*	*
12	*	*	*	*
13	*	*	*	*
14	*	*	*	*
15	*	*	*	*
16	*	*	*	*
17	*	*	*	*
18	*	*	*	*
19	*	*	*	*

(b)

Path #	k
	1
1	*
2	*
3	*
4	*
5	*
6	*
7	*
8	F
9	F
10	F
11	F
12	*
13	*
14	*
15	F
16	*
17	*
18	*
19	*

(c)

Figure 5.7. RTL CDkF path identification results for (a)single-transition model, (b)generalized model, (c) method in [33]

path identification, and therefore, an increase in test quality can be achieved if we consider multi-cycle RTL false paths. Note that the results would also vary depending on the transition model used, and both models that are presented in this work represent polar extreme cases.

5. Concluding Remarks

In this chapter, we have introduced the concept of multi-cycle false paths at RTL and derived delay-independent sufficient conditions for multi-cycle RTL false path identification for two transition propagation models. Furthermore, these models

and conditions can be easily modified to apply to various circuit designs. The case study has shown that paths can be classified as false or not depending on the transition propagation model used as well as the number of clock cycles allowed per test. Thus, there is no doubt that if multi-cycle paths exist in a circuit, they should be classified and tested appropriately. As was shown in [33], rapidly identifying RTL false paths can increase test efficiency and quality and reduce ATPG time and over-testing. It can be concluded that extending the path analysis to cover multi-cycle RTL paths can further increase test quality and efficiency.

Chapter 6

Conclusion and Future Work

1. Summary of the Thesis

VLSI testing has always played an important part in delivering fully functional and reliable chips. As VLSI manufacturing processes improve and the level of integration increases, the increasing chip design complexity becomes a barrier to thorough test and chip verification. Furthermore, the current trend is towards low-power, low-temperature designs for increasingly mobile applications, especially for System-on-Chips. More specifically, problems such as increasing test time and cost, power and temperature control during test must be handled while trying to maintain, or if possible, improve test quality.

This thesis presents several works which presents solutions that cover the problems mentioned above. More specifically, it targets modern SoC designs which have several re-used embedded IP cores running at different clock domains. The solutions range from introducing new Design-for-Testability techniques targeting individual cores as well as the entire design at chip level, to introducing ways to reduce test costs at higher abstraction levels (i.e. RTL).

Chapter 1 introduces the concept of Core-based design, Design-for-Testability and gives an overview of scan-based design as well as the TAM/wrapper DFT methodology for core-based testing. The TAM and wrapper combination forms the basis for most of the techniques introduced in this thesis. Chapter 2 discusses a wrapper design methodology targeting cores with multiple clock domains. More specifically, the concept of separating the cores into smaller *virtual cores* and

controlling their shift frequency via bandwidth matching techniques to control power dissipation is discussed. The method increases flexibility during scheduling, ensuring more efficient schedules under a power constraint.

Chapter 3 discusses the limitations of power constrained test, the need for thermal-awareness, and a thermal-safe TAM/Wrapper design methodology targeting SoCs with flexible TAM configurations. A complete TAM/wrapper design methodology, starting from wrapper and TAM design to test schedule optimization is presented. The technique basically checks the overall temperature of the SoC during test via thermal simulation, and reconfigures the test accordingly. Furthermore, the usage of cycle-accurate power profiles enabled us to produce more realistic results. Also, by developing a simple thermal model to represent temperature, we are able to avoid unnecessary thermal simulations.

Chapter 4 discusses ways to extend and improve the methodology presented in Chapter 3. More specifically, the concept of test partitioning and interleaving is introduced. Furthermore, we also made use of bandwidth matching techniques to throttle shift frequency and reduce power. To illustrate the effectiveness of the proposed techniques, the methodology was applied to SoCs with fixed-TAM architectures. Nevertheless, even with the decreased TAM design flexibility, we were able to show that the new techniques can give better results compared to those in Chapter 3.

Chapter 5 discusses the problem of multi-cycle false path identification and introduces a way to rapidly identify them. More specifically, the idea of identifying false paths at RTL is introduced. At RTL, there is significantly less paths to examine compared to gate-level circuits. A method to identify a subset of multi-cycle paths by control-signal analysis is given, and their application is illustrated via a case study.

2. Future Work

In this thesis, the problems facing SoC test engineers, mainly high power dissipation, overheating, and long test generation times, have been addressed separately. In truth, all of the above problems must taken into account at the same time during test planning. For example, it was shown that controlling power dissipation

does not always ensure the prevention of overheating during test. Conversely, by only looking at temperature, we ignore possible problems such as IR-drops which can invalidate test results. Thus, there is a need to develop more complete and integrated test methodologies that take into account more parameters and constraints.

Furthermore, there is strong evidence of SoCs moving towards 3-dimensional or "stacked" designs. Obviously, while these designs are more compact, they intensify the problems related to power dissipation and temperature. There is then, a need to design new DFT methodologies for these new generation of SoCs.

From the perspective of false-path identification, we can only conclude that as designs become for complex, there will be an increased need to do pre-processing at higher abstraction levels. Of course, techniques to bridge these levels, such as RTL path to gate-level path mapping techniques, would be needed to make these data useful.

Acknowledgements

First of all, I would like to thank God for giving me the opportunity to travel to Japan and gain valuable knowledge here in NAIST. I would like to express my deepest gratitude to Prof. Hideo Fujiwara for accepting me into his laboratory, for being patient with me, and for imparting me wisdom that helped me during my stay here. To Prof. Yasuhiko Nakashima and Assoc. Prof. Michiko Inoue, thank you for your patience, guidance and unfaltering support. To Prof. Satoshi Ohtake, thank for your great patience, your steady hand, and for infecting me with your positive attitude that always made things brighter and more enjoyable. To Prof. Tomokazu Yoneda, who has been with me thru all my troubles and triumphs during the course of my studies, thank you for always being there to give me advice and listen to what I have to say, many thanks for being supportive, and for being a teacher and a friend at the same time. To all the members of the Computer Design and Test Laboratory of the Nara Institute of Science and Technology, thank you for the valuable support and assistance you have given me during the course of this research, for welcoming me as both colleague and friend. I will never forget all the good (and bad) times we have shared.

I would also like to thank Prof. Danella Zhao of The Center For Advanced Computer Studies at the University of Louisiana at Lafayette, Prof. Krishnendu Chakrabarty of Duke University, Prof. Eric Larsson of Linköping University, Sweden, Prof. Tomoo Inoue and Prof. Yuki Yoshikawa of the Hiroshima City University for their comments, advice and for supplying valuable data for reference during my research.

This work was supported in part by Japan Society for the Promotion of Science (JSPS) under Grants-in-Aid for Scientific Research B(No.15300018), JSPS under Grants-in-Aid for Young Scientists (B)(No. 18700046), the grant of JSPS Research Fellowship (No. S06089), Semiconductor Technology Academic Research Center (STARC) under the Research Project and JSPS under Grants-in-Aid for Scientific Research B (No. 20300018). The work of K. Chakrabarty was supported in part by the US National Science Foundation under grant no. OISE-0403217. To all of these supporting organizations, thank you.

Finally, I would like to say thank you to my wife, Dr. Joanne Dy, for always being there. I am glad I found you.

References

[1] E. J. Marinissen, S. K. Goel and M. Lousberg, "Wrapper Design for Embedded Core Test," *Proc. of IEEE International Test Conference (ITC)*, pp. 911–920, 2000.

[2] S. K. Goel and E. J. Marinissen, "Effective and Efficient Test Architecture Design for SoCs," *Proc. of IEEE International Test Conference (ITC)*, pp. 529–538, 2002.

[3] V. Iyengar, K. Chakrabarty and E. J. Marinissen, "Test Wrapper and Test Access Mechanism Co-Optimization for System-on-Chip," *Journal of Electronic Testing: Theory and Application*, vol. 18, pp. 213–230, April 2002.

[4] V. Iyengar, K. Chakrabarty and E. J. Marinissen, "Test Access Mechanism Optimization, Test Scheduling, and Tester Data Volume Reduction for System-on-Chip," *IEEE Trans. on Computers*, vol. 52, no. 12, pp. 1619–1632, December 2003.

[5] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan and J. Rajski, "Logic BIST for Large Industrial Designs: Real Issues and Case Studies," *Proc. of IEEE International Test Conference (ITC)*, pp. 358–367, 1999.

[6] K. Hatayama, M. Nakao and Y. Sato, "At-Speed Built-in Test for Logic Circuits with Multiple Clocks," *Proc. of the 11th Asian Test Symposium (ATS)*, pp. 292–297, 2002.

[7] V. Jain and J. Waicukauski, "Scan Test Volume Reduction in Multi-Clocked Designs with Safe Capture Technique," *Proc. of IEEE International Test Conference (ITC)*, pp. 148–153, 2002.

[8] Q. Xu and N. Nicolici, "Wrapper Design for Testing IP Cores with Multiple Clock Domains," *Proc. of Design, Automation and Test in Europe* (DATE), pp. 416–421, 2004.

[9] Q. Xu, N. Nicolici and K. Chakrabarty, "Multi-Frequency Wrapper Design and Optimization for Embedded Cores Under Average Power Constraints,"

Proc. of ACM/IEEE Design Automation Conference (DAC), pp. 123–128, 2005.

[10] D. Zhao, and U. Chandran, "Design of A Time-Gated Multi-Frequency Wrapper Architecture for Modular SoC Testing," *Proc. of IEEE 15th North Atlantic Test Workshop (NATW)*, May 2006.

[11] T. Yoneda, K. Masuda and H. Fujiwara, "Power-Constrained Test Scheduling for Multi-Clock Domain SoCs," *Proc. of Design, Automation, and Test in Europe (DATE)*, pp. 297–302, 2006.

[12] Y. Xia, M. Chrzanowska-Jeske, B. Wang and M. Jeske, "Using a Distributed Rectangle Bin-Packing Approach for Core-based SoC Test Scheduling with Power Constraints," *Proc. of the International Conference on Computer-Aided Design (ICCAD)*, pp. 100–105, 2003.

[13] Y. Huang et al., "Optimal Core Wrapper Width Selection and SOC Test Scheduling Based on 3-D Bin Packing Algorithm," *Proc. of IEEE International Test Conference (ITC)*, pp. 74–82, 2002.

[14] Y. Zorian, E.J. Marinissen and S. Dey, "Testing Embedded Core Based System Chips," *Proc. of IEEE International Test Conference (ITC)*, pp. 130-143, 1998.

[15] E. Larsson, Introduction to Advanced System-on-Chip Test Design and Optimization, *Springer Dordrecht*, ISBN 1-4020-3207-2, 2005.

[16] M. Abramovici, M.A. Breuer and A.D. Friedman, Digital Systems Testing and Testable Design, *IEEE Press*, ISBN 0-7803-1062-4, 1990.

[17] S. Samii, E. Larsson, K. Chakrabarty and Z. Peng, "Cycle-Accurate Test Power Modeling and its Application to SoC Test Scheduling," *Proc. of IEEE International Test Conference (ITC)*, pp. 1–10, 2006.

[18] E.J. Marinissen, V. Iyengar and K. Chakrabarty, "A Set of Benchmarks for Modular Testing of SoCs," *Proc. of IEEE International Test Conference (ITC)*, pp. 519–528, 2002.

[19] P. Rosinger, B. Al-Hashimi and K. Chakrabarty, "Rapid Generation of Thermal-safe Test Schedules," *Proc. of Design, Automation, and Test in Europe (DATE)*, pp. 840–845, 2005.

[20] C. Liu, K. Veeraraghavan and V. Iyengar, "Thermal-Aware Test Scheduling and Hot Spot Temperature Minimization for Core-Based Systems," *Proc. of 20th IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT'05)*, pp. 552–56-, 2005.

[21] Z. He et al., "Thermal-Aware SoC Test Scheduling with Test Set Partitioning and Interleaving," *Proc. of 21st IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT'06)*, pp. 477–485, 2006.

[22] K. Skadron et al., "Temperature-aware Microarchitecture," *Proc. of the 30th Annual Int'l Symposium on Computer Architecture (ISCA'03)*, pp. 2–13, 2003.

[23] K. Skadron et al., "Temperature-aware Microarchitecture: Extended discussion and results," *Tech. Rep. CS-2003-08, University of Virginia, Department of Computer Science*, Apr. 2003.

[24] M.R. Stan et al., "Hotspot: A dynamic compact thermal model at the processor-architecture level," *Microelectronics Journal: Circuits and Systems 34*, pp. 1153–1165, Dec. 2003.

[25] S. Lee et al., "Constriction/Spreading Resistance Model for Electronics Packaging," *Proc. of ASME/JSME Thermal Engineering Conference*, pp. 199–206, Mar. 1995.

[26] A. Krstic and K.T. Cheng, "Delay Fault Testing for VLSI Circuits," *Kluwer Academic Publishers*, 1998.

[27] K.T. Cheng and H.C. Chen, "Classification and Identification of Nonrobust Untestable Path Delay Faults," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 8, pp. 854-853, Aug. 1996.

[28] S. Kajihara, K. Kinoshita, I. Pomeranz, and S.M. Reddy, "A Method for Identifying Robust Dependent and Functionally Unsensitizable Paths," *Proc. International Conference on VLSI Design*, pp. 82-87, 1997.

[29] S.M. Reddy, S. Kajihara, and I. Pomeranz, "An Efficient Method to Identify Untestable Path Delay Faults," *Proc. of 10th Asian Test Symposium*, pp. 233-238, Nov. 2001.

[30] A. Krstic, S.T. Chakradhar, and K.T. Cheng, "Testable Path Delay Fault Cover for Sequential Circuits," *Proc. of European Design Automation Conference*, pp. 220-226, Sep. 1996.

[31] W.C. Lai, A. Krstic, and K.T. Cheng, "On Testing The Path Delay Faults of a Microprocessor Using Its Instruction Set," *Proc. of 18th VLSI Test Symposium*, pp. 15-20, May 2000.

[32] K. Yang and K.T. Cheng, "Efficient Identification of Multi-Cycle False Path," *Proc. of Conference on Asia South Pacific Design Automation*, pp. 360-365, 2006.

[33] Y. Yoshikawa, S. Ohtake and H. Fujiwara, "False Path Identification using RTL Information and Its Application to Over-testing Reduction for Delay Faults," *Proc. of 16th Asian Test Symposium*, pp. 65-68, Oct. 2007.

[34] M. Syal, M.S. Hsiao, S. Natarajan and S. Chakravarty, "Untestable Multi-Cycle Path Delay Faults in Industrial Designs," *Proc. of 14th Asian Test Symposium*, pp. 194-201, 2005.

[35] P. Kalla and M. Ciesielski, "Testability of Sequential Circuits with Multi-Cycle False Paths," *Proc. of 15th VLSI Test Symposium*, pp. 322-328, 1997.

[36] A. Ajami et al., "Analysis of Non-Uniform Temperature-Dependent Interconnect Performance in High Performance ICs," *Proc. of Design Automation Conference*, pp. 567–572, 2001.